NXP Semiconductors
Reference Manual

Document identifier: LA9310RM
Rev. 1, 23/01/2025

LA9310 Reference Manual

Contents

Chapter 1
Overview

1.1 Introduction

The Layerscape LA9310 is a small, programmable, baseband processor targeted at a sub-6GHz 5G such as network listening modules, repeaters, repeater controllers, and 1T1R or 2T2R radio units. It can also be used in proprietary wireless applications such as software defined radio. The LA9310 features an ARM M4 core operating at 307.2 MHz along with an innovative programmable signal processing accelerator, and integrated high-speed analog/digital data converters. LA9310 includes on-chip RAM for packet buffering and operates without any external DRAM. The device can boot from host processor via PCIe. The LA9310 is built for ultra-low power consumption and is offered in a small package for compact form-factor applications.

1.2 Block diagram

The figure below shows the major functional units within LA9310.

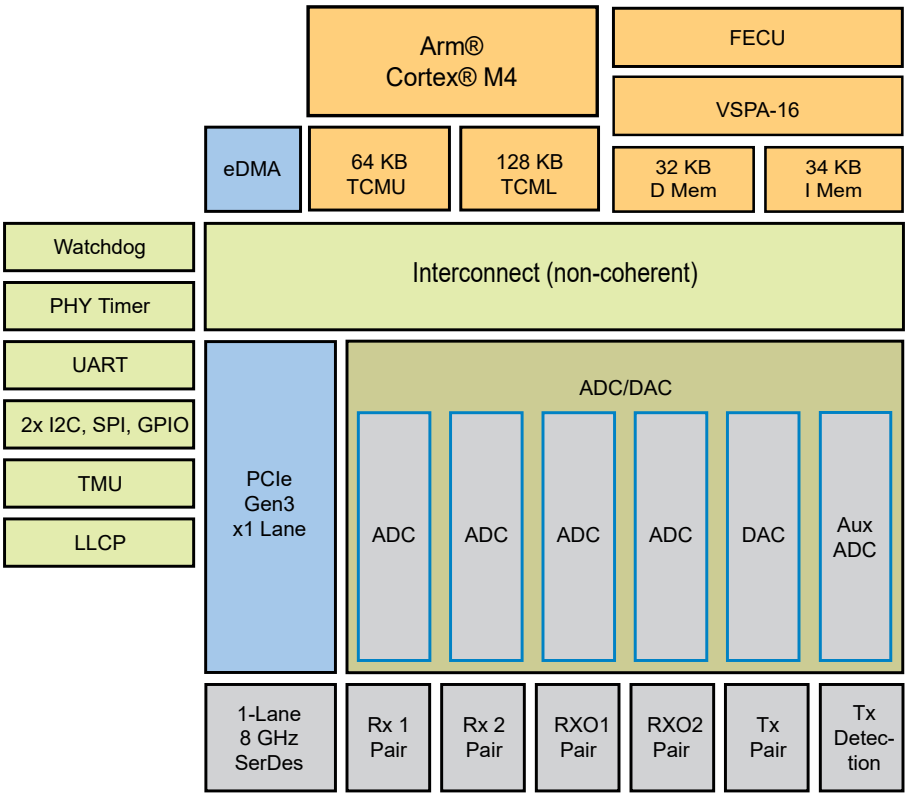


Figure 1. LA9310 Block Diagram

1.3 Features summary

This chip includes the following functions and features:

- Cortex M4 Complex
  - One Cortex M4 CPU running at 307.2 MHz
  - 128 KByte Tightly Coupled ICode and DCode memory (TCML)
  - 64 KByte Tightly Coupled System Data Memory (TCMU)

- VSPA Complex
  - VSPA-16AU running at 614.4 MHz
  - Forward Error Correction Unit
  - 34 KB (32KB for VCPU + 2KB for IPPU) Program Memory
  - 32 KB Data Memory
- AXI i/q Bridge (AXIQ)
- PHY Timer
- NIC 301 Interconnect running at 307.2 MHz (sub-partitions of the interconnect run at 614.4 MHz)
- ADC/DAC Subsystem
  - Four ADC with differential input for complex baseband i/q paths
  - One DAC with differential output for complex baseband i/q paths, two differential pairs
  - One ADC with single input for Tx detection, single input signal
  - ADC and DAC can operate at clock input rate or divide by 2
- Single lane PCIe Gen3 SerDes
- General purpose single lane Serial peripheral interface (SPI) (LVCMOS signaling) with four Chip Selects
- RFIC Interface using LLCP (Lightweight LVDS Communication Protocol) and RSSI data
- I^2C controller
- UART

1.4 Modules on LA9310

The chip includes the following modules.

1.4.1 ARM® Cortex®-M4

The chip has a single Cortex-M4 processor running at 307.2 MHz.

The processor implements these features.

- A Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low latency interrupt processing.
- Multiple high-performance bus interfaces.
- The processor has the following external interfaces: Multiple memory and device bus interfaces. Debug port interface.
- 128 KB Tightly Coupled Code Memory (TCML for “lower” TCM address range) and a 64 KB Tightly Coupled Data Memory (TCMU for “upper” TCM address range). Both of these memories can be accessed by external masters connected to the NIC301.
- A bus dedicated to instruction fetches of the Code memory, and a bus dedicated to Load/Store accesses to the Code memory. These buses are muxed together to access the code memory array. The M4 also has a System Bus for accesses to all memories .

1.4.2 CoreLink™ Network Interconnect NIC-301

CoreLink™ Network Interconnect NIC-301 is the primary interconnect on the chip. The various IPs and external I/Os communicate through a complete high performance, optimized AMBA-compliant network infrastructure (NIC-301).

Features contain eDMA, Cortex M4, PCIe and test port to the network.

- Contains a 614.4 MHz switch that connects the VSPA complex to the network.

- VSPA communicates directly through the NIC to the AXIQ in the 614.4 MHz domain.
- All CCSR registers are accessible through APB targets of the NIC.
- The LLCP master updates from VSPA and communicate directly through the NIC in the 614.4 MHz domain.
- The NIC contains elastic buffering to mitigate stalls because of arbitration.

1.4.3 Universal asynchronous receiver/transmitter (UART)

The UART includes these distinctive features:

- Programming model compatible with original PC16450 UART and PC16550D (improved version of PC16450 that also operates in FIFO mode)
- PC16450 register reset values
- Configurable FIFO mode for both transmitter and receiver, providing 64-byte FIFOs
- Serial data encapsulation and decapsulation with standard asynchronous communication bits (START, STOP, and parity)
- Maskable transmit, receive, line status, and modem status interrupts
- Software-programmable baud generators that divide the platform clock/2 by 1 to (2^{16} - 1) and generate a 16x clock for the transmitter and receiver engines
- Software-selectable serial interface data format (data length, parity, 1/1.5/2 STOP bit, baud rate)
- Line and modem status registers
- Line-break detection and generation
- Internal diagnostic support, local loopback, and break functions
- Prioritized interrupt reporting
- Overrun, parity, and framing error detection

1.4.4 Serial peripheral interface (SPI)

The module supports the following features:

- Full-duplex, three-wire synchronous transfers
- Master mode
- Data streaming operation in slave mode with continuous slave selection
- Buffered transmit operation using the transmit first in first out (TX FIFO) entries
- Support for 8/16-bit accesses to the PUSH TX FIFO Register Data Field
- Buffered receive operation using the receive FIFO (RX FIFO) entries
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues
- Visibility into TX and RX FIFOs for ease of debugging
- Programmable transfer attributes on a per-frame basis:
  - Two transfer attribute registers
  - Two extended transfer attribute registers
  - Serial clock (SCK) with programmable polarity and phase
  - Various programmable delays
  - Programmable serial frame size: 4 to 16 bits
    - SPI frames longer than 16 bits can be supported using the continuous selection format.

- Continuously held chip select capability
- Parity control

1.4.5 PCI Express

The chip supports a single PCI Express controller compatible with the *PCI Express Base Specification Revision 3.0*. Key features of each PCI Express controller include the following:

- End-point support only - no root complex
- The physical layer operates at 2.5, 5, or 8 Gbaud data rate
- x1 link width supported
- 32-bit addressing and 256-byte maximum payload size.
- Address translation unit maps inbound reads and outbound writes into 4 KB address translation region.
- Inbound INTx transactions
- Message Signaled Interrupt (MSI) transactions.
- Only one MSI address.
- Up to eight MSI data values (varying the low order 3 bits of the MSI data value)
- MSI Sources
  - M4 core writes
  - VSPA DMA - triggered by VSPA CPU
  - eDMA
  - MSIs can be generated by software using the MSI configuration data in the PCIe header, as written by the host.
  - MSI transactions can be generated by eDMA
- 2 inbound BARs-
  - BAR0 (256MB) for non-prefetchable access to SoC Config, Control, Status Register (CCSR) space
  - BAR1-2 for 64-bit addressability into 16MB prefetchable region for TCM and VSPA D-MEM
- 2 outbound windows-
  - One 1GB window for access to host memory
  - One 4KB window for mapping MSI transactions

1.4.6 Enhanced Direct Memory Access (eDMA)

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

1.4.7 Inter-integrated circuit (I2C)

The I^2C module has the following key features:

- Compatible with I^2C bus standard compliant with I^2C 2.0 standard with the exception that HS (high speed) mode is not supported
- Operating speeds
  - Up to 100 kbps in Standard Mode
  - Up to 400 kbps in Fast Mode
  - Operation at higher baud rates (up to a maximum of module clock/20) with reduced bus loading
  - Actual baud rate dependent on the SCL rise time (which depends on external pull-up resistor values and bus loading)
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- Maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF

1.4.8 Vector Signal Processing Acceleration (VSPA)

The VSPA includes these distinctive features:

- Multi-format floating point operations with transparent fixed point conversion

- 16 complex MAC operations per clock cycle
- One radix-2 DIT/DIF butterfly operation per clock cycle
- Single cycle special function operations, that is, 1/x, 1/sqrt(x), and exp(jx)
- Multi-channel DMA enables zero-overhead management of input/output data flows and control structures
- Low power Idle state with automatic wake-up
- Simple, low cycle count, register control from the host software driver
- Hardware and software breakpoints with internal register visibility

1.4.9 PHY Timer

The PHY-Timer consists of a free-running counter, and 23 comparators. Twenty two comparators can be used to generate a single trigger output per comparator. One comparator is expected to be used for triggered capture rather than compare match. The PHY timer and comparators provide a means for HRTM firmware to generate time intervals with precise relationships to PHY operations, as is needed for actions such as turning the transmitter off at the end of an outgoing PPDU, or enabling the receiver at the proper instant within SIFS so that an incoming PPDU that begins at the SIFS boundary will be detected. The PHY timer comparators compare against the value of a 32-bit, free-running PHY timer which is counting rising edges of the synchronized slower 80 MHz timer input.

- AMBA APB bus for access to configuration and status registers (32-bit access only)
- External Timer Clock
- 32-bit Free-running Counter (counter increments on rising edge of External Timer Clock Input)
- 23 Trigger Output Signals out of which one is used for triggered capture rather than compare match

1.4.10 Lightweight LVDS Communication Protocol (LLCP)

The chip includes a lightweight low-voltage differential signaling (LVDS) communication protocol block (LLCP). The interface supports 8b/10b encoding across a serial data and strobe pair.

1.5 Debug support

The debugging and performance monitoring capability enabled by the device hardware coexists within a debug ecosystem that offers a rich variety of tools at different levels of the hardware/software stack. Software development and debug tools from NXP CodeWarrior for VSPA core, as well as third-party vendor, Arm Development Studio for Arm Coretex-M4, provide a rich set of options for configuring, controlling, and analyzing debug and performance related events.

Chapter 2
Memory Map

2.1 Memory Map Overview

There are several address domains within the device, including the following:

- Logical, virtual, and physical (real) address spaces within the Arm Architecture core(s)
- Internal local address space
- Internal configuration, control, and status register (CCSR) address space, which is a special-purpose subset of the internal local address space
- External memory, I/O, and configuration address spaces of the serial RapidIO link
- External memory, I/O, and configuration address spaces of the PCI Express links

The MMU in the core handles translation of logical (effective) addresses, into virtual addresses, and ultimately to the physical addresses for the local address space. The local address map refers to the physical address space seen by the core as it accesses memory and I/O space.

2.2 System Memory Map

Target Destination	Start address	End address	Size	M4/TCM Allocation	NIC301 Allocation	PCIe BAR
Code Memory 512 MB	0x0000_0000	0x0000_1FFF	8KB	Code Bus - System Bus	ROM	
	0x0000_2000	0x1F7F_FFFF	8 KB		Default Slave (AEM)	
	0x1F80_0000	0x1F81_FFFF	128KB	Code Bus -> TCML	TCML	BAR 1 (128KB) (32b)
	0x1F82_0000	0x1FFF_FFFF	8MB-128KB	Code Bus -> System Bus	Default Slave (AEM)	
SRAM 512 MB	0x2000_0000	0x2000_FFFF	64KB	System Bus -> TCMU	TCMU	BAR2 (8MB) (64-bit)
	0x2001_0000	0x203F_FFFF	504 MB	System Bus	Default Slave (AEM)	
	0x2040_0000	0x207F_FFFF	4MB		VSPA	
	0x2080_0000	0x3FFF_FFFF	Reserved		Default Slave (AEM)	
Peripheral 512 MB	0x4000_0000	0x43FF_FFFF	64MB	System Bus	CCSR	BAR 0 (256MB) (32-bit)
	0x4400_0000	0x4400_FFFF	64KB		AXIQ	
	0x4401_0000	0x4401_FFFF	64KB		LLCP	
	0x4402_0000	0x47FF_FFFF	Reserved		Default Slave (AEM)	

Table continues on the next page...

Table continued from the previous page...

Target Destination	Start address	End address	Size	M4/TCM Allocation	NIC301 Allocation	PCIe BAR
	0x4800_0000	0x4FFF_FFFF	Reserved		Reserved	
	0x5000_0000	0x5FFE_FFFF	Reserved		Default Slave (AEM)	
	0x5FFF_0000	0x5FFF_FFFF	64KB		PCIe MSI	
External RAM 1GB	0x6000_0000	0x9FFF_FFFF	1GB	System Bus	Default Slave (AEM)	
External Device 1GB	0xA000_0000	0xDFFF_FFFF	1GB	System Bus	PCIe Host Memory	
Private Peripherals	0xE004_0000	0xE00F_FFFF	768KB	Private Peripheral Bus (External) (DCSR and Coresight)	Default Slave (AEM)	

2.3 CCSR Address Map

The following table lists the address offset of each block from the base of the 64 MB CCSR space.

Table 1. CCSR Block Base Address Map

Block Base Address (Hex)	Block	Size
0x100_0000	VSPA	64kB
0x101_0000	Reserved	64kB
0x102_0000	PHY Timer	64kB
0x103_0000	Reserved	64kB
0x104_0000	ADC/DAC Data Conversion System	64kB
.....	Reserved	
0x120_0000	Address Error Manager (AEM)	64kB
.....	Reserved	
0x136_0000	Platform Clock Generation Unit (CGU)	64kB
.....	Reserved	
0x137_0000	Clocking Registers	64kB
.....	Reserved	
0x1E0_0000	DCFG (Device Configuration)	64kB
.....	Reserved	64kB

Table continues on the next page...

Table 1. CCSR Block Base Address Map (continued)

Block Base Address (Hex)	Block	Size
0x1E3_0000	Power Management Unit	64kB
.....	Reserved	
0x1E6_0000	Reset	64kB
.....	Reserved	
0x1EA_0000	SerDes	64kB
.....	Reserved	
0x1F8_0000	TMU (Thermal Monitor Unit)	64kB
.....	Reserved	
0x1FC_0000	Message Unit	64kB
.....	Reserved	
0x1FF_0000	Pin Control	64kB
.....	Reserved	
0x200_0000	I2C 1	64kB
0x201_0000	I2C 2	64kB
.....	Reserved	
0x210_0000	SPI	64kB
0x211_0000	Reserved	
0x21C_0000	UART	64kB
.....	Reserved	
0x22C_0000	eDMA	64kB
.....	Reserved	
0x22E_0000	LLCP	64kB
.....	Reserved	
0x230_0000	GPIO	64kB
.....	Reserved	
0x23C_0000	Watchdog Timer	64kB
.....	Reserved	
0x340_0000	PCI-Express Controller PF0	64kB
0x34C_0000	PCI-Express Controller PF0 controls	64kB

2.4 Source ID Assignments

The table below shows the Source ID values that are associated with transactions from the given master. These values are generally strapped at the input to the NIC301 port connected to the master.

Table 2. Source ID assignments

Master	Source ID[3:0]
Cortex M4 Code Bus	0x0
Cortex M4 System Bus	0x1
VSPA	0x2
.....	.....
PCIe	0xD
eDMA	0xE

2.5 Endianness

All IPs use little endian format.

The Cortex M4 core will execute in Little Endian mode. The VSPA core supports Little Endian exclusively. All registers operate in little endian mode.

Chapter 3
Signal Descriptions

3.1 Signals Introduction

This chapter describes the external signals and is organized into the following sections:

- Overview of signals and cross-references for signals that serve multiple functions
- List of output signal states at reset

3.2 Signals Overview

Note that individual chapters of this document provide details for each signal, describing each signal's behavior when the signal is asserted or negated and when the signal is an input or an output.

The following tables provides a summary of the signals grouped by function. This table details the signal name, interface, alternate functions, number of signals, and whether the signal is an input, output, or bidirectional. The direction of the multiplexed signals applies for the primary signal function listed in the left-most column of the table for that row (and does not apply for the state of the reset configuration signals). Finally, the tables provide a pointer to the table where the signal function is described.

Table 3. LA9310 Signal Reference by Functional Block

Name	Description	Alternate Function(s)	Pin type
RFCTL (Used as PHY Timer Triggers)			
LNA1_EN	Low Noise Amp1 Enable	GPIO_11 cfg_test_port_dis	O
LNA2_EN	Low Noise Amp2 Enable	GPIO_10 cfg_pcie_gen	O
LNA3_EN	Low Noise Amp3 Enable	GPIO_09 cfg_boot_ho	O
PA_EN	Power Amp Enable	GPIO_12 cfg_rst_hndshk	O
TXRX0	Receive / Transmit Switch 0	GPIO_07 cfg_boot_src0	O
TXRX1	Receive / Transmit Switch 1	GPIO_08 cfg_boot_src1	O
Phy Timer			
PPS_IN	Pulse Per Second In	GPIO_04	IO
PPS_OUT	Pulse Per Second Out	GPIO_03 cfg_wrm_rstb	IO

Table continues on the next page...

Table 3. LA9310 Signal Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Pin type
LLCP			
LLCP_DIN_N	Data In -	-	I
LLCP_DIN_P	Data In +	-	I
LLCP_DOUT_N	Data Out -	-	O
LLCP_DOUT_P	Data Out +	-	O
LLCP_STIN_N	Strobe In -	-	I
LLCP_STIN_P	Strobe In +	-	I
LLCP_STOUT_N	Strobe Out -	-	O
LLCP_STOUT_P	Strobe Out +	-	O
SPI			
SPI_CLK	SPI Clock	-	O
SPI_CS0_B	Chip Select0	-	O
SPI_CS1_B	Chip Select 1	UART1_SIN	O
SPI_CS2_B	Chip Select 2	UART1_SOUT	O
SPI_CS3_B	Chip Select 3	GPIO_16	O
SPI_MISO	Master In / Slave Out	-	I
SPI_MOSI	Master Out / Slave In		O
I2C			
IIC1_SCL	Serial Clock	GPIO_02	IO
IIC1_SDA	Serial Data	GPIO_01	IO
GPIO			
GPIO_01	General Purpose Input/Output	**IIC1_SDA**	IO
GPIO_02	General Purpose Input/Output	**IIC1_SCL**	IO
GPIO_03	General Purpose Input/Output	**PPS_OUT** cfg_wrm_rstb	IO
GPIO_04	General Purpose Input/Output	**PPS_IN**	IO
GPIO_05	General Purpose Input/Output The PMUXCR0[GPIO_05] bit must be set to select the GPIO functionality after reset.	Reserved	IO

Table continues on the next page...

Table 3. LA9310 Signal Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Pin type
GPIO_06	General Purpose Input/Output The PMUXCR0[GPIO_06] bit must be set to select the GPIO functionality after reset.	Reserved	IO
GPIO_07	General Purpose Input/Output	**TXRX0** cfg_boot_src0	IO
GPIO_08	General Purpose Input/Output	**TXRX1** cfg_boot_src1	IO
GPIO_09	General Purpose Input/Output	**LNA3_EN** cfg_boot_ho	IO
GPIO_10	General Purpose Input/Output	**LNA2_EN** cfg_pcie_gen	IO
GPIO_11	General Purpose Input/Output	**LNA1_EN** cfg_test_port_dis	IO
GPIO_12	General Purpose Input/Output	**PA_EN** cfg_rst_hndshk	IO
GPIO_16	General Purpose Input/Output	**SPI_CS3_B**	IO
GPIO_17	General Purpose Input/Output		IO
GPIO_18	General Purpose Input/Output	-	IO
GPIO_19	General Purpose Input/Output	ASLEEP	IO
System Control			
ASLEEP	ASLEEP	**GPIO_19**	O
HRESET_B	Power On Reset	-	I
Clocking			
CLKOUT	Clock Output	-	O
DCS_CLK_N	DCS and System CLK -	-	I
DCS_CLK_P	DCS and System CLK +	-	I
PCI_CLK_N	Boot and PCI Ref Clock -	-	I
PCI_CLK_P	Boot and PCI Ref Clock +	-	I
DFT			
SCAN_MODE_B	Reserved	-	I

Table continues on the next page...

Table 3. LA9310 Signal Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Pin type
JTAG			
TCK	Test Clock	-	I
TDI	Test Data In	-	I
TDO	Test Data Out	-	O
TMS	Test Mode Select	-	I
TRST_B	Test Reset	-	I
SerDes 1			
SD1_IMP_CAL_RX	SerDes Receive Impedence Calibration	-	I
SD1_IMP_CAL_TX	SerDes Transmit Impedence Calibration	-	I
SD1_RX_N	SerDes Receive Data -	-	I
SD1_RX_P	SerDes Receive Data +	-	I
SD1_TX_N	SerDes Transmit Data -	-	O
SD1_TX_P	SerDes Transmit Data +	-	O
ADC/DAC			
IREF_TX	Analog Transmit Current Reference	-	I
RO0_I_N	Analog Observation (I-) Vin 0	-	I
RO0_I_P	Analog Observation (I+) Vin 0	-	I
RO0_Q_N	Analog Observation (Q-) Vin 0	-	I
RO0_Q_P	Analog Observation (Q+) Vin 0	-	I
RO1_I_N	Analog Observation (I-) Vin 1	-	I
RO1_I_P	Analog Observation (I+) Vin 1	-	I
RO1_Q_N	Analog Observation (Q-) Vin 1	-	I
RO1_Q_P	Analog Observation (Q+) Vin 1	-	I
RX0_I_N	Analog Receive (I-) Vin 0	-	I
RX0_I_P	Analog Receive (I+) Vin 0	-	I
RX0_Q_N	Analog Receive (Q-) Vin 0	-	I
RX0_Q_P	Analog Receive (Q+) Vin 0	-	I
RX1_I_N	Analog Receive (I-) Vin 1	-	I
RX1_I_P	Analog Receive (I+) Vin 1	-	I
RX1_Q_N	Analog Receive (Q-) Vin 1	-	I
RX1_Q_P	Analog Receive (Q+) Vin 1	-	I
TX_DET	Analog Auxiliary input (Tx Power Detect)	-	I

Table continues on the next page...

Table 3. LA9310 Signal Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Pin type
TX_I_N	Analog Transmit (I-) Iout	-	O
TX_I_P	Analog Transmit (I+) Iout	-	O
TX_Q_N	Analog Transmit (Q-) Iout	-	O
TX_Q_P	Analog Transmit (Q+) Iout	-	O
VREF_RX_N	Analog Rx Voltage Ref - (Left bump)	-	I
VREF_RX_P	Analog Rx Voltage Ref + (Left bump)	-	I
Analog Signals			
TD1_ANODE	Thermal diode anode	-	IO
TD1_CATHODE	Thermal diode cathode	-	IO
Power-On-Reset Configuration			
cfg_boot_ho	Boot hold-Off	**LNA3_EN** GPIO_09	I
cfg_boot_src0	Boot source	**TXRX0** GPIO_07	I
cfg_boot_src1	Boot source	**TXRX1** GPIO_08	I
cfg_pcie_gen	PCIe generation	**LNA2_EN** GPIO_10	I
cfg_rst_hndshk	Reset HW/SW handshake	**PA_EN** GPIO_12	I
cfg_test_port_dis	Test port enable	**LNA1_EN** GPIO_11	I
cfg_wrm_rstb	Warm reset	**PPS_OUT** GPIO_03	I
UART			
UART1_SIN	SIN	**SPI_CS1_B**	I
UART1_SOUT	SOUT	**SPI_CS2_B**	O

3.3 Output Signal States During Reset

When a system reset is initiated (HRESET_B sampled asserted by the chip), the chip aborts all current internal and external transactions and releases the bidirectional I/O signals to a high-impedance state. However, some signals get stable values at power-on reset.

While the the chip is in reset, it drives HRESET_B asserted and ignores most input signals (except for the reset configuration signals) and drives most of the output-only signals to an inactive state.

Chapter 4
Reset, Clocking, and Initialization

4.1 Reset, clocking, and initialization overview

This chapter describes reset, clocking, and initialization, including a definition of the reset configuration signals and the options they select. Note that other chapters in this book may describe specific aspects of initialization for individual blocks.

4.2 External Signal Description

The table below lists Reset block external signals.

Table 4. External Signal Description

Signal	I/O	Description	
HRESET_B	I	Power-on reset	
		State Meaning	Asserted-Put entire SoC into reset, clearing all state, except state which is controlled by TRST_B. Negated-Release SoC from reset
		Timing	Assertion- May be asserted asynchronously. Must be asserted for a minimum of 32 PCI_CLK cycles. Negation- May be negated asynchronously

4.2.1 External Clock Signals

The table below describes some of the external clock signals of the chip.

Note that some clock signals are specific to modules within the chip, and although some of their functionality is described here, they are defined in detail in their respective chapters.

Table 5. Clock External Signals-Detailed Signal Descriptions

Signal	I/O	Description	
PCI_CLK_P, PCI_CLK_N, DCS_CLK_P, DCS_CLK_N	I	PCI_CLK_P/N is the bypass clock during early POR until after pre-boot configuration. After pre-boot configuration, the PLL locks on the DCS_CLK_P/N.	
		Timing	Assertion/Negation-See the chip for specific timing information for these signals.
CLK_OUT	O	Diagnostic clock output. This output may be configured to offer one of a variety of internal system clocks to external hardware for diagnostic or debug purposes.	

4.2.2 System control signals

The table below describes some of the system control signals.

Table 6. System control signals: Detailed signal descriptions

Signal	I/O	Description	
HRESET_B	I/O	Power on reset. Causes the chip to abort all current internal and external transactions and set all registers to their default values. HRESET_B may be asserted completely asynchronously with respect to all other signals.	
		State Meaning	Asserted/Negated- See Power-On Reset Sequence Detailed Description
		Timing	Assertion/Negation-The chip data sheet gives specific timing information for this signal.

4.3 Reset Sequence Summary

- HRESET_B asserted
- Power supplies, configuration inputs, and clocks are stable
  - PCIe reference clock used as boot clock with system in PLL bypass
- HRESET_B negated
- POR CFG inputs sampled
- Fuses values read and loaded into registers
- IO drivers enabled per their default function
- Serdes released from reset and serdes PLL locks
- System logic is released from reset
- M4 is released from reset and performs initial configuration of PCIe interface including setting CFG_READY
- PCIe host provides Pre-boot initialization and Boot Loader pointers
- M4 execute Pre-boot initialization to enable primary clock source
- Platform PLL locks and Reset State Machine switches to PLL output to drive clock tree
- M4 detects completion of reset sequence and begins loading Boot Loader

4.4 Power-On Reset Sequence Timing

The figure below shows a timing diagram of the POR sequence.

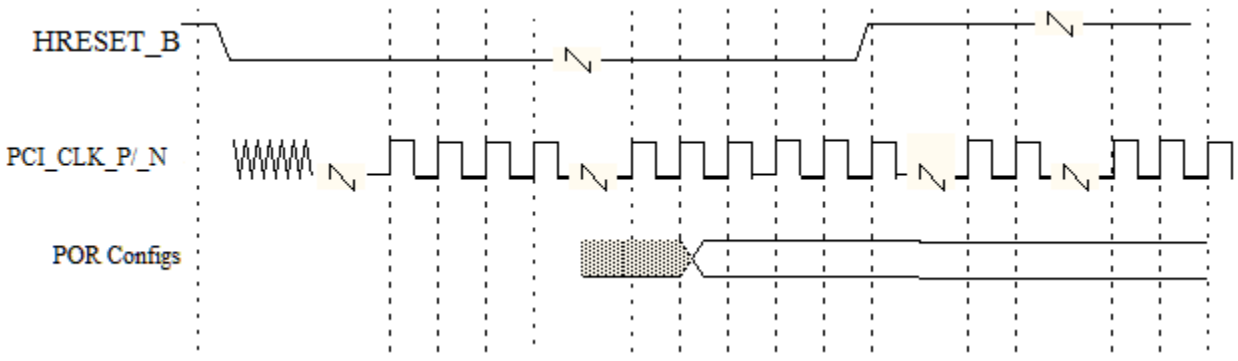


Figure 2. Power-On Reset Sequence

4.5 Power-On Reset Detailed Sequence

4.5.1 Power-On Reset Sequence Detailed Description

The table below contains the detailed POR sequence.

Table 7. Power-On Reset Sequence

Stage	Step
Initial Stages up to HRESET_B De-assertion	
1	Power Applied. Power is applied to comply to the *Device's Datasheet*.
2	HRESET_B Pin Assertion by External System Logic.
	Note: The common on-chip processor (COP), requires the ability to independently assert HRESET_B and TRST_B to fully control the processor. If a JTAG/COP port is used, follow the JTAG/COP interface connection recommendations given in the chip's data sheet. If the JTAG interface and COP header are not being used, it is recommended that TRST_B be tied to HRESET_B so that TRST_B is asserted when HRESET_B is asserted, ensuring that the JTAG scan chain is initialized during the power-on reset flow. All JTAG state machine logic is only reset with TRST_B assertion, and not on a HRESET_B assertion. See the JTAG configuration signals section in the chip's data sheet for more information.
3	PCI_CLK and Configuration Inputs Applied. PCI_CLK is the bypass clock during early POR until after pre-boot configuration. After pre-boot configuration, the PLL locks on DCS_CLK. Platform PLL is running in Bypass mode.
4	HRESET_B Pin signal De-assertion by External System Logic. External system logic negates HRESET_B after its required hold time (32 PCI_CLK cycles) and after POR configuration inputs have been valid for at least 4 PCI_CLK cycles after HRESET_B de-assertion.
POR Configuration	
5	Configuration Pins Sampled. The device samples the POR configuration input pins on de-assertion of HRESET_B.
Reset Stages	
6	Reset Platform Logic. The Reset block initiates and completes the reset of the rest of the platform logic.
7	SerDes Released from Reset. If PCIe is the boot source then the SerDes block is released from reset and the SerDes PLLs are locked . Note that once the cores are permitted to boot, software should check the the SerDes Control block registers to ensure that the SerDes PLLs have successfully locked as expected. If PCIe is not the boot source then SerDes PLLs are not locked at this point – boot loader software can release SerDes later if needed.
Pre-Boot Configuration Stages	
8	M4 Processor Released from Reset.

Table continues on the next page...

Table 7. Power-On Reset Sequence (continued)

Stage	Step
	The Cortex M4 core is released to begin execution from reset.
9	HW/SW Handshake: The POR Configuration options (CFG_RST_HNDSHK) cause this handshake stage to allow software to perform necessary configuration. This software can be a Boot Plugin routine, or it can be the bootloader or RTOS code downloaded by BootROM, or both. After the code sequence is complete, software (either BootROM or RTOS) performs a handshake with the reset state machine to allow the reset sequence to continue. The software then either begins polling for SYSREADY state or enters a WFE loop waiting for an event indicating SYSREADY has been reached, depending on CFG_SYSRDY_EVT configuration.
10	Pre-boot Configuration. The chip loads pre-boot configuration code from the boot source and the M4 core executes this code. After the code sequence is complete, execution returns to boot ROM code which performs a handshake with the reset state machine to allow the reset sequence to continue.
11	PLL begins to lock. The PLL begins to lock. SerDes PLL begin to lock in step 8.
PLL Locking and Reset Pause	
12	Platform PLL Drives Platform Clock Tree. I/O Drivers Enabled. All other I/O drivers are enabled at this point. Clocks are enabled to all IP including VSPA, AXIQ, and ADC-DAC.
13	HRESET_B De-assertion. The chip does not drive HRESET_B.
Further Initialization and System Ready	
14	Completion of IP Initialization. The Reset block waits for all IPs to complete their initialization processes. Time Zero now get established.
15	System Ready State. The device enters the System Ready state when all initialization processes have completed. The PCI-Express interfaces are released to accept external requests and the M4 core is allowed to exit reset if allowed by the Reset block Boot Release Register (BRR).

4.6 Power-On Reset Configuration

Various device functions are initialized by sampling certain signals during power on reset.

The values of all these signals are sampled into registers when PORESET_B is deasserted. These inputs are to be pulled high or low by external resistors. During PORESET_B, all other signal drivers connected to these signals must be in the highimpedance state.

All POR configuration signals have internal pull-up resistors so that if the desired setting is high, there is no need for a pull-up resistor on the board. See the chip data sheet for proper pull-down resistor values for POR configuration signals, when necessary. This section describes the functions and modes configured by the POR configuration signals.

Configuration Functionality (Soc Pin Name)	Description	Register Where Value Recorded
CFG_BOOT_SRC[1:0] (TXRX[1:0])	Boot Source. 2'b00: Reserved 2'b01: Reserved 2'b10: I2C EEPROM (pulldown on TXRX[0]) 2'b11: PCIe Host Memory (no pulldowns)	PORSR1[15:14] in DCFG block also PORSR2[23:22]
CFG_BOOT_HO* (LNA3_EN)	Boot Holdoff (active low) 0: Reserved 1: Normal boot with M4 executing BootROM and performing handshake.	PORSR1[13] in DCFG block also PORSR2[21]
CFG_PCIE_GEN (LNA2_EN)	PCIe Generation (consumed by BootROM only) 0: PCIe interface will be restricted to PCIe Gen2 frequencies. 1: PCIe interface will allow negotiation to PCIe Gen3 frequencies	PORSR1[12] in DCFG block also PORSR2[20]
CFG_TEST_PORT_DIS (LNA1_EN)	Reserved	PORSR1[20] in DCFG block
CFG_RST_HNDSHK* (PA_EN)	Reset HW/SW Handshake 0: Reserved 1: Normal boot with M4 executing BootROM and performing handshake.	PORSR2[30] in DCFG block
CFG_WRM_RSTB (PPS_OUT)	Warm Reset (active low) (consumed by BootROM only) 0: Reset has been asserted after a functional mode has been reached previously (and specifically after memories have been initialized). Power was not removed prior to the reset assertion. Memory initialization steps may be skipped. 1: Memories must be initialized.	PORSR2[31] in DCFG block
CFG_SYSRDY_EVT (SPI_MOSI)	SYSREADY Event (active low) (consumed by BootROM only) 0: Boot Sequence determines SYSREADY state via an event which wakes M4 from WFE	PORSR2[19] in DCFG block also PORSR1[7]

Table continues on the next page...

Table continued from the previous page...

Configuration Functionality (Soc Pin Name)	Description	Register Where Value Recorded
	1: Boot Sequence determines SYSREADY state by polling Reset FSM state	

4.7 Reset register descriptions

The table below shows the memory-mapped and register definition of the Reset module and lists the offset, name, and a cross-reference to the complete description of each register. These registers only support 32-bit accesses.

4.7.1 RST_CCSR Memory map

RST base address: 1E6_0000h

Offset	Register	Width (In bits)	Access	Reset value
Ch	Reset Status Register (RSTSR)	32	RO	0000_0000h
10h	Reset Request Mask Register (RSTRQMR1)	32	RW	0002_0000h
18h	Reset Request Status Register (RSTRQSR1)	32	RW	0000_0000h
60h	Boot Release Register (BRRL)	32	RW	0000_0000h
90h	Core Enable Boot Release Status Register (BRCORENBR)	32	RO	0000_0000h
104h	RCW Completion Register (RCW_COMPLETIONR)	32	RW	0000_0000h
114h	PBI Completion Register (PBI_COMPLETIONR)	32	RW	0000_0000h
400h	Core Reset Status Register n (CRSTSR0)	32	W1C	0000_0000h
BF8h	IP Block Revision 1 Register (IP_REV1)	32	RO	0281_0500h
BFCh	IP Block Revision 2 Register (IP_REV2)	32	RO	See description.

4.7.2 Reset Status Register (RSTSR)

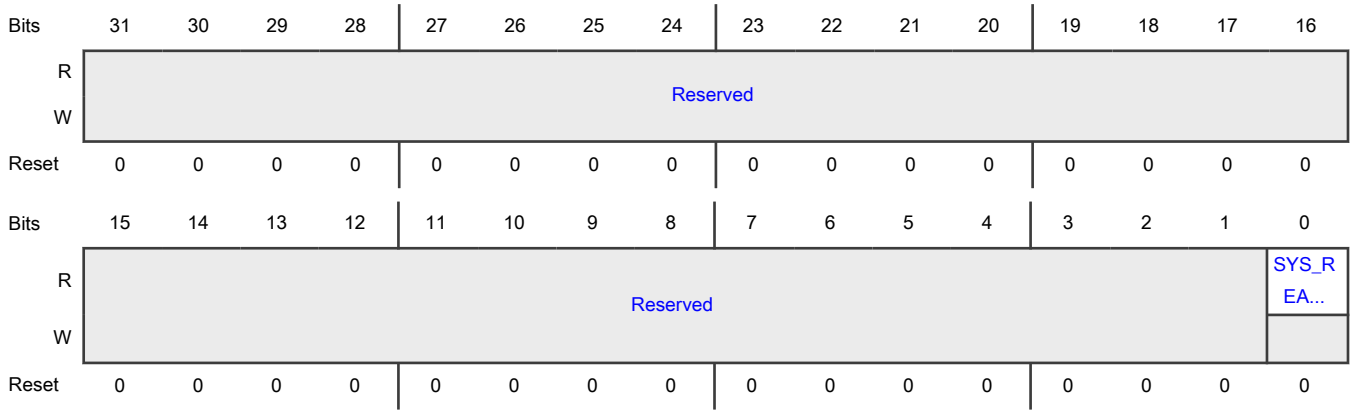
Offset

Register	Offset
RSTSR	Ch

Function

The Reset Status register indicates the system ready status.

Diagram



Fields

Field	Function
31-1 —	Reserved.
0 SYS_READY	This bit indicates the system ready status 0b - System is not in ready status 1b - System is in ready status

4.7.3 Reset Request Mask Register (RSTRQMR1)

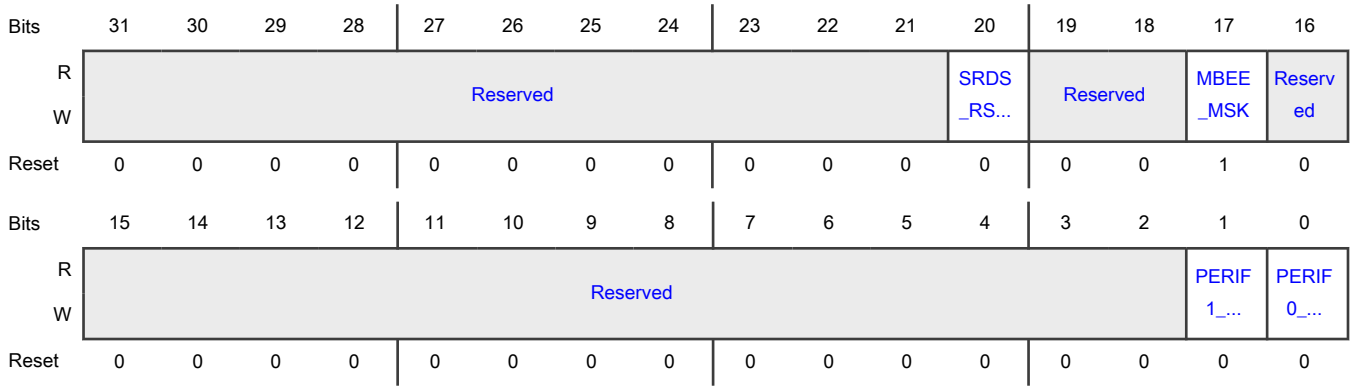
Offset

Register	Offset
RSTRQMR1	10h

Function

The RSTRQMR1 contains mask bits that trigger eDMA to generate an MSI request over PCIe to the host.

Diagram



Fields

Field	Function
31-21 —	Reserved.
20 SRDS_RST_MSK	SerDes reset request event mask 0b - SerDes reset event can cause a reset request 1b - SerDes reset event cannot cause a reset request
19-18 —	Reserved.
17 MBEE_MSK	Multi-bit ECC error reset request mask 0b - Multi-bit ECC error event can cause a reset request 1b - Multi-bit ECC error event cannot cause a reset request. MBEE_MSK field must be preset to 1 to not cause a reset request due to multi-bit ECC error on boot due to some IP reading memories before they are initialized.
16-2 —	Reserved.
1 PERIF1_MSK	M4 lockup error request 0b - M4 lockup error request request is not active 1b - M4 lockup error request request is active
0 PERIF0_MSK	M4 WDOG second timeout request 0b - M4 WDOG second timeout request is not active 1b - M4 WDOG second timeout request is active

4.7.4 Reset Request Status Register (RSTRQSR1)

Offset

Register	Offset
RSTRQSR1	18h

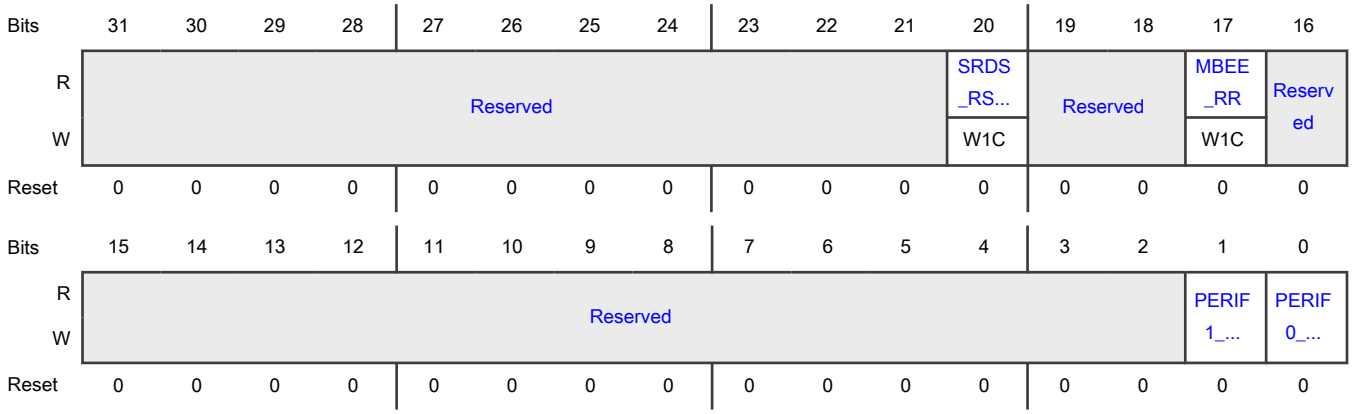
Function

The RSTRQSR1 contains status bits that record the reasons that triggered the eDMA to generate an MSI request over PCIe to the host.

NOTE

For the different sources captured in this register, some of these can be serviced HRESET_B.

Diagram



Fields

Field	Function
31-21 —	Reserved.
20 SRDS_RST_RR	SerDes reset event. Occurs if any enabled SerDes PLL does not lock. 0b - SerDes reset request event not active 1b - SerDes reset request event active
19-18 —	Reserved.
17 MBEE_RR	Multi-bit ECC reset request Platform internal memory multi-bit ECC error requires device level PORESET_B or HRESET_B. 0b - Multi-bit ECC error reset request event not active 1b - Multi-bit ECC error reset request event active
16-2 —	Reserved.
1 PERIF1_RR	M4 lockup error request 0b - M4 lockup error request request is not active 1b - M4 lockup error request request is active
0 PERIF0_RR	M4 WDOG second timeout request 0b - M4 WDOG second timeout request is not active 1b - M4 WDOG second timeout request is active

4.7.5 Boot Release Register (BRRL)

Offset

Register	Offset
BRRL	60h

Function

The BRR contains control bits for enabling boot for each core.

NOTE

If you change a bit from 1 to 0 outside of warm reset, results are boundly undefined for that core.

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															CR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved.
0 CR0	Core 0 Release. 0b - Core is in Boot Holdoff and not released for Booting 1b - Core released for Booting

4.7.6 Core Enable Boot Release Status Register (BRCORENBR)

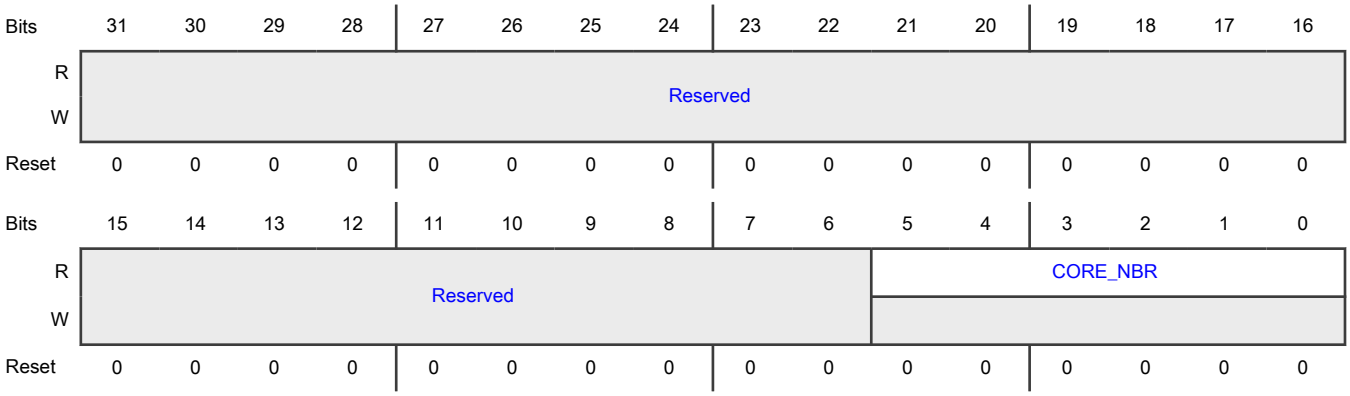
Offset

Register	Offset
BRCORENBR	90h

Function

The BRCORENBR contains the status bits to indicate the Boot Release location.

Diagram



Fields

Field	Function
31-6 —	Reserved.
5-0 CORE_NBR	Core Number This field designates which core will begin booting out of reset if enabled using POR Config Input BOOT_HO (which refers to this core as 'core X' 000000b - Core 0 can be release for booting on existing reset(BRR is loaded at reset with 32'h0000_0001)

4.7.7 RCW Completion Register (RCW_COMPLETIONR)

Offset

Register	Offset
RCW_COMPLETIONR	104h

Function

The RCW_COMPLETIONR contains the control bits written by the Service Processor to indicate that it finished the RCW loading.

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															DONE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved.
0 DONE	RCW done bit. 0b - M4 core has not yet completed the RCW loading 1b - M4 core has completed the RCW loading

4.7.8 PBI Completion Register (PBI_COMPLETIONR)

Offset

Register	Offset
PBI_COMPLETIONR	114h

Function

The PBI_COMPLETIONR contains the control bits written by the M4 core to indicate that it finished the PBI phase.

Diagram

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

Fields

Field	Function
31-24 —	Reserved.
23-16 ERR_CODE	PBI error code. 00000000b - M4 core is in Boot Holdoff and not released for Booting 00000001b - M4 core released for Booting
15-1 —	Reserved.
0 DONE	0b - M4 core has not yet completed the PBI phase 1b - M4 core has completed the PBI phase

4.7.9 Core Reset Status Register n (CRSTSR0)

Offset

Register	Offset
CRSTSR0	400h

Function

The CRSTSRn contains the reset status bits for each core

A power-on reset of the device causes the following to occur:

- RST_HRESET is set
- All other bits are cleared

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	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	Reserved				Reserved				Reserved				Reserved	Reserved		RST_POR...
W																W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved.
27-24 —	Reserved.
23-20 —	Reserved.
19-16 —	Reserved.
15-12 —	Reserved.
11-8 —	Reserved.
7-4 —	Reserved.
3 —	Reserved.
2-1 —	Reserved.
0 RST_PORST	Core was reset due to a PORESET

4.7.10 IP Block Revision 1 Register (IP_REV1)

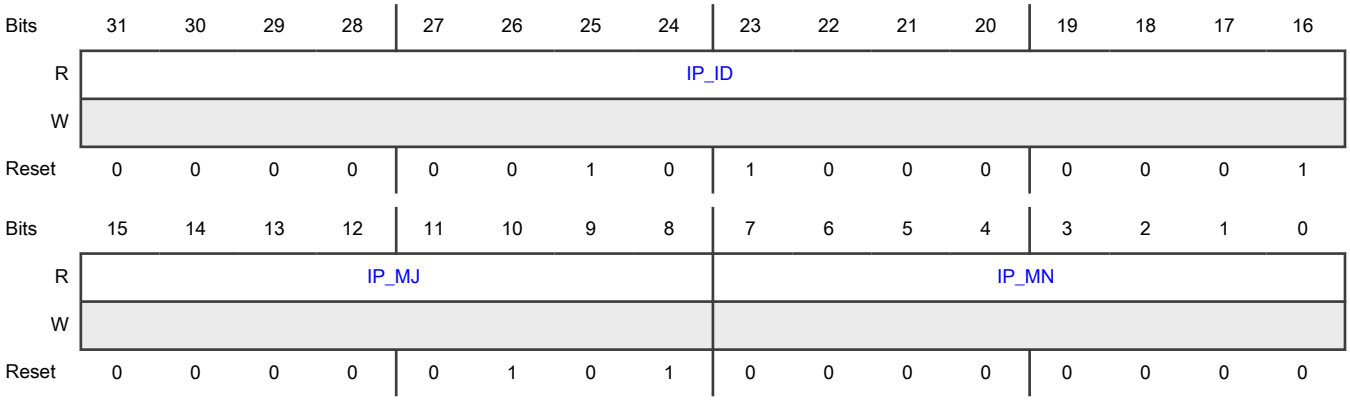
Offset

Register	Offset
IP_REV1	BF8h

Function

The IP_REV1 Register contains the block revision fields.

Diagram



Fields

Field	Function
31-16 IP_ID	Block ID 0001.
15-8 IP_MJ	Major revision 04.
7-0 IP_MN	Minor revision 01.

4.7.11 IP Block Revision 2 Register (IP_REV2)

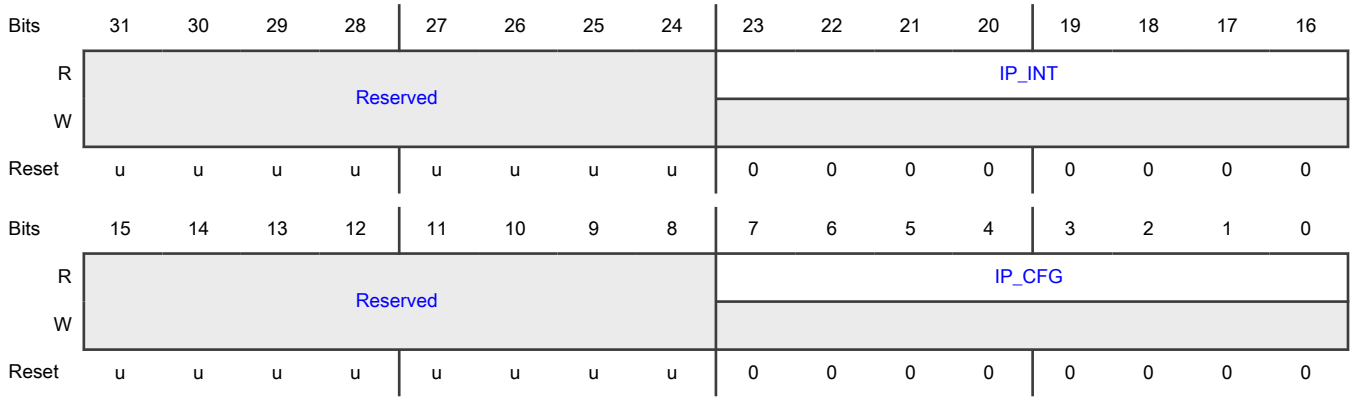
Offset

Register	Offset
IP_REV2	BFCh

Function

The IP_REV2 Register contains the block revision fields.

Diagram



Fields

Field	Function
31-24 —	Reserved.
23-16 IP_INT	Block ID 0001.
15-8 —	Reserved.
7-0 IP_CFG	IP block configuration ID (option).

4.8 Clocking

Once the reset sequence is complete, the chip is a completely synchronous system (with the exception of the SerDes interface), driven from PLL. The reference clock for the PLL is a differential input of up 153.6MHz, and in 5G applications is often set to 122.88MHz. Future discussions of internal frequencies assume a 153.6MHz input. It comes from an external source like VCXO. The PLL will drive a 614.4 MHz clock to a single clock domain to drive clock regenerators throughout the chip. VSPA and portions of the system interconnect will run at the full 614.4 MHz; the remainder of the chip will run at lower frequency outputs of the clock regen circuits, as controlled by the ipg_sync signals.

Since the 153.6 MHz clock is not available, the chip runs off of the 100 MHz PCIe reference clock during the reset sequence and then the PLL locks to the the 153.6 MHz clock. The chip is clocked from the 153.6MHz clock thereafter.

4.9 Clock Distribution and Configuration

The Clock Generation Unit on the chip supports a single PLL apart from the SerDes. The following figure describes the high level clocking architecture of the chip.

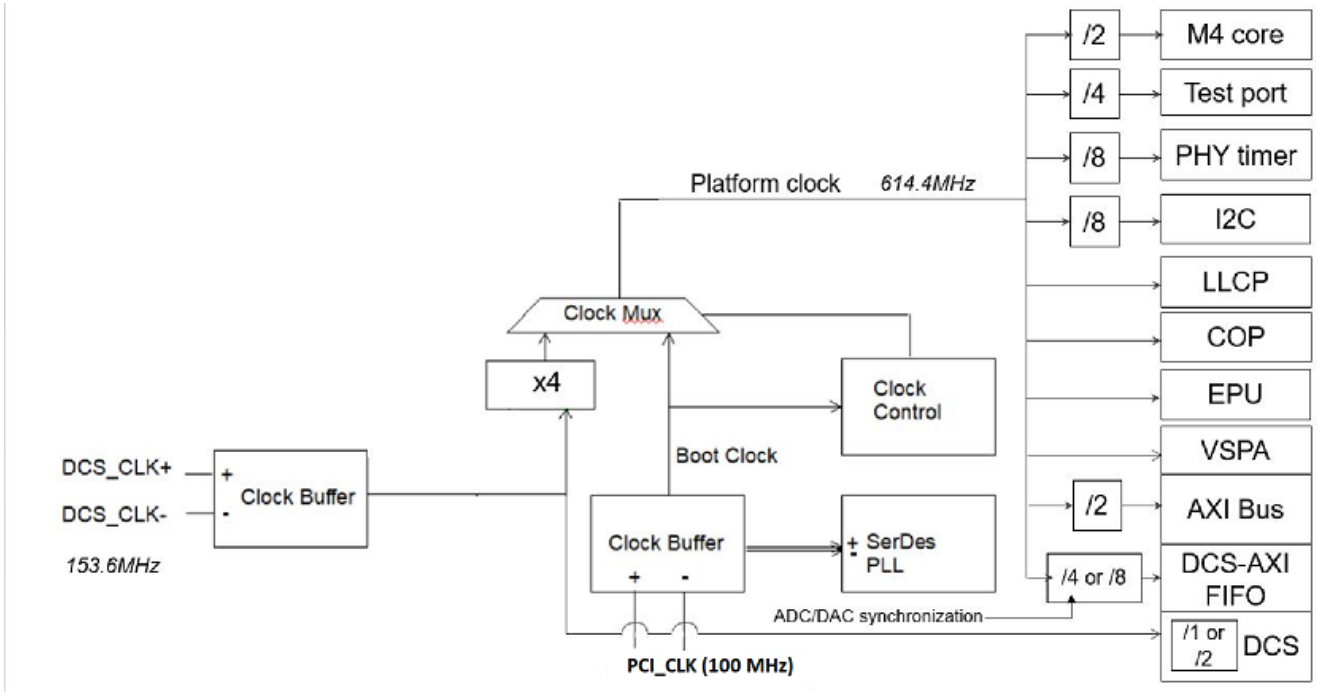


Figure 3. Clock Generation

4.10 Reset register descriptions

The table below shows the memory-mapped and register definition of the Reset module and lists the offset, name, and a cross-reference to the complete description of each register. These registers only support 32-bit accesses.

4.10.1 RST_CCSR Memory map

RST base address: 1E6_0000h

Offset	Register	Width (In bits)	Access	Reset value
Ch	Reset Status Register (RSTSR)	32	RO	0000_0000h
10h	Reset Request Mask Register (RSTRQMR1)	32	RW	0002_0000h
18h	Reset Request Status Register (RSTRQSR1)	32	RW	0000_0000h
60h	Boot Release Register (BRRL)	32	RW	0000_0000h
90h	Core Enable Boot Release Status Register (BRCORENBR)	32	RO	0000_0000h
104h	RCW Completion Register (RCW_COMPLETIONR)	32	RW	0000_0000h
114h	PBI Completion Register (PBI_COMPLETIONR)	32	RW	0000_0000h
400h	Core Reset Status Register n (CRSTSR0)	32	W1C	0000_0000h
BF8h	IP Block Revision 1 Register (IP_REV1)	32	RO	0281_0500h
BFCh	IP Block Revision 2 Register (IP_REV2)	32	RO	See description.

4.10.2 Reset Status Register (RSTSR)

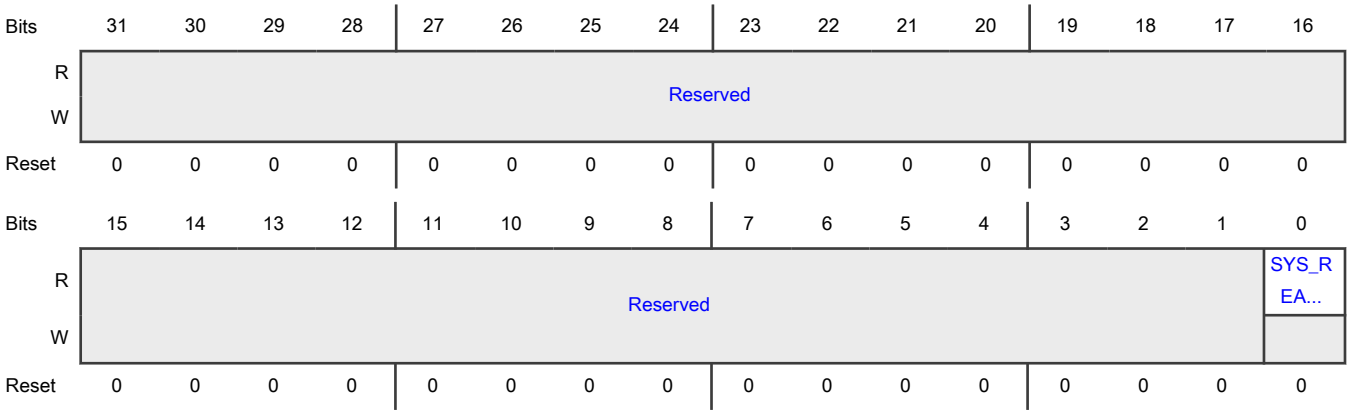
Offset

Register	Offset
RSTSR	Ch

Function

The Reset Status register indicates the system ready status.

Diagram



Fields

Field	Function
31-1 —	Reserved.
0 SYS_READY	This bit indicates the system ready status 0b - System is not in ready status 1b - System is in ready status

4.10.3 Reset Request Mask Register (RSTRQMR1)

Offset

Register	Offset
RSTRQMR1	10h

Function

The RSTRQMR1 contains mask bits that trigger eDMA to generate an MSI request over PCIe to the host.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved											SRDS_RS...	Reserved		MBEE_MSK	Reserved
Reset	0	0	0	0	0	0	0	0	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 W	Reserved														PERIF1_...	PERIF0_...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-21 —	Reserved.
20 SRDS_RST_MSK	SerDes reset request event mask 0b - SerDes reset event can cause a reset request 1b - SerDes reset event cannot cause a reset request
19-18 —	Reserved.
17 MBEE_MSK	Multi-bit ECC error reset request mask 0b - Multi-bit ECC error event can cause a reset request 1b - Multi-bit ECC error event cannot cause a reset request. MBEE_MSK field must be preset to 1 to not cause a reset request due to multi-bit ECC error on boot due to some IP reading memories before they are initialized.
16-2 —	Reserved.
1 PERIF1_MSK	M4 lockup error request 0b - M4 lockup error request request is not active 1b - M4 lockup error request request is active
0 PERIF0_MSK	M4 WDOG second timeout request 0b - M4 WDOG second timeout request is not active 1b - M4 WDOG second timeout request is active

4.10.4 Reset Request Status Register (RSTRQSR1)

Offset

Register	Offset
RSTRQSR1	18h

Function

The RSTRQSR1 contains status bits that record the reasons that triggered the eDMA to generate an MSI request over PCIe to the host.

NOTE

For the different sources captured in this register, some of these can be serviced HRESET_B.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved											SRDS_RS...	Reserved		MBEE_RR	Reserved
W												W1C			W1C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved														PERIF1_...	PERIF0_...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-21 —	Reserved.
20 SRDS_RST_RR	SerDes reset event. Occurs if any enabled SerDes PLL does not lock. 0b - SerDes reset request event not active 1b - SerDes reset request event active
19-18 —	Reserved.
17 MBEE_RR	Multi-bit ECC reset request Platform internal memory multi-bit ECC error requires device level PORESET_B or HRESET_B. 0b - Multi-bit ECC error reset request event not active

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Multi-bit ECC error reset request event active
16-2 —	Reserved.
1 PERIF1_RR	M4 lockup error request 0b - M4 lockup error request request is not active 1b - M4 lockup error request request is active
0 PERIF0_RR	M4 WDOG second timeout request 0b - M4 WDOG second timeout request is not active 1b - M4 WDOG second timeout request is active

4.10.5 Boot Release Register (BRRL)

Offset

Register	Offset
BRRL	60h

Function

The BRR contains control bits for enabling boot for each core.

NOTE

If you change a bit from 1 to 0 outside of warm reset, results are boundly undefined for that core.

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															CR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved.
0 CR0	Core 0 Release. 0b - Core is in Boot Holdoff and not released for Booting 1b - Core released for Booting

4.10.6 Core Enable Boot Release Status Register (BRCORENBR)

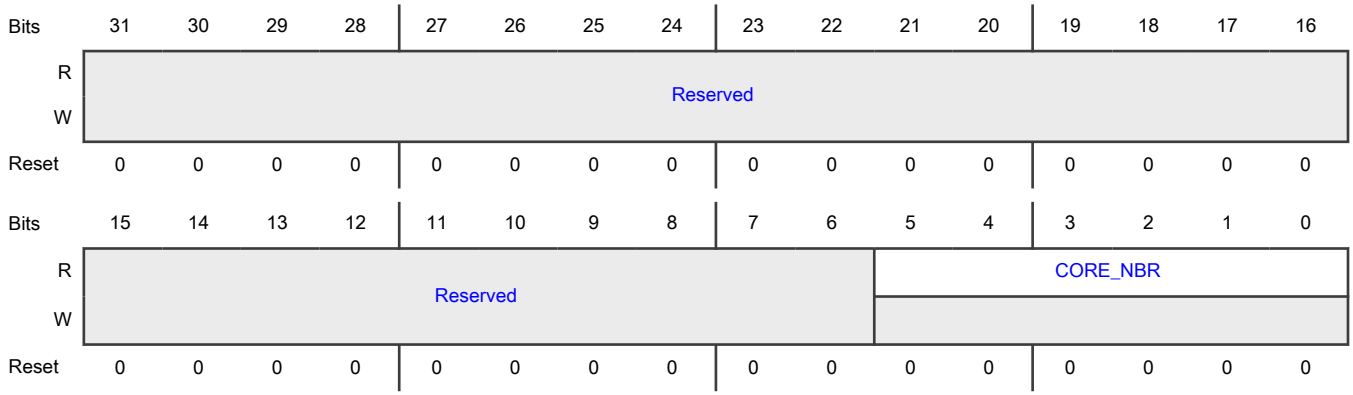
Offset

Register	Offset
BRCORENBR	90h

Function

The BRCORENBR contains the status bits to indicate the Boot Release location.

Diagram



Fields

Field	Function
31-6 —	Reserved.
5-0 CORE_NBR	Core Number This field designates which core will begin booting out of reset if enabled using POR Config Input BOOT_HO (which refers to this core as 'core X'

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000000b - Core 0 can be release for booting on existing reset(BRR is loaded at reset with 32'h0000_0001)

4.10.7 RCW Completion Register (RCW_COMPLETIONR)

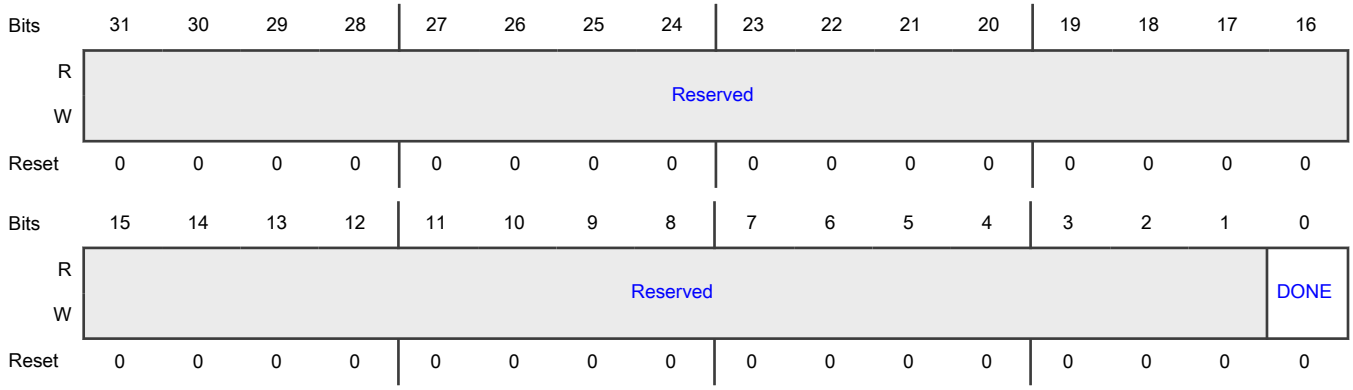
Offset

Register	Offset
RCW_COMPLETIONR	104h

Function

The RCW_COMPLETIONR contains the control bits written by the Service Processor to indicate that it finished the RCW loading.

Diagram



Fields

Field	Function
31-1 —	Reserved.
0 DONE	RCW done bit. 0b - M4 core has not yet completed the RCW loading 1b - M4 core has completed the RCW loading

4.10.8 PBI Completion Register (PBI_COMPLETIONR)

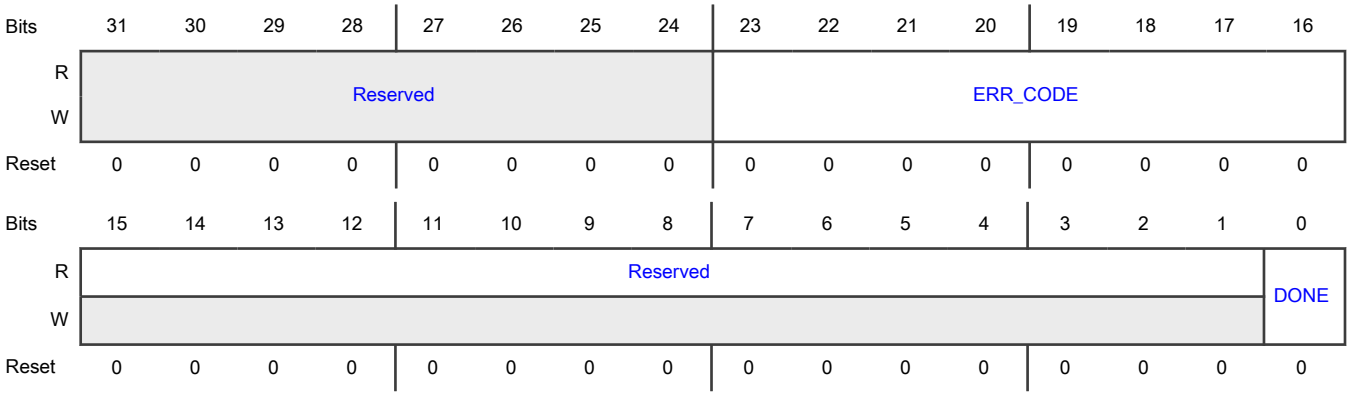
Offset

Register	Offset
PBI_COMPLETIONR	114h

Function

The PBI_COMPLETIONR contains the control bits written by the M4 core to indicate that it finished the PBI phase.

Diagram



Fields

Field	Function
31-24 —	Reserved.
23-16 ERR_CODE	PBI error code. 00000000b - M4 core is in Boot Holdoff and not released for Booting 00000001b - M4 core released for Booting
15-1 —	Reserved.
0 DONE	0b - M4 core has not yet completed the PBI phase 1b - M4 core has completed the PBI phase

4.10.9 Core Reset Status Register n (CRSTSR0)

Offset

Register	Offset
CRSTSR0	400h

Function

The CRSTSRn contains the reset status bits for each core

A power-on reset of the device causes the following to occur:

- RST_HRESET is set
- All other bits are cleared

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				Reserved				Reserved				Reserved	Reserved		RST_POR...
W																W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved.
27-24 —	Reserved.
23-20 —	Reserved.
19-16 —	Reserved.
15-12 —	Reserved.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-8 —	Reserved.
7-4 —	Reserved.
3 —	Reserved.
2-1 —	Reserved.
0 RST_PORST	Core was reset due to a PORESET

4.10.10 IP Block Revision 1 Register (IP_REV1)

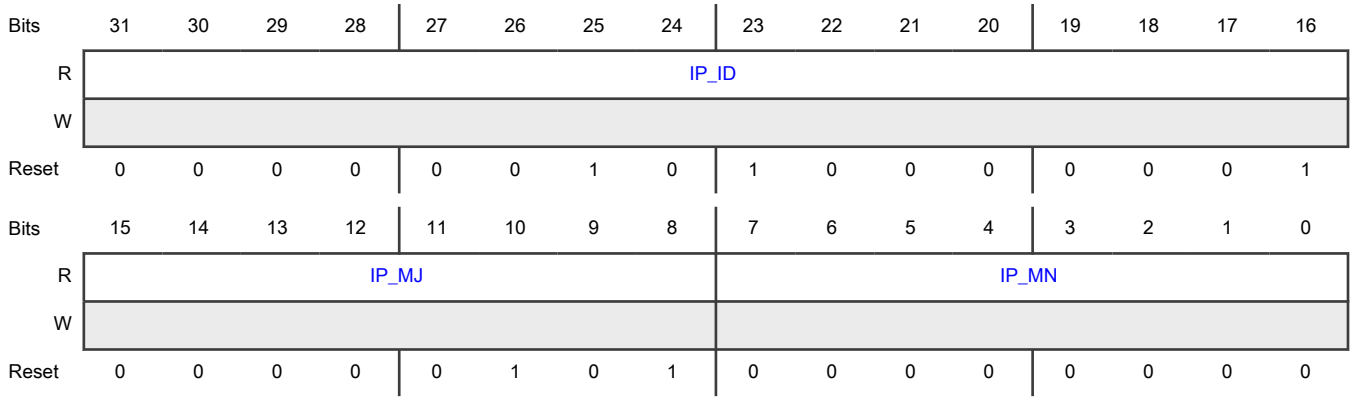
Offset

Register	Offset
IP_REV1	BF8h

Function

The IP_REV1 Register contains the block revision fields.

Diagram



Fields

Field	Function
31-16 IP_ID	Block ID 0001.
15-8 IP_MJ	Major revision 04.
7-0 IP_MN	Minor revision 01.

4.10.11 IP Block Revision 2 Register (IP_REV2)

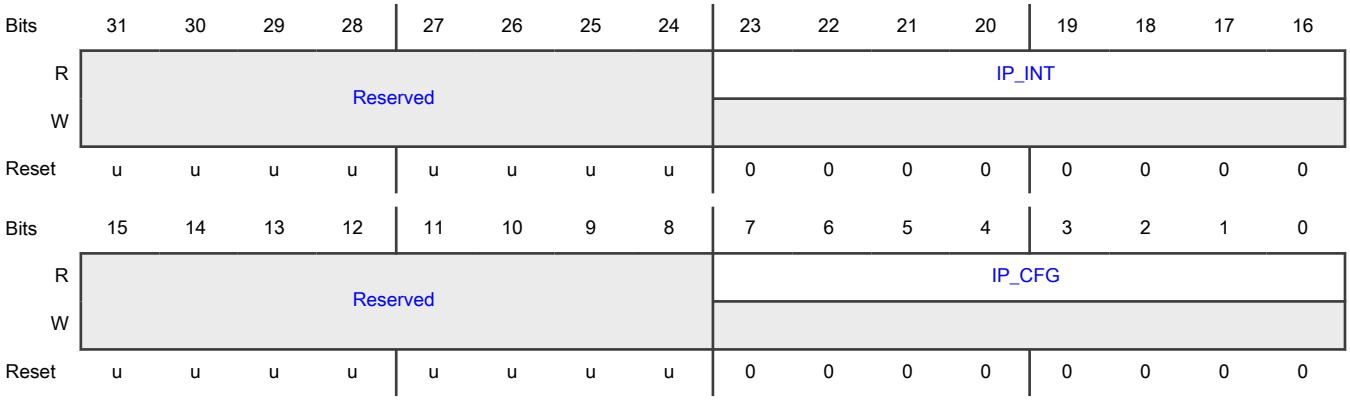
Offset

Register	Offset
IP_REV2	BFCh

Function

The IP_REV2 Register contains the block revision fields.

Diagram



Fields

Field	Function
31-24 —	Reserved.
23-16	Block ID

Table continues on the next page...

Table continued from the previous page...

Field	Function
IP_INT	0001.
15-8 —	Reserved.
7-0 IP_CFG	IP block configuration ID (option).

4.11 CCU register descriptions

4.11.1 CCU_ccsr Memory map

COPCLK base address: 137_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Core Cluster 1 Clock Control/Status Register (CLKC1CSR)	32	RW	0000_0000h
20h	Core Cluster 2 Clock Control/Status Register (CLKC2CSR)	32	RW	0000_0000h

4.11.2 Core Cluster a Clock Control/Status Register (CLKC1CSR - CLKC2CSR)

Offset

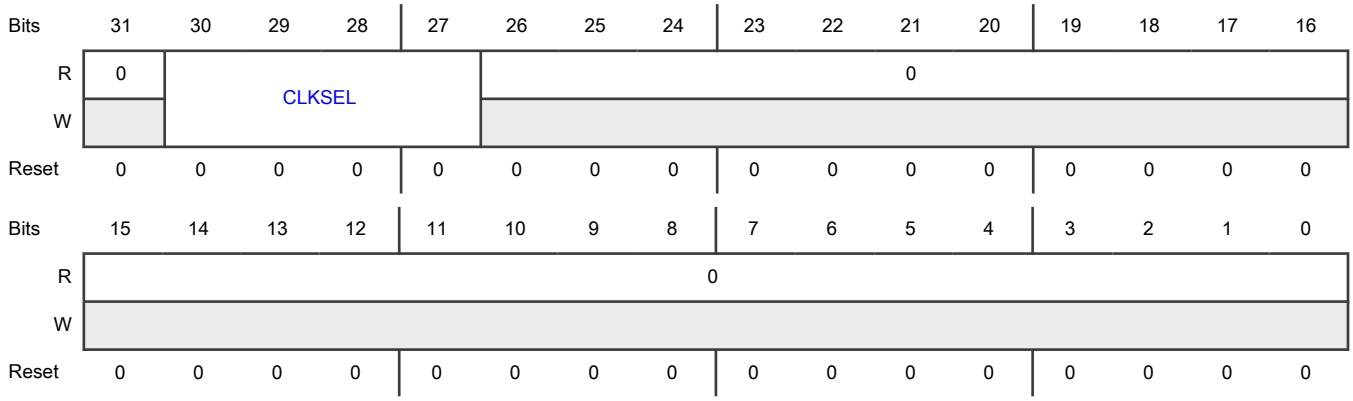
For a = 1 to 2:

Register	Offset
CLKCaCSR	-20h + (a × 20h)

Function

The CLK1-2 CSR registers control the clock frequency selection for each clock generator's core cluster clock mux. CLK1CSR corresponds to core cluster 1.

Diagram



Fields

Field	Function
31 —	Reserved.
30-27 CLKSEL	CLKSEL[3:0] Selects the clock source for this clock domain. 0000 - Cluster PLL1 output 0001 - Cluster PLL1 divide-by-2 0010 - Cluster PLL1 divide-by-4 0011 - Reserved 0100 - Cluster PLL2 output 0101 - Cluster PLL2 divide-by-2 0110 - Cluster PLL2 divide-by-4 0111 - Reserved For two PLLs per cluster: 0111 - 1111 - Reserved 0000b - Cluster PLL1 output 0001b - Cluster PLL1 divide-by-2 0010b - Cluster PLL1 divide-by-4 0011b - Reserved 0100b - Cluster PLL2 output 0101b - Cluster PLL2 divide-by-2 0110b - Cluster PLL2 divide-by-4 0111b - Reserved 0111-1111b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
26-0 —	Reserved.

Chapter 5
Boot Overview

5.1 Boot overview

Boot Source Scenarios

1. PCIe boot from host memory.
2. I2C boot from EEPROM.

In all boot source scenarios, the100MHz PCIe reference clock is required even if there will never be any PCIe communication.

Standard Boot Sequence Overview

The following sequence provides an overview of the boot process, beginning with the release of the M4 core from reset while the SoC is being clocked by the “boot clock”, which will be the 100MHz reference clock needed for the PCIe interface.

- Reset State Machine releases the M4 core to execute from Boot ROM
- Boot ROM performs initial configuration to allow communication with PCIe host including setting CFG_READY
- Host configures PCIe endpoint configuration header and address translation windows
- Host writes Boot Header to LA9310 TCM, pointing the Boot ROM code to Boot Plugin Routine for Pre-Boot Initialization
- Boot ROM pulls Boot Plugin routine (and associated data/firmware) to TCM Code memory and executes it as a subroutine.
- Boot ROM returns control to the Reset FSM through a register handshake and then waits for notification that the reset sequence is complete (including PLL lock and transition of clock tree to PLL output). This is done through either polling Reset FSM state or a hardware event (based on CFG_SYSRDY_EVT input).
- Boot ROM then pulls Boot Loader from its source as indicated previously in the Boot Header. It can place it at an offset from the base of TCM Code Memory, as specified in the Boot Header.
- Boot ROM jumps to the Boot Loader, either at its Destination offset from the base of TCM code memory, or at an entry point offset from the Destination Offset by an Entry Point value specified in the Boot Header.

5.2 Boot RoM

When the Cortex M4 core is released from reset it attempts to execute instructions from a Boot ROM located at address 0x0000_0000 in the LA9310 address map. Note that the Boot ROM does *not* poll for an RCW_REQ input in LA9310 as it does in Chassis based devices; it will progress immediately as long as it is not held in Boot Holdoff mode. See Section 11.7, Boot Holdoff Mechanism. The Boot ROM occupies 8KB of memory space. During execution the Boot ROM code uses the highest 4KB region of TCMU (Tightly Coupled Data Memory) as its stack space – 0x2000_F000-0x2000_FFFF.

5.3 Boot Plugin Routine

The Boot Plugin Routine is an executable software routine that runs on the M4 core to do pre-boot initialization of the device or system. It is intended to do any initialization that is required before the Reset State Machine completes the reset process.

Specifically, in a system where the RFIC provides the reference clock to the LA9310 PLL, and the RFIC must first be initialized with firmware and other configuration commands before it supplies this clock. Therefore, a basic Boot Plugin Routine may be employed to perform this initialization early in the reset sequence before the Reset FSM can complete the reset sequence, locking the LA9310 PLL to the input clock from the RFIC.

NOTE

The above example motivated the creation of a Boot Plugin Routine capability. However, RFIC configuration may actually be carried out by a FreeRTOS driver after BootROM exits. See Early_Exit in Table 11-2 as well as **Section 11.6**, *BootROM vs RTOS for Reset Sequence Handshake*

5.4 Boot header

The Boot Header consists of 8 32-bit words of information used by the Boot ROM. The structure is 32-bit aligned and all elements are padded to fill 32-bit total size. It includes the following:

Table 8.

Header field	Field size	Description
preamble	32b	Preamble identifying a valid boot header. It is located at the base address of the boot source memory. Must equal 0xAA55_AA55
plugin_size	32b	The size, in bytes, of the Plugin Routine to be loaded. Includes instructions and associated data and firmware. If plugin_size == 0, then skip execution of Plugin Routine.
plugin_offset	32b	The offset, in bytes, of the Plugin Routine in the boot source memory. This is a non-zero offset from the base address of the boot source memory where the preamble was found. The Plugin Routine will be copied to the base of TCM Code memory
bl_size	32b	The size, in bytes, of the Boot Loader. Includes instructions and associated data and firmware.
bl_src_offset	32b	The offset, in bytes, of the Boot Loader from the base address of the boot source memory. If the source is I2C, this is the offset from the beginning of the preamble. If the source is PCIe, this is the offset from the base of the host memory made accessible to LA9310. (PCIe space, 0xA000_0000)
bl_dest	32b	The absolute 32-bit physical address to which the Boot Loader will be copied in TCM Code Memory (TCML). The Boot Loader may or may not overwrite any Plugin Routine that was placed in TCM Code memory based on this destination address.
bl_entry	32b	The absolute 32-bit physical address of the code entry point in the Boot Loader. Boot Loader execution will begin at this address.
flags	32b	Boot options specified by Host.

Boot header details

The target containing the Boot Plugin and Boot Loader is indicated by the Power-On Reset Configuration input field CFG_BOOT_SRC, which can be read from the PORSR register in the Device Configuration block (DCFG). If there is no Boot Plugin routine, the the Boot Plugin size should be set to zero, and Boot ROM code will skip past the Plugin phase. If the Boot Source is PCIe Host memory, then the offset is from the base of the host memory range that is exposed to LA9310. If the Boot Source is I2C Extended Addressing EEPROM, the offset is relative to address 0x0 of the I2C target. If the Boot Source is I2C, the offset is relative to address 0x0 of the I2C target.

NOTE

Extended Addressing I2C Boot Source

"Extended Addressing" refers to EEPROMs with a calling address of 7'b1010aaa (ROM code only supports one calling address with 'aaa' = 3'h0) which require two full bytes of data word address after the calling address.

Boot Header Flags

The bits of the flags field are used to convey information to the BootROM regarding the processing as described in Figure 11-1.

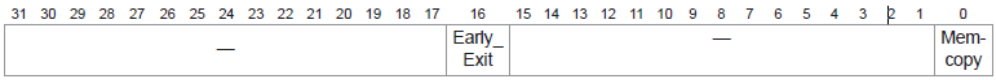


Figure 11-1. Boot Header Flags

Table 11-2. Boot Header Flag Descriptions

Bits	Name	Description
31-17	—	Reserved.
16	Early_Exit	Early Exit to RTOS When set, this flag causes the BootROM to exit and begin executing the next level of Boot Loader or RTOS immediately after downloading it. It becomes the responsibility of the downloaded code to complete the Rattler reset sequence as the BootROM would have done. The expectation is that the "boot loader" downloaded is actually the full RTOS.
15-1	—	Reserved.
0	Memcopy	Memcopy based code download When set, this flag causes BootROM to use a memcopy routine run on the M4 core to copy the plugin routine and/or the bootloader from host memory. When left clear, the BootROM uses the eDMA engine to copy code from the host memory

5.5 Boot sequence

- M4 begins execution in ROM
- M4 Reads CFG_BOOT_SRC from POR Configuration settings in the Device Configuration block (DCFG) block
- If CFG_BOOT_SRC = 2'b00 (preloaded instruction memory)
  - — There is no Pre-Boot Initialization or Boot Loader download or TCM initialization
  - — The boot ROM code jumps past the Pre-Boot Initialization sequence and instead immediately handshakes with the Reset FSM to allow it to complete the reset sequence. — The code sequence then jumps past any Boot Loader download and jumps immediately to offset 0 from the base of the TCM Code memory (TCML).
  - — No further ROM code is executed because the Boot Loader in M4 Code memory takes over.
- If CFG_BOOT_SRC = 2'b10 (I2C EEPROM - calling address 7'h50)
  - — ROM initializes both TCM memory arrays with writes of 0x0000 to initialize ECC memory to valid values.
  - — ROM reads I2C offset 0x0 for a valid Boot Preamble

- Supported devices are extended address EEPROM with calling address 7’b101_0aaa. BootROM only supports ‘aaa’=3’b000.
- ROM reads the following 7 words to find size and offset of the Plugin Routing and Boot Loader
- If the Plugin Routine size is nonzero:
- ROM code downloads the Plugin Routine from the I2C EEPROM and writes it to the base of TCM Code memory at local address 0x1F80_0000.
- When the copy is complete, ROM code makes a subroutine call to TCM Code memory at 0x1F80_0000.
- The Plugin routine executes (e.g. programming the RFIC and configuring a valid clock to support completion of the reset sequence) then returns to the ROM code.
- If header.flags[16]=0 (no “Early_Exit”):
- ROM code handsakes with the Reset FSM allowing hardware to complete the reset sequence all the way to the SYSREADY state
- ROM code waits for reset sequence to complete (either through polling the Reset FSM state or by configuring EPU to generate RXEV event upon Reset FSM reaching SYSREADY then entering a WFE loop - based on CFG_SYSRDY_EVT input).
- ROM code then downloads the Boot Loader from the I2C EEPROM according to the size and offset previously found in the Boot Header and writes it to the TCM Code or Data memory as specified by the BL_DEST value.
- When the copy is complete, code jumps from ROM to the Boot Loader at the BL_ENTRY location.

- If CFG_BOOT_SRC = 2’b11 (PCIe Host)
  - ROM initializes both TCM memory arrays with writes of 0x0000 to initialize ECC memory to valid values.
  - ROM configures an iATU inbound translation for BARs 0-2:
  - BAR0 iATU maps 256MB to a base address of 0x4000_0000, which is the base of CCSR space. This BAR also encompasses LLCP and MBIST targets.
  - BAR1 iATU maps 128KB to a base of 0x1F80_0000, which is the base of TCM Code memory.
  - BAR2 iATU maps 8MB to a base of 0x2000_0000, which is the base of TCM System (data) memory. This BAR also encompasses VSPA DMEM.
  - ROM configures PCIe MSI Capability Structure “Multiple Message Capable” field to 3’b011 to advertise 8 MSI capability.
  - ROM configures Device ID and Revision Number. See **Chapter 18** PHY-Timer/Comparator, PPS_OUT interrupt requires setting Cross Trigger Enable (CTE) for corresponding channel of PHY Timer and configuring NVIC for pulse input.
  - ROM configures Max Frequency based on POR Config input CFG_PCIE_GEN by configuring the Link Control 2 register and the Link Capabilities 2 register.
  - ROM then sets the Link Training Enable (LTSSM_EN) bit in the PEX_PF0_Config register to enable training at the specified frequency.
  - ROM code then sets the PCIe CFG_READY bit to allow inbound configuration transactions from the host. (Before CFG_READY is set inbound configuration transactions are retried
  - The host will then perform discovery and enumeration functions for the PCIe link, including assigning a base PCI address for each BAR.
  - The host sets up an outbound iATU in LA9310 to map the PCIe address range in LA9310’s local SoC address map to a PCIe address which will in turn be mapped by the hosts inbound address translation mechanism to the base of the host memory given to LA9310.
  - The host will then write the 8 words of “Boot Header” to the first 8 words of TCM Data Memory. See Boot Overview above.

- ROM code polls the contents of offset 0x0 from the TCM Data Memory base address, 0x2000_0000 for a valid Boot Header Preamble.
- When ROM finds a valid Boot Header it reads in subsequent 7 words to find size and offset of the Plugin Routine and Boot Loader from the PCIe base address.
- If the Plugin Routine size is nonzero:
  - ROM code sets up an eDMA transfer to copy the Plugin Routine from host memory and write it to the base of TCM Code memory at local address 0x1F80_0000.
  - When the copy is complete, ROM code makes a subroutine call to TCM Code memory at 0x1F80_0000.
  - The Plugin Routine executes (e.g. programming the RFIC and configuring a valid clock to support completion of the reset sequence) then returns to the ROM code.
- If header.flags[16]=0 (no “Early_Exit”):
  - ROM code handshakes with the Reset FSM allowing hardware to complete the reset sequence all the way to the SYSREADY state
  - ROM code waits for reset sequence to complete (either through polling the Reset FSM state or by configuring EPU to generate RXEV event upon Reset FSM reaching SYSREADY then entering a WFE loop - based on CFG_SYSRDY_EVT input).
  - ROM code sets up an eDMA transfer to copy the Boot Loader from host memory using the size and adding the offset from the base of local PCIe address space, 0xA000_0000. The data is stored at the location in TCM memory specified by BL_DEST.
  - When the copy is complete, code jumps from ROM to the Boot Loader at the BL_ENTRY location.

5.6 BootROM vs RTOS for Reset Sequence Handshake

For boot from I2C or PCIe, the Reset FSM pauses for a handshake with software before locking the internal PLL to the DCS_CLK input. This allows software to fully configure the source of the DCS clock. See HW/SW Handshake stage of Reset Sequence in section **4.5 Power-On Reset Detailed Sequence**

Two mechanisms are supported for performing this handshake:

A boot plugin routine can do necessary configuration and then return control to the BootROM to perform the handshake with hardware.

Alternately, a flag in the Boot Header can force the BootROM to skip the handshake and immediately download bootloader or RTOS code and jump to the downloaded code. In this case the downloaded code takes the responsibility for completing the handshake with the Reset FSM and determining when the Reset FSM reaches the SYSREADY state. See “Boot Header Flags”

The later method, allowing an RTOS to be in control of the Reset Sequence Handshake, is supported in order to let the RTOS communicate with a Linux-based driver running on the PCIe host to configure the source of the reference clock (DCS_CLK).

Chapter 6
Interrupt Assignments

6.1 Introduction

This chapter describes the interrupt assignments for the chip. These are the on-chip interrupt sources from peripheral logic within the integrated device.

6.2 Interrupt Assignments

The table below shows Interrupt assignments for the chip.

Interrupt Number	Assignment	Comment
0	GPIO	
1	I2C1	
2	I2C2	
3	PCIe	
4	SPI1	
5-8	Reserved	
9	eDMA	
10	MSG1	
11	MSG2	
12	MSG3	
13	WatchDog	
14	UART	
15	Address Error Manager	
16	MBEE	Multi-bit ECC error
17	AXIQ	Overflow/Underrun Error Interrupt
18	ADC/DAC Data Conversion System	
19	VSPA	Multiple interrupts combined and converted to active high interrupts
20	TMU Thermal Alarm	
21	TMU Thermal Critical Alarm	
22	Reserved	
23	Pulse Per Second in	PPS_IN from pin
24	Pulse Per Second Out	PPS_OUT from PHY Timer [1]
25 - 31	Reserved	

1. PPS_OUT interrupt requires setting Cross Trigger Enable (CTE) for corresponding channel of PHY Timer and configuring NVIC for pulse input.

Chapter 7
Arm Modules

7.1 Introduction

The chip includes the following ARM modules:

- ARM® Cortex®-M4 core
- CoreLink™ Network Interconnect NIC-301
- Watchdog Timer Unit

For more information on these modules, see the ARM documentation that accompanies this reference manual.

Table 9. Related resources from ARM

Resource	IP Revision
ARM® Cortex™-M4 Processor Technical Reference Manual	r0p1
CoreLink™ Network Interconnect NIC-301 Technical Reference Manual	r2p3

Watchdog Timer Unit is included in this Reference Manual. Please refer The WDOG Timer module as implemented on the chip

7.2 ARM® Cortex®-M4

The chip has a single Cortex-M4 processor running at 307.2 MHz.

The processor implements these features.

- A Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low latency interrupt processing.
- Multiple high-performance bus interfaces.
- The processor has the following external interfaces: Multiple memory and device bus interfaces. Debug port interface.
- 128KB Tightly Coupled Code Memory (TCML for “lower” TCM address range) and a 64 KB Tightly Coupled Data Memory (TCMU for “upper” TCM address range). Both of these memories can be accessed by external masters connected to the NIC301.
- A bus dedicated to instruction fetches of the Code memory, and a bus dedicated to Load/Store accesses to the Code memory. These buses are muxed together to access the code memory array. The M4 also has a system bus for accesses to all memories .

7.3 CoreLink™ Network Interconnect NIC-301

CoreLink™ Network Interconnect NIC-301 is the primary interconnect on the chip. The various IPs and external I/Os communicate through a complete high performance, optimized AMBA-compliant network infrastructure (NIC-301).

Features contains eDMA, Cortex M4, PCIe and test port to the network.

- Contains a 614.4 MHz switch that connects the VSPA complex to the network.
- VSPA communicates directly through the NIC to the AXIQ in the 614.4 MHz domain.
- All CCSR registers are accessible through APB targets of the NIC.
- The LLCP master updates from VSPA and communicate directly through the NIC in the 614.4 MHz domain.
- The NIC contains elastic buffering to mitigate stalls because of arbitration.

Table 10. LA9310 Connectivity

	Target						
Master	TCM	VSPA D-Mem	AXIQ	PCIe Host	RFIC I/F (LLCP)	CCSR	ROM
M4 Code							yes
M4 System		yes		yes	yes	yes	
VSPA_DMA	yes	yes	yes	yes	yes	yes	
PCIe_Mstr	yes	yes			yes	yes	
eDMA	yes	yes		yes	yes	yes	

Chapter 8
Device Configuration and Pin Control

8.1 Introduction

The Device Configuration Unit provides general purpose configuration and status for the device.

8.3 JTAG IDCODE

JTAG IDCODE for LA9310 is 0x0008_201D.

8.5 Introduction

The Device Configuration Unit (DCFG), provides general purpose configuration and status for the device.

8.5.1 Features

The device configuration unit features the following:

- Pin sampling of device configuration pins at power-on reset and a corresponding POR status register for capturing the values of these configuration pins
- Core and device disable registers used for gating off clocks for any IP blocks or cores which are not used at all by an application
- Two small sets of scratch registers:
  - One set of read / write scratch registers
  - One set of write-once / read scratch registers

8.6 Memory Map and Register Definition

This section describes the memory map for the Config Unit including the memory map tables.

8.6.1 Device Config register descriptions

The tables below shows the memory-mapped CCSR of the Device Configuration module and lists the offset, name, and a cross-reference to the complete description of each register. These registers only support 32-bit accesses.

8.6.1.1 DCFG_ccsr Memory map

DCFG base address: 1E0_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	POR Status Register 1 (PORSR1)	32	RO	See description.
4h	POR Status Register 2 (PORSR2)	32	RO	See description.
70h	Device Disable Register 1 (DEVDISR1)	32	RW	0040_0000h
78h	Device Disable Register 3 (DEVDISR3)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
7Ch	Device Disable Register 4 (DEVDISR4)	32	RW	0000_0000h
80h	Device Disable Register 5 (DEVDISR5)	32	RW	0000_0000h
A4h	System Version Register (SVR)	32	RO	8141_1010h
200h	Scratch Read / Write Register n (SCRATCHRW1)	32	RW	0000_0000h
204h	Scratch Read / Write Register n (SCRATCHRW2)	32	RW	0000_0000h
208h	Scratch Read / Write Register n (SCRATCHRW3)	32	RW	0000_0000h
20Ch	Scratch Read / Write Register n (SCRATCHRW4)	32	RW	0000_0000h
210h	Scratch Read / Write Register n (SCRATCHRW5)	32	RW	0000_0000h
214h	Scratch Read / Write Register n (SCRATCHRW6)	32	RW	0000_0000h
218h	Scratch Read / Write Register n (SCRATCHRW7)	32	RW	0000_0000h
21Ch	Scratch Read / Write Register n (SCRATCHRW8)	32	RW	0000_0000h
220h	Scratch Read / Write Register n (SCRATCHRW9)	32	RW	0000_0000h
224h	Scratch Read / Write Register n (SCRATCHRW10)	32	RW	0000_0000h
228h	Scratch Read / Write Register n (SCRATCHRW11)	32	RW	0000_0000h
22Ch	Scratch Read / Write Register n (SCRATCHRW12)	32	RW	0000_0000h
230h	Scratch Read / Write Register n (SCRATCHRW13)	32	RW	0000_0000h
234h	Scratch Read / Write Register n (SCRATCHRW14)	32	RW	0000_0000h
238h	Scratch Read / Write Register n (SCRATCHRW15)	32	RW	0000_0000h
23Ch	Scratch Read / Write Register n (SCRATCHRW16)	32	RW	0000_0000h
240h	Scratch Read / Write Register n (SCRATCHRW17)	32	RW	0000_0000h
244h	Scratch Read / Write Register n (SCRATCHRW18)	32	RW	0000_0000h
248h	Scratch Read / Write Register n (SCRATCHRW19)	32	RW	0000_0000h
24Ch	Scratch Read / Write Register n (SCRATCHRW20)	32	RW	0000_0000h
250h	Scratch Read / Write Register n (SCRATCHRW21)	32	RW	0000_0000h
254h	Scratch Read / Write Register n (SCRATCHRW22)	32	RW	0000_0000h
258h	Scratch Read / Write Register n (SCRATCHRW23)	32	RW	0000_0000h
25Ch	Scratch Read / Write Register n (SCRATCHRW24)	32	RW	0000_0000h
260h	Scratch Read / Write Register n (SCRATCHRW25)	32	RW	0000_0000h
264h	Scratch Read / Write Register n (SCRATCHRW26)	32	RW	0000_0000h
268h	Scratch Read / Write Register n (SCRATCHRW27)	32	RW	0000_0000h
26Ch	Scratch Read / Write Register n (SCRATCHRW28)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
270h	Scratch Read / Write Register n (SCRATCHRW29)	32	RW	0000_0000h
274h	Scratch Read / Write Register n (SCRATCHRW30)	32	RW	0000_0000h
278h	Scratch Read / Write Register n (SCRATCHRW31)	32	RW	0000_0000h
27Ch	Scratch Read / Write Register n (SCRATCHRW32)	32	RW	0000_0000h
300h	Scratch Read Register n (SCRATCHW1R1)	32	WONCE	0000_0000h
304h	Scratch Read Register n (SCRATCHW1R2)	32	WONCE	0000_0000h
308h	Scratch Read Register n (SCRATCHW1R3)	32	WONCE	0000_0000h
30Ch	Scratch Read Register n (SCRATCHW1R4)	32	WONCE	0000_0000h
620h	General Control Register (GENCR1)	32	RW	2000_0000h
BF8h	IP Block Revision Register 1 (IPBRR1)	32	RO	0280_0600h
BFCh	IP Block Revision Register 2 (IPBRR2)	32	RO	0000_0000h

8.6.1.2 POR Status Register 1 (PORSR1)

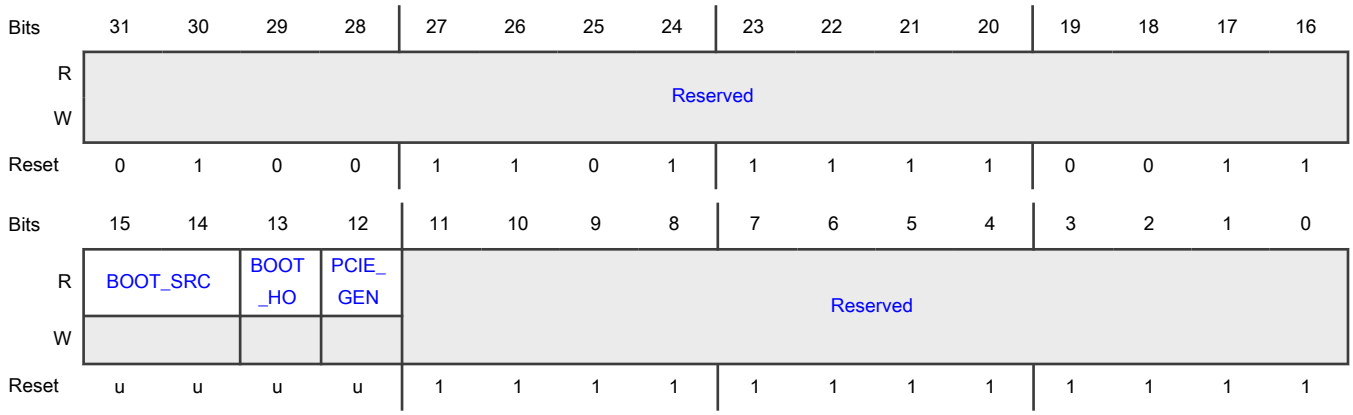
Offset

Register	Offset
PORSR1	0h

Function

PORSR1 captures the values of the device's POR configuration pins.

Diagram



Fields

Field	Function
31-16 —	Reserved.
15-14 BOOT_SRC	Boot Source 00b - Reserved 01b - Reserved 10b - I2C EEPROM 11b - PCIe Host Memory
13 BOOT_HO	Boot Holdoff 0b - Boot of M4 is held off until an external agent writes the Boot Release Register in the Reset block 1b - Boot of M4 core proceeds immediately upon reset state machine reaching SYSREADY
12 PCIE_GEN	PCIe Generation 0b - PCIe interface will be restricted to PCIe Gen2 frequencies 1b - PCIe interface will allow negotiation to PCIe Gen3 frequencies
11-0 —	Reserved.

8.6.1.3 POR Status Register 2 (PORSR2)

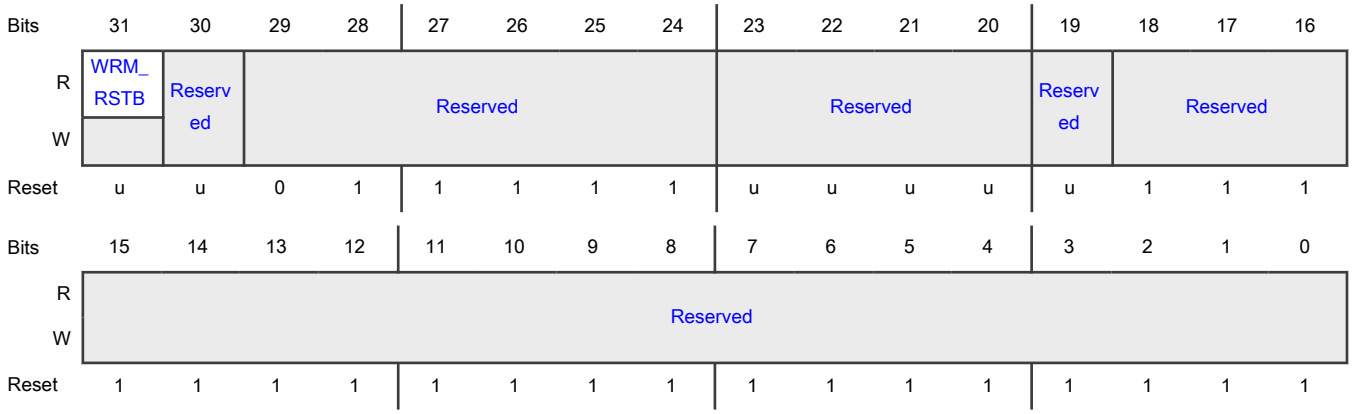
Offset

Register	Offset
PORSR2	4h

Function

PORSR2 captures the values of the device's POR configuration pins.

Diagram



Fields

Field	Function
31 WRM_RSTB	Warm Reset (active low) (consumed by BootROM only) 0b - Reset has been asserted after a functional mode has been reached previously (and specifically after memories have been initialized). Power was not removed prior to the reset assertion. Memory initialization steps may be skipped. 1b - After reset. Memories must be initialized.
30 —	Reserved.
29-24 —	Reserved.
23-20 —	Reserved.
19 —	Reserved.
18-0 —	Reserved.

8.6.1.4 Device Disable Register 1 (DEVDISR1)

Offset

Register	Offset
DEVDISR1	70h

Function

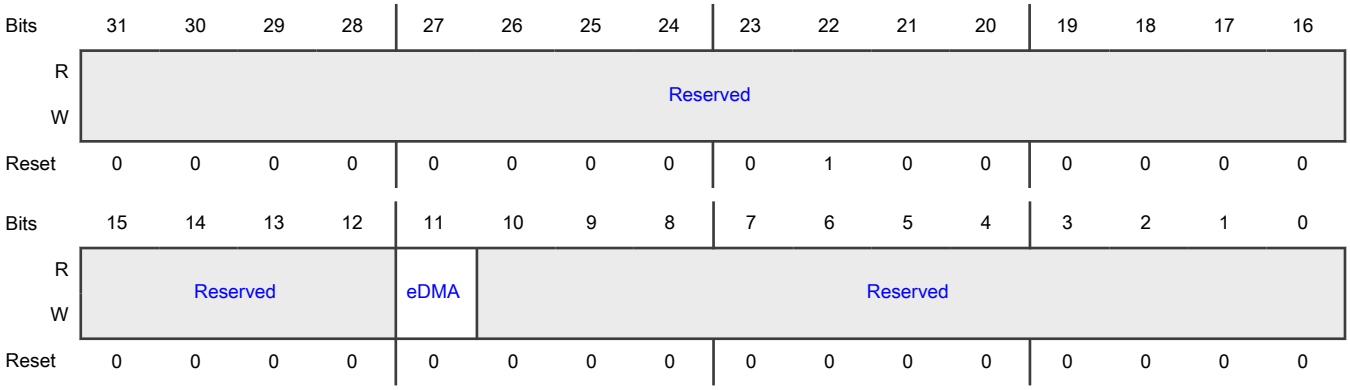
Device Disable Register n

A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISRn provides a mechanism for gating clocks to IP blocks that are not used when running an application.

NOTE

IP Blocks disabled by setting the corresponding bit in the DEVDISRn register must not be re-enabled via software. The only supported mechanism for re-enabling an IP block disabled in this manner is via power-on or hard reset.

Diagram



Fields

Field	Function
31-12 —	Reserved.
11 eDMA	eDMA 0b - Module is enabled 1b - Module is disabled
10-0 —	Reserved.

8.6.1.5 Device Disable Register 3 (DEVDISR3)

Offset

Register	Offset
DEVDISR3	78h

Function

Device Disable Register n

A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISRn provides a mechanism for gating clocks to IP blocks that are not used when running an application.

NOTE

IP Blocks disabled by setting the corresponding bit in the DEVDISRn register must not be re-enabled via software. The only supported mechanism for re-enabling an IP block disabled in this manner is via power-on or hard reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved									LLCP	VSPA	DCS	Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Fields

Field	Function
31-23 —	Reserved
22 LLCP	LLCP 0b - Module is enabled 1b - Module is disabled
21 VSPA	VSPA 0b - Module is enabled 1b - Module is disabled
20 DCS	DCS 0b - Module is enabled 1b - Module is disabled
19-1 —	Reserved.
0 PCIe	PCIe 0b - Module is enabled 1b - Module is disabled

8.6.1.6 Device Disable Register 4 (DEVDISR4)

Offset

Register	Offset
DEVDISR4	7Ch

Function

Device Disable Register n

A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISRn provides a mechanism for gating clocks to IP blocks that are not used when running an application.

NOTE

IP Blocks disabled by setting the corresponding bit in the DEVDISRn register must not be re-enabled via software. The only supported mechanism for re-enabling an IP block disabled in this manner is via power-on or hard reset.

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				Reserved				SPI	Reserved			UART	Reserved	I2C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-10 —	Reserved.
9-6 —	Reserved.
5 SPI	SPI 0b - Module is enabled 1b - Module is disabled
4-3 —	Reserved.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 UART	UART 0b - Module is enabled 1b - Module is disabled
1 —	Reserved.
0 I2C	I2C1 0b - Module is enabled 1b - Module is disabled

8.6.1.7 Device Disable Register 5 (DEVDISR5)

Offset

Register	Offset
DEVDISR5	80h

Function

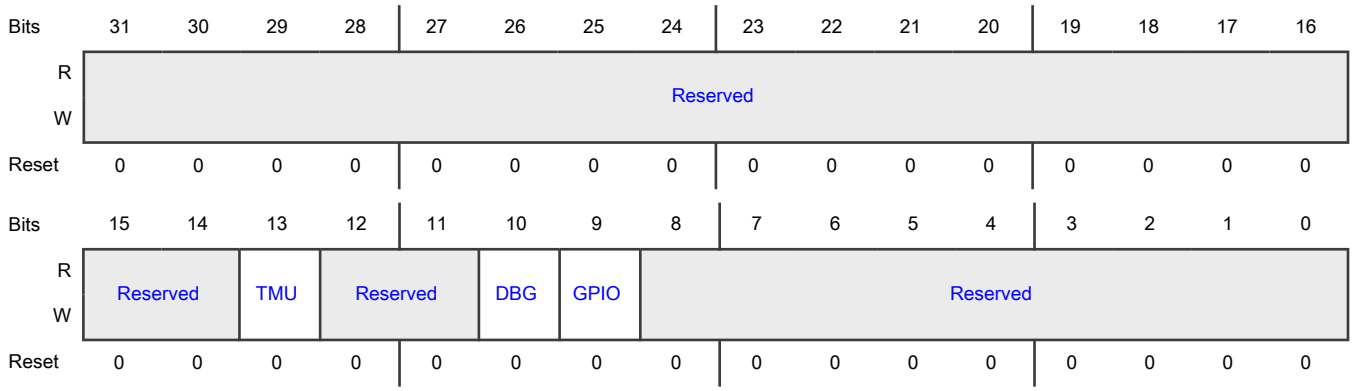
Device Disable Register n

A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISRn provides a mechanism for gating clocks to IP blocks that are not used when running an application.

NOTE

IP Blocks disabled by setting the corresponding bit in the DEVDISRn register must not be re-enabled via software. The only supported mechanism for re-enabling an IP block disabled in this manner is via power-on or hard reset.

Diagram



Fields

Field	Function
31-14 —	Reserved.
13 TMU	TMU 0b - Module is enabled 1b - Module is disabled
12-11 —	Reserved.
10 DBG	DBG 0b - Module is enabled 1b - Module is disabled
9 GPIO	GPIO 0b - Module is enabled 1b - Module is disabled
8-0 —	Reserved.

8.6.1.8 System Version Register (SVR)

Offset

Register	Offset
SVR	A4h

Function

The SVR contains the system version number for the device.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MFR_ID				FAMILY				Reserved				SOC_DEV_ID			
W																
Reset	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		VAR_PER						MAJOR_REV				MINOR_REV			
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0

Fields

Field	Function
31-28 MFR_ID	Manufacturer ID
27-24 FAMILY	Family
23-22 —	Reserved.
21-16 SOC_DEV_ID	SoC Device ID
15-14 —	Reserved.
13-8 VAR_PER	Various Personalities
7-4 MAJOR_REV	Major Revision Number
3-0 MINOR_REV	Minor Revision Number

8.6.1.9 Scratch Read / Write Register n (SCRATCHRW1)

Offset

Register	Offset
SCRATCHRW1	200h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram

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

Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.10 Scratch Read / Write Register n (SCRATCHRW2)

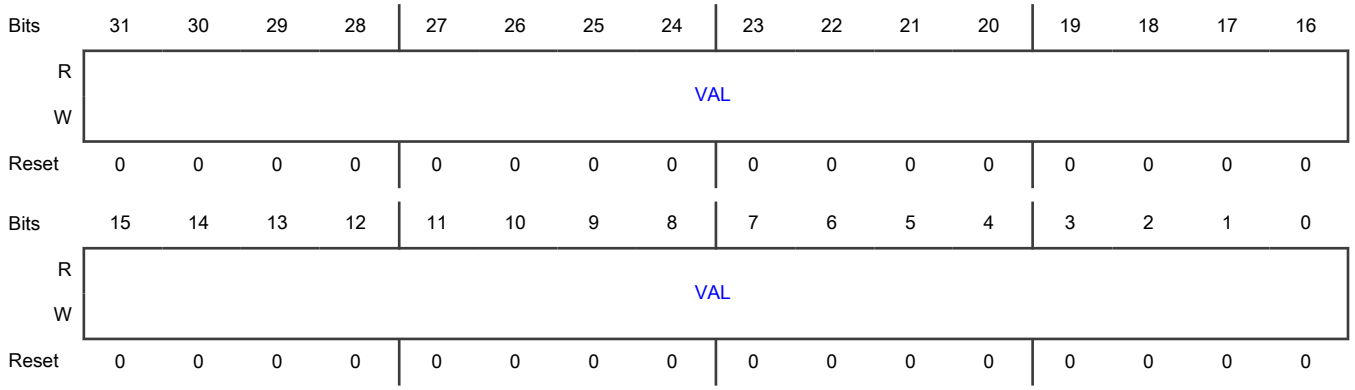
Offset

Register	Offset
SCRATCHRW2	204h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.11 Scratch Read / Write Register n (SCRATCHRW3)

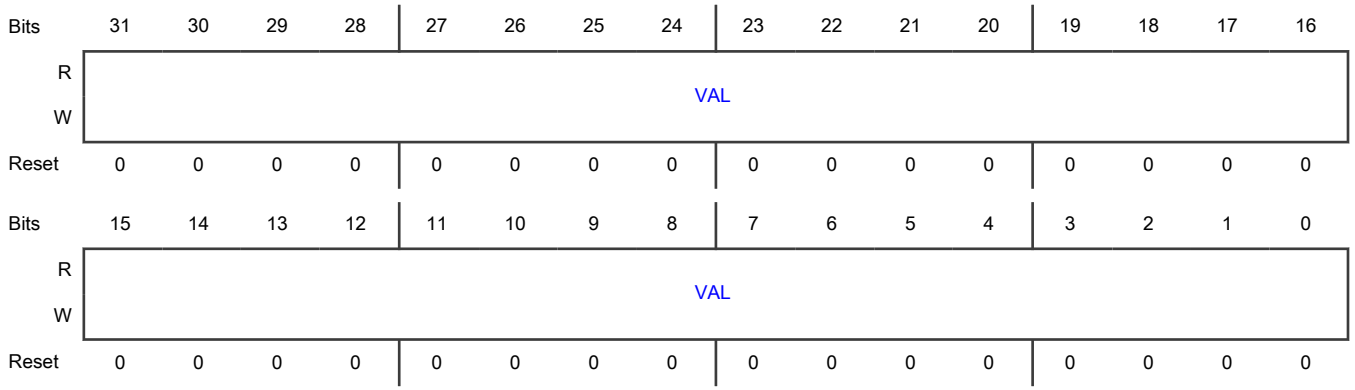
Offset

Register	Offset
SCRATCHRW3	208h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.12 Scratch Read / Write Register n (SCRATCHRW4)

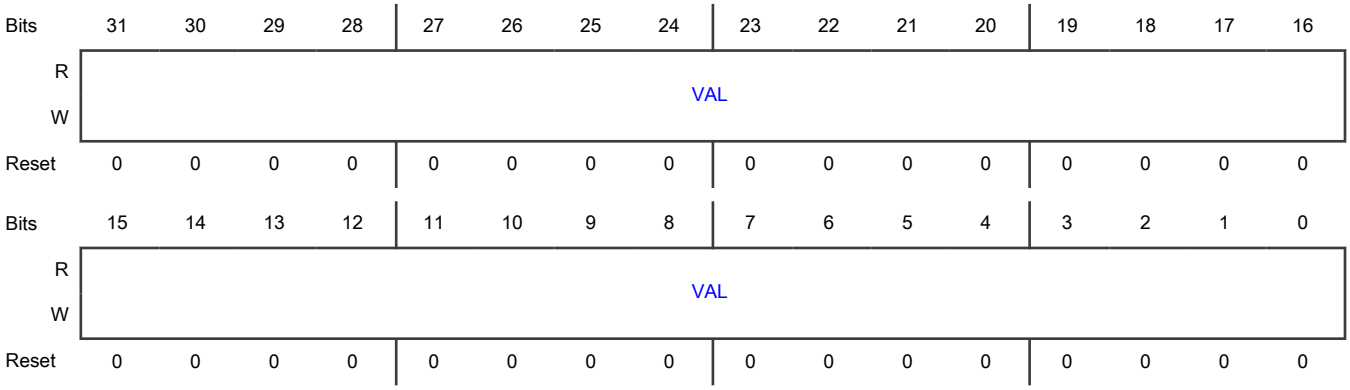
Offset

Register	Offset
SCRATCHRW4	20Ch

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.13 Scratch Read / Write Register n (SCRATCHRW5)

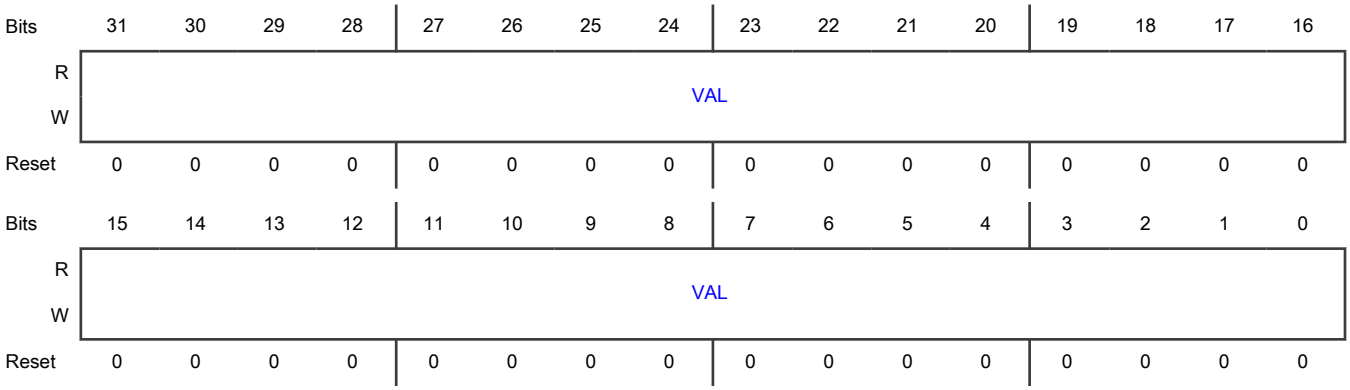
Offset

Register	Offset
SCRATCHRW5	210h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.14 Scratch Read / Write Register n (SCRATCHRW6)

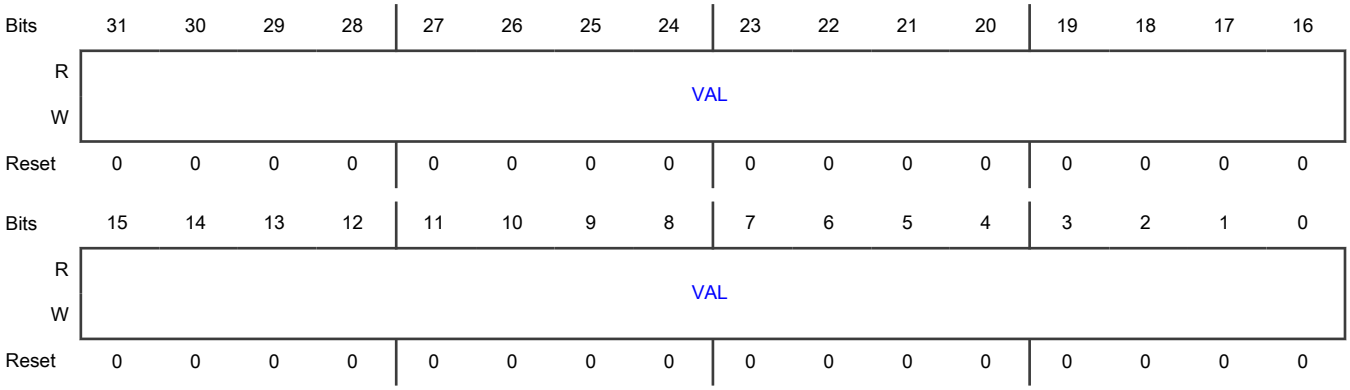
Offset

Register	Offset
SCRATCHRW6	214h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.15 Scratch Read / Write Register n (SCRATCHRW7)

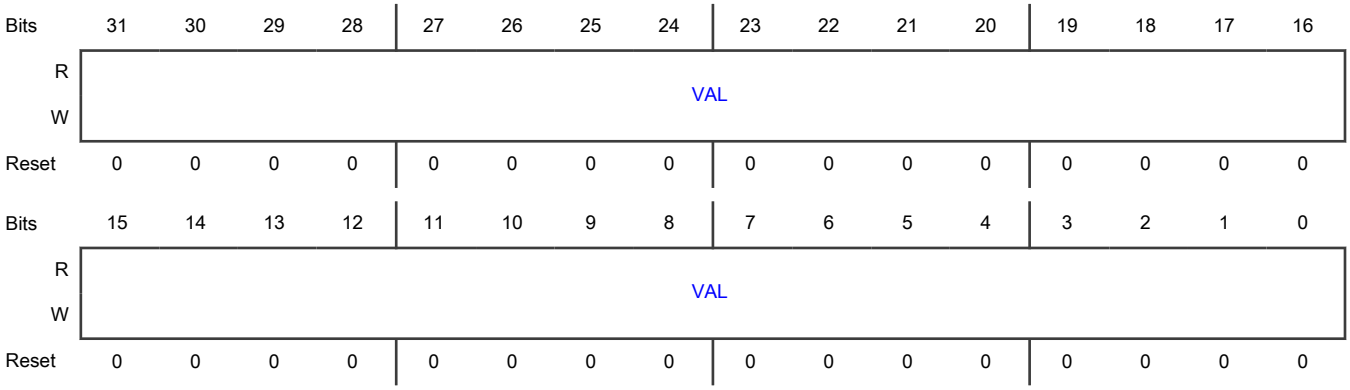
Offset

Register	Offset
SCRATCHRW7	218h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.16 Scratch Read / Write Register n (SCRATCHRW8)

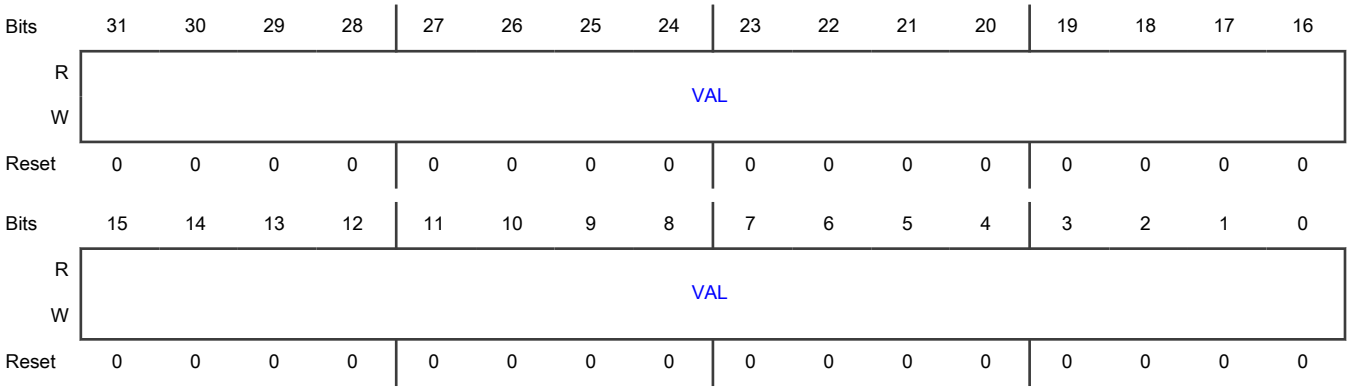
Offset

Register	Offset
SCRATCHRW8	21Ch

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.17 Scratch Read / Write Register n (SCRATCHRW9)

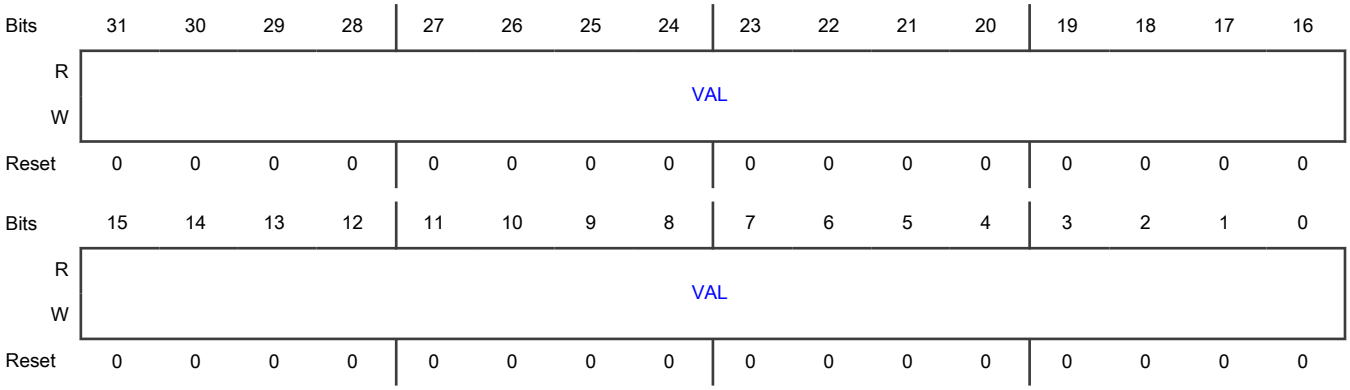
Offset

Register	Offset
SCRATCHRW9	220h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.18 Scratch Read / Write Register n (SCRATCHRW10)

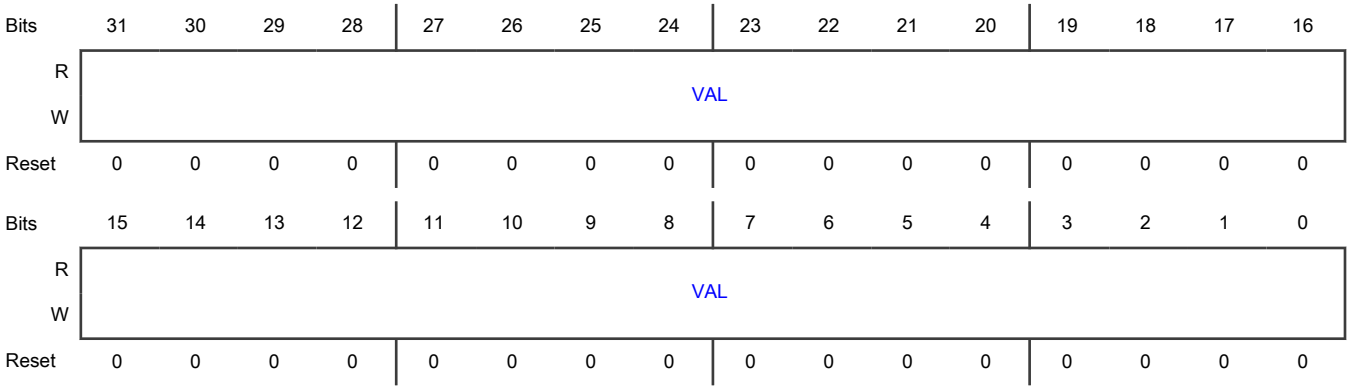
Offset

Register	Offset
SCRATCHRW10	224h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.19 Scratch Read / Write Register n (SCRATCHRW11)

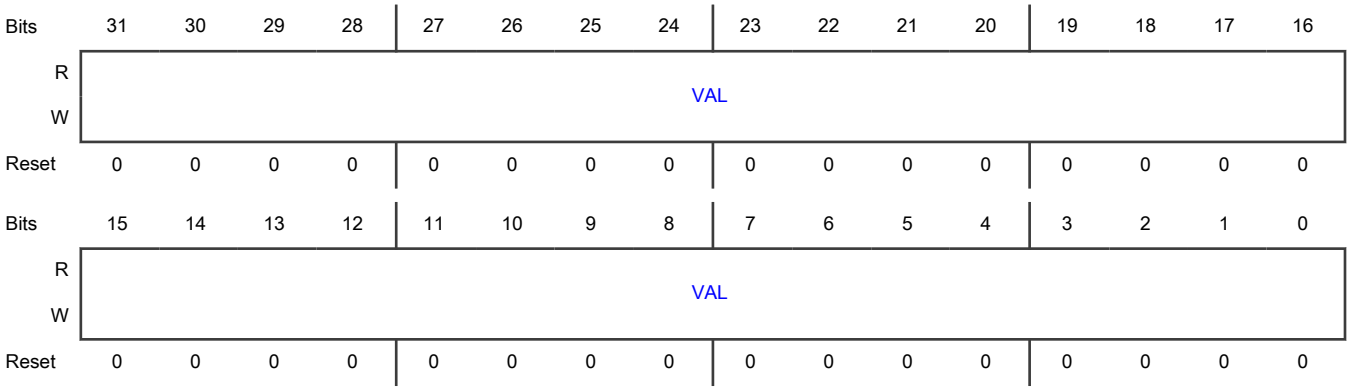
Offset

Register	Offset
SCRATCHRW11	228h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.20 Scratch Read / Write Register n (SCRATCHRW12)

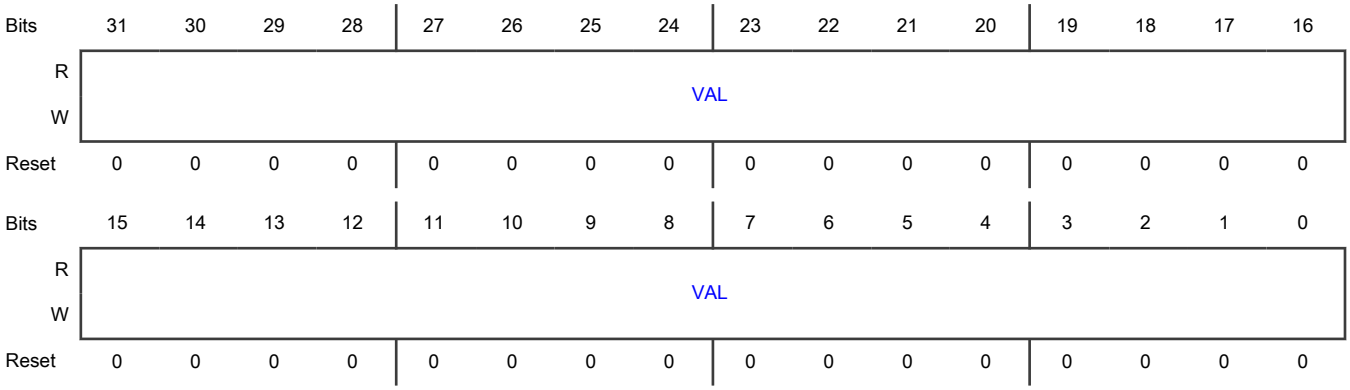
Offset

Register	Offset
SCRATCHRW12	22Ch

Function

Used to supplement the CRSTSRn register for reporting core warm resets.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.21 Scratch Read / Write Register n (SCRATCHRW13)

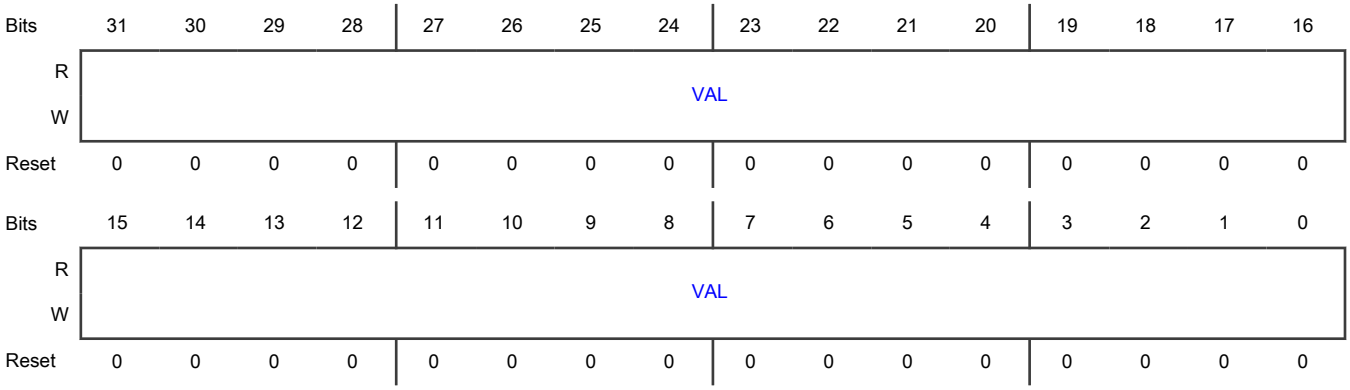
Offset

Register	Offset
SCRATCHRW13	230h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.22 Scratch Read / Write Register n (SCRATCHRW14)

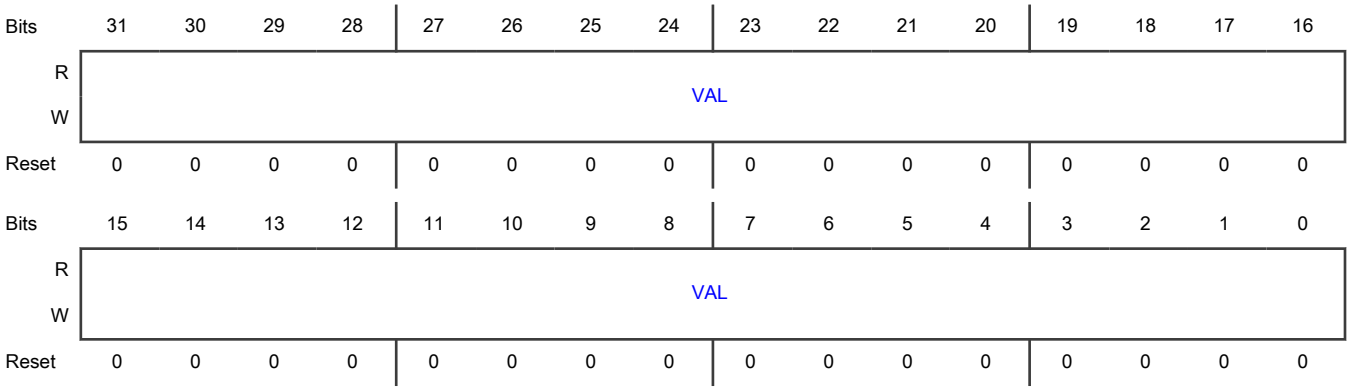
Offset

Register	Offset
SCRATCHRW14	234h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.23 Scratch Read / Write Register n (SCRATCHRW15)

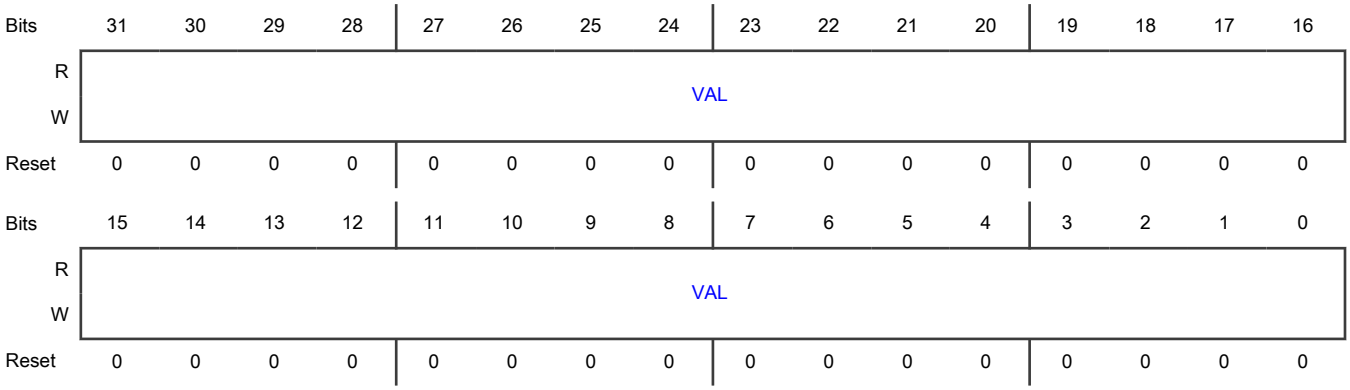
Offset

Register	Offset
SCRATCHRW15	238h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.24 Scratch Read / Write Register n (SCRATCHRW16)

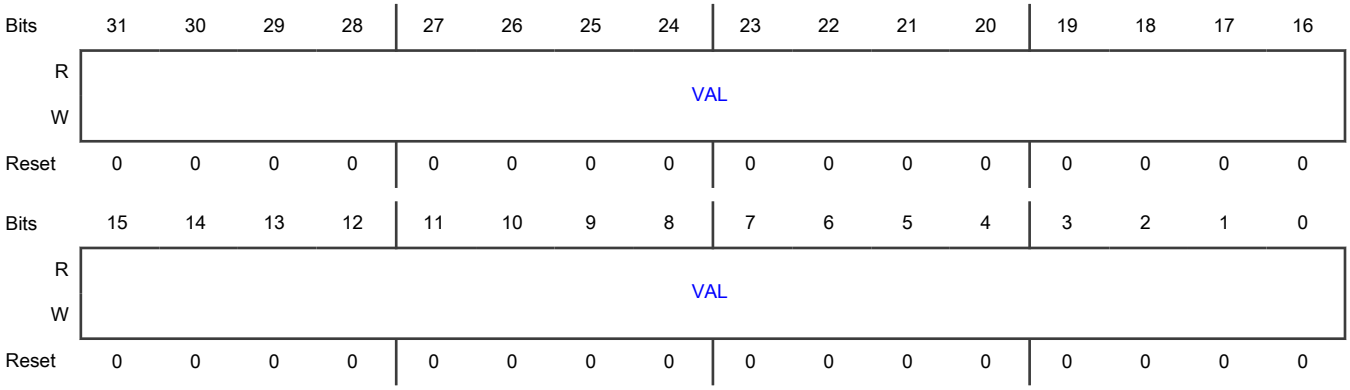
Offset

Register	Offset
SCRATCHRW16	23Ch

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.25 Scratch Read / Write Register n (SCRATCHRW17)

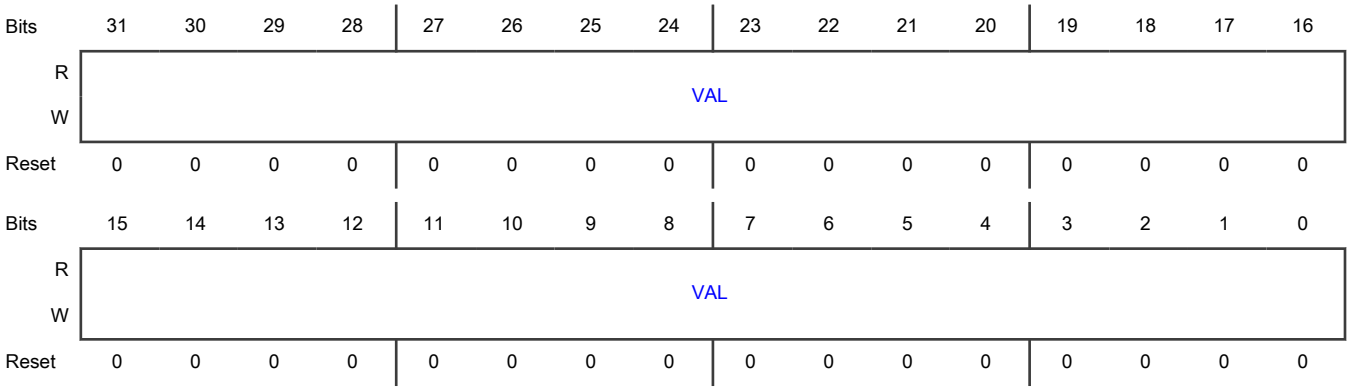
Offset

Register	Offset
SCRATCHRW17	240h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.26 Scratch Read / Write Register n (SCRATCHRW18)

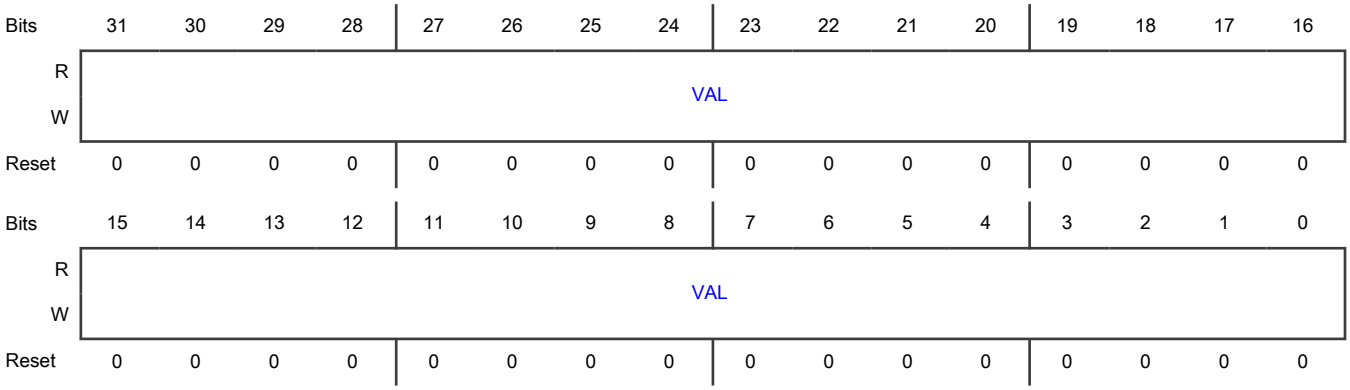
Offset

Register	Offset
SCRATCHRW18	244h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.27 Scratch Read / Write Register n (SCRATCHRW19)

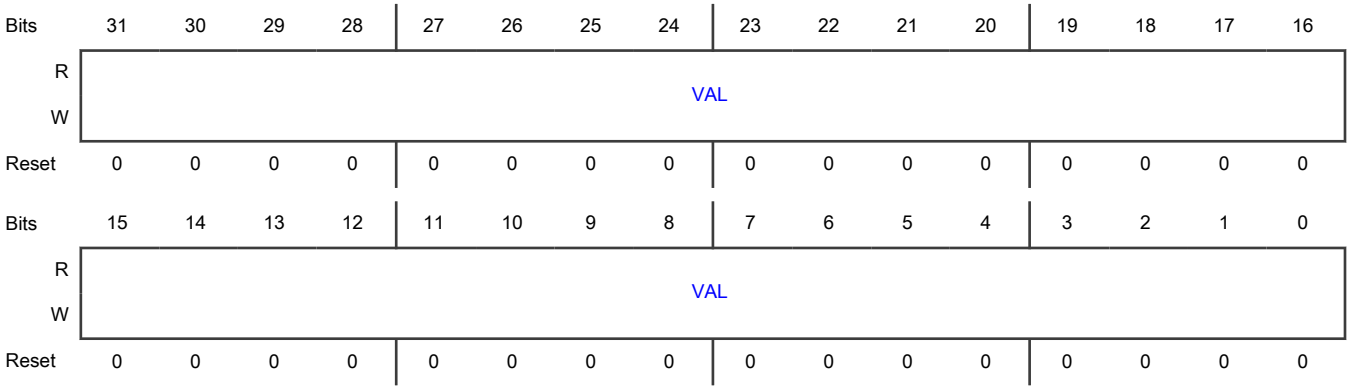
Offset

Register	Offset
SCRATCHRW19	248h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.28 Scratch Read / Write Register n (SCRATCHRW20)

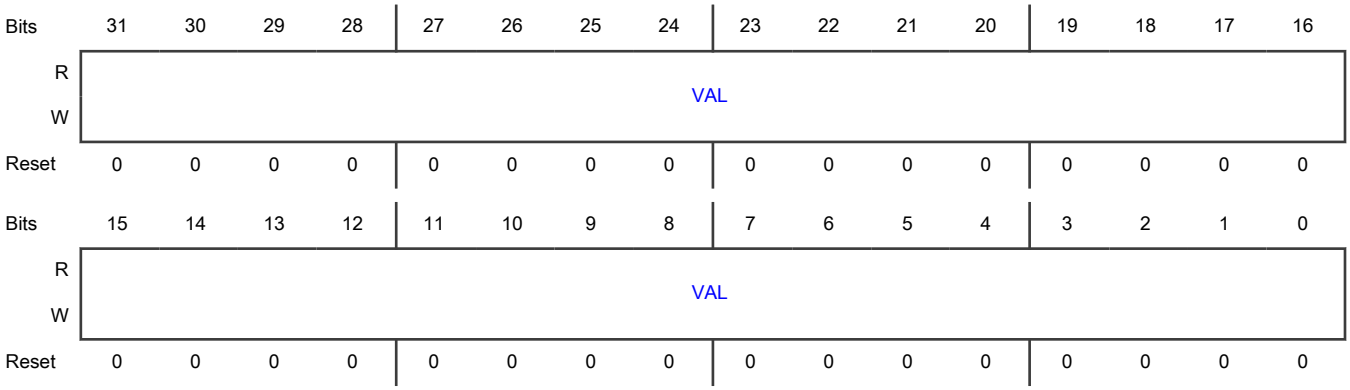
Offset

Register	Offset
SCRATCHRW20	24Ch

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.29 Scratch Read / Write Register n (SCRATCHRW21)

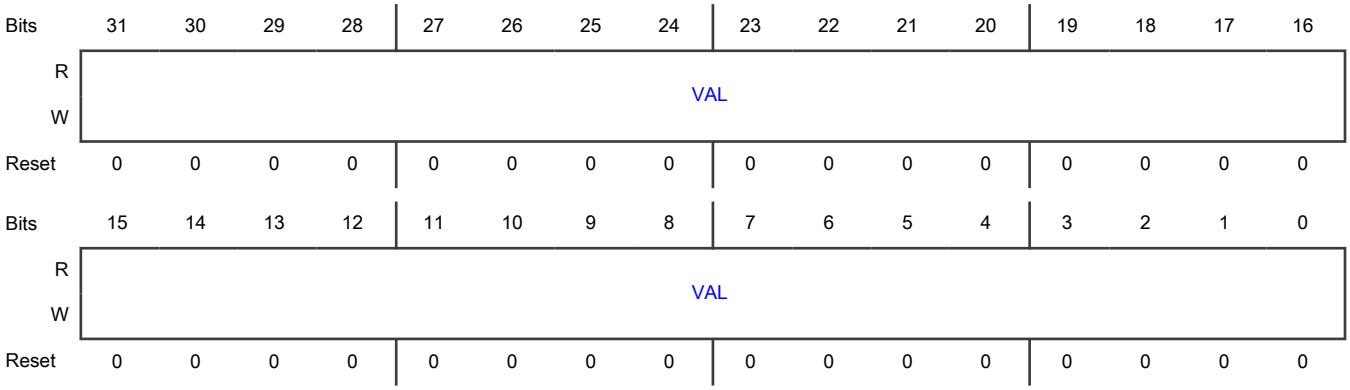
Offset

Register	Offset
SCRATCHRW21	250h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.30 Scratch Read / Write Register n (SCRATCHRW22)

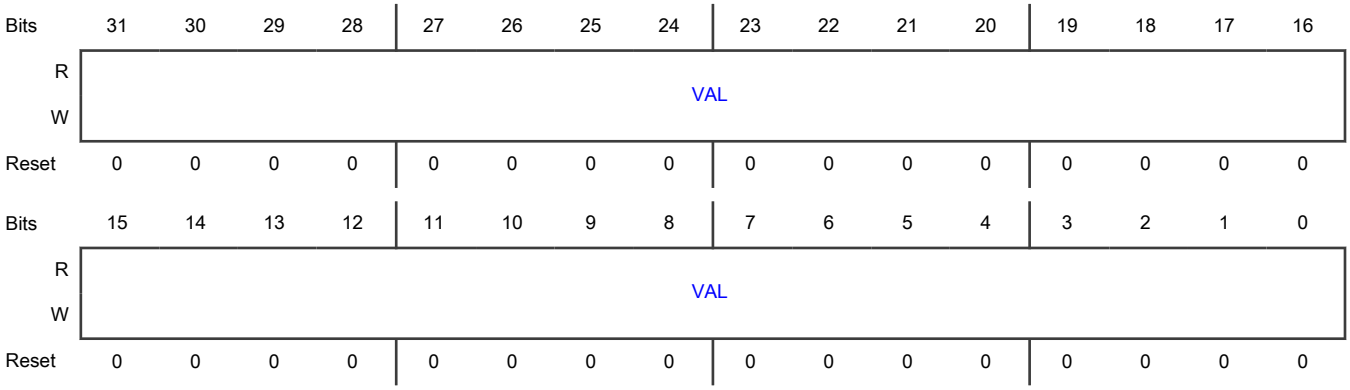
Offset

Register	Offset
SCRATCHRW22	254h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.31 Scratch Read / Write Register n (SCRATCHRW23)

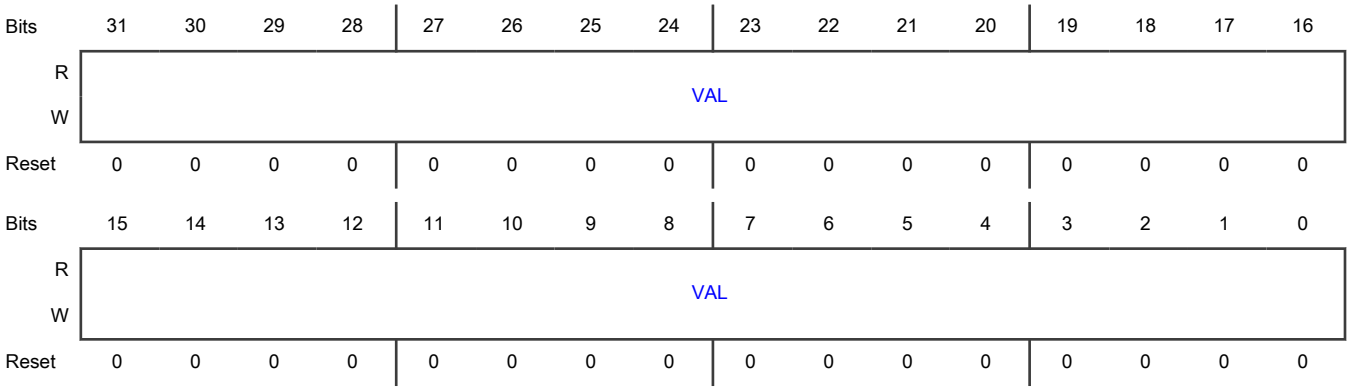
Offset

Register	Offset
SCRATCHRW23	258h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.32 Scratch Read / Write Register n (SCRATCHRW24)

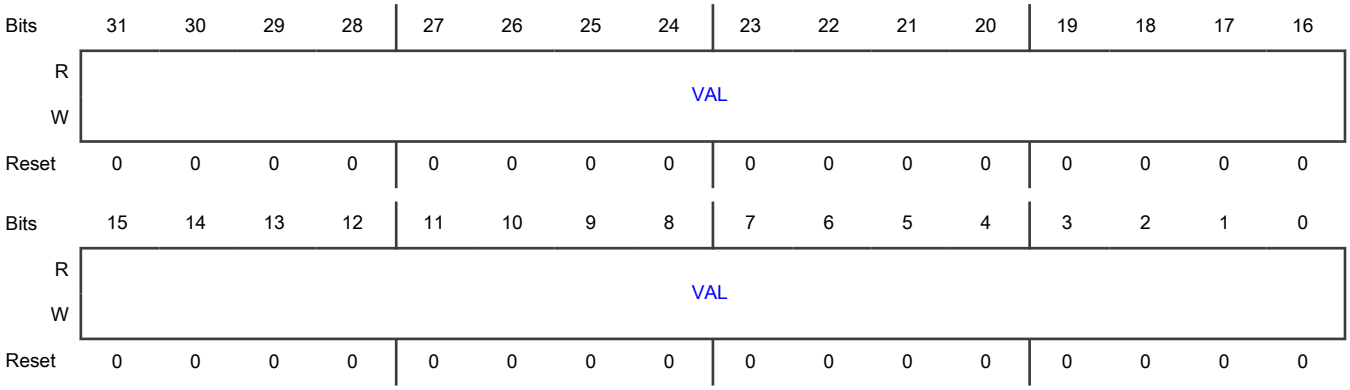
Offset

Register	Offset
SCRATCHRW24	25Ch

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.33 Scratch Read / Write Register n (SCRATCHRW25)

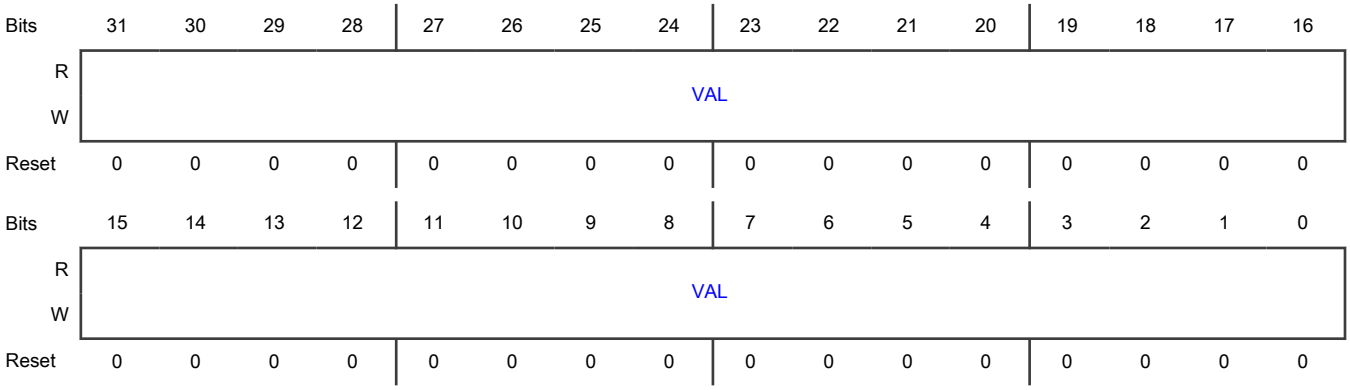
Offset

Register	Offset
SCRATCHRW25	260h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.34 Scratch Read / Write Register n (SCRATCHRW26)

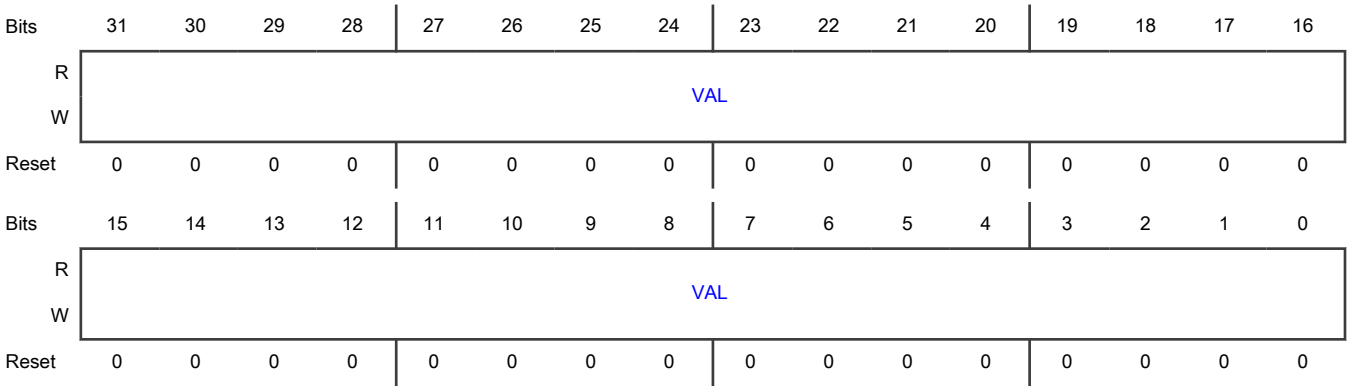
Offset

Register	Offset
SCRATCHRW26	264h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.35 Scratch Read / Write Register n (SCRATCHRW27)

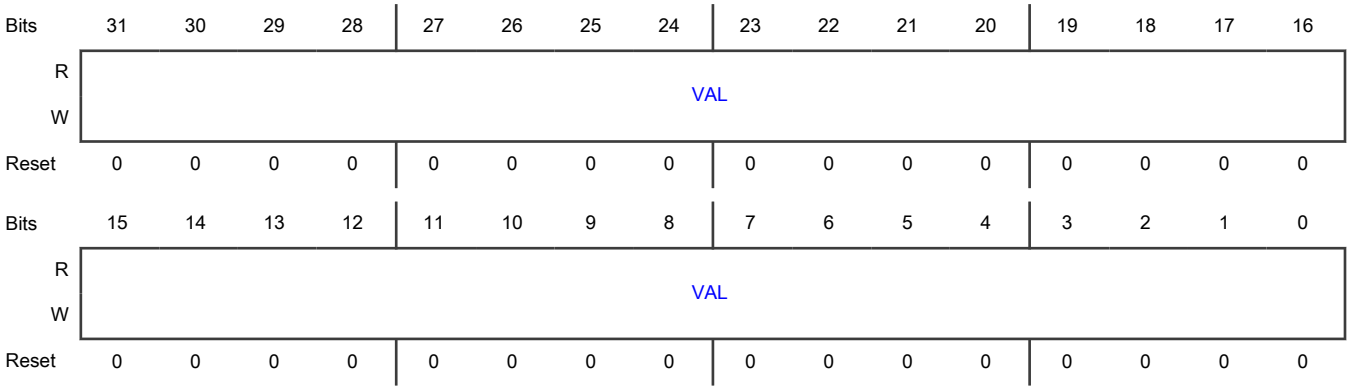
Offset

Register	Offset
SCRATCHRW27	268h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.36 Scratch Read / Write Register n (SCRATCHRW28)

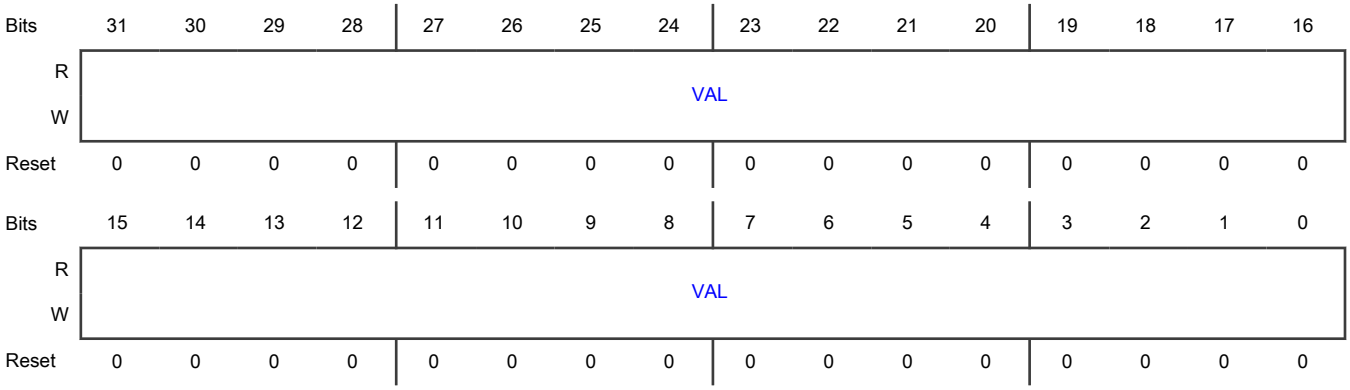
Offset

Register	Offset
SCRATCHRW28	26Ch

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.37 Scratch Read / Write Register n (SCRATCHRW29)

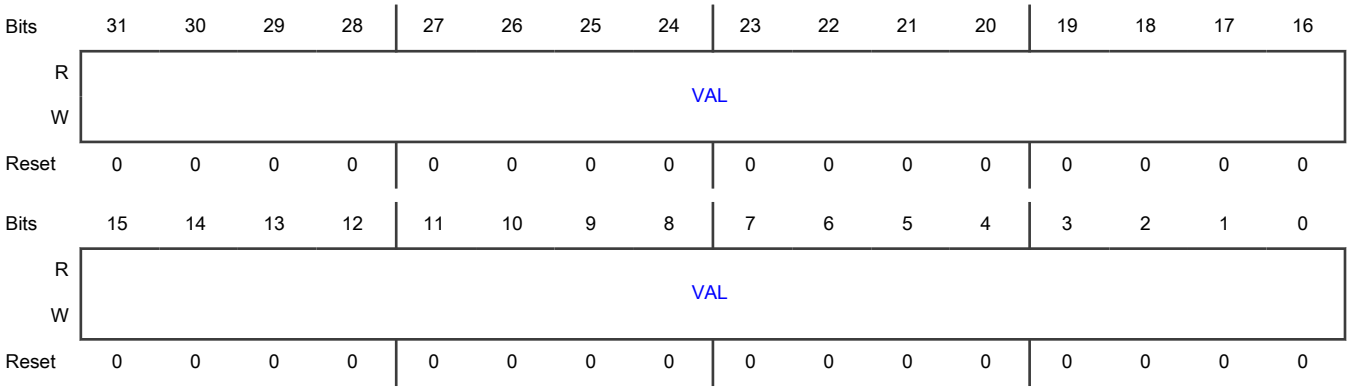
Offset

Register	Offset
SCRATCHRW29	270h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.38 Scratch Read / Write Register n (SCRATCHRW30)

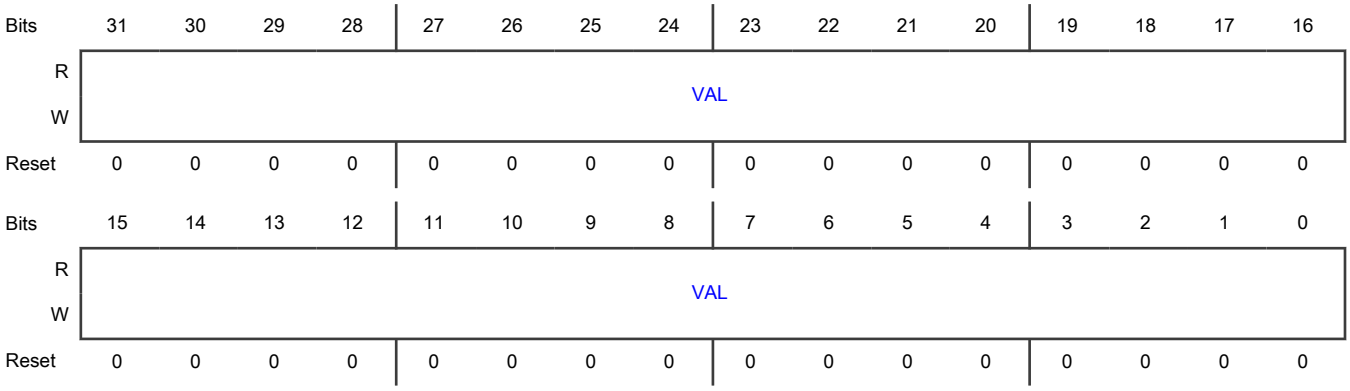
Offset

Register	Offset
SCRATCHRW30	274h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.39 Scratch Read / Write Register n (SCRATCHRW31)

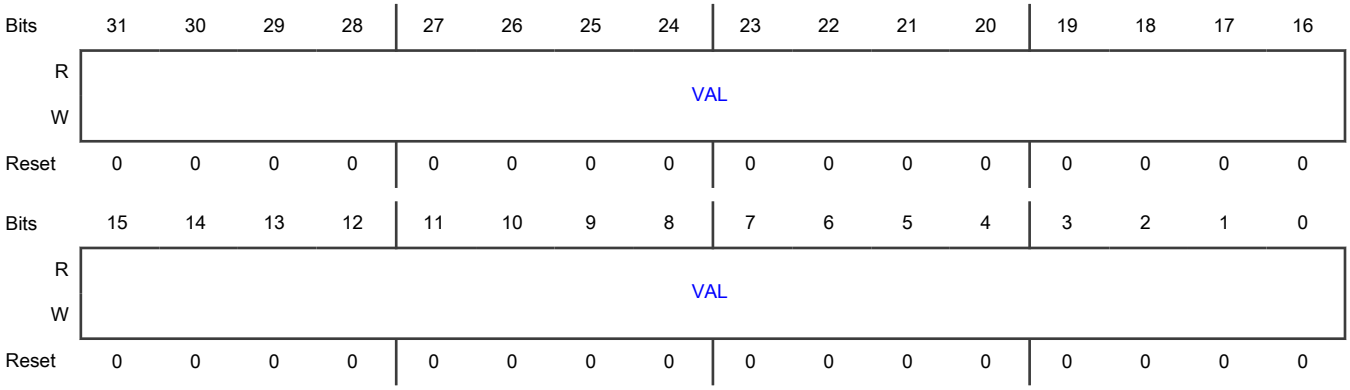
Offset

Register	Offset
SCRATCHRW31	278h

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.40 Scratch Read / Write Register n (SCRATCHRW32)

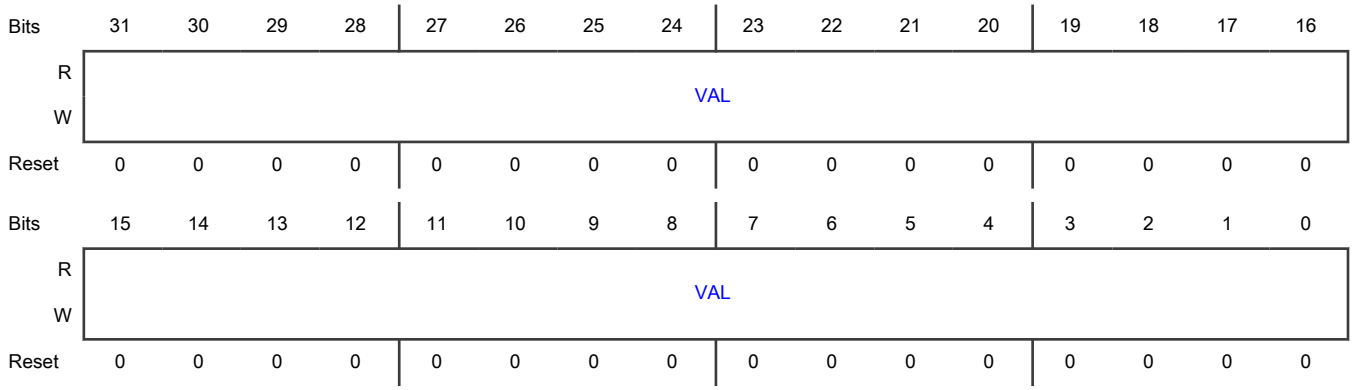
Offset

Register	Offset
SCRATCHRW32	27Ch

Function

The SCRATCHRWn provides read / write scratch register locations available to the user.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.41 Scratch Read Register n (SCRATCHW1R1)

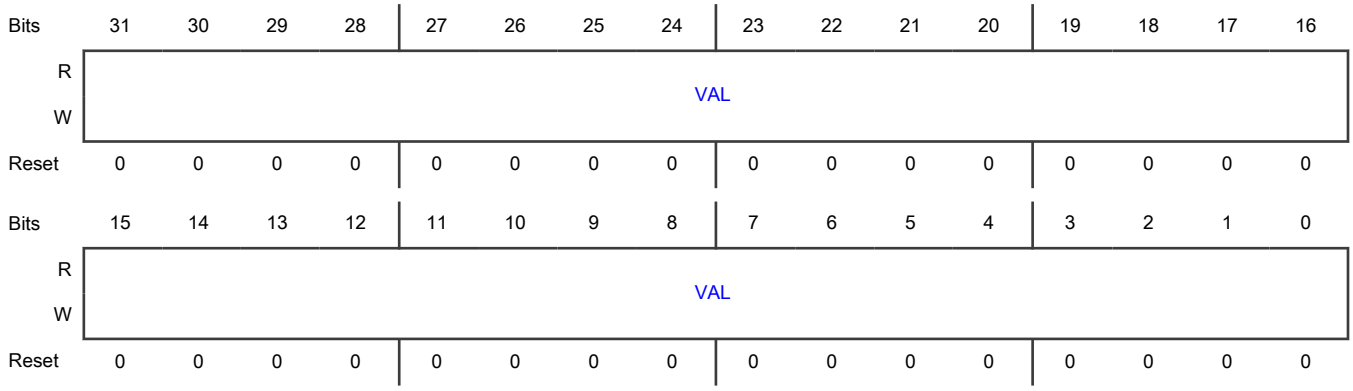
Offset

Register	Offset
SCRATCHW1R1	300h

Function

The SCRATCHW1Rn provides scratch register locations available to the user. These are write-once registers. After these have been written once, they can only be written after a power-on or hard reset.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.42 Scratch Read Register n (SCRATCHW1R2)

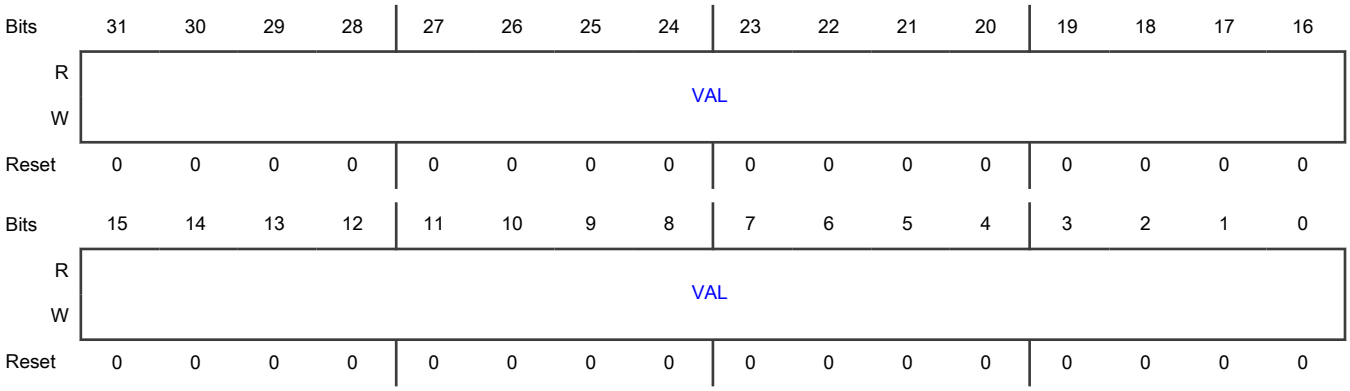
Offset

Register	Offset
SCRATCHW1R2	304h

Function

The SCRATCHW1Rn provides scratch register locations available to the user. These are write-once registers. After these have been written once, they can only be written after a power-on or hard reset.

Diagram



Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.43 Scratch Read Register n (SCRATCHW1R3)

Offset

Register	Offset
SCRATCHW1R3	308h

Function

The SCRATCHW1Rn provides scratch register locations available to the user. These are write-once registers. After these have been written once, they can only be written after a power-on or hard reset.

Diagram

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

Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.44 Scratch Read Register n (SCRATCHW1R4)

Offset

Register	Offset
SCRATCHW1R4	30Ch

Function

The SCRATCHW1Rn provides scratch register locations available to the user. These are write-once registers. After these have been written once, they can only be written after a power-on or hard reset.

Diagram

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

Fields

Field	Function
31-0 VAL	32-bit scratch content

8.6.1.45 General Control Register (GENCR1)

Offset

Register	Offset
GENCR1	620h

Function

The GENCRn provides expansion bits for device control.

NOTE

All 32-bits must be implemented in silicon and routed to terminals at the edge of this block even if unused. This allows for fixing of issues found late in the design cycle.

Diagram

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

Fields

Field	Function
31-16 —	Reserved.
15-12 —	Reserved.
11	Resume Reset Sequence

Table continues on the next page...

Table continued from the previous page...

Field	Function
RRCS	**NOTE** This bit is unused if reset pause feature is disabled. 0b - Reset state machine cannot exit the state which, when reset pause is enabled, waits for updates to be made 1b - Reset state machine can exit the state which, when reset pause is enabled, waits for updates to be made
10 RRS	Resume Reset Sequence **NOTE** This bit is unused if reset pause feature is disabled. 0b - Reset state machine cannot exit the state which, when reset pause is enabled, waits for PLLs to lock and other special reset pause locks to be freed. 1b - Reset state machine can exit the state which, when reset pause is enabled, waits for PLLs to lock and other special reset pause locks to be freed.
9-0 —	Reserved.

8.6.1.46 IP Block Revision Register 1 (IPBRR1)

Offset

Register	Offset
IPBRR1	BF8h

Function

IP block revision register.

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	1	0	1	0	0	0	0	0	0	0

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

Fields

Field	Function
31-16 IP_ID	IP block ID
15-8 IP_MJ	Major revision
7-0 IP_MN	Minor revision

8.6.1.47 IP Block Revision Register 2 (IPBRR2)

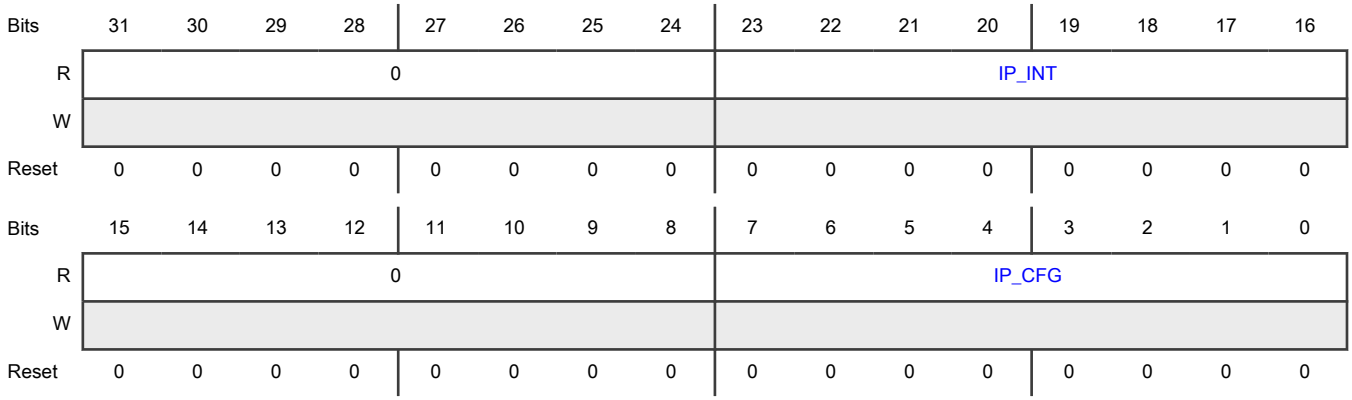
Offset

Register	Offset
IPBRR2	BFCh

Function

IP block revision register.

Diagram



Fields

Field	Function
31-24 —	Reserved.
23-16	IP block integration options. Unused.

Table continues on the next page...

Table continued from the previous page...

Field	Function
IP_INT	
15-8 —	Reserved.
7-0 IP_CFG	IP block configuration options. Unused.

8.7 Specific Pin Connectivity

Pulse-Per-Second input signal connected to:

- PHY timer - causes Timestamp to be latched in a PHY Timer register where it can be queried by M4

Pulse-Per-Second output signal connected to:

- PHY timer - for generation of PPS_OUT based on timer expiry

IRQ input is connected to GPIO[0] which must be configured as an input with interrupt generation enabled.

8.8 Pin Muxing

GPIO functionality exists as an alternate function on the following 14 pins:

- IRQ - (GPIO[0])
- I2C SCL and SDA signals - (GPIO[2] and GPIO[1])
- PPS_IN and PPS_OUT signals - (GPIO[4] and GPIO[3])
- Reserved - (GPIO[6:5])
- RFCTL[5:0] signals - (GPIO[12:7])
- SPI_CS3_B - (GPIO[13])

UART pins for debug mode exist as an alternate function on the following pins:

- SPI_CS2_B (UART - SOUT)
- SPI_CS1_B (UART - SIN)

8.9 Pins register descriptions

The PMUXCR0 and PMUXCR1 registers in the pins block control the function of the pins that have alternate functions.

8.9.1 Pins Memory map

pins base address: 1FF_0000h

Offset	Register	Width (In bits)	Access	Reset value
E00h	PMUXCR0 (PMUXCR0)	32	RW	0000_0000h
E04h	PMUXCR1 (PMUXCR1)	32	RW	0000_0000h

8.9.2 PMUXCR0 (PMUXCR0)

Offset

Register	Offset
PMUXCR0	E00h

Function

PMUXCR0 switches pins from their primary function to GPIO functionality.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved												Reserved			GPIO16
Reset	0	0	0	0	0	0	0	0	0	0	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			GPIO12	GPIO11	GPIO10	GPIO9	GPIO8	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved.
19-17 —	Reserved.
16 GPIO16	Enable GPIO functionality If the bit is set, then GPIO functionality is enabled on the associated pin instead of the primary function of the pin. 0b - SPI_CS3_B 1b - GPIO_16
15-13 —	Reserved.
12 GPIO12	Enable GPIO functionality If the bit is set, then GPIO functionality is enabled on the associated pin instead of the primary function of the pin.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - PA_EN 1b - GPIO_12
11 GPIO11	Enable GPIO functionality If the bit is set, then GPIO functionality is enabled on the associated pin instead of the primary function of the pin. 0b - LNA1_EN 1b - GPIO_11
10 GPIO10	Enable GPIO functionality If the bit is set, then GPIO functionality is enabled on the associated pin instead of the primary function of the pin. 0b - LNA2_EN 1b - GPIO_10
9 GPIO9	Enable GPIO functionality If the bit is set, then GPIO functionality is enabled on the associated pin instead of the primary function of the pin. 0b - LNA3_EN 1b - GPIO_09
8 GPIO8	Enable GPIO functionality If the bit is set, then GPIO functionality is enabled on the associated pin instead of the primary function of the pin. 0b - TXRX1 1b - GPIO_08
7 GPIO7	Enable GPIO functionality If the bit is set, then GPIO functionality is enabled on the associated pin instead of the primary function of the pin. 0b - TXRX0 1b - GPIO_07
6 GPIO6	Enable GPIO functionality This bit must be set, to select GPIO functionality after reset. The primary function of the pin is reserved. 0b - TXRX1 1b - GPIO_06
5 GPIO5	Enable GPIO functionality This bit must be set, to select GPIO functionality after reset. The primary function of the pin is reserved.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Reserved 1b - GPIO_05
4 GPIO4	Enable GPIO functionality If the bit is set, then GPIO functionality is enabled on the associated pin instead of the primary function of the pin. 0b - PPS_IN 1b - GPIO_04
3 GPIO3	Enable GPIO functionality If the bit is set, then GPIO functionality is enabled on the associated pin instead of the primary function of the pin. 0b - PPS_OUT 1b - GPIO_03
2 GPIO2	Enable GPIO functionality If the bit is set, then GPIO functionality is enabled on the associated pin instead of the primary function of the pin. 0b - IIC1_SDA 1b - GPIO_02
1 GPIO1	Enable GPIO functionality If the bit is set, then GPIO functionality is enabled on the associated pin instead of the primary function of the pin. 0b - IIC1_SCL 1b - GPIO_01
0 —	Reserved.

8.9.3 PMUXCR1 (PMUXCR1)

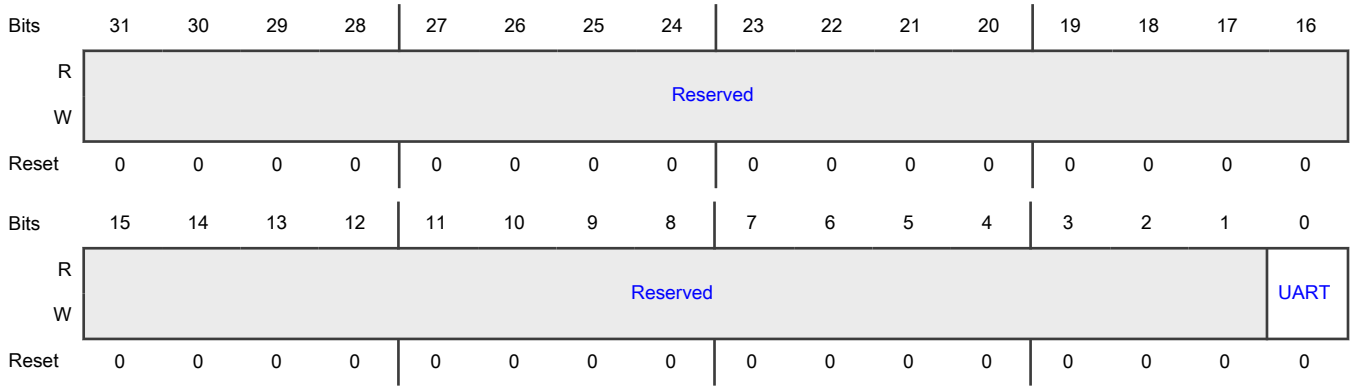
Offset

Register	Offset
PMUXCR1	E04h

Function

PMUXCR1 switches pins from their primary function to UART functionality.

Diagram



Fields

Field	Function
31-1 —	Reserved.
0 UART	Enable UART functionality on SPI_CS1_B and SPI_CS2_B pins If set, SPI_CS1_B becomes UART signal UART1_SIN and SPI_CS2_B becomes UART signal UART1_SOUT.

8.10 AXIQ Loopback Capability

A loop back capability is built around the AXIQ that allows data from the Transmit channel to be fed into any subset of the Receive channels (except the RSSI channel). The loop back capability is controlled via the DBGGENCR register in the Device Configuration block (DCFG), as follows.

8.11 DBGGENCR register descriptions

8.11.1 DBGGENCR Memory map

DBGGENCR base address: 0h

Offset	Register	Width (In bits)	Access	Reset value
8_00ECh	Debug General control (DCFG_DCSR_DBGGENCR1)	32	RW	0000_0000h

8.11.2 Debug General control Register (DCFG_DCSR_DBGGENCR1)

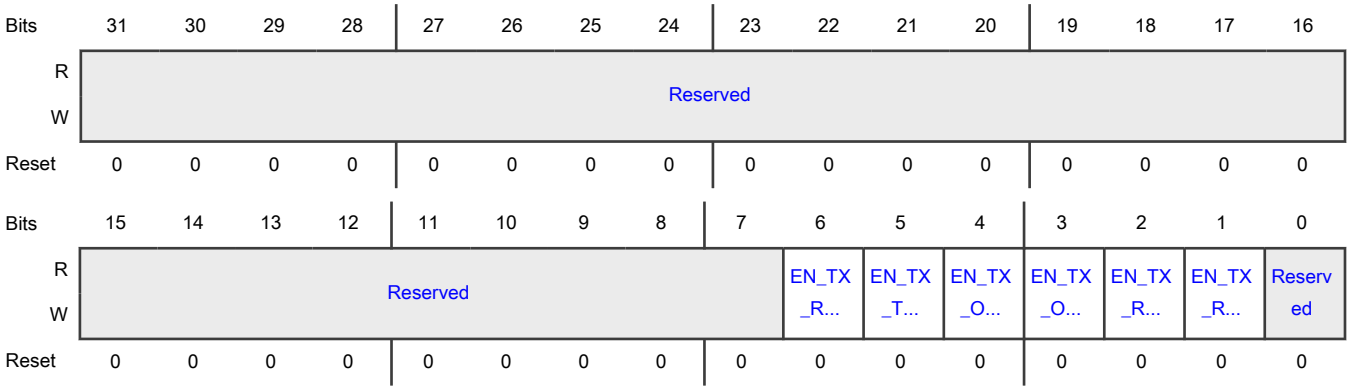
Offset

Register	Offset
DCFG_DCSR_DBGGENCR1	8_00ECh

Function

DBGGENCR provides expansion bits for device control.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-7 —	Reserved
6 EN_TX_READY	Enable TX_READY 0b - Drive TX_READY to 0 1b - Drive TX_READY to 1
5 EN_TX_TXDET_LPBK	Enable loopback from TX to TXDET 0b - Disable 1b - Enable
4 EN_TX_OBS1_LPBK	Enable loopback from TX to OBS1 0b - Disable 1b - Enable
3 EN_TX_OBS0_LPBK	Enable loopback from TX to OBS0 0b - Disable 1b - Enable
2 EN_TX_RX1_LPBK	Enable loopback from TX to RX1 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 EN_TX_RX0_LPBK	Enable loopback from TX to RX0 0b - Disable 1b - Enable
0 —	Reserved.

Chapter 9
ADC DAC Data Conversion System

9.1 Introduction

This section provides an introduction to the ADC-DAC module.

9.2 Features

The ADC-DAC includes the following features:

- ADC pairs and DAC pairs are grouped in a cluster.
- Supports programmable clock frequency (153.6 MHz) to each high speed ADC pair and high speed DAC pair
- The data converter subsystem has four high speed IQ ADC pair, one high speed IQ DAC pair, and one auxiliary ADC pair
- Supports input clock as defined in Clocking and Clock Distribution and Configuration
- Support for high speed ADC and high speed DAC debug through I2C interface

9.3 Block diagram

Data converter subsystem consists of a cluster of ADCs and DAC. Details are shown in diagram below. On north side it connects to AXIQ bridge and on south side it is connected to the transceivers. It has other interfaces such as the I2C interface (one I2C interface per High Speed ADC and High Speed DAC), which are used to access the HS-ADC/HS-DAC configuration/debug registers. Subsystem has skyblue interface which is used to program registers required to do system functionality.

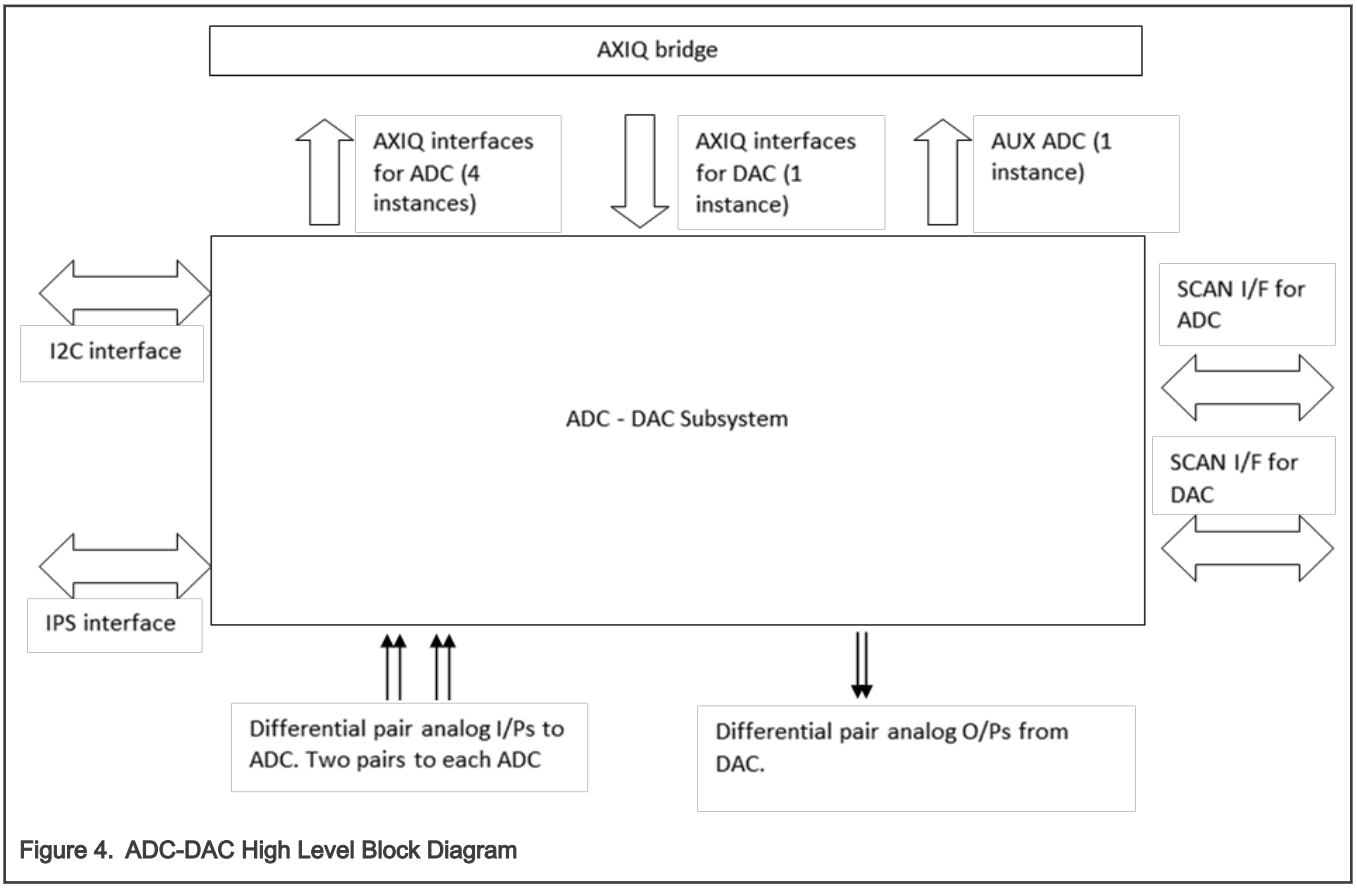


Figure 4. ADC-DAC High Level Block Diagram

A detailed diagram below shows the arrangement of the ADC/DAC and the clocking mechanism for each of them. The RX/RXO ADCs and the TX DAC sampling clocks are derived from a low jitter clock (maximum 153.6 MHz). This clock is divided based on ipg_sync signals (generated by the CCM of the SOC) in order to use either a divide by 2 version of the external clock or the undivided external clock. It is guaranteed by design to maintain synchronous relation between the external clock and the platform clock (using which the ipg_sync signals are derived). P1 and P2 are two pipeline stages that are used to ease timing for the data provided by the ADC to the AXIQ bridge and data provided to the DAC.

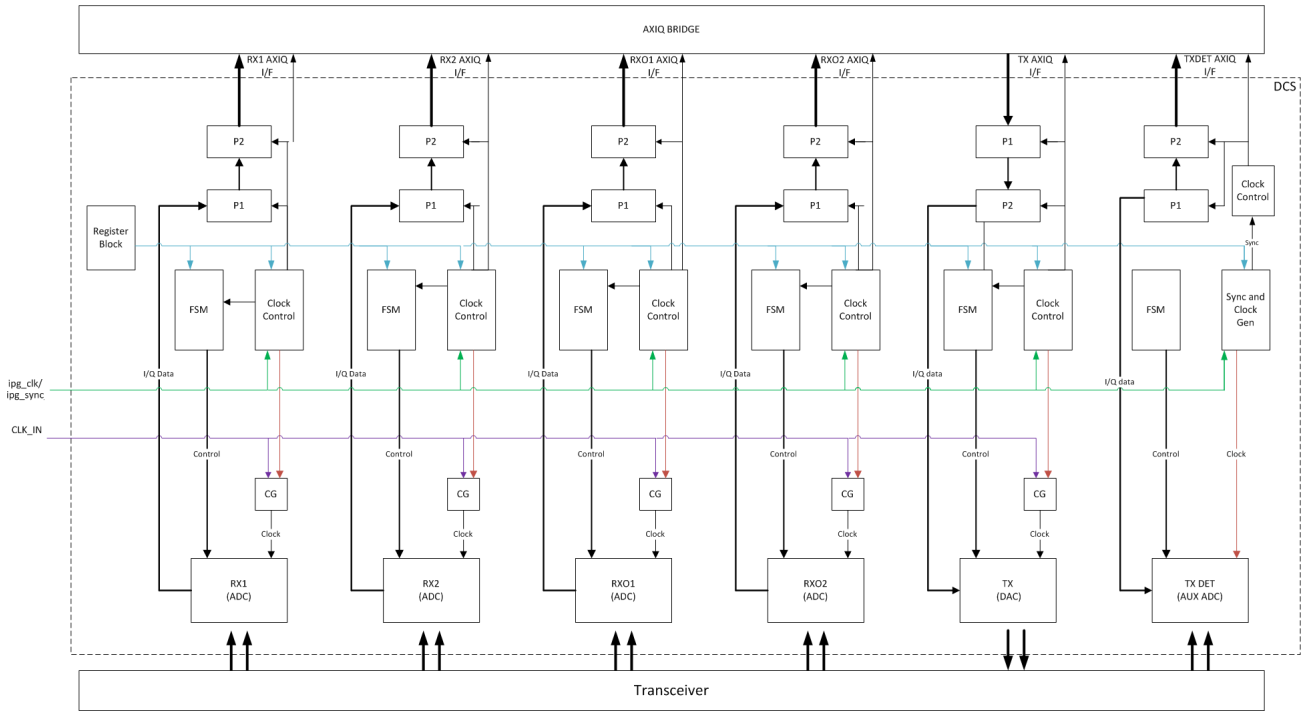


Figure 5. Detailed Description Diagram

9.4 Signals

This table describes the ADC-DAC module signals.

Table 11. ADC-DAC Signals

Signal Name	Description
ADC/DAC Receive Channels	
RX_I_P[1]	Analog Receive I Data 1 +
RX_I_N[1]	Analog Receive I Data 1 -
RX_I_P[0]	Analog Receive I Data 0 +
RX_I_N[0]	Analog Receive I Data 0 -
RX_Q_P[1]	Analog Receive Q Data 1 +
RX_Q_N[1]	Analog Receive Q Data 1 -
RX_Q_P[0]	Analog Receive Q Data 0 +
RX_Q_N[0]	Analog Receive Q Data 0 -

Table continues on the next page...

Table 11. ADC-DAC Signals (continued)

Signal Name	Description
VREF_RX_P	Analog Rx Voltage Reference +
VREF_RX_N	Analog Rx Voltage Reference -
ADC/DAC Observation Channels	
RO_I_P[1]	Analog Observation I Data 1 +
RO_I_N[1]	Analog Observation I Data 1 -
RO_I_P[0]	Analog Observation I Data 0 +
RO_I_N[0]	Analog Observation I Data 0 -
RO_Q_P[1]	Analog Observation Q Data 1 +
RO_Q_N[1]	Analog Observation Q Data 1 -
RO_Q_P[0]	Analog Observation Q Data 0 +
RO_Q_N[0]	Analog Observation Q Data 0 -
VREF_RO_P	Analog Rx (Obs) Voltage Reference +
VREF_RO_N	Analog Rx (Obs) Voltage Reference -
ADC/DAC Transmit Channel	
TX_I_P	Analog Transmit I Data +
TX_I_N	Analog Transmit I Data -
TX_Q_P	Analog Transmit Q Data +
TX_Q_N	Analog Transmit Q Data -
IREF_TX	Analog Transmit Current Reference
ADC/DAC Auxiliary Input	
TX_DET	
VREF_TX_DET	Analog Aux Channel Voltage Reference
AGNDREF_TX__DET	Analog Aux Channel GND Reference

9.5 Memory Map and register definition

This section includes the ADC-DAC module memory map and detailed descriptions of all registers.

9.5.1 ADC_DAC_SUBSYSTEM register descriptions

9.5.1.1 ADC_DAC_SUBSYSTEM Memory map

adc_dac_subsystem base address: 104_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	ADC-DAC Subsystem Version Control Register (SSVCR)	32	RO	0000_0001h
4h	Components Version Control Register (CVCR)	32	RO	0001_0101h
20h	High Speed ADC Reset Control Register (HSADC_RSTCTL)	32	RW	0000_0000h
30h	High Speed RX ADC0 Configuration Control Register (HS_RX_ADC0_CFGCTL)	32	RW	0000_1000h
34h	High Speed RX ADC1 Configuration Control Register (HS_RX_ADC1_CFGCTL)	32	RW	0002_1000h
38h	High Speed RXO ADC0 Configuration Control Register (HS_RXO_ADC0_CFGCTL)	32	RW	0004_1000h
3Ch	High Speed RXO ADC1 Configuration Control Register (HS_RXO_ADC1_CFGCTL)	32	RW	0006_1000h
50h	High Speed RX ADC0 Timing Control Register1 (HS_RX_ADC0_TMNGCTL1)	32	RW	1F40_D8CCh
54h	High Speed RX ADC1 Timing Control Register1 (HS_RX_ADC1_TMNGCTL1)	32	RW	1F40_D8CCh
58h	High Speed RXO ADC0 Timing Control Register1 (HS_RXO_ADC0_TMNGCTL1)	32	RW	1F40_D8CCh
5Ch	High Speed RXO ADC1 Timing Control Register1 (HS_RXO_ADC1_TMNGCTL1)	32	RW	1F40_D8CCh
70h	High Speed RX ADC0 Timing Control Register2 (HS_RX_ADC0_TMNGCTL2)	32	RW	0015_000Ah
74h	High Speed RX ADC1 Timing Control Register2 (HS_RX_ADC1_TMNGCTL2)	32	RW	0015_000Ah
78h	High Speed RXO ADC0 Timing Control Register2 (HS_RXO_ADC0_TMNGCTL2)	32	RW	0015_000Ah
7Ch	High Speed RXO ADC1 Timing Control Register2 (HS_RXO_ADC1_TMNGCTL2)	32	RW	0015_000Ah
90h	High Speed ADC Calibration Control Register (HSADC_CALCTL)	32	RW	0000_0000h
A0h	High Speed ADC Enable Control Register (HSADC_ENCTL)	32	RW	0000_0000h
100h	Auxiliary ADC Reset Control Register (AUXADC_RSTCTL)	32	RW	0000_0000h
110h	AUX ADC Configuration Control Register (AUXADC_CFGCTL)	32	RW	0000_3035h
120h	AUX ADC Timing Control Register (AUXADC_TMNGCTL)	32	RW	0019_0002h
130h	Auxiliary ADC Enable Control Register (AUXADC_ENCTL)	32	RW	0000_0000h
200h	High Speed DAC Reset Control Register (HSDAC_RSTCTL)	32	RW	0000_0000h
210h	High Speed DAC Configuration Control Register1 (HSDAC_CFGCTL1)	32	RW	0000_0700h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
214h	High Speed DAC Configuration Control Register2 (HSDAC_CFGCTL2)	32	RW	0002_3093h
230h	High Speed DAC Timing Control Register1 (HSDAC_TMNGCTL1)	32	RW	5DC0_2710h
234h	High Speed DAC Timing Control Register2 (HSDAC_TMNGCTL2)	32	RW	0000_000Ah
250h	High Speed DAC Enable Control Register (HSDAC_ENCTL)	32	RW	0000_0000h
300h	ADC DAC Clock Configuration Register (ADC_DAC_CLKCFG)	32	RW	0000_0000h
308h	AUX ADC Clock Configuration Register (AUX_ADC_CLKCFG)	32	RW	0000_0000h
310h	Clock Control Register (CLK_CTRL)	32	RW	0000_0000h
318h	I2C Operation Control Register (I2C_OPCTL)	32	RW	0000_0000h
320h	Gate Ready Register (GATE_READY)	32	RW	0001_010Fh
330h	Subsystem Timeout Interrupt Enable Register (SUBSYS_INTREN)	32	RW	0000_0000h
340h	Subsystem Interrupt Type Register (SUBSYS_INTRTYP)	32	RW	0000_0000h
400h	High Speed ADC Status Register (HSADC_STAT)	32	RO	0000_0000h
408h	High Speed ADC Error Status Register1 (HSADC_ERRSTAT1)	32	RO	0000_0000h
40Ch	High Speed ADC Error Status Register2 (HSADC_ERRSTAT2)	32	RO	0000_0000h
420h	Auxiliary ADC Status Register (AUXADC_STAT)	32	RO	0000_0000h
428h	Auxiliary ADC Error Status Register (AUXADC_ERRSTAT)	32	RO	0000_0000h
440h	High Speed DAC Status Register (HSDAC_STAT)	32	RO	0000_0000h
448h	High Speed DAC Error Status Register1 (HSDAC_ERRSTAT1)	32	RO	0000_0000h
44Ch	High Speed DAC Error Status Register2 (HSDAC_ERRSTAT2)	32	RO	0000_0000h
C10h	Test Loopback Configuration Control Register (TEST_LPBK_CFGCTL)	32	RW	0000_0000h
C20h	Test Configuration Control Register1 (TEST_CFGCTL1)	32	RW	0000_0000h
C24h	Test Configuration Control Register2 (TEST_CFGCTL2)	32	RW	0000_0000h
C28h	Test Configuration Control Register3 (TEST_CFGCTL3)	32	RW	0000_0000h
C2Ch	Test Configuration Control Register4 (TEST_CFGCTL4)	32	RW	0000_0000h
C30h	Test Configuration Control Register5 (TEST_CFGCTL5)	32	RW	0000_0000h
C34h	Test MUX Control Register (TEST_MUXCTL)	32	RW	See description.
F20h	Test Group Digital Output Mux Control Register (TEST_GRP_DOMUXCTL)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
F24h	Test Miscellaneous Digital Output Mux Control Register (TEST_MISC_DOMUXCTL)	32	ROZ	See description.
F28h	Test Offset Configuration Register (TEST_OFFSET_CFG)	32	RW	0000_00FAh
F2Ch	Test Pin Expose Configuration Register (TEST_PIN_EXPOSE_CFG)	32	ROZ	See description.
F30h	Test Status Register (TEST_STAT)	32	RO	0000_0000h
F34h	Register Space Synchronize Register (REG_SPACE_SYNC)	32	RW	0000_0000h

9.5.1.2 ADC-DAC Subsystem Version Control Register (SSVCR)

Offset

Register	Offset
SSVCR	0h

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								SUBSYSTEM_VER_NO							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Fields

Field	Function
31-8 —	Reserved.
7-0 SUBSYSTEM_VER_NO	This field contains the ADC-DAC Subsystem version number

9.5.1.3 Components Version Control Register (CVCR)

Offset

Register	Offset
CVCR	4h

Diagram

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

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

Fields

Field	Function
31-24 —	Reserved.
23-16 HIGH_SPEED_DAC_VER_NO	This field contains the High Speed DAC version number
15-8 AUX_ADC_VER_NO	This field contains the Auxiliary ADC version number
7-0 HIGH_SPEED_ADC_VER_NO	This field contains the High Speed ADC version number

9.5.1.4 High Speed ADC Reset Control Register (HSADC_RSTCTL)

Offset

Register	Offset
HSADC_RSTCTL	20h

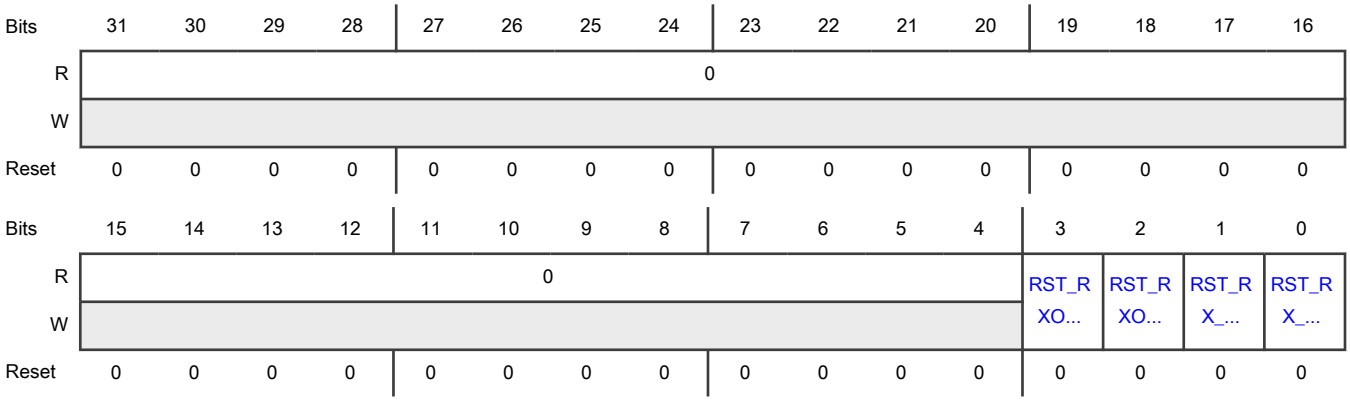
Function

ADC Reset. Correctly initializes the ADC.

Mandatory to be performed after power supplies ramp-up.

Write 1 in the respective bit field to reset the corresponding high speed ADC. This bit is self-clearing. Hardware blocks all skyblue writes to this bit until it is cleared. It is thus expected that sofwatre, after setting this bit, will poll until the bit is cleared.

Diagram



Fields

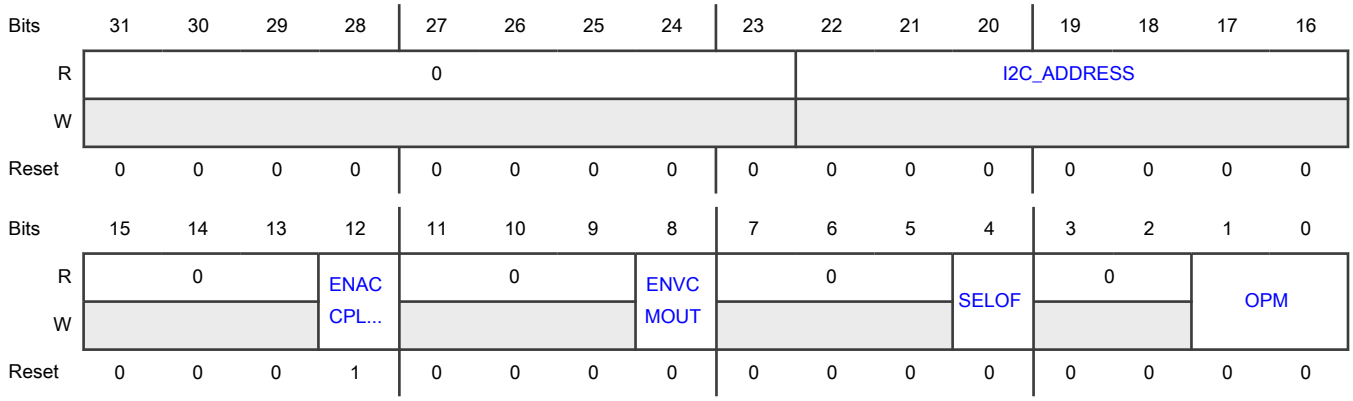
Field	Function
31-4 —	Reserved.
3 RST_RXO_ADC1	Reset RXO ADC 1
2 RST_RXO_ADC0	Reset RXO ADC 0
1 RST_RX_ADC1	Reset RX ADC 1
0 RST_RX_ADC0	Reset RX ADC 0

9.5.1.5 High Speed RX ADC0 Configuration Control Register (HS_RX_ADC0_CFGCTL)

Offset

Register	Offset
HS_RX_ADC0_CFGCTL	30h

Diagram



Fields

Field	Function
31-23 —	Reserved.
22-16 I2C_ADDRESS	I2C device slave address. Denotes the I2C address assigned to the ADC The I2C addresses pre-assgined to ADC are: I Channel - n*2 where n is the ADC number Q Channel - (n*2)+1 where n is the ADC number ADC number relation is described below- RX ADC0 - 0 RX ADC1 - 1 RX0 ADC0 - 2 RX0 ADC1 - 3
15-13 —	Reserved.
12 ENACCPLING	Control ADC input configuration to support AC coupling on Vin{p|n}{i|q} signals
11-9 —	Reserved.
8 ENVCMOUT	Control internal common-mode voltage generation 1: Internal common-mode enabled. Vcmout with ADC internal common voltage provided at the pin 0: Internal common-mode generator disabled. Vcmout with AVDD value. Vin{p|n}{i|q} common mode voltage to be provided externally

Table continues on the next page...

Table continued from the previous page...

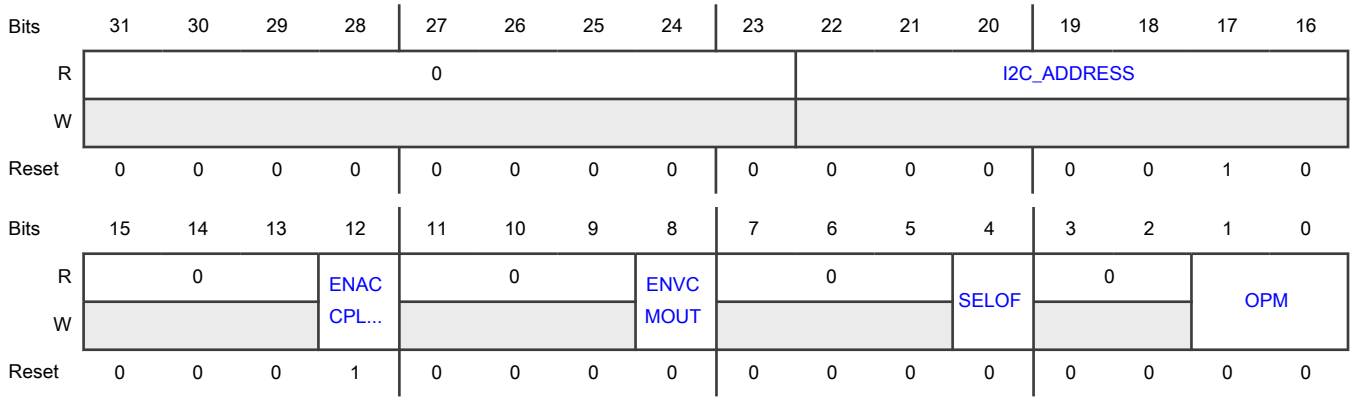
Field	Function
7-5 —	Reserved.
4 SELOF	Select ADC digital output format 1: 2's compliment 0: Straight binary
3-2 —	Reserved.
1-0 OPM	Operational Mode Control Signal 11: Active Mode 01 | 10: Sleep Mode 00: Power Down Mode

9.5.1.6 High Speed RX ADC1 Configuration Control Register (HS_RX_ADC1_CFGCTL)

Offset

Register	Offset
HS_RX_ADC1_CFGCTL	34h

Diagram



Fields

Field	Function
31-23	Reserved.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
22-16 I2C_ADDRESS	I2C device slave address. Denotes the I2C address assigned to the ADC The I2C addresses pre-assgined to ADC are: I Channel - n*2 where n is the ADC number Q Channel - (n*2)+1 where n is the ADC number ADC number relation is described below- RX ADC0 - 0 RX ADC1 - 1 RX0 ADC0 - 2 RX0 ADC1 - 3
15-13 —	Reserved.
12 ENACCPLING	Control ADC input configuration to support AC coupling on Vin{p|n}{i|q} signals
11-9 —	Reserved.
8 ENVCMOUT	Control internal common-mode voltage generation 1: Internal common-mode enabled. Vcmout with ADC internal common voltage provided at the pin 0: Internal common-mode generator disabled. Vcmout with AVDD value. Vin{p|n}{i|q} common mode voltage to be provided externally
7-5 —	Reserved.
4 SELOF	Select ADC digital output format 1: 2's compliment 0: Straight binary
3-2 —	Reserved.
1-0 OPM	Operational Mode Control Signal 11: Active Mode 01 | 10: Sleep Mode

Table continues on the next page...

Table continued from the previous page...

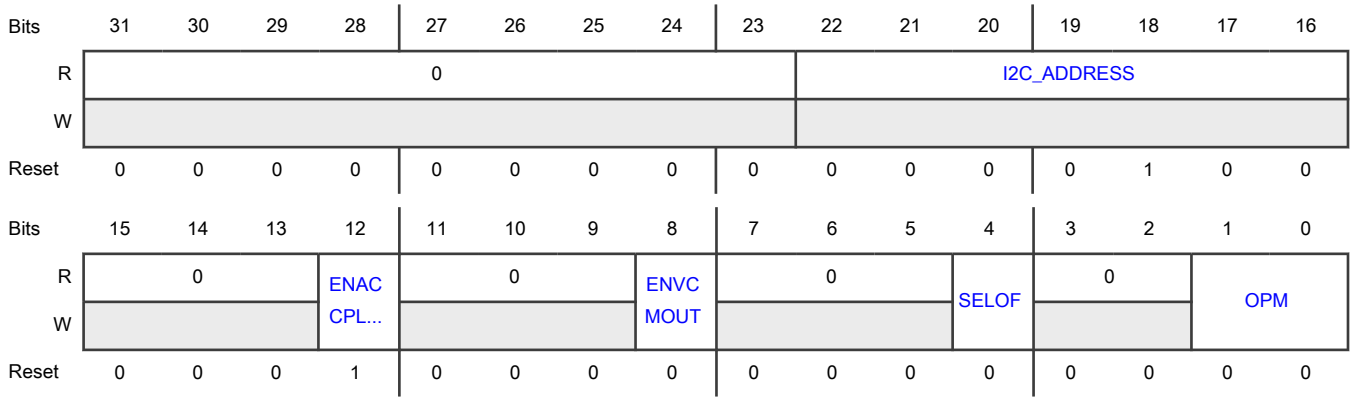
Field	Function
	00: Power Down Mode

9.5.1.7 High Speed RXO ADC0 Configuration Control Register (HS_RXO_ADC0_CFGCTL)

Offset

Register	Offset
HS_RXO_ADC0_CFGCTL	38h

Diagram



Fields

Field	Function
31-23 —	Reserved.
22-16 I2C_ADDRESS	I2C device slave address. Denotes the I2C address assigned to the ADC The I2C addresses pre-assgined to ADC are: I Channel - n*2 where n is the ADC number Q Channel - (n*2)+1 where n is the ADC number ADC number relation is described below- RX ADC0 - 0 RX ADC1 - 1 RX0 ADC0 - 2

Table continues on the next page...

Table continued from the previous page...

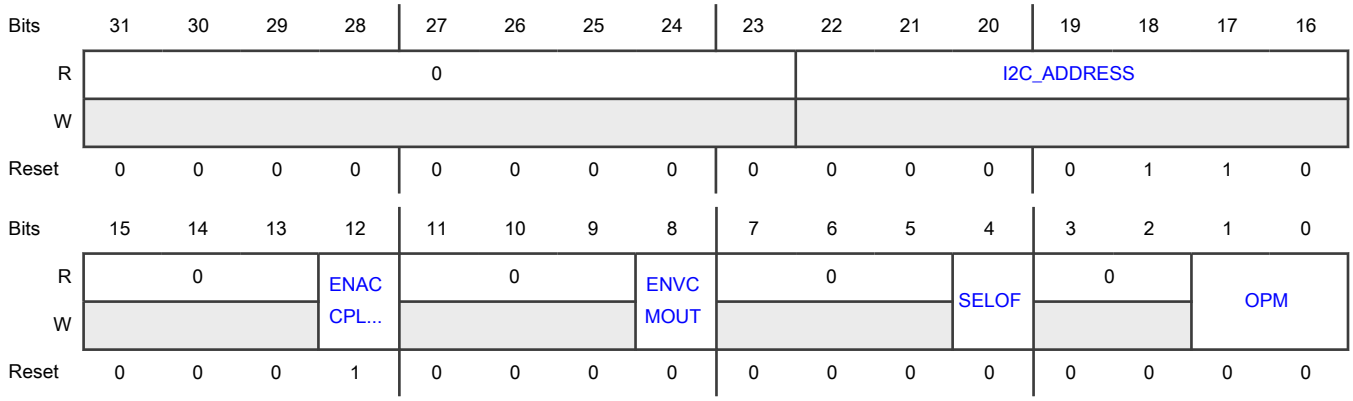
Field	Function
	RX0 ADC1 - 3
15-13 —	Reserved.
12 ENACCPLING	Control ADC input configuration to support AC coupling on Vin{p|n}{i|q} signals
11-9 —	Reserved.
8 ENVCMOUT	Control internal common-mode voltage generation 1: Internal common-mode enabled. Vcmout with ADC internal common voltage provided at the pin 0: Internal common-mode generator disabled. Vcmout with AVDD value. Vin{p|n}{i|q} common mode voltage to be provided externally
7-5 —	Reserved.
4 SELOF	Select ADC digital output format 1: 2's compliment 0: Straight binary
3-2 —	Reserved.
1-0 OPM	Operational Mode Control Signal 11: Active Mode 01 | 10: Sleep Mode 00: Power Down Mode

9.5.1.8 High Speed RXO ADC1 Configuration Control Register (HS_RXO_ADC1_CFGCTL)

Offset

Register	Offset
HS_RXO_ADC1_CFGCTL	3Ch

Diagram



Fields

Field	Function
31-23 —	Reserved.
22-16 I2C_ADDRESS	I2C device slave address. Denotes the I2C address assigned to the ADC The I2C addresses pre-assgined to ADC are: I Channel - n*2 where n is the ADC number Q Channel - (n*2)+1 where n is the ADC number ADC number relation is described below- RX ADC0 - 0 RX ADC1 - 1 RX0 ADC0 - 2 RX0 ADC1 - 3
15-13 —	Reserved.
12 ENACCPLING	Control ADC input configuration to support AC coupling on Vin{p|n}{i|q} signals
11-9 —	Reserved.
8 ENVCMOUT	Control internal common-mode voltage generation 1: Internal common-mode enabled. Vcmout with ADC internal common voltage provided at the pin 0: Internal common-mode generator disabled. Vcmout with AVDD value. Vin{p|n}{i|q} common mode voltage to be provided externally

Table continues on the next page...

Table continued from the previous page...

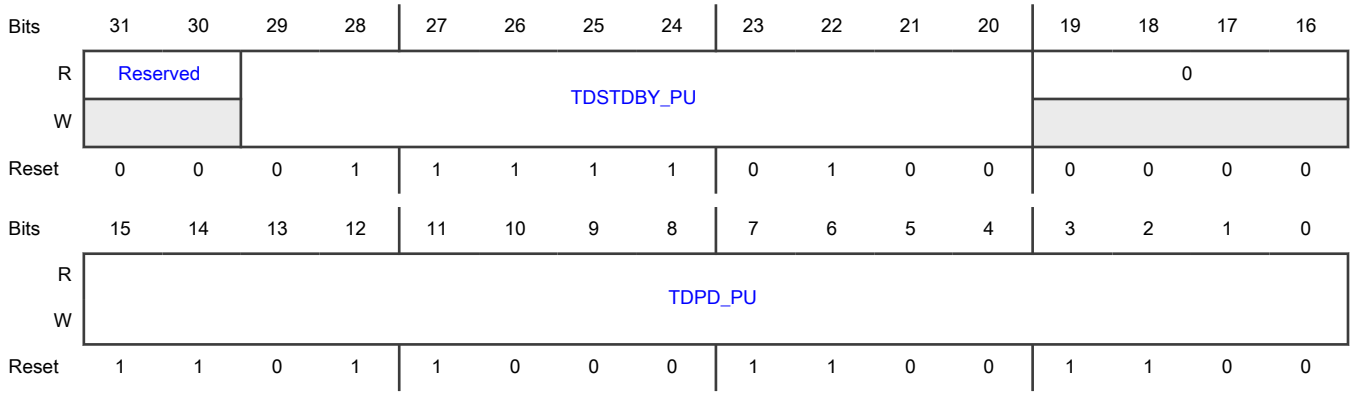
Field	Function
7-5 —	Reserved.
4 SELOF	Select ADC digital output format 1: 2's compliment 0: Straight binary
3-2 —	Reserved.
1-0 OPM	Operational Mode Control Signal 11: Active Mode 01 | 10: Sleep Mode 00: Power Down Mode

9.5.1.9 High Speed RX ADC0 Timing Control Register1 (HS_RX_ADC0_TMNGCTL1)

Offset

Register	Offset
HS_RX_ADC0_TMNGCTL1	50h

Diagram



Fields

Field	Function
31-30 —	Reserved.
29-20 TDSTDBY_PU	This field indicates the time it takes for the ADC to assert adcrdy signal when it is moved from sleep/standby state to active state. This duration is to be programmed in terms of number of ADC clock cycles
19-16 —	Reserved.
15-0 TDPD_PU	This field indicates the time it takes for the ADC to assert adcrdy signal when it is moved from powerdown to active state. This duration is to be programmed in terms of number of ADC clock cycles

9.5.1.10 High Speed RX ADC1 Timing Control Register1 (HS_RX_ADC1_TMNGCTL1)

Offset

Register	Offset
HS_RX_ADC1_TMNGCTL1	54h

Diagram

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

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

Fields

Field	Function
31-30 —	Reserved.

Table continues on the next page...

Table continued from the previous page...

Field	Function
29-20 TDSTDBY_PU	This field indicates the time it takes for the ADC to assert adcrdy signal when it is moved from sleep/standby state to active state. This duration is to be programmed in terms of number of ADC clock cycles
19-16 —	Reserved.
15-0 TDPD_PU	This field indicates the time it takes for the ADC to assert adcrdy signal when it is moved from powerdown to active state. This duration is to be programmed in terms of number of ADC clock cycles

9.5.1.11 High Speed RXO ADC0 Timing Control Register1 (HS_RXO_ADC0_TMNGCTL1)

Offset

Register	Offset
HS_RXO_ADC0_TMNGCTL1	58h

Diagram

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

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

Fields

Field	Function
31-30 —	Reserved.
29-20 TDSTDBY_PU	This field indicates the time it takes for the ADC to assert adcrdy signal when it is moved from sleep/standby state to active state. This duration is to be programmed in terms of number of ADC clock cycles
19-16	Reserved.

Table continues on the next page...

Table continued from the previous page...

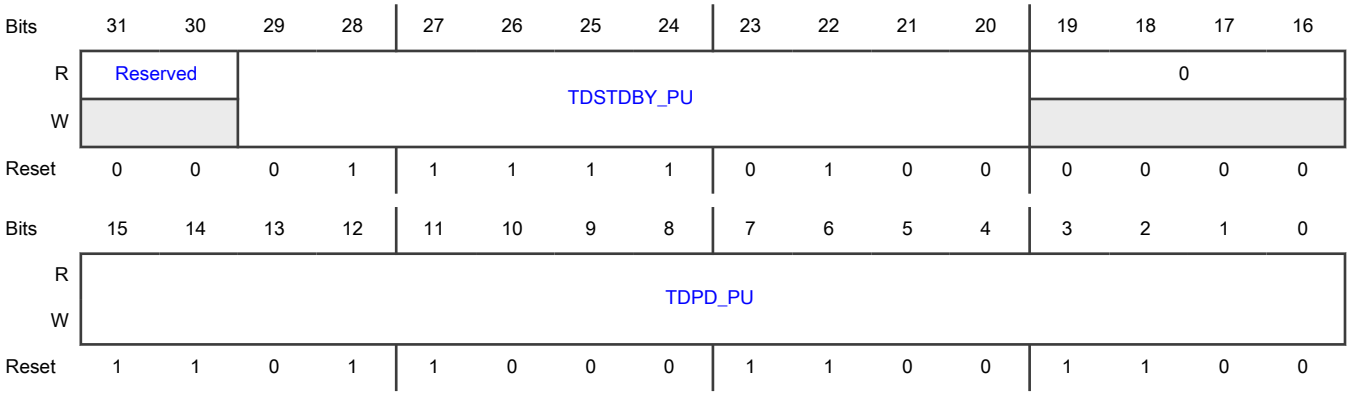
Field	Function
—	
15-0 TDPD_PU	This field indicates the time it takes for the ADC to assert adcrdy signal when it is moved from powerdown to active state. This duration is to be programmed in terms of number of ADC clock cycles

9.5.1.12 High Speed RXO ADC1 Timing Control Register1 (HS_RXO_ADC1_TMNGCTL1)

Offset

Register	Offset
HS_RXO_ADC1_TMNGCTL1	5Ch

Diagram



Fields

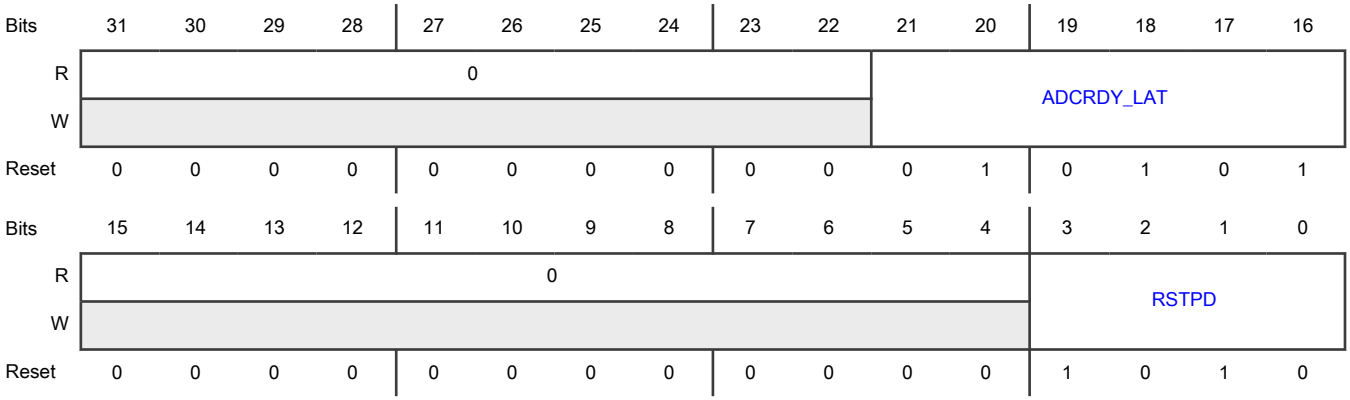
Field	Function
31-30 —	Reserved.
29-20 TDSTDBY_PU	This field indicates the time it takes for the ADC to assert adcrdy signal when it is moved from sleep/standby state to active state. This duration is to be programmed in terms of number of ADC clock cycles
19-16 —	Reserved.
15-0 TDPD_PU	This field indicates the time it takes for the ADC to assert adcrdy signal when it is moved from powerdown to active state. This duration is to be programmed in terms of number of ADC clock cycles

9.5.1.13 High Speed RX ADC0 Timing Control Register2 (HS_RX_ADC0_TMNGCTL2)

Offset

Register	Offset
HS_RX_ADC0_TMNGCTL2	70h

Diagram



Fields

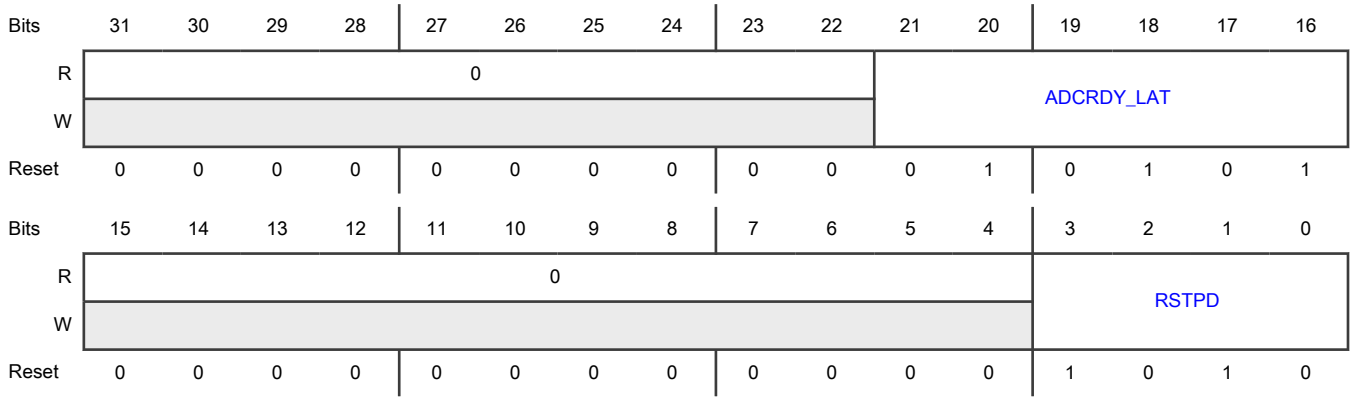
Field	Function
31-22 —	Reserved.
21-16 ADCRDY_LAT	This field corresponds to the latency from the time adcrdy is asserted to valid data output from the ADC This duration is to be programmed in terms of number of ADC clock cycles
15-4 —	Reserved.
3-0 RSTPD	ADC Reset Pulse Duration This field is used to program the duration for which reset needs to be asserted to the ADC. This duration is to be programmed in terms of number of ADC clock cycles

9.5.1.14 High Speed RX ADC1 Timing Control Register2 (HS_RX_ADC1_TMNGCTL2)

Offset

Register	Offset
HS_RX_ADC1_TMNGCTL2	74h

Diagram



Fields

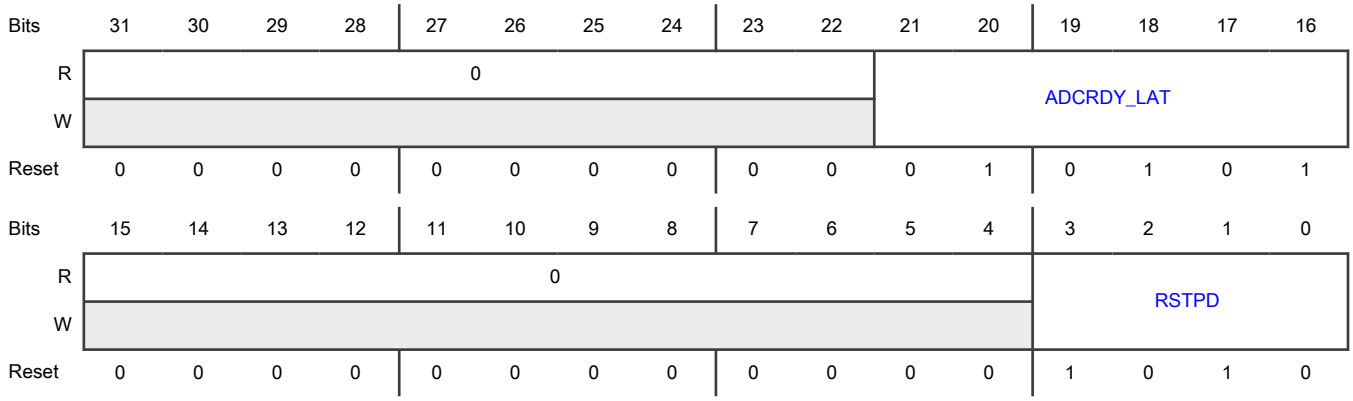
Field	Function
31-22 —	Reserved.
21-16 ADCRDY_LAT	This field corresponds to the latency from the time adcrdy is asserted to valid data output from the ADC This duration is to be programmed in terms of number of ADC clock cycles
15-4 —	Reserved.
3-0 RSTPD	ADC Reset Pulse Duration This field is used to program the duration for which reset needs to be asserted to the ADC. This duration is to be programmed in terms of number of ADC clock cycles

9.5.1.15 High Speed RXO ADC0 Timing Control Register2 (HS_RXO_ADC0_TMNGCTL2)

Offset

Register	Offset
HS_RXO_ADC0_TMNGCTL2	78h

Diagram



Fields

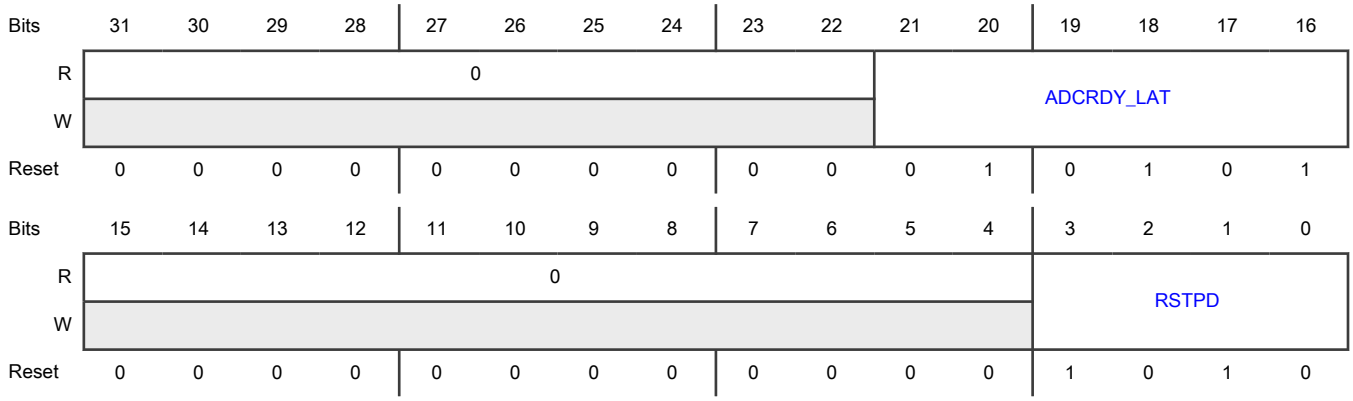
Field	Function
31-22 —	Reserved.
21-16 ADCRDY_LAT	This field corresponds to the latency from the time adcrdy is asserted to valid data output from the ADC This duration is to be programmed in terms of number of ADC clock cycles
15-4 —	Reserved.
3-0 RSTPD	ADC Reset Pulse Duration This field is used to program the duration for which reset needs to be asserted to the ADC. This duration is to be programmed in terms of number of ADC clock cycles

9.5.1.16 High Speed RXO ADC1 Timing Control Register2 (HS_RXO_ADC1_TMNGCTL2)

Offset

Register	Offset
HS_RXO_ADC1_TMNGCTL2	7Ch

Diagram



Fields

Field	Function
31-22 —	Reserved.
21-16 ADCRDY_LAT	This field corresponds to the latency from the time adcrdy is asserted to valid data output from the ADC This duration is to be programmed in terms of number of ADC clock cycles
15-4 —	Reserved.
3-0 RSTPD	ADC Reset Pulse Duration This field is used to program the duration for which reset needs to be asserted to the ADC. This duration is to be programmed in terms of number of ADC clock cycles

9.5.1.17 High Speed ADC Calibration Control Register (HSADC_CALCTL)

Offset

Register	Offset
HSADC_CALCTL	90h

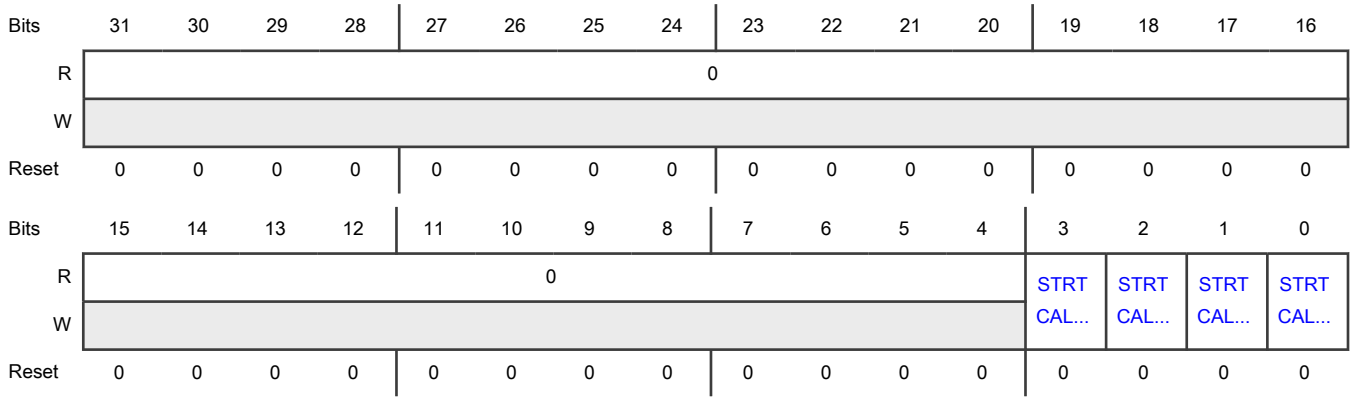
Function

Trigger ADC calibration cycle.

Apply a high pulse with minimum tcal cycles to trigger feature. Default value is low.

The bits in this register are self-clearing. Thus, once the bit corresponding to a particular ADC is set, it will be cleared by hardware when the calibration control signal to ADC is deasserted.

Diagram



Fields

Field	Function
31-4 —	Reserved.
3 STRTCAL_RXO_ADC1	Trigger calibration for RXO ADC 1
2 STRTCAL_RXO_ADC0	Trigger calibration for RXO ADC 0
1 STRTCAL_RX_ADC1	Trigger calibration for RX ADC 1
0 STRTCAL_RX_ADC0	Trigger calibration for RX ADC 0

9.5.1.18 High Speed ADC Enable Control Register (HSADC_ENCTL)

Offset

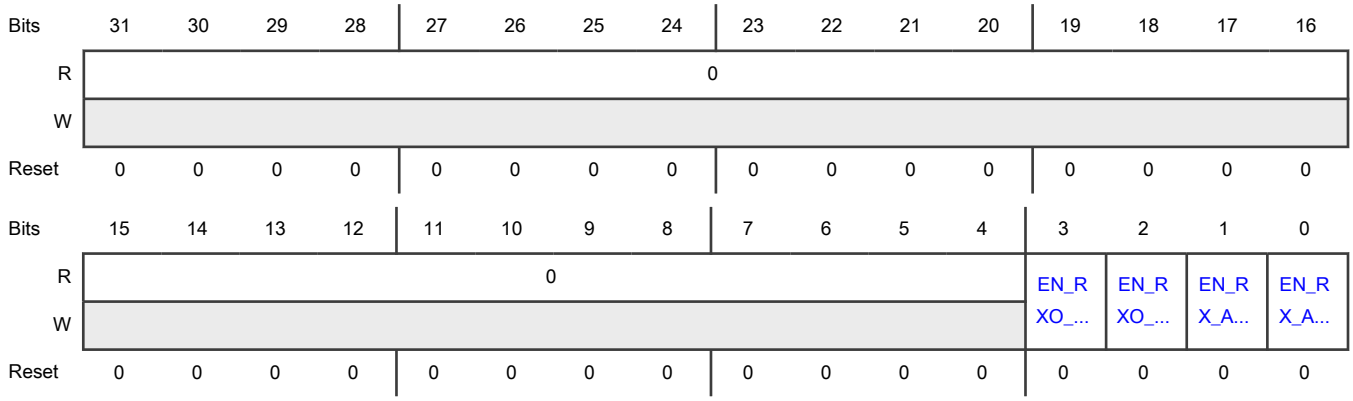
Register	Offset
HSADC_ENCTL	A0h

Function

The bits in this register are expected to be set only once the high speed ADCs have powered-up.

Clearing any of these bits will result in the corresponding ADC being brought down to its powerdown state.

Diagram



Fields

Field	Function
31-4 —	Reserved.
3 EN_RXO_ADC1	Enable RX0 ADC 1
2 EN_RXO_ADC0	Enable RXO ADC 0
1 EN_RX_ADC1	Enable RX ADC 1
0 EN_RX_ADC0	Enable RX ADC 0

9.5.1.19 Auxiliary ADC Reset Control Register (AUXADC_RSTCTL)

Offset

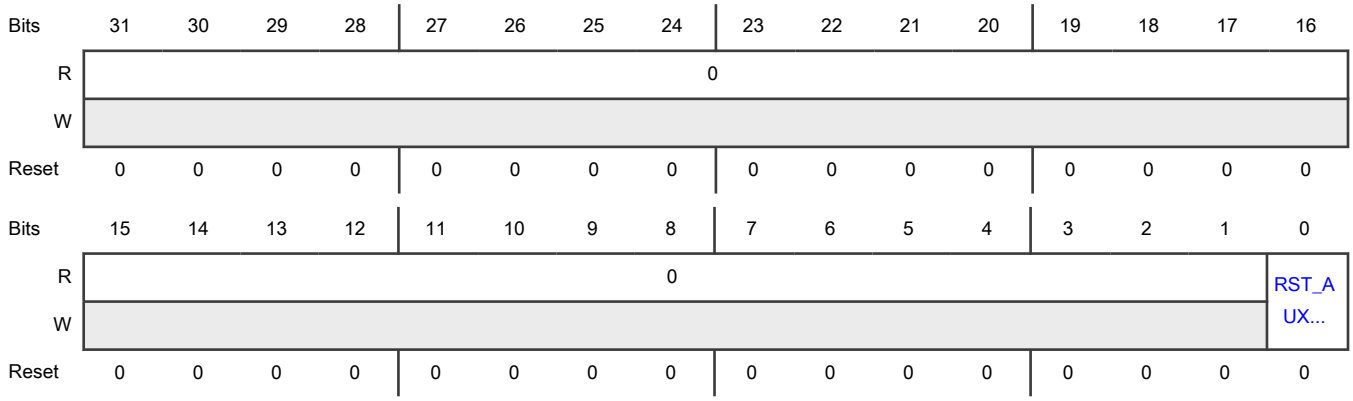
Register	Offset
AUXADC_RSTCTL	100h

Function

Asynchronous reset of internal buffers and registers.

Write 1 in the RST_AUXADC0 bit field to reset the auxiliary ADC. This bit is self-clearing. Hardware blocks all skyblue writes to this bit until it is cleared. It is thus expected that sofwatre, after setting this bit, will poll until the bit is cleared.

Diagram



Fields

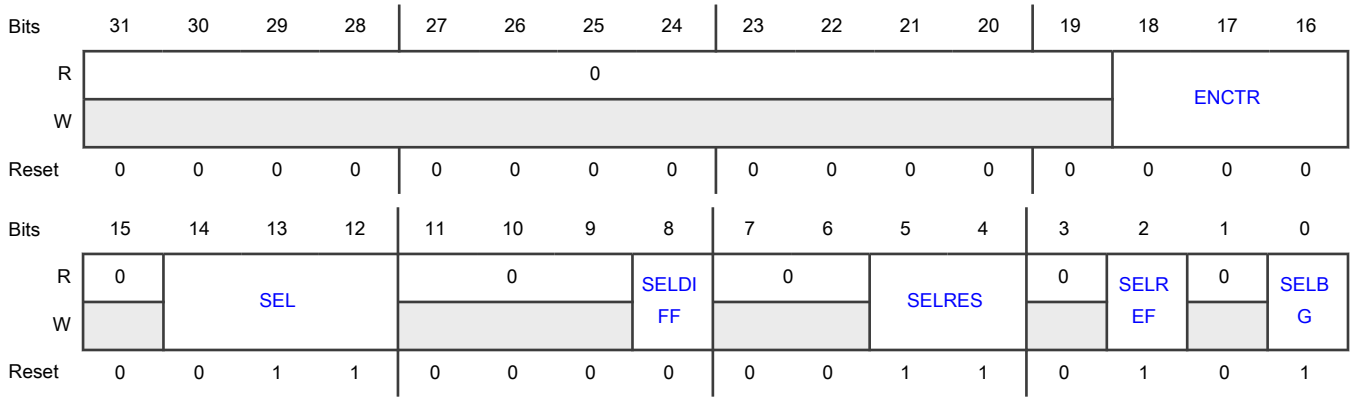
Field	Function
31-1 —	Reserved.
0 RST_AUXADC0	Reset Auxiliary ADC 0

9.5.1.20 AUX ADC Configuration Control Register (AUXADC_CFGCTL)

Offset

Register	Offset
AUXADC_CFGCTL	110h

Diagram



Fields

Field	Function
31-19 —	Reserved.
18-16 ENCTR	Digital signals to enable internal analog programmability modes used for test purposes. In normal operation, these signals should be connected into gnd.
15 —	Reserved.
14-12 SEL	Input multiplexer control signals: vinp|n0 selected when sel=0 vinp|n7 selected when sel=7
11-9 —	Reserved.
8 SELDIFF	Selects the ADC input mode (8 selectable inputs are available in both configurations): 1: Differential inputs 0: Single ended inputs
7-6 —	Reserved.
5-4 SELRES	Selects the ADC resolution: 11: 12 bit mode 10: 10 bit mode 01: 8 bit mode 00: 6 bit mode
3 —	Reserved.
2 SELREF	Selects if the fullscale input range of the converter is defined by: 1: The internal reference generator, depending on the bandgap voltage 0: The voltage applied to vref
1 —	Reserved.
0 SELBG	Selects if the internal bandgap voltage generator is active or not. 1: Internal bandgap is active. The vbg pin is at high impedance

Table continues on the next page...

Table continued from the previous page...

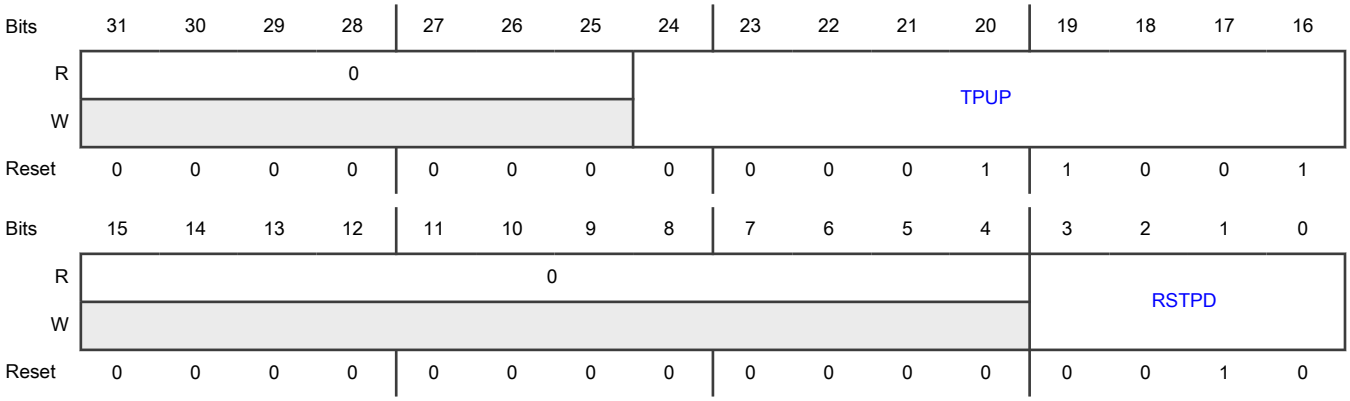
Field	Function
	0: Internal bandgap is in powerdown. User should connect an external 1.2V voltage to vbg pin if the built-in reference buffer is to be used (SELREF = 1)

9.5.1.21 AUX ADC Timing Control Register (AUXADC_TMNGCTL)

Offset

Register	Offset
AUXADC_TMNGCTL	120h

Diagram



Fields

Field	Function
31-25 —	Reserved.
24-16 TPUP	This field defines the power-up time for auxiliary ADC when internal reference voltage is selected. This duration is to be programmed in terms of number of auxiliary ADC clock cycles
15-4 —	Reserved.
3-0 RSTPD	Auxiliary ADC Reset Pulse Duration This field is used to program the duration for which reset needs to be asserted to the ADC. This duration is to be programmed in terms of number of auxiliary ADC clock cycles

9.5.1.22 Auxiliary ADC Enable Control Register (AUXADC_ENCTL)

Offset

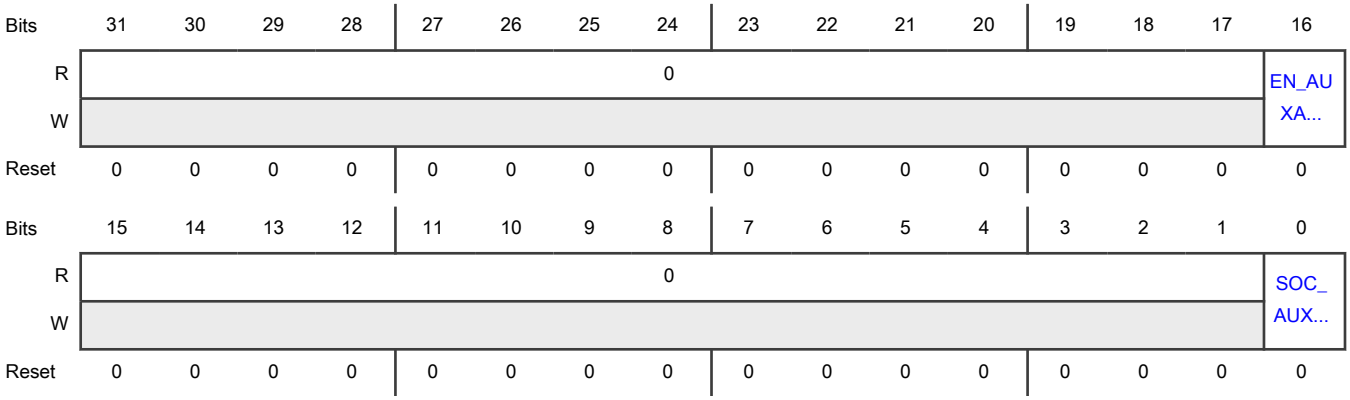
Register	Offset
AUXADC_ENCTL	130h

Function

The bits in this register are expected to be set only once the auxiliary ADCs have powered-up.

Clearing any of these bits will result in the corresponding ADC being brought down to its powerdown state.

Diagram



Fields

Field	Function
31-17 —	Reserved.
16 EN_AUXADC0	Enable auxiliary ADC 0 0b - Auxiliary ADC 0 is disabled 1b - Auxiliary ADC 0 is enabled
15-1 —	Reserved.
0 SOC_AUXADC0	Start of conversion for auxiliary ADC 0

9.5.1.23 High Speed DAC Reset Control Register (HSDAC_RSTCTL)

Offset

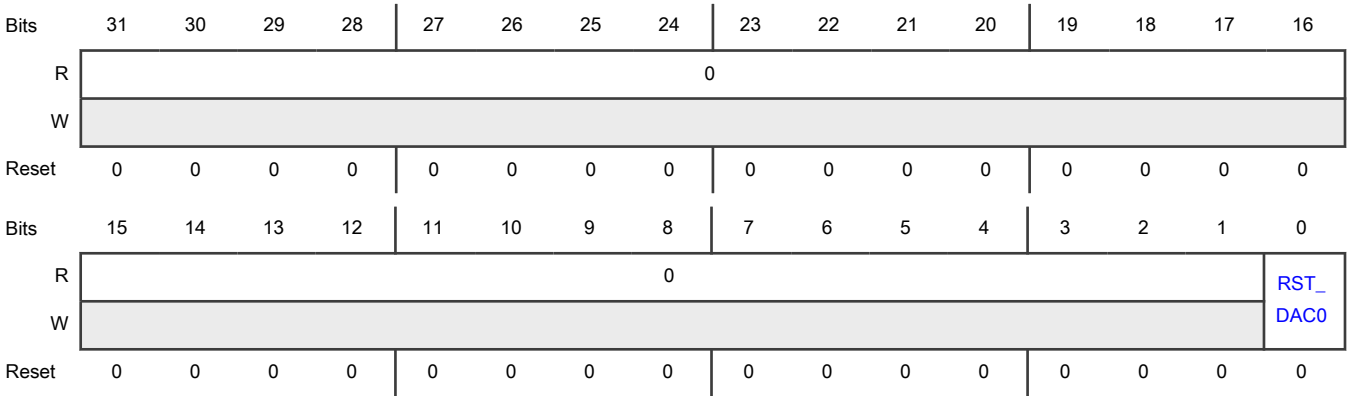
Register	Offset
HSDAC_RSTCTL	200h

Function

This is the cell reset signal.

Write 1 in the RST_DAC0 bit field to reset the high speed DAC. This bit is self-clearing. Hardware blocks all skyblue writes to this bit until it is cleared. It is thus expected that sofwatre, after setting this bit, will poll until the bit is cleared.

Diagram



Fields

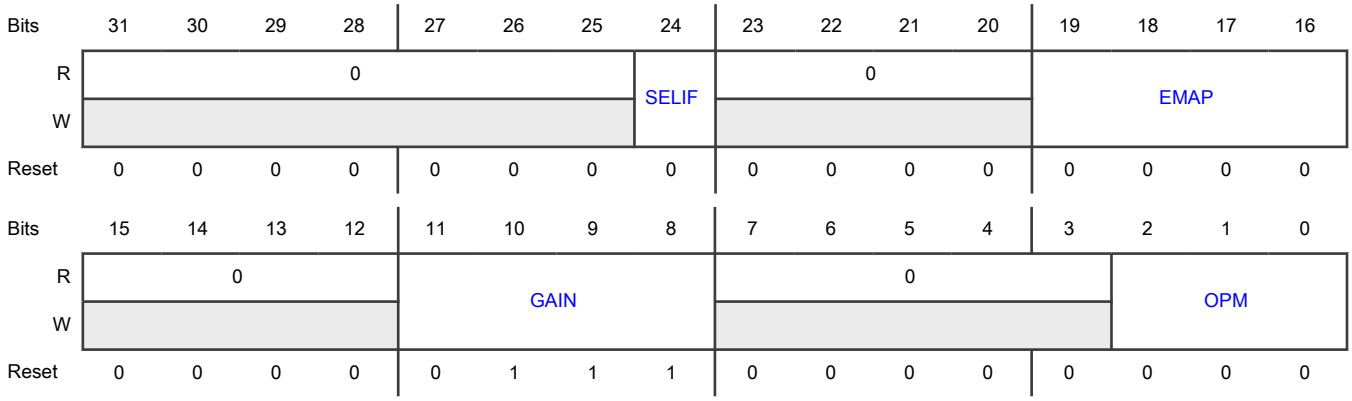
Field	Function
31-1 —	Reserved.
0 RST_DAC0	Reset DAC 0

9.5.1.24 High Speed DAC Configuration Control Register1 (HSDAC_CFGCTL1)

Offset

Register	Offset
HSDAC_CFGCTL1	210h

Diagram



Fields

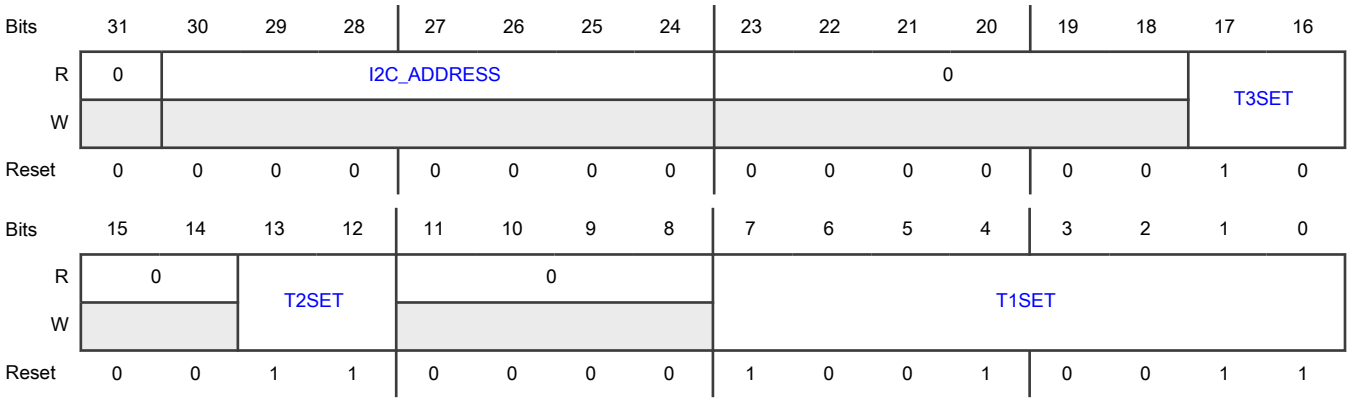
Field	Function
31-25 —	Reserved.
24 SELIF	Input binary format control: 1: DAC input word in 2's compliment 0: DAC input word in straight binary
23-20 —	Reserved.
19-16 EMAP	MSB and LSB mapping and operation modes. Set emap=0000 in normal operation
15-12 —	Reserved.
11-8 GAIN	Output fullscale current control
7-3 —	Reserved.
2-0 OPM	Basic operating modes

9.5.1.25 High Speed DAC Configuration Control Register2 (HSDAC_CFGCTL2)

Offset

Register	Offset
HSDAC_CFGCTL2	214h

Diagram



Fields

Field	Function
31 —	Reserved.
30-24 I2C_ADDRESS	I2C interface register address bus. Denotes the I2C address assigned to the corresponding DAC. Set to 0 in normal operation. The I2C address pre-assgined to DAC is n where n is the DAC number I Channel - n*2 where n is the ADC number Q Channel - (n*2)+1 where n is the ADC number ADC number relation is described below- RX ADC0 - 0 RX ADC1 - 1 RX0 ADC0 - 2 RX0 ADC1 - 3
23-18 —	Reserved.
17-16 T3SET	Powerup start-up time control setting. Set T3SET=10 in normal operation

Table continues on the next page...

Table continued from the previous page...

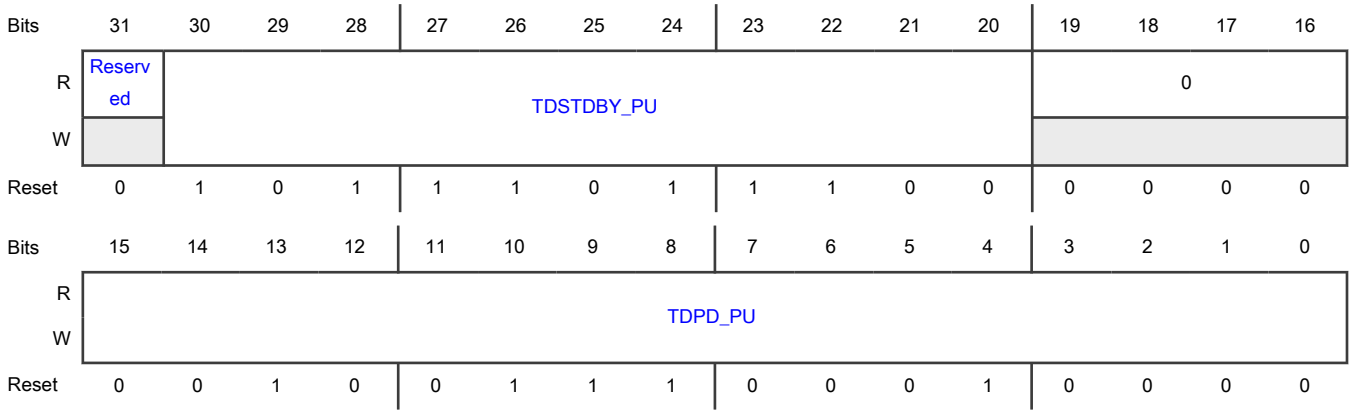
Field	Function
15-14 —	Reserved.
13-12 T2SET	Standby start-up time control setting. Set T2SET=11 in normal operation
11-8 —	Reserved.
7-0 T1SET	Master start-up time control setting. Set T1SET=10010011 for F_{clk}=153.6MSPS and PowerUpTime=15µs in normal operation To respect the startup time specifications, a minimum number of clock cycles shall be set according to the following formula: # clk cycles > *PowerUpTime* * F_{clk} *PowerUpTime* is the power up time from complete shutdown in seconds, and F_{clk} is the clock frequency in Hertz. The number of clock cycles is set using T1SET[6:0] according to the following formulas: Coarse Mode, F_{clk} > 60 MHz, T1SET[7]=1 # clk cycles = T1SET[6] * 2^{13} + T1SET[5] * 2^{12} + ... + T1SET[0] * 2^{7} Fine Mode, F_{clk} <= to 60 MHz, T1SET[7]=0 # clk cycles = T1SET[6] * 2^{9} + T1SET[5] * 2^{8} + ... + T1SET[0] * 2^{3}

9.5.1.26 High Speed DAC Timing Control Register1 (HSDAC_TMNGCTL1)

Offset

Register	Offset
HSDAC_TMNGCTL1	230h

Diagram



Fields

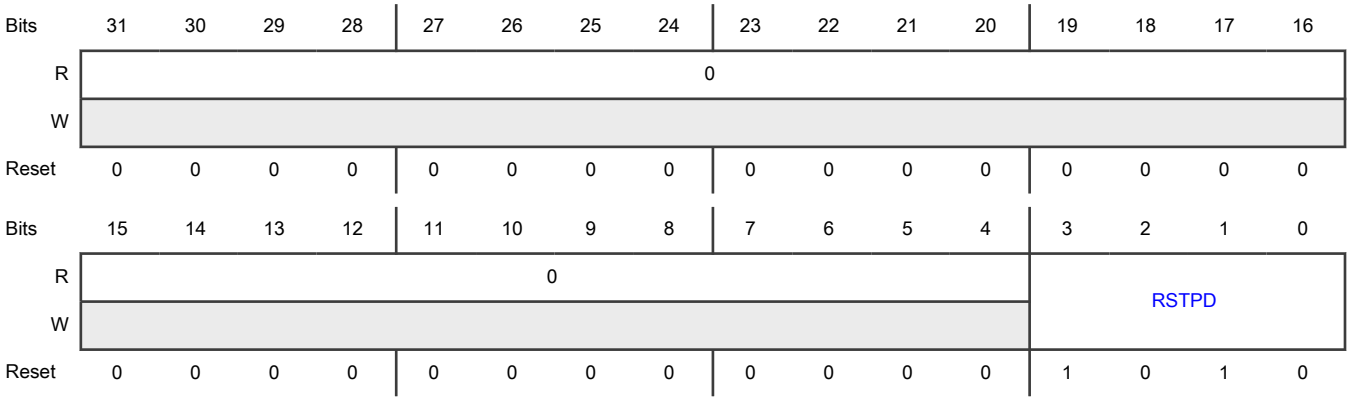
Field	Function
31 —	Reserved.
30-20 TDSTDBY_PU	This field indicates the time it takes for the DAC to assert adcrdy signal when it is moved from sleep/standby state to active state. This duration is to be programmed in terms of number of DAC clock cycles
19-16 —	Reserved.
15-0 TDPD_PU	This field indicates the time it takes for the DAC to assert adcrdy signal when it is moved from powerdown to active state. This duration is to be programmed in terms of number of DAC clock cycles To respect the startup time specifications, a minimum number of clock cycles shall be set according to the following formula: # clk cycles > *PowerUpTime* * F_{clk} *PowerUpTime* is the power up time from complete shutdown in seconds, and F_{clk} is the clock frequency in Hertz. *(Refer to DAC Databook for the value of PowerUpTime).*

9.5.1.27 High Speed DAC Timing Control Register2 (HSDAC_TMNGCTL2)

Offset

Register	Offset
HSDAC_TMNGCTL2	234h

Diagram



Fields

Field	Function
31-4 —	Reserved.
3-0 RSTPD	DAC Reset Pulse Duration This field is used to program the duration for which reset needs to be asserted to the DAC. This duration is to be programmed in terms of number of DAC clock cycles

9.5.1.28 High Speed DAC Enable Control Register (HSDAC_ENCTL)

Offset

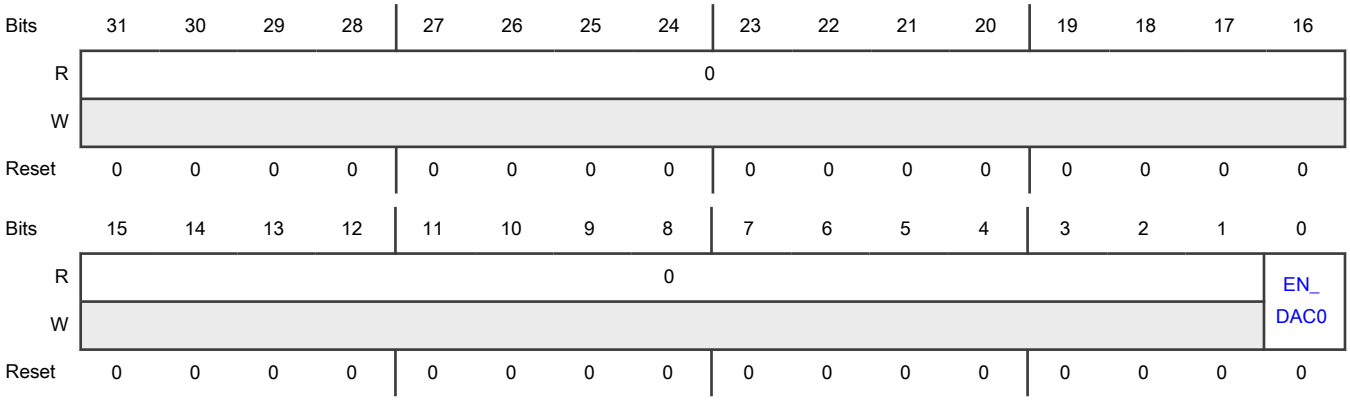
Register	Offset
HSDAC_ENCTL	250h

Function

The bit in this register is expected to be set only once the high speed DAC have powered-up.

Clearing of the bit will result in the DAC being brought down to its powerdown state.

Diagram



Fields

Field	Function
31-1 —	Reserved.
0 EN_DAC0	Enable DAC 0

9.5.1.29 ADC DAC Clock Configuration Register (ADC_DAC_CLKCFG)

Offset

Register	Offset
ADC_DAC_CLKCFG	300h

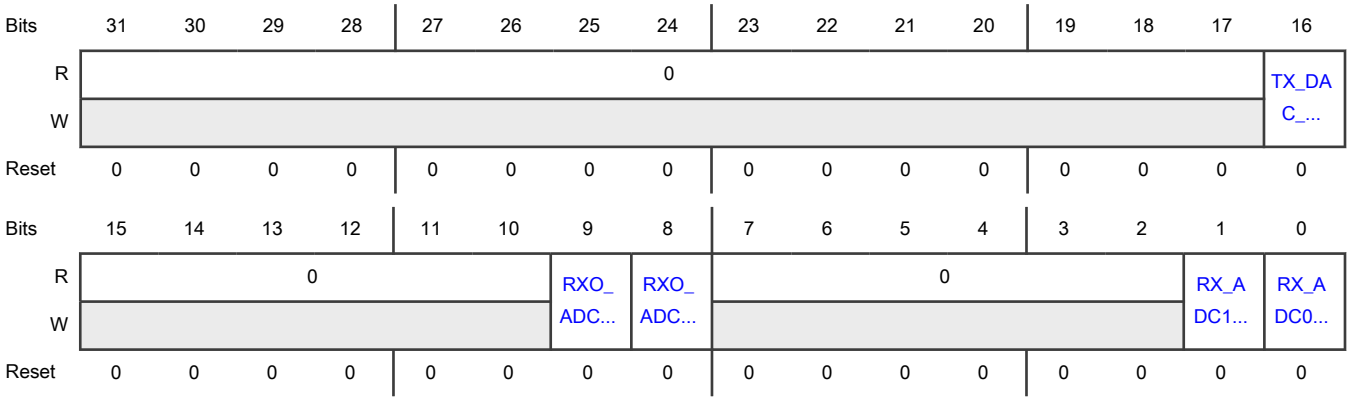
Function

This register is used to select the incoming sync signal to divide the external clock(153.6 MHz).

1 : Divide by 2

0 : Divide by 1

Diagram



Fields

Field	Function
31-17 —	Reserved.
16 TX_DAC_CLK_DIV	This bit is used to select the incoming sync signal to divide the external clock for DAC 1 : Divide by 2 0 : Divide by 1
15-10 —	Reserved.
9 RXO_ADC1_CLK_DIV	This bit is used to select the incoming sync signal to divide the external clock for RXO_ADC1 1 : Divide by 2 0 : Divide by 1
8 RXO_ADC0_CLK_DIV	This bit is used to select the incoming sync signal to divide the external clock for RXO_ADC0 1 : Divide by 2 0 : Divide by 1
7-2 —	Reserved.
1 RX_ADC1_CLK_DIV	This bit is used to select the incoming sync signal to divide the external clock for RX_ADC1 1 : Divide by 2 0 : Divide by 1
0 RX_ADC0_CLK_DIV	This bit is used to select the incoming sync signal to divide the external clock for RX_ADC0 1 : Divide by 2 0 : Divide by 1

9.5.1.30 AUX ADC Clock Configuration Register (AUX_ADC_CLKCFG)

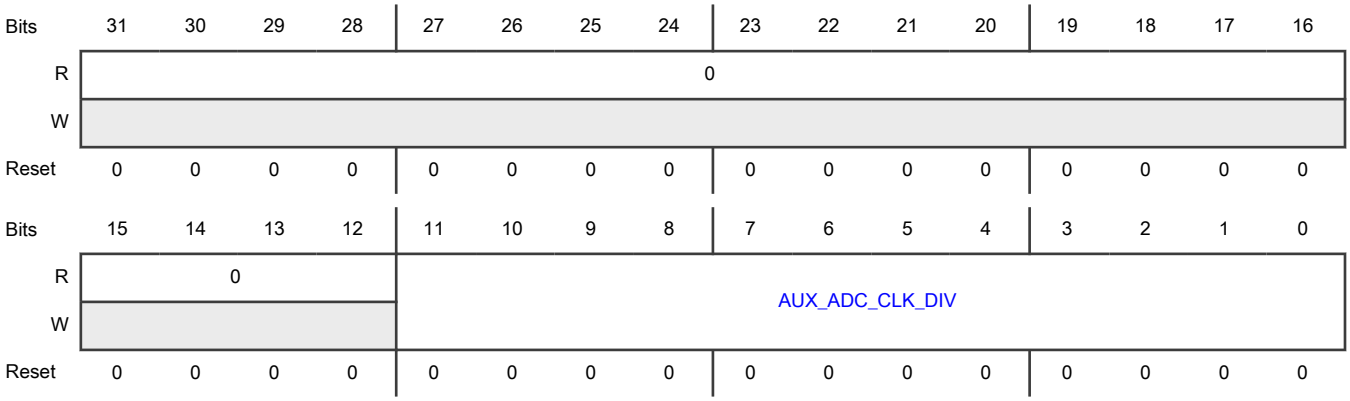
Offset

Register	Offset
AUX_ADC_CLKCFG	308h

Function

This register is used to set the clock divider value for AUX ADC.

Diagram



Fields

Field	Function
31-12 —	Reserved.
11-0 AUX_ADC_CLK_DIV	This field specifies the even integral factor by which the input clock needs to be divided in order to generate the AUX ADC clock. Odd clock division ratio is not supported 000000000000b - Divide by 4096 000000000010b - Divide by 2 111111111100b - Divide by 4092 ... 111111111110b - Divide by 4094

9.5.1.31 Clock Control Register (CLK_CTRL)

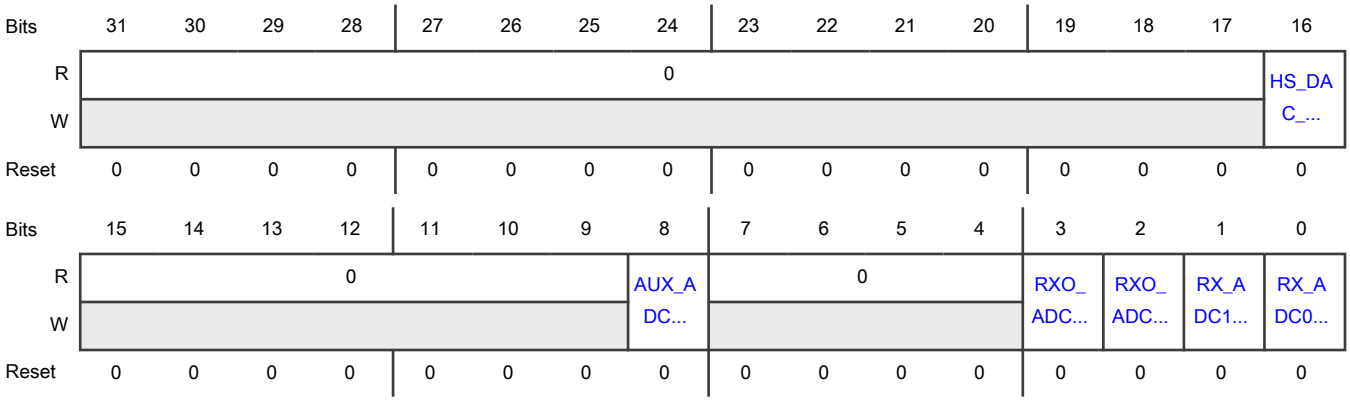
Offset

Register	Offset
CLK_CTRL	310h

Function

This register is used to enable clock divider for ADC and DAC.

Diagram



Fields

Field	Function
31-17 —	Reserved.
16 HS_DAC_CLK_DIV_EN	0: The clock control module gates the divided clock to the corresponding data converter. 1: The clock control module enables the divided clock to the corresponding data converter.
15-9 —	Reserved.
8 AUX_ADC_CLK_DIV_EN	0: The clock divider gates the output divided clock to the AUX ADC. 1: The clock divider enables the output divided clock to the AUX ADC.
7-4 —	Reserved.
3 RXO_ADC1_CLK_DIV_EN	0: The clock control module gates the divided clock to the corresponding data converter. 1: The clock control module enables the divided clock to the corresponding data converter.
2 RXO_ADC0_CLK_DIV_EN	0: The clock control module gates the divided clock to the corresponding data converter. 1: The clock control module enables the divided clock to the corresponding data converter.
1	0: The clock control module gates the divided clock to the corresponding data converter. 1: The clock control module enables the divided clock to the corresponding data converter.

Table continues on the next page...

Table continued from the previous page...

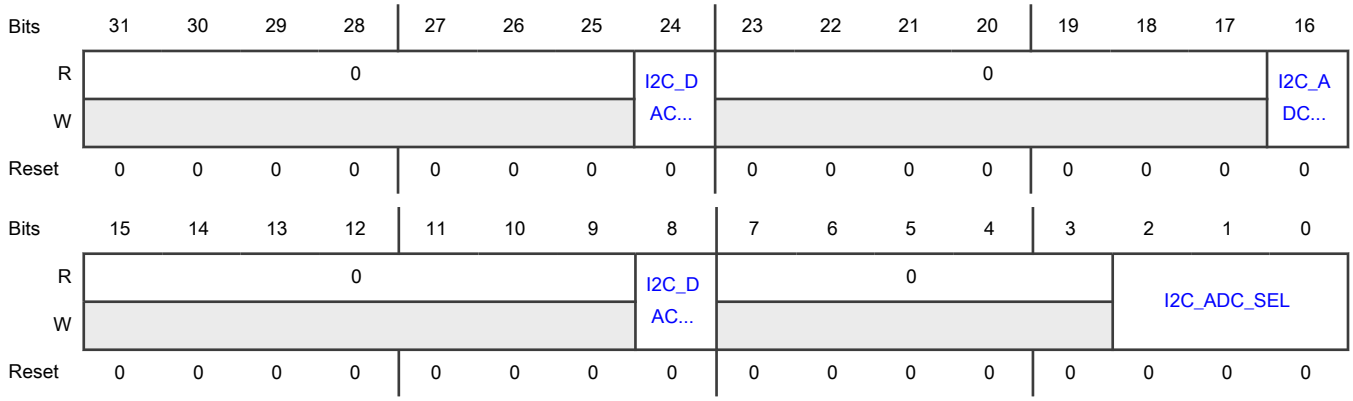
Field	Function
RX_ADC1_CLK_DIV_EN	
0 RX_ADC0_CLK_DIV_EN	0: The clock control module disables the divided clock to the corresponding data converter. 1: The clock control module enables the divided clock to the corresponding data converter.

9.5.1.32 I2C Operation Control Register (I2C_OPCTL)

Offset

Register	Offset
I2C_OPCTL	318h

Diagram



Fields

Field	Function
31-25 —	Reserved.
24 I2C_DAC_EN	Enable I2C access to DACs 0b - I2C access is disabled 1b - I2C access is enabled
23-17 —	Reserved.

Table continues on the next page...

Table continued from the previous page...

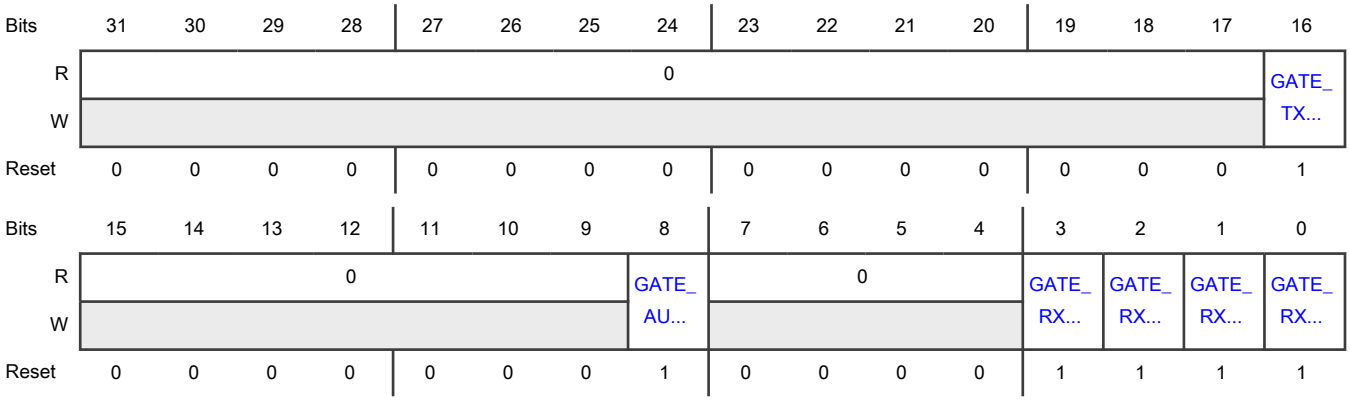
Field	Function
16 I2C_ADC_EN	Enable I2C access to ADCs 0b - I2C access is disabled 1b - I2C access is enabled
15-9 —	Reserved.
8 I2C_DAC_SEL	Select the DAC for I2C transaction. 0x1: DAC is selected
7-3 —	Reserved.
2-0 I2C_ADC_SEL	Select the ADC to which the I2C transaction is to be routed. 0x111: RXO ADC1 channel Q is selected 0x110: RXO ADC1 channel I is selected 0x101: RXO ADC0 channel Q is selected 0x100: RXO ADC0 channel I is selected 0x011: RX ADC1 channel Q is selected 0x010: RX ADC1 channel I is selected 0x001: RX ADC0 channel Q is selected 0x000: RX ADC0 channel I is selected

9.5.1.33 Gate Ready Register (GATE_READY)

Offset

Register	Offset
GATE_READY	320h

Diagram



Fields

Field	Function
31-17 —	Reserved.
16 GATE_TX_DAC_RDY	0: The gate ready field allows propagation of the ready signal from the data converter to the corresponding AXIQ I/F. 1: The gate ready field gates the propagation of the ready signal (drives a 0) from the data converter to the corresponding AXIQ I/F.
15-9 —	Reserved.
8 GATE_AUX_ADC_RDY	0: The gate ready field allows propagation of the ready signal from the data converter to the corresponding AXIQ I/F. 1: The gate ready field gates the propagation of the ready signal (drives a 0) from the data converter to the corresponding AXIQ I/F.
7-4 —	Reserved.
3 GATE_RXO_ADC1_RDY	0: The gate ready field allows propagation of the ready signal from the data converter to the corresponding AXIQ I/F. 1: The gate ready field gates the propagation of the ready signal (drives a 0) from the data converter to the corresponding AXIQ I/F.
2 GATE_RXO_ADC0_RDY	0: The gate ready field allows propagation of the ready signal from the data converter to the corresponding AXIQ I/F. 1: The gate ready field gates the propagation of the ready signal (drives a 0) from the data converter to the corresponding AXIQ I/F.
1	0: The gate ready field allows propagation of the ready signal from the data converter to the corresponding AXIQ I/F.

Table continues on the next page...

Table continued from the previous page...

Field	Function
GATE_RX_ADC1_RDY	1: The gate ready field gates the propagation of the ready signal (drives a 0) from the data converter to the corresponding AXIQ I/F.
0 GATE_RX_ADC0_RDY	0: The gate ready field allows propagation of the ready signal from the data converter to the corresponding AXIQ I/F. 1: The gate ready field gates the propagation of the ready signal (drives a 0) from the data converter to the corresponding AXIQ I/F.

9.5.1.34 Subsystem Timeout Interrupt Enable Register (SUBSYS_INTREN)

Offset

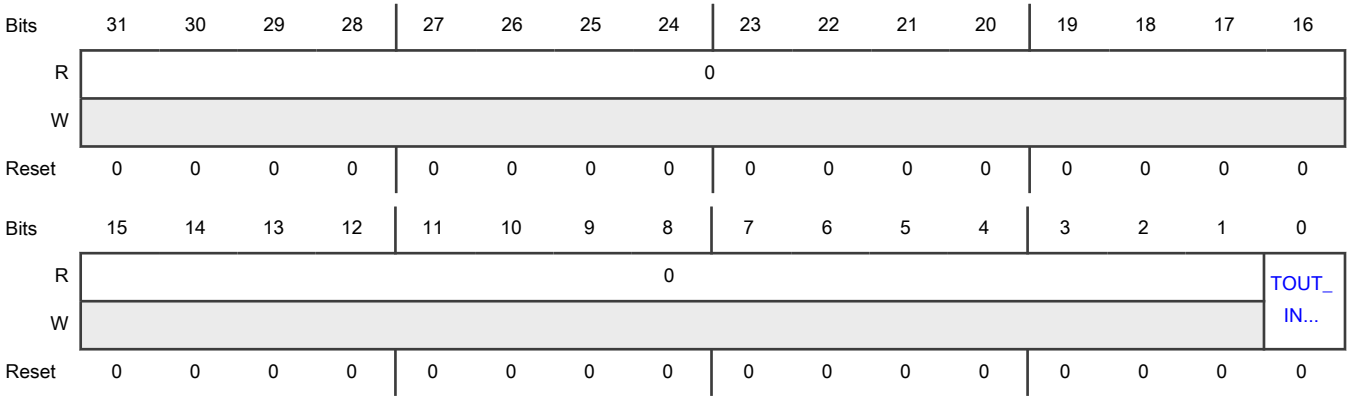
Register	Offset
SUBSYS_INTREN	330h

Function

This register is used to enable interrupt generation from the ADC-DAC subsytem in case of a timeout error.

A common interrupt is generated for various timeout events from different ADCs and DAC.

Diagram



Fields

Field	Function
31-1 —	Reserved.
0 TOUT_INTR_EN	1b - Interrupt generation is enabled 0b - Interrupt generation is disabled

9.5.1.35 Subsystem Interrupt Type Register (SUBSYS_INTRTYP)

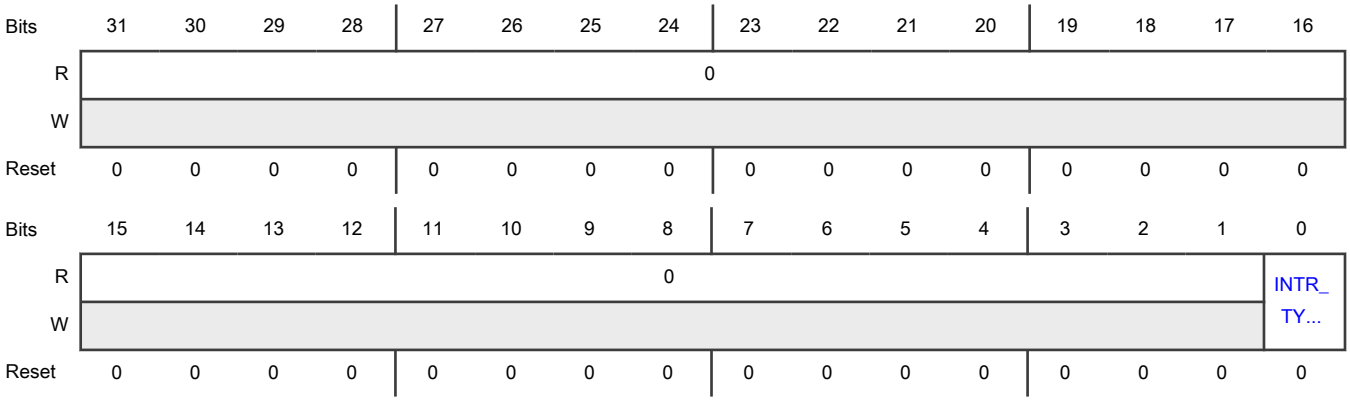
Offset

Register	Offset
SUBSYS_INTRTYP	340h

Function

This register is used to program the type of interrupt that is generated from the ADC-DAC subsytem in case of a timeout error when interrupt generation is enabled.

Diagram



Fields

Field	Function
31-1 —	Reserved.
0 INTR_TYPE	1b - Interrupt generated is a pulse interrupt 0b - Interrupt generated is a level interrupt

9.5.1.36 High Speed ADC Status Register (HSADC_STAT)

Offset

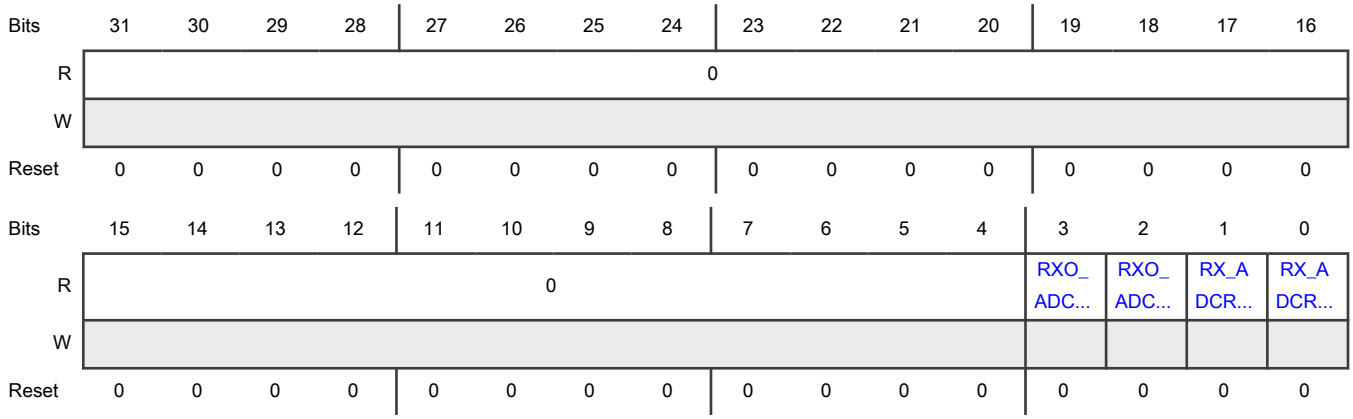
Register	Offset
HSADC_STAT	400h

Function

This field denotes the high speed ADC ready flag.

When asserted, indicates that the ADC is ready to sample the input

Diagram



Fields

Field	Function
31-4 —	Reserved.
3 RXO_ADCRDY_1	Ready signal for RXO ADC 1
2 RXO_ADCRDY_0	Ready signal for RXO ADC 0
1 RX_ADCRDY_1	Ready signal for RX ADC 1
0 RX_ADCRDY_0	Ready signal for RX ADC 0

9.5.1.37 High Speed ADC Error Status Register1 (HSADC_ERRSTAT1)

Offset

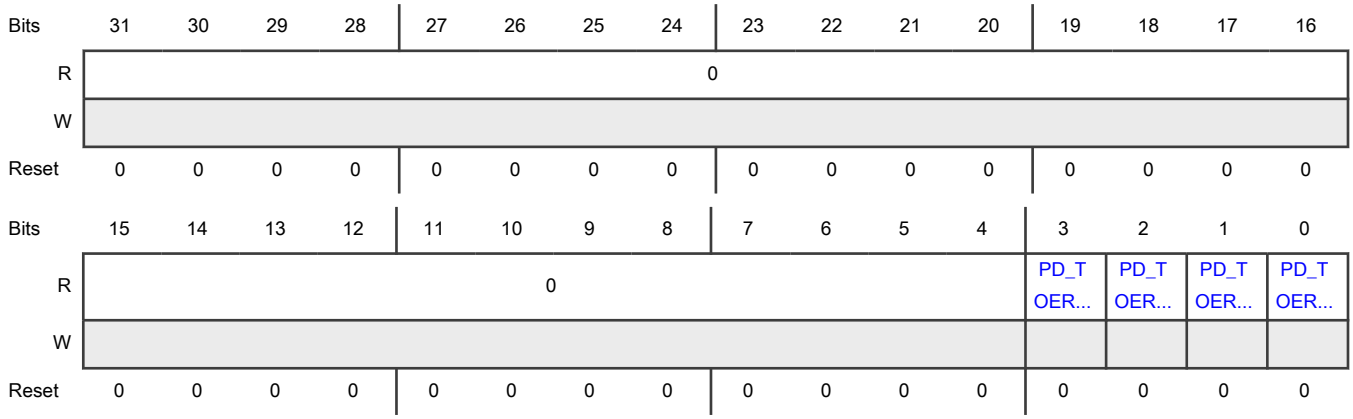
Register	Offset
HSADC_ERRSTAT1	408h

Function

This register reflects the status of the adcrdy bit for various high speed ADCs after the ADC has been moved from powerdown to active state.

When set, this bit indicates that adcrdy for the corresponding ADC was not asserted within the time period specified in the TDPD_PU field of the HSADC_TMNGCTL1 register

Diagram



Fields

Field	Function
31-4 —	Reserved.
3 PD_TOERR_RXO_ADC1	When set, this bit indicates that adcrdy from RXO ADC1 was not received when moving from powerdown to active state and hence, a timeout error has occured.
2 PD_TOERR_RXO_ADC0	When set, this bit indicates that adcrdy from RXO ADC0 was not received when moving from powerdown to active state and hence, a timeout error has occured.
1 PD_TOERR_RX_ADC1	When set, this bit indicates that adcrdy from RX ADC1 was not received when moving from powerdown to active state and hence, a timeout error has occured.
0 PD_TOERR_RX_ADC0	When set, this bit indicates that adcrdy from RX ADC0 was not received when moving from powerdown to active state and hence, a timeout error has occured.

9.5.1.38 High Speed ADC Error Status Register2 (HSADC_ERRSTAT2)

Offset

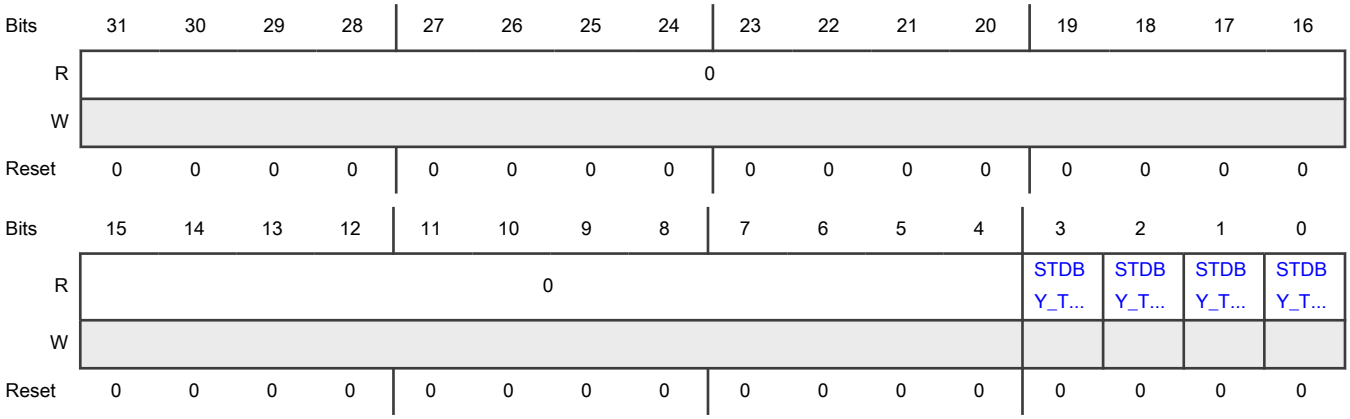
Register	Offset
HSADC_ERRSTAT2	40Ch

Function

This register reflects the status of the adcrdy bit for various high speed ADCs after the ADC has been moved from sleep/standby mode to active mode.

When set, this bit indicates that adcrdy for the corresponding ADC was not asserted within the time period specified in the TDSTDBY_PU field of the HSADC_TMNGCTL1 register

Diagram



Fields

Field	Function
31-4 —	Reserved.
3 STDBY_TOERR_RXO_ADC1	When set, this bit indicates that adcrdy from RXO ADC1 was not received when moving from sleep/standby mode to active state and hence, a timeout error has occured.
2 STDBY_TOERR_RXO_ADC0	When set, this bit indicates that adcrdy from RXO ADC0 was not received when moving from sleep/standby mode to active state and hence, a timeout error has occured.
1 STDBY_TOERR_RX_ADC1	When set, this bit indicates that adcrdy from RX ADC1 was not received when moving from sleep/standby mode to active state and hence, a timeout error has occured.
0 STDBY_TOERR_RX_ADC0	When set, this bit indicates that adcrdy from RX ADC0 was not received when moving from sleep/standby mode to active state and hence, a timeout error has occured.

9.5.1.39 Auxiliary ADC Status Register (AUXADC_STAT)

Offset

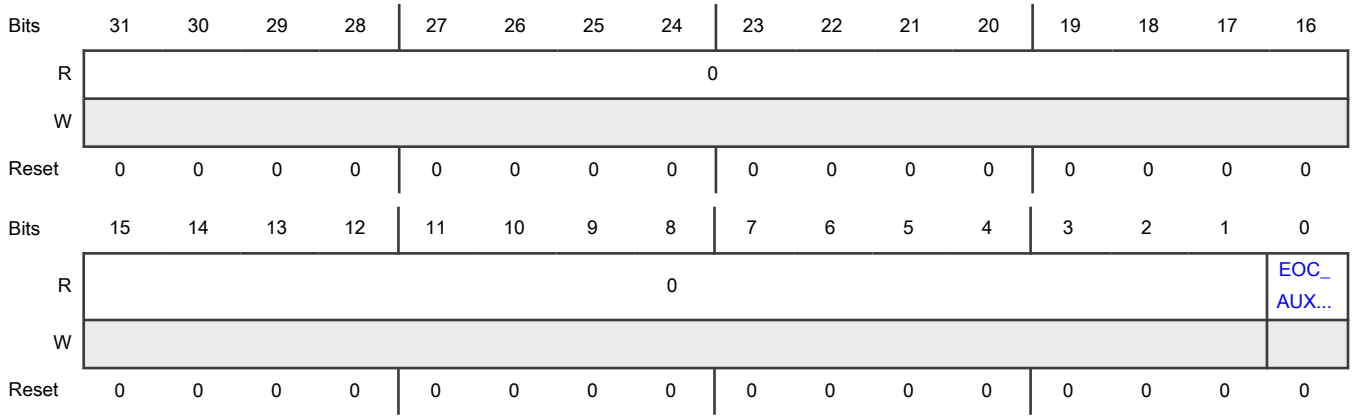
Register	Offset
AUXADC_STAT	420h

Function

This denotes the auxiliary ADCs end of conversion.

The rising edge indicates that conversion terminated and output data is valid.

Diagram



Fields

Field	Function
31-1 —	Reserved.
0 EOC_AUXADC 0	End of conversion for auxiliary ADC 0

9.5.1.40 Auxiliary ADC Error Status Register (AUXADC_ERRSTAT)

Offset

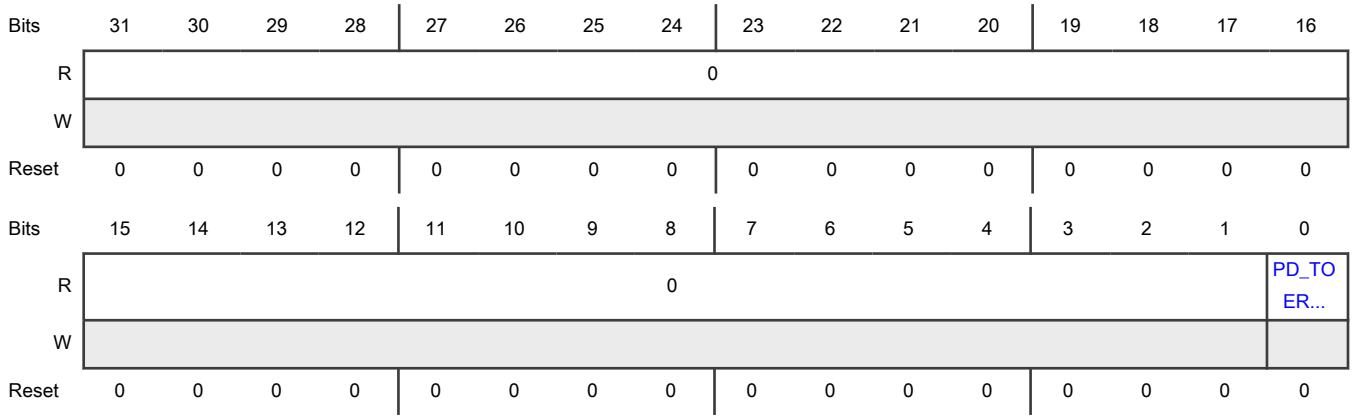
Register	Offset
AUXADC_ERRSTAT	428h

Function

This register reflects the status of the eoc bit for the auxiliary ADCs.

When set, this bit indicates that eoc for the corresponding auxiliary ADC was not asserted within the time period specified in the TPUP field of the AUXADC_TMNGCTL register

Diagram



Fields

Field	Function
31-1 —	Reserved.
0 PD_TOERR_AUX_ADC0	When set, this bit indicates that eoc from auxiliary ADC0 was not received and hence, a timeout error has occured.

9.5.1.41 High Speed DAC Status Register (HSDAC_STAT)

Offset

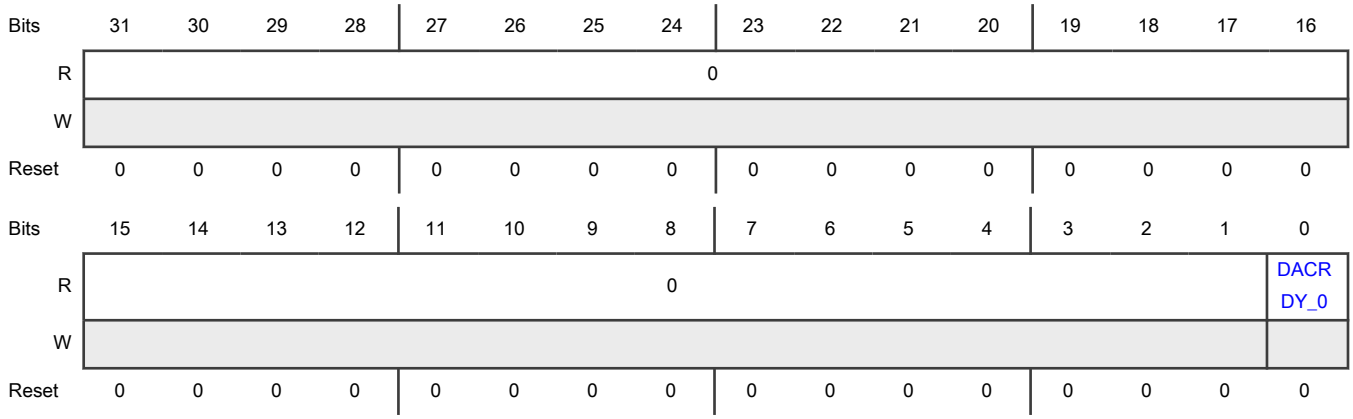
Register	Offset
HSDAC_STAT	440h

Function

This field denotes the high speed DAC ready flag.

This output flags the readiness of the DAC after a powerup (from complete shutdown) or wakeup (from standby) event. The signal goes to 1 after a Tpu or Twu for a powerup or wakeup event respectively, as specified in specifications table.

Diagram



Fields

Field	Function
31-1 —	Reserved.
0 DACRDY_0	Ready signal for DAC 0

9.5.1.42 High Speed DAC Error Status Register1 (HSDAC_ERRSTAT1)

Offset

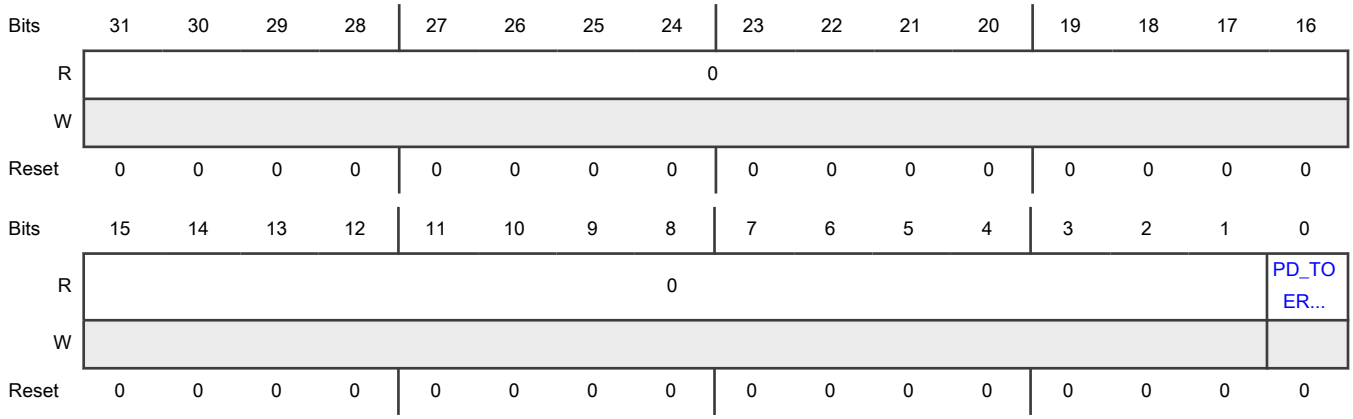
Register	Offset
HSDAC_ERRSTAT1	448h

Function

This register reflects the status of the dacrdy bit for various high speed DACs after the DAC has been moved from powerdown to active mode.

When set, this bit indicates that dacrdy for the corresponding DAC was not asserted within the time period specified in the TDPD_PU field of the HSDAC_TMNGCTL1 register

Diagram



Fields

Field	Function
31-1 —	Reserved.
0 PD_TOERR_DAC0	When set, this bit indicates that dacrdy from DAC0 was not received when moving from powerdown to active state and hence, a timeout error has occured.

9.5.1.43 High Speed DAC Error Status Register2 (HSDAC_ERRSTAT2)

Offset

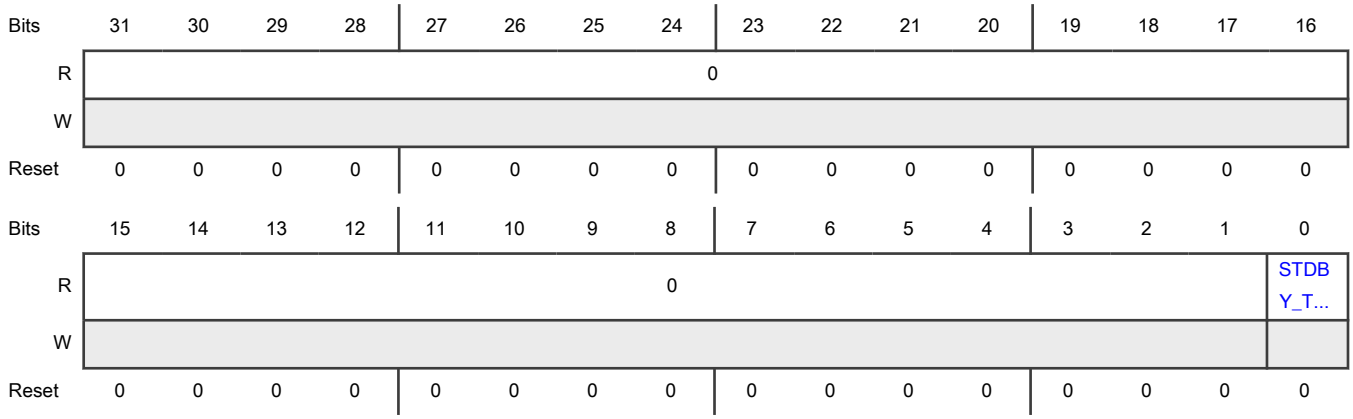
Register	Offset
HSDAC_ERRSTAT2	44Ch

Function

This register reflects the status of the dacrdy bit for various high speed DACs after the DAC has been moved from sleep/standby mode to active mode.

When set, this bit indicates that dacrdy for the corresponding DAC was not asserted within the time period specified in the TDSTDBY_PU field of the HSDAC_TMNGCTL1 register

Diagram



Fields

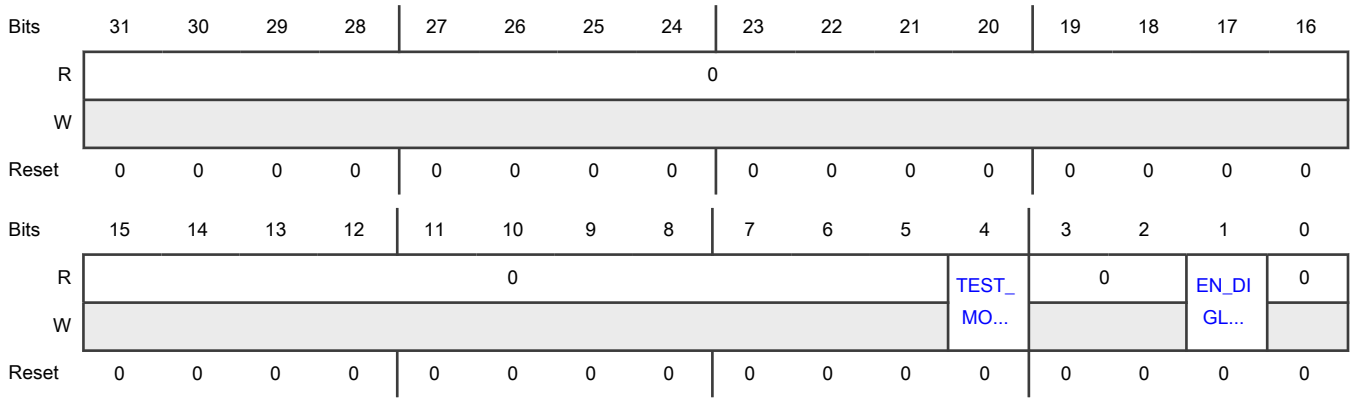
Field	Function
31-1 —	Reserved.
0 STDBY_TOERR_DAC0	When set, this bit indicates that dacrdy from DAC0 was not received when moving from sleep/standby mode to active state and hence, a timeout error has occured.

9.5.1.44 Test Loopback Configuration Control Register (TEST_LPBK_CFGCTL)

Offset

Register	Offset
TEST_LPBK_CFGCTL	C10h

Diagram



Fields

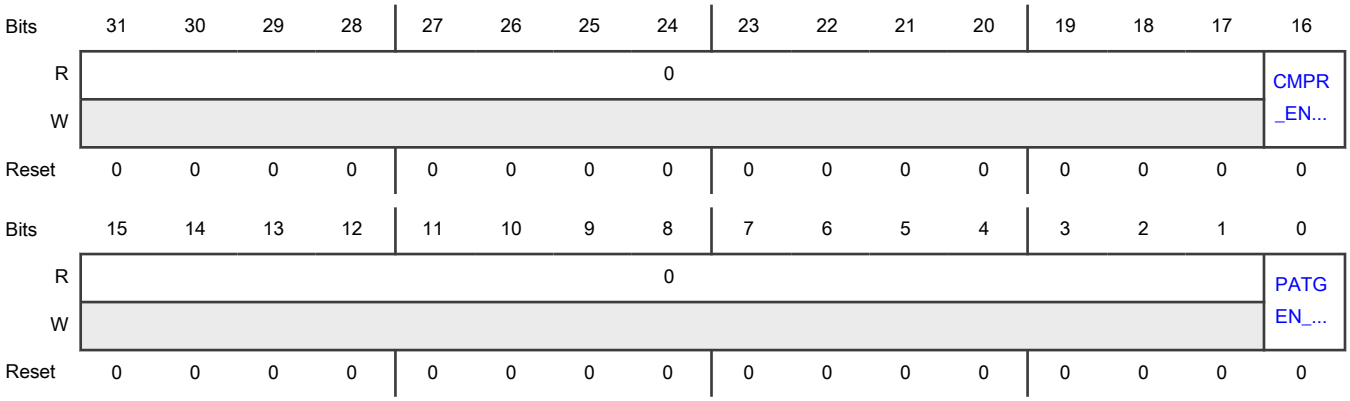
Field	Function
31-5 —	Reserved.
4 TEST_MODE_SEL	Enable test mode for DAC input. If the bit is 1, DAC input is either from pattern generator or looped back from ADC
3-2 —	Reserved.
1 EN_DIGLPBK	Enable digital loopback This is used for testing. Digital loopback is formed within ADC-DAC when this bit is enabled.
0 —	Reserved.

9.5.1.45 Test Configuration Control Register1 (TEST_CFGCTL1)

Offset

Register	Offset
TEST_CFGCTL1	C20h

Diagram



Fields

Field	Function
31-17	Reserved.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
16 CMPR_EN_PAIR_0	Enable the pattern comparator.
15-1 —	Reserved.
0 PATGEN_EN_PAIR_0	Enable the pattern generator for DAC 0. The bit clears automatically once all patterns are driven and the comparator has compared the last pattern.

9.5.1.46 Test Configuration Control Register2 (TEST_CFGCTL2)

Offset

Register	Offset
TEST_CFGCTL2	C24h

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										LOOP_CTR_VAL		0		TEST_PATTRN_SEL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-6 —	Reserved.
5-4	Selects the number of iterations for which the selected pattern is to be driven.

Table continues on the next page...

Table continued from the previous page...

Field	Function
LOOP_CTR_VAL	
3-2 —	Reserved.
1-0 TEST_PATTRN_SEL	Select the pattern to be driven Note:Comparator should be disabled when TEST_PATTRN_SEL=10b 00b - Constant DC pattern is driven. 01b - Step input of programmable step size and fixed duration 10b - Step input of prorammable step size and programmable duration

9.5.1.47 Test Configuration Control Register3 (TEST_CFGCTL3)

Offset

Register	Offset
TEST_CFGCTL3	C28h

Diagram

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

Fields

Field	Function
31-28 —	Reserved.
27-16	The last pattern value to be driven. It must be greater than the first pattern and obey following rule : LAST_PATTRN = FIRST_PATTRN + n * PATTRN_STEP_SIZE (where n represents non-zero integer)

Table continues on the next page...

Table continued from the previous page...

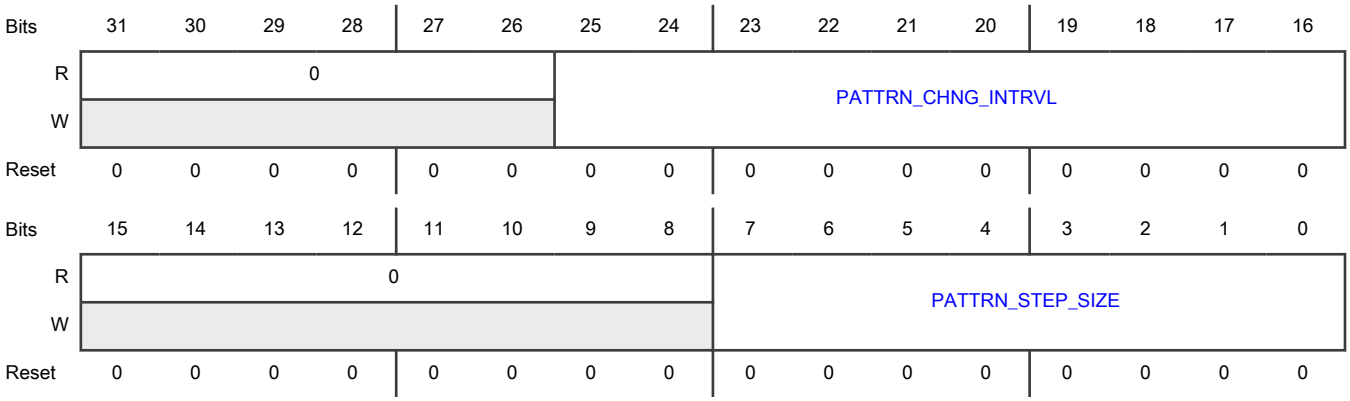
Field	Function
LAST_PATTRN	
15-12 —	Reserved.
11-0 FIRST_PATTRN	The first pattern value to be driven. For DC pattern type, this will be the only value driven.

9.5.1.48 Test Configuration Control Register4 (TEST_CFGCTL4)

Offset

Register	Offset
TEST_CFGCTL4	C2Ch

Diagram



Fields

Field	Function
31-26 —	Reserved.
25-16 PATTRN_CHNG_INTRVL	For TEST_PATTRN_SEL=10b, the number of DAC clock cycles after which pattern changes. The minimum value of this field is 1.
15-8	Reserved.

Table continues on the next page...

Table continued from the previous page...

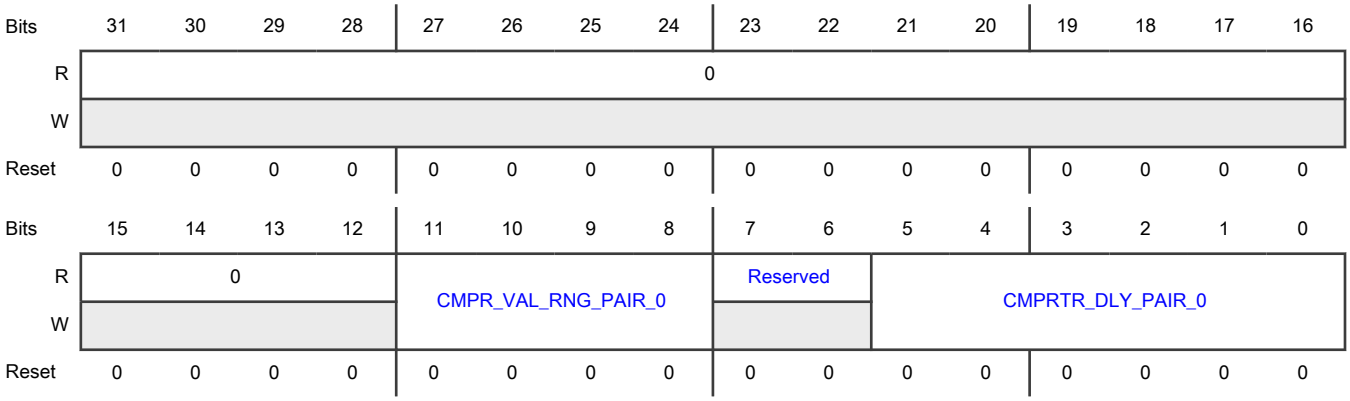
Field	Function
—	
7-0 PATTRN_STEP_SIZE	Step size for generated pattern when TEST_PATTRN_SEL is not 00b. Step size is the difference between consecutive patterns generated.

9.5.1.49 Test Configuration Control Register5 (TEST_CFGCTL5)

Offset

Register	Offset
TEST_CFGCTL5	C30h

Diagram



Fields

Field	Function
31-12 —	Reserved.
11-8 CMPR_VAL_RNG_PAIR_0	Permissible difference in the actual and expected data for comparator. When expected data is within range of actual_data +/-CMPR_VAL_RNG_PAIR_0 then the comparison is considered as pass and PASS_CNT_I/Q_PAIR_0 is incremented.
7-6 —	Reserved.

Table continues on the next page...

Table continued from the previous page...

Field	Function
5-0 CMPRTR_DLY_PAIR_0	Number of ADC clock cycles after which the comparator should compare the data at its input with expected data. It should be greater than the DAC-ADC loopback latency. Loopback latency is the number of ADC clock cycles taken by pattern to travel from pattern generator to comparator.

9.5.1.50 Test MUX Control Register (TEST_MUXCTL)

Offset

Register	Offset
TEST_MUXCTL	C34h

Diagram

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

Fields

Field	Function
31-27 —	Reserved.
26-24 TEST_LPBK_ADC_SEL	Selects which ADC to use for loopback with DAC 000b - Selects clock and data of RX0 to be sent to comparator 001b - Selects clock and data of RX1 to be sent to comparator 010b - Selects clock and data of RO0 to be sent to comparator 011b - Selects clock and data of RO1 to be sent to comparator 100b - Selects clock and data of AUX ADC to be sent to comparator 101b - Reserved

Table continues on the next page...

Table continued from the previous page...

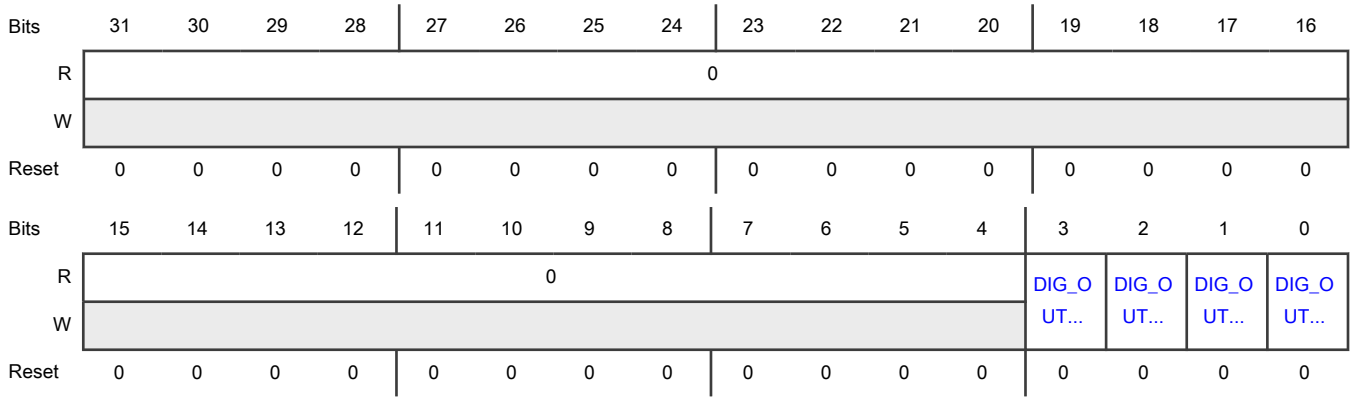
Field	Function
	110b - Reserved 111b - The clock to the comparator is gated
23-10 —	Reserved.
9-8 —	Reserved.
7-2 —	Reserved.
1-0 —	Reserved.

9.5.1.51 Test Group Digital Output Mux Control Register (TEST_GRP_DOMUXCTL)

Offset

Register	Offset
TEST_GRP_DOMUXCTL	F20h

Diagram



Fields

Field	Function
31-4	Reserved.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
3 DIG_OUT_MUX_GSEL_3	
2 DIG_OUT_MUX_GSEL_2	
1 DIG_OUT_MUX_GSEL_1	
0 DIG_OUT_MUX_GSEL_0	

9.5.1.52 Test Miscellaneous Digital Output Mux Control Register (TEST_MISC_DOMUXCTL)

Offset

Register	Offset
TEST_MISC_DOMUXCTL	F24h

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												Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	u	u	u	u

Fields

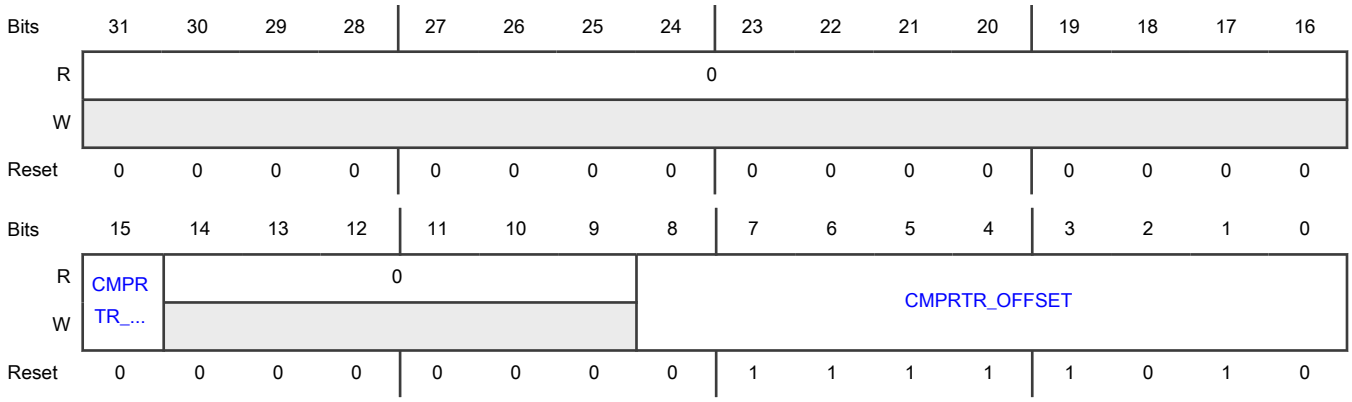
Field	Function
31-4 —	Reserved.
3-0 —	Reserved.

9.5.1.53 Test Offset Configuration Register (TEST_OFFSET_CFG)

Offset

Register	Offset
TEST_OFFSET_CFG	F28h

Diagram



Fields

Field	Function
31-16 —	Reserved.
15 CMPRTR_OFFSET_EN	Enable consideration of the offset value (CMPRTR_OFFSET) in comparison between expected and actual value done by comparator. This functionality is particularly useful when loopback is enabled for AUX ADC.
14-9 —	Reserved.

Table continues on the next page...

Table continued from the previous page...

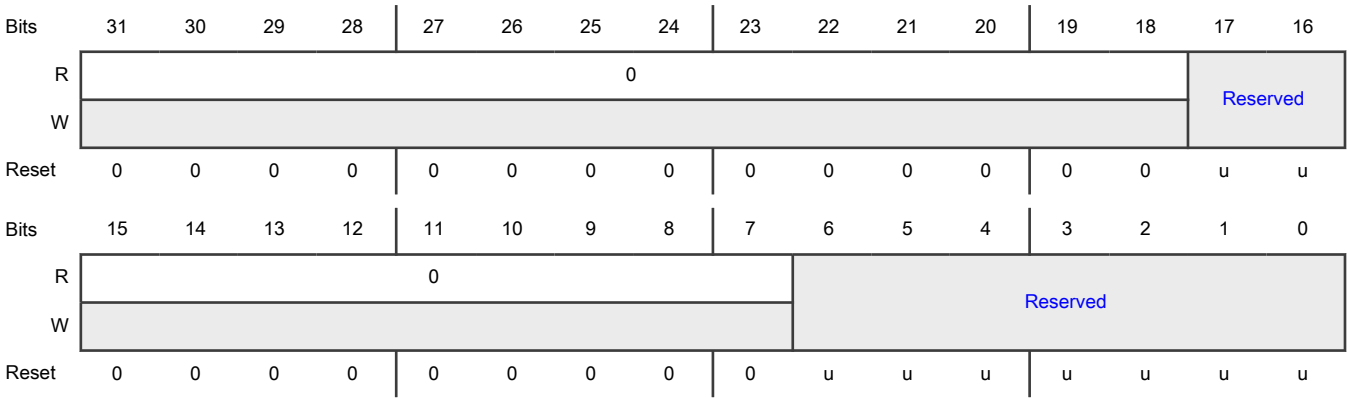
Field	Function
8-0 CMPRTR_OFFSET	Set the expected offset between the values driven by pattern generator and values received at comparator. When CMPRTR_OFFSET is set comparator adds this value to the expected value before comparing it with the received value. This functionality is particularly useful when loopback is enabled for AUX ADC. For AUX ADC it is should be set to 250 for 12-bit resolution.

9.5.1.54 Test Pin Expose Configuration Register (TEST_PIN_EXPOSE_CFG)

Offset

Register	Offset
TEST_PIN_EXPOSE_CFG	F2Ch

Diagram



Fields

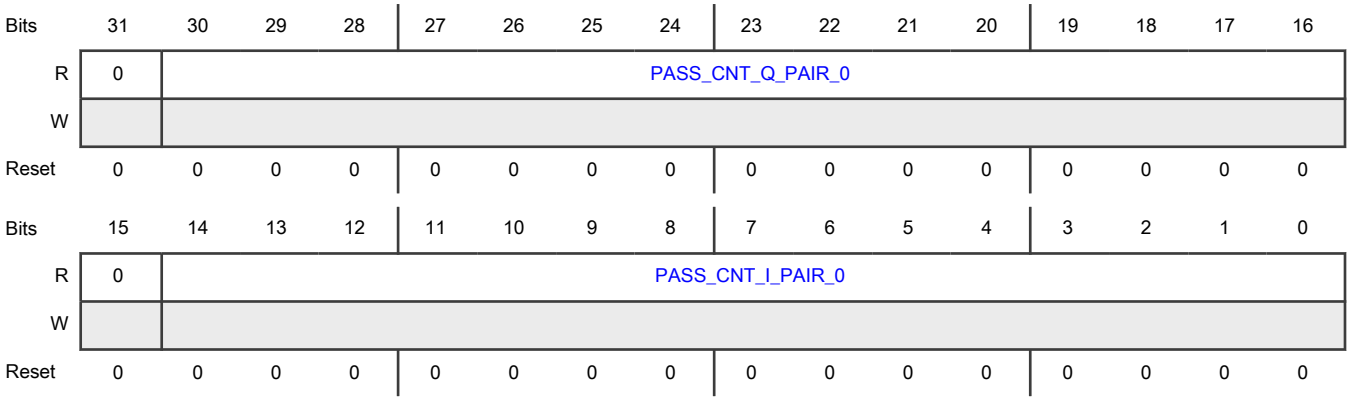
Field	Function
31-18 —	Reserved.
17-16 —	Reserved.
15-7 —	Reserved.
6-0 —	Reserved.

9.5.1.55 Test Status Register (TEST_STAT)

Offset

Register	Offset
TEST_STAT	F30h

Diagram



Fields

Field	Function
31 —	Reserved.
30-16 PASS_CNT_Q_PAIR_0	Number of patterns on selected ADC-DAC pair Q-channel, for which difference between expected and actual data is within the specified range(CMPR_VAL_RNG_PAIR_0). When AUX ADC is selected CMPRTR_OFFSET is added to the actual data before comparison.
15 —	Reserved.
14-0 PASS_CNT_I_PAIR_0	Number of patterns on selected ADC-DAC pair I-channel, for which difference between expected and actual data is within the specified range(CMPR_VAL_RNG_PAIR_0). When AUX ADC is selected CMPRTR_OFFSET is added to the actual data before comparison.

9.5.1.56 Register Space Synchronize Register (REG_SPACE_SYNC)

Offset

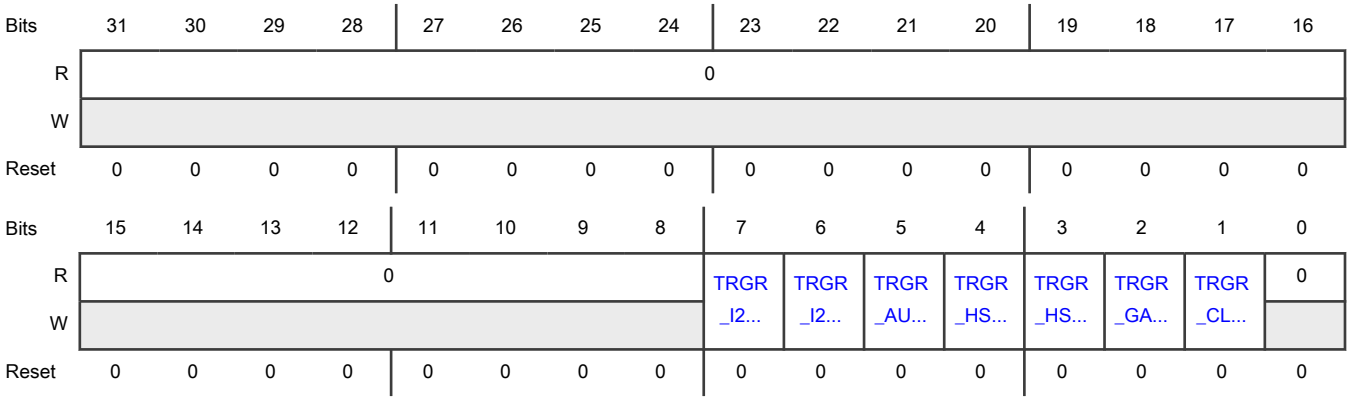
Register	Offset
REG_SPACE_SYNC	F34h

Function

This register consists of trigger bits used to synchronize the register programming into the corresponding clock domain.

Once set, hardware blocks all skyblue writes to this bit until it is cleared.

Diagram



Fields

Field	Function
31-8 —	Reserved.
7 TRGR_I2C_DAC_SYNC	This bit is expected to be set by the user once all the programming registers pertaining to the DAC I2C access have been written. Upon setting this trigger bit, the hardware synchronizes the I2C register bits into corresponding clock domain. This is a self-clearing bit and is cleared once these register bits have been synchronized. It is expected that the user will poll this bit after setting it and will not perform write operations on the I2C register until this bit is cleared by hardware.
6 TRGR_I2C_ADC_SYNC	This bit is expected to be set by the user once all the programming registers pertaining to the ADC I2C access have been written. Upon setting this trigger bit, the hardware synchronizes the I2C register bits into corresponding clock domain. This is a self-clearing bit and is cleared once these register bits have been synchronized. It is expected that the user will poll this bit after setting it and will not perform write operations on the I2C register until this bit is cleared by hardware.
5 TRGR_AUXADC_SYNC	This bit is expected to be set by the user once all the programming registers pertaining to the auxiliary ADC have been written. Upon setting this trigger bit, the hardware synchronizes the auxiliary ADC register bits into auxiliary ADC State Machine’s corresponding clock domain. This is a self-clearing bit and is cleared once these register bits have been synchronized. It is expected that the user will poll this bit after setting it and will not perform write operations on the auxiliary ADC register until this bit is cleared by hardware.

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 TRGR_HSDAC_SYNC	This bit is expected to be set by the user once all the programming registers pertaining to the high speed DAC have been written. Upon setting this trigger bit, the hardware synchronizes the high speed DAC register bits into high speed DAC State Machine's corresponding clock domain. This is a self-clearing bit and is cleared once these register bits have been synchronized. It is expected that the user will poll this bit after setting it and will not perform write operations on the high speed DAC registers until this bit is cleared by hardware.
3 TRGR_HSADC_SYNC	This bit is expected to be set by the user once all the programming registers pertaining to any/all of the high speed ADCs have been written. Upon setting this trigger bit, the hardware synchronizes all the high speed ADC register bits into each high speed ADC State Machine's corresponding clock domain. This is a self-clearing bit and is cleared once these register bits have been synchronized. It is expected that the user will poll this bit after setting it and will not perform write operations on any of the high speed ADC registers until this bit is cleared by hardware.
2 TRGR_GATE_RDY_SYNC	This bit is expected to be set by the user once the gate ready programming register pertaining to any/all of the ADC/DACs have been written. Upon setting this trigger bit, the hardware synchronizes all ADC/DAC gate ready register bits into each ADC/DAC State Machine's corresponding clock domain. This is a self-clearing bit and is cleared once these register bits have been synchronized. It is expected that the user will poll this bit after setting it and will not perform write operations on the gate ready registers until this bit is cleared by hardware.
1 TRGR_CLKDIV_SYNC	This bit is expected to be set by the user when configuring the clock divider and clock enables for the clock control modules of all RX/RXO ADCs/DAC. Upon setting this trigger bit, the hardware synchronizes all the register bits (clock division and clock enable) associated with the various clock control modules into the platform clock domain. This is a self-clearing bit and is cleared once all register bits associated with various clock dividers have been synchronized. It is expected that the user will poll this bit after setting it and will not perform write operations on any of the registers associated with any clock divider until this bit is cleared by hardware.
0 —	Reserved.

9.6 Functional description

The following sections describe functional details of the ADC-DAC module.

9.6.1 Operating modes

This section describes all functional operating modes of the ADC-DAC module.

9.6.1.1 Low power modes

This section describes all low power modes of operation of the ADC_DAC_SUBSYSTEM module. For power saving, the ADC/DAC may be sent to lower power modes when not being used. The exact power consumption and recovery time can be found in their respective datasheets. The ADC supports two low power modes

- Sleep mode- low power saving , less recovery time
- Powerdown mode- more power saving, greater recovery time

Similarly DAC supports two low power modes

- Standby mode
- Powerdown mode

These modes may be entered once the ADC/DAC are in active state. To send high speed ADC to sleep mode, configure HS_RX_ADC0_CFGCTL[OPM] or HS_RX_ADC1_CFGCTL[OPM] or HS_RXO_ADC0_CFGCTL[OPM] or HS_RXO_ADC1_CFGCTL[OPM] to value 2'b01. Set the TRGR_HSADC_SYNC bit of REG_SPACE_SYNC register and poll for its clearing. The corresponding bit in HSADC_STAT register should go low, indicating the adcrdy signal has been deasserted. To send high speed DAC to standby mode, configure HSDAC_CFGCTL1[OPM] to value 3'b111. Set the TRGR_HSDAC_SYNC bit of REG_SPACE_SYNC register and poll for its clearing. The corresponding bit in HSDAC_STAT register should go low, indicating the dacrdy signal has been deasserted. To send the ADC/DAC to powerdown mode, configure the respective fields to value 2'b00/3'b000 in a similar way as mentioned above.

9.6.2 Operations

This section describes the ADC-DAC module’s operations.

To bring up the ADC DAC subsystem following steps must be followed:

1. Once the system is out of POR and the platform clock and external clock are stable set the enable bit of required clock dividers and configure the division ratios.
2. Set the TRGR_CLKDIV_SYNC bit of REG_SPACE_SYNC register and poll for its clearing.
3. Program ADC/DAC for operations, corresponding bit in HSADC_ENCTL/HSDAC_ENCTL must be set. Set the TRGR_HSADC_SYNC/TRGR_HSDAC_SYNC bit of REG_SPACE_SYNC register and poll for its clearing.
4. The ADC/DAC must be reset initially. For this set the corresponding bit in HSADC_RSTCTL/HSDAC_RSTCTL register. Set the TRGR_HSADC_SYNC/TRGR_HSDAC_SYNC bit of REG_SPACE_SYNC register and poll for its clearing. To validate the completion of reset, poll the corresponding bit in HSADC_RSTCTL/HSDAC_RSTCTL register for clearing.
5. To configure ADC for use, control fields/parameters (other than OPM) should be programmed as required. Set the TRGR_HSADC_SYNC bit of REG_SPACE_SYNC register and poll for its clearing. These fields must be configured before configuring the OPM field.
6. Configure HS_RX_ADC0_CFGCTL[OPM] or HS_RX_ADC1_CFGCTL[OPM] or HS_RXO_ADC0_CFGCTL[OPM] or HS_RXO_ADC1_CFGCTL[OPM] to value 2'b11. Other fields in these registers may be set as required. Set the TRGR_HSADC_SYNC bit of REG_SPACE_SYNC register and poll for its clearing.
7. To configure DAC for use, control fields/parameters (other than OPM) should be programmed as required. For DAC, fields t1set,t2set and t3set have to be configured according to the operational frequency. The reset value of these fields is for 153.6 MHz frequency. Once the fields are programmed, set the TRGR_HSDAC_SYNC bit of REG_SPACE_SYNC register and poll for its clearing. These fields must be configured before configuring the OPM field.
8. Configure HSDAC_CFGCTL1[OPM] to value 2'b011. Set the TRGR_HSDAC_SYNC bit of REG_SPACE_SYNC register and poll for its clearing.
9. Once ADC/DAC are configured, poll the corresponding bit in HSDAC_STAT/HSADC_STAT register to know ADC/DAC is ready.
10. Once all the required ADC/DAC ready status is high, clear the corresponding bits in GATE_READY register. Set the TRGR_GATE_RDY_SYNC bit of REG_SPACE_SYNC register and poll for its clearing. Ready signals for all active ADC's are asserted simultaneously on AXIQ interface. Ready signal for DAC is asserted one clock cycle later than active ADC's.
11. To update ADC/DAC sampling frequency put the IP in power-down, change frequency to the desired value and put IP back to active mode.

9.6.3 Clocks

This section describes clocks and special clocking requirements of the ADC-DAC module.

Refer Figure 2 for clocking scheme of the system.

9.6.4 Reset

This section describes how to reset the ADC-DAC module and explains special requirements related to reset.

To reset the ADC_DAC_SUBSYSTEM, the ipg_hard_async_reset_b must be asserted. This will reset all the individual modules within the subsystem. In addition to this the ADC/DACs can be reset individually, if required in the functional case. This can be done by register programming. The intended ADC/DAC must have been enabled.

- Read the HSADC_RSTCTL/HSDAC_RSTCTL register to ensure there is no pending reset on that ADC/DAC.
- If the bit is clear, program the bit to 1. Set the TRGR_HSADC_SYNC/TRGR_HSDAC_SYNC bit of REG_SPACE_SYNC register and poll for its clearing.
- To validate the completion of reset, poll the corresponding bit in HSADC_RSTCTL/HSDAC_RSTCTL register for clearing.

9.6.5 Interrupts

This section describes all the interrupts that the ADC-DAC module generates.

ADC_DAC_SUBSYSTEM generates a single interrupt adc_dac_intr. This is an active high interrupt, generated when a timeout condition occurs for ready signals of data converters. If the data converters do not assert corresponding ready signal within a programmed duration, while recovering to active state from sleep/powerdown/unpowered state, the interrupt is asserted. The time interval for timeout is different for sleep and powerdown/unpowered conditions and is programmable. The interrupt can be disabled by register programming. It can be programmed to be level or pulse interrupt. In case the interrupt is level, it will be deasserted only after all the source data converters have been given soft reset.

9.6.6 Built-in Self Test (BIST) Operation

This section describes the configuration of ADC-DAC module for BIST operations.

The ADC_DAC_SUBSYSTEM have pattern generator (PG) and comparator (CMP) for BIST operation. The subsystem is being tested in a loopback fashion. The diagram shown below represents the loopback scheme implemented.

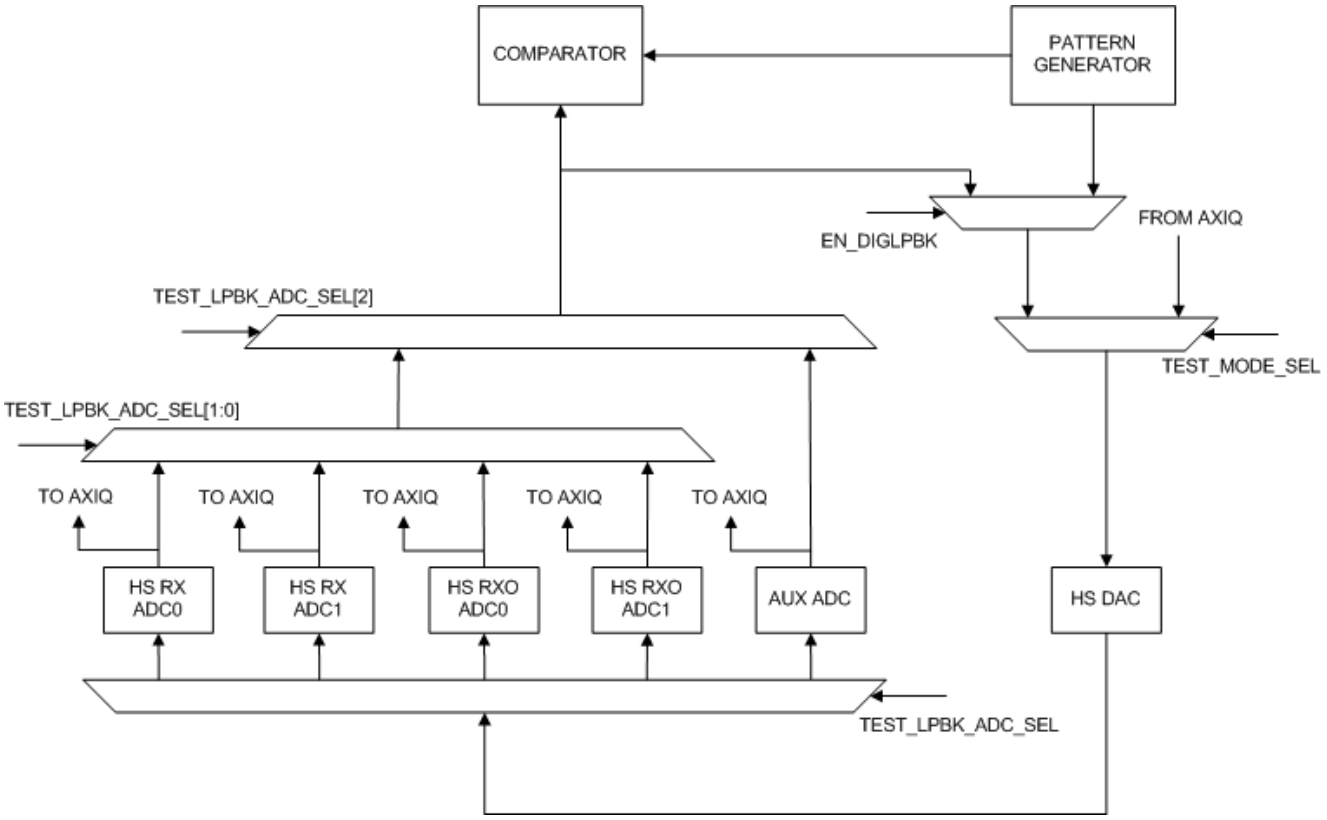


Figure 6. Analog and Digital Loopback Paths

The pattern generator drives a known pattern at DAC input. The same data is sent to the comparator which later compares this data with the loopback data coming out of the ADC being chosen for loopback. The DAC analog output must be looped back into the ADC externally on the board. The comparator compares the output of ADC and the expected data received from the pattern generator and updates the pass count in the status register. The pattern generator drives three types of pattern based on the configugation done in TEST_CFGCTL2[TEST_PATTRN_SEL]. For the first (00b) and second (01b) type PG drives data which is looped back and checked by the comparator. The third mode (10b) does not support the built in comparison of data being looped back. It is required to disable the comparator in third mode. The checking of data in third mode needs to be done by the software by examining ADC output data at the AXIQ interface. The following steps need to be performed to configure the test mode:

- The intended ADC/DAC must be in ready state.
- Program the TEST_LPBK_CFGCTL[TEST_MODE_SEL] to 1, to select DAC input from pattern generator.
- Program the TEST_MUXCTL[TEST_LPBK_ADC_SEL] to select which ADC to loopback.
- Program the comparator specific configurations in TEST_CFGCTLk registers. Configure TEST_MUXCTL[TEST_LPBK_ADC_SEL] to select the clock and data of desired ADC. Set comparator enable TEST_CFGCTL1[CMPR_EN_PAIR_0] to 1.
- When the comparator is disabled, there is a provision to observe the output of the selected ADC in the Test Status Register(TEST_STAT). In order to observe the output, the TEST_MUXCTL[TEST_LPBK_ADC_SEL] field to be set to select the desired ADC. The outputs can be observed in the PASS_CNT_I_PAIR_0[11:0] and PASS_CNT_Q_PAIR_0[27:16] register fields.
- In case the AUX ADC is being looped back with DAC, the TEST_OFFSET_CFG[CMPRTR_OFFSET_EN] field should be set to enable TEST_OFFSET_CFG[CMPRTR_OFFSET] during comparison.
- To synchronize the comparator specific fields in TEST_CFGCTLk registers, set the REG_SPACE_SYNC[TRGR_HSADC_SYNC] field and poll for its clearing.

- Program the pattern generator specific configurations in TEST_CFGCTLk registers. Set the pattern generator enable TEST_CFGCTL1[PATGEN_EN_PAIR_0] to 1.
- To synchronize the pattern generator specific fields in TEST_CFGCTLk registers, set the REG_SPACE_SYNC[TRGR_HSDAC_SYNC] field and poll for its clearing.
- Once all the patterns have been driven and compared, the pattern generator enable bit TEST_CFGCTL1[PATGEN_EN_PAIR_0] gets cleared.
- Read the TEST_STAT register for the pass count.

9.7 Application information

This section describes applications supported by the ADC-DAC module.

- To update High Speed ADC/DAC sampling frequency put the IP in power-down, change frequency to the desired value and put IP back to active mode.
- When the comparator is disabled, there is a provision to observe the output of the selected ADC in the Test Status Register(TEST_STAT). In order to observe the output, the TEST_MUXCTL[TEST_LPBK_ADC_SEL] field to be set to select the desired ADC. The outputs can be observed in the PASS_CNT_I_PAIR_0[11:0] and PASS_CNT_Q_PAIR_0[27:16] register fields.
- AUX ADC needs to support sampling rate of ~1 MSPS. Clock divider of 2-4096 (Even clock divider) is supported. To achieve this AUX_ADC_CLKCFG[AUX_ADC_CLK_DIV] is configured 12'h028 to get 16 MHz input clock frequency to AUX ADC, thus providing ~1 MSPS sampling rate.

Chapter 10
AXIQ

10.1 Introduction

The AXI I/Q interface block (AXIQ) provides a DMA-based, AXI-slave interface to auxiliary devices. The AXIQ can receive data (data transfer from auxiliary devices to VSPA DMEM) and transmit data (data transfer from VSPA DMEM to auxiliary devices). The AXIQ includes 6 recieve channels and 5 transmit channels which can operate simultaneously. Each channel has a built-in independent FIFO.

- The AXIQ provides FIFO buffering of the complex I/Q samples (or simple 8/16 bit sample in some cases as detailed below), with buffer accesses optimized for the AMBA AXI bus protocol and DMA side bus control and status signals.

AXIQ receives data samples from other modules, such as ADC and RSSI, stores the samples in an internal FIFO, and triggers the VSPA DMA to read the samples from the FIFO via the system AXI bus.

On the Transmit side, the VSPA DMA to writes the I/Q samples into the AXIQ FIFO via the system AXI bus. The AXIQ communicate the samples to other modules such as DAC. The AXIQ triggers the DMA to write the samples to the FIFO via the system AXI bus.

The AXIQ includes these key features:

- AXI bus interface with 128-bit Data bus
- Maximum 614.4 MHz AXI bus clock frequency
- Side bus signaling to DMA
  - Hardware DMA triggers are supported on each FIFO
  - DMA Reads (Rx Channels) and Writes (Tx Channels) the FIFO in Burst mode via the AXI bus
- AXIQ has both receive and transmit functionality:
  - In the receive channels, data is written into the AXIQ FIFO from other modules (such as ADC and RSSI), and then data is read from the FIFO via the AXI bus.
  - In the transmit channels, data is written into the AXIQ FIFO via the AXI bus, and then data is read from the FIFO by other modules (such as the DAC and proprietary IP).
- Support up to 6 Receive modules and 5 Transmit modules operating simultaneously
  - Maximum channel frequency of 307.2 MHz clock frequency, with clock edge synchronous to the AXI bus clock
- Seven FIFOs are implemented. Each FIFO is connected to an external module
  - Six receive channels are connected to 4 ADC modules, one "TX-Detect" module and one RSSI module.
  - One transmit channel is connected to 1 DAC module.
- Configurable subword swap of I and Q samples
  - Certain channels, can swap 16-bit I and Q

10.2 Interface Specification

The AXIQ module I/Os are shown in the following diagram.

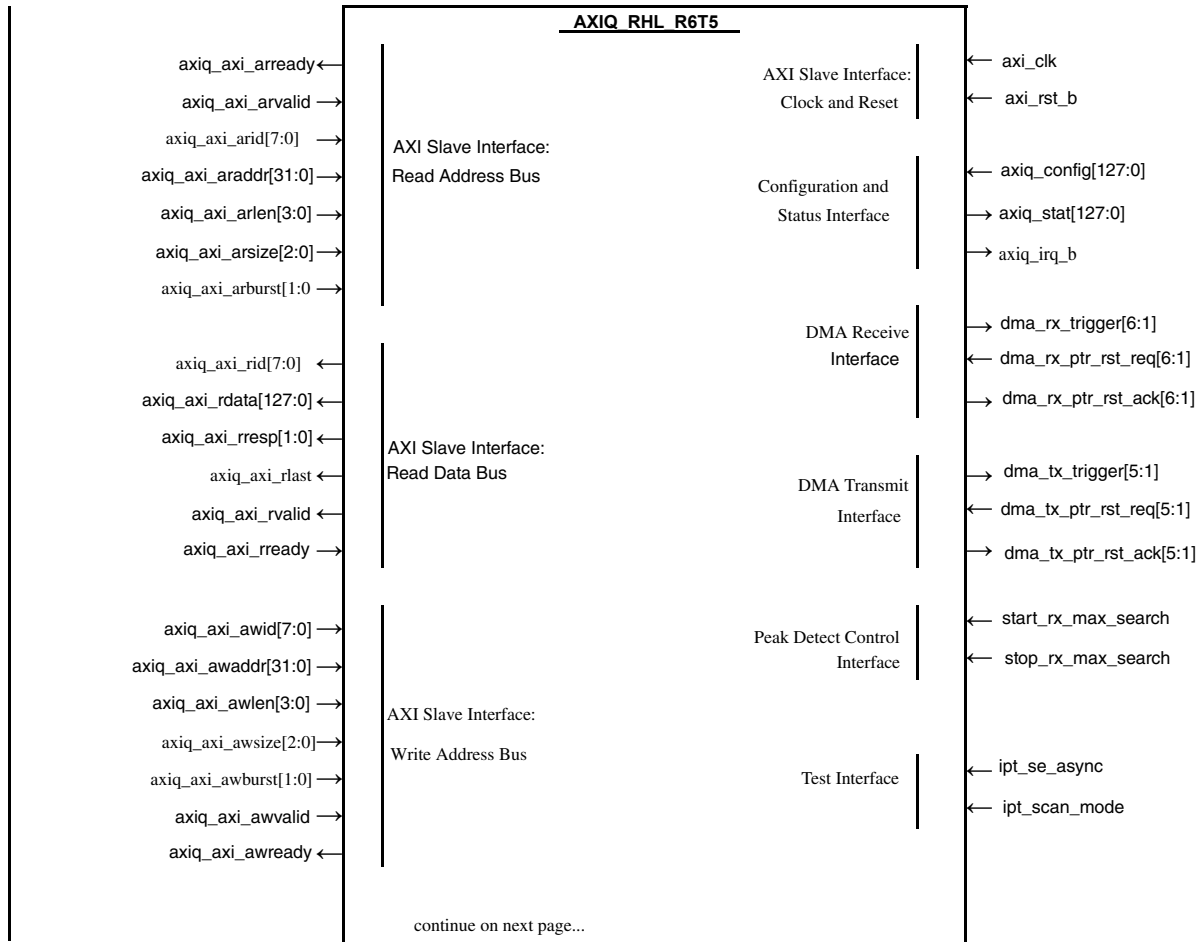


Figure 7. AXIQ Interface Signals (1/3)

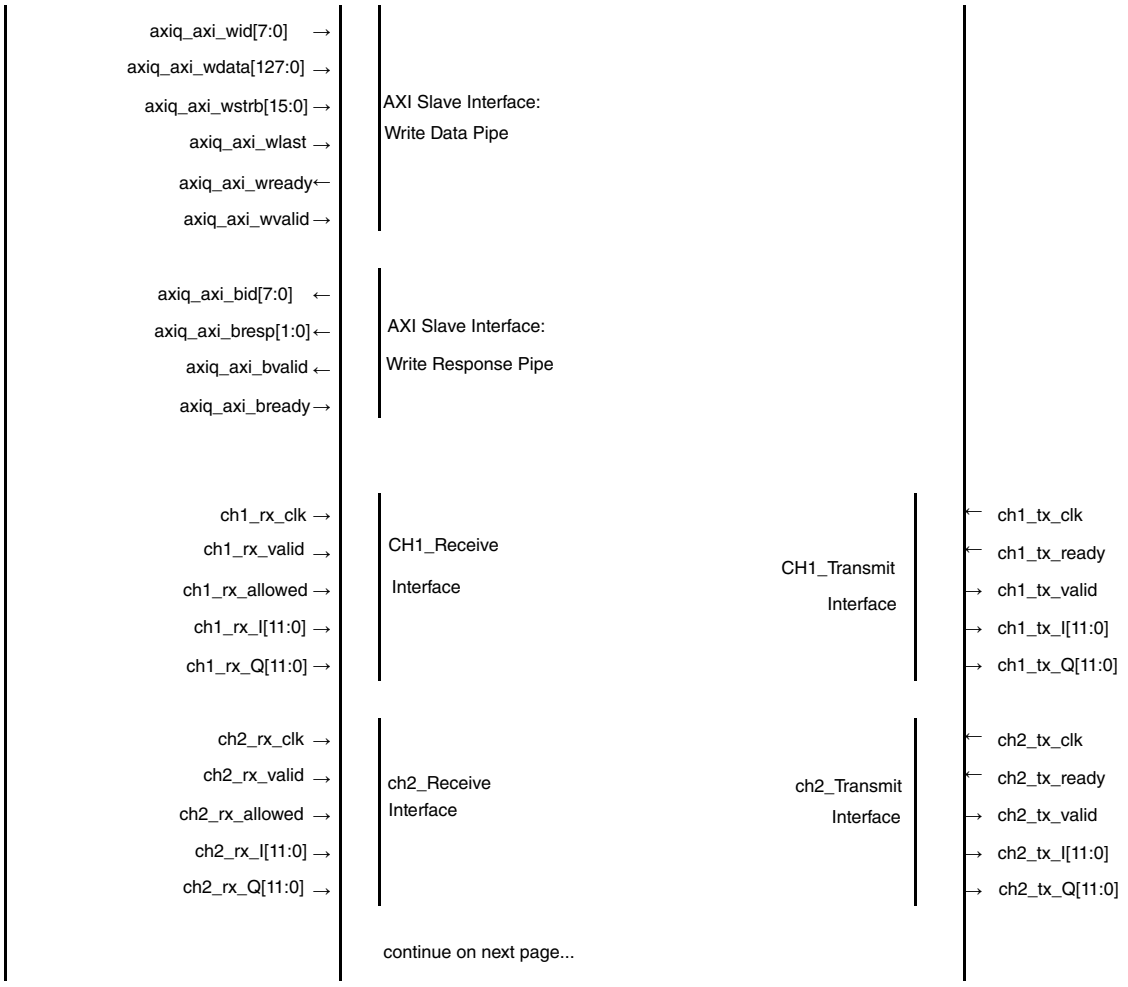


Figure 8. AXIQ Interface Signals (2/3)

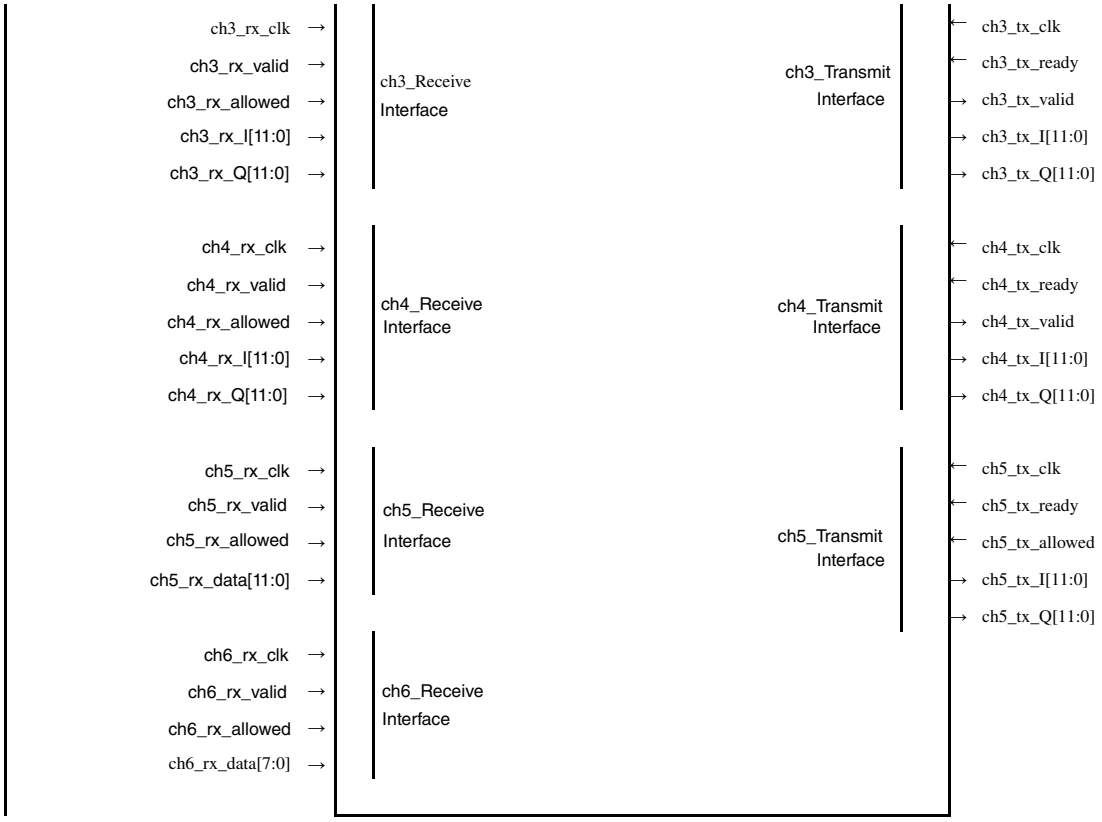


Figure 9. AXIQ Interface Signals (3/3)

* - K represents the value of AXI_ID_WIDTH. AXI_ID_WIDTH is a parameter whose value is set by the integrator.

10.2.1 Signals

The following diagram shows the port list for the AXIQ core.

Table 12. AXIQ Interface signals

Signal Name	I/O	Active State	Description	Reset State	Timing	
					Sync/ Async	Clock Sync
AXI Slave Interface: Clock and Reset						
axi_clk	I		AXI Clock		A	
axi_rst_b	I	Low	AXI asynchronous reset		S	axi_clk
AXI Slave Interface: Address Read Pipe						

Table continues on the next page...

Table 12. AXIQ Interface signals (continued)

Signal Name	I/O	Active State	Description	Reset State	Timing	
					Sync/ Async	Clock Sync
axiq_axi_arready	O	High	AR Pipe Ready	1	S	axi_clk
axiq_axi_arvalid	I	High	AR Pipe Valid		S	axi_clk
axiq_axi_arid[7:0]	I		Transaction ID		S	axi_clk
axiq_axi_araddr[31:0]	I		Read Address		S	axi_clk
axiq_axi_arlen[3:0]	I		Burst Length		S	axi_clk
axiq_axi_arsize[2:0]	I		Beat Size		S	axi_clk
axiq_axi_arburst[1:0]	I		Burst Type		S	axi_clk
AXI Slave Interface: Read Data Pipe						
axiq_axi_rid[7:0]	O		Transaction ID	0	S	axi_clk
axiq_axi_rdata[127:0]	O		Read Data	0	S	axi_clk
axiq_axi_rresp[1:0]	O		Read Response	0	S	axi_clk
axiq_axi_rlast	O	High	Read Last Sentinel	0	S	axi_clk
axiq_axi_rvalid	O	High	R Pipe Valid	0	S	axi_clk
axiq_axi_rready	I	High	R Pipe Ready		S	axi_clk
AXI Slave Interface: Address Write Pipe						
axiq_axi_awready	O	High	AW Pipe Ready	1	S	axi_clk
axiq_axi_awvalid	I	High	AW Pipe Valid		S	axi_clk
axiq_axi_awid[7:0]	I		Transaction ID		S	axi_clk
axiq_axi_awaddr[31:0]	I		Write Address		S	axi_clk
axiq_axi_awlen[3:0]	I		Burst Length		S	axi_clk
axiq_axi_awsize[2:0]	I		Beat Size		S	axi_clk
axiq_axi_awburst[1:0]	I		Burst Type		S	axi_clk
AXI Slave Interface: Write Data Pipe						

Table continues on the next page...

Table 12. AXIQ Interface signals (continued)

Signal Name	I/O	Active State	Description	Reset State	Timing	
					Sync/ Async	Clock Sync
axiq_axi_wready	O	High	W Pipe Ready	0	S	axi_clk
axiq_axi_wvalid	I	High	W Pipe Valid		S	axi_clk
axiq_axi_wid[7:0]	I		Transaction ID		S	axi_clk
axiq_axi_wdata[127:0]	I		Write Data		S	axi_clk
axiq_axi_wstrb[15:0]	I	High	Write Strobes		S	axi_clk
axiq_axi_wlast	I	High	Write Last Sentinel		S	axi_clk
AXI Slave Interface: Write Response Pipe						
axiq_axi_bid[7:0]	O		Transaction ID	0	S	axi_clk
axiq_axi_bresp[1:0]	O		Write Response	0	S	axi_clk
axiq_axi_bvalid	O	High	B Pipe Valid	0	S	axi_clk
axiq_axi_bready	I	High	B Pipe Ready		S	axi_clk
DMA Interface						
dma_rx_trigger[6:1]	O	High	DMA Receive Channel Trigger	0	S	axi_clk
dma_rx_ptr_rst_ack[6:1]	O	High	DMA Receive Channel Pointer Reset Acknowledgment	0	S	axi_clk
dma_rx_ptr_rst_req[6:1]	I	High	DMA Receive Channel Pointer Reset Request		A	axi_clk
dma_tx_trigger[5:5]	O	High	DMA Transmit Channel Trigger	0	S	axi_clk
dma_tx_ptr_rst_ack[5:5]	O	High	DMA Transmit Channel Pointer Reset Acknowledgment	0	S	axi_clk
dma_tx_ptr_rst_req[5:5]	I	High	DMA Transmit Channel Pointer Reset Request		A	axi_clk
AXIQ Configuration						
axiq_config[127:0]	I	High	AXIQ Configuration bus Please refer to AXIQ Configuration Bus for more details.	0	A	axi_clk

Table continues on the next page...

Table 12. AXIQ Interface signals (continued)

Signal Name	I/O	Active State	Description	Reset State	Timing	
					Sync/ Async	Clock Sync
AXIQ Status						
axiq_status[127:0]	O	High	AXIQ Status bus Please refer to AXIQ Status Bus for more details.	0	S	axi_clk
Interrupt Request						
axiq_irq_b	O	Low	AXIQ Interrupt Request	0	A	axi_clk
Channel 1 Receive Interface						
ch1_rx_clk	I		Channel 1 Receive Clock		S	axi_clk
ch1_rx_valid	I	High	Channel 1 Receive Valid Signal		S	ch1_rx_clk
ch1_rx_allowed	I	High	Channel 1 Receive Allowed Signal		S	ch1_rx_clk
ch1_rx_I[11:0]	I		Channel 1 Receive I Data		S	ch1_rx_clk
ch1_rx_Q[11:0]	I		Channel 1 Receive Q Data		S	ch1_rx_clk
Channel 2 Receive Interface						
ch2_rx_clk	I		Channel 2 Receive Clock		S	axi_clk
ch2_rx_valid	I	High	Channel 2 Receive Valid Signal		S	ch2_rx_clk
ch2_rx_allowed	I	High	Channel 2 Receive Allowed Signal		S	ch2_rx_clk
ch2_rx_I[11:0]	I		Channel 2 Receive I Data		S	ch2_rx_clk
ch2_rx_Q[11:0]	I		Channel 2 Receive Q Data		S	ch2_rx_clk
Channel 3 Receive Interface						
ch3_rx_clk	I		Channel 3 Receive Clock		S	axi_clk
ch3_rx_valid	I	High	Channel 3 Receive Valid Signal		S	ch3_rx_clk
ch3_rx_allowed	I	High	Channel 3 Receive Allowed Signal		S	ch3_rx_clk
ch3_rx_I[11:0]	I		Channel 3 Receive I Data		S	ch3_rx_clk
ch3_rx_Q[11:0]	I		Channel 3 Receive Q Data		S	ch3_rx_clk

Table continues on the next page...

Table 12. AXIQ Interface signals (continued)

Signal Name	I/O	Active State	Description	Reset State	Timing	
					Sync/ Async	Clock Sync
Channel 4 Receive Interface						
ch4_rx_clk	I		Channel 4 Receive Clock		S	axi_clk
ch4_rx_valid	I	High	Channel 4 Receive Valid Signal		S	ch4_rx_clk
ch4_rx_allowed	I	High	Channel 4 Receive Allowed Signal		S	ch4_rx_clk
ch4_rx_I[11:0]	I		Channel 4 Receive I Data		S	ch4_rx_clk
ch4_rx_Q[11:0]	I		Channel 4 Receive Q Data		S	ch4_rx_clk
Channel 5 Receive Interface						
ch5_rx_clk	I		Channel 5 Receive Clock		S	axi_clk
ch5_rx_valid	I	High	Channel 5 Receive Valid Signal		S	ch5_rx_clk
ch5_rx_allowed	I	High	Channel 5 Receive Allowed Signal		S	ch5_rx_clk
ch5_rx_data[11:0]	I		Channel 5 Receive Data		S	ch5_rx_clk
Channel 6 Receive Interface						
ch6_rx_clk	I		Channel 6 Receive Clock		S	axi_clk
ch6_rx_valid	I	High	Channel 6 Receive Valid Signal		S	ch6_rx_clk
ch6_rx_allowed	I	High	Channel 6 Receive Allowed Signal		S	ch6_rx_clk
ch6_rx_data[7:0]	I		Channel 6 Receive Data		S	ch6_rx_clk
Channel 5 Transmit Interface						
ch5_tx_clk	I		Channel 5 Transmit Clock		S	axi_clk
ch5_tx_allowed	O	High	Channel 5 Transmit Allowed Signal		S	ch5_tx_allowed
ch5_tx_ready	I	High	Channel 5 Transmit Ready Signal		S	ch5_tx_clk
ch5_tx_I[11:0]	O		Channel 5 Transmit I Data		S	ch5_tx_clk
ch5_tx_Q[11:0]	O		Channel 5 Transmit Q Data		S	ch5_tx_clk

Table continues on the next page...

Table 12. AXIQ Interface signals (continued)

Signal Name	I/O	Active State	Description	Reset State	Timing	
					Sync/ Async	Clock Sync
Peak Detect Interface						
start_rx_max_search	I	High	Start Receive Peak (Max) search See Receive Peak (Max Value) Detect detailed description		S	all RX clocks
stop_rx_max_search	I	High	Stop Receive Peak (Max) search For additional details See Receive Peak (Max Value) Detect detailed description		S	all RX clocks
Test Interface						
ipt_se_async	I	High	Scan Enable: reset sync output async reset		A	
ipt_scan_mode	I	High	Scan Mode Enable: all clock gates are opened		A	

10.2.1.1 AXIQ Configuration Bus

The AXIQ Configuration bus is used to control the AXIQ module. Each channel is individually configured and controlled. The use of a bus is merely to simplify the connectivity.

Note: Unused axiq_config bits and the corresponding GPOUT bits that are driving them should be always set to zero.

It is assumed the source of the axiq_config bus is the VSPA GPOUT registers. In the following table, the source column indicates the source specific register and bit/s. It is assumed the axiq_config bus is composed of the VSPA GPOUT registers as follows: axiq_config[127:0] = {GPOUT_7[31:0], GPOUT_6[31:0], GPOUT_5[31:0], GPOUT_4[31:0]}.

Either VSPA or external host (Arm core) can set the AXIQ Configuration bus via the IPS bus interface.

Table 13. AXIQ_Configuration Bus signals

					Source	
Signal Name	Active State	Description	Reset State	axiq_config bits	Register	Bit/s
Channel 1 Receive Control						
ch1_rx_en	High	Channel 1 Enable	0	axiq_config[0]	GPOUT_4	[0]
static_ch1_rx_fifo_thresh[1:0]	High	Channel 1 Rx FIFO Threshold	00	axiq_config[2:1]	GPOUT_4	[2:1]

Table continues on the next page...

Table 13. AXIQ_Configuration Bus signals (continued)

					Source	
Signal Name	Active State	Description	Reset State	axiq_config bits	Register	Bit/s
static_ch1_rx_swap_IQ	High	Channel 1 Rx Swap I and Q	0	axiq_config[3]	GPOUT_4	[3]
ch1_rx_clear_error	High	Channel 1 Rx Clear Overflow and Underflow Errors	0	axiq_config[4]	GPOUT_4	[4]
Channel 2 Receive Control						
ch2_rx_en	High	Channel 2 Enable	0	axiq_config[8]	GPOUT_4	[8]
static_ch2_rx_fifo_thresh[1:0]	High	Channel 2 Rx FIFO Threshold	00	axiq_config[10:9]	GPOUT_4	[10:9]
static_ch2_rx_swap_IQ	High	Channel 2 Rx Swap I and Q	0	axiq_config[11]	GPOUT_4	[11]
ch2_rx_clear_error	High	Channel 2 Rx Clear Overflow and Underflow Errors	0	axiq_config[12]	GPOUT_4	[12]
Channel 3 Receive Control						
ch3_rx_en	High	Channel 3 Enable	0	axiq_config[16]	GPOUT_4	[16]
static_ch3_rx_fifo_thresh[1:0]	High	Channel 3 Rx FIFO Threshold	00	axiq_config[18:17]	GPOUT_4	[18:17]
static_ch3_rx_swap_IQ	High	Channel 3 Rx Swap I and Q	0	axiq_config[19]	GPOUT_4	[19]
ch3_rx_clear_error	High	Channel 3 Rx Clear Overflow and Underflow Errors	0	axiq_config[20]	GPOUT_4	[20]
Channel 4 Receive Control						
ch4_rx_en	High	Channel 4 Enable	0	axiq_config[24]	GPOUT_4	[24]
static_ch4_rx_fifo_thresh[1:0]	High	Channel 4 Rx FIFO Threshold	00	axiq_config[26:25]	GPOUT_4	[26:25]
static_ch4_rx_swap_IQ	High	Channel 4 Rx Swap I and Q	0	axiq_config[27]	GPOUT_4	[27]

Table continues on the next page...

Table 13. AXIQ_Configuration Bus signals (continued)

					Source	
Signal Name	Active State	Description	Reset State	axiq_config bits	Register	Bit/s
ch4_rx_clear_error	High	Channel 4 Rx Clear Overflow and Underflow Errors	0	axiq_config[28	GPOUT_4	[28]
Channel 5 Receive Control						
ch5_rx_en	High	Channel 5 Enable	0	axiq_config[32]	GPOUT_5	[0]
static_ch5_rx_fifo_thresh[1:0]	High	Channel 5 Rx FIFO Threshold	00	axiq_config[34:33]	GPOUT_5	[2:1]
Reserved			0	axiq_config[35]	GPOUT_5	[3]
ch5_rx_clear_error	High	Channel 5 Rx Clear Overflow and Underflow Errors	0	axiq_config[36]	GPOUT_5	[4]
Channel 6 Receive Control						
ch6_rx_en	High	Channel 6 Enable	0	axiq_config[40]	GPOUT_5	[8]
static_ch6_rx_fifo_thresh[1:0]	High	Channel 6 Rx FIFO Threshold	00	axiq_config[42:41]	GPOUT_5	[10:9]
Reserved			0	axiq_config[43]	GPOUT_5	[11]
ch6_rx_clear_error	High	Channel 6 Rx Clear Overflow and Underflow Errors	0	axiq_config[44]	GPOUT_5	[12]
ch6_rx_wcnt[3:0]		Channel 6 Rx Half-Word (2-bytes) count		axiq_config[59:56]	GPOUT_5	[27:24]
Channel 5 Transmit Control						
ch5_tx_en	High	Channel 5 Enable	0	axiq_config[96]	GPOUT_7	[0]
static_ch5_tx_fifo_thresh[1:0]	High	Channel 5 Tx FIFO Threshold	00	axiq_config[98:97]	GPOUT_7	[2:1]
static_ch5_tx_swap_IQ	High	Channel 5 Tx Swap I and Q	0	axiq_config[99]	GPOUT_7	[3]

Table continues on the next page...

Table 13. AXIQ_Configuration Bus signals (continued)

					Source	
Signal Name	Active State	Description	Reset State	axiq_config bits	Register	Bit/s
ch5_tx_clear_error	High	Channel 5 Tx Clear Overflow and Underflow Errors	0	axiq_config[100]	GPOUT_7	[4]
Peak (Max) Search Control						
vspa_restart_rx_max_bit	High	VSPA restart max (peak) search For additional details See Receive Peak (Max Value) Detect detailed description	0	axiq_config[127]	GPOUT_7	[31]

[**] (GPOUT_Register * 32) + Bit/s

10.2.1.2 AXIQ Status Bus

The AXIQ Status bus is used to indicate the AXIQ module status. The status of each channel is individually signaled. The use of a bus is merely to simplify the connectivity.

Note: The state of the unused axiq_stat bits and the corresponding GPIN bits is not defined, and reading these bits may result in any value. However, typically they are read as zero.

It is assumed the destination of the axiq_stat bus is the VSPA GPIN registers. In the table below, the source column indicates the destination specific register and bit/s. It is assumed the VSPA GPIN registers are driven by the axiq_stat bus as follows:

{GPIN_3[31:0], GPIN_2[31:0], GPIN_1[31:0], GPIN_0[31:0]} = axiq_stat[127:0].

Table 14. AXIQ_Status Bus signals

					Source	
Signal Name	Active State	Description	Reset State	axiq_config bits	Register	Bit/s
Channel 1 Rx Status						
ch1_rx_enabled	High	Channel 1 Rx Enabled	0	axiq_stat[0]	GPIN_0	0
ch1_rx_fifo_not_empty	High	Channel 1 Rx Fifo is NOT empty	0	axiq_stat[1]	GPIN_0	1
ch1_rx_underflow_error	High	Channel 1 Rx Underflow Error Flag	0	axiq_stat[2]	GPIN_0	2
ch1_rx_overflow_error	High	Channel 1 Rx Overflow Error Flag	0	axiq_stat[3]	GPIN_0	3
Channel 2 Rx Status						
ch2_rx_enabled	High	Channel 2 Rx Enabled	0	axiq_stat[4]	GPIN_0	4

Table continues on the next page...

Table 14. AXIQ_Status Bus signals (continued)

ch2_rx_fifo_not_empty	High	Channel 2 Rx Fifo is NOT empty	0	axiq_stat[5]	GPIN_0	5
ch2_rx_underflow_error	High	Channel 2 Rx Underflow Error Flag	0	axiq_stat[6]	GPIN_0	6
ch2_rx_overflow_error	High	Channel 2 Rx Overflow Error Flag	0	axiq_stat[7]	GPIN_0	7
Channel 3 Rx Status						
ch3_rx_enabled	High	Channel 3 Rx Enabled	0	axiq_stat[8]	GPIN_0	8
ch3_rx_fifo_not_empty	High	Channel 3 Rx Fifo is NOT empty	0	axiq_stat[9]	GPIN_0	9
ch3_rx_underflow_error	High	Channel 3 Rx Underflow Error Flag	0	axiq_stat[10]	GPIN_0	10
ch3_rx_overflow_error	High	Channel 3 Rx Overflow Error Flag	0	axiq_stat[11]	GPIN_0	11
Channel 4 Rx Status						
ch4_rx_enabled	High	Channel 4 Rx Enabled	0	axiq_stat[12]	GPIN_0	12
ch4_rx_fifo_not_empty	High	Channel 4 Rx Fifo is NOT empty	0	axiq_stat[13]	GPIN_0	13
ch4_rx_underflow_error	High	Channel 4 Rx Underflow Error Flag	0	axiq_stat[14]	GPIN_0	14
ch4_rx_overflow_error	High	Channel 4 Rx Overflow Error Flag	0	axiq_stat[15]	GPIN_0	15
Channel 5 Rx Status						
ch5_rx_enabled	High	Channel 5 Rx Enabled	0	axiq_stat[16]	GPIN_0	16
ch5_rx_fifo_not_empty	High	Channel 5 Rx Fifo is NOT empty	0	axiq_stat[17]	GPIN_0	17
ch5_rx_underflow_error	High	Channel 5 Rx Underflow Error Flag	0	axiq_stat[18]	GPIN_0	18
ch5_rx_overflow_error	High	Channel 5 Rx Overflow Error Flag	0	axiq_stat[19]	GPIN_0	19
Channel 6 Rx Status						
ch6_rx_enabled	High	Channel 6 Rx Enabled	0	axiq_stat[20]	GPIN_0	20
ch6_rx_fifo_not_empty	High	Channel 6 Rx Fifo is NOT empty	0	axiq_stat[21]	GPIN_0	21
ch6_rx_underflow_error	High	Channel 6 Rx Underflow Error Flag	0	axiq_stat[22]	GPIN_0	22
ch6_rx_overflow_error	High	Channel 6 Rx Overflow Error Flag	0	axiq_stat[23]	GPIN_0	23
Channel 5 Tx Status						
ch5_tx_enabled	High	Channel 5 Tx Enabled	0	axiq_stat[48]	GPIN_1	16
ch5_tx_fifo_not_full	High	Channel 5 Tx Fifo is NOT full	0	axiq_stat[49]	GPIN_1	17

Table continues on the next page...

Table 14. AXIQ_Status Bus signals (continued)

ch5_tx_underflow_error	High	Channel 5 Tx Underflow Error Flag	0	axiq_stat[50]	GPIN_1	18
ch5_tx_overflow_error	High	Channel 5 Tx Overflow Error Flag	0	axiq_stat[51]	GPIN_1	19
Peak (Max) Search Status						
ch1_max_rx_data[15:0]		Channel 1 Peak (Max) data	0	axiq_stat[79:64]	GPIN_2	15:0
ch2_max_rx_data[15:0]		Channel 2 Peak (Max) data	0	axiq_stat[95:80]	GPIN_2	31:16
ch3_max_rx_data[15:0]		Channel 3 Peak (Max) data	0	axiq_stat[111:96]	GPIN_3	15:0
ch4_max_rx_data[15:0]		Channel 4 Peak (Max) data	0	axiq_stat[127:112]	GPIN_3	31:16

10.2.2 ADC/DAC subsystem AXIQ Interfaces

AXIQ interface for ADC-

- RX_CLK (OUTPUT CLOCK)
- RX_VALID (OUTPUT DATA VALID)
- RX_I_DATA [11:0] (OUTPUT I DATA)
- RX_Q_DATA [11:0] (OUTPUT Q DATA)

AXIQ interface for DAC-

- TX_CLK (OUTPUT CLOCK)
- TX_READY (INPUT DATA VALID)
- TX_I_DATA[11:0] (INPUT I DATA)
- TX_Q_DATA[11:0] (INPUT Q DATA)

SCAN interface for ADC and DAC (50 MHz) -

- SCAN_CLK (INPUT SCAN CLOCK)
- SCAN_IN (INPUT SERIAL DATA)
- SCAN_MODE (INPUT MODE)
- SCAN_EN (INPUT SCAN ENABLE)
- SCAN_RESETZ (INPUT SCAN RESET)
- SCAN_OUT (OUTPUT SERIAL DATA)

10.2.3 AXIQ Signal description

10.2.3.1 DMA Interface

In the table below, x and y indicate AXIQ channel number: (x = 1-6, and y = 5).

Table 15. DMA Interface

Name	Description
dma_rx_trigger[x]	DMA Receive Trigger Signal

Table continues on the next page...

Table 15. DMA Interface (continued)

Name	Description
	The Receive DMA trigger signal tells the external DMA to perform a read access to the corresponding AXIQ Receive channel. When not in FLUSH mode, dma_rx_trigger is 1 when the receive FIFO level is greater-than-or-equal-to the receive threshold specified by static_rx_fifo_thresh[1:0] and the AXI read interface is not in a read burst, 0 otherwise. When the channel is in FLUSH mode, then dma_trigger[n] is set to 1 (except during in an AXI read transaction).
dma_rx_ptr_rst_req[x]	DMA Receive Pointer Reset Request dma_rx_ptr_rst_req is set to 1 by the DMA when it has finished its read transfers from the corresponding receive channel. Once asserted, the DMA must hold dma_rx_ptr_rst_req set until the AXIQ acknowledges the request by setting the corresponding dma_ptr_rst_ack to 1. The dma_rx_ptr_rst_req==1 is also used by the channel to exit FLUSH mode.
dma_rx_ptr_rst_ack[x]	DMA Receive Pointer Reset Acknowledgement The value of dma_rx_ptr_rst_ack will follow the value dma_rx_ptr_rst_req after the Pointer Reset (FLUSH mode exit) has occurred.
dma_tx_trigger[y]	DMA Transmit Trigger Signal The Transmit DMA trigger signal tells the external DMA to perform a write access to the AXIQ channel. When not in FLUSH mode, dma_tx_trigger is 1 when the number of empty entries in the transmit FIFO is greater-than-or-equal-to the transmit threshold specified by static_tx_fifo_thresh[1:0] and the AXI write interface is not in a write burst, 0 otherwise. When the channel is in FLUSH mode, then dma_trigger[n] is set to 1 (except during in an AXI write transaction).
dma_tx_ptr_rst_req[y]	DMA Transmit Pointer Reset Request dma_tx_ptr_rst_req is set to 1 by the DMA when it has finished its write transfers to the corresponding transmit channel. Once asserted, the DMA must hold dma_tx_ptr_rst_req set until the AXIQ acknowledges the request by setting the corresponding dma_ptr_rst_ack to 1. The dma_tx_ptr_rst_req==1 is also used by the channel to exit FLUSH mode.
dma_tx_ptr_rst_ack[y]	DMA Transmit Pointer Reset Acknowledgement The value of dma_tx_ptr_rst_ack will follow the value dma_tx_ptr_rst_req after the Pointer Reset (FLUSH mode exit) has occurred.

10.2.3.2 Configuration Interface signal description

The AXIQ Configuration bus is used to control the AXIQ module. Each channel is individually configured and controlled. The use of a bus is merely to simplify the connectivity.

Note: Unused axiq_config bits and the corresponding GPOUT bits that are driving them should be always set to zero.

It is assumed the source of the axiq_config bus is the VSPA GPOUT registers. In AXIQ Configuration Bus, the source column indicates the source specific register and bit/s. It is assumed the axiq_config bus is composed of the VSPA GPOUT registers as follows: axiq_config[127:0] = {GPOUT_7[31:0], GPOUT_6[31:0], GPOUT_5[31:0], GPOUT_4[31:0]}.

Either VSPA or external host (Arm core) can set the AXIQ Configuration bus via the IPS bus interface.

Configuration signals may be prefixed by the static_ prefix. The static_ prefix means that the signal is expected to be static when the receive channel is active.

A Receive channel is active when either the corresponding chX_rx_en configuration signal or the corresponding ChX_rx_enabled status signal is set to one. Signals with the static_ prefix can be changed ONLY when the corresponding chX_rx_en and chX_rx_enabled signals are both set to 0 (X = 1 - 6).

A Transmit channel is active when either the corresponding ch5_tx_en configuration signal or the corresponding Ch5_tx_enabled status signal is set to one. Signals with the static_ prefix can be changed ONLY when the corresponding ch5_tx_en and ch5_tx_enabled signals are both set to 0 (X = 5).

In the following table X indicates the channel number (X = 1-6).

Table 16. Rx Configuration Signals

Name	Description
Channel Control (All Channels)	
chX_rx_en	Channel Enable This signal is used to control if a channel is enabled or disabled. 0: Channel is Disabled 1: Channel is Enabled A transition of chX_rx_en from one to zero has special meaning: it puts the receive channel in FLUSH mode. In FLUSH mode, the signal dma_rx_trigger is always high (except during an AXI read transaction), allowing any outstanding DMA transfers to complete (although, remaining receive data will be garbage data).
static_chX_rx_fifo_thresh[1:0]	Receive FIFO Threshold This threshold defines how many entries are in the receive FIFO before the receive DMA interface triggers the external DMA to retrieve data. 0: 16 entries or more 1: 8 entries or more 2: 4 entries or more 3: 2 entries or more **NOTE** The corresponding VSPA DMA channel must be programmed properly for FIFO threshold to work properly in 8-entries, 4-entries and 2-entries modes. Refer to the VSPA DMA programming guide for additional information.
static_chX_rx_swap_IQ X=1-4	Receive Swap I and Q This bit is used to swap how I and Q samples are packed into the FIFO words. See Rx I/Q Packer and Tx I/Q Unpacker for more details. 0: Do not swap 1: Swap I and Q
chX_rx_clear_error	Receive Channel Clear Error Signal

Table continues on the next page...

Table 16. Rx Configuration Signals (continued)

Name	Description
	When set, it clears the chX_rx_overflow_error and the chX_rx_underflow_error status signals. When cleared, the chX_rx_clear_error has no affect on these status bits. **NOTE** This signal must remain 1 for more than 8 axi_clk periods to be guarantee detection.
vspa_restart_rx_max_bit	VSPA Restart Peak (Max) Search Signal When rising edge is detected (changes from zero to one), Receive Peak (Max Value) Detect starts or restarts the peak (max) value search. When cleared, the vspa_restart_rx_max_bit has no affect on Receive Peak (Max Value) Detect circuitry. **NOTE** This signal must remain 1 for more than 8 axi_clk periods to guarantee detection. Please refer to Receive Peak (Max Value) Detect detailed description for additional details.
ch6_rx_wcnt[3:0]	Channel 6 Receive Half-Word (two-bytes) Count Channel 6 Receive half-Word (two-byte) Count is used to control when channel 6 asserts the DMA external trigger, which in turn causes the DMA to start transferring data from the Channel 6 FIFO into DMEM. When not zero, the value in ch6_rx_wcnt[3:0] overrides the value set in ch6_rx_fifo_thresh[1:0]. For detailed description refer to section Channel 6 Half-Word Count detailed description. 0: ch6_rx_fifo_thresh[1:0] is used as described above 1: 2 Bytes 2: 4 Bytes 3: 6 Bytes 4: 8 Bytes 5: 10 Bytes 6: 12 Bytes 7: 14 Bytes 8: 16 Bytes 9: 18 Bytes 10: 20 Bytes 11: 22 Bytes 12: 24 Bytes 13: 26 Bytes

Table continues on the next page...

Table 16. Rx Configuration Signals (continued)

Name	Description
	14: 28 Bytes 15: 30 Bytes

In the following table X indicates the channel number (X = 5).

Table 17. Tx Configuration Signals

Name	Description
Channel Control (All Channels)	
chX_tx_en	Channel Enable This signal is used to control if a channel is enabled or disabled. 0: Channel is Disabled 1: Channel is Enabled A transition of chX_tx_en from one to zero has special meaning: it puts the transmit channel in FLUSH mode. In FLUSH mode, the signal dma_tx_trigger is always high (except during an AXI read transaction), allowing any outstanding DMA transfers to complete (although remaining transmit data will be garbage data).
static_chX_tx_fifo_thresh[1:0]	Transmit FIFO Threshold This threshold defines how many available (empty) entries will be in the transmit FIFO before the transmit DMA interface triggers the external DMA to retrieve data. 0: 16 entries or more 1: 8 entries or more 2: 4 entries or more 3: 2 entries or more **NOTE** The corresponding VSPA DMA channel must be programmed properly for FIFO threshold to work properly in 8-entries, 4-entries and 2-entries modes. Refer to the VSPA DMA programming guide for additional information.
static_chX_tx_swap_IQ	Transmit Swap I and Q This bit is used to swap how I and Q samples are packed into FIFO entries. See Rx I/Q Packer and Tx I/Q Unpacker for more details. 0: Do not swap 1: Swap I and Q
chX_tx_clear_error	Transmit Channel Reset Error Signal When set, it clears the chX_tx_overflow_error and the chX_tx_underflow_error status signals. When cleared, the chX_tx_clear_error has no affect on these status bits.

Table continues on the next page...

Table 17. Tx Configuration Signals (continued)

Name	Description
	NOTE This signal must remain 1 for more than 4 axi_clk periods to be guarantee detection.

10.2.3.3 Status and Interrupt Interface

The AXIQ Status bus is used to indicate the AXIQ module status. The status of each channel is individually signaled. The use of a bus is merely to simplify the connectivity.

NOTE

Note: The state of the unused axiq_stat bits and the corresponding GPIN bits is not defined, and reading these bits may result in any value. However, typically they are read as zero.

It is assumed the destination of the axiq_stat bus is the VSPA GPIN registers. In Table 14, the source column indicates the destination specific register and bit/s. It is assumed the VSPA GPIN registers are driven by the axiq_stat bus as follows:

{GPIN_3[31:0], GPIN_2[31:0], GPIN_1[31:0], GPIN_0[31:0]} = axiq_stat[127:0].

NOTE

The state of the unused axiq_stat bits and the corresponding GPIN bits they are driving is not guaranteed, and they may read any value at any time. However, typically they are set to zero.

In the following table X indicate the channel number (X = 1- 6).

Table 18. Rx Channels Status Signals

Name	Description
chX_rx_enabled	"Channel X Receive Enabled" Flag 0: The channel is not enabled. 1: The channel is enabled.
chX_rx_fifo_not_empty	"Channel X Receive FIFO is NOT empty" Flag 0: The FIFO is empty. 1: The FIFO is not empty. When the Receive FIFO is not empty, at least one additional FIFO entry can be read from the FIFO without causing FIFO underflow error condition. This indication can be used, during debug, to read the Rx FIFO manually, entry by entry, from AXI bus masters other than the VSPA DMA.
chX_rx_underflow_error	Receive Channel Underflow Error Flag chX_rx_underflow_error is set to 1 when AXI read from an empty receive FIFO. The bit is set to 0 when chX_rx_clear_error is 1.
chX_rx_overflow_error	Receive Channel Overflow Error Flag chX_rx_overflow_error is set to 1 when the receive interface writes to a full receive FIFO. The bit is set to 0 when chX_rx_clear_error is 1.

Table continues on the next page...

Table 18. Rx Channels Status Signals (continued)

Name	Description
In the following entries Y indicate the channel number (Y = 1 - 4).	
chY_max_rx_data[15:0]	Receive Channel Max (Peak) saved value. Please refer to Receive Peak (Max Value) Detect detailed description for more details.

In the following table X indicate the channel number (X = 5).

Table 19. Tx Channels Status Signals

Name	Description
chX_tx_enabled	"Channel X Transmit Enabled" Flag 0: The channel is not enabled. 1: The channel is enabled.
chX_tx_fifo_not_full	"Channel X Transmit FIFO is NOT full" Flag 0: The FIFO is full. 1: The FIFO is not full. When the Transmit FIFO is not full, at least one additional FIFO entry can be written to the FIFO without causing FIFO overflow error condition. This indication can be used, during debug, to fill the Tx FIFO manually, entry by entry, from AXI bus masters other than the VSPA DMA.
chX_tx_underflow_error	Transmit Channel Underflow Error Flag chX_tx_underflow_error is set to 1 when AXI read from an empty transmit FIFO. The bit is set to 0 when chX_tx_clear_error is 1. **NOTE** Only valid condition is X=5 , underflow can happen on only TX 5 Cannot generate underflow error because of valid-ready protocol on TX channels 1-4
chX_tx_overflow_error	Transmit Channel Overflow Error Flag chX_tx_overflow_error is set to 1 when the transmit interface writes to a full transmit FIFO. The bit is set to 0 when chX_tx_clear_error is 1.

10.2.3.3.1 Interrupt Interface

The AXIQ Interrupt signal is used to indicate to the Host (Arm or VSPA) that an error bit is set and requires host attention.

The AXIQ interrupt request is an indication that at least one of the channels has the *overflow_error* or the *underflow_error* set to one. Any Receive channel or Transmit channel can cause the interrupt assertion.

The interrupt request is clear when all the corresponding error signals are cleared.

When the host receive the interrupt request, it should read the Status Register to identify the detailed cause. All the error flags must be cleared to negate the interrupt request.

Table 20. Status Signals

Name	Description
axiq_irq_b	Receive Channel Interrupt request, **active low.** 0: interrupt request. At least one of the channels has one (or both) the signals chX_rx_overflow_error, the chX_rx_underflow_error, chX_tx_overflow_error, or the chX_tx_underflow_error set to one. 1: No interrupt request

10.2.3.4 Receive Interface

X indicates channel number: (X = 1 - 6).

Table 21. Receive Interface

Name	Description
chX_rx_clk	Receive clock (X = 1 - 6)
chX_rx_valid	Receive data is valid (X = 1 - 6)
chX_rx_allowed	Channel X is allowed to receive data (X = 1 - 6) The chX_rx_allowed signal acts as external on/off switch for the channel. 0: Receiver is off. I/Q Data and chX_rx_valid are ignored. 1: Receiver is on.
chX_rx_I[11:0]	Receive I data Channel X (12-bit data) (X = 1 - 4) Data is captured by the AXIQ on rising edge of chX_rx_clk when chX_rx_valid and chX_rx_allowed are both asserted, and the channel is enabled (chX_rx_enabled==1)
chX_rx_Q[11:0]	Receive Q data Channel X (12-bit data) (X = 1 - 4) Data is captured by the AXIQ on rising edge of chX_rx_clk when chX_rx_valid and chX_rx_allowed are both asserted, and the channel is enabled (chX_rx_enabled==1)
ch5_rx_data[11:0]	Receive data Channel 5 (12-bit data) Data is captured by the AXIQ on rising edge of ch5_rx_clk when ch5_rx_valid and ch5_rx_allowed are both asserted, and the channel is enabled (ch5_rx_enabled==1)
ch6_rx_data[7:0]	Receive data Channel 6 (8-bit data) Data is captured by the AXIQ on rising edge of ch6_rx_clk when ch6_rx_valid and ch6_rx_allowed are both asserted, and the channel is enabled (ch6_rx_enabled==1)

10.2.3.5 Transmit Interface

X indicates channel number: (X = 5).

Table 22. Transmit Interface

Name	Description
chX_tx_clk	Transmit clock
chX_tx_ready	Transmit data is ready (X = 1 - 5) When high on rising edge of chX_tx_clk, the destination module is ready to accept the tx_data. When low on rising edge of chX_tx_clk, destination is not ready to accept the tx_data. When both chX_tx_valid and chX_tx_valid are high on rising edge of chX_tx_clk, both the transmitter (AXIQ channel) and the receiver must consider the data on tx_data taken.
chX_tx_valid	Transmit data is valid (X = 1 - 4) When high on rising edge of chX_tx_clk the data pins have valid data. When low on rising edge of chX_tx_clk the data pins do not have valid data. When both chX_tx_valid and chX_tx_valid are high on rising edge of chX_tx_clk, both the Transmitter (data producer AXIQ channel) and the receiver (data producer) must consider the data on tx_data taken.
ch5_tx_allowed	Channel 5 is allowed to transmit data The ch5_tx_allowed signal acts as external on/off switch for the channel. 0: Transmitter (data producer) is off. Data on ch1_5x_Q is not valid * 1: Transmitter (data producer) is on. * See tx_allowed to data valid latency.
chX_tx_I[11:0]	Transmit I data Channel X (12-bit data) (X = 1 - 5) Data is captured by the AXIQ on rising edge of chX_tx_clk when chX_tx_valid and chX_tx_allowed are both asserted, and the channel is enabled (chX_tx_enabled==1)
chX_tx_Q[11:0]	Transmit Q data Channel X (12-bit data) (X = 1 - 5) Data is captured by the AXIQ on rising edge of chX_tx_clk when chX_tx_valid and chX_tx_allowed are both asserted, and the channel is enabled (chX_tx_enabled==1)

10.2.3.6 Peak Detect Control Interface

Table 23. Peak Detect Control Interface

Name	Description
start_rx_max_search	The signals *start_max_search* is an input from the PHY Timer. A rising edge of the *start_max_search* signal causes the max_val search to begin, and reset the internal max_val to zero.
stop_rx_max_search	The signals *start_max_search* is an input from the PHY Timer. A rising edge of the *stop_max_search* signal saves the current max value into the saved_max_val buffer and stops the search.

10.3 Block Diagram

The following figure shows the block diagram for the AXIQ.

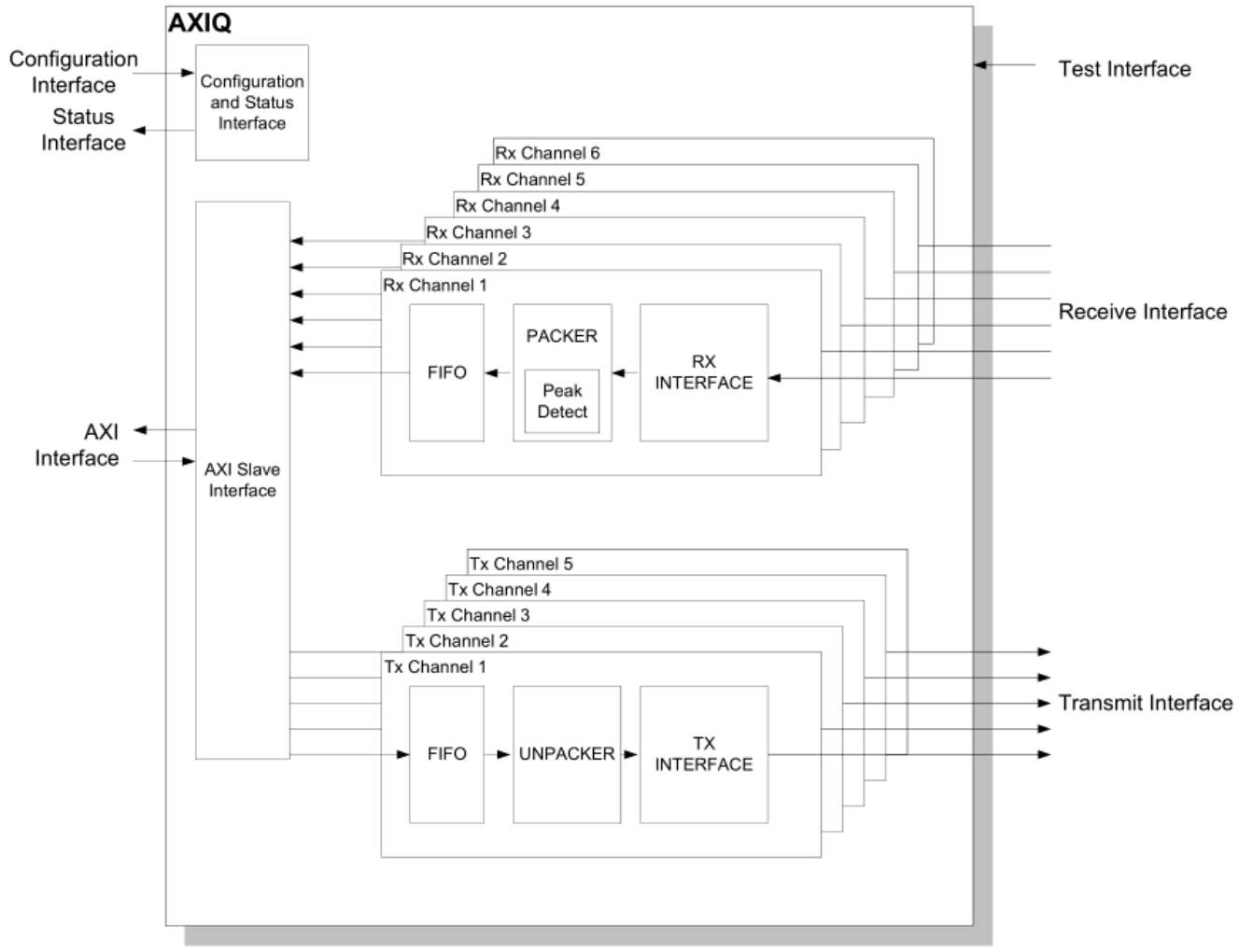


Figure 10. Block Diagram for AXIQ

10.4 Functional Description

10.4.1 AXI Slave Interface

The AXI Slave implements an AXI3 slave interface as described in the *AMBA AXI Protocol Specification.*

Both the AXI Read channel and the AXI Write channels are implemented on the AXIQ.

Features and limitations are:

- Parameterized number of bits in the transaction ID used for the AXI channels.
- The AXI Read channel and the AXI write channel can operate simultaneously.
- Out-of-order completion of transactions is unsupported.
  - The AXIQ responds to read transactions in the order reads are requested. Only a single outstanding transaction is supported. Once a transaction is outstanding, axiq_axi_aready will be negated until the transaction is completed.

  - Only a single outstanding write transaction is supported. Once a transaction is outstanding, axiq_axi_awready will be negated until the transaction is completed.

- Accesses are limited to aligned 128-bit only
  - Only axiq_axi_arsize[2:0] == 3'b100 is supported (128-bit data). Other values will generate SLVERR error indication on the axiq_axi_rresp.
  - Only axiq_axi_arburst == FIXED and axiq_axi_arburst == INCR are supported. Other values will generate SLVERR error indication on the axiq_axi_rresp.
  - Only axiq_axi_awsize[2:0] == 3'b100 is supported (128-bit data). Other values will generate SLVERR error indication on the axiq_axi_bresp.
  - Only axiq_axi_awburst == FIXED and axiq_axi_arburst == INCR are supported. Other values will generate SLVERR error indication on the axiq_axi_bresp.
- The AXI Read channel can read ONLY the Receive FIFOs for channels 1-6. The Transmit FIFOs 1-5 are not readable via the AXI interface.
- The AXI Write channel can write ONLY the Transmit FIFOs for channels 1-5. The Receive FIFOs 1-6 are not writeable via the AXI interface.
- Only the Receive Channels' FIFOs are readable via the AXI bus. The Transmit Channels' FIFOs are not readable via the AXI bus.
- Only the Transmit Channels' FIFOs are writeable able via the AXI bus. The Receive Channels' FIFOs are not writable via the AXI bus.

The following optional signal and responses are not used by the slave interface:

- Address Read Channel
  - ARLOCK
  - ARCACHE[3:0]
  - ARPROT[2:0]
  - AWLOCK
  - AWCACHE[3:0]
  - AWPROT[2:0]
- Read/Write Data and Response Channels
  - RRESP[1:0] responses OKAY and SLVERR
  - BRESP[1:0] responses OKAY and SLVERR

10.4.1.1 AXI Slave Memory Map

The AXI Slave interfaces maps (16 x 4KB = 64KB) regions to up to 16 write and 16 read channels. The AXIQ uses a single read region for the Receive FIFO (see "Read Addres Map" table) and a single write region for the Transmit FIFO (see "Write Address Map" table)>.

Table 24. Read Address Map

Device	ARADDR[15:12]	AXI_RADDR	Use VSPA DMA Channel
Reserved	0000	-	
Channel 1 Receive FIFO	0001	$AXI_BASE+0x0000_1000	1

Table continues on the next page...

Table 24. Read Address Map (continued)

Device	ARADDR[15:12]	AXI_RADDR	Use VSPA DMA Channel
Channel 2 Receive FIFO	0010	$AXI_BASE+0x0000_2000	2
Channel 3 Receive FIFO	0011	$AXI_BASE+0x0000_3000	3
Channel 4 Receive FIFO	0100	$AXI_BASE+0x0000_4000	4
Channel 5 Receive FIFO	0101	$AXI_BASE+0x0000_5000	5
Channel 6 Receive FIFO	0110	$AXI_BASE+0x0000_6000	6
Reserved	0111	-	-
Reserved	1000	-	-
Reserved	1001	-	-
Reserved	1010	-	-
Reserved	1011	-	-
Reserved	1100	-	-
Reserved	1101	-	-
Reserved	1110	-	-
Reserved	1111	-	-

Table 25. Write Address Map

Device	AWRADDR[15:12]	AXI_WADDR	Use VSPA DMA Channel
Reserved	0000	-	-
Reserved	0001	-	-
Reserved	0010	-	-
Reserved	0011	-	-
Reserved	0100	-	-
Reserved	0101	-	-
Reserved	0110	-	-
Reserved	0111	-	-
Reserved	1000	-	-

Table continues on the next page...

Table 25. Write Address Map (continued)

Device	AWRADDR[15:12]	AXI_WADDR	Use VSPA DMA Channel
Reserved	1001	-	-
Reserved	1010	-	-
Transmit Channel FIFO	1011	$AXI_BASE+0x0000_b000	11
Reserved	1100	-	-
Reserved	1101	-	-
Reserved	1110	-	-
Reserved	1111	-	-

10.4.2 FIFO

Each AXIQ Receive and Transmit channel has a FIFO. The AXIQ has a total of 7 FIFOs. The FIFO is a flip-flop based asynchronous FIFO. In this implementation, however, this FIFO is used with push/pop clocks that are edge aligned. This make it's operation synchronous. The FIFO is currently sized at 32 entries x 128-bit/entry: enough space to buffer up to two 16-beat bursts from AXI.

AXIQ FIFOs are offset from each other by 4 KB The intended association of VSPA DMA channel to AXIQ FIFO is to use the DMA AXIQ channel number in address bits 15:12 of the AXIQ The address mappings of the AXIQ channel FIFOs are shown in below table along with the intended association with VSPA DMA Channels.

Table 26. AXIQ FIFO Address Map and Intended VSPA DMA Association

AXIQ Channel	Intended VSPA DMA Channel	Address Offset from AXIQ Base	Notes
RX1	1	0x0000_1000	Observation Channel 0 ADC
RX2	2	0x0000_2000	Observation Channel 1 ADC
RX3	3	0x0000_3000	1X2 Receiver Channel 0 ADC
RX4	4	0x0000_4000	1x2 Receiver Channel 1 ADC
RX5	5	0x0000_5000	AUX Receive - Tx Detect
RX6	6	0x0000_6000	RSSI
-	7-10	0x0000_7000 - 0x0000_A000	Unused
TX5	11	0x0000_B000	1X2 Transmission Channel DAC

10.4.3 Receive DMA and Transmit DMA Trigger Consideration

The DMA interfaces primarily monitor the FIFO level and triggers the DMA whenever the FIFO level has gone beyond a programmed threshold. To avoid system issues related to buffered data in the AXI fabric and pipeline or synchronization delays on the trigger itself, the DMA interface negates the trigger while a transaction burst is occurring. Thus, the DMA will not receive/consider the trigger until its last AXI transaction completes. This should protect the DMA-AXIQ from accidently overflowing or underflowing.

10.4.3.1 VSPA DMA special Use With FIFOs

There is a special characteristic of the VSPA DMA that can be used to allow the user to control the burst beat count when transferring data to or from a FIFO.

Ordinarily, the VSPA DMA performs 16-beat burst transfers whenever possible in order to achieve the maximum possible bus utilization efficiency. The AXI protocol requires that a burst must never cross a 4K byte address boundary. So, the only times the DMA will use a burst size of less than 16 beats is when it would cause a 4K boundary crossing, or when there is less data to be transferred in order to complete the programmed command.

The DMA calculates and obeys a "virtual" 4K boundary restriction even when it is transferring to a fixed address, i.e. a FIFO. So, if the FIFO is decoded into a full 4K byte region where any access to the region is considered a FIFO access, this feature can be used to control the burst size. To use a 16 beat burst, the DMA should be programmed in fixed burst mode, with the starting address of the FIFO set to the lowest numerical address that will access the FIFO. To use a 8 beat burst, the DMA should be programmed in fixed burst mode, with the starting address of the FIFO set to the highest AXI bus width aligned numerical address that will access the FIFO. Consider the examples below:

FIFO address decode range 0x03000 - 0x03FFF (4K byte address space) for channel 3.

AXI data bus width of 128 bits (16 bytes)

16 beat burst - set DMA AXI address to 0x03000 (any aligned address 0x03000 - 0x03F00 would work)

8 beat burst - set DMA AXI address to 0x03F80 (0x04000 - (8 beats * 16 bytes))

4 beat burst - set DMA AXI address to 0x03FC0 (0x04000 - (4 beats * 16 bytes))

2 beat burst - set DMA AXI address to 0x03FE0 (0x04000 - (2 beats * 16 bytes))

Luckily, the AXIQ maps 4KB region for each read and write and channel.

10.4.3.2 AXIQ DMA INTERFACE FLUSH MODE

The DMA interfaces can also enter a special mode called FLUSH mode.

Receive Channel Flush Mode:

Normally, the dma_rx_trigger follows the FIFO level, letting the DMA know when there is enough data to complete another DMA read burst. When shutting down a receive link, there might be transfers still pending at the DMA. To allow the DMA to finish its outstanding transactions, the DMA interface enters FLUSH mode. In FLUSH mode the trigger is always 1 (regardless of the level of the FIFO), thus continually triggering the DMA to transfer until all its pending transactions are complete. The data sent from the AXIQ to the DMA is garbage as the AXIQ channel stopped receiving data (and thus is in flush mode). This axi data, while might be written by the DMA to the memory buffer, does not contain valid data and should be ignored.

DMA FLUSH mode is entered by using the falling edge of async_chan_rx_en. DMA FLUSH mode is exited when the DMA interface receives the dma_rx_ptr_rst_req = 1 signal.

Transmit Channel Flush Mode:

Normally, the dma_tx_trigger follows the FIFO level, letting the DMA know when there is enough room to complete another DMA write burst. When shutting down a transmit link, there might be transfers still pending at the DMA. To allow the DMA to finish its outstanding transactions, the DMA interface enters FLUSH mode. In FLUSH mode the trigger is always 1 (regardless of the level of the FIFO), thus continually triggering the DMA to transfer until all its pending transactions are complete. The data received from the DMA while in flush mode is ignored by the AXIQ.

DMA FLUSH mode is entered by using the falling edge of async_chan_tx_en. DMA FLUSH mode is exited when the DMA interface transmits the dma_tx_ptr_rst_req = 1 signal.

10.4.4 Rx I/Q Packer and Tx I/Q Unpacker

The unpacker/packer is used to unpack/pack samples from/to a 128-bit word destined for transfer over AXI. The unpacker/packer also handles the I/Q swap. The channel type (Rx or Tx) and the channel number determine how samples are packed/unpacked to/from the 128-bit AXI word.

For most channels, 4 x 32-bit I/Q pairs (16-bit I and 16-bit Q) are packed as shown in Table 27 .

A configuration bit is defined for the receive paths (*static_rx_swap_IQ*) and the transmit paths (*static_tx_swap_IQ*) that will swap the order of I and Q. The swapped mappings are shown in Table 28.

For Receive Channel 5, 8 x 16-bit data words are packed as shown in Table 29 .

For Receive Channel 6, 8-bit receive data packing, please refer to Channel 6 Half-Word Count detailed description for details regarding packing channel 6 receive data.

VSPA processes I/Q samples as 16-bit each, both for Transmit and Receive. However, the I/Q samples received and transmitted via Receive Channels 1-4 and Transmit Channels 1-5 are 12-bit I/Q data.

For the Receive channels, the AXIQ does the 12-bit to 16 bit conversion as it receives the data. The 12-bit data is padded by 4 zeros at the lower 4 bits as follows: I/Q[15:0] = {chX_rx_I/Q[11:0], 4'b0}. Similarly, for Receive Channel 5, the 12-bit data is padded by 4 zeros at the lower 4 bits as follows: data[15:0] = {ch5_rx_data[11:0], 4'b0}.

For the Transmit channels, the AXIQ does the 12-bit to 16 bit conversion as it receives the data. The 16-bit data is rounded from the lower 4 bits as follows: chX_rx_I/Q[11:0] = I/Q[11:0] + round_value(I/Q[3:0]). Rounding is preferred on truncating as it add about 1/2 accuracy and improves the SNR.

Note, no data conversion is done on Receive Channel 6. The 8-bit received data is packed as illustrated in section Channel 6 Half-Word Count detailed description.

Table 27. 16-bit I/Q AXI Word Packing(Rx) and Unpacking (Tx) (non-swapped)

		AXI 128-bit Word															
		127	112	111	96	95	80	79	64	63	48	47	32	31	16	15	0
AXI 16-beat burst	0	Q3[15:0]		I3[15:0]		Q2[15:0]		I2[15:0]		Q1[15:0]		I1[15:0]		Q0[15:0]		I0[15:0]	
	1	Q7[15:0]		I7[15:0]		Q6[15:0]		I6[15:0]		Q5[15:0]		I5[15:0]		Q4[15:0]		I4[15:0]	
	2	Q11[15:0]		I11[15:0]		Q10[15:0]		I10[15:0]		Q9[15:0]		I9[15:0]		Q8[15:0]		I8[15:0]	
	3	Q15[15:0]		I15[15:0]		Q14[15:0]		I14[15:0]		Q13[15:0]		I13[15:0]		Q12[15:0]		I12[15:0]	
	4	Q19[15:0]		I19[15:0]		Q18[15:0]		I18[15:0]		Q17[15:0]		I17[15:0]		Q16[15:0]		I16[15:0]	
	5	Q23[15:0]		I23[15:0]		Q22[15:0]		I22[15:0]		Q21[15:0]		I21[15:0]		Q20[15:0]		I20[15:0]	
	6	Q27[15:0]		I27[15:0]		Q26[15:0]		I26[15:0]		Q25[15:0]		I25[15:0]		Q24[15:0]		I24[15:0]	
	7	Q31[15:0]		I31[15:0]		Q30[15:0]		I30[15:0]		Q29[15:0]		I29[15:0]		Q28[15:0]		I28[15:0]	
	8	Q35[15:0]		I35[15:0]		Q34[15:0]		I34[15:0]		Q33[15:0]		I33[15:0]		Q32[15:0]		I32[15:0]	
	9	Q39[15:0]		I39[15:0]		Q38[15:0]		I38[15:0]		Q37[15:0]		I37[15:0]		Q36[15:0]		I36[15:0]	
	10	Q43[15:0]		I43[15:0]		Q42[15:0]		I42[15:0]		Q41[15:0]		I41[15:0]		Q40[15:0]		I40[15:0]	
	11	Q47[15:0]		I47[15:0]		Q46[15:0]		I46[15:0]		Q45[15:0]		I45[15:0]		Q44[15:0]		I44[15:0]	
	12	Q51[15:0]		I51[15:0]		Q50[15:0]		I50[15:0]		Q49[15:0]		I49[15:0]		Q48[15:0]		I48[15:0]	
	13	Q55[15:0]		I55[15:0]		Q54[15:0]		I54[15:0]		Q53[15:0]		I53[15:0]		Q52[15:0]		I52[15:0]	

Table continues on the next page...

Table 27. 16-bit I/Q AXI Word Packing(Rx) and Unpacking (Tx) (non-swapped) (continued)

	14	Q59[15:0]	I59[15:0]	Q58[15:0]	I58[15:0]	Q57[15:0]	I57[15:0]	Q56[15:0]	I56[15:0]
	15	Q63[15:0]	I63[15:0]	Q62[15:0]	I62[15:0]	Q61[15:0]	I61[15:0]	Q60[15:0]	I60[15:0]

Table 28. 16-bit I/Q AXI Word Packing(Rx) and Unpacking (Tx) (swapped)

		AXI 128-bit Word															
		127	112	111	96	95	80	79	64	63	48	47	32	31	16	15	0
AXI 16-beat burst	0	I3[15:0]		Q3[15:0]		I2[15:0]		Q2[15:0]		I1[15:0]		Q1[15:0]		I0[15:0]		Q0[15:0]	
	1	I7[15:0]		Q7[15:0]		I6[15:0]		Q6[15:0]		I5[15:0]		Q5[15:0]		I4[15:0]		Q4[15:0]	
	2	I11[15:0]		Q11[15:0]		I10[15:0]		Q10[15:0]		I9[15:0]		Q9[15:0]		I8[15:0]		Q8[15:0]	
	3	I15[15:0]		Q15[15:0]		I14[15:0]		Q14[15:0]		I13[15:0]		Q13[15:0]		I12[15:0]		Q12[15:0]	
	4	I19[15:0]		Q19[15:0]		I18[15:0]		Q18[15:0]		I17[15:0]		Q17[15:0]		I16[15:0]		Q16[15:0]	
	5	I23[15:0]		Q23[15:0]		I22[15:0]		Q22[15:0]		I21[15:0]		Q21[15:0]		I20[15:0]		Q20[15:0]	
	6	I27[15:0]		Q27[15:0]		I26[15:0]		Q26[15:0]		I25[15:0]		Q25[15:0]		I24[15:0]		Q24[15:0]	
	7	I31[15:0]		Q31[15:0]		I30[15:0]		Q30[15:0]		I29[15:0]		Q29[15:0]		I28[15:0]		Q28[15:0]	
	8	I35[15:0]		Q35[15:0]		I34[15:0]		Q34[15:0]		I33[15:0]		Q33[15:0]		I32[15:0]		Q32[15:0]	
	9	I39[15:0]		Q39[15:0]		I38[15:0]		Q38[15:0]		I37[15:0]		Q37[15:0]		I36[15:0]		Q36[15:0]	
	10	I43[15:0]		Q43[15:0]		I42[15:0]		Q42[15:0]		I41[15:0]		Q41[15:0]		I40[15:0]		Q40[15:0]	
	11	I47[15:0]		Q47[15:0]		I46[15:0]		Q46[15:0]		I45[15:0]		Q45[15:0]		I44[15:0]		Q44[15:0]	
	12	I51[15:0]		Q51[15:0]		I50[15:0]		Q50[15:0]		I49[15:0]		Q49[15:0]		I48[15:0]		Q48[15:0]	
	13	I55[15:0]		Q55[15:0]		I54[15:0]		Q54[15:0]		I53[15:0]		Q53[15:0]		I52[15:0]		Q52[15:0]	
	14	I59[15:0]		Q59[15:0]		I58[15:0]		Q58[15:0]		I57[15:0]		Q57[15:0]		I56[15:0]		Q56[15:0]	
	15	I63[15:0]		Q63[15:0]		I62[15:0]		Q62[15:0]		I61[15:0]		Q61[15:0]		I60[15:0]		Q60[15:0]	

Table 29. 16-bit data AXI Word Packing - channel 5 Receive

		AXI 128-bit Word															
		127	112	111	96	95	80	79	64	63	48	47	32	31	16	15	0
AXI 16-beat burst	0	data7[15:0]		data6[15:0]		data5[15:0]		data4[15:0]		data3[15:0]		data2[15:0]		data1[15:0]		data0[15:0]	

Table continues on the next page...

Table 29. 16-bit data AXI Word Packing - channel 5 Receive (continued)

	1	data15[15:0]	data14[15:0]	data13[15:0]	data12[15:0]	data11[15:0]	data10[15:0]	data9[15:0]	data8[15:0]
	2	data23[15:0]	data22[15:0]	data21[15:0]	data20[15:0]	data19[15:0]	data18[15:0]	data17[15:0]	data16[15:0]
	3	data31[15:0]	data30[15:0]	data29[15:0]	data28[15:0]	data27[15:0]	data26[15:0]	data25[15:0]	data24[15:0]
	4	data39[15:0]	data38[15:0]	data37[15:0]	data36[15:0]	data35[15:0]	data34[15:0]	data33[15:0]	data32[15:0]
	5	data47[15:0]	data46[15:0]	data45[15:0]	data44[15:0]	data43[15:0]	data42[15:0]	data41[15:0]	data40[15:0]
	6	data55[15:0]	data55415:0]	data53[15:0]	data52[15:0]	data51[15:0]	data50[15:0]	data49[15:0]	data48[15:0]
	7	data63[15:0]	data62[15:0]	data61[15:0]	data60[15:0]	data59[15:0]	data58[15:0]	data57[15:0]	data56[15:0]
	8	data71[15:0]	data70[15:0]	data69[15:0]	data68[15:0]	data67[15:0]	data66[15:0]	data65[15:0]	data64[15:0]
	9	data79[15:0]	data78[15:0]	data77[15:0]	data76[15:0]	data75[15:0]	data74[15:0]	data73[15:0]	data72[15:0]
	10	data87[15:0]	data86[15:0]	data85[15:0]	data84[15:0]	data83[15:0]	data82[15:0]	data81[15:0]	data80[15:0]
	11	data95[15:0]	data94[15:0]	data93[15:0]	data92[15:0]	data91[15:0]	data90[15:0]	data89[15:0]	data88[15:0]
	12	data103[15:0]	data102[15:0]	data101[15:0]	data100[15:0]	data99[15:0]	data98[15:0]	data97[15:0]	data96[15:0]
	13	data111[15:0]	data110[15:0]	data109[15:0]	data108[15:0]	data107[15:0]	data106[15:0]	data105[15:0]	data104[15:0]
	14	data119[15:0]	data118[15:0]	data117[15:0]	data116[15:0]	data115[15:0]	data114[15:0]	data113[15:0]	data112[15:0]
	15	data127[15:0]	data126[15:0]	data125[15:0]	data124[15:0]	data123[15:0]	data122[15:0]	data121[15:0]	data120[15:0]

10.4.5 Receive Interface

The AXIQ Receive Interface is to communicate with an external modules which provide the receive data (12-bit I/Q, 12-bit data, or 8-bit data). The AXIQ acts as a FIFO to the external module as shown in the following figure. The AXIQ packs 8-bit, 16-bit or 32-bit words (channel number dependent) received from the external module as received into 128-bit AXI words (no data format assumed by the AXIQ). The receive interfaces use the valid signal to capture the data.

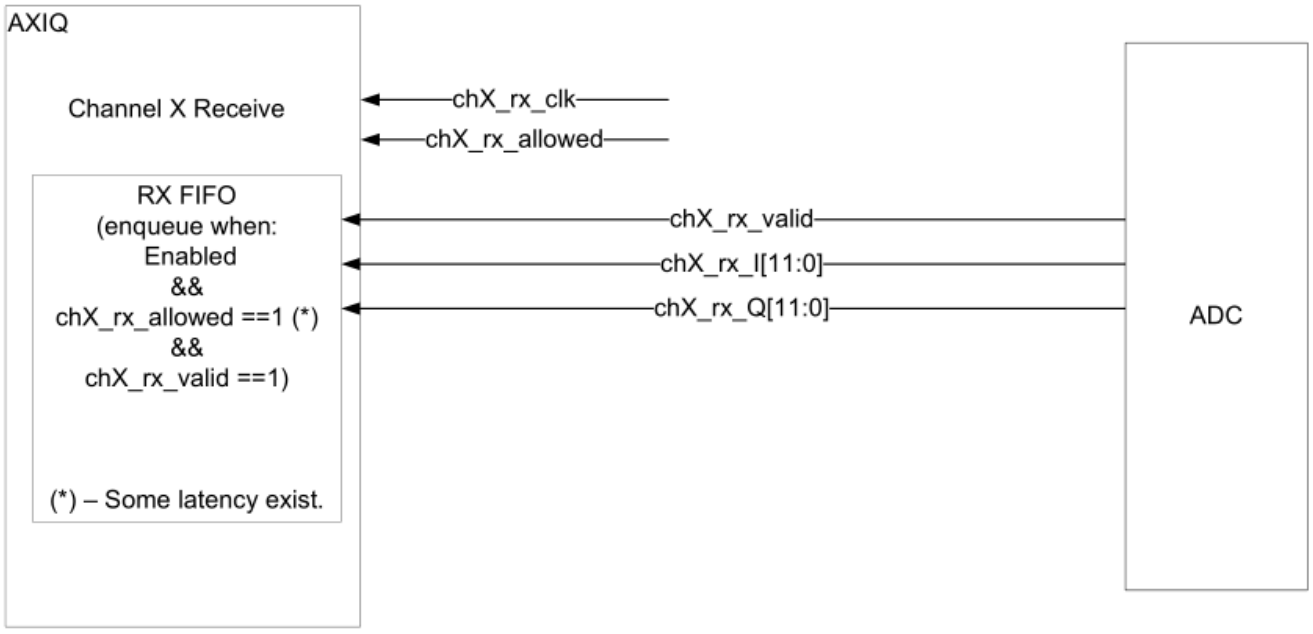


Figure 11. Concept of Receive Interface

The receive interface uses a valid signal

valid: output of the producer (external module such as ADC); 1 when the data contains valid data, 0 otherwise

ready: The AXIQ channel is ready when the channel is enabled and its FIFO is NOT full;

The figure below shows 2 pieces of data being received.

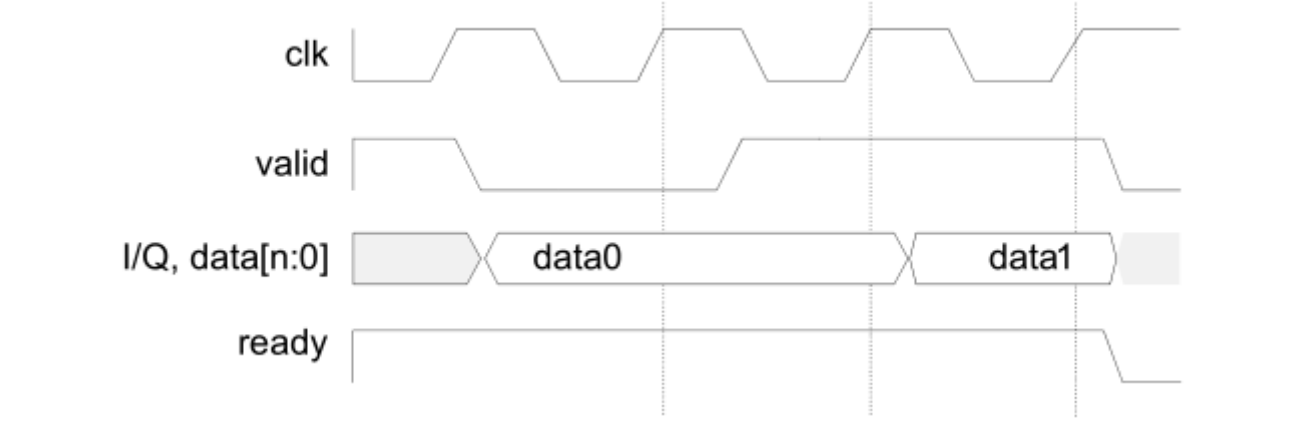


Figure 12. Receive Valid and Data

10.4.6 Transmit Interface

The AXIQ Transmit Interface is to communicate with an external modules which accepts the transmit data (12-bit I/Q) from the AXIQ. The AXIQ acts as a FIFO to the external module as shown in the following figure. The AXIQ extracts tx data from 128-bit AXI words. The AXIQ constructs 2 x16-bit (32-bit) words per sample (four times per 128-bit AXI data word) transmitted to the external module (data rounding is performed by the AXIQ). The transmit interfaces use the ready signal to advance the data.

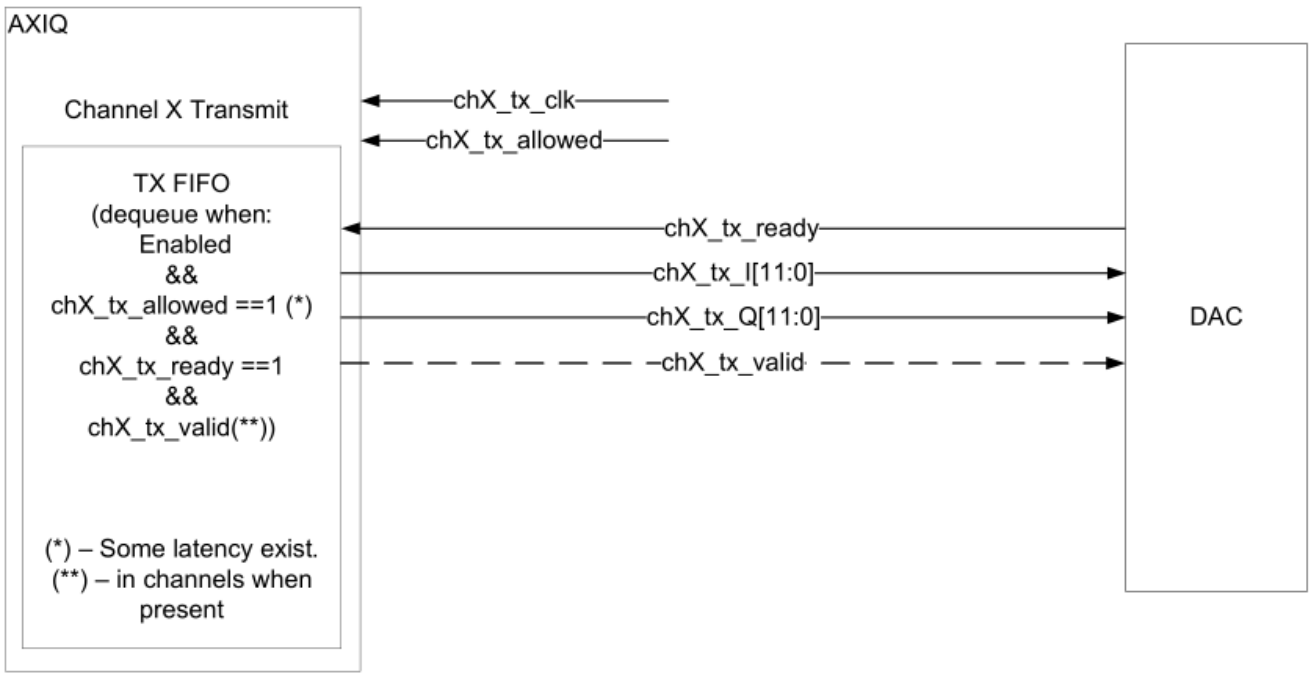


Figure 13. Concept of Transmit Interface

The transmit interface uses a ready signal and a valid signal (only on channel 5).

ready: output of the data consumer (external module in this case);

The external module is ready to accept AXIQ tx_data;

valid: output of the producer (AXIQ in this case);

1 when the tx_data contains valid data, 0 otherwise

The AXIQ assume the data is consumed by the consumer when both the tx_valid==1 and the tx_ready==1 (in case of channel 1 - when tx_ready==1 and the AXIQ has valid data (internal tx_valid==1). The AXIQ will continue to drive the same tx_data until is consumed.

The figure below shows 2 pieces of data being transmitted.

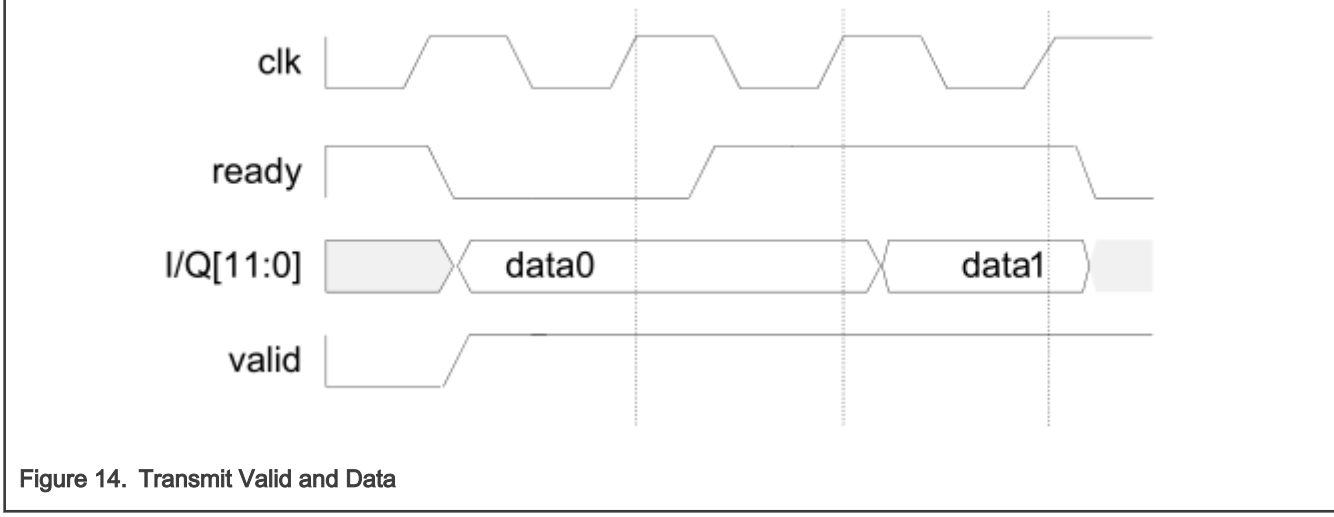


Figure 14. Transmit Valid and Data

10.4.7 Receive Peak (Max Value) Detect detailed description

The AXIQ can detect max value of the incoming I/Q samples on Receive Channels 1-4. Each channel has a circuitry that detect the maximum of the absolute value of the I/Q streams. Each channel outputs a signal *chX_max_rx_data[15:0] (x=1-4)* which carries the maximum absolute value during the period starting at the rising edge of the signal *start_max_search* and ending at the rising edge of the signal *stop_max_search.*

The signals *start_max_search* and *stop_max_search* are inputs from the PHY Timer. All four receive channels (ch1_rx-ch4_rx) use the same signals to control the search engine start and stop.

The AXIQ routes 4x16-bit signals, c*hX_max_rx_data[15:0] (x=1-4), with* the saved 16-bit peak (max) value, one fro for each channel 1-4. These signals are connected via the axiq_stat bus to VSPA GPIN registers.

During the search period, the max search circuitry looks for the max value of the absolute value of I, the absolute value of Q, and previous max_val. Here is pseudo code representing the circuitry functionality:

```
Initially, reset max_val to zero;
foreach I,Q sample
    {max_val = max(max_val, abs(I), abs(Q)}
```

A rising edge of the *start_max_search* signal causes the max_val search to begin, and reset the internal max_val to zero.

A rising edge of the *stop_max_search* signal saves the current max value into the saved_max_val buffer and stops the search.

Additional VSPA_GPOUT signal, *vspa_restart_rx_max_bit,* is used to control the search starting and stopping. A rising edge of *vspa_restart_rx_max_bit* saves the current max value into the saved_max_val buffer signal and causes the search to restart. All four receive channels (ch1_rx-ch4_rx) use the same signal to control the search engine restart.

The signals *chX_max_rx_data[15:0] (x=1-4)* are updated when a rising edge of the *stop_max_search* or the rising edge of the *vspa_restart_rx_max_bit* is detected. The *chX_max_rx_data[15:0]* signals are not changed during the search.

Note: the *stop_max_search* from the PHY Timer is also connected a VSPA External_Go input. This simplifies VSPA running as result of the stop the timer (and capture saved_max_val) event.

Integration Requirements:

1.The AXIQ and VSPA get the same clock.

2.The PHY Timer signals are generated at the same clock domain as the AXIQ RX/TX clocks.

10.4.8 Channel 6 Half-Word Count detailed description

VSPA needs to process RSSI data when a (relatively) small quantity of data arrives. The following mechanism is designed to support this requirement.

A 4-bit counter value, ch6_rx_wcnt[3:0], is sent from VSPA GPOUT to the AXIQ via the axiq_config bus "axiq_config[115:112]".

VSPA sets the value of this counter when configuring the AXIQ, as well as setting the same value*2 in the byte count parameter of the DMA channel transferring the data from the AXIQ Rx Channel 6 (RSSI) to DMEM (of course, the external trigger option must also set for this DMA transfer).

The value in ch6_rx_wcnt[3:0] determines when AXIQ assert the DMA external trigger. When receiving data, the AXIQ counts the bytes received in the RSSI (ch_6_rx) channel. When the count reaches the value in the (2*ch6_rx_wcnt[3:0]), the AXIQ channel will assert the dma_external_trigger which, in turn, would cause the DMA to start transferring the data to DMEM. Additionally, since the DMA byte count is set to the (2*ch6_rx_wcnt[3:0]), the DMA will assert the VSPA_DMA_GO upon completion of the transfer, if so programmed.

The value in ch6_rx_wcnt[3:0] determines data organization in memory. The AXIQ will start filling a new line after the count reached the value(2*ch6_rx_wcnt[3:0]); thus, next DMA transfer would have the next data (data of next buffer) (re)aligned to the axi data width.

For the RSSI AXIQ channel (ch6_rx), when the value of the 'ch6_rx_wcnt' counter is zero, the AXIQ uses the default threshold value programmed in the 'ch6_rx_fifo_thresh[1:0]' field. When the value in the 'ch6_rx_wcnt' counter is non-zero, the threshold value programmed in the 'ch6_rx_fifo_thresh[1:0]' field is ignored and the value of the 'ch6_rx_wcnt' counter is used instead.

An example of data arrangement in FIFO with ch6_rx_wcnt[3:0] = 10 - i.e. (2*10=) 20 bytes:

```
      Bit Num: 127.120 119.112 111.104 103.96 95..88 87..80 79..72 71..64 63..56 55..48 47..40 39..32
31..24 23..16 15..8 7..0
      FIFO_Line_0: Byte15 Byte14 Byte13 Byte12 Byte11 Byte10 Byte9 Byte8 Byte7 Byte6 Byte5 Byte4
Byte3 Byte2 Byte1 Byte0
      FIFO_Line_1: junk junk junk junk junk junk junk junk junk junk junk junk Byte19 Byte18 Byte17
Byte16
      FIFO_Line_2: Byte35 Byte34 Byte33 Byte32 Byte31 Byte30 Byte29 Byte28 Byte27 Byte26 Byte25
Byte24 Byte23 Byte22 Byte21 Byte20
      FIFO_Line_3: junk junk junk junk junk junk junk junk junk junk junk junk Byte39 Byte38 Byte37
Byte36
      FIFO_Line_4: Byte55 Byte54 Byte53 Byte52 Byte51 Byte50 Byte49 Byte48 Byte47 Byte46 Byte45
Byte44 Byte43 Byte42 Byte41 Byte40
      FIFO_Line_5: junk junk junk junk junk junk junk junk junk junk junk junk Byte59 Byte58 Byte57
Byte56
```

FIFO_Line_0 and FIFO_Line_1 will be stored in the first DMA buffer.

FIFO_Line_2 and FIFO_Line_3 will be stored in the second DMA buffer.

FIFO_Line_4 and FIFO_Line_5 will be stored in the third DMA buffer.

etc

The bytes marked 'junk' would not be saved in the DMA buffer as they exceed the DNA buffer size (byte count).

Note: the pointer_reset_request (ptr_rst_req) bit in the DMA control field for this transfer should be set to zero. Setting this bit to one may result is a loss of a few RSSI bytes.

As with other AXIQ channels, the pointer_reset_request (ptr_rst_req) bit in the DMA control field should be set to one when setting the AXIQ channel enable bit to zero, thus shutting off this channel.

10.5 Operating Modes

10.5.1 Low-Power Mode

The AXIQ is in low-power mode if all its receive channels are disabled ({chX_rx_en} == 2'b00).

10.6 Resets

The AXIQ input has a single asynchronous reset input: axi_rst_b.

10.7 Interrupts

The AXIQ generates one interrupt request signal. The Interrupt Request Signal is expected to be connected to the ARM Host interrupt Request input. Optionally, the AXIQ interrupt request signal can be connected to VSPA's ext_go input to perform similar functionality with VSPA. Refer to VSPA Creation and Integration Guide for more details.

10.8 Performance

Performance is based on the rx_clk and tx_clk rates selected.

10.9 VSPA GPIO Registers Mapping

This section is a suggestion for the VSPA GPIO mapping.

10.9.1 GPOUT Mapping

Table 30. VSPA GPOUT mapping

Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPOUT 4	Reserved			ch2 Rx clear errors	ch2 Rx swap	ch2 Rx fifo thresh		ch2 Rx En	Reserved			ch1 Rx clear errors	ch1 Rx swap	ch1 Rx fifo thresh		ch1 Rx En
	Reserved			ch4 Rx clear errors	ch4 Rx swap	ch4 Rx fifo thresh		ch4 Rx En	Reserved			ch3 Rx clear errors	ch3 Rx swap	ch3 Rx fifo thresh		ch3 Rx En
GPOUT 5	Reserved			ch6 Rx clear errors	Reserved	ch6 Rx fifo thresh		ch6 Rx En	Reserved			ch5 Rx clear errors	Reserved	ch5 Rx fifo thresh		ch5 Rx En
	Reserved				ch6_rx_wcnt				Reserved							
GPOUT 6	Reserved															
	Reserved															
GPOUT 7	Reserved											ch5 Tx clear errors	ch5 Tx swap	ch5 Tx fifo thresh		ch5 Tx En
	vspa restart rx max bit	Reserved														

10.9.2 GPIN Mapping

Table 31. GPIN mapping

Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Table continues on the next page...

Table 31. GPIN mapping (continued)

GPIN 0	Ch4 rx overflow	Ch4 rx underflow	Ch4 rx fifo not empty	Ch4 rx enabled	Ch3 rx overflow	Ch3 rx underflow	Ch3 rx fifo not empty	Ch3 rx enabled	Ch2 rx overflow	Ch2 rx underflow	Ch2 rx fifo not empty	Ch2 rx enabled	Ch1 rx overflow	Ch1 rx underflow	Ch1 rx fifo not empty	Ch1 rx enabled
	Reserved								Ch6 rx overflow	Ch6 rx underflow	Ch6 rx fifo not empty	Ch6 rx enabled	Ch5 rx overflow	Ch5 rx underflow	Ch5 rx fifo not empty	Ch5 rx enabled
GPIN 1	-	Cannot generate underflow error because of valid-ready protocol on TX channel	-	-	-	Cannot generate underflow error because of valid-ready protocol on TX channel	-	-	-	Cannot generate underflow error because of valid-ready protocol on TX channel	-	-	-	Cannot generate underflow error because of valid-ready protocol on TX channel	-	-
	Reserved												Ch5 tx overflow	Ch5 tx underflow	Ch5 tx fifo not full	Ch5 tx enabled
GPIN 2	ch1_max_rx_data															
	ch2_max_rx_data															
GPIN 3	ch3_max_rx_data															
	ch4_max_rx_data															

Chapter 11
Access Error Management (AEM)

11.1 Introduction

The Access Error Management Block (AEM) is used for the detection, error logging, and reporting of all decode access errors on an interconnect .

11.2 Overview

This section contains the following:

11.2.1 Features

Features of the AEM include the following:

- Detection of AXI transactions with decode errors
- Error logging using capture registers for the failing transaction
- User-configured signalling on detections, including:
  - Error Interrupt generation
  - Performance event generation
- Set of auxilliary ports provide access to AEM's error logging and signaling capabilities for target IP blocks which do not natively possess this
- All features controllable and configurable via memory map access by development tools and by embedded software.

Parameterization determines the capabilities of the AEM on a particular instantiation.

11.2.2 Block Diagram

One Access Error Management Block (AEM) connects up to a single output port of each interconnect requiring proper error handling, as shown in the system view found in the figure below. In addition, a set of auxilliary error logging ports are available, provide error logging and signaling capabilities for target IP blocks which do not have this capability.

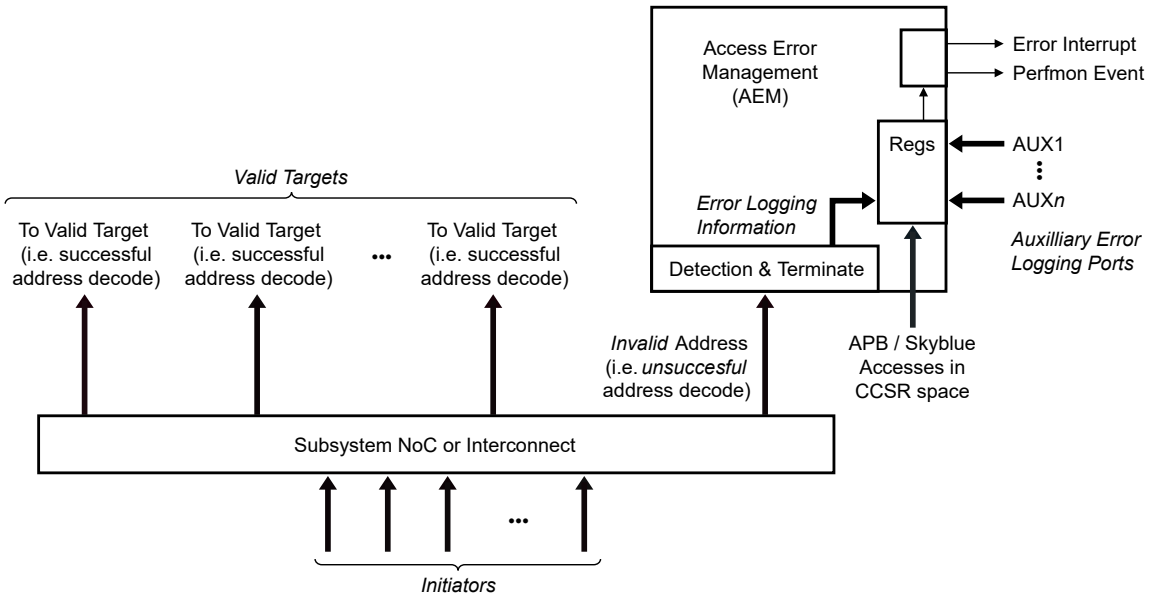


Figure 15. System View showing AEM Block Diagram Connected to an Interconnect

11.2.3 Operation

As initiators issues transactions into the interconnect with valid target addresses, these are routed to the "Valid Target[s]" shown coming out of the interconnect. The AEM block does not receive any of these transactions and nothing is signalled.

Whenever an initiator issues a transaction with an address that does not decode to a valid target, then the interconnect must route the transaction to a special target port which connects to the AEM block.

The AEM block, on receiving one of these transactions, properly terminates the transactions signalling an AXI 'decode error' and also does the following:

- Captures the address and attributes of the first failing transaction in a set of capture registers
- Optionally delivers an interrupt on the first failing transaction if enabled
- Pulses the "Perfmon Event" signal for all failing transactions which connects to the supporting debug infrastructure

Errors reported from Targets through the AUXn Expansion Ports

The block can also be optionally configured by an SoC to receive errors to be logged and reported through its expansion AUX ports.

When an error is received from a target block on any of these ports, the AEM block, on receiving a reported error (as indicated by the valid bit on the AUX expansion bus), does the following:

- Captures the address and attributes of the first failing transaction in a set of capture registers (where this information is provided to the AEM block through the AUX expansion bus)
- Optionally delivers an interrupt on the first failing transaction if enabled
- Pulses the "Perfmon Event" signal for all failing transactions which connects to the supporting debug infrastructure

Rearming the AEM Block

A simple mechanism is also provided for re-arming the AEM block for future detections.

11.2.4 Interfaces

The AEM contains the following interfaces:

Standard Buses

Standard transactions buses include the following:

- One Target AXI bus (parameterized width) to receive transactions with decode errors from interconnect
- One 32-bit APB (or Skyblue) bus for access by the SoC to AEM configuration registers
- Implementation Note: The AEM does not necessarily need to receive the AXI data links so there are opportunities for reductions in connectivity

Expansion Buses

AEM expansion input buses include the following:

- 0 to 8 AUX ports, each with the width necessary to capture all error logging information and one valid bit

NOTE

If an AUXn port is used for reporting additional errors from other target sources not supporting error capture registers and error signalling, all attributes reportable in AEM's error capture registers must be correctly provided on the AUX expansion ports (address, AxID, AxUSER, AXI attributes, etc.).

Output Signals

AEM output signaling includes the following:

- One Error Interrupt signal for delivery of a detected access event to the appropriate interrupt controller
- One performance monitoring output for delivery to a debug subsystem

Input Signals

- clock
- reset

11.3 Modes of Operation

There are two simple modes of operation:

- Reset
- Functional

11.3.1 Reset

The reset mode is entered when the AEM's reset terminal is asserted. It is recommended that this terminal is tied to an SoC's power-on reset, hard reset, and warm resets, i.e. any type of reset.

11.3.2 Functional

After reset, the block enters its functional mode. By default, error checking and reporting is turned off and must be explicitly enabled by software.

11.4 Initialization Information

This block should be properly initialized for detection and proper delivery of the detection event for every interconnect which it supports. Initialization should be done before runtime operation is enabled, i.e., as part of the boot process.

11.5 Application Information

Warm Reset of the AEM

A warm reset of this block can be done simply by the following steps:

- Write all AEM registers with their default values out of reset

11.6 AEM register descriptions

AEM Control and Status Registers

These registers enable detection and interrupt reporting as well as providing appropriate status.

AEM Error Capture Registers

General Description of Capture Register Operation

A set of capture registers is provided for capturing the address and attributes associated with the first detected error either from a transaction received or from any of the AUX ports. After the first capture, the registers are then locked and subsequent errors are not captured.

A write 1 to clear operation to any of the error status bits in the ERR_STATUS register (DEC_ERR, AUXn_ERR) clears the capture condition, allowing another error to be captured.

There is no capture of data because these interconnects do not detect errors on data.

11.6.1 AEM Memory map

AEM base address: 120_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Error Detect Register (ERR_DETECT)	32	RW	0000_0000h
4h	Error Status Register (ERR_STATUS)	32	RW	0000_0000h
8h	Interrupt Enable Register (INT_EN)	32	RW	0000_0000h
100h	Error Address Capture (CAPTURE_ADDRESS)	32	RW	0000_0000h
104h	Error Extended Address Capture (CAPTURE_EXT_ADDRESS)	32	RW	0000_0000h
110h	Error Attributes Capture Register 1 (CAPTURE_ATTRIBUTES1)	32	RW	0000_0000h
114h	Error Attributes Capture Register 2 (CAPTURE_ATTRIBUTES2)	32	RW	0000_0000h
118h	Error Attributes Capture Register 3 (CAPTURE_ATTRIBUTES3)	32	RW	0000_0000h

11.6.2 Error Detect Register (ERR_DETECT)

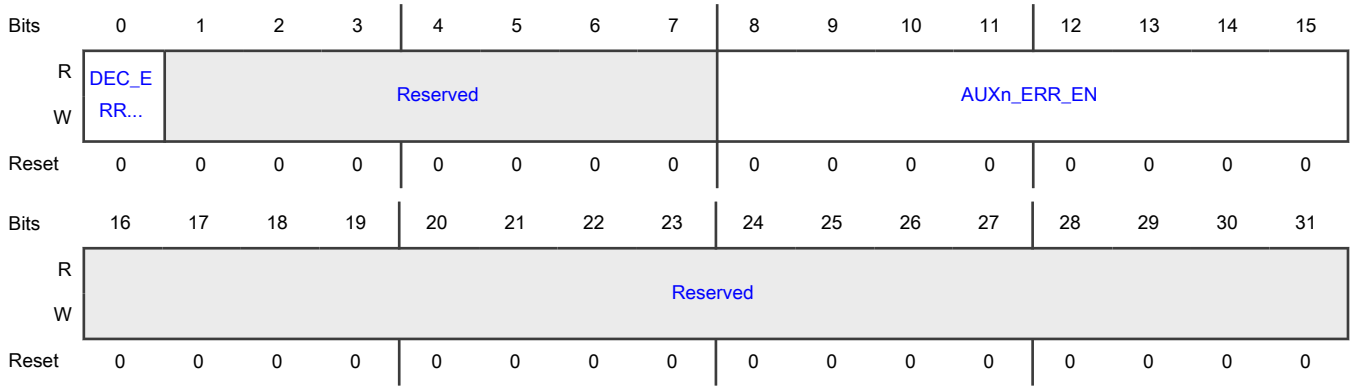
Offset

Register	Offset
ERR_DETECT	0h

Function

ERR_DETECT enables detection of individual error sources.

Diagram



Fields

Field	Function
0 DEC_ERR_EN	DEC_ERR_EN Decode Error Enable. This bit enables detection of decode errors received by the AEM block. 0b - Detection of Decode Errors by this block is disabled. 1b - Detection of Decode Errors by this block is enabled.
1-7 —	- Reserved
8-15 AUXn_ERR_EN	AUXn_ERR_EN Error Enable for AUXn ports (1 bit per port). This bit enables detection of decode errors received by the AEM block. 00000000b - Reception of Decode Errors on this AUX port is disabled. 00000001b - Reception of Decode Errors on this AUX port is enabled.
16-31 —	- Reserved

11.6.3 Error Status Register (ERR_STATUS)

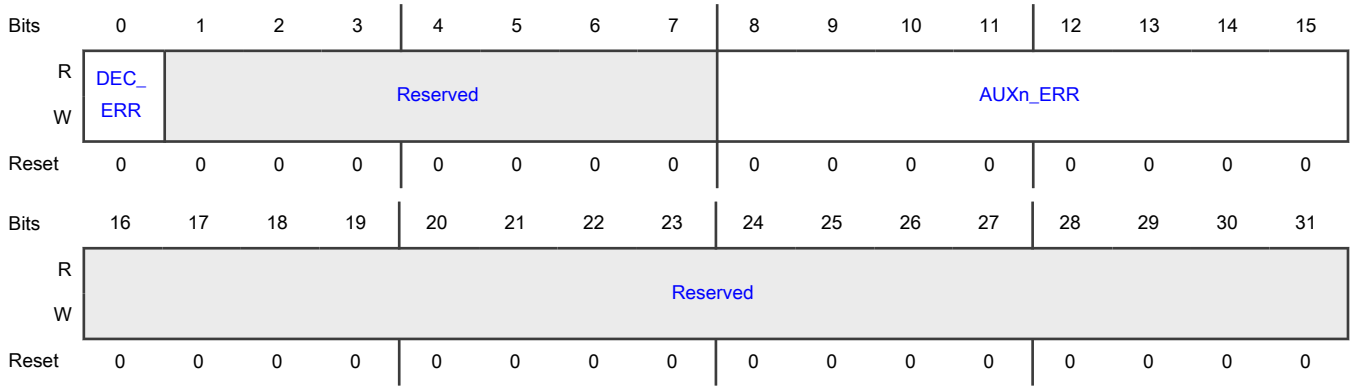
Offset

Register	Offset
ERR_STATUS	4h

Function

ERR_STATUS captures error status for individual error sources. This register captures only the first source reporting an error among all the possible sources.

Diagram



Fields

Field	Function
0 DEC_ERR	DEC_ERR Decode Error Status. This bit reports detection of decode errors which have been routed to the AEM block. This status bit is write one to clear. 0b - No Decode Errors has been detected by the AEM block. 1b - Decode error has been detected by the AEM block.
1-7 —	- Reserved
8-15 AUXn_ERR	AUXn_ERR Error Status for AUXn ports (1 bit per port). This bit reports detection of decode errors which have been routed to the AEM block. This status bit is write one to clear. 00000000b - No Decode Errors have been reported on this AUX port. 00000001b - Decode error has been reported on this AUX port.
16-31 —	- Reserved

11.6.4 Interrupt Enable Register (INT_EN)

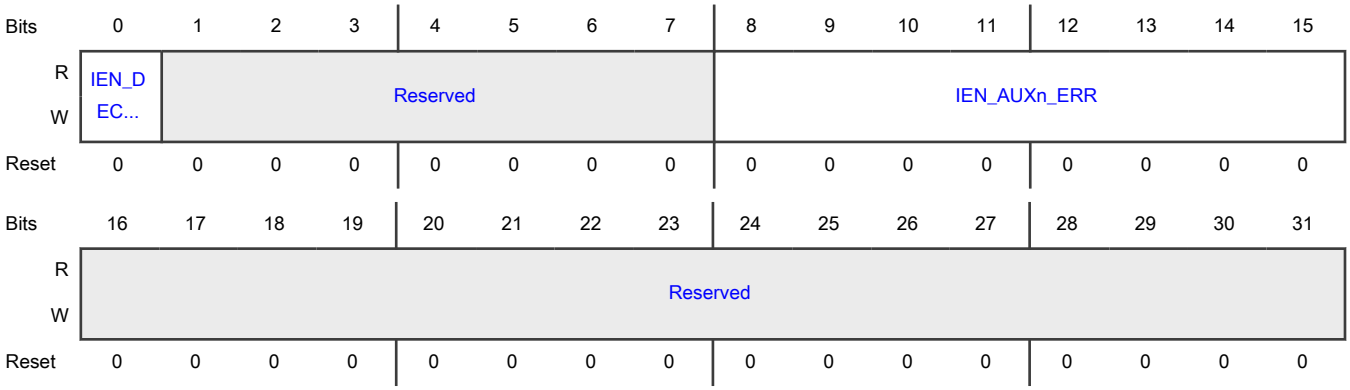
Offset

Register	Offset
INT_EN	8h

Function

INT_EN provides interrupt enables for individual error sources.

Diagram



Fields

Field	Function
0 IEN_DEC_ERR	IEN_DEC_ERR Interrupt Enable for Decode Errors. This bit enables interrupts on detection of a decode error. 0b - Interrrupt on Decode Errors is disabled. 1b - Interrupt is delivered on detection of a decode error.
1-7 —	- Reserved
8-15 IEN_AUXn_ERR	IEN_AUXn_ERR Interrupt Enable for Errors received on AUXn ports (1 bit per port). This bit enables interrupts on detection of a decode error. 00000000b - Interrrupt on Error received from this AUX port is disabled. 00000001b - Interrupt is delivered on detection of an error received on this AUX port.
16-31 —	- Reserved

11.6.5 Error Address Capture (CAPTURE_ADDRESS)

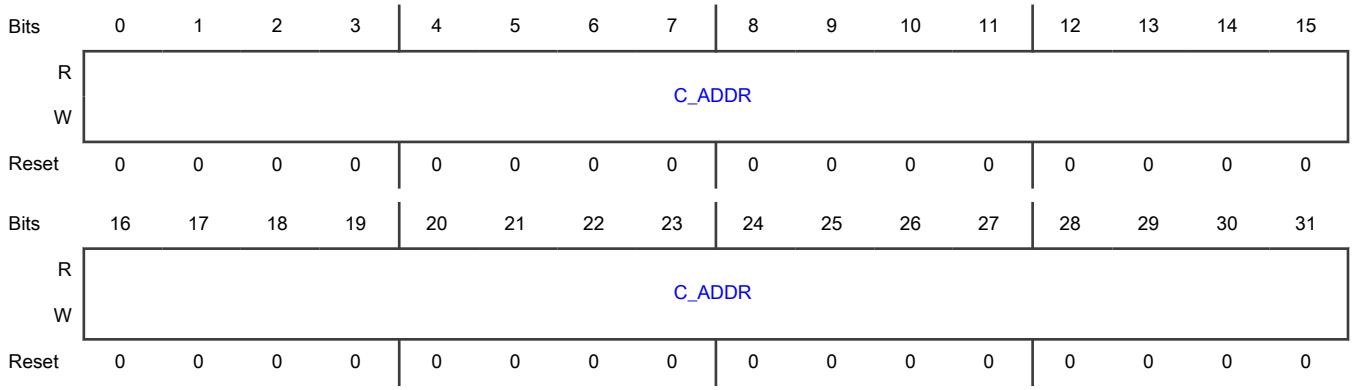
Offset

Register	Offset
CAPTURE_ADDRESS	100h

Function

CAPTURE_ADDRESS captures the lowest 32-bits of the address associated with the transaction causing the error.

Diagram



Fields

Field	Function
0-31 C_ADDR	C_ADDR Capture Address. Captures the least significant 32-bits of the 49-bit address.

11.6.6 Error Extended Address Capture (CAPTURE_EXT_ADDRESS)

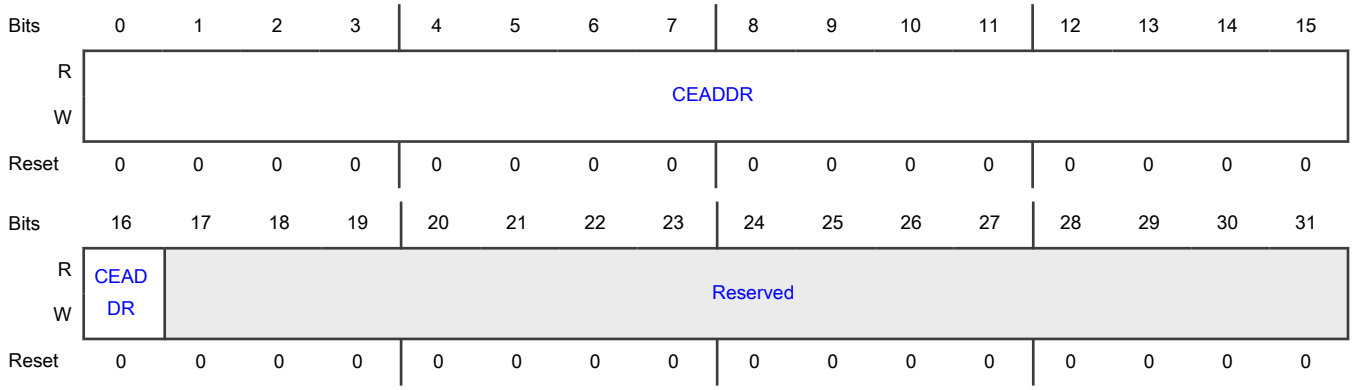
Offset

Register	Offset
CAPTURE_EXT_ADDRESS	104h

Function

CAPTURE_EXT_ADDRESS captures the most significant address bits associated with the transaction causing the error.

Diagram



Fields

Field	Function
0-16 CEADDR	CEADDR Capture Extended Address. Captures the most significant 17-bits of the 49-bit address.
17-31 —	- Reserved

11.6.7 Error Attributes Capture Register 1 (CAPTURE_ATTRIBUTES1)

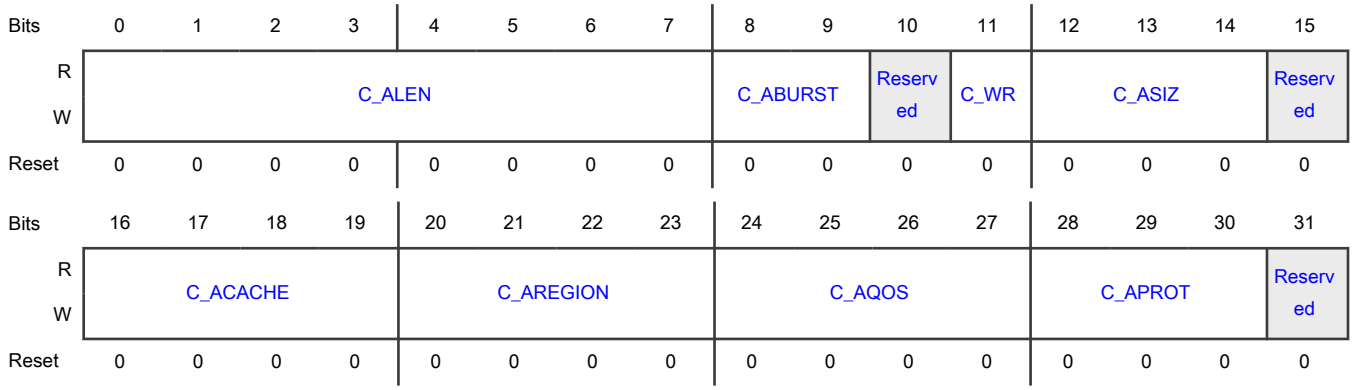
Offset

Register	Offset
CAPTURE_ATTRIBUTES1	110h

Function

CAPTURE_ATTRIBUTES1 captures attributes associated with the transaction causing the error.

Diagram



Fields

Field	Function
0-7 C_ALEN	C_ALEN Captures AxLEN signals.
8-9 C_ABURST	C_ABURST Captures AxABURST signals.
10 —	- Reserved
11 C_WR	C_WR Captures direction of the transfer 0b - Read Transaction 1b - Write Transaction
12-14 C_ASIZ	C_ASIZ Captures AxSIZ signals.
15 —	- Reserved
16-19 C_ACACHE	C_ACACHE Captures AxCACHE signals.
20-23 C_AREGION	C_AREGION Captures AxREGION signals.
24-27 C_AQOS	C_AQOS Captures AxQOS signals.
28-30	C_APROT

Table continues on the next page...

Table continued from the previous page...

Field	Function
C_APROT	Captures AxPROT signals. • Note that bit 28 indicates instruction (1'b1) vs data (1'b0) accesses
31 —	- Reserved

11.6.8 Error Attributes Capture Register 2 (CAPTURE_ATTRIBUTES2)

Offset

Register	Offset
CAPTURE_ATTRIBUTES2	114h

Function

CAPTURE_ATTRIBUTES2 captures attributes associated with the transaction causing the error.

Diagram

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

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

Fields

Field	Function
0-31 C_AUSER	C_AUSER Captures AxUSER signals.

11.6.9 Error Attributes Capture Register 3 (CAPTURE_ATTRIBUTES3)

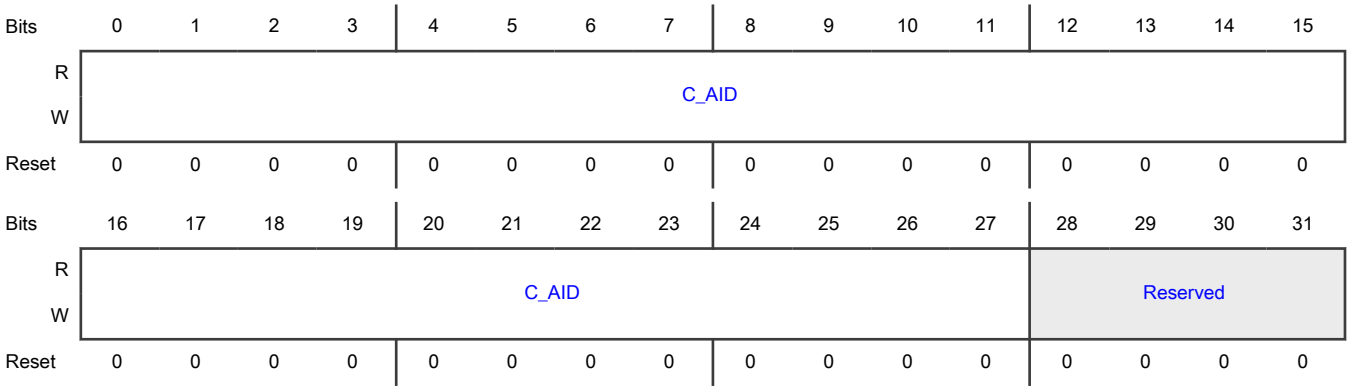
Offset

Register	Offset
CAPTURE_ATTRIBUTES3	118h

Function

CAPTURE_ATTRIBUTES3 captures attributes associated with the transaction causing the error.

Diagram



Fields

Field	Function
0-27 C_AID	C_AID Captures AxID signals.
28-31 —	- Reserved

Chapter 12
Enhanced Direct Memory Access (eDMA)

12.1 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

12.1.1 eDMA system block diagram

Figure 16 illustrates the components of the eDMA system, including the eDMA module ("engine").

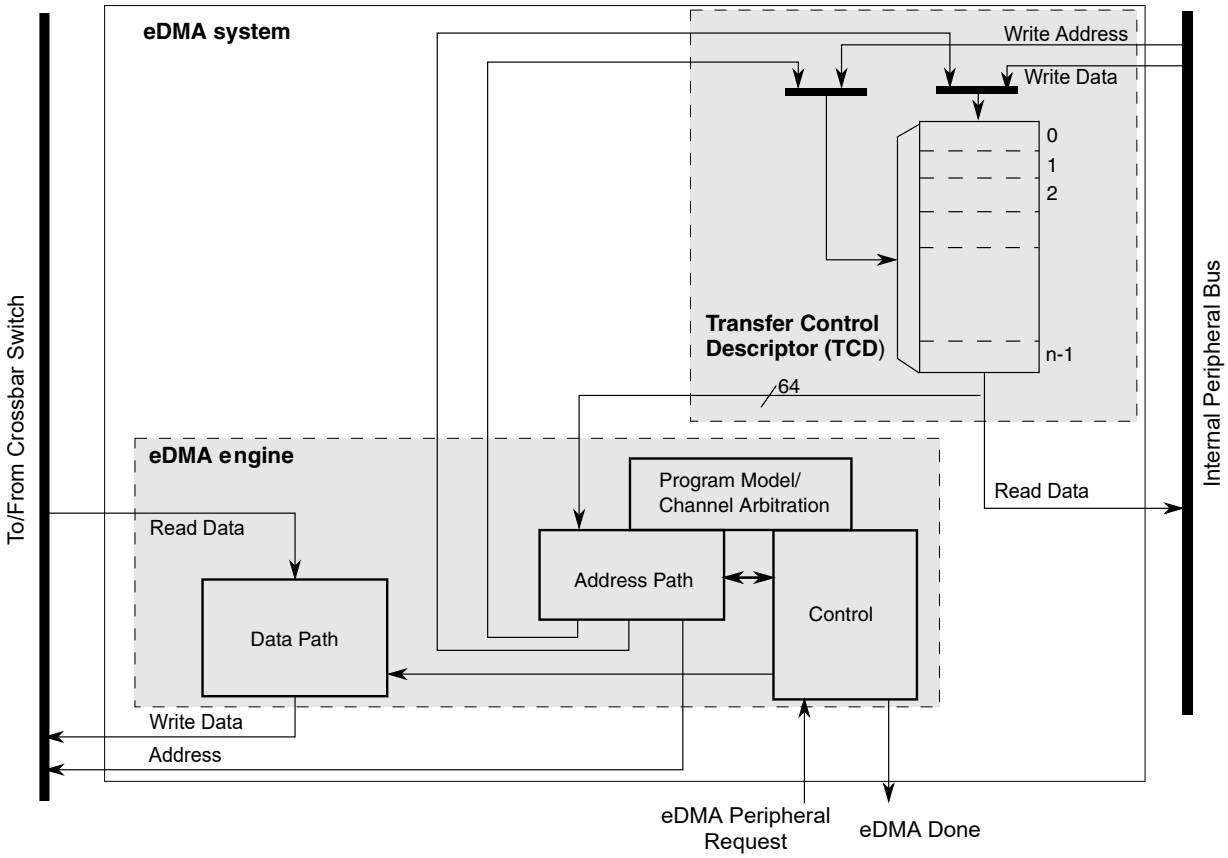


Figure 16. eDMA system block diagram

12.1.2 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 32. eDMA engine submodules

Submodule	Function
Address path	The address path block: • Provides registered versions of two channel Transfer Control Descriptors (TCDs)—channel x (normal start) and channel y (preemption start) • Manages all master bus-address calculations. All 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 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 33. 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.

12.1.3 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.

12.2 Modes of operation

The eDMA operates in the following modes:

Table 34. 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:

Table continues on the next page...

Table 34. Modes of operation (continued)

Mode	Description
	• 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.

12.3 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

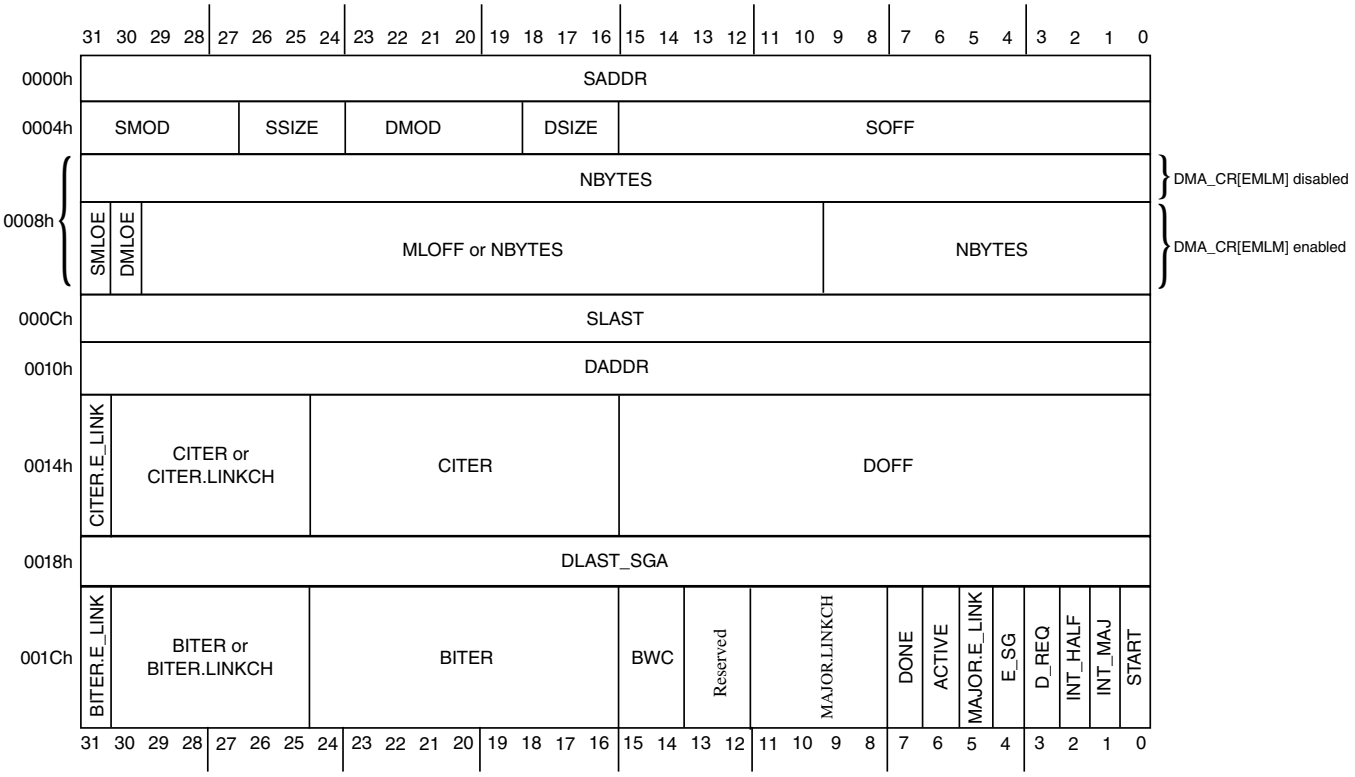
12.3.1 TCD memory

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.

12.3.2 TCD initialization

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

12.3.3 TCD structure



12.3.4 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.

12.3.5 DMA register descriptions

12.3.5.1 DMA Memory map

DMA base address: 22C_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control Register (CR)	32	RW	See description.
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
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

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
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	See description.
1004h - 11E4h	TCD Signed Source Address Offset (TCD0_SOFF - TCD15_SOFF)	16	RW	See description.
1006h - 11E6h	TCD Transfer Attributes (TCD0_ATTR - TCD15_ATTR)	16	RW	See description.
1008h - 11E8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO - TCD15_NBYTES_MLNO)	32	RW	See description.
1008h - 11E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO - TCD15_NBYTES_MLOFFNO)	32	RW	See description.
1008h - 11E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES - TCD15_NBYTES_MLOFFYES)	32	RW	See description.
100Ch - 11ECh	TCD Last Source Address Adjustment (TCD0_SLAST - TCD15_SLAST)	32	RW	See description.
1010h - 11F0h	TCD Destination Address (TCD0_DADDR - TCD15_DADDR)	32	RW	See description.
1014h - 11F4h	TCD Signed Destination Address Offset (TCD0_DOFF - TCD15_DOFF)	16	RW	See description.
1016h - 11F6h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD15_CITER_ELINKNO)	16	RW	See description.
1016h - 11F6h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD15_CITER_ELINKYES)	16	RW	See description.
1018h - 11F8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA - TCD15_DLASTSGA)	32	RW	See description.
101Ch - 11FCh	TCD Control and Status (TCD0_CSR - TCD15_CSR)	16	RW	See description.

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
101Eh - 11FEh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD15_BITER_ELINKNO)	16	RW	See description.
101Eh - 11FEh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD15_BITER_ELINKYES)	16	RW	See description.

12.3.5.2 Control Register (CR)

Offset

Register	Offset
CR	0h

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

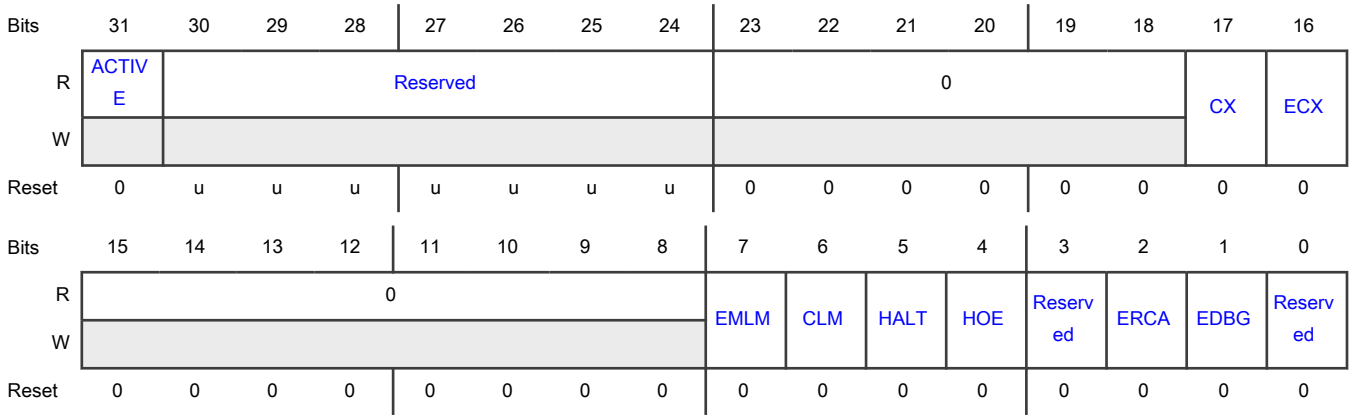
For correct operation, writes to the CR register must be performed only when the DMA channels are inactive; that is, when TCDn_CSR[ACTIVE] bits are cleared.

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.

Diagram



Fields

Field	Function
31 ACTIVE	DMA Active Status 0b - eDMA is idle. 1b - eDMA is executing a channel.
30-24 —	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	Enable Minor Loop Mapping

Table continues on the next page...

Table continued from the previous page...

Field	Function
EMLM	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.
3 —	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 When this field is 0 and the chip enters Debug mode, eDMA continues operation. When this field is 1, entry of the chip into Debug mode causes the eDMA to stall the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the chip exits Debug mode or you write 0 to this field. 0b - When the chip is in Debug mode, the eDMA continues to operate.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - When the chip is in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete.
0 —	Reserved

12.3.5.3 Error Status Register (ES)

Offset

Register	Offset
ES	4h

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.

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

Fields

Field	Function
31	VLD

Table continues on the next page...

Table continued from the previous page...

Field	Function
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 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

Table continues on the next page...

Table continued from the previous page...

Field	Function
	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

12.3.5.4 Enable Request Register (ERQ)

Offset

Register	Offset
ERQ	Ch

Function

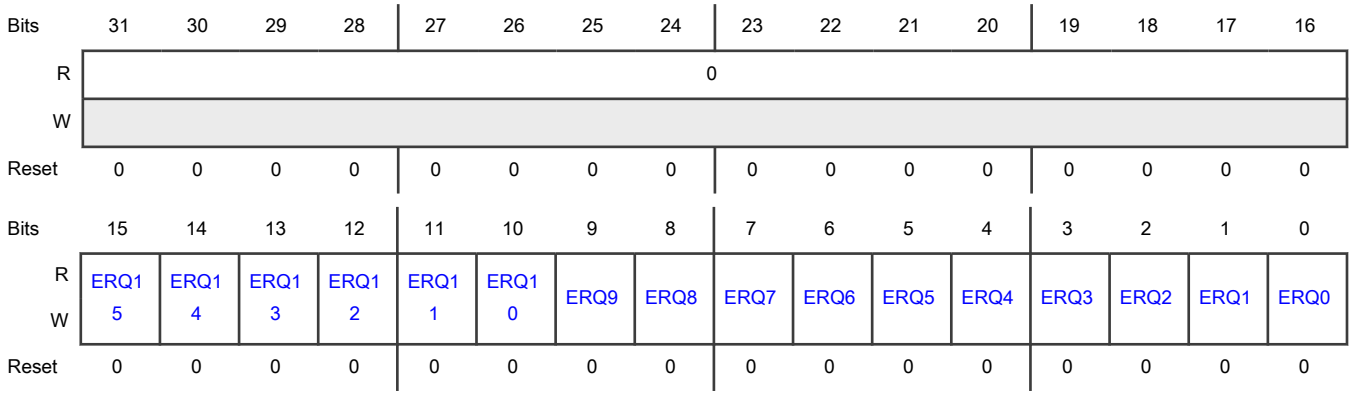
The ERQ register provides a bit map for the 16 channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable 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.

Diagram



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

Table continues on the next page...

Table continued from the previous page...

Field	Function
	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
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

12.3.5.5 Enable Error Interrupt Register (EEI)

Offset

Register	Offset
EEI	14h

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.

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 W	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI9	EEI8	EEI7	EEI6	EEI5	EEI4	EEI3	EEI2	EEI1	EEI0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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	Enable Error Interrupt 13

Table continues on the next page...

Table continued from the previous page...

Field	Function
EEI13	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 EEI7	Enable Error Interrupt 7 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

Table continues on the next page...

Table continued from the previous page...

Field	Function
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

12.3.5.6 Clear Enable Error Interrupt Register (CEEI)

Offset

Register	Offset
CEEI	18h

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.

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

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

12.3.5.7 Set Enable Error Interrupt Register (SEEI)

Offset

Register	Offset
SEEI	19h

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.

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

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

12.3.5.8 Clear Enable Request Register (CERQ)

Offset

Register	Offset
CERQ	1Ah

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.

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

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.

12.3.5.9 Set Enable Request Register (SERQ)

Offset

Register	Offset
SERQ	1Bh

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.

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

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

12.3.5.10 Clear DONE Status Bit Register (CDNE)

Offset

Register	Offset
CDNE	1Ch

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.

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

Fields

Field	Function
7 NOP	No Op enable 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]

12.3.5.11 Set START Bit Register (SSRT)

Offset

Register	Offset
SSRT	1Dh

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.

Diagram

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

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]

12.3.5.12 Clear Error Register (CERR)

Offset

Register	Offset
CERR	1Eh

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.

Diagram

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

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

12.3.5.13 Clear Interrupt Request Register (CINT)

Offset

Register	Offset
CINT	1Fh

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.

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

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

12.3.5.14 Interrupt Request Register (INT)

Offset

Register	Offset
INT	24h

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.

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

Fields

Field	Function
31-16 —	Reserved
15 INT15	Interrupt Request 15 0b - The interrupt request for channel 15 is cleared 1b - The interrupt request for channel 15 is active
14 INT14	Interrupt Request 14 0b - The interrupt request for channel 14 is cleared 1b - The interrupt request for channel 14 is active
13 INT13	Interrupt Request 13 0b - The interrupt request for channel 13 is cleared 1b - The interrupt request for channel 13 is active
12 INT12	Interrupt Request 12 0b - The interrupt request for channel 12 is cleared 1b - The interrupt request for channel 12 is active
11 INT11	Interrupt Request 11 0b - The interrupt request for channel 11 is cleared 1b - The interrupt request for channel 11 is active
10 INT10	Interrupt Request 10 0b - The interrupt request for channel 10 is cleared 1b - The interrupt request for channel 10 is active
9 INT9	Interrupt Request 9

Table continues on the next page...

Table continued from the previous page...

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

12.3.5.15 Error Register (ERR)

Offset

Register	Offset
ERR	2Ch

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.

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	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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	Error In Channel 14

Table continues on the next page...

Table continued from the previous page...

Field	Function
ERR14	0b - An error in this channel has not occurred 1b - An error in this channel has occurred
13 ERR13	Error In Channel 13 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

Table continues on the next page...

Table continued from the previous page...

Field	Function
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

12.3.5.16 Hardware Request Status Register (HRS)

Offset

Register	Offset
HRS	34h

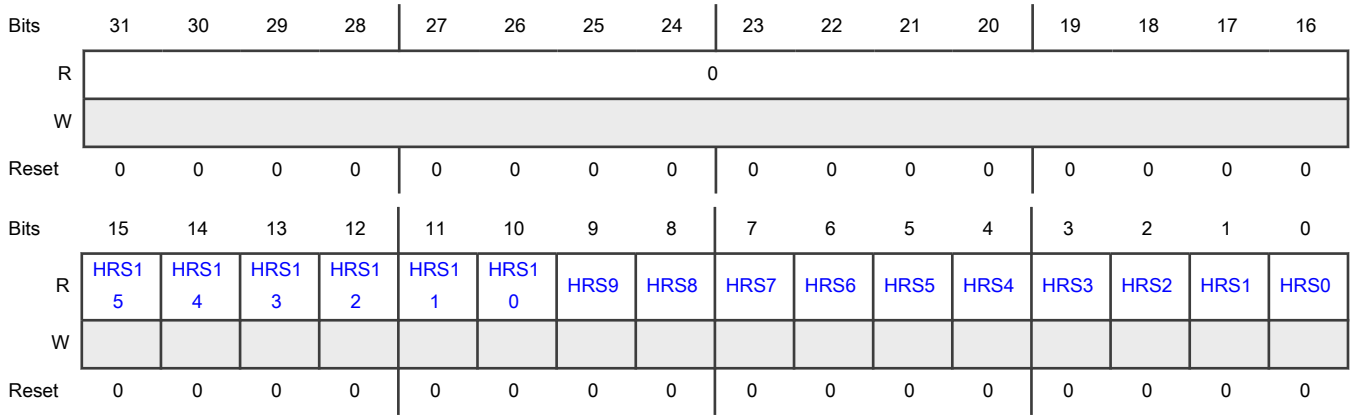
Function

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.

Diagram



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. 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.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	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
6 HRS6	Hardware Request Status Channel 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
	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

Table continues on the next page...

Table continued from the previous page...

Field	Function
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

12.3.5.17 Channel Priority Register (DCHPRI0 - DCHPRI15)

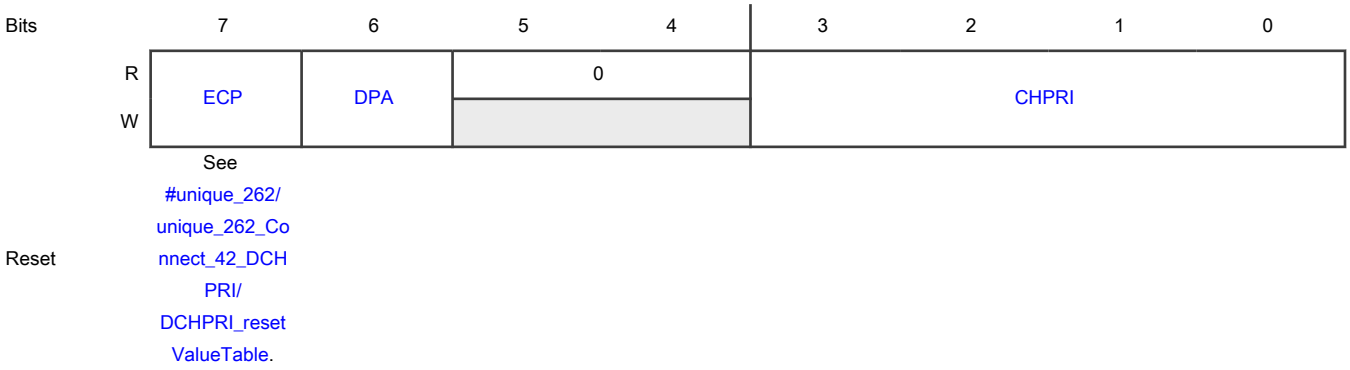
Offset

Register	Offset
DCHPRI3	100h
DCHPRI2	101h
DCHPRI1	102h
DCHPRI0	103h
DCHPRI7	104h
DCHPRI6	105h
DCHPRI5	106h
DCHPRI4	107h
DCHPRI11	108h
DCHPRI10	109h
DCHPRI9	10Ah
DCHPRI8	10Bh
DCHPRI15	10Ch
DCHPRI14	10Dh
DCHPRI13	10Eh
DCHPRI12	10Fh

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.

Diagram



Register reset values

Register	Reset value
DCHPRI0	00h
DCHPRI1	01h
DCHPRI2	02h
DCHPRI3	03h
DCHPRI4	04h
DCHPRI5	05h
DCHPRI6	06h
DCHPRI7	07h
DCHPRI8	08h
DCHPRI9	09h
DCHPRI10	0Ah
DCHPRI11	0Bh
DCHPRI12	0Ch
DCHPRI13	0Dh
DCHPRI14	0Eh
DCHPRI15	0Fh

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.

Table continues on the next page...

Table continued from the previous page...

Field	Function
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

12.3.5.18 TCD Source Address (TCD0_SADDR - TCD15_SADDR)

Offset

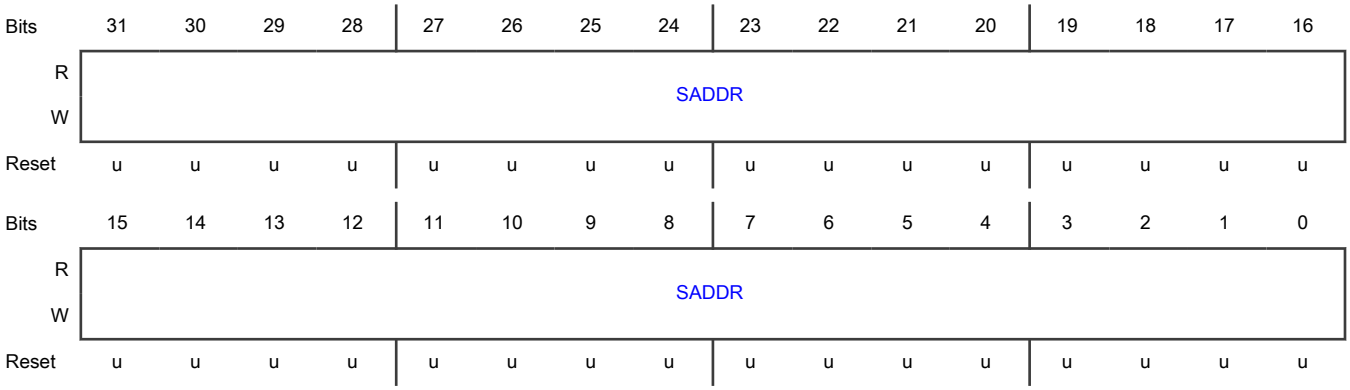
For n = 0 to 15:

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

Function

This register contains the source address of the transfer.

Diagram



Fields

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

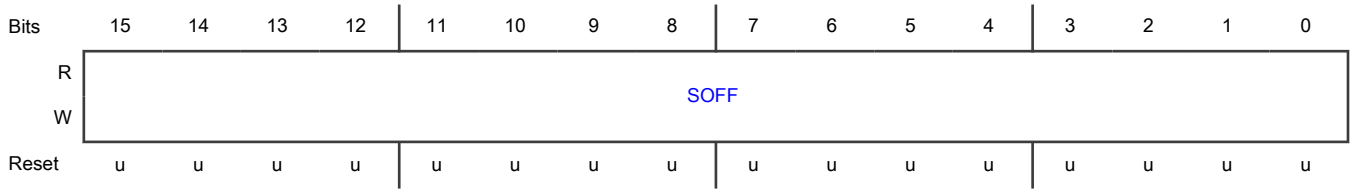
12.3.5.19 TCD Signed Source Address Offset (TCD0_SOFF - TCD15_SOFF)

Offset

For n = 0 to 15:

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

Diagram



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.

12.3.5.20 TCD Transfer Attributes (TCD0_ATTR - TCD15_ATTR)

Offset

For n = 0 to 15:

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

Diagram

Fields

Field	Function
15-11	Source Address Modulo

Table continues on the next page...

Table continued from the previous page...

Field	Function
SMOD	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. **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

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

Offset

For n = 0 to 15:

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

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.

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

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.

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

Offset

For n = 0 to 15:

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

Function

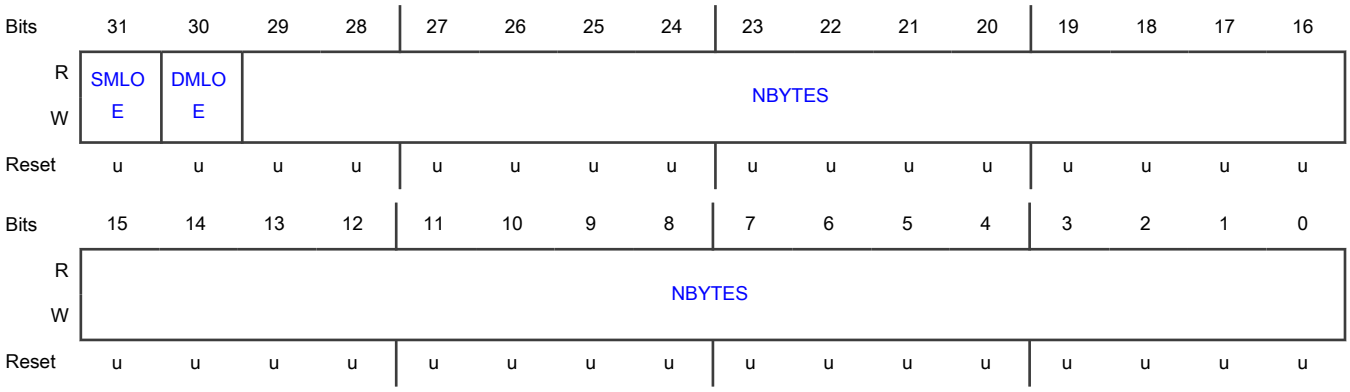
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

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.

Diagram



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. 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.

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

Offset

For n = 0 to 15:

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

Function

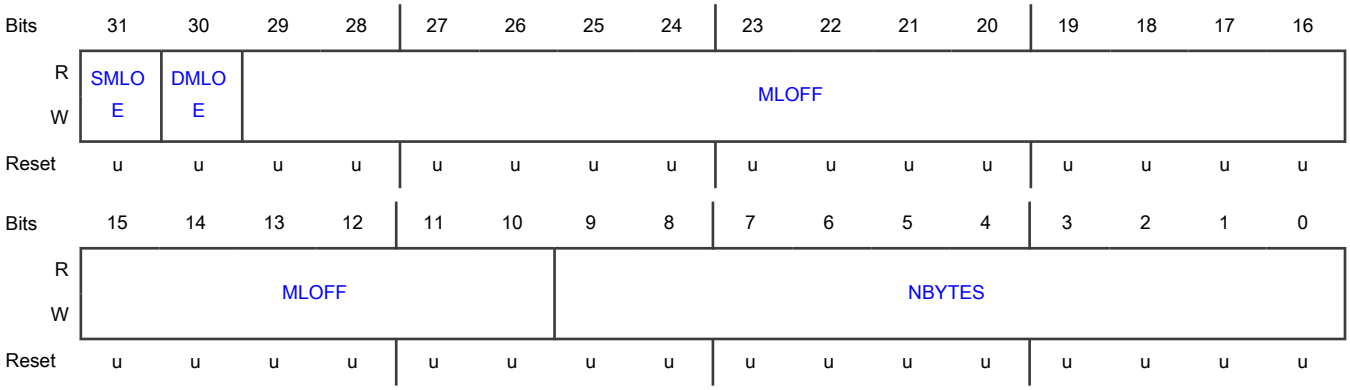
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

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.

Diagram



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.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	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.

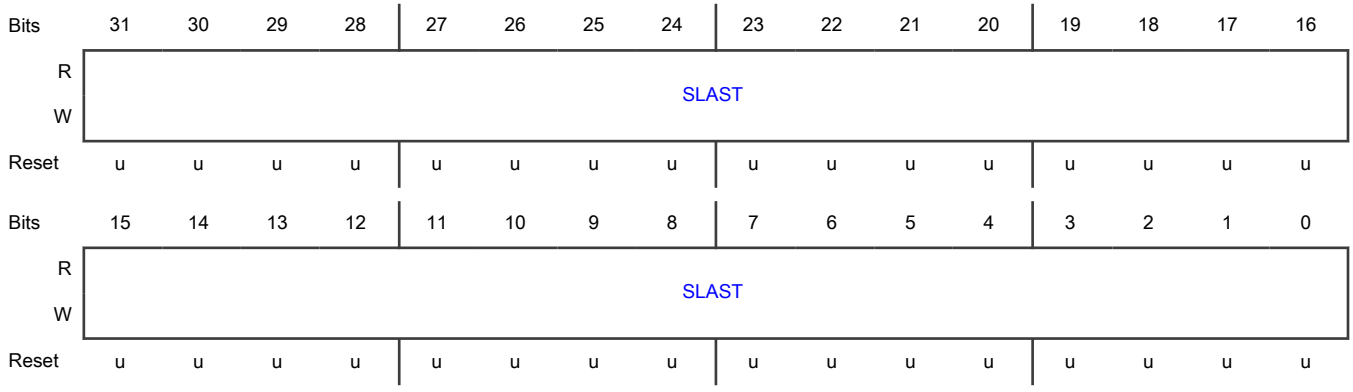
12.3.5.24 TCD Last Source Address Adjustment (TCD0_SLAST - TCD15_SLAST)

Offset

For n = 0 to 15:

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

Diagram



Fields

Field	Function
31-0	Last Source Address Adjustment

Table continues on the next page...

Field	Function
SLAST	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.

12.3.5.25 TCD Destination Address (TCD0_DADDR - TCD15_DADDR)

Offset

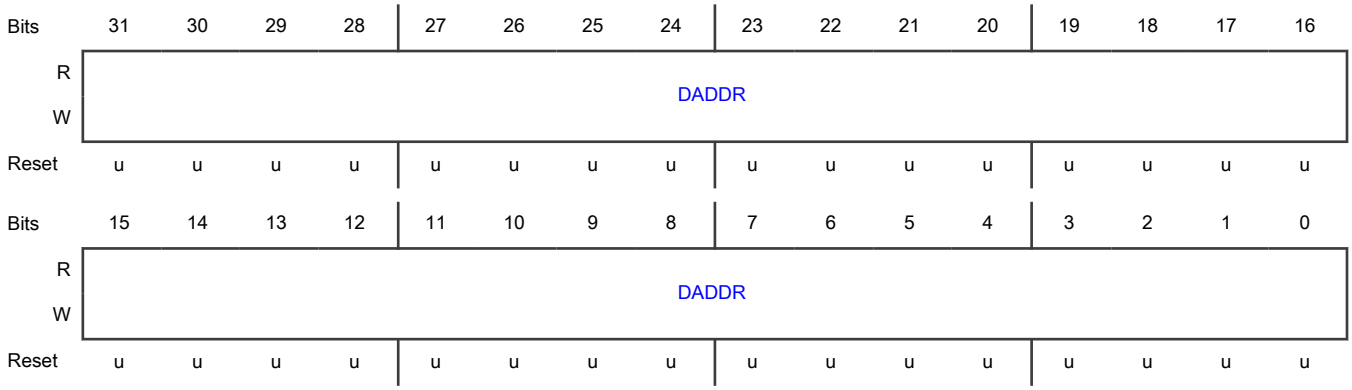
For n = 0 to 15:

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

Function

This register contains the destination address of the transfer.

Diagram



Fields

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

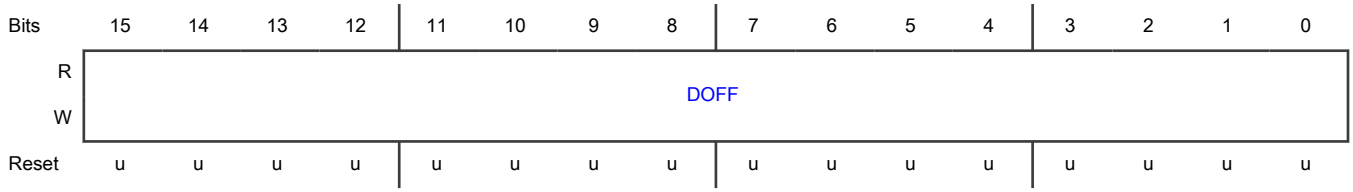
12.3.5.26 TCD Signed Destination Address Offset (TCD0_DOFF - TCD15_DOFF)

Offset

For n = 0 to 15:

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

Diagram



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.

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

Offset

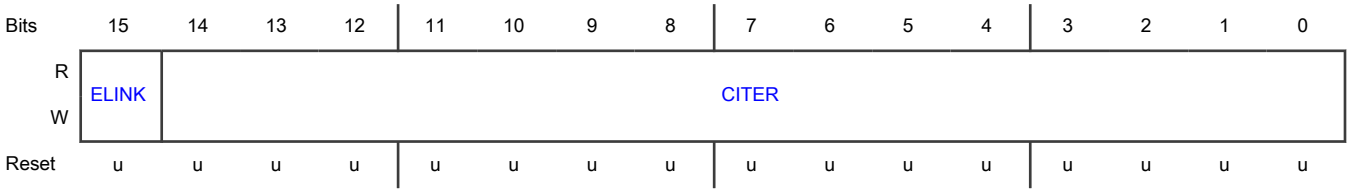
For n = 0 to 15:

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

Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as #unique_262/unique_262_Connect_42_TCD0_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.

Diagram



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
14-0 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.

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

Offset

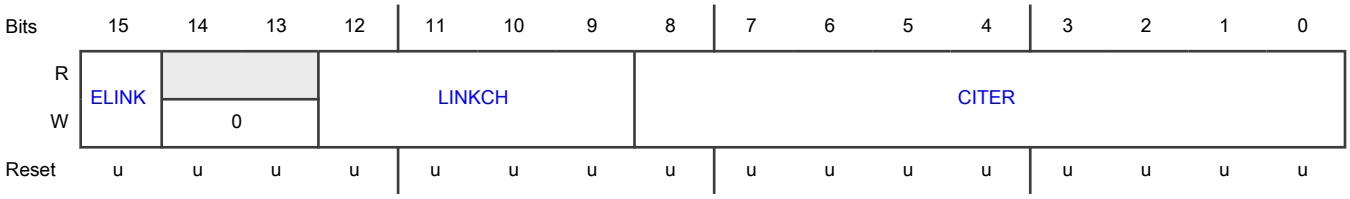
For n = 0 to 15:

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

Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as #unique_262/unique_262_Connect_42_TCD0_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.

Diagram



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.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

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

Offset

For n = 0 to 15:

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

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

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:

Table continues on the next page...

Field	Function
	• 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.

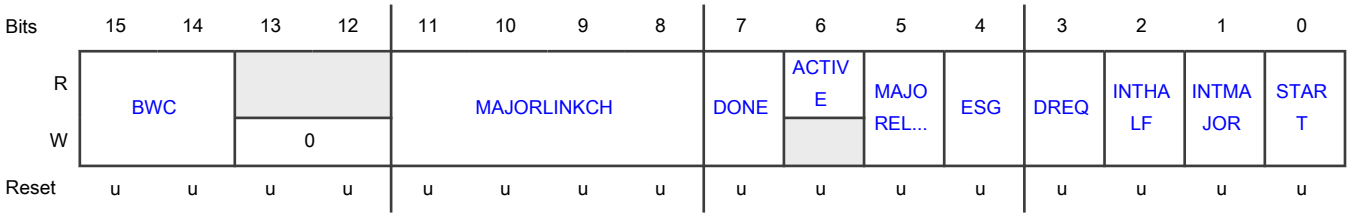
12.3.5.30 TCD Control and Status (TCD0_CSR - TCD15_CSR)

Offset

For n = 0 to 15:

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

Diagram



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

Table continues on the next page...

Table continued from the previous page...

Field	Function
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 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	Disable Request

Table continues on the next page...

Table continued from the previous page...

Field	Function
DREQ	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.

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

Offset

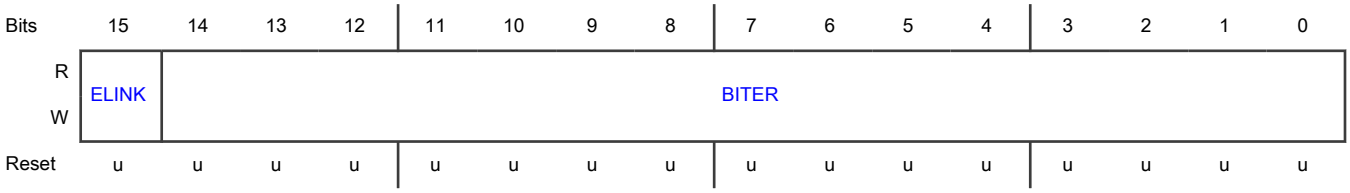
For n = 0 to 15:

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

Function

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

Diagram



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. 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 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. 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.

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

Offset

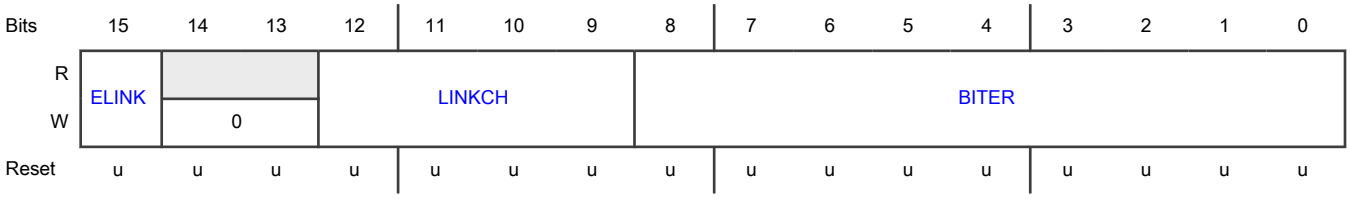
For n = 0 to 15:

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

Function

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

Diagram



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 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. 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. 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.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE When the software loads the TCD, this field must be set equal to the corresponding CITER field. 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.

12.4 Functional description

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

12.4.1 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:

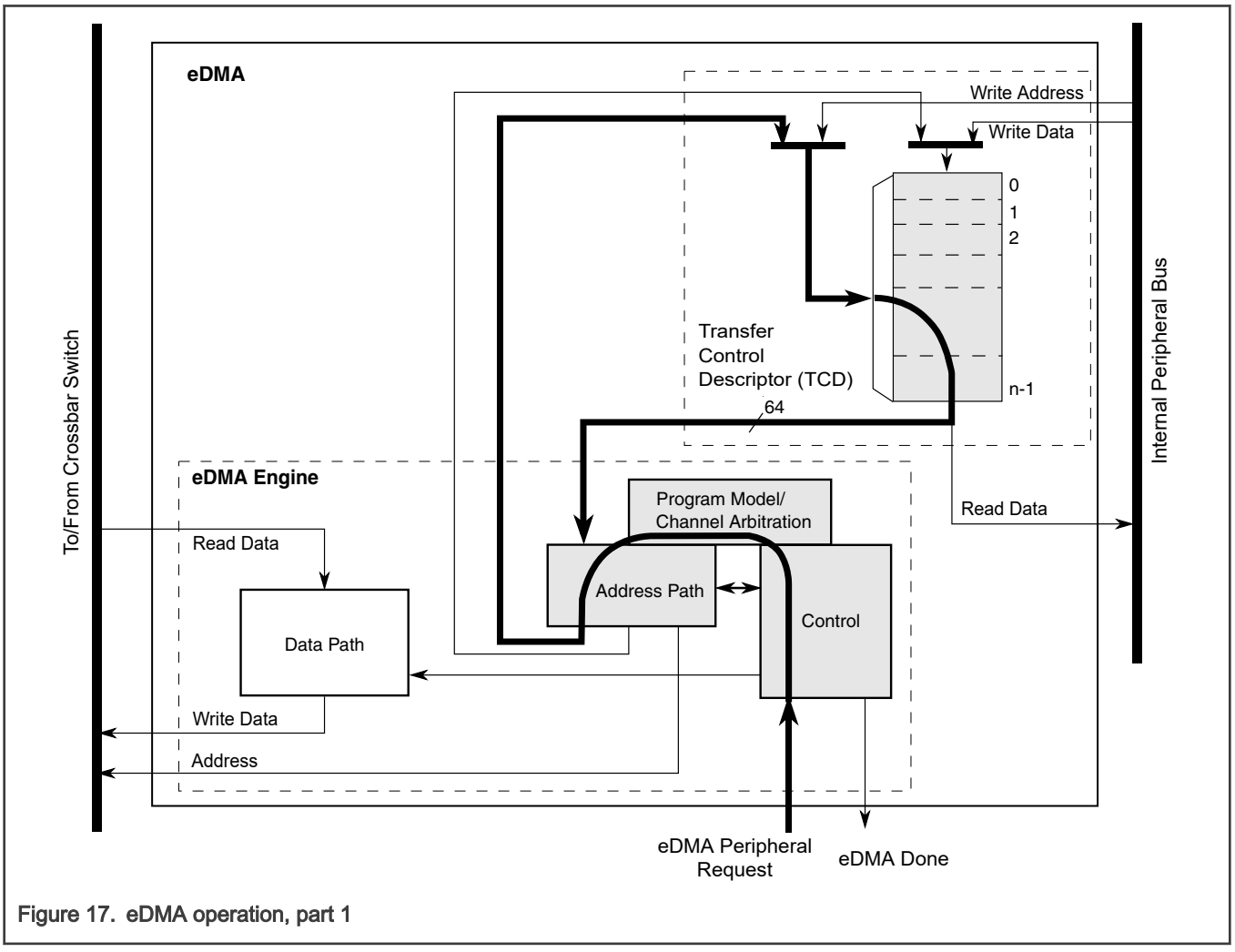


Figure 17. eDMA operation, part 1

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's internal register file. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the internal register file.

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

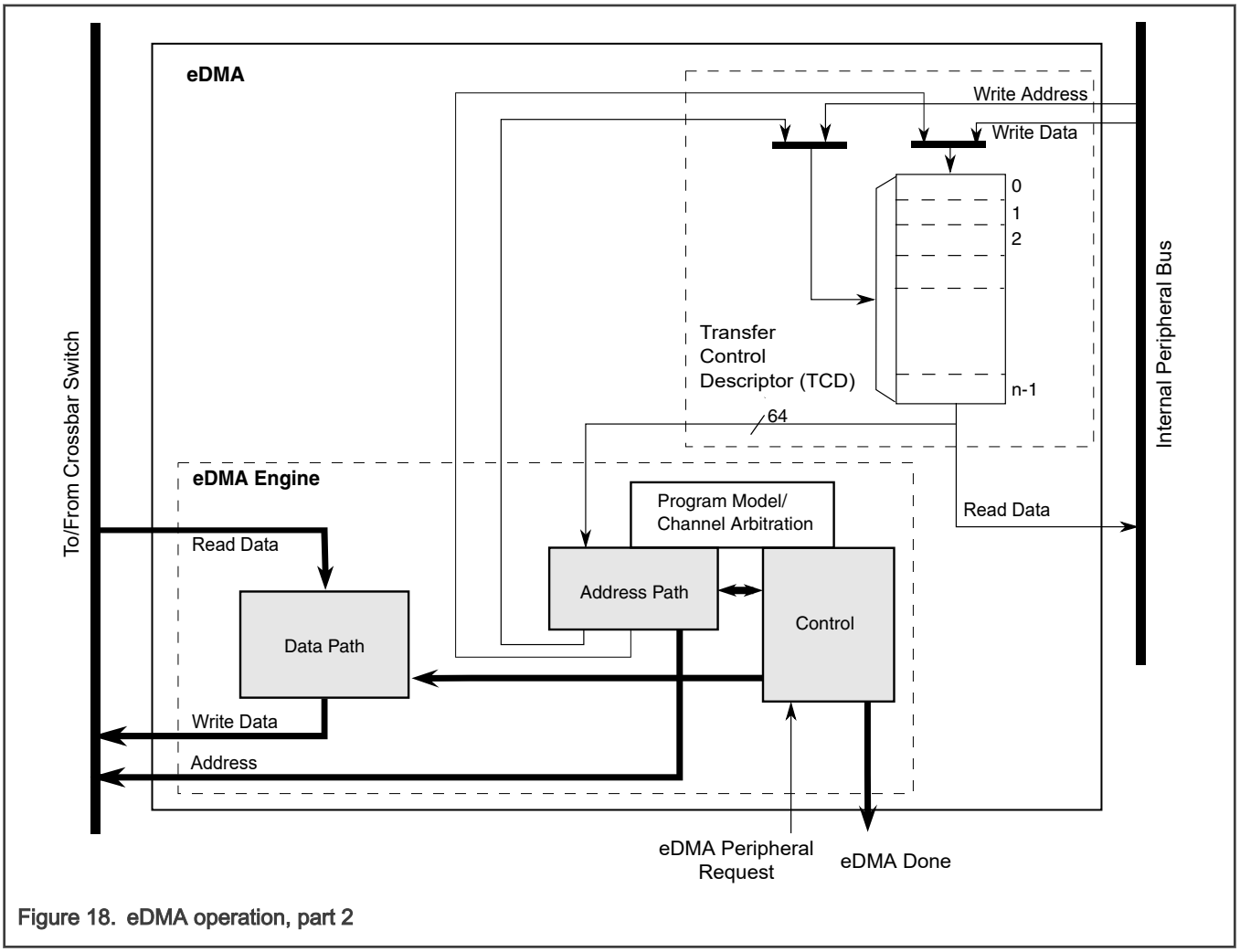


Figure 18. eDMA operation, part 2

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.

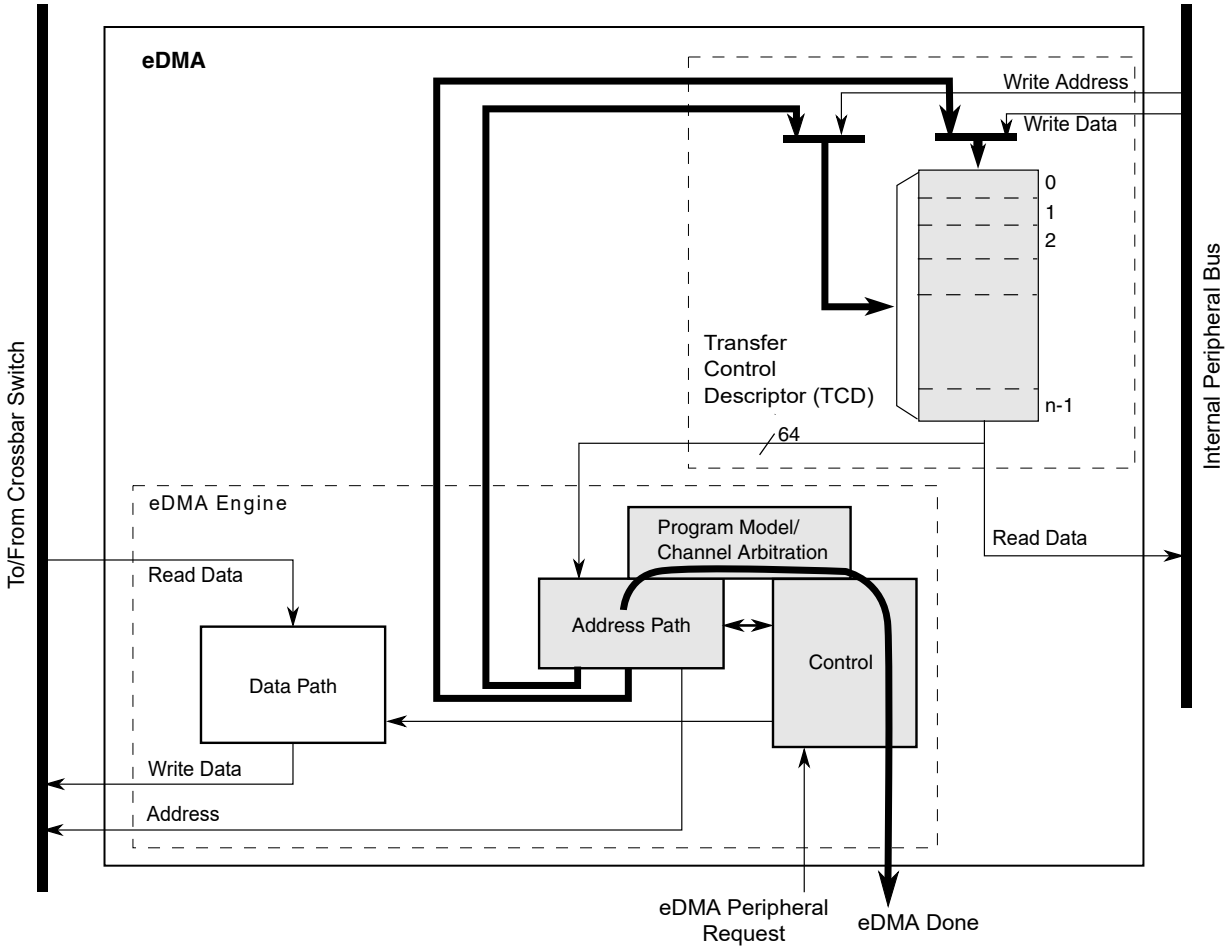


Figure 19. eDMA operation, part 3

12.4.2 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.

12.4.3 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.

12.5 Initialization/application information

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

12.5.1 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 system bus, unless a configuration error is detected. Transfers from the source, as defined by TCD*n*_SADDR, to the destination, as defined by TCD*n*_DADDR, continue until the number of bytes specified by TCD*n*_NBYTES 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 35. 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)

Table continues on the next page...

Table 35. TCD Control and Status fields (continued)

TCD*n*_CSR field name	Description
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).

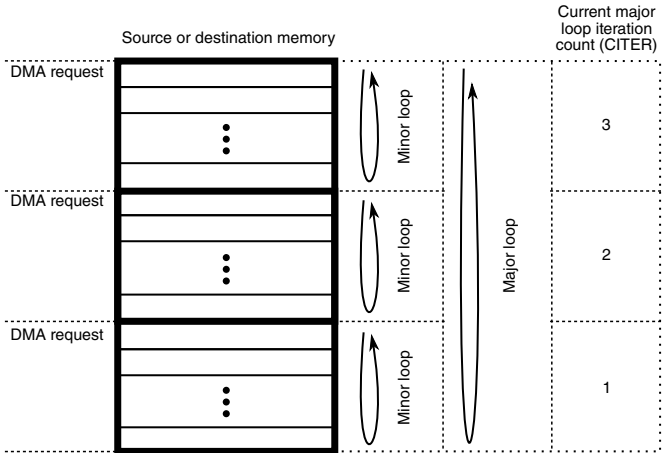


Figure 20. Example of multiple loop iterations

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

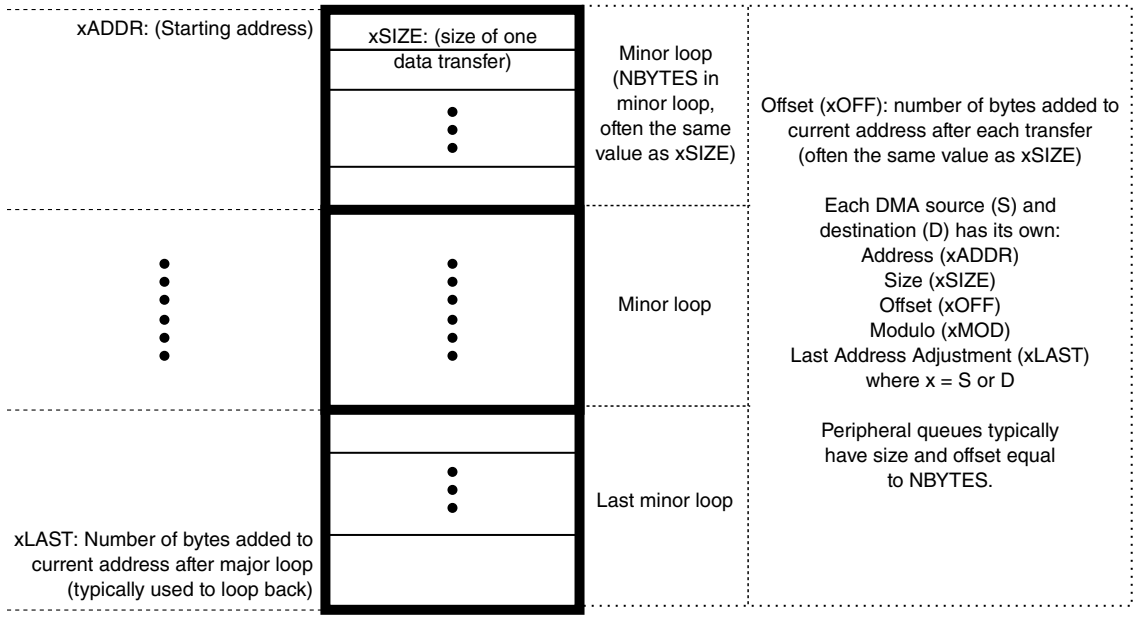


Figure 21. Memory array terms

12.5.2 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.

12.5.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

12.5.3.1 Fixed channel arbitration

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

12.5.3.2 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.

12.5.4 Performing DMA transfers

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

12.5.4.1 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.

12.5.4.2 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.

12.5.4.3 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 36. Modulo example

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

12.5.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

12.5.5.1 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.

12.5.5.2 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.

12.5.5.3 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.

12.5.6 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] = 0x3
```

executes as:

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

When minor loop linking is enabled (TCD*n*_CITER[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 37. 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

12.5.7 Dynamic programming

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

12.5.7.1 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.

12.5.7.2 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.

12.5.7.3 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.

12.5.7.3.1 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).

12.5.7.3.2 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).

12.5.8 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.

12.5.8.1 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.

12.5.8.2 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 13
Forward Error Correction Unit (FECU)

13.1 FECU overview

The FECU module handles forward error correction (FEC) for proprietary protocols similar to WiFi. It performs FEC for the data fields. FECU is comprised of multiple sub-modules. Each sub-module performs an operation and hands the data off to the next sub-module in the chain. Once the chain completes, the entire operation is finished, and an interrupt is generated.

Input data can be loaded into DMEM using the AXI slave, or VSPA's DMA can fetch the input data and load it into DMEM. FECU's output data is written to DMEM, and VSPA DMA can transfer it to any AXI address.

FECU is configured by FECU IP registers. See Register descriptions for detailed description of these registers.

13.2 FECU features

- Supports up to 256QAM
- Can support multiple streams through VSPA firmware.
- Runs at 614.4 MHz
- Context save / restore for multi-user support
- Encoding operations
  - Scrambling
  - Convolutional encoding
  - Interleaving
  - LDPC tone mapping
  - LDPC Encoding
- Decoding operations
  - De-interleaving (reuse interleaver)
  - LDPC tone de-mapping
  - Viterbi Decoding
  - De-scrambling (reuse scrambler)
  - LDPC Decoding

13.3 FECU block diagram

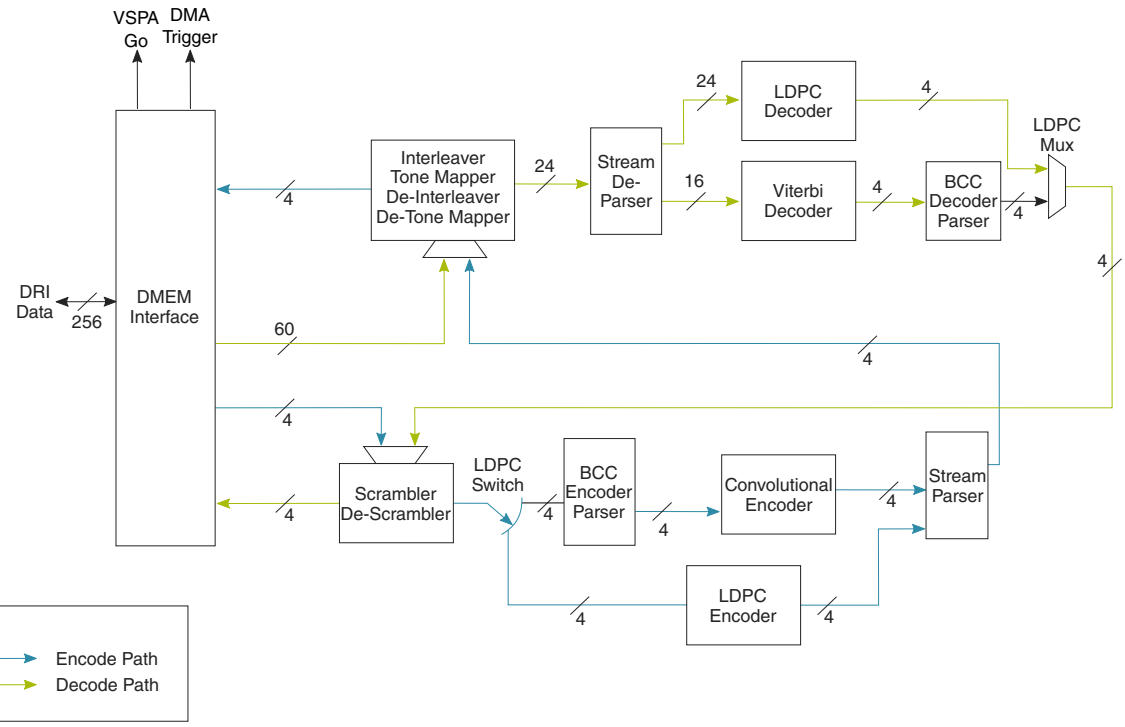


Figure 22. FECU block diagram

13.4 FECU clock generation

- FECU contains internal clock generation used to create the gated clocks for the FECU submodules.
- The FECU internal data paths run on dedicated clocks at the VSPA clock rate that are automatically enabled/disabled as needed during operation to minimize current consumption.
- Clock enable override register bits are provided for all internally-generated clocks.

13.5 FECU low power modes

- Sleep Mode
  - All clocks disabled
  - No FECU operation
  - All FECU source clocks disabled by SoC level clock control
- Idle Mode
  - Decoders not operating
  - FECU IP clock is enabled by SoC level clock control
  - FECU internal decoder clocks are disabled by the hardware automatically
- Decoder Active Mode
  - The decoder is operating.
  - IP and any required internal clocks enabled by SoC level clock control

- Internal decoder clocks are enabled by the hardware automatically.

13.6 FECU reset

- The FECU module is reset using the hard asynchronous reset for the chip.
- In addition, a software reset is provided. The software reset is enabled by writing to a register bit in the FECU_CONFIG register. It remains set until cleared by software. This reset is not required for normal operation.

13.7 FECU interrupts and VSPA go

The FECU module has one interrupt output. When FECU completes a command and the irqen_done bit in FECU CONTROL register is set it will set the irq_pend_fecu_done bit (bit 6) in VSPA STATUS register. When this bit is set and the corresponding irqen_fecu_done bit (bit 6) in IRQEN register is also set it will generate an interrupt request out of VSPA.

When FECU completes a command and vcpu_go_enable bit in FECU CONTROL register is set, it will set the bit 5 in FECU CONTROL register. This will cause VSPA to go.

The FECU command completion will also enable IPPU and DMA.

13.8 Viterbi Decoder overview

The Viterbi decoder performs the Viterbi algorithm to optimally decode convolutionally coded data. The Viterbi decoder takes its input data from the stream de-parser, and sends its output to the BCC decoder parser.

- Implements de-puncturing. Programmable de-puncture mask.
  - Supports 1/2, 2/3, 3/4, and 5/6 rates
- Constraint length 7
- 4 bit soft decision (LLR) receive input word size
- Processes 2 output bits per cycle
- 128 bit trace back
- Delays data by 384 bits, except for the last symbol.
- Uses 4 instances of 64 (d) x 128 (w) embedded trace back RAMs
- Processes 1 BCC blocks concurrently

13.9 Interleaver overview

The interleaver performs BCC interleaving and LDPC tone mapping. In does both encode and decode operations. During decode, it writes one constellation point per cycle (e.g. 64 QAM – 6 LLRs), and reads 4 LLRs per cycle. During encode, it writes 4 bits per cycle, and reads one constellation point per cycle.

- Performs BCC de-interleaving on 6 bit LLRs, and BCC interleaving on 1 bit data
- Performs LDPC tone de-mapping on 6 bit LLRs, and LDPC tone mapping on 1 bit data
- Configurable to handle 20,40,80, or 160 MHz interleaving
- Performs section 20.3.11.8 and 18.3.5.7 interleaving
- Processes 1 data streams at a time
- Uses 1 instance of 64 (d) x 240 (w) embedded RAMs
  - 160 MHz, 1024 QAM support

13.10 Convolutional Encoder overview

- Performs section "18.3.5.6 Convolutional encoder."

- Supports puncturing
  - Configurable puncturing patterns
  - Supports 1/2, 2/3, 3/4, and 5/6 rates
- Processes 4 input and output bits per clock cycle
- Processes 1 BCC blocks concurrently

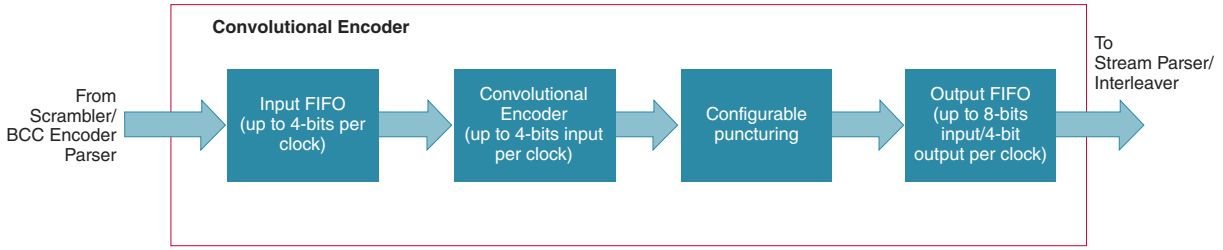


Figure 23. Convolutional Encoder

13.11 LDPC Encoder overview

The low density parity check (LDPC) encoder computes the parity check bits from the input message bits. It takes its input from the scrambler, and sends its output to the stream parser. The LDPC encoder computes the parity bits as it receives message bits. When it gets enough message bits, it waits the input and sends out the parity bits. The encoder will not send out more than coded_bits_per_symbol bits during any operation. Extra parity or repeat bits will be saved inside the encoder until the next operation.

- Performs repetition, shortening, and puncturing
- Supports 1/2, 2/3, 3/4, and 5/6 rates
- Supports 648, 1296 and 1944 block sizes
- Processes 81 bits per clock cycle

13.12 LDPC Decoder overview

The low density parity check (LDPC) decoder find the message and parity bits that satisfy the parity check matrix and are closest to the received LLRs. It takes 6 bit input LLRs from the stream de-parser, and when it gets enough LLRs, it starts the iterative decoding operation. If the decoder can't find message and parity bits that satisfy the parity check matrix, the decoder will declare a decoder failure. It if succeeds, it will send its output to the de-scrambler. The LDPC decoder only sends the message bits, and does not send parity, shortening, or repetition bits. The number of output bits will be a multiple of the LDPC block size.

- Performs de-repetition, de-shortening, and de-puncturing
- Supports 1/2, 2/3, 3/4, and 5/6 rates
- Supports 648, 1296 and 1944 block sizes
- Separate input and output buffers to allow for near 100% utilization
  - FRAM reads and de-scramble happen in parallel with decode
- 1 sub-matrix processing engines
- Processes 1 blocks in parallel

13.13 Scrambler overview

- Performs section 18.3.5.5 and 16.2.4 scrambling and de-scrambling
- Processes 4 bits per clock cycle
- LFSR State
  - Starting scrambling state specified in IP registers
  - Starting de-scrambling state
    - Section 18.3.5.5 extracted from data
    - Section 16.2.4 self synchronizes

13.14 Register descriptions

13.14.1 FECU Memory map

FECU base address: 100_0000h

Offset	Register	Width (In bits)	Access	Reset value
300h	FECU Configuration register (FECU_CONFIG)	32	RW	See description.
304h	FECU Symbol size register (FECU_SIZES)	32	RW	0000_0000h
308h	FECU Number of padding bits register (FECU_NUM_PAD)	32	RW	See description.
30Ch	FECU Binary Convolutional Code (BCC) puncture mask register (FECU_BCC_PUNC_MASK)	32	RW	0000_0003h
310h	FECU Binary Convolutional Code (BCC) configuration register (FECU_BCC_CONFIG)	32	RW	See description.
314h	FECU LDPC configuration register (FECU_LDPC_CONFIG)	32	RW	See description.
318h	FECU LDPC repeat, parity, and shortening sizes register (FECU_LDPC_SIZES)	32	RW	See description.
31Ch	FECU LDPC blocks with an extra shortening bit register (FECU_LDPC_EXTRA_SHORT)	32	RW	0000_0000h
320h	FECU LDPC blocks with an extra puncturing or repetition bit register (FECU_LDPC_EXTRA_REP)	32	RW	0000_0000h
324h	FECU Bypass register (FECU_BYPASS)	32	RW	See description.
328h	FECU Scrambler / De-scrambler configuration register (FECU_SC_CONFIG)	32	RW	See description.
32Ch	FECU DMEM Read count register (FECU_DMEM_READ_COUNT)	32	RW	See description.

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
330h	FECU DMEM Source address register (FECU_DMEM_SRC_ADR)	32	RW	See description.
334h	FECU DMEM Destination address register (FECU_DMEM_DST_ADR)	32	RW	See description.
338h	FECU DMEM 2nd address register (FECU_DMEM_2ND_ADR)	32	RW	See description.
33Ch	FECU DMEM 3rd address register (FECU_DMEM_3RD_ADR)	32	RW	See description.
340h	FECU DMEM 4th address register (FECU_DMEM_4TH_ADR)	32	RW	See description.
344h	FECU DMEM 5th address register (FECU_DMEM_5TH_ADR)	32	RW	See description.
348h	FECU DMEM 6th address register (FECU_DMEM_6TH_ADR)	32	RW	See description.
34Ch	FECU DMEM 7th address register (FECU_DMEM_7TH_ADR)	32	RW	See description.
350h	FECU DMEM 8th address register (FECU_DMEM_8TH_ADR)	32	RW	See description.
354h	FECU Save and restore configuration register (FECU_SAVE_RESTORE)	32	RW	See description.
358h	FECU Control register (FECU_CONTROL)	32	RW	See description.
364h	FECU Status register (FECU_STATUS)	32	W1C	See description.
368h	FECU DMEM Write count register (FECU_DMEM_WRITE_COUNT)	32	RO	See description.
36Ch	FECU LDPC encoder block sizes register (FECU_LDPC_ENC_BLOCK)	32	RO	See description.
370h	FECU LDPC encoder status register (FECU_LDPC_ENC_STATUS)	32	RO	See description.
374h	FECU LDPC decoder block sizes and counts register (FECU_LDPC_DEC_BLOCK)	32	RO	0000_0000h
378h	FECU LDPC decoder status register (FECU_LDPC_DEC_STATUS)	32	W1C	See description.
380h	FECU Hardware parameters / capabilities of FECU (FECU_HW_PARAMS)	32	RO	See description.

Table continues on the next page...

Table continued from the previous page...

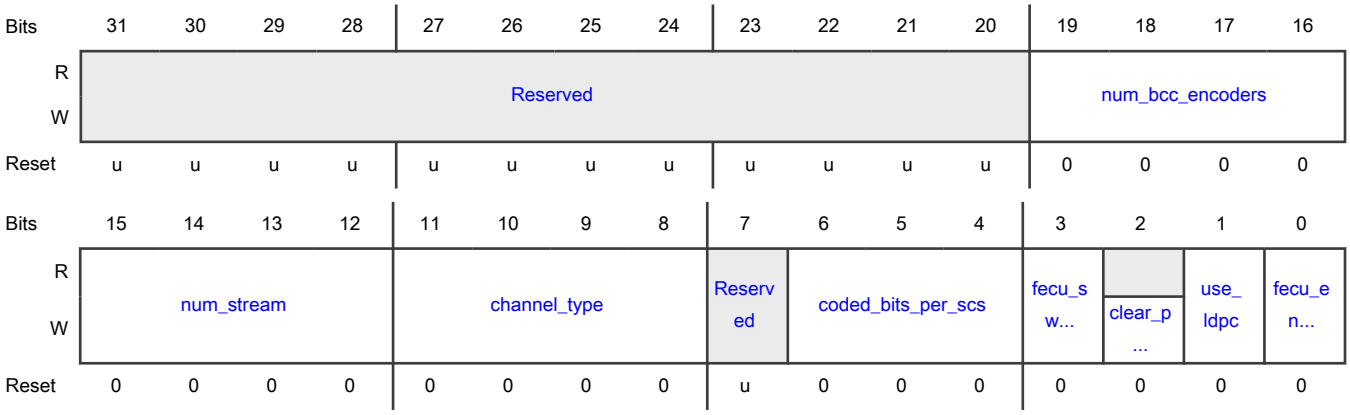
Offset	Register	Width (In bits)	Access	Reset value
384h	FECU Hardware parameters / capabilities of the LDPC encoder and decoder in FECU (FECU_LDPC_HW_PARAMS)	32	RO	0101_1B00h

13.14.2 FECU Configuration register (FECU_CONFIG)

Offset

Register	Offset
FECU_CONFIG	300h

Diagram



Fields

Field	Function
31-20 —	- Reserved
19-16 num_bcc_encoders	num_bcc_encoders The number of BCC encoders or decoders used. Must be <= max_num_bcc_encoders.
15-12 num_stream	num_stream The number of spatial streams. Must be <= max_num_streams.
11-8 channel_type	channel_type The RF bandwidth of the channel.

Table continues on the next page...

Table continued from the previous page...

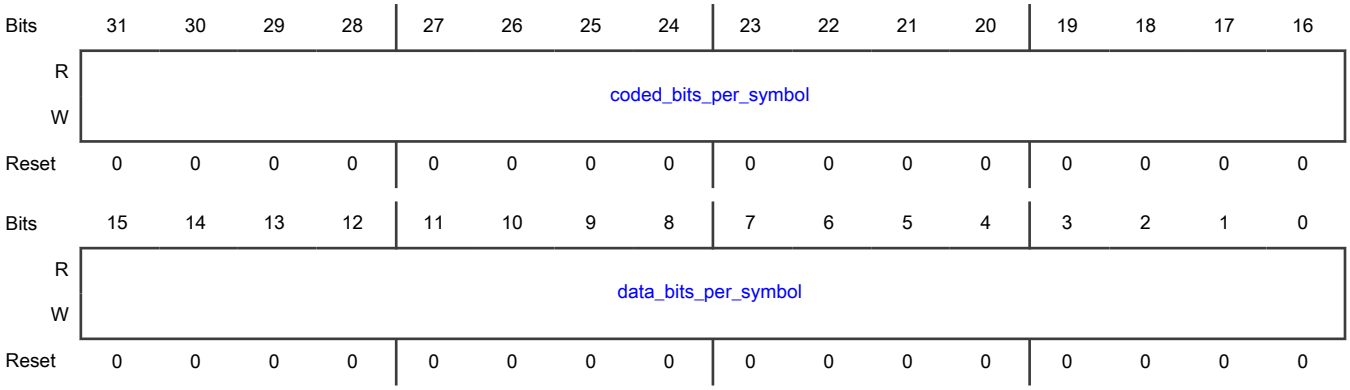
Field	Function
	0000b 20 MHz_11a 0001b 20 MHz__11ac 0010b 40 MHz_11ac 0011b 80 MHz_11ac 1000b RU26_11ax 1001b RU52_11ax 1010b RU106_11ax 1011b RU242_11ax 1100b RU484_11ax 1101b RU996_11ax
7 —	- Reserved
6-4 coded_bits_per_scs	coded_bits_per_scs The number of bits per sub-carrier per symbol. Also, the number of bits in a constellation point. 000b BPSK 001b QPSK 010b 16 QAM 011b 64 QAM 100b 256 QAM
3 fecu_sw_reset	fecu_sw_reset When high the FECU block is held in reset, and all pending commands are cleared. In order to reset FECU, this bit must remain high for 20 clock cycles.
2 clear_pending	clear_pending Writing a one to this bit will clear all pending operations in the command FIFO. This bit always reads as 0.
1 use_ldpc	use_ldpc 0b Use BCC (convolutional / Viterbi) 1b Use LDPC encoding / decoding
0 fecu_encode	fecu_encode 0b FECU is decoding 1b FECU is encoding

13.14.3 FECU Symbol size register (FECU_SIZES)

Offset

Register	Offset
FECU_SIZES	304h

Diagram



Fields

Field	Function
31-16 coded_bits_per_symbol	coded_bits_per_symbol The number of bits per symbol after the size is increased by encoding. Only used for LDPC encoding. Specifies the maximum number of bits the LDPC encoder will output.
15-0 data_bits_per_symbol	data_bits_per_symbol The number of bits per symbol before the size is increased by encoding.

13.14.4 FECU Number of padding bits register (FECU_NUM_PAD)

Offset

Register	Offset
FECU_NUM_PAD	308h

Diagram

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

Fields

Field	Function
31-16 —	
15-0 num_pad_bits	num_pad_bits The number of bits after the tail bits. This value is ignored unless final_symbol is set. Only used for BCC.

13.14.5 FECU Binary Convolutional Code (BCC) puncture mask register (FECU_BCC_PUNC_MASK)

Offset

Register	Offset
FECU_BCC_PUNC_MASK	30Ch

Diagram

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

Fields

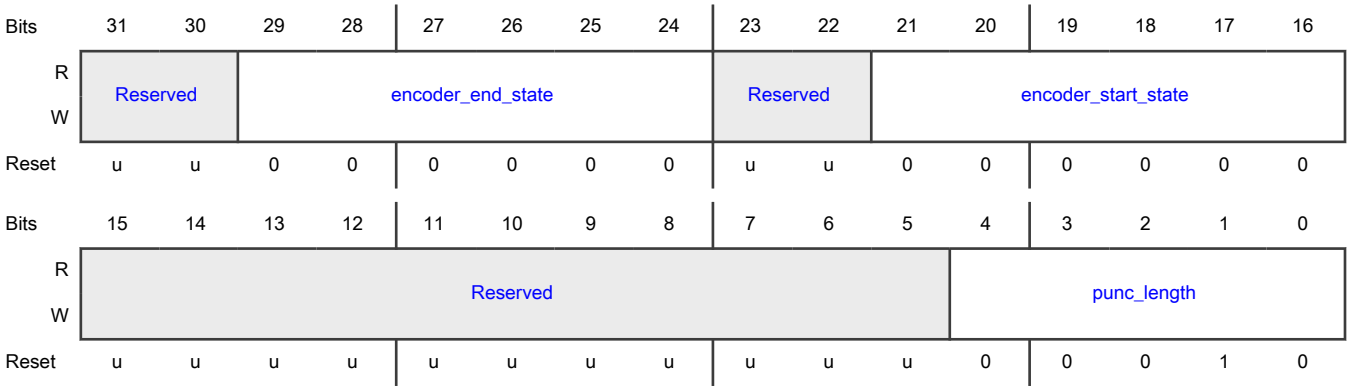
Field	Function
31-0 punc_mask	punc_mask The puncture mask used for BCC encoding and decoding.

13.14.6 FECU Binary Convolutional Code (BCC) configuration register (FECU_BCC_CONFIG)

Offset

Register	Offset
FECU_BCC_CONFIG	310h

Diagram



Fields

Field	Function
31-30 —	- Reserved
29-24 encoder_end_state	encoder_end_state The ending state of the BCC encoder. Used for both encoding and decoding.
23-22 —	- Reserved
21-16 encoder_start_state	encoder_start_state The starting state of the BCC encoder. Used for both encoding and decoding.

Table continues on the next page...

Table continued from the previous page...

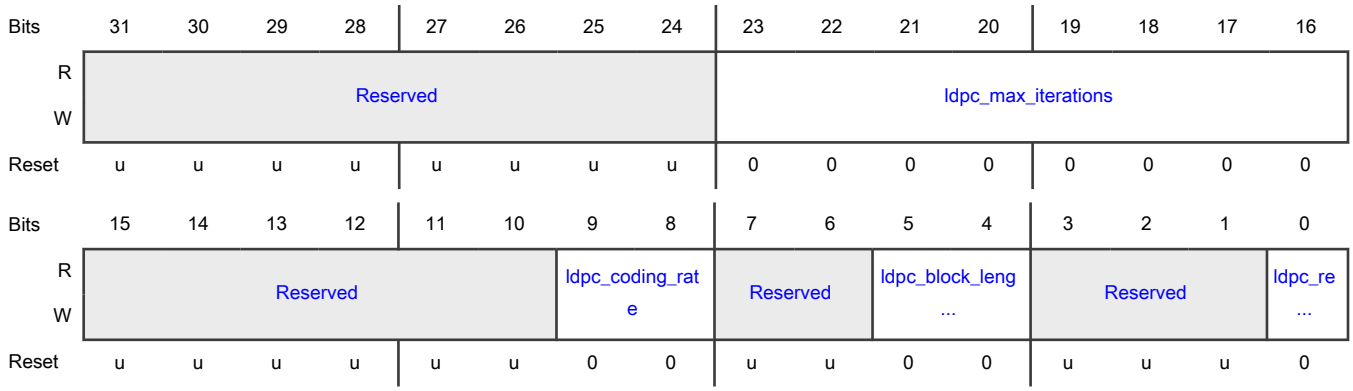
Field	Function
15-5 —	- Reserved
4-0 punc_length	punc_length The length of the valid bits in punc_mask.

13.14.7 FECU LDPC configuration register (FECU_LDPC_CONFIG)

Offset

Register	Offset
FECU_LDPC_CONFIG	314h

Diagram



Fields

Field	Function
31-24 —	- Reserved
23-16 ldpc_max_iterations	ldpc_max_iterations The maximum number of LDPC decoder iterations allowed before a decoder failure is declared. Setting this to 0 will allow 256 iterations.
15-10 —	- Reserved
9-8	ldpc_coding_rate

Table continues on the next page...

Table continued from the previous page...

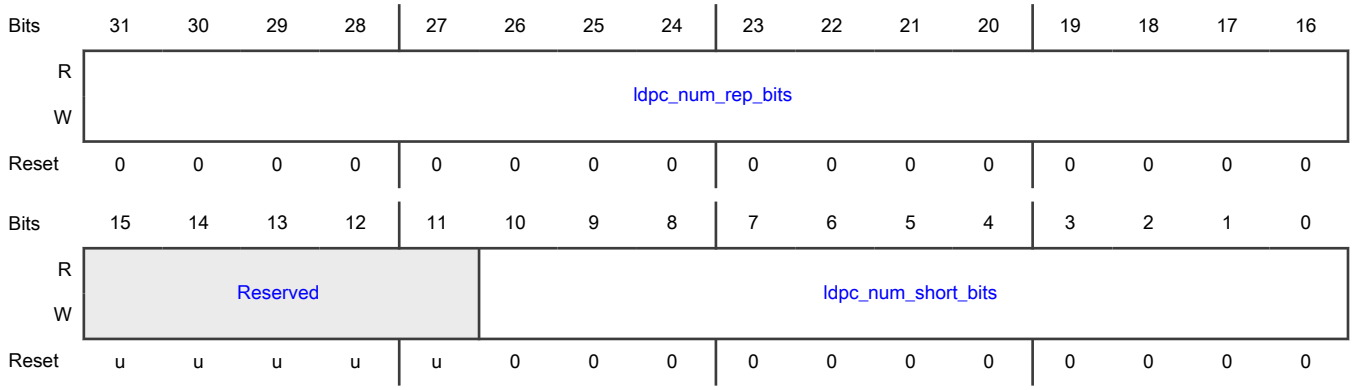
Field	Function
ldpc_coding_rate	Specifies the coding rate used for LDPC. 00b 1/2 01b 2/3 10b 3/4 11b 5/6
7-6 —	- Reserved
5-4 ldpc_block_length	ldpc_block_length Specifies the length of the LDPC codeword. 00b 648 01b 1296 10b 1944 11b Reserved
3-1 —	- Reserved
0 ldpc_repeat	ldpc_repeat Used for both encoding and decoding. 0b Use puncturing 1b Use repetition when processing LDPC blocks

13.14.8 FECU LDPC repeat, parity, and shortening sizes register (FECU_LDPC_SIZES)

Offset

Register	Offset
FECU_LDPC_SIZES	318h

Diagram



Fields

Field	Function
31-16 ldpc_num_rep_bits	ldpc_num_rep_bits The number of bits to be repeated or punctured per block, based on the ldpc_repeat bit.
15-11 —	- Reserved
10-0 ldpc_num_short_bits	ldpc_num_short_bits The number of shortening bits to add before encoding, or remove after decoding per block.

13.14.9 FECU LDPC blocks with an extra shortening bit register (FECU_LDPC_EXTRA_SHORT)

Offset

Register	Offset
FECU_LDPC_EXTRA_SHORT	31Ch

Diagram

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

Fields

Field	Function
31-0 ldpc_extra_short	ldpc_extra_short The number of LDPC blocks that have 1 extra shortening bit.

13.14.10 FECU LDPC blocks with an extra puncturing or repetition bit register (FECU_LDPC_EXTRA_REP)

Offset

Register	Offset
FECU_LDPC_EXTRA_REP	320h

Diagram

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

Fields

Field	Function
31-0 ldpc_extra_rep	ldpc_extra_rep The number of LDPC blocks that have 1 extra puncture or repeat bit.

13.14.11 FECU Bypass register (FECU_BYPASS)

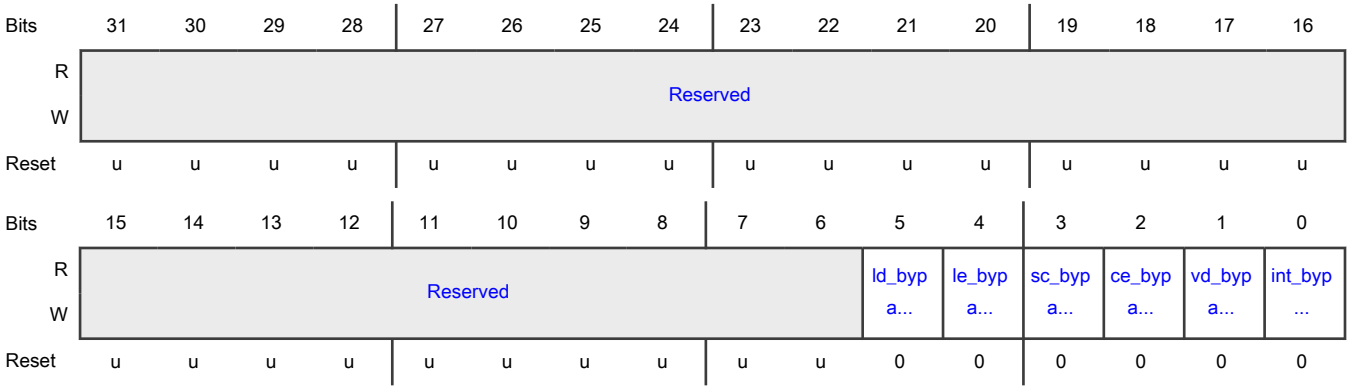
Offset

Register	Offset
FECU_BYPASS	324h

Function

Control bits used to skip some of the FECU operations.

Diagram



Fields

Field	Function
31-6 —	- Reserved
5 ld_bypass	ld_bypass When high, the LDPC decoder is bypassed.
4 le_bypass	le_bypass When high, the LDPC encoder is bypassed.
3 sc_bypass	sc_bypass When high, the scrambler is bypassed.

Table continues on the next page...

Table continued from the previous page...

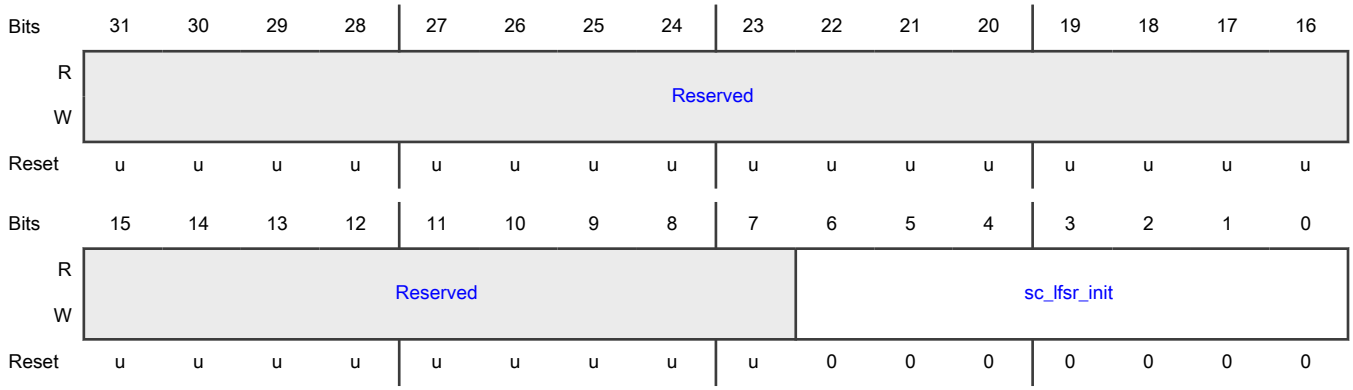
Field	Function
2 ce_bypass	ce_bypass When high, the convolutional encoder is bypassed.
1 vd_bypass	vd_bypass When high, the Viterbi decoder is bypassed.
0 int_bypass	int_bypass When high, the interleaver is bypassed.

13.14.12 FECU Scrambler / De-scrambler configuration register (FECU_SC_CONFIG)

Offset

Register	Offset
FECU_SC_CONFIG	328h

Diagram



Fields

Field	Function
31-7 —	- Reserved
6-0 sc_lfsr_init	sc_lfsr_init The initial state of the scrambler LFSR. Only used during encoding.

13.14.13 FECU DMEM Read count register (FECU_DMEM_READ_COUNT)

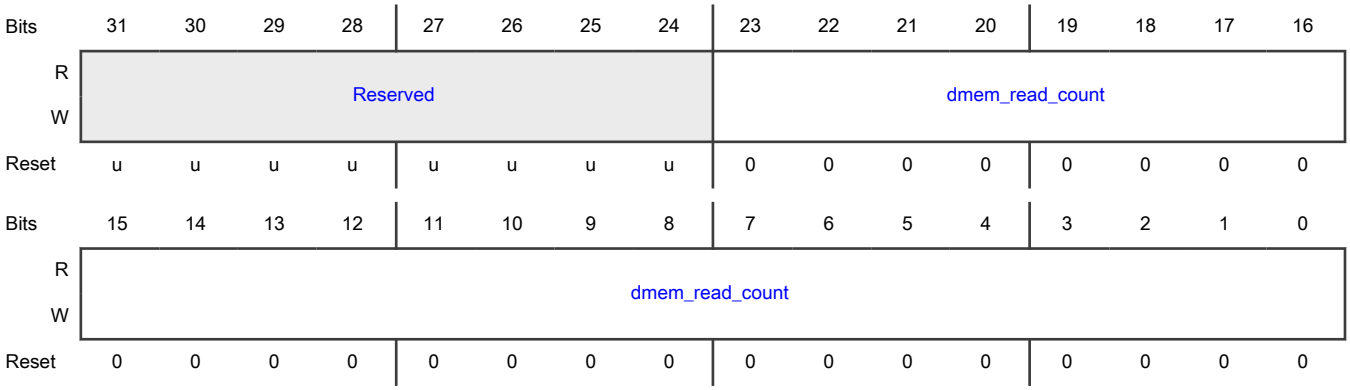
Offset

Register	Offset
FECU_DMEM_READ_COUNT	32Ch

Function

The number of items FECU should read from DMEM.

Diagram



Fields

Field	Function
31-24 —	- Reserved
23-0 dmem_read_count	dmem_read_count Number of LLRs (decode) or bits (encode) to read from DMEM.

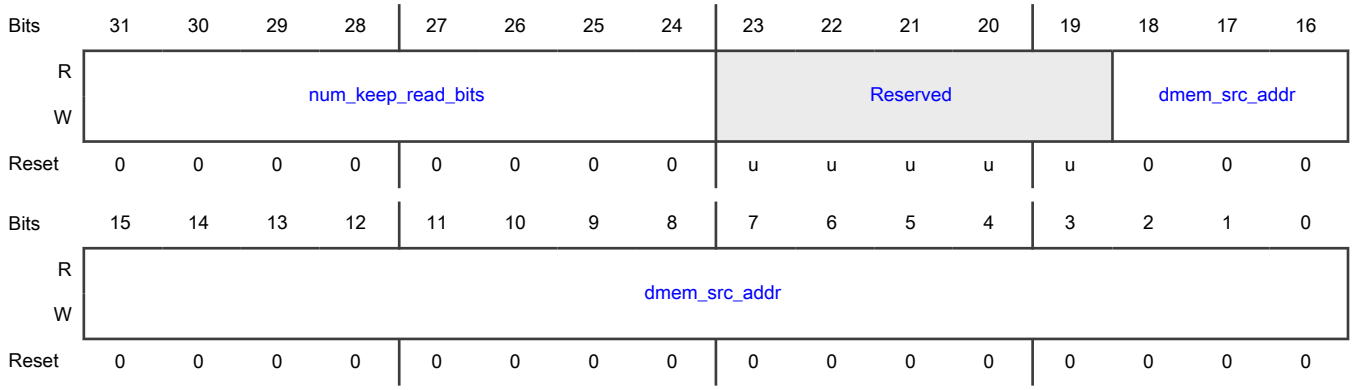
13.14.14 FECU DMEM Source address register (FECU_DMEM_SRC_ADR)

Offset

Register	Offset
FECU_DMEM_SRC_ADR	330h

Function

Diagram



Fields

Field	Function
31-24 num_keep_read_bits	num_keep_read_bits The number of bits that are read from DMEM, but not sent out to FECU for processing during this symbol. These num_keep_read_bits, will be sent out before any new data in the next symbol, if first_symbol=0. When first_symbol=1, keep bits from the previous symbol are discarded, and no keep bits are sent to FECU before new data. However, when first_symbol=1, num_keep_read_bits will be read and saved for the next symbol. The maximum value allowed is only AXI_DATA_WIDTH, even though 8 bits are allocated for this bit field.
23-19 —	- Reserved
18-0 dmem_src_addr	dmem_src_addr The source half-word (16 bit) address data is read from DMEM.

13.14.15 FECU DMEM Destination address register (FECU_DMEM_DST_ADR)

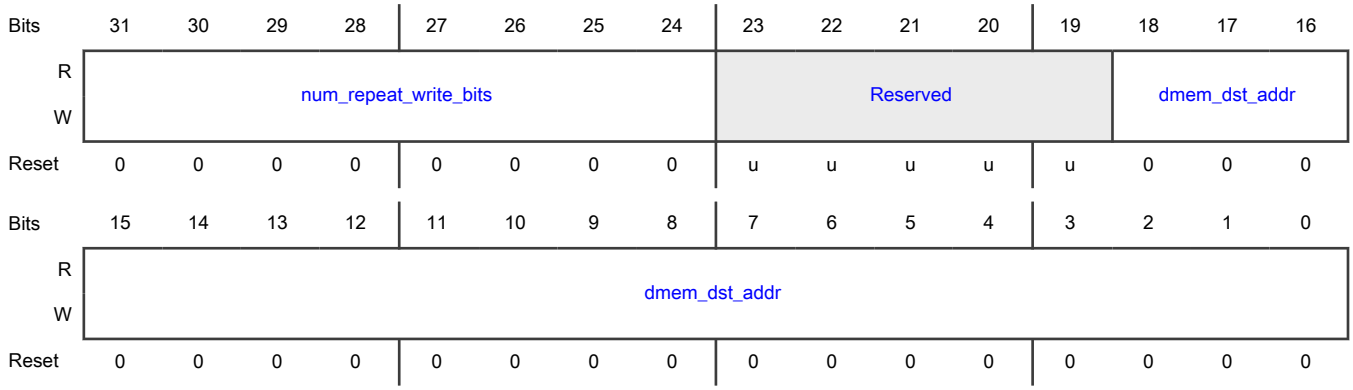
Offset

Register	Offset
FECU_DMEM_DST_ADR	334h

Function

The destination half-word (16 bit) address in DMEM.

Diagram



Fields

Field	Function
31-24 num_repeat_write_bits	num_repeat_write_bits The number of bits to repeat. These bits are written before new bits coming from FECU. They are the last bits from the previous operation. They are written starting at address dmem_dst_addr. The maximum value allowed is only AXI_DATA_WIDTH, even though 8 bits are allocated for this bit field.
23-19 —	- Reserved
18-0 dmem_dst_addr	dmem_dst_addr The half-word (16 bit) address data is written to DMEM.

13.14.16 FECU DMEM 2nd address register (FECU_DMEM_2ND_ADR)

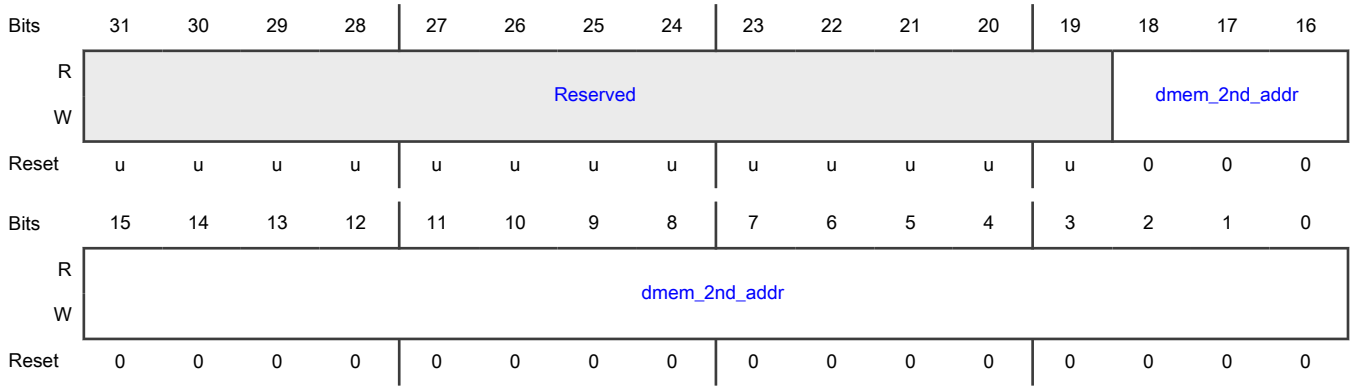
Offset

Register	Offset
FECU_DMEM_2ND_ADR	338h

Function

The half-word (16 bit) DMEM address of the 2nd stream.

Diagram



Fields

Field	Function
31-19 —	- Reserved
18-0 dmem_2nd_addr	dmem_2nd_addr The half-word (16 bit) address used for the 2nd stream. Read address when decoding, write when encoding. Only implemented when FECU_MAX_NSS >= 2.

13.14.17 FECU DMEM 3rd address register (FECU_DMEM_3RD_ADR)

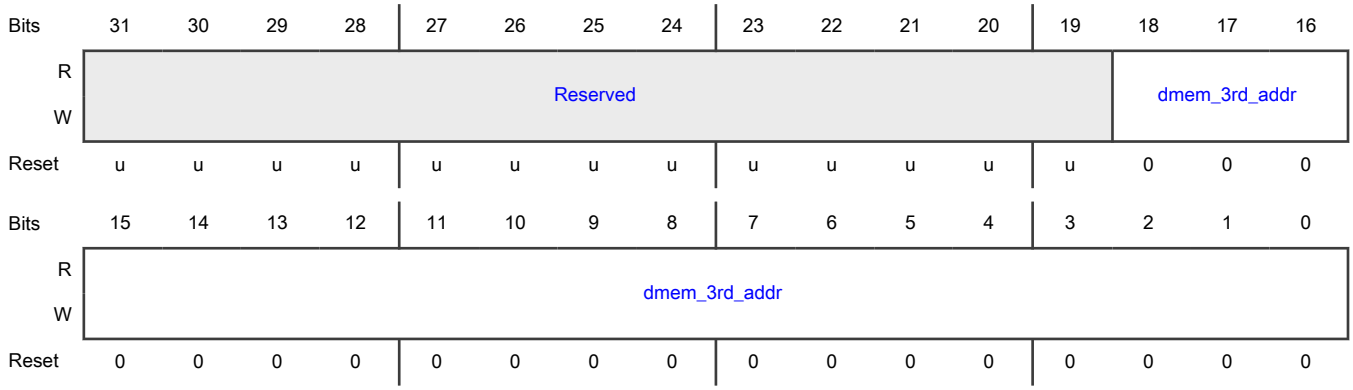
Offset

Register	Offset
FECU_DMEM_3RD_ADR	33Ch

Function

The half-word (16 bit) DMEM address of the 3rd stream.

Diagram



Fields

Field	Function
31-19 —	- Reserved
18-0 dmem_3rd_addr	dmem_3rd_addr The half-word (16 bit) address used for the 3rd stream. Read address when decoding, write when encoding. Only implemented when FECU_MAX_NSS >= 3.

13.14.18 FECU DMEM 4th address register (FECU_DMEM_4TH_ADR)

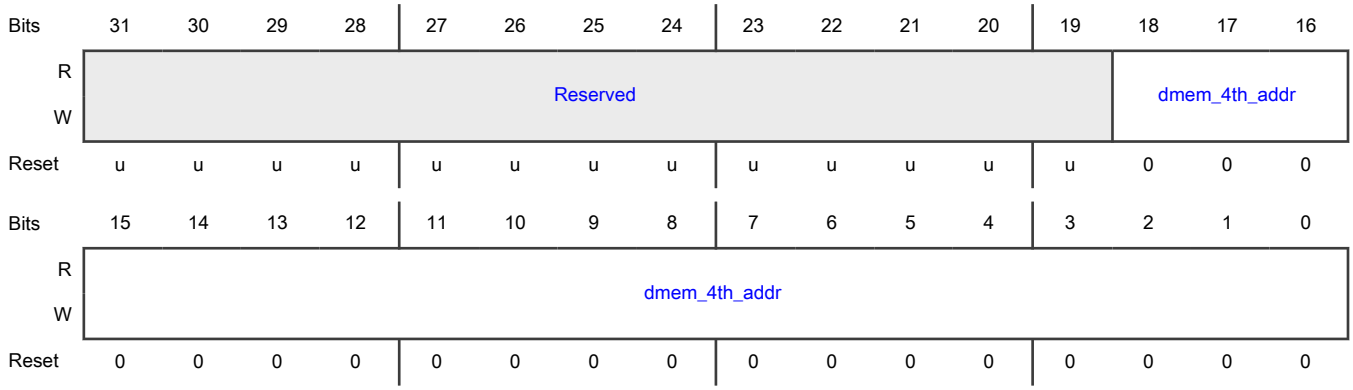
Offset

Register	Offset
FECU_DMEM_4TH_ADR	340h

Function

The half-word (16 bit) DMEM address of the 4th stream.

Diagram



Fields

Field	Function
31-19 —	- Reserved
18-0 dmem_4th_addr	dmem_4th_addr The half-word (16 bit) address used for the 4th stream. Read address when decoding, write when encoding. Only implemented when FECU_MAX_NSS >= 4.

13.14.19 FECU DMEM 5th address register (FECU_DMEM_5TH_ADR)

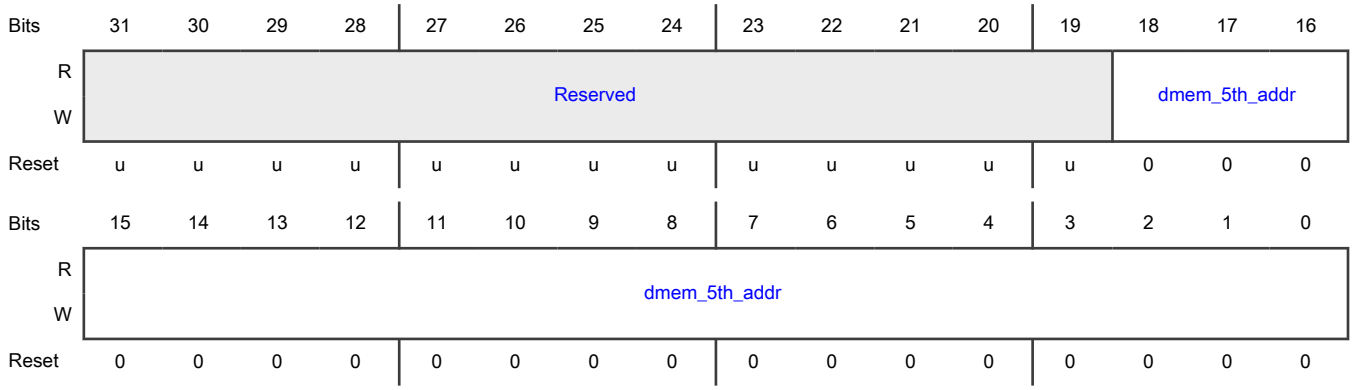
Offset

Register	Offset
FECU_DMEM_5TH_ADR	344h

Function

The half-word (16 bit) DMEM address of the 5th stream.

Diagram



Fields

Field	Function
31-19 —	- Reserved
18-0 dmem_5th_addr	dmem_5th_addr The half-word (16 bit) address used for the 5th stream. Read address when decoding, write when encoding. Only implemented when FECU_MAX_NSS >= 5.

13.14.20 FECU DMEM 6th address register (FECU_DMEM_6TH_ADR)

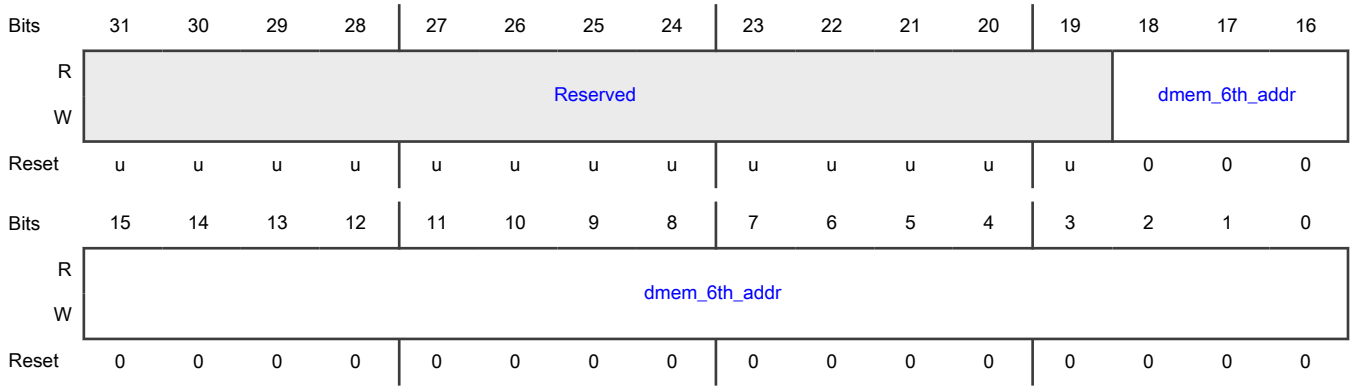
Offset

Register	Offset
FECU_DMEM_6TH_ADR	348h

Function

The half-word (16 bit) DMEM address of the 6th stream.

Diagram



Fields

Field	Function
31-19 —	- Reserved
18-0 dmem_6th_addr	dmem_6th_addr The half-word (16 bit) address used for the 6th stream. Read address when decoding, write when encoding. Only implemented when FECU_MAX_NSS >= 6.

13.14.21 FECU DMEM 7th address register (FECU_DMEM_7TH_ADR)

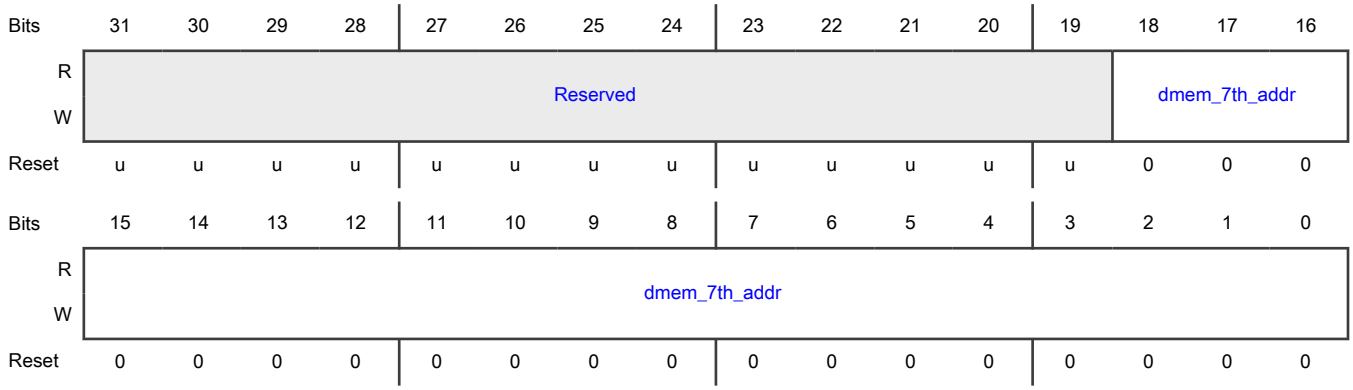
Offset

Register	Offset
FECU_DMEM_7TH_ADR	34Ch

Function

The half-word (16 bit) DMEM address of the 7th stream.

Diagram



Fields

Field	Function
31-19 —	- Reserved
18-0 dmem_7th_addr	dmem_7th_addr The half-word (16 bit) address used for the 7th stream. Read address when decoding, write when encoding. Only implemented when FECU_MAX_NSS >= 7.

13.14.22 FECU DMEM 8th address register (FECU_DMEM_8TH_ADR)

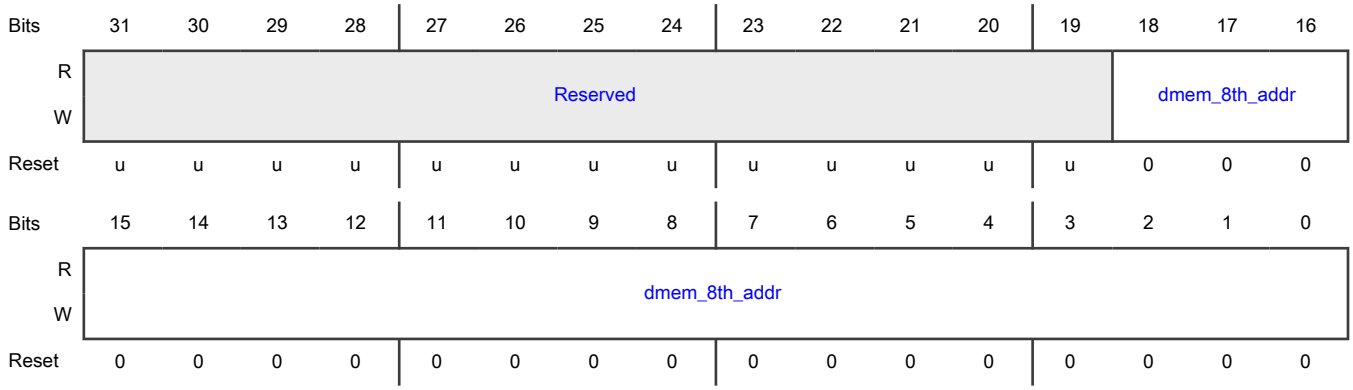
Offset

Register	Offset
FECU_DMEM_8TH_ADR	350h

Function

The half-word (16 bit) DMEM address of the 8th stream.

Diagram



Fields

Field	Function
31-19 —	- Reserved
18-0 dmem_8th_addr	dmem_8th_addr The half-word (16 bit) address used for the 8th stream. Read address when decoding, write when encoding. Only implemented when FECU_MAX_NSS >= 8.

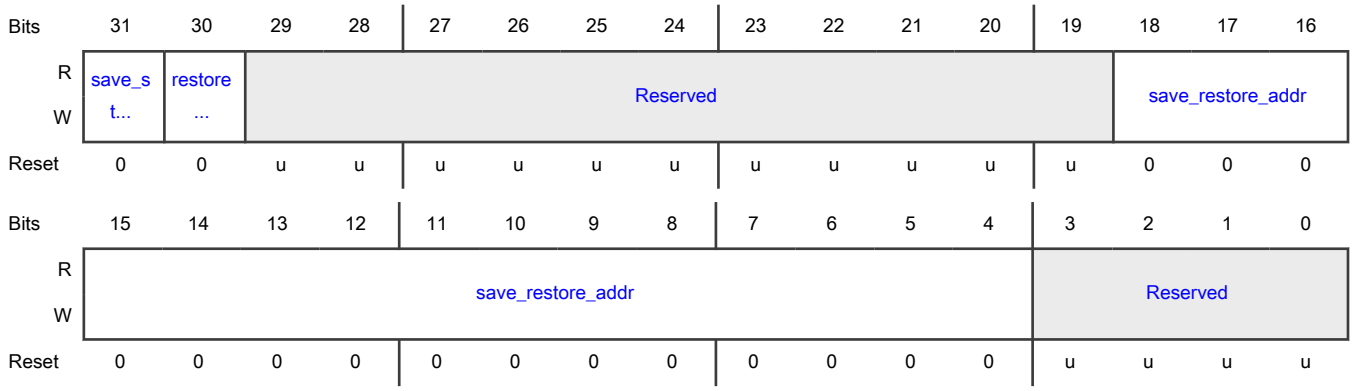
13.14.23 FECU Save and restore configuration register (FECU_SAVE_RESTORE)

Offset

Register	Offset
FECU_SAVE_RESTORE	354h

Function

Diagram



Fields

Field	Function
31 save_state	save_state When set, FECU will store FECU state (context) after the current operation completes.
30 restore_state	restore_state When set, FECU will restore FECU state before the current operation starts. When clear, FECU will continue from ending state of the last operation.
29-19 —	- Reserved
18-4 save_restore_addr	save_restore address The pointer to store FECU state after the current operation completes, or restore the state before the current operation starts.
3-0 —	- Reserved

13.14.24 FECU Control register (FECU_CONTROL)

Offset

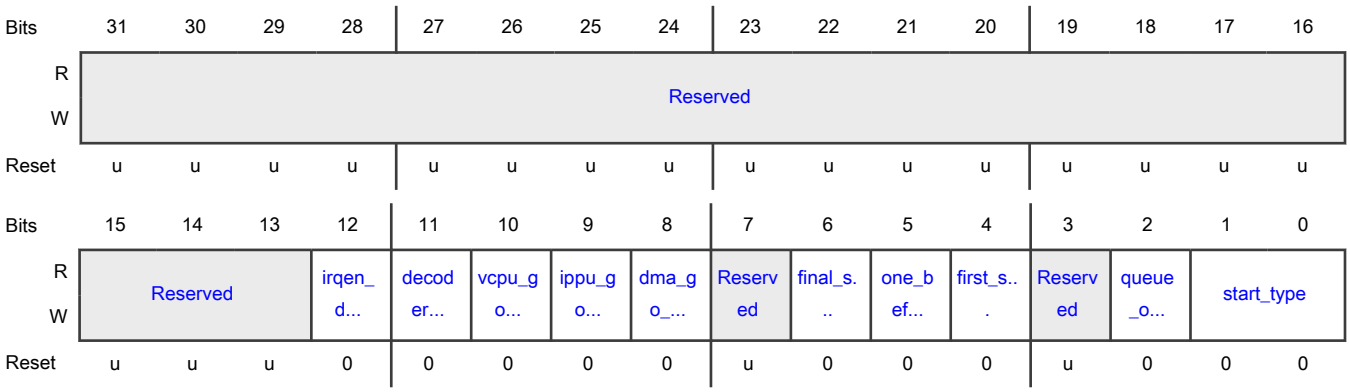
Register	Offset
FECU_CONTROL	358h

Function

A write to this register will add a pending command in the command FIFO.

Reads return the last value written.

Diagram



Fields

Field	Function
31-13 —	- Reserved
12 irqen_done	irqen_done When set, FECU will send an interrupt to the host upon a fecu_done event. When cleard, FECU will not generate a host interrupt for this FECU operation.
11 decoder_error_vcpu_go	decoder_error_vcpu_go When set, FECU will send a VCPU go when the decoder gets a decode error, and the ldpc_decode_error bit is set. When cleared, no VCPU go events will be generated as a result of ldpc_decode_error.
10 vcpu_go_enable	vcpu_go_enable When set, FECU will generate a VCPU go when FECU is done with an output buffer. When cleared, FECU will not generate a VCPU go when it is done with an output buffer.
9 ippu_go_enable	ippu_go_enable When set, FECU will generate an IPPU go when FECU is done with an output buffer. When cleared, FECU will not generate an IPPU go when it is done with an output buffer.
8 dma_go_enable	dma_go_enable When set, FECU will trigger a DMA transfer when FECU is done with an output buffer. When cleared, FECU will not trigger a DMA transfer when it is done with an output buffer.
7 —	- Reserved
6 final_symbol	final_symbol Set when this is the last symbol in a code block.
5 one_before_final_symbol	one_before_final_symbol Set when this operation is the one before the last symbol in a code block.
4 first_symbol	first_symbol Set when this is the first symbol in a code block.
3 —	- Reserved
2	queue_output When set, the write to the FECU_CONTROL register creates an output buffer command.

Table continues on the next page...

Table continued from the previous page...

Field	Function
queue_output	When cleared, no output buffer command is created. An input buffer command is always created.
1-0 start_type	start_type Determines when FECU should start. 00b Start immediately 01b Wait for an external DMA trigger 10b Wait for an external IPPU trigger 11b Reserved

13.14.25 FECU Status register (FECU_STATUS)

Offset

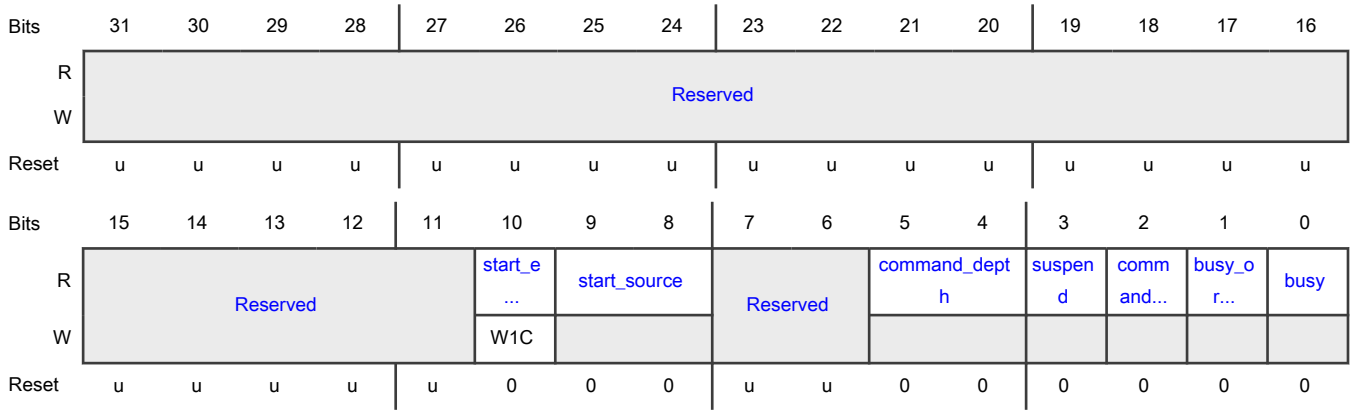
Register	Offset
FECU_STATUS	364h

Function

A write to this register will add a pending command in the command FIFO.

Reads return the last value written.

Diagram



Fields

Field	Function
31-11 —	- Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 start_error	start_error High after a write to the FECU_CONTROL register when the command FIFO is full. Cleared by writing 1 to this bit.
9-8 start_source	start_source Indicates the source of start for the current operation or last completed operation. 00b Immediate start 01b dma trigger 10b ippu trigger 11b Reserved
7-6 —	- Reserved
5-4 command_depth	command_depth Number of pending operations in the command FIFO. A maximum of 2 operations can be pending. Further attempts to start an operation will set start_error.
3 suspend	suspend High if FECU is suspended by the debugger. In this state, FECU will not access DMEM.
2 command_fifo_full	command_fifo_full High when command_depth equals 2.
1 busy_or_pending	busy_or_pending High when busy is set or there is at least one command pending in the FIFO.
0 busy	busy High while FECU is currently running an operation. Busy will be set until all output buffers have completed.

13.14.26 FECU DMEM Write count register (FECU_DMEM_WRITE_COUNT)

Offset

Register	Offset
FECU_DMEM_WRITE_COUNT	368h

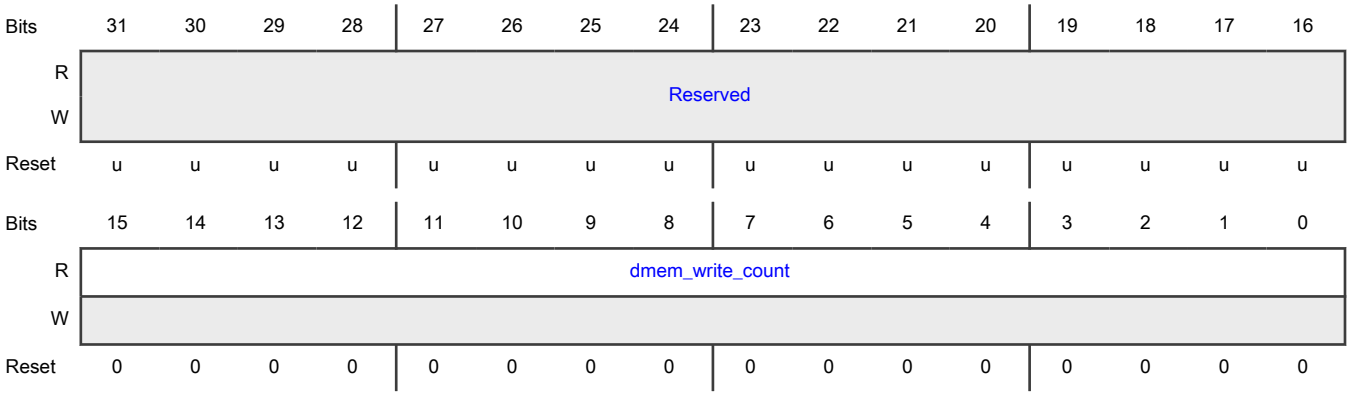
Function

The number of items FECU wrote to DMEM.

Updated as each line is written.

Used for diagnotics / debug.

Diagram



Fields

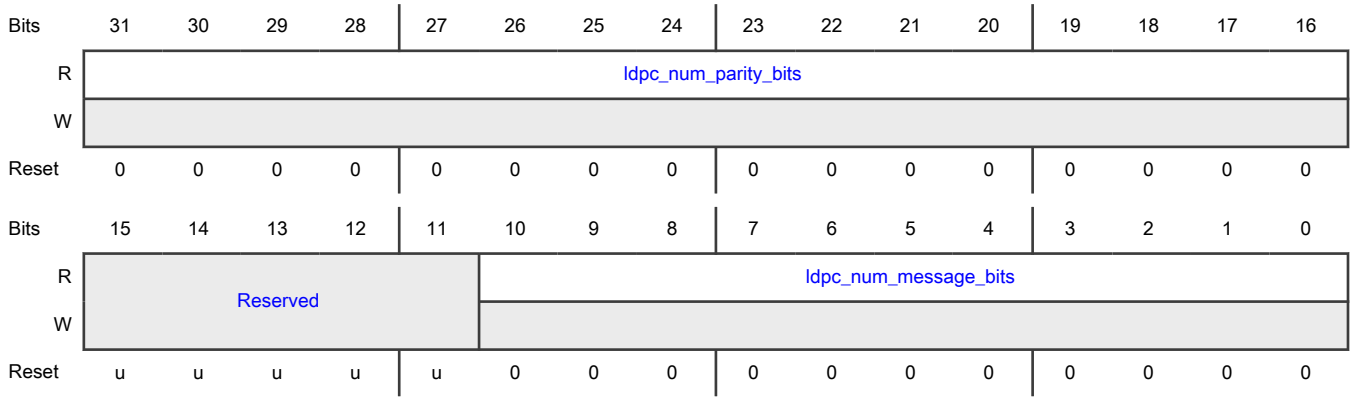
Field	Function
31-16 —	- Reserved
15-0 dmem_write_count	dmem_write_count Number of new LLRs (decode) or new bits (encode) written to DMEM. This count does not include repeated bits.

13.14.27 FECU LDPC encoder block sizes register (FECU_LDPC_ENC_BLOCK)

Offset

Register	Offset
FECU_LDPC_ENC_BLOCK	36Ch

Diagram



Fields

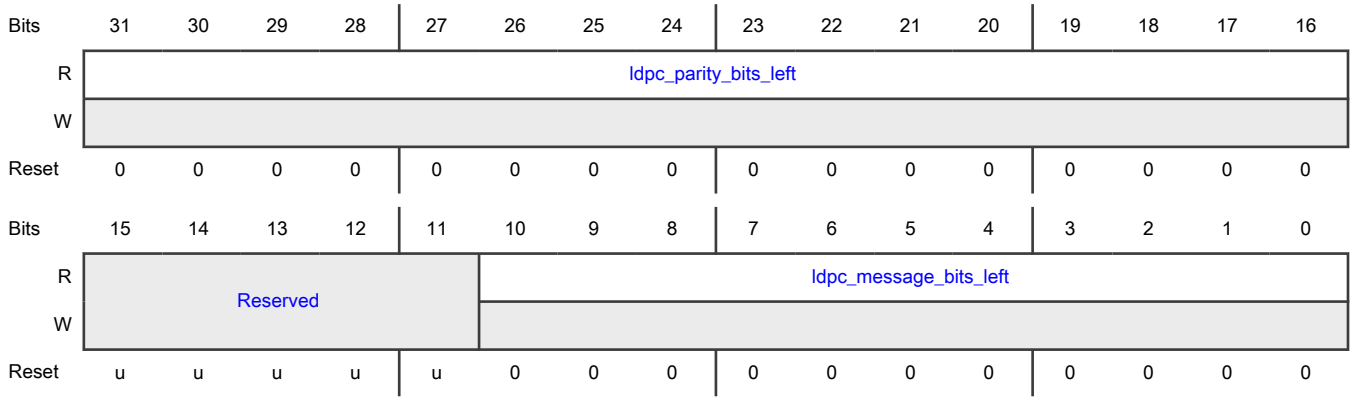
Field	Function
31-16 ldpc_num_parity_bits	ldpc_num_parity_bits The total number of parity plus repetition bits in the block the LDPC encoder is currently processing.
15-11 —	- Reserved
10-0 ldpc_num_message_bits	ldpc_num_message_bits The total number of message bits in the block the LDPC encoder is currently processing.

13.14.28 FECU LDPC encoder status register (FECU_LDPC_ENC_STATUS)

Offset

Register	Offset
FECU_LDPC_ENC_STATUS	370h

Diagram



Fields

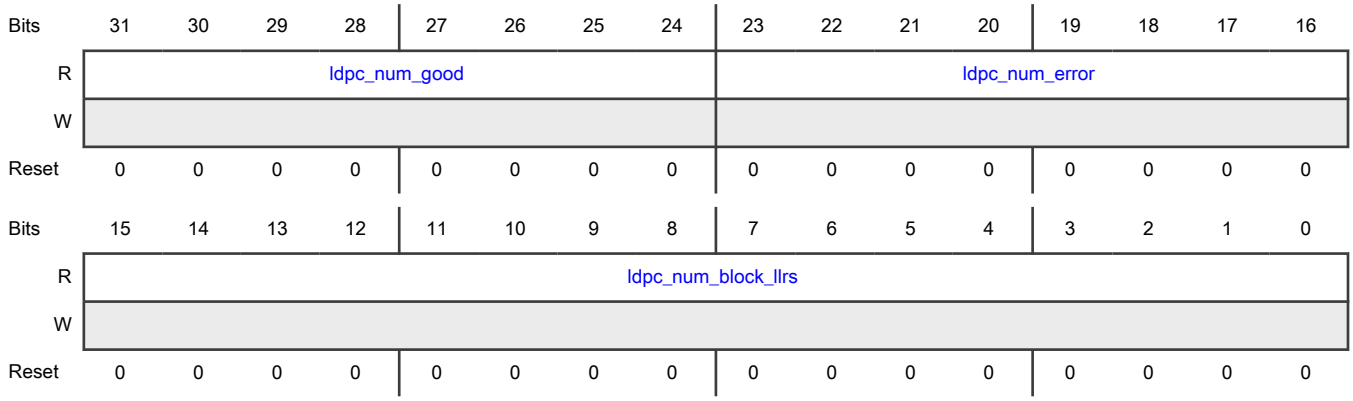
Field	Function
31-16 ldpc_parity_bits_left	ldpc_parity_bits_left The number of parity plus repeat bits that FECU still needs to send out in order to complete the current LDPC block.
15-11 —	- Reserved
10-0 ldpc_message_bits_left	ldpc_message_bits_left The number of input message bits that are still required to complete the current LDPC block.

13.14.29 FECU LDPC decoder block sizes and counts register (FECU_LDPC_DEC_BLOCK)

Offset

Register	Offset
FECU_LDPC_DEC_BLOCK	374h

Diagram



Fields

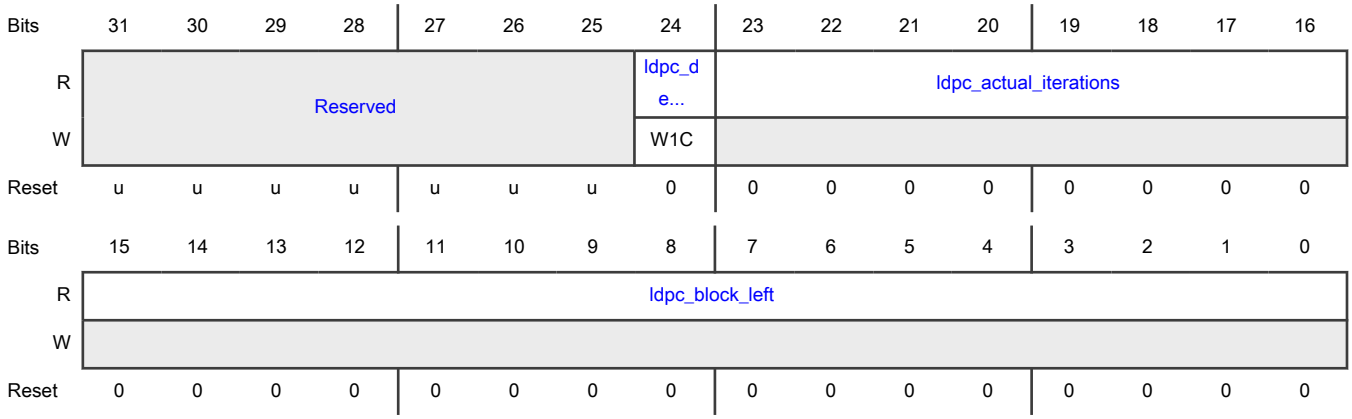
Field	Function
31-24 ldpc_num_good	ldpc_num_good The number of LDPC blocks that the decoder was able to decode, during the last symbol. The decoder is able to decode when its output satisfies the parity check matrix.
23-16 ldpc_num_error	ldpc_num_error The number of LDPC blocks that the decoder failed to decode, during the last symbol. A decoder failure occurs when the LDPC decoder's output does not satisfy the parity check matrix after ldpc_max_iterations number of iterations.
15-0 ldpc_num_block_llrs	ldpc_num_block_llrs The total number of LLRs in the block the LDPC decoder is currently processing.

13.14.30 FECU LDPC decoder status register (FECU_LDPC_DEC_STATUS)

Offset

Register	Offset
FECU_LDPC_DEC_STATUS	378h

Diagram



Fields

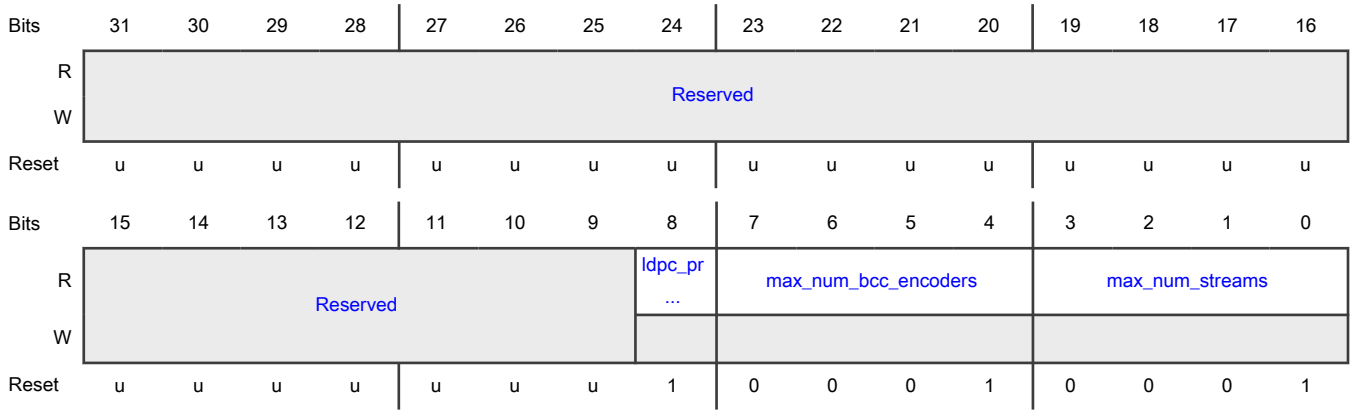
Field	Function
31-25 —	- Reserved
24 ldpc_decode_error	ldpc_decode_error The bit will be set when the LDPC decoder fails to decode a block for ldpc_max_iterations. This bit is cleared by a write to this register with a 1 in this bit position. This bit can be used to send a VCPU go event.
23-16 ldpc_actual_iterations	ldpc_actual_iterations The actual number of iterations used to decode the last LDPC block.
15-0 ldpc_block_left	ldpc_block_left The number of input LLRs that are still required to complete the current LDPC block.

13.14.31 FECU Hardware parameters / capabilities of FECU (FECU_HW_PARAMS)

Offset

Register	Offset
FECU_HW_PARAMS	380h

Diagram



Fields

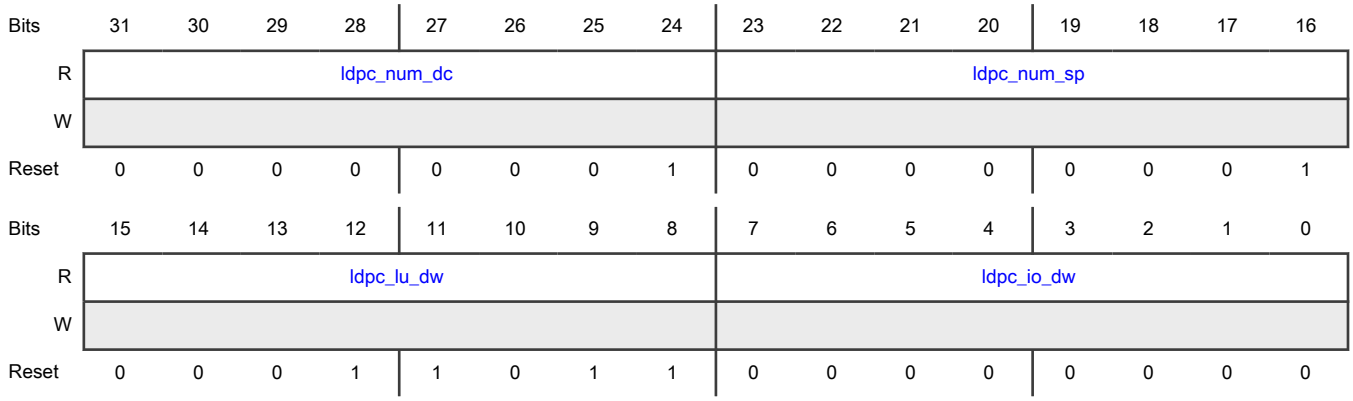
Field	Function
31-9 —	- Reserved
8 ldpc_present	ldpc_present Set when FECU includes an LDPC encoder and decoder.
7-4 max_num_bcc_encoders	max_num_bcc_encoders The maximum number of BCC encoders the FECU hardware supports that is (1).
3-0 max_num_streams	max_num_streams The maximum number of spatial streams the FECU hardware supports that is (1).

13.14.32 FECU Hardware parameters / capabilities of the LDPC encoder and decoder in FECU (FECU_LDPC_HW_PARAMS)

Offset

Register	Offset
FECU_LDPC_HW_PARAMS	384h

Diagram



Fields

Field	Function
31-24 ldpc_num_dc	ldpc_num_dc Number of decoder cores in the LDPC decoder.
23-16 ldpc_num_sp	ldpc_num_sp Number of sub-matrix processors in a single LDPC decoder core.
15-8 ldpc_lu_dw	ldpc_lu_dw LDPC load / un-load data width. The number of LLRs loaded in the LDPC core every cycle.
7-0 ldpc_io_dw	ldpc_io_dw The number of LLRs loaded into the LDPC decoder per cycle. Also, the number of bits that come out of the LDPC decoder each cycle.

Chapter 14
General Purpose I/O (GPIO)

14.1 The GPIO module as implemented on the chip

This section provides details about how the GPIO module is integrated into this chip.

Table 38. GPIO module as implemented on chip

Module name	Module base address	Supported	Number of ports
GPIO	0x230_0000	12:1, 19:16	16

NOTE

PPS_OUT, RFCTL[5:0], SPI_CS3_B, and 'Reserve for internal use' signals are outputs by default and will actively drive after reset until software configures the pin mux logic and GPIO functionality. To avoid contentions, these pins should not be connected to external devices that will drive the pin as if it were an input before software configures the pin appropriately.

GPIO functionality exists as an alternate function on the following-

Table 39. GPIO Pin Muxing

Signal Name	GPIO Signal Name
IIC1_SDA	GPIO[1]
IIC1_SCL	GPIO[2]
PPS_OUT	GPIO[3]
PPS_IN	GPIO[4]
Reserved	GPIO[6:5][1]
RFCTL[5:0]	GPIO[12:7]
SPI_CS3_B	GPIO[16]

1. The primary signal function is reserved for internal use. The GPIO[6:5] functionality, if it is to be used, must be configured by software as the alternate function after any reset.

The table below lists the primary GPIO pins and their alternate function.

Table 40. GPIO Pin Muxing

GPIO Signal Name	Alternate Function Pin name
GPIO[19]	ASLEEP
GPIO[18]	-
GPIO[17]	-

14.2 GPIO overview

This chapter describes the general-purpose I/O (GPIO) module, including signal descriptions, register settings and interrupt capabilities. This figure shows the block diagram for the GPIO module.

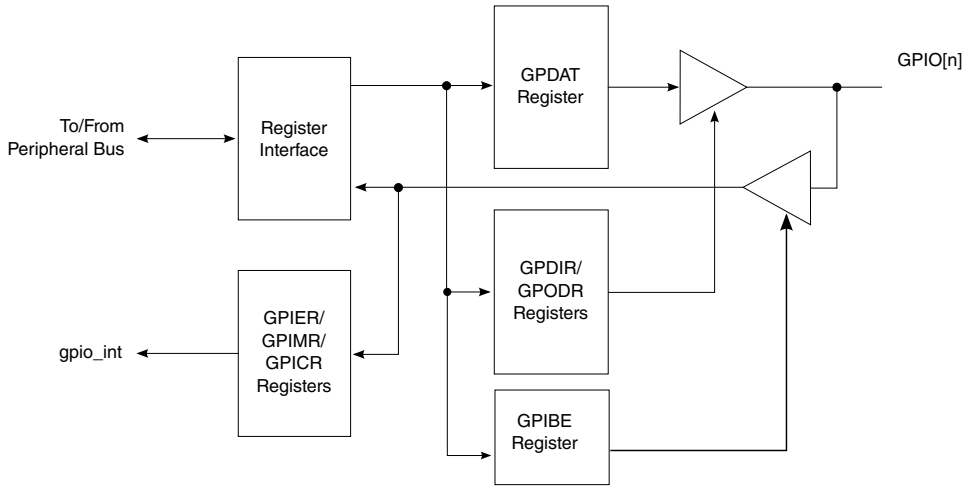


Figure 24. GPIO module block diagram

In general, the GPIO module supports up to 32 general-purpose I/O ports. Each port can be configured as an input or as an output. However, some implementations may restrict specific ports to input-only, output-only, or reserved (unimplemented). See "The GPIO module as implemented on the chip" section for more information. If a port is configured as an input, it can optionally generate an interrupt upon detection of a change. If a port is configured as an output, it can be individually configured as an open-drain or a fully active output.

14.3 GPIO features summary

The GPIO unit includes the following features:

- Supports 32 general-purpose input/output ports
- All signals are high-impedance during reset
- Open-drain capability on all ports
- All ports can optionally generate an interrupt upon changing their state
- Ports may be multiplexed with other functional signals

14.4 GPIO signal descriptions

This table provides detailed descriptions of the external GPIO signals. Note that depending on the signal multiplexing, some signals may not be available.

Table 41. GPIO external signals-detailed signal descriptions

Signal	I/O	Description	
GPIO*n*[0:31]	I/O	General Purpose I/O. Each signal can be individually set to act as input or output, according to application needs. Some implementations may restrict specific ports to input-only, output-only, or reserved (unimplemented). See "The GPIO module as implemented on the chip" section for more information.	
		State Meaning	Asserted/Negated-Defined per application.

Table continues on the next page...

Table 41. GPIO external signals-detailed signal descriptions (continued)

Signal	I/O	Description	
		Timing	Assertion/Negation-Inputs can be asserted completely asynchronously. Outputs are asynchronous to any externally visible clock.

14.5 GPIO register descriptions

The programmable register map for the GPIO module occupies 28 bytes of memory-mapped space. The full register address is comprised of the base address (specified in CCSR address space) plus the module base address, plus the specific register's offset within the module. The table below shows the memory map for the GPIO module.

All GPIO registers are 32 bits wide located on 32-bit address boundaries. Note that reading undefined portions of the memory map returns all zeros and writing has no effect.

14.5.1 GPIO Memory map

GPIO base address: 230_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	GPIO direction register (GPDIR)	32	RW	0000_0000h
4h	GPIO open drain register (GPODR)	32	RW	0000_0000h
8h	GPIO data register (GPDAT)	32	RW	0000_0000h
Ch	GPIO interrupt event register (GPIER)	32	W1C	See description.
10h	GPIO interrupt mask register (GPIMR)	32	RW	0000_0000h
14h	GPIO interrupt control register (GPICR)	32	RW	0000_0000h
18h	GPIO Input Buffer Enable Register (GPIBE)	32	RW	0000_0000h

14.5.2 GPIO direction register (GPDIR)

Offset

Register	Offset
GPDIR	0h

Function

The GPIO direction register (GPDIR) defines the direction of the individual ports.

Diagram

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

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

Fields

Field	Function
0-31 DRn	Direction. Indicates whether a pin is used as an input or an output. 00000000000000000000000000000000b - The corresponding pin is an input. 00000000000000000000000000000001b - The corresponding pin is an output.

14.5.3 GPIO open drain register (GPODR)

Offset

Register	Offset
GPODR	4h

Function

The GPIO open drain register (GPODR) defines the way individual ports drive their output.

Diagram

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

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

Fields

Field	Function
0-31 ODn	Open drain configuration. Indicates whether a signal is actively driven as an output or is an open-drain driver. This register has no effect on signals programmed as inputs in GPDIR. 00000000000000000000000000000000b - The corresponding signal is actively driven as an output. 00000000000000000000000000000001b - The corresponding signal is an open-drain driver. As an output, the signal is driven active-low, otherwise it is not driven (high impedance).

14.5.4 GPIO data register (GPDAT)

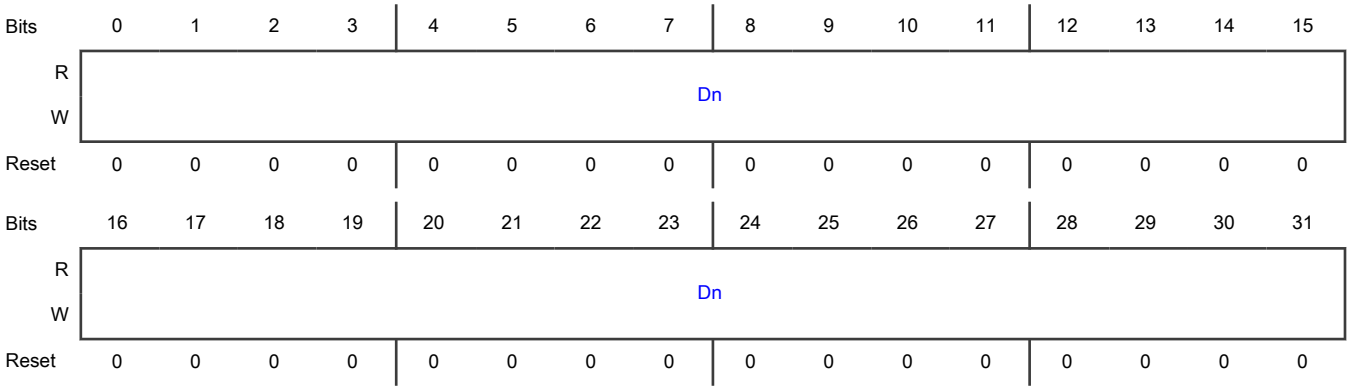
Offset

Register	Offset
GPDAT	8h

Function

The GPIO data register (GPDAT) carries the data in/out for the individual ports.

Diagram



Fields

Field	Function
0-31 Dn	Data. Writes to this register latches the data which is presented on the external pins provided the corresponding GPDIR bit is configured as an output. When GPDIR is in output mode, GPDAT read operation returns data at pin. When GPDIR is in input mode, GPDAT read operation returns state of the port.

14.5.5 GPIO interrupt event register (GPIER)

Offset

Register	Offset
GPIER	Ch

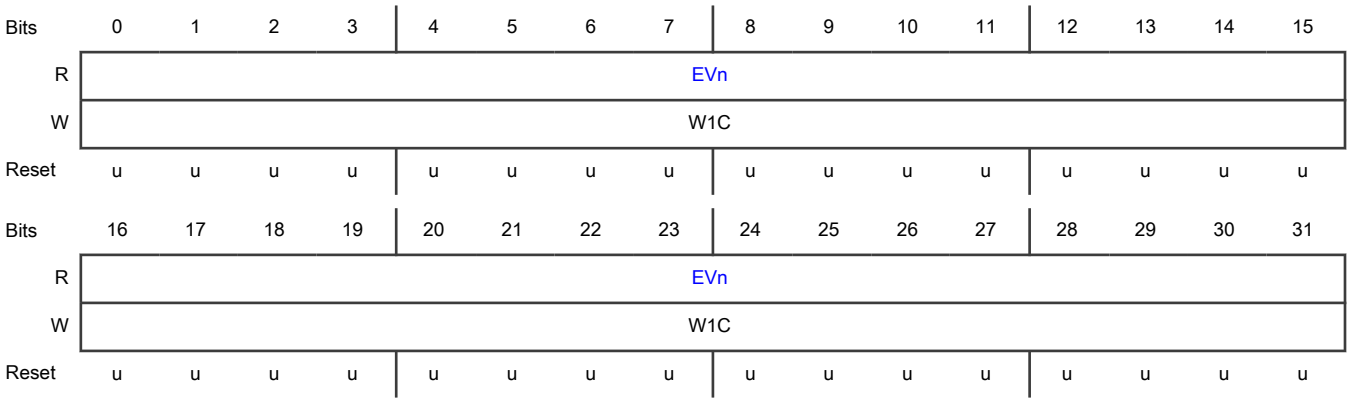
Function

The GPIO interrupt event register (GPIER) carries information of the events that caused an interrupt. Each bit in GPIER, corresponds to an interrupt source. GPIER bits are cleared by writing ones. However, writing zero has no effect.

NOTE

Some implementations may ignore the interrupt mask as configured in GPIMR. In these implementations, a GPIER bit can be set even though the associated interrupt is masked. See The "GPIO module as implemented on the chip" section for more information.

Diagram



Fields

Field	Function
0-31 EVn	Interrupt events. Indicates whether an interrupt event occurred on the corresponding GPIO signal. 00000000000000000000000000000000b - No interrupt event occurred on the corresponding GPIO signal. 00000000000000000000000000000001b - An interrupt event occurred on the corresponding GPIO signal.

14.5.6 GPIO interrupt mask register (GPIMR)

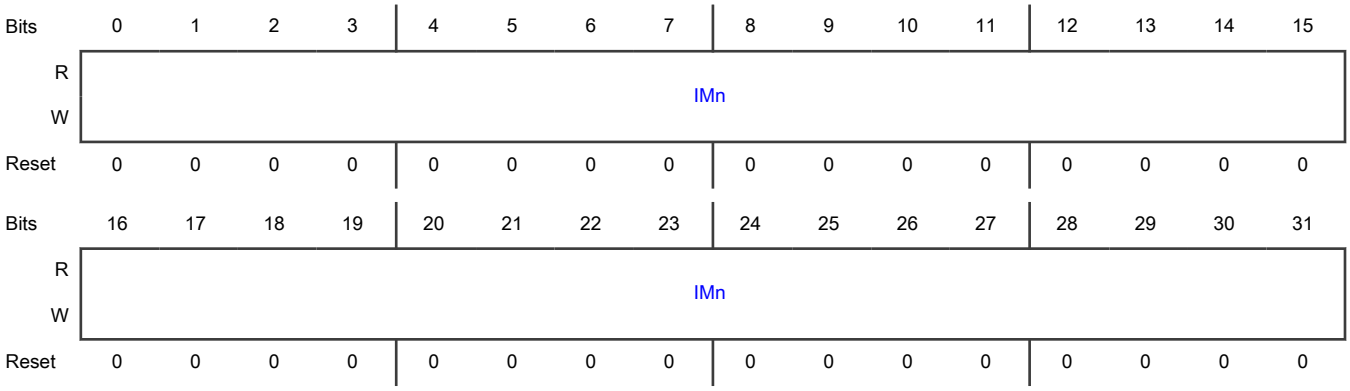
Offset

Register	Offset
GPIMR	10h

Function

The GPIO interrupt mask register (GPIMR) defines the interrupt masking for the individual ports. When a masked interrupt request occurs, the corresponding GPIER bit is set, regardless of the GPIMR state. When one or more non-masked interrupt events occur, the GPIO module issues an interrupt to the interrupt controller.

Diagram



Fields

Field	Function
0-31 IMn	Interrupt mask. Indicates whether an interrupt event is masked or not masked for the corresponding GPIO signal. 00000000000000000000000000000000b - The input interrupt signal is masked (disabled). 00000000000000000000000000000001b - The input interrupt signal is not masked (enabled).

14.5.7 GPIO interrupt control register (GPICR)

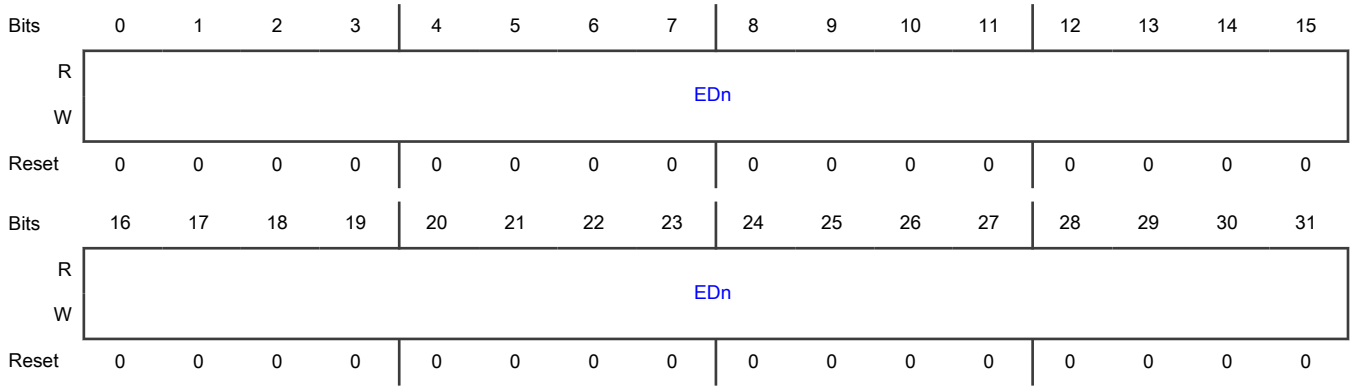
Offset

Register	Offset
GPICR	14h

Function

The GPIO interrupt control register (GPICR) determines whether the corresponding port line asserts an interrupt request upon either a high-to-low change or any change on the state of the signal.

Diagram



Fields

Field	Function
0-31 EDn	Edge detection mode. The corresponding port line asserts an interrupt request according to the following: 00000000000000000000000000000000b - Any change on the state of the port generates an interrupt request. 00000000000000000000000000000001b - High-to-low change on the port generates an interrupt request.

14.5.8 GPIO Input Buffer Enable Register (GPIBE)

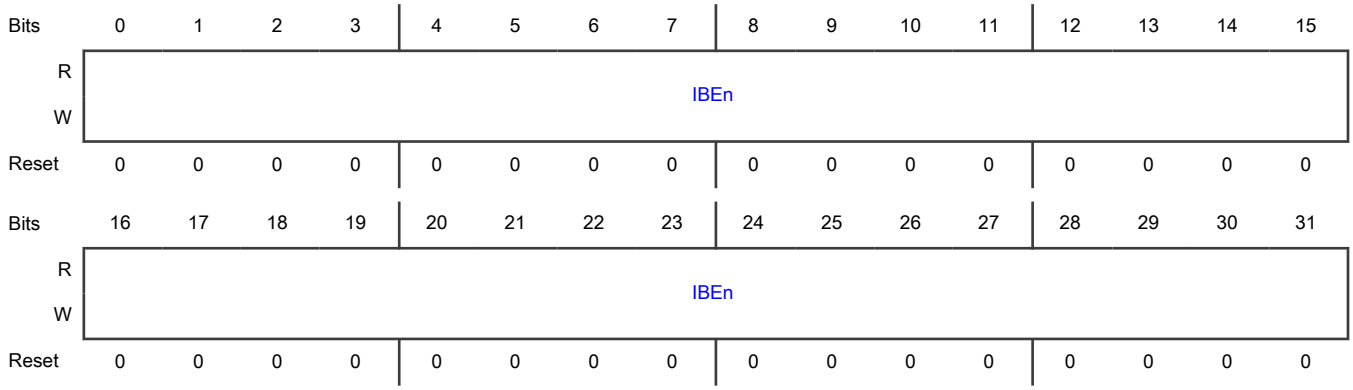
Offset

Register	Offset
GPIBE	18h

Function

The GPIO Input Buffer Enable register is used to control the input enable of each individual GPIO port. When an individual GPIO port's direction is set to input (GPIO_GPDIR[DRn=0]), the associated input enable must be set (GPIOxGPIE[IEn]=1) to propagate the port value to the GPIO Data Register. When a port's input enable is disabled, the port value does not propagate to the GPIO Data Register. Unused GPIO ports may be non-connected on the board if their "direction" and "input enable" are configured for "input" (GPIO_GPDIR[DRn=0]) and "disabled" (GPIBE[IEn=0]) respectively, which is the default configuration (based on GPIO input signal GPIBE_RESET_VAL[31:0] tied to 0x0000_0000 which sets the reset value of this register).

Diagram



Fields

Field	Function
0-31 IBEn	Input Enable. Indicates whether a GPIO port's input is enabled. 00000000000000000000000000000000b - The corresponding port input is disabled. 00000000000000000000000000000001b - The corresponding port input is enabled.

Chapter 15
Inter-Integrated Circuit (I2C)

15.1 I2C module as implemented on the chip

This section provides details about how the I2C module is integrated into this chip.

I2C slave mode is not implemented on this chip.

Table 42. I2C module as implemented on chip

Module name	Module base address
I2C1	0x200_0000
I2C2	0x201_0000

15.2 Overview

This chapter describes the Inter-Integrated Circuit (I^2C) module implemented on this chip and presents the following topics:

- Introduction to I^2C
- External signal descriptions
- Memory map and register definition
- Functional description
- Initialization/application information

15.3 Introduction to I^2C

This section presents the following topics:

- Definition: I^2C module
- Advantages of the I^2C bus
- Module block diagram
- Features
- Modes of operation
- Definition: I^2C conditions

15.3.1 Definition: I^2C module

The I^2C module is a functional unit that provides a two-wire—serial data (SDA) and serial clock (SCL)—bidirectional serial bus. This bus provides a simple and efficient method of data exchange between this chip and other devices, such as microcontrollers, EEPROMs, real-time clock devices, analog-to-digital converters, and LCDs.

15.3.2 Advantages of the I^2C bus

The synchronous, multiple-master two-wire I^2C bus:

- Minimizes interconnections between devices
- Allows the connection of additional devices to the bus for expansion and system development
- Includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously

- Does not require an external address decoder

15.3.3 Module block diagram

The following figure shows a block diagram of the I²C module.

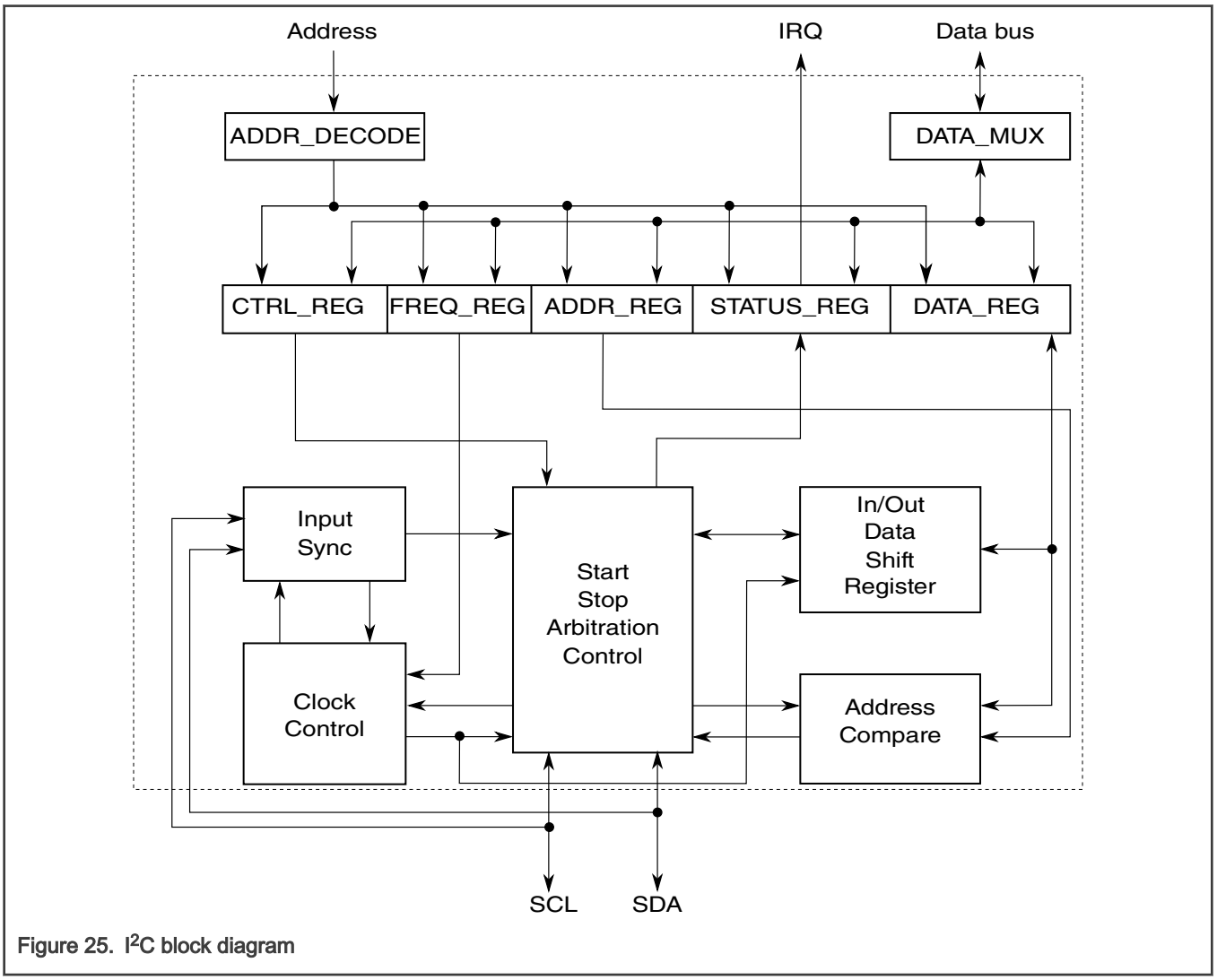


Figure 25. I²C block diagram

15.3.4 Features

The I²C module has the following key features:

- Compatible with I²C bus standard[1]
- Operating speeds
  - Up to 100 kbps in Standard Mode
  - Up to 400 kbps in Fast Mode
  - Operation at higher baud rates (up to a maximum of module clock/20) with reduced bus loading
  - Actual baud rate dependent on the SCL rise time (which depends on external pullup resistor values and bus loading)
- Multi-master operation

[1] Compliant with I²C 2.0 standard with the exception that HS (high speed) mode is not supported

- Software-programmable for one of 256 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- Maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF

15.3.5 Modes of operation

The I^2C module supports the chip modes described in Table 43.

Table 43. Chip modes supported by the I^2C module

Chip mode	Description	Important notes
RUN	Basic mode of operation	—
STOP	The lowest-power mode that allows the chip to turn off all the clocks to the I^2C module	The I^2C module can enter this mode when there are no active transfers (active data between valid START and STOP conditions) on the bus. See STOP mode.
DEBUG	Allows the chip to freeze all ongoing activities (such as an ongoing transaction, counter values, and register status) for debugging	See DEBUG mode.

In addition to chip modes, the I^2C module has several module-specific modes. These are described in Table 44.

Table 44. Module-specific modes supported by the I^2C module

Module mode	Description	Important notes
Master mode	The I^2C module is the driver of the SDA line.	• Do not use the I^2C module's slave address as a calling address. • The I^2C module cannot be a master and a slave simultaneously.
Slave mode	The I^2C module is not the driver of the SDA line.	• Enable the I^2C module before a START condition from a non-I^2C master is detected. • By default the I^2C module performs as a slave receiver.

15.3.6 Definition: I^2C conditions

The following table shows the I^2C-specific conditions defined for the I^2C module.

Table 45. I²C conditions

Condition	Description
START	A condition that denotes the beginning of a new data transfer and awakens all slaves. Each data transfer contains several bytes of data. It is defined as a high-to-low transition of SDA while SCL is high, as shown in the following figure.
STOP	A condition generated by the master to terminate a transfer and free the bus. It is defined as a low-to-high transition of SDA while SCL is high, as shown in the following figure.
Repeated START	A START condition that is generated without a STOP condition to terminate the previous transfer.

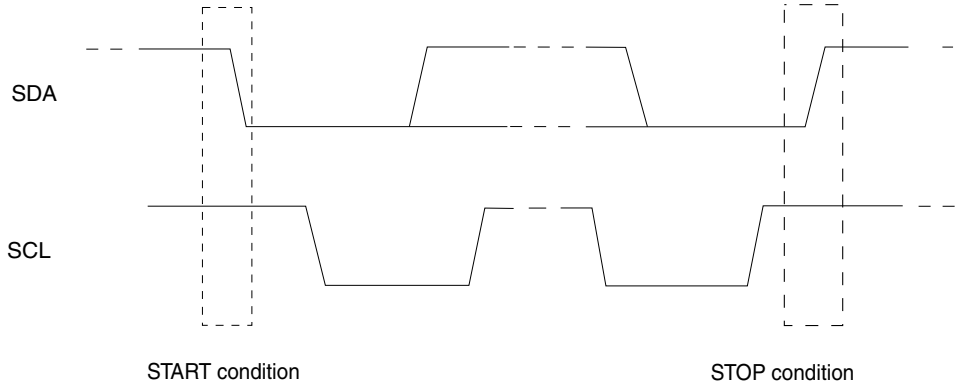


Figure 26. START and STOP conditions

15.4 External signal descriptions

This section presents the following topics:

- Signal overview
- Detailed external signal descriptions

15.4.1 Signal overview

The I²C module uses the Serial Data (SDA) and Serial Clock (SCL) signals as a communication interconnect with other devices. The signal patterns driven on the SDA signal represent address, data, or read/write information at different stages of the protocol.

All devices connected to the SDA and SCL signals must have open-drain or open-collector outputs. The I²C bus connects the SDA lines of all devices together, thus performing a logical AND of the SDA signals. Similarly, the I²C bus connects all the SCL lines together, thus performing a logical AND of the SCL signals. The combined SDA and SCL lines also connect to their respective pullup resistors. For the electrical characteristics of these signals, see the data sheet for this chip.

15.4.2 Detailed external signal descriptions

The SDA and SCL signals are described in the following table.

Table 46. External signal descriptions

Signal	Description
SCL	Bidirectional serial clock line of the module, compatible with the I²C bus specification
SDA	Bidirectional serial data line of the module, compatible with the I²C bus specification

15.5 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the I^{2}C module. It presents the following topics:

- Register accessibility
- I2C Bus Address Register (IBAD)
- I2C Bus Frequency Divider Register (IBFD)
- I2C Bus Control Register (IBCR)
- I2C Bus Status Register (IBSR)
- I2C Bus Data I/O Register (IBDR)
- I2C Bus Interrupt Config Register (IBIC)
- I2C Bus Debug Register (IBDBG)

15.5.1 Register accessibility

Address location 0x0007 is a reserved location, but access to this location will not generate any bus error.

All the I^{2}C registers are one byte wide. Reads and writes to these registers must be byte-wide operations.

15.5.2 I2C register descriptions

The memory map for the I^{2}C module is given below. The total address for each register is the sum of the base address for the I^{2}C module and the address offset for each register.

15.5.2.1 I2C Memory map

I2C1 base address: 200_0000h

I2C2 base address: 201_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	I2C Bus Address Register (IBAD)	8	RW	00h
1h	I2C Bus Frequency Divider Register (IBFD)	8	RW	00h
2h	I2C Bus Control Register (IBCR)	8	RW	80h
3h	I2C Bus Status Register (IBSR)	8	W1C	80h
4h	I2C Bus Data I/O Register (IBDR)	8	RW	00h
5h	I2C Bus Interrupt Config Register (IBIC)	8	RW	00h
6h	I2C Bus Debug Register (IBDBG)	8	RW	00h

15.5.2.2 I2C Bus Address Register (IBAD)

Offset

Register	Offset
IBAD	0h

Function

This register contains the address that the I^{2}C bus responds to when addressed as a slave. This is not the address sent on the bus during the address transfer.

Access: Supervisor mode only (the module ignores user mode accesses and does not assert an error response if you access this register in user mode)

Diagram

Bits	7	6	5	4	3	2	1	0
R	ADR							0
W								
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7-1 ADR	Slave address Specific slave address to be used by the I^{2}C module. **NOTE** The default mode of I^{2}C bus is slave mode for an address match on the bus.
0 —	Reserved

15.5.2.3 I2C Bus Frequency Divider Register (IBFD)

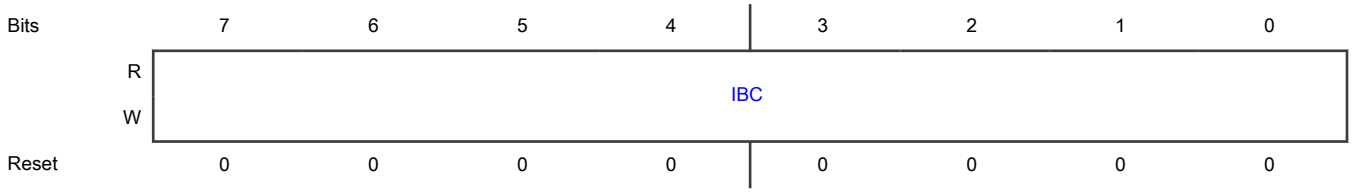
Offset

Register	Offset
IBFD	1h

Function

Access: Supervisor mode only (the module ignores user mode accesses and does not assert an error response if you access this register in user mode)

Diagram



Fields

Field	Function
7-0 IBC	IBC I-Bus clock rate. This field is used to prescale the bus clock for bit rate selection. See Clock rate and IBFD settings.

15.5.2.4 I2C Bus Control Register (IBCR)

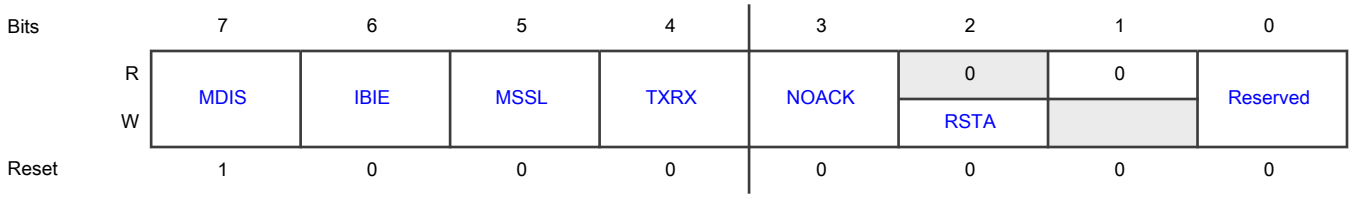
Offset

Register	Offset
IBCR	2h

Function

Access: Supervisor mode only (the module ignores user mode accesses and does not assert an error response if you access this register in user mode)

Diagram



Fields

Field	Function
7 MDIS	Module disable This field controls the software reset of the entire I^2C module.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE If the I^2C module is enabled in the middle of a byte transfer, the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent START condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated, the current bus cycle may become corrupt. This ultimately results in either the current bus master or the I^2C module losing arbitration, after which bus operation returns to normal. 0b - The module is enabled. You must clear this field before any other IBCR fields have any effect. 1b - The module is reset and disabled. This is the power-on reset situation. When high, the interface is held in reset, but registers can still be accessed. Status register fields (IBSR) are not valid when the module is disabled.
6 IBIE	I2C bus interrupt enable 0b - Interrupts from the I^2C module are disabled. This does not clear any currently pending interrupt condition. 1b - Interrupts from the I^2C module are enabled. An I^2C interrupt occurs if IBSR[IBIF]=1.
5 MSSL	Master/slave mode select When you change this field from 0 to 1, the module generates a START signal on the bus and selects the master mode. When you change this field from 1 to 0, the module generates a STOP signal and changes the operation mode from master to slave. You should generate a STOP signal only if IBSR[IBIF]=1. The module clears this field without generating a STOP signal when the master loses arbitration. 0b - Slave mode 1b - Master mode
4 TXRX	Transmit/receive mode select This field selects the direction of master and slave transfers. When the I^2C module is addressed as a slave, your software must program this field based on IBSR[SRW]: • When IBSR[SRW]=0, program TXRX=0. • When IBSR[SRW]=1, program TXRX=1. In master mode, your software must program this field according to the type of transfer required. Therefore, for address cycles, this field will always be 1. 0b - Receive 1b - Transmit
3 NOACK	Data acknowledge disable This field specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. When the I^2C module is enabled, it always acknowledges address matches, regardless of the value of NOACK.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE Values written to this field are only used when the I^2C module is a receiver, not a transmitter. 0b - The module sends an acknowledge signal to the bus at the 9th clock bit after receiving one byte of data. 1b - The module does not send an acknowledge-signal response (that is, acknowledge bit = 1).
2 RSTA	Repeat START If the I^2C module is the current bus master, and you program RSTA=1, the I^2C module generates a repeated START condition. This field always reads as a 0. If you attempt a repeated START at the wrong time—if the bus is owned by another master—the result is loss of arbitration. 0b - No effect 1b - Generate repeat start cycle
1 —	Reserved
0 —	Reserved Although this field supports read/write access, writing to this field is not recommended, and can produce unexpected results.

15.5.2.5 I2C Bus Status Register (IBSR)

Offset

Register	Offset
IBSR	3h

Function

Access: Supervisor mode only (the module ignores user mode accesses and does not assert an error response if you access this register in user mode)

Diagram

Bits	7	6	5	4	3	2	1	0
R	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
W				W1C			W1C	
Reset	1	0	0	0	0	0	0	0

Fields

Field	Function
7 TCF	Transfer complete The I2C module programs this field to 0 during the time that the module transfers one byte of data. After the I2C module detects the falling edge of the 9th clock of a byte transfer, the module programs this field to 1. p> **NOTE** This field is relevant to you only during or immediately following a transfer to or from the I2C module. 0b - Transfer in progress 1b - Transfer complete
6 IAAS	Addressed as a slave When its own specific address (I-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IBIE is set. Then the CPU needs to check the SRW bit and set the TXRX field accordingly. Writing to the I-Bus Control Register clears this bit. 0b - Not addressed 1b - Addressed as a slave
5 IBB	Bus busy This field indicates the status of the bus. When a START signal is detected, IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state. **NOTE** Software must ensure that the I2C bus is idle by checking the IBSR[IBB] field (bus busy) before switching to master mode and attempting a START cycle. 0b - Bus is Idle 1b - Bus is busy
4 IBAL	IBAL Arbitration Lost. The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances: • SDA is sampled low when the master drives a high during an address or data transmit cycle.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• SDA is sampled low when the master drives a high during the acknowledge bit of a data receive cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in slave mode. • A stop condition is detected when the master did not request it. This bit must be cleared by software, by writing a one to it. A write of zero has no effect.
3 —	Reserved
2 SRW	SRW Slave Read/Write. When the IAAS bit is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is only valid when the I-Bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated. By reading this field, the CPU can detect slave transmit/receive mode according to the command of the master. 0b - Slave receive, master writing to slave 1b - Slave transmit, master reading from slave
1 IBIF	I2C bus interrupt flag The IBIF bit is set when one of the following conditions occurs: • Arbitration lost (IBAL bit set) • Byte transfer complete • Addressed as slave (IAAS bit set) • NoAck from Slave (MS & Tx bits set) • I^2C Bus going idle (IBB high-low transition and enabled by BIIE) A processor interrupt request will be caused if the IBIE bit is set. This bit must be cleared by software, by writing a one to it. A write of zero has no effect on this bit. All other conditions still apply.
0 RXAK	Received acknowledge This is the value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. This bit is valid only after transfer is complete. 0b - Acknowledge received 1b - No acknowledge received

15.5.2.6 I2C Bus Data I/O Register (IBDR)

Offset

Register	Offset
IBDR	4h

Function

In master transmit mode, when your software writes to IBDR, the I^2C module initiates a data transfer. The module sends the most significant bit first.

In master receive mode, when your software reads IBDR, the I^2C module initiates the reception of the next data byte.

In slave mode, the same functions are available after an address match has occurred.

NOTE

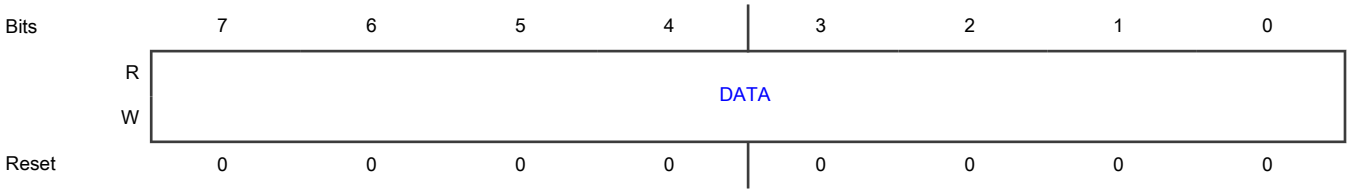
The IBCR[TXRX] field must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the I^2C is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

When the the I^2C module is configured for master-receive or slave-receive modes, and your software reads IBDR, the I^2C module returns the most-recent byte received. The IBDR does not reflect every byte transmitted on the I^2C bus. Your software also cannot verify that it wrote a byte to IBDR correctly by reading the byte back.

In master transmit mode, the I^2C module takes the first byte of data written to IBDR after assertion of MSSL and uses this byte for the address transfer. This byte and must comprise the calling address (in position DATA[7:1]) concatenated with the required R/W_b bit (in position D0). (In the I^2C bus specification, the R/W_b bit is presented as R/W with an overbar on the W.)

Access: Supervisor mode only (the module ignores user mode accesses and does not assert an error response if you access this register in user mode)

Diagram



Fields

Field	Function
7-0 DATA	DATA Data transmitted or received

15.5.2.7 I2C Bus Interrupt Config Register (IBIC)

Offset

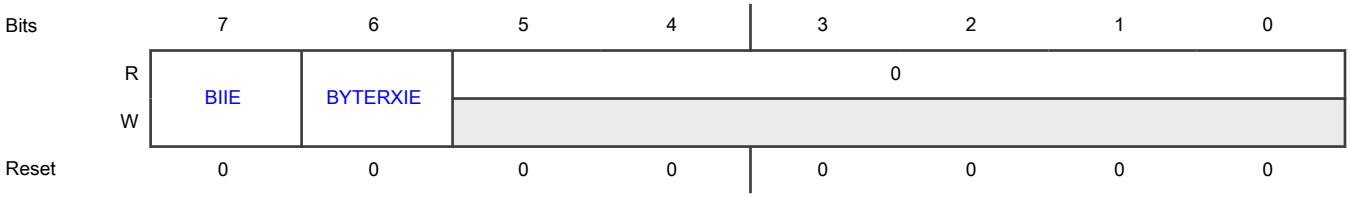
Register	Offset
IBIC	5h

Function

To program BIIE = 1, you must ensure that IBCR[MDIS] = 0.

Access: Supervisor mode only (the module ignores user mode accesses and does not assert an error response if you access this register in user mode)

Diagram



Fields

Field	Function
7 BIIE	Bus idle interrupt enable You can use this configuration field to enable the generation of an interrupt after the I^2C bus becomes idle. After BIIE=1, a high-to-low transition of IBSR[IBB] results in IBSR[IBIF]=1. You can use this feature to inform the CPU about the completion of a STOP on the I^2C bus. 0b - Bus idle interrupts disabled 1b - Bus idle interrupts enabled
6 BYTERXIE	Byte receive interrupt enable You can use this field to generate an interrupt every time the I^2C master/slave receives a new byte. This feature can be useful when an I^2C master receives data from a slave that transmits on an irregular basis. The I^2C module updates BYTERXIE only when the module is enabled (IBCR[MDIS]=0).
5-0 —	Reserved

15.5.2.8 I2C Bus Debug Register (IBDBG)

Offset

Register	Offset
IBDBG	6h

Function

Access: Supervisor mode only (the module ignores user mode accesses and does not assert an error response if you access this register in user mode)

Diagram

Bits	7	6	5	4	3	2	1	0
R	0				GLFLT_EN	0	IPG_DEB...	IPG_DEB...
W								
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7-4 —	Reserved
3 GLFLT_EN	Glitch filter enable On this chip, you must program this field to 1.
2 —	Reserved
1 IPG_DEBUG_HALTED	Debug halted This is a status field that you can be read back after asserting the debug enable signal. The field informs you whether the I^{2}C module entered the debug mode or not. 0b - The I2C module is still executing a transaction. 1b - The I2C module entered the debug mode.
0 IPG_DEBUG_EN	Debug enable This field allows the I^{2}C module to enter the debug mode when the IPG DEBUG signal is high. All the registers, counter values and status bits are frozen and can be accessed by the CPU. **NOTE** If the assertion of this bit along with the IPG DEBUG signal happens in the middle of a transaction, IP enters the debug mode only after successful completion of the current transaction after which no further transaction can take place until the debug mode is exited. 0b - Normal operation. The bus idle interrupts are disabled. 1b - The I2C module is in debug mode.

15.6 Functional description

This section presents the following topics:

- Notes about module operation
- Transactions

- Arbitration procedure
- Clock behavior
- Interrupts
- STOP mode
- DEBUG mode

15.6.1 Notes about module operation

- The I²C module always performs as a slave receiver by default, unless explicitly programmed to be a master or slave transmitter.
- When the I²C module is acting as a master, it must not try to call its own slave address.

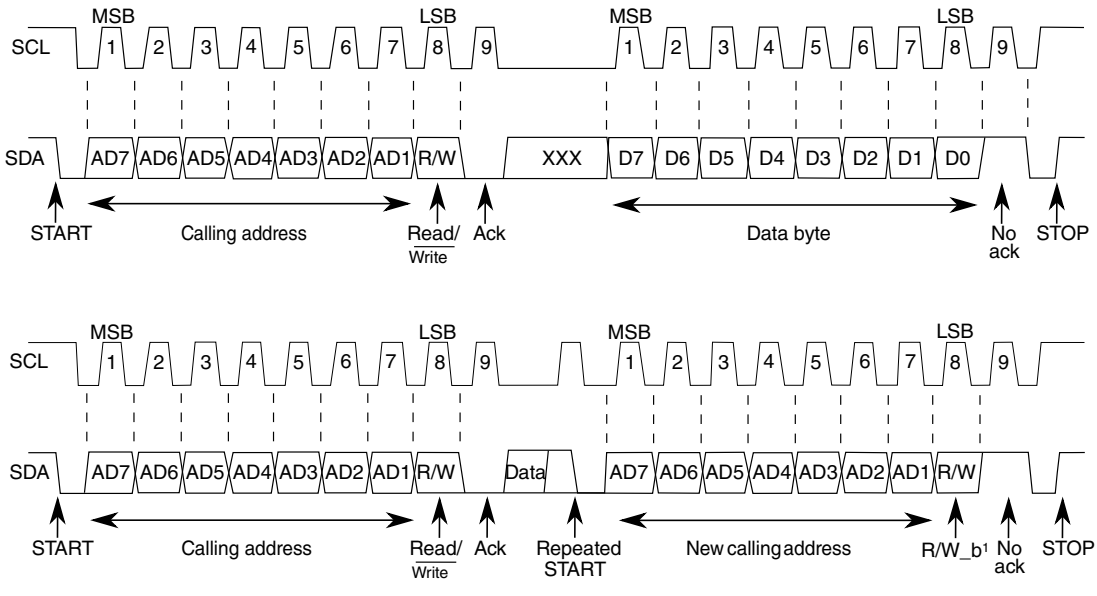
15.6.2 Transactions

This section presents the following topics:

- Protocol overview
- Transaction protocol definitions
- High-level protocol steps
- START condition
- Slave address transmission
- Data transmission
- STOP condition
- Repeated START condition

15.6.2.1 Protocol overview

The following figure shows the behavior of SCL and SDA during a typical I²C transaction.



[1] In the I²C bus specification, this bit is labeled R/$\overline{W}$.

Figure 27. I²C transaction protocol

15.6.2.2 Transaction protocol definitions

This section defines several important terms presented in Figure 27.

Table 47. I²C definitions

Term	Definition
START	A START condition, as defined in Definition: I²C conditions
STOP	A STOP condition, as defined in Definition: I²C conditions
Calling (slave) address	A seven-bit address used to identify a slave on the I²C bus. The requirements for specifying this address are presented in I²C calling address requirements
Read/write (R/W)	A bit that specifies the direction of the data transfer to the slave as follows: • 0=The data is being transferred from the master to the slave ("write") • 1=The data is being transferred from the slave to the master ("read")
Ack	A bit that specifies the acknowledgement of a calling address, indicated by pulling SDA low.

15.6.2.3 I²C calling address requirements

The calling addresses of the devices used on an I²C network are subject to the following requirements:

- Each slave must have a unique calling address.
- A master must not transmit a calling address that is the same as its own slave address.

15.6.2.4 High-level protocol steps

The I²C protocol conceptually supports two types of transfers, as illustrated in Figure 27. The significant steps in these transfers are presented in the following table. Details of each of these steps are presented in subsequent sections.

Table 48. I²C high-level protocol steps

Standard transfer	Repeated START transfer
1. START condition	1. START condition
2. Slave target or general call address transmission	2. Slave target or general call address transmission
3. Acknowledgment from slave	3. Acknowledgment from slave
4. Data transfer	4. Data transfer
5. STOP condition	5. Repeated START condition
6. (repeat Steps 1-4)	6. (repeat Steps 2-4 as needed)
	7. STOP condition
	8. (repeat Steps 1-7)

15.6.2.5 START condition

When the bus is free (no master device engages the bus; both SCL and SDA lines are at logical high), a master can initiate communication by sending a START condition (see Definition: I²C conditions). This signal denotes the beginning of a new data transfer (which may contain several bytes of data) and brings all slaves out of their idle states.

See Clock rate and IBFD settings and I2C Bus Frequency Divider Register (IBFD) for the associated timing requirements.

15.6.2.6 Slave address transmission

The master transmits the slave address on the next clock cycle after the START condition (see START condition). The following table presents the process of slave address transmission.

Table 49. Slave address transmission process

Step	Action
1	The master transmits the seven-bit slave address.
2	The master transmits the R/W bit.
3	Each slave examines the transmitted address and compares it to its own. If the addresses match, the slave device returns the acknowledge bit on the ninth SCL clock cycle.
4	The master waits for the acknowledge bit and determines the next step as follows: • The acknowledge bit is set: The master must initiate a data transfer followed by either a STOP condition or a repeated START condition. • The acknowledge bit is cleared: The master must wait for SCL to return to logic zero.

15.6.2.7 Data transmission

A data transfer session has the following characteristics:

- Can transmit one or more bytes of data
- Awakens all slaves
- Proceeds on a byte-by-byte basis in the direction specified by the R/W bit sent by the calling master

The transmitted data is subject to the following requirements:

- Each data byte must consist of 8 bits.
- Data bits can be changed only while SCL is low and must be held stable while SCL is high.
- One data bit is transmitted during one SCL clock pulse.
- The most significant bit (msb) must be transmitted first.
- Each data byte must be followed by an acknowledge bit on the ninth SCL clock pulse.

If the slave receiver does not acknowledge the master, the slave must leave the SDA line high. The master can then generate a STOP condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that as reaching the end of data. The slave then releases the SDA line for the master to generate a STOP or a START condition.

15.6.2.8 STOP condition

The master can terminate the communication by generating a STOP condition (see Definition: I^2C conditions). It can do so even if the slave has generated an acknowledge, at which point the slave must release the bus.

A master is not required to send a STOP condition at the end of every transfer. For more information, see Repeated START condition.

See Clock rate and IBFD settings and I2C Bus Frequency Divider Register (IBFD) for the associated timing requirements.

15.6.2.9 Repeated START condition

The I^{2}C protocol also supports a repeated START condition, which can be generated without a preceding STOP condition. A master device can use this condition to communicate with another slave or with the same slave in a different mode without releasing the bus. This condition is illustrated in the second timing diagram of Figure 27.

15.6.3 Arbitration procedure

The I^{2}C bus is a true multi-master bus that allows more than one master to be connected to the bus. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock. For this clock, the low period is equal to the longest clock low period and the high period is equal to the shortest period among the masters. A data arbitration procedure determines the relative priority of the contending masters. A bus master loses arbitration if it transmits logic "1" while another master transmits logic "0". The losing masters immediately switch over to slave mode and stop driving the SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, the I^{2}C module sets IBSR[IBAL] to indicate loss of arbitration.

15.6.4 Clock behavior

This section presents the following topics:

- Clock synchronization
- Clock stretching
- Handshaking
- Clock rate and IBFD settings

15.6.4.1 Clock synchronization

Due to the wired-AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached.

However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, the synchronized clock signal, SCL, is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see the following figure). When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

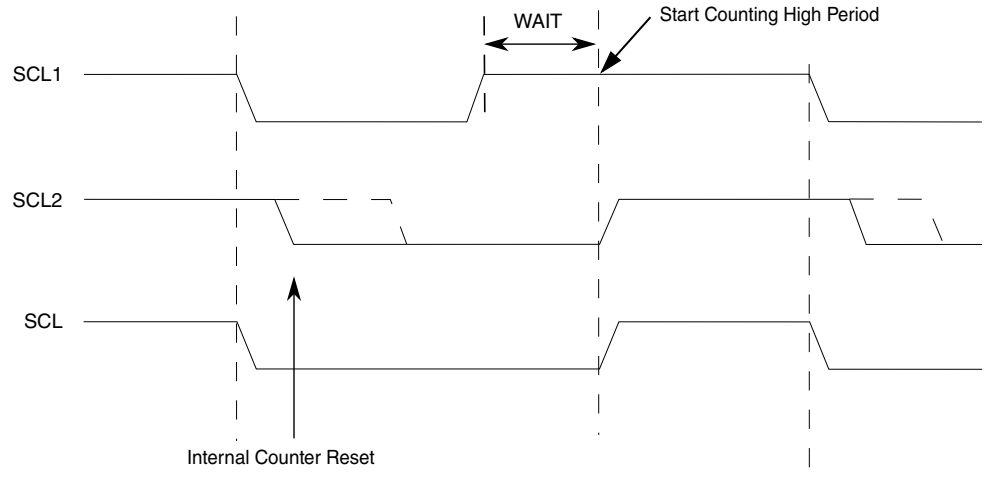


Figure 28. I^{2}C bus clock synchronization

15.6.4.2 Clock stretching

Slaves can use the clock synchronization mechanism to slow down the bit rate of a transfer. After the master drives SCL low, the slave can drive SCL low for the required period and then release it. If the slave's SCL low period is greater than the master's SCL low period, the resulting SCL bus signal low period is stretched.

15.6.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait state until the slave releases the SCL line.

15.6.4.4 Clock rate and IBFD settings

15.6.4.4.1 Timing definitions

Table 50. Timing definitions relevant to clock rate and IBFD settings

Term	Definition
SCL Divider	The factor used to prescale the CPU clock for bit rate selection (see Figure 29 and Table 52)
SCL period	(CPU clock period)×(SCL Divider)
SCL Hold	The required number of CPU clocks to generate a START or STOP condition (see Figure 29 and Table 52)
SDA Hold	See Figure 30 and Table 52

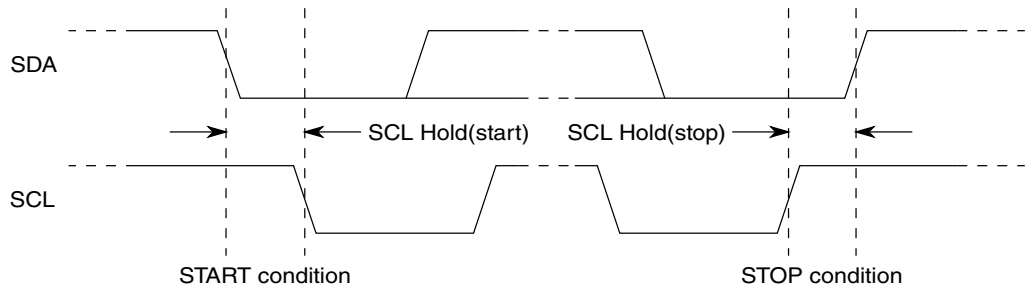


Figure 29. SCL Divider and SDA Hold

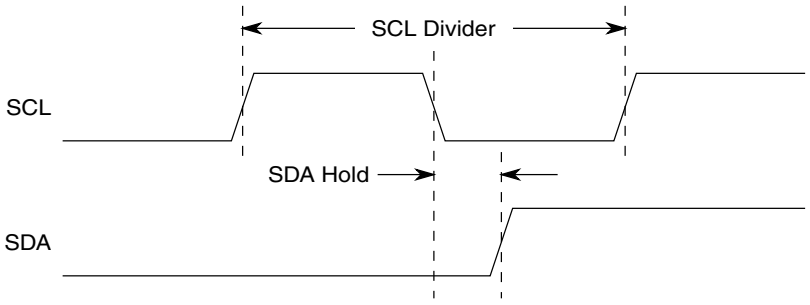


Figure 30. SDA Hold time

15.6.4.4.2 Divider and hold values

Table 52 provides the values for:

- SCL Divider
- SDA Hold
- SCL Hold (start)
- SCL Hold (stop)

when the glitch filter is disabled. (To disable this filter, write IBDBG[GLFLT_EN]=0.)

You have up to three MUL options available for all divider values, as shown in Table 51. Your choice of MUL determines the internal monitor rate of the I^2C bus (SCL and SDA signals):

- A lower MUL value results in a higher sampling rate of the I^2C signals.
- A higher MUL value results in a lower sampling rate of the I^2C signals. This gives the I^2C module greater immunity against glitches in the I^2C signals.

Table 51. MUL values as a function of IBC

When IBC is...	MUL is...
00h–3Fh	1
40h–7Fh	2
80h–BFh	4

NOTE

The SCL clock frequency obtained using the divider values listed in Table 52 and Table 53 is the nominal frequency. The actual frequency depends on the rise time at the SCL pad. The observed I^2C frequency may be lower. For operation at up to 400kbps and an SCL pad rise time of 100ns, the actual baud rate could be up to 15% lower.

Table 52. I^2C divider and hold values when glitch filter is disabled

IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
00	34	7	6	11
01	36	7	7	12
02	38	8	8	13
03	40	8	9	14
04	42	9	10	15
05	44	9	11	16
06	46	10	13	18
07	54	10	16	21
08	44	7	10	15

Table continues on the next page...

Table 52. I²C divider and hold values when glitch filter is disabled (continued)

IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
09	48	7	12	17
0A	52	9	14	19
0B	56	9	16	21
0C	60	11	18	23
0D	64	11	20	25
0E	72	13	24	29
0F	84	13	30	35
10	64	9	18	25
11	72	9	22	29
12	80	13	26	33
13	88	13	30	37
14	96	17	34	41
15	104	17	38	45
16	120	21	46	53
17	144	21	58	65
18	96	9	38	41
19	112	9	46	49
1A	128	17	54	57
1B	144	17	62	65
1C	160	25	70	73
1D	176	25	78	81
1E	208	33	94	97
1F	256	33	118	121
20	160	17	78	81
21	192	17	94	97

Table continues on the next page...

Table 52. I²C divider and hold values when glitch filter is disabled (continued)

IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
22	224	33	110	113
23	256	33	126	129
24	288	49	142	145
25	320	49	158	161
26	384	65	190	193
27	480	65	238	241
28	320	33	158	161
29	384	33	190	193
2A	448	65	222	225
2B	512	65	254	257
2C	576	97	286	289
2D	640	97	318	321
2E	768	129	382	385
2F	960	129	478	481
30	640	65	318	321
31	768	65	382	385
32	896	129	446	449
33	1024	129	510	513
34	1152	193	574	577
35	1280	193	638	641
36	1536	257	766	769
37	1920	257	958	961
38	1280	129	638	641
39	1536	129	766	769
3A	1792	257	894	897

Table continues on the next page...

Table 52. I²C divider and hold values when glitch filter is disabled (continued)

IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
3B	2048	257	1022	1025
3C	2304	385	1150	1153
3D	2560	385	1278	1281
3E	3072	513	1534	1537
3F	3840	513	1918	1921
40	68	14	12	22
41	72	14	14	24
42	76	16	16	26
43	80	16	18	28
44	84	18	20	30
45	88	18	22	32
46	96	20	26	36
47	108	20	32	42
48	88	14	20	30
49	96	14	24	34
4A	104	18	28	38
4B	112	18	32	42
4C	120	22	36	46
4D	128	22	40	50
4E	144	26	48	58
4F	168	26	60	70
50	128	18	36	50
51	144	18	44	58
52	160	26	52	66
53	176	26	60	74

Table continues on the next page...

Table 52. I²C divider and hold values when glitch filter is disabled (continued)

IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
54	192	34	68	82
55	208	34	76	90
56	240	42	92	106
57	288	42	116	130
58	192	18	76	82
59	224	18	92	98
5A	256	34	108	114
5B	288	34	124	130
5C	320	50	140	146
5D	356	50	156	162
5E	416	66	188	194
5F	512	66	236	242
60	320	34	156	162
61	384	34	188	194
62	448	66	220	226
63	512	66	252	258
64	576	98	284	290
65	640	98	316	322
66	768	130	380	386
67	960	130	476	482
68	640	66	316	322
69	768	66	380	386
6A	896	130	444	450
6B	1024	130	508	514
6C	1152	194	572	578

Table continues on the next page...

Table 52. I²C divider and hold values when glitch filter is disabled (continued)

IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
6D	1280	194	636	642
6E	1536	258	764	770
6F	1920	258	956	962
70	1280	130	636	642
71	1536	130	764	770
72	1792	258	892	898
73	2048	258	1020	1026
74	2304	386	1148	1154
75	2560	386	1276	1282
76	3072	514	1532	1538
77	3840	514	1916	1922
78	2560	258	1276	1282
79	3072	258	1532	1538
7A	3584	514	1788	1794
7B	4096	514	2044	2050
7C	4608	770	2300	2306
7D	5120	770	2556	2562
7E	6144	1026	3068	3074
7F	7680	1026	3836	3842
80	136	28	24	44
81	144	28	28	48
82	152	32	32	52
83	160	32	36	56
84	168	36	40	60
85	176	36	44	64

Table continues on the next page...

Table 52. I²C divider and hold values when glitch filter is disabled (continued)

IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
86	192	40	52	72
87	216	40	64	84
88	176	28	40	60
89	192	28	48	68
8A	208	36	56	76
8B	224	36	64	84
8C	240	44	72	92
8D	256	44	80	100
8E	288	52	96	116
8F	336	52	120	140
90	256	36	72	100
91	288	36	88	116
92	320	52	104	132
93	352	52	120	148
94	384	68	136	164
95	416	68	152	180
96	480	84	184	212
97	576	84	232	260
98	358	36	152	164
99	448	36	184	196
9A	512	68	216	228
9B	576	68	248	260
9C	640	100	280	292
9D	704	100	312	324
9E	832	132	376	388

Table continues on the next page...

Table 52. I²C divider and hold values when glitch filter is disabled (continued)

IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
9F	1024	132	472	484
A0	640	68	312	324
A1	768	68	376	388
A2	896	132	440	452
A3	1024	132	504	516
A4	1152	196	568	580
A5	1280	196	632	644
A6	1536	260	760	772
A7	1920	260	952	964
A8	1280	132	632	644
A9	1536	132	760	772
AA	1792	260	888	900
AB	2048	260	1016	1028
AC	2304	388	1144	1156
AD	2560	388	1272	1284
AE	3072	516	1528	1540
AF	3840	516	1912	1924
B0	2560	260	1272	1284
B1	3072	260	1528	1540
B2	3584	516	1784	1796
B3	4096	516	2040	2052
B4	4608	772	2296	2308
B5	5120	772	2552	2564
B6	6144	1028	3064	3076
B7	7680	1028	3832	3844

Table continues on the next page...

Table 52. I²C divider and hold values when glitch filter is disabled (continued)

IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
B8	5120	516	2552	2564
B9	6144	516	3064	3076
BA	7168	1028	3576	3588
BB	8192	1028	4088	4100
BC	9216	1540	4600	4612
BD	10240	1540	5112	5124
BE	12288	2052	6136	6148
BF	15360	2052	7672	7684

Table 53 provides the values for:

- SCL Divider
- SDA Hold
- SCL Hold (start)
- SCL Hold (stop)

when the glitch filter is enabled. (To enable this filter, write IBDBG[GLFLT_EN]=1.)

Table 53. I²C divider and hold values when glitch filter is enabled

IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
00	34	14	6	18
01	36	14	7	19
02	38	15	8	20
03	40	15	9	21
04	42	16	10	22
05	44	16	11	23
06	48	17	13	25
07	54	17	16	28
08	44	15	10	23
09	48	15	12	25
0A	52	17	14	27
0B	56	17	16	29

Table continues on the next page...

Table 53. I²C divider and hold values when glitch filter is enabled (continued)

0C	60	19	18	31
0D	64	19	20	33
0E	72	21	24	37
0F	84	21	30	43
10	64	17	18	33
11	72	17	22	37
12	80	21	26	41
13	88	21	30	45
14	96	25	34	49
15	104	25	38	53
16	120	29	46	61
17	144	29	58	73
18	96	17	38	49
19	112	17	46	57
1A	128	25	54	65
1B	144	25	62	73
1C	160	33	70	81
1D	176	33	78	89
1E	208	41	94	105
1F	256	41	11	129
20	160	17	78	81
21	192	17	94	97
22	224	33	110	113
23	256	33	126	129
24	288	49	142	145
25	320	49	158	161
26	384	65	190	193
27	480	65	238	241
28	320	33	158	161
29	384	33	190	193
2A	448	65	222	225
2B	512	65	254	257
2C	576	97	286	289

Table continues on the next page...

Table 53. I²C divider and hold values when glitch filter is enabled (continued)

2D	640	97	318	321
2E	768	129	382	385
2F	960	129	478	481
30	640	65	318	321
31	768	65	382	385
32	896	129	446	449
33	1024	129	510	513
34	1152	193	574	577
35	1280	193	638	641
36	1536	257	766	769
37	1920	257	958	961
38	1280	129	638	641
39	1536	129	766	769
3A	1792	257	894	897
3B	2048	257	1022	1025
3C	2304	385	1150	1153
3D	2560	385	1278	1281
3E	3072	513	1534	1537
3F	3840	513	1918	1921
40	68	28	12	36
41	72	28	14	38
42	76	30	16	40
43	80	30	18	42
44	84	32	20	44
45	88	32	22	46
46	96	34	26	50
47	108	34	32	56
48	88	30	20	46
49	96	30	24	50
4A	104	34	28	54
4B	112	34	32	58
4C	120	38	36	62
4D	128	38	40	66

Table continues on the next page...

Table 53. I²C divider and hold values when glitch filter is enabled (continued)

4E	144	42	48	74
4F	168	42	60	86
50	128	34	36	66
51	144	34	44	74
52	160	42	52	82
53	176	42	60	90
54	192	50	68	98
55	208	50	76	106
56	240	58	92	122
57	288	58	116	146
58	192	34	76	98
59	224	34	92	114
5A	256	50	108	130
5B	288	50	124	146
5C	320	66	140	162
5D	352	66	156	178
5E	416	82	188	210
5F	512	82	236	258
60	320	34	156	162
61	384	34	188	194
62	448	66	220	226
63	512	66	252	258
64	576	98	284	290
65	640	98	316	322
66	768	130	380	386
67	960	130	476	482
68	640	66	316	322
69	768	66	380	386
6A	896	130	444	450
6B	1024	130	508	514
6C	1152	194	572	578
6D	1280	194	636	642
6E	1536	258	764	770

Table continues on the next page...

Table 53. I²C divider and hold values when glitch filter is enabled (continued)

6F	1920	258	956	962
70	1280	130	636	642
71	1536	130	764	770
72	1792	258	892	898
73	2048	258	1020	1026
74	2304	386	1148	1154
75	2560	386	1276	1282
76	3072	514	1532	1538
77	3840	514	1916	1922
78	2560	258	1276	1282
79	3072	258	1532	1538
7A	3584	514	1788	1794
7B	4096	514	2044	2050
7C	4608	770	2300	2306
7D	5120	770	2556	2562
7E	6144	1026	3068	3074
7F	7680	1026	3836	3842
80	136	56	24	72
81	144	56	28	76
82	152	60	32	80
83	160	60	36	84
84	168	64	40	88
85	176	64	44	92
86	192	68	52	100
87	216	68	64	112
88	176	60	40	92
89	192	60	48	100
8A	208	68	56	108
8B	224	68	64	116
8C	240	76	72	124
8D	256	76	80	132
8E	288	84	96	148
8F	336	84	120	172

Table continues on the next page...

Table 53. I²C divider and hold values when glitch filter is enabled (continued)

90	256	68	72	132
91	288	68	88	148
92	320	84	104	164
93	352	84	120	180
94	384	100	136	196
95	416	100	152	212
96	480	116	184	244
97	576	116	232	292
98	384	68	152	196
99	448	68	184	228
9A	512	100	216	260
9B	576	100	248	292
9C	640	132	280	324
9D	704	132	312	356
9E	832	164	376	420
9F	1024	164	472	516
A0	640	68	312	324
A1	768	68	376	388
A2	896	132	440	452
A3	1024	132	504	516
A4	1152	196	568	580
A5	1280	196	632	644
A6	1536	260	760	772
A7	1920	260	952	964
A8	1280	132	632	644
A9	1536	132	760	772
AA	1792	260	888	900
AB	2048	260	1016	1028
AC	2304	388	1144	1156
AD	2560	388	1272	1284
AE	3072	516	1528	1540
AF	3840	516	1912	1924
B0	2560	260	1272	1284

Table continues on the next page...

Table 53. I²C divider and hold values when glitch filter is enabled (continued)

B1	3072	260	1528	1540
B2	3584	516	1784	1796
B3	4096	516	2040	2052
B4	4608	772	2296	2308
B5	5120	772	2552	2564
B6	6144	1028	3064	3076
B7	7680	1028	3832	3844
B8	5120	516	2552	2564
B9	6144	516	3064	3076
BA	7168	1028	3576	3588
BB	8192	1028	4088	4100
BC	9216	1540	4600	4612
BD	10240	1540	5112	5124
BE	12288	2052	6136	6148
BF	15360	2052	7672	7684

15.6.5 Interrupts

This section presents the following topics:

- Interrupt vector
- Interrupt description

15.6.5.1 Interrupt vector

The I²C module uses only one interrupt vector.

Table 54. Interrupt summary

Interrupt	Offset	Vector	Priority	Source	Description
I²C interrupt	—	—	—	IBAL, TCF, IAAS, IBB fields in IBSR	When any of IBAL, TCF or IAAS fields are 1, the I²C module may cause an interrupt based on Arbitration lost, Transfer Complete or Address Detect conditions. If enabled by BIIE, the de-assertion of IBB can also cause an interrupt, indicating that the bus is idle.

15.6.5.2 Interrupt description

There are five types of internal interrupts in the I²C. The interrupt service routine can determine the interrupt type by reading the Status register.

I²C Interrupt can be generated on the following events:

- Arbitration Lost condition (IBSR[IBAL]=1)
- Byte Transfer condition (IBSR[TCF]=1)

- Address Detect condition (IBSR[IAAS]=1)
- No Acknowledge from slave received when expected
- Bus idle (IBSR[IBB]=0)

The IBCR[IBIE] field enables the I^{2}C interrupt. To clear this field, program IBSR[IBIF]=1 in the interrupt service routine.

To use the Bus idle interrupt, you must enable it by using the IBIC[BIIE] field.

15.6.6 STOP mode

This mode allows the software to put the I^{2}C module in power-down state. Once the STOP request is asserted, the I^{2}C module comes to a graceful halt after completing all the ongoing transactions.

As soon as the I^{2}C module enters STOP mode:

- The I^{2}C clock is disabled.
- No transaction can take place.
- All registers are inaccessible.

The I^{2}C module enters this mode only after successfully completing the current ongoing transaction, hence the low-power request is acknowledged once the STOP condition occurs. You must ensure that the bus is free when STOP is requested. To request STOP for ongoing transmission, you must:

- Wait until the transmission is complete.
- Program IBCR[TXRX]=0.

See the figure below for more details.

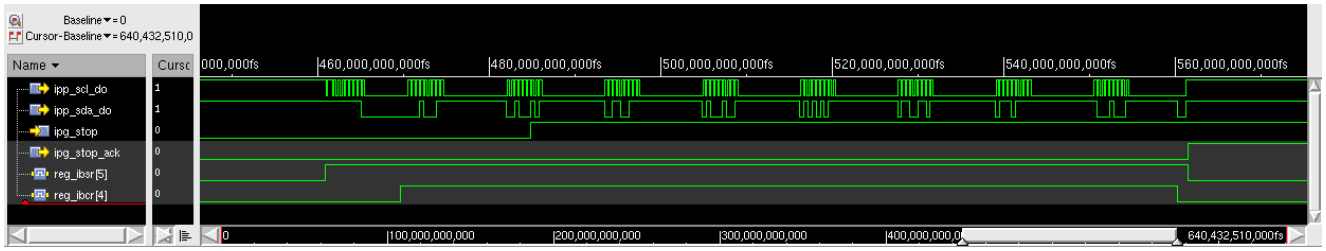


Figure 31. I^{2}C stop mode behavior when master is receiving and slave is transmitting

15.6.7 DEBUG mode

This mode allows the CPU to debug the I^{2}C module by freezing all the counters, registers and status bits. After the Debug request is asserted along with IBDBG[IPG_DEBUG_EN] field, the I^{2}C module comes to a graceful halt after completing all the ongoing transactions. A Debug halted signal IBDBG[IPG_DEBUG_HALTED] is also asserted to indicate that the debug request has been successfully serviced and is de-asserted when the Debug request is de-asserted.

As soon as the I^{2}C module enters DEBUG mode:

- No transaction can take place.
- All the registers, counters and status bits are frozen. They all can be accessed by the CPU to enable the debugging.

The I^{2}C module enters this mode only after successfully completing the current ongoing transaction. For example, if the current transaction consists of 8 bytes and you initiated DEBUG mode at the time of the second byte, the IPG Debug Halted signal will be asserted after the 8th byte is transmitted/received. See the simulation result in Figure 32 for more details.

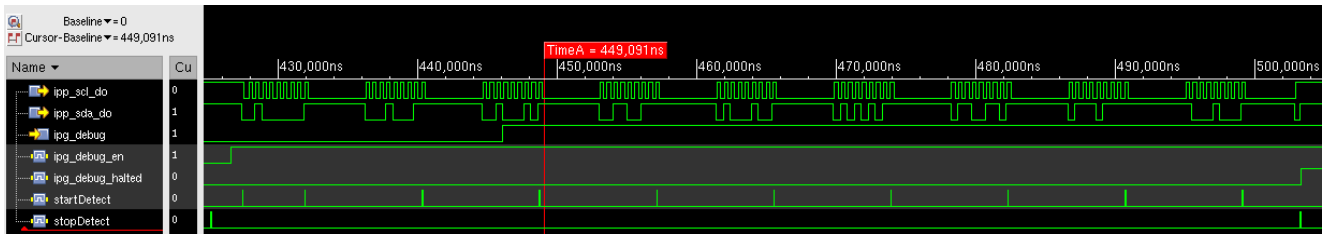


Figure 32. Simulation result of IPG Debug Halted in case of frame transmission

No more transaction can take place until the debug signal is de-asserted after which the I^2C module starts functioning normally. There is a status halted signal IBDBG[IPG_DEBUG_HALTED] to indicate to the user that the I^2C module has entered the DEBUG mode. See the following figure for more details.

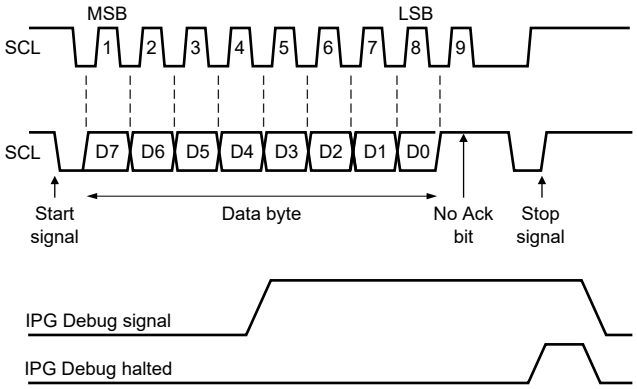
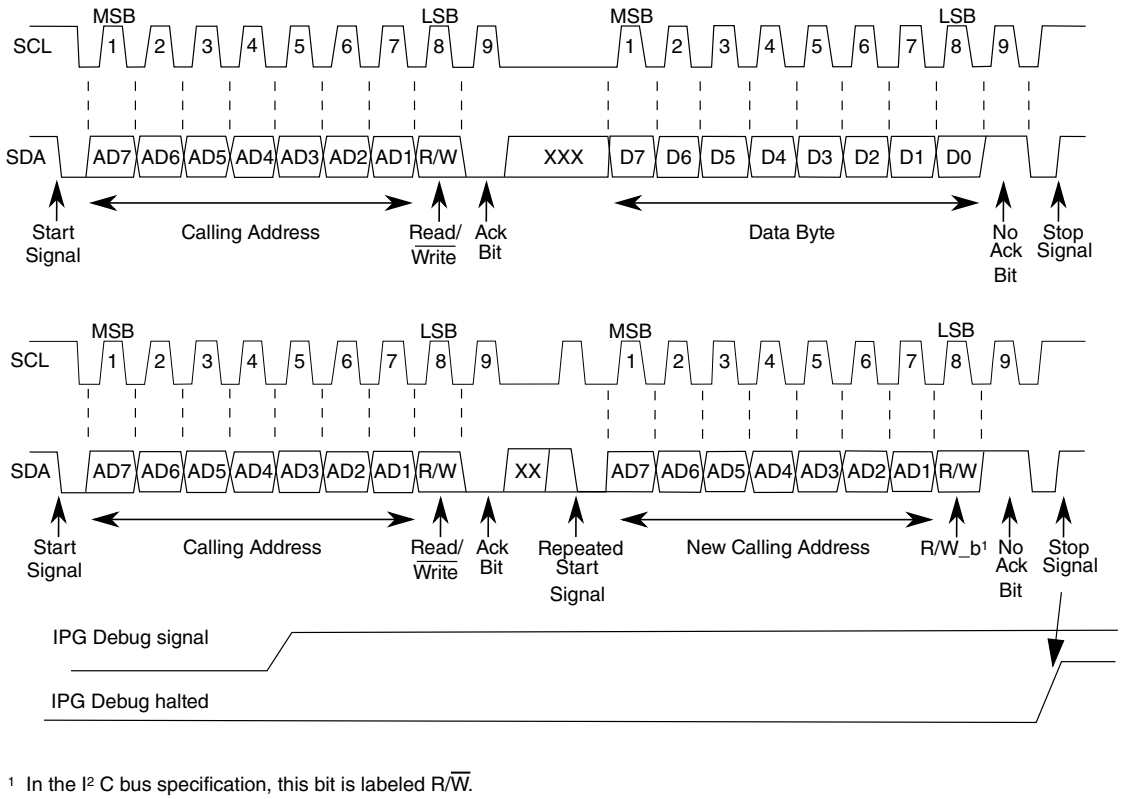


Figure 33. DEBUG mode

The following figure shows a case of DEBUG mode with repeated START transaction. In this case, if the debug signal is asserted in between the transaction, the entire transaction with multiple repeated start will be completed before the I^2C module enters DEBUG mode.



[1] In the I² C bus specification, this bit is labeled R/$\overline{W}$.

Figure 34. DEBUG mode with repeated start

15.7 Initialization/application information

This section presents the following topics:

- Recommended interrupt service flow
- General programming guidelines (for both master and slave mode)
- Programming guidelines specific to master mode
- Programming guidelines specific to slave mode

15.7.1 Recommended interrupt service flow

The following figure shows a flowchart for the recommended I^{2}C interrupt service routine. Deviation from the flowchart may result in unpredictable I^{2}C bus behavior.

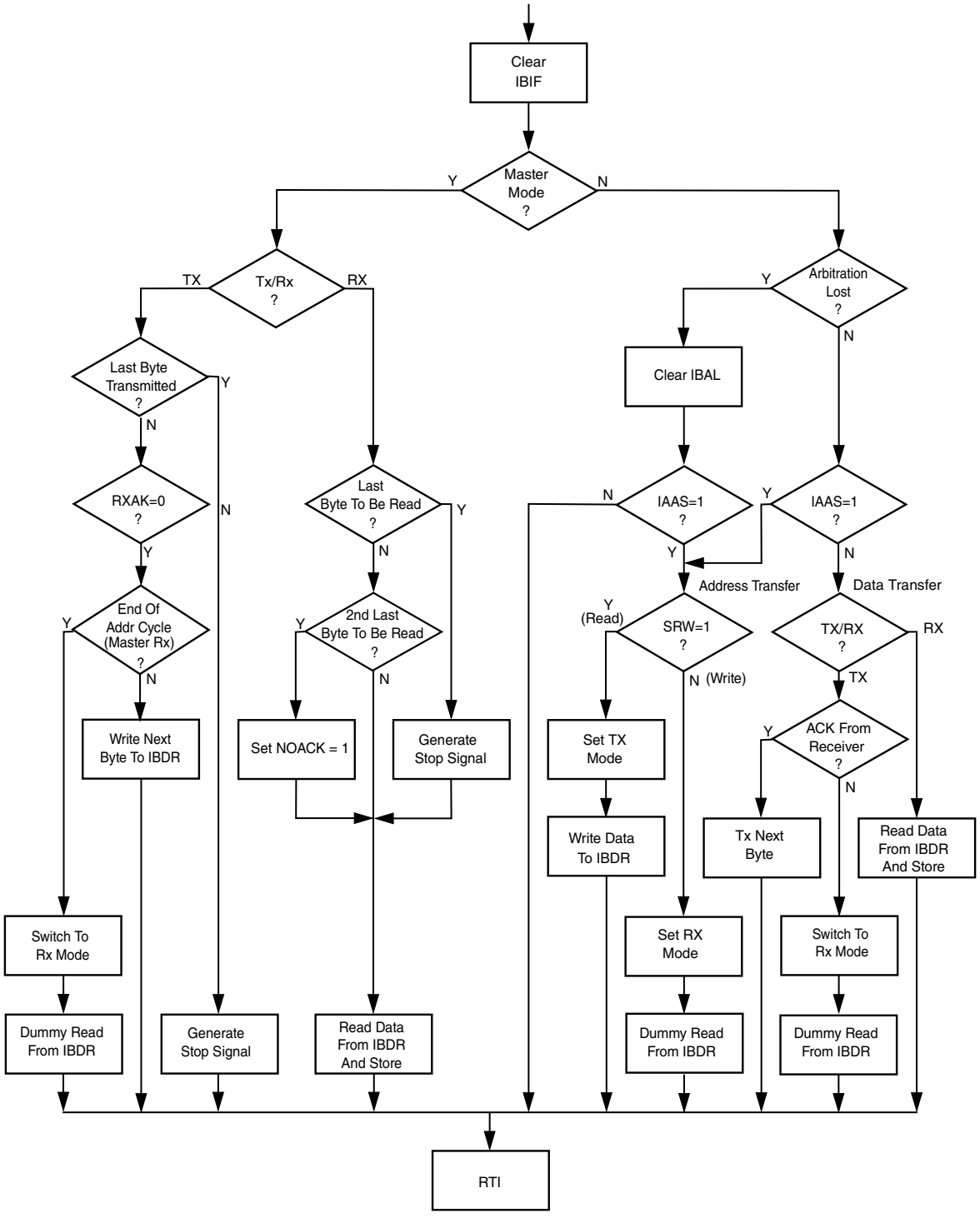


Figure 35. Recommended I²C interrupt service routine flowchart

15.7.2 General programming guidelines (for both master and slave mode)

This section provides programming guidelines recommended for the I²C module in both master and slave mode. It presents the following topics:

- Initializing the I²C module

- Software response after a transfer

15.7.2.1 Initializing the I²C module

The following table describes how to initialize the I²C module.

Table 55. I²C initialization procedure

Step	Action
1	Use I2C Bus Frequency Divider Register (IBFD) to select the required division ratio to obtain SCL frequency from the system clock.
2	Use I2C Bus Address Register (IBAD) to define the slave address.
3	Clear the MDIS field in I2C Bus Control Register (IBCR) to enable the I²C interface system.
4	Use I2C Bus Control Register (IBCR) to select Master/Slave mode, Transmit/Receive mode, and whether interrupts are enabled or disabled.
5	(Optional) Use I2C Bus Interrupt Config Register (IBIC) to further refine the interrupt behavior.

15.7.2.2 Software response after a transfer

Transmission or reception of a byte will result in IBSR[TCF]=1. This indicates one byte communication is finished. The interrupt flag, IBSR[IBIF], is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBCR[IBIE] field. IBSR[IBIF]The IBIF can be cleared by writing one (in the interrupt service routine, if interrupts are used).

IBSR[TCF] will be cleared to indicate data transfer in progress whenever data register is written to in transmit mode, or during reading out from data register in receive mode. Do not use IBSR[TCF] as a "data transfer complete" flag. This is because the flag timing depends on a number of factors, including the I²C bus frequency. This field may not conclusively provide an indication of a transfer-complete situation. It is recommended that transfer complete situations are detected using IBSR[IBIF].

Software may service the I²C I/O in the main program by monitoring IBSR[IBIF] if the interrupt function is disabled. Polling should monitor IBSR[IBIF] instead of IBSR[TCF] because their operation is different when arbitration is lost.

When a "Transfer Complete" interrupt occurs at the end of the address cycle, the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit sent with slave calling address, then the Tx/Rx bit at Master side should be toggled at this stage. If Master does not receive an ACK from Slave, then transmission must be re-initiated or terminated.

In slave mode, IBSR[IAAS] will get set if Slave address (I2C Bus Address Register (IBAD)) matches the Master calling address. This is an indication that Master-Slave data communication can now start. During address cycles (IBSR[IAAS]=1), IBSR[SRW] is read to determine the direction of the subsequent transfer and IBCR[TXRX] is programmed accordingly. For slave mode data cycles (IBSR[IAAS]=0), IBSR[SRW] is not valid. IBCR[TXRX] should be read to determine the direction of the current transfer.

15.7.3 Programming guidelines specific to master mode

This section presents the following topics:

- Generating START
- Transmit/receive sequence
- Generating STOP
- Generating repeated START
- Loss of arbitration

15.7.3.1 Generating START

After the I²C module completes its initialization procedure, the module can transmit serial data when your software selects the 'master transmitter' mode. If the device is connected to a multi-master bus system, your software must test the state of IBSR[IBB] to check whether the serial bus is free.

Your software can configure the I²C module to send the START condition and the first byte (the slave address) if the bus is free (IBSR[IBB]=0). The data that your software writes to the data register (I2C Bus Data I/O Register (IBDR)) comprises the slave calling address and the LSB. Your software configures the LSB to indicate the direction of transfer required from the slave.

The bus free time (the time between a STOP condition and the following START condition) is built into the I²C module that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I²C is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of the sequence of events which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
while (IBSR[IBB]==1)               // wait in loop for IBB flag to clear
IBCR[MS/MSL] and IBCR[]Tx/Rx] = 1  // master and transmit mode, that is,
                                   // generate start condition
IBDR = calling_address             // send the calling address to the data register
while (bit 5, IBSR ==0)            // wait in loop for IBB flag to be set
```

15.7.3.2 Transmit/receive sequence

The following tables present the sequences for:

- Master transmit
- Master receive
- Slave transmit
- Slave receive

Table 56. Master transmit sequence

Step	Action
a	Use I2C Bus Frequency Divider Register (IBFD) to select the required division ratio to obtain SCL frequency from Platform clock/2.
b	Write 0 to IBCR[MDIS] to enable the I²C interface system.
c	Use I2C Bus Control Register (IBCR) to select Master mode, Transmit mode, and interrupt enable.
d	Write 0 to the IBIF field in I2C Bus Status Register (IBSR).
e	Write data to I2C Bus Data I/O Register (IBDR).
f	Observe changes in the TCF field of I2C Bus Status Register (IBSR): • When IBSR[TCF] becomes 0, the transfer is in progress. • When IBSR[TCF] becomes 1, the transfer is complete.
g	Wait until the IBIF field in I2C Bus Status Register (IBSR) becomes 1.
h	Read the fields in I2C Bus Status Register (IBSR) to determine what happened: • If TCF = 1, the transfer completed.

Table continues on the next page...

Table 56. Master transmit sequence (continued)

Step	Action
	• If RXAK = 1, a No Acknowledge condition occurred. • If IBB = 0, the bus transitioned from Busy to Idle state. • If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **NOTE** You can ignore Address Detect (IAAS = 1) for master mode (it is valid only for slave mode).
i	Examine the RXAK field in I2C Bus Status Register (IBSR) for an acknowledgment from the slave.
j	Repeat steps d through i to transfer the next consecutive bytes of data.

Table 57. Master receive sequence

Step	Action
a	Follow steps a through i in Table 56 for address dispatch.
b	Write 0 to the IBIF field in I2C Bus Status Register (IBSR).
c	Write 0 to the TXRX field in I2C Bus Control Register (IBCR) to select Receive mode.
d	Perform a dummy read of I2C Bus Data I/O Register (IBDR) to initiate the receive operation.
e	Wait until the TCF field in I2C Bus Status Register (IBSR) becomes 1. (This proves that the transfer is complete.)
f	Wait until the IBIF field in I2C Bus Status Register (IBSR) becomes 1.
g	Read the fields in I2C Bus Status Register (IBSR) to determine what happened: • If TCF = 1, the reception completed. • If IBB = 0, the bus transitioned from Busy to Idle state. • If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **NOTE** You can ignore the No Acknowledge condition (RXAK = 1) for receive mode.
h	Read I2C Bus Data I/O Register (IBDR) to determine the data received from the slave.
i	Change IBSR[IBIF] and IBSR[TCF] to 0 by writing 1 to those fields.
j	Repeat steps e through i to read subsequent bytes.

Table 58. Slave transmit sequence

Step	Action
a	Use I2C Bus Address Register (IBAD) to define the slave address.
b	Write 0 to IBCR[MDIS] to enable the I^2C interface system.
c	Examine fields in I2C Bus Status Register (IBSR) as follows: • If IAAS = 1, examine IBSR[SRW]. • If IAAS = 1 and SRW = 1, write 1 to IBCR[TXRX] to select Transmit mode.

Table continues on the next page...

Table 58. Slave transmit sequence (continued)

Step	Action
d	Write data to I2C Bus Data I/O Register (IBDR).
e	Wait until the IBIF field in I2C Bus Status Register (IBSR) becomes 1.
f	Wait until the RXAK field in I2C Bus Status Register (IBSR) becomes 0.
g	Write 0 to the IBIF field in I2C Bus Status Register (IBSR).
h	Repeat steps d through g for the next consecutive data transfers.

Table 59. Slave receive sequence

Step	Action
a	Use I2C Bus Address Register (IBAD) to define the slave address.
b	Write 0 to IBCR[MDIS] to enable the I^2C interface system.
c	Examine fields in I2C Bus Status Register (IBSR) as follows: • If IAAS = 1, examine IBSR[SRW]. • If IAAS = 1 and SRW = 0, write 0 to IBCR[TXRX] to select Receive mode.
d	Write 0 to the IBIF field of I2C Bus Status Register (IBSR).
e	Perform a dummy read of I2C Bus Data I/O Register (IBDR) to initiate the receive operation.
f	Wait until the TCF field of I2C Bus Status Register (IBSR) becomes 1. (This proves that the transfer is complete.)
g	Wait until the IBIF field of I2C Bus Status Register (IBSR) becomes 1.
h	Read the fields in I2C Bus Status Register (IBSR) to determine what happened: • If TCF = 1, the reception completed. • If IBB = 0, the bus transitioned from Busy to Idle state. • If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **NOTE** You can ignore the No Acknowledge condition (RXAK = 1) for receive mode.
i	Read I2C Bus Data I/O Register (IBDR) to determine the data received from the master.
j	Change IBSR[IBIF] and IBSR[TCF] to 0 by writing 1 to those fields.
k	Repeat steps f through j to read subsequent bytes.

15.7.3.3 Generating STOP

A data transfer ends with a STOP condition generated by the master device. A master transmitter can simply generate a STOP condition after it transmits all the data. transmitted. The following is an example showing how a master transmitter generates a STOP condition.

```
if (tx_count == 0) or     // check to see if all data bytes have been transmitted
  (bit 0, IBSR == 1) {    // or if no ACK generated
    clear bit 5, IBCR     // generate STOP condition
  }
else {
```

```
 IBDR = data_to_transmit  // write byte of data to DATA register
   tx_count --            // decrement counter
 }                        // return from interrupt
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data. The master receiver can do this by programming IBCR[NOACK]=1 before reading the second-to-last byte of data. Before reading the last byte of data, the master receiver must first generate a STOP condition. The following is an example showing how a master receiver generates a STOP condition.

```
rx_count --            // decrease the rx counter
if (rx_count ==1)      // 2nd last byte to be read ?
 bit 3, IBCR = 1       // disable ACK
 if (rx_count == 0)    // last byte to be read ?
 bit 5, IBCR = 0       // generate STOP condition
else
data_received = IBDR   // read RX data and store
```

15.7.3.4 Generating repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
bit 2, IBCR = 1             // generate another start (restart)
IBDR == calling_address     // transmit the calling address
```

15.7.3.5 Loss of arbitration

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices that lost arbitration are immediately switched to slave mode by the I^2C module. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBSR[IBAL]=1 and IBCR[MSSL]=0. If one master attempts to start transmission, while the bus is being engaged by another master, the hardware:

- Inhibits the transmission
- Switches IBCR[MSSL] from 1 to 0 without generating a STOP condition
- Generates an interrupt to the CPU
- Programs IBSR[IBAL]=1 to indicate that the attempt to engage the bus is failed
- Does not program IBSR[TCF]=1 due to the loss of data during arbitration

When considering these cases, your slave service routine should test IBSR[IBAL] first and your software should program IBSR[IBAL]=0 if IBSR[IBAL]=1.

15.7.4 Programming guidelines specific to slave mode

In the slave interrupt service routine, IBSR[IAAS] should be tested to check if a calling of its own address has just been received. If IBSR[IAAS]=1, software should program IBCR[TXRX]=1 according to IBSR[SRW]. Writing to the IBCR clears IBSR[IAAS] automatically. The only time you will read a value of 1 from IBSR[IAAS] is from the interrupt at the end of the address cycle where an address match occurred. Interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR for slave transmits or dummy reading from IBDR in slave receive mode. The slave will drive SCL low between byte transfers. SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, IBSR[RXAK] must be tested before transmitting the next byte of data. Setting RXAK means an "end of data" signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

Chapter 16
Lightweight LVDS Communication Protocol (LLCP)

16.1 Introduction

The LLCP is a lightweight low-voltage differential signaling (LVDS) communication protocol block. The LLCP is responsible for communication to the RFIC via a serial data and strobe interface. APB accesses will be converted to transactions that will support 8b/10b encoding across the serial interface.

16.1.1 Overview

The SoC configuration block diagram of LLCP is shown below as would be instantiated within a system. In this configuration during functional mode the LLCP is clocked at 614.4 MHz and data is transmitted at 614.4 Msps on the LLCP_DOUT/STOUT pins. Only a small portion of the design detailed below in the Data Strobe Decoder block is clocked at a different frequency, the recovered clock. Data is sampled at 153.6 MHz on a dual data rate in order to provide the 307.2 Msps required. An asynchronous FIFO is used for the boundary crossing.

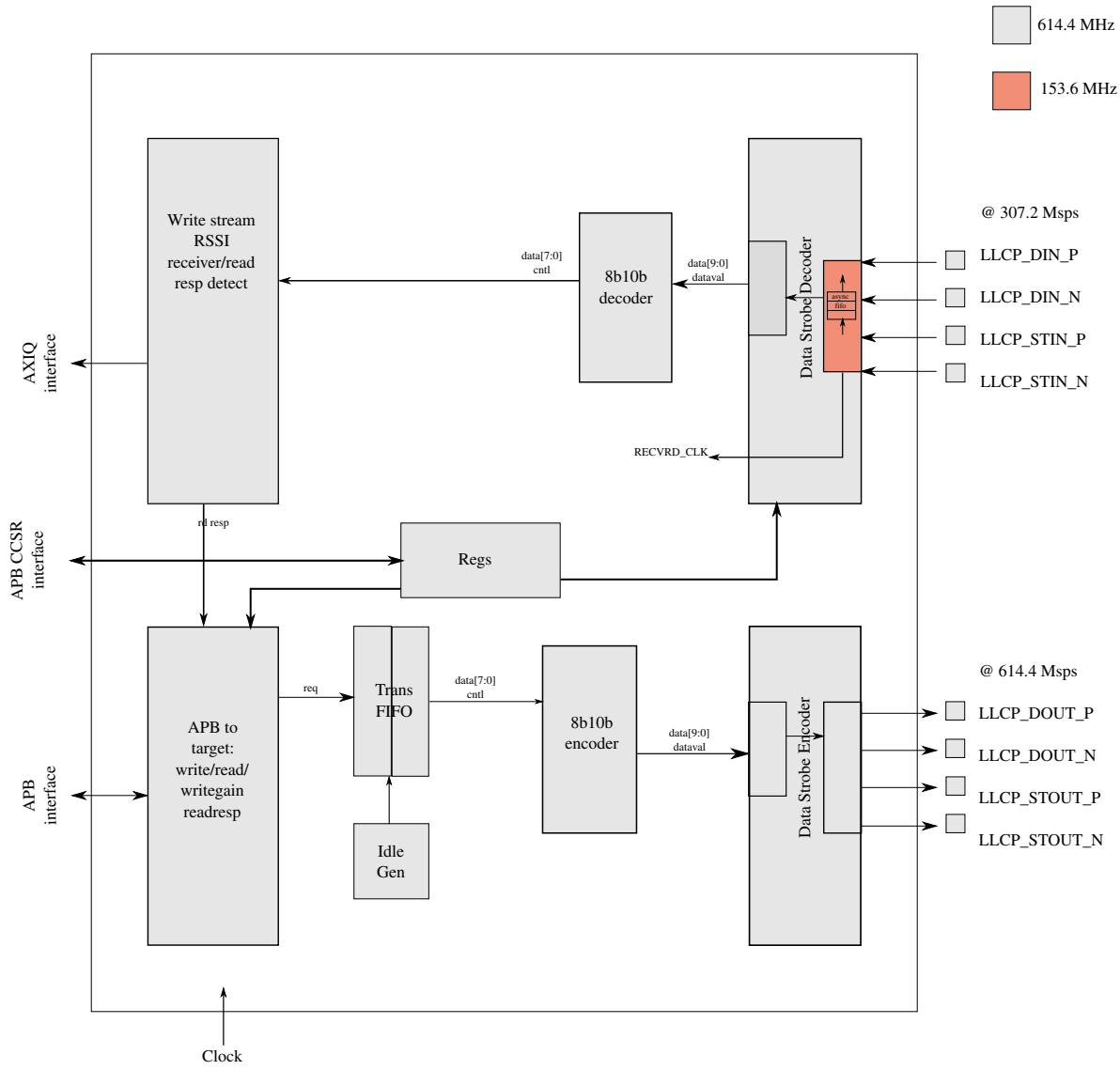


Figure 36. LLCP SoC Configuration Block Diagram

16.1.2 Features

The lightweight LVDS communication protocol block (LLCP) has the following features:

- Maintains legacy I2C support for legacy RFICs operating up to 1 MHz.
- Simple clock recovery from data strobe encoding.
- Supports generation of writes, reads, or write gain requests while in a SoC configuration.
- Supports reception of RSSI data stream writes or read responses while in a SoC configuration.
- Support LVDS transmission up to 614.4 Msps.
- Support LVDS reception of up to 307.2 Msps

16.1.3 Modes of Operation

LLCP has the following modes of operation.

- Functional Mode
  - Indicates normal functional mode.
- Repeater Mode
  - Special internal loop back mode for testing purposes. This is achieved by having software LLCPCSR[RPTR]==1.

16.2 External Signal Description

The following pins will be instantiated externally to the LLCP block.

Table 60. External Signals

Signal Name	Description	I/O
LLCP_DOUT_P	Data out p	O
LLCP_DOUT_N	Data out n	O
LLCP_STOUT_P	Strobe out p	O
LLCP_STOUT_N	Strobe out n	O
LLCP_DIN_P	Data in p	I
LLCP_DIN_N	Data in n	I
LLCP_STIN_P	Strobe in p	I
LLCP_STIN_N	Strobe in n	I

16.3 LLCP register descriptions

The table shows the memory map for management of the LLCP resources.

16.3.1 LLCP Memory map

LLCP base address: 22E_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	LLCP Control and Status Register 0 (LLCPCSR)	32	RW	0000_0000h
40h	LLCP Request Timeout 0 (LLCPTO)	32	RW	0000_0A00h
60h	LLCP Delay Register 0 (LLCPDLY)	32	RW	0000_0000h
100h	LLCP IO Control Register 0 (LLCPIOCR)	32	RW	0001_0404h
140h	LLCP RSSI Generator Debug Register 0 (LLCPRSDBG)	32	RW	0000_0000h

16.3.2 LLCP Control and Status Register 0 (LLCPCSR)

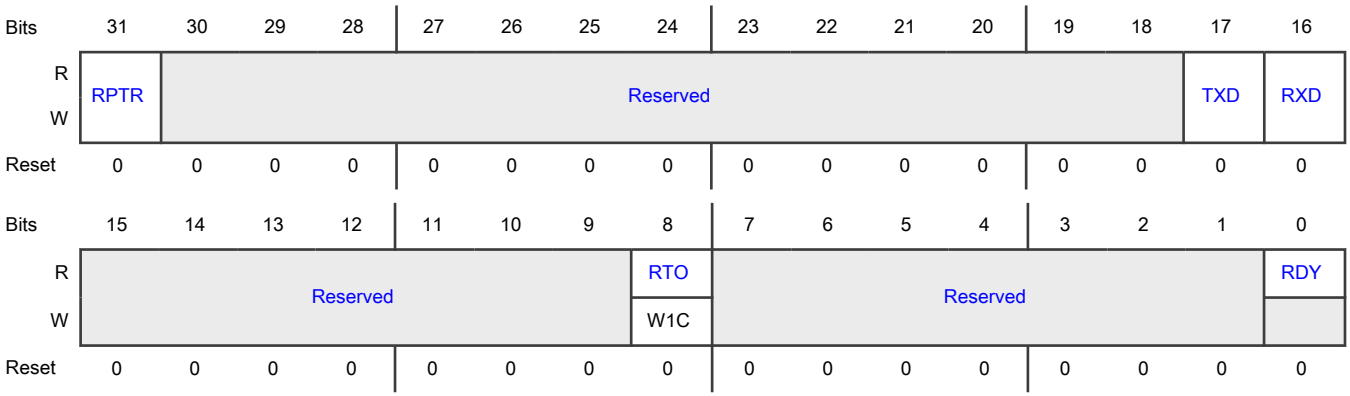
Offset

Register	Offset
LLCPCSR	0h

Function

Control and status bits for LLCP.

Diagram



Fields

Field	Function
31 RPTR	LLCP Repeater Mode Enable 'h0 - Repeater mode not enabled. 'h1 - Repeater mode enabled.
30-18 —	Reserved.
17 TXD	LLCP TX Disable. When set, all baseband SoC outbound requests will not be transmitted externally. Writes will be ignored and reads will return zeros. 'h0 - TX enabled and transmitting requests normally. 'h1 - TX disabled and not transmitting requests.
16 RXD	LLCP RX Disable. When set, all incoming baseband SoC requests will be ignored. 'h0 - RX enabled and processing requests normally. 'h1 - RX disabled and not processing requests.
15-9	Reserved.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
8 RTO	LLCP Request Timeout Detected. A baseband SoC read request to the LLCP has not received a response before the programmed timeout value. This bit is set by hardware and software must write a one to clear this bit. 'h0 - No timeout has been detected. 'h1 - Request timeout encountered.
7-1 —	Reserved.
0 RDY	LLCP Ready Signal. Please see the section on IDLE Generator for more information. 'h0 - Not ready to transmit. 'h1 - Ready to transmit.

16.3.3 LLCP Request Timeout 0 (LLCPTO)

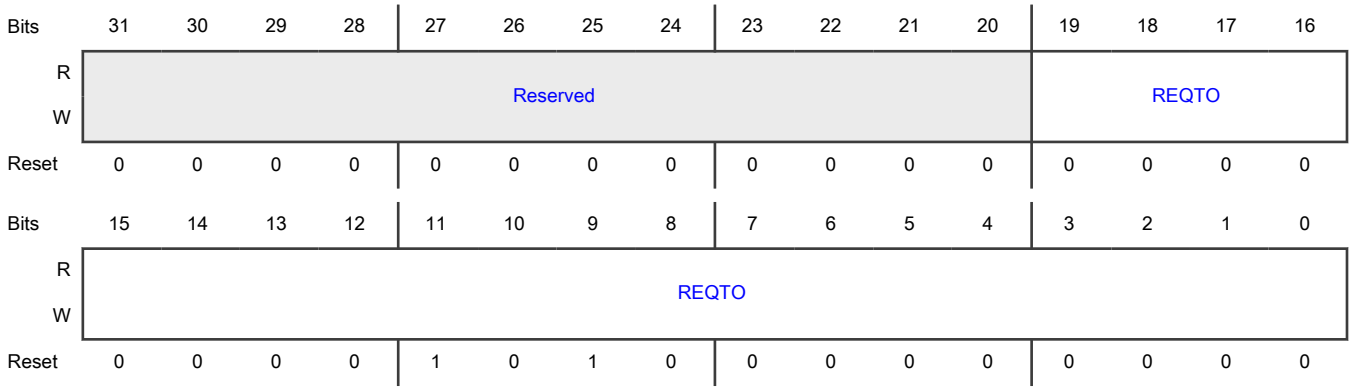
Offset

Register	Offset
LLCPTO	40h

Function

Represents the 20 bit request timeout used for read accesses. This register contains a programmable value that represents the number of 614.4 MHz clock cycles that the LLCP will wait before asserting a timeout condition, as indicated by the LLCPCSR[RTO] bit. Software can then clear this bit by writing a "1" to that field. Default is 0xA00, which represents an approximate 4us timeout. A maximum value of 0xFFFFF represents an approximate 2ms timeout.

Diagram



Fields

Field	Function
31-20 —	Reserved.
19-0 REQTO	Request Timeout Field represents bits 19-0 of the request timeout. Each programmable value represents a number of 614.4 MHz clock cycles.

16.3.4 LLCP Delay Register 0 (LLCPDLY)

Offset

Register	Offset
LLCPDLY	60h

Function

This register is used to configure the gate-level compensation delay of the input/output pins. Software can program a delay of up to approximately 100ps to control skew detween data and strobe inputs and outputs. Default is no buffer delay added. Please see the section on Compensation Delay for more details.

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				SIDLY				DIDLY				SODLY			DODLY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	Reserved.
11-9 SIDLY	LLCP_STIN pin delay select. 'h0 - No delay added.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	'h1 - One buffer delay added. '. . . 'h7 - Seven buffer delay added.
8-6 DIDLY	LLCP_DIN pin delay select. 'h0 - No delay added. 'h1 - One buffer delay added. '. . . 'h7 - Seven buffer delay added.
5-3 SODLY	LLCP_STOUT pin delay select. 'h0 - No delay added. 'h1 - One buffer delay added. '. . . 'h7 - Seven buffer delay added.
2-0 DODLY	LLCP_DOUT pin delay select. 'h0 - No delay added. 'h1 - One buffer delay added. '. . . 'h7 - Seven buffer delay added.

16.3.5 LLCP IO Control Register 0 (LLCPIOCR)

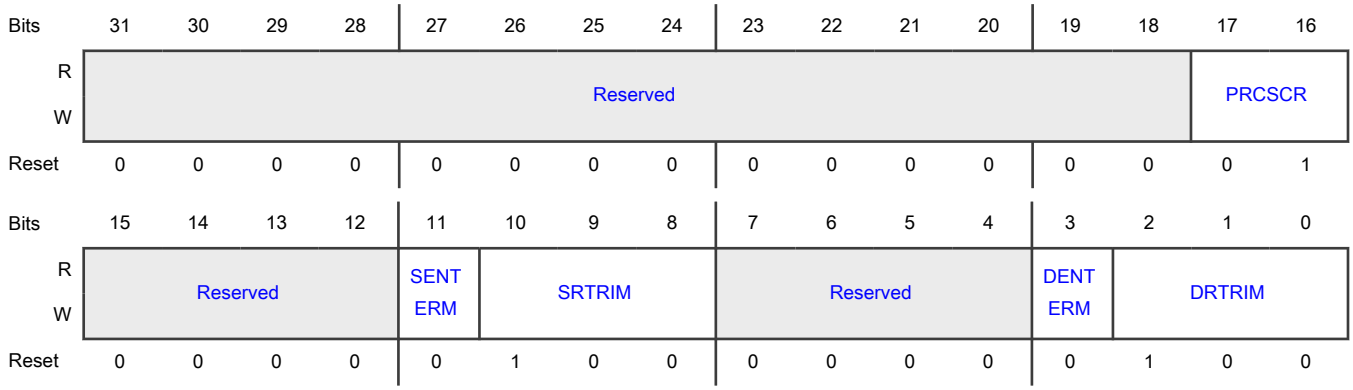
Offset

Register	Offset
LLCPIOCR	100h

Function

The LLCP contains IO pins instantiated within the block wrapper. Various settings are driven by fuse configurations and may be edited to override the default configurations as needed.

Diagram



Fields

Field	Function
31-18 —	Reserved.
17-16 PRCSCR	2-bit Bandgap setting to adjust the output currents for process deviation. 'b00: adjust bandgap current for LVDS transmitter for ff process. 'b01 or 'b10: adjust bandgap current for LVDS transmitter for typical process. 'b11: adjust bandgap current for LVDS transmitter for ss process.
15-12 —	Reserved.
11 SENTERM	STIN RX pin ENTERM setting. 1-bit control to enable internal 100 Ohm termination resistor. 'b0: Disabled. 'b1: Enabled.
10-8 SRTRIM	STIN RX pin RTRIM setting. 3-bit control to compensate internal termination resistor value deviation in process corners. 'b000: Setting for BCS process corner. 'b100: Setting for TYP process corner. 'b111: Setting for WCS process corner.
7-4 —	Reserved.
3 DENTERM	DIN RX pin ENTERM setting. 1-bit control to enable internal 100 Ohm termination resistor. 'b0: Disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	'b1: Enabled.
2-0 DRTRIM	DIN RX pin RTRIM setting. 3-bit control to compensate internal termination resistor value deviation in process corners. 'b000: Setting for BCS process corner. 'b100: Setting for TYP process corner. 'b111: Setting for WCS process corner.

16.3.6 LLCP RSSI Generator Debug Register 0 (LLCPRSDBG)

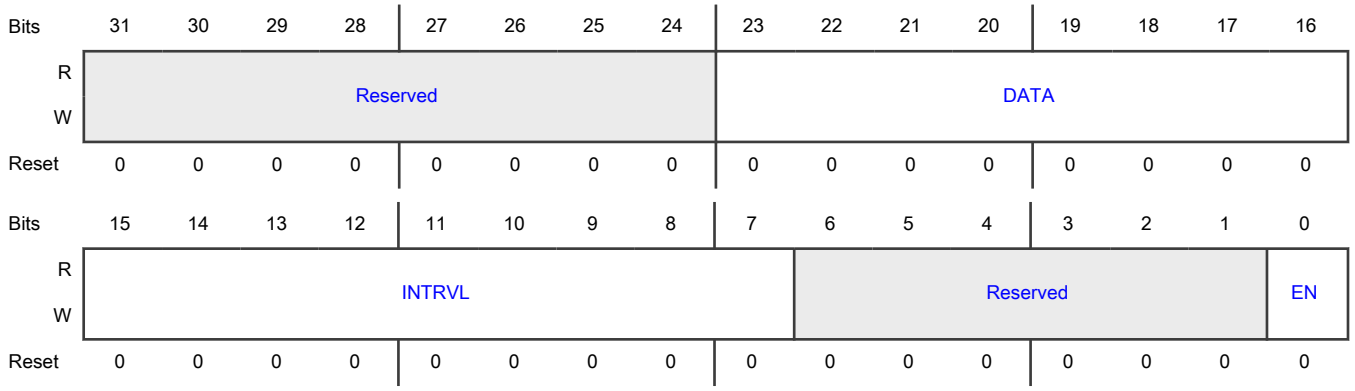
Offset

Register	Offset
LLCPRSDBG	140h

Function

Software controlled RSSI Generator. When enabled, software must program the RSSI Data that will be sent every interval cycles to the AXIQ interface.

Diagram



Fields

Field	Function
31-24 —	Reserved.
23-16	Data

Table continues on the next page...

Table continued from the previous page...

Field	Function
DATA	8-bits of data programmed to be sent across AXIQ interface. Software must ensure updates to this field occur before next interval will occur, if desired.
15-7 INTRVL	Interval Programmed value represents interval for RSSI Generator. A valid byte of data, DATA[7:0], will be transmitted across the AXIQ interface once every (INTRVL[8:0] + 1) clock cycles. This will yield data rates of about 1 to 614.4 MB/s. A programmed value of "0" translates to sending one byte of data every clock cycle. [~614.4 MB/s] A programmed value of "1" translates to sending one byte of data every two clock cycles. [~307.2 MB/s] A programmed value of "2" translates to sending one byte of data every three clock cycles. [~204.8 MB/s]
6-1 —	Reserved.
0 EN	Enable RSSI Generator is enabled, all incoming RSSI data is ignored. 'b0: Disabled. 'b1: Enabled.

16.4 Functional Description

16.4.1 IDLE Generator

If there are no packets being transmitted both SoC and RFIC configurations of the LLCP will ensure a random sequence of IDLE control characters are being transmitted across the LLCP and the corresponding LVDS Data and Strobe pins. The IDLE generator is in charge of maintaining the proper insertion of random K,A,R control symbols over the link. It ensures pseudo-randomization and proper alignment.

The LLCP port will assert the LLCPCSR[RDY] signal when it has achieved character boundary synchronization. The Serial RapidIO comma detect logic watches for four consecutive commas on the same alignment to create "char_synched", an indication that a good character boundary has been found. After this, if four consecutive commas have a different alignment, then char_synched will be de-asserted, and a new character synchronization must begin.

Software must not send any APB transactions until the LLCPCSR[RDY] bit has been asserted.

16.4.2 External Addressing

The LLCP has the ability to address off chip memory located on the RFIC. Since the LLCP has the concept of registers being indivisible 32 bit quantities accessed with a byte address and the RFIC has the concept of registers being indivisible 16 bit quantities accessed with a register address there needs to be some alignment. The SoC will utilize the APB address to determine which half word is being accessed. The RFIC memory map consists of RFIC registers and firmware memory. Refer to the Figure 37 for a physical diagram.

The RFIC will setup regions to represent the RFIC registers and the firmware memory space. An access will return the appropriate data based on it's configuration in the memory. Software must ensure addresses are aligned when issuing 16bit accesses to RFIC memory.

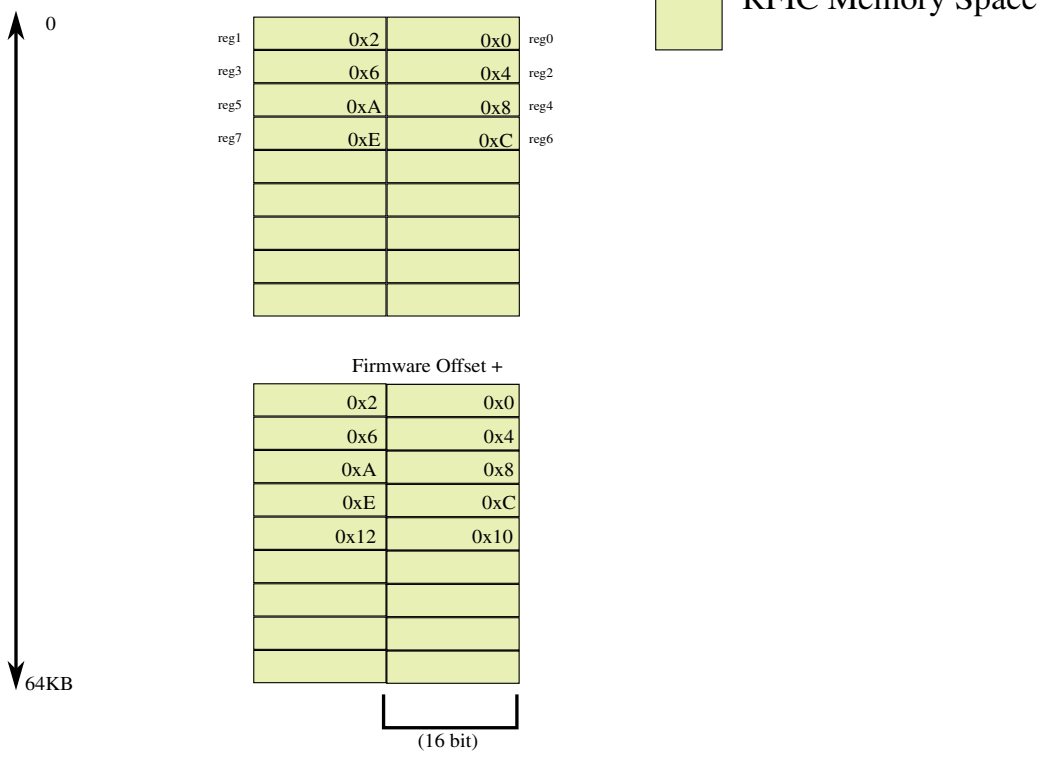


Figure 37. RFIC Memory Map Diagram

16.4.3 Transaction Types

Hardware supports sending and receiving the following transaction types across the interface. The read/write transactions will be initiated by a functional APB bus within the SoC. The following table details the APB to transaction mapping.

Table 61. Transaction Mapping

Request	Condition	LLCP Transaction
APB READ	-	READ
APB WRITE	APB ADDR == LLCPGA[GADDR]	WRITE GAIN
APB WRITE	APB ADDR != LLCP[GADDR]	WRITE
APB READ RSP	-	READ RESP
AXIQ WRITE	-	WRITE STREAM Divisible data is sent straight to the AXIQ 8bit interface.

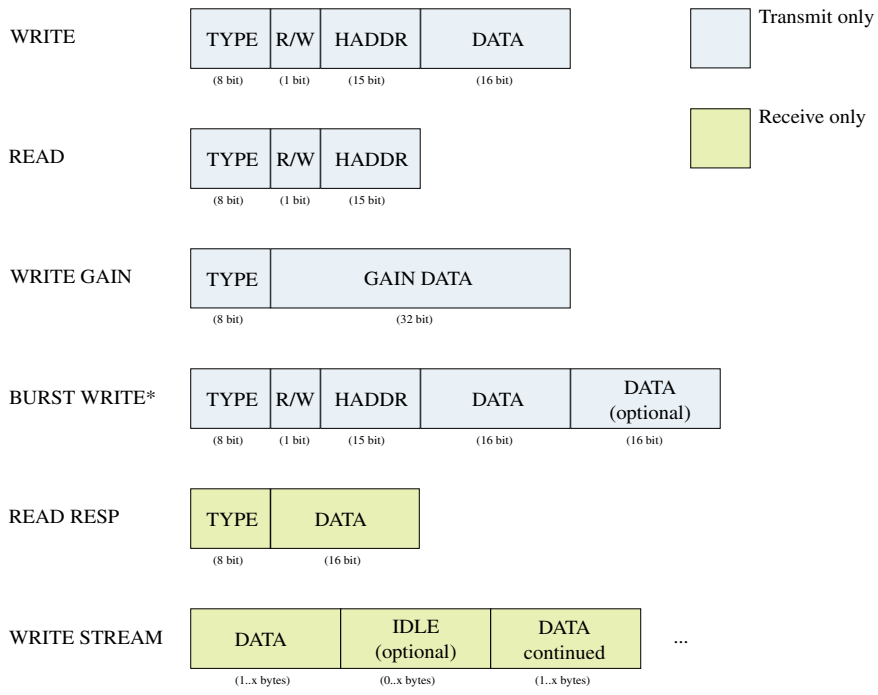


Figure 38. Transactions

*The burst write transaction is not currently implemented.

Table 62. Transaction Fields

Field	Bit width	Description
TYPE	8	Transaction type. RapidIO characters used here. 'h7C (K28.3) "PD" character: Indicates Read/Write during transmit or Read Response during receive. 'h3C (K28.1) "M" character: Indicates Write Gain during transmit.
R/W	1	Read or write field. 0 = Read 1 = Write
HADDR	15	Half-word Address. Indicates read/write request address to RFIC memory space. Used to access either registers or firmware memory space. RFIC target address is calculated by adding 1 bit of "0" to the HALF WORD address. 'h0000 = Target address 0000. 'h0001 = Target address 0002. 'h0002 = Target address 0004. 'h0003 = Target address 0006. ... 'h7fff = Target address FFFE.
GAIN DATA	32	32 bit gain data structure.

Table continues on the next page...

Table 62. Transaction Fields (continued)

Field	Bit width	Description
		[31:24] = Channel *c1* data. [23:16] = Channel *c2* data. [15: 8] = Channel *c3* data. [7: 0] = Channel *c4* data. Each byte represents separate data for each channel.
DATA	16	16 bit data associated with the write/read access. (Refer to External Addressing for more details).
IDLE	8	Represented by RapidIO Idle Sequence characters. 'hBC (K28.5) "K" character. 'hFB (K27.7) "A" character. 'hFD (K29.7) "R" character.

16.4.4 8b/10b Encoding

The LLCP transmits data over LVDS using a data and strobe encoding. A clock is recovered and used to capture data across the LLCP_DOUT pins. Data to be transmitted is 8b/10b encoded prior to transmission as per the Serial RapidIO definition. Please examine the RapidIO Interconnect Specification Part 6: 1x/4x LP-Serial Physical Layer Specification for more information.

The resulting encoded 10-bit data is transmitted LSB bit first across the data and strobe interface at 614.4 MHz (614.4 Msps). The receive side will recover a clock running at 307.2 MHz and; sample on each positive and negative edge to maintain the 614.4 Msps data rate. The following figure provides a high level diagram of the transmission and reception of data.

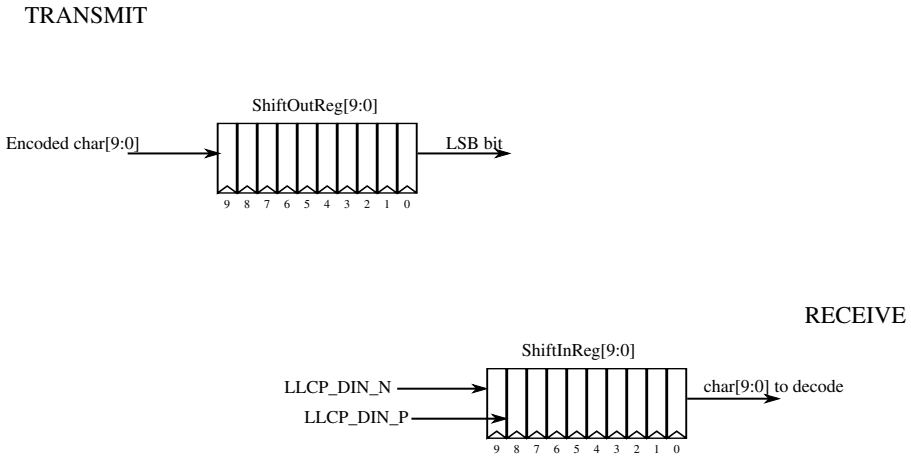


Figure 39. Data transmit and receive

16.4.5 Data Strobe Encoding

Data-strobe (DS) encoding encodes the transmission clock with the data into the data and strobe, thus allowing the clock to be recovered by xor-ing the data and strobe lines together. The strobe signal toggles if and only if the data signal does not change. The SoC configuration of LLCP transmits data launched at a 614.4 MHz clock frequency (614.4 Msps data rate). The resulting recovered clock will be a 307.2 MHz clock frequency, and in order to ensure the same data rate of 614.4 Msps, data will be

sampled at the positive and negative edges of the recovered clock. The figure below provides a simple diagram of the clock recovery structure.

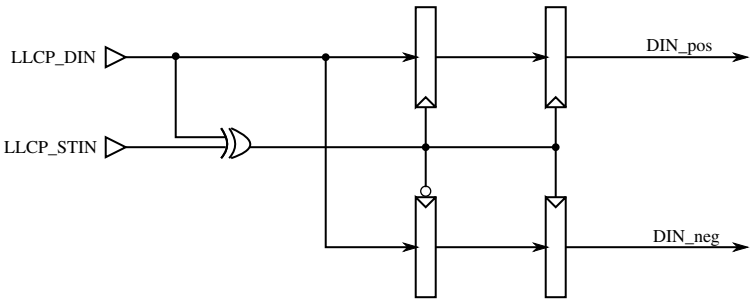


Figure 40. Recovered Clock Structure

In the example timing diagram below, the strobe signal toggles if and only when the current data is the same as the previous data value and does not change. The xor recovered clock will be 307.2 MHz in this example.

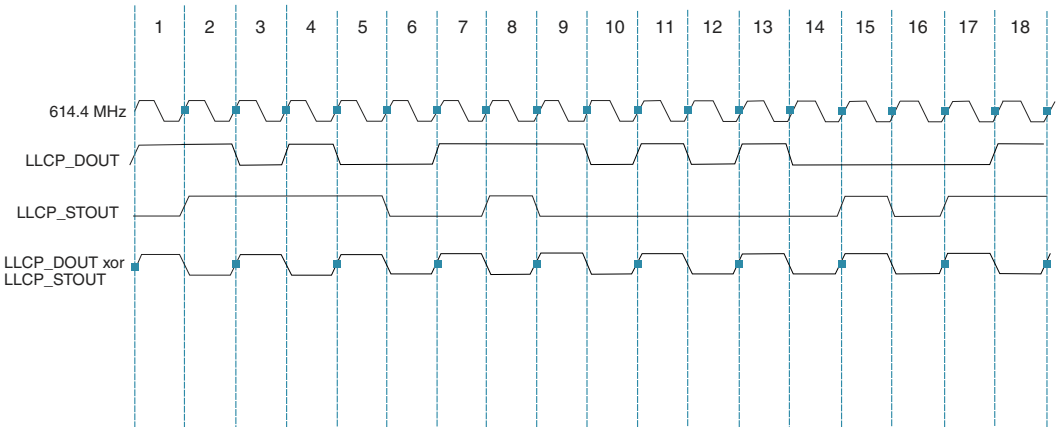


Figure 41. Data and Strobe Timing Diagram

16.4.6 Software Considerations

There are various software considerations among the transactions being transmitted and received through the LLCP.

1. Software must ensure sufficient time has passed before submitting another write gain transaction across the APB system bus. This ensures proper detection of the active transaction has been completed. Realistically once every 1us.
2. Software must be aware of read response data delays on the APB system bus as they are transmitted across chip to the RFIC via the Data/Strobe pins. An APB read request will not complete until data has been returned from the RFIC.

16.4.7 Compensation Delay Circuit

The LLCP generates a recovered clock from the Data and Strobe pins on the receive side logic. Data and Strobe will be XORd to achieve this. To help account for skew between the data and strobe signals, a compensation delay module have been added as shown in the figure below. The rectangles in red below are the circuits that will add up to ~100ps. On the RX pins the delay module is added after the IO and DFT mux structure. On the TX pins the delay module is added before the DFT mux structure.

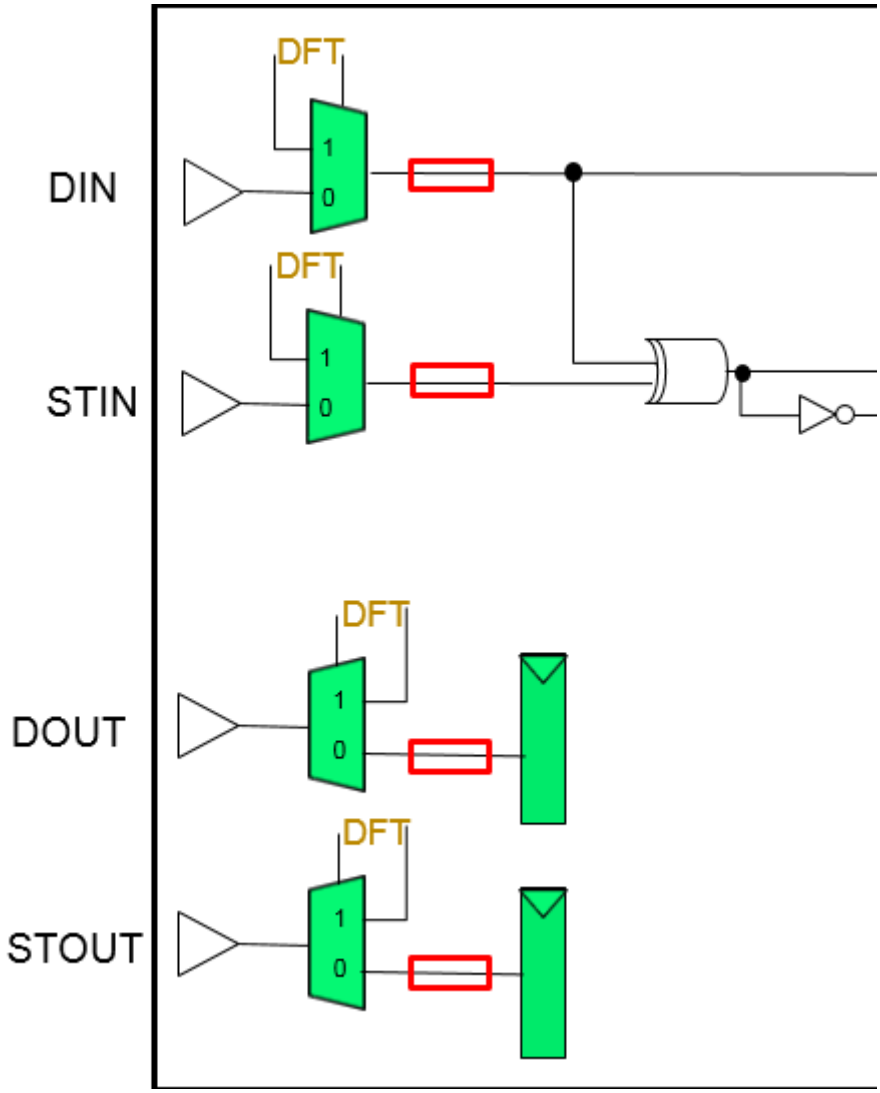


Figure 42. LLCP Top-level RX/TX pins

Programming the LLCPDLY register will configure the selects on the compensation delay mux as shown. The diagram below illustrates the simple circuit representing the logic, along with the desired delays. There are four selects contained in the LLCPDLY register, one for each pin; LLCP_DIN, LLCP_STIN, LLCP_DOUT, LLCP_STOUT. The default is no buffer delay added and only includes the delay through the mux. Each buffer delay will vary but be in approximation of a typical buffer delay in cln28hpm technology.

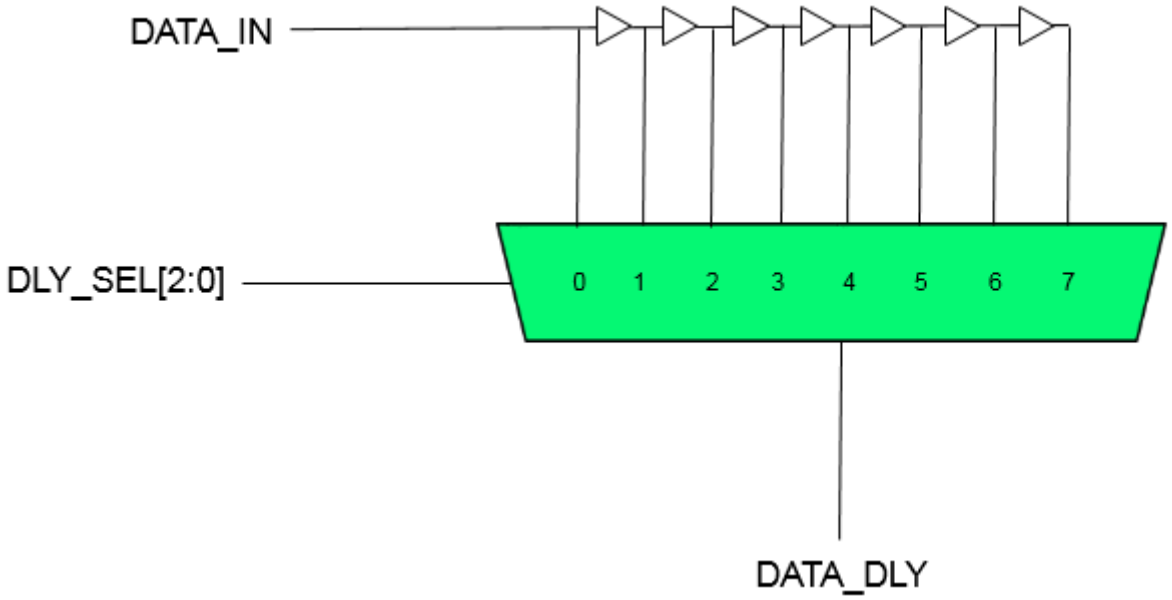


Figure 43. LLCP Compensation Delay Circuit

Chapter 17
Message Unit

17.1 Introduction

Message Unit converts memory mapped write transactions to interrupt signals to the M4 NVIC. Writes to the MSIIR register simultaneously generate an interrupt signal and set a bit in the MSIR register associated with each interrupt so that multiple “messages” can be sent and processed separately. There are three sets of MSIIR/MSIR registers and three associated interrupts to the M4 interrupt controller. The primary use of the message unit is to allow the PCIe host to send a message to the M4 core to process commands. However VSPA can also use the Message registers to generate an interrupt to the M4.

17.2 MSU register descriptions

17.2.1 MSU Memory map

msu base address: 1FC_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Message Signaled Interrupt Index Register (MSIIR1)	32	WO	0000_0000h
4h	Message Signaled Interrupt Register (MSIR1)	32	RO	0000_0000h
8h	Message Signaled Interrupt Index Register (MSIIR2)	32	WO	0000_0000h
Ch	Message Signaled Interrupt Register (MSIR2)	32	RO	0000_0000h
10h	Message Signaled Interrupt Index Register (MSIIR3)	32	WO	0000_0000h
14h	Message Signaled Interrupt Register (MSIR3)	32	RO	0000_0000h

17.2.2 Message Signaled Interrupt Index Register (MSIIR1 - MSIIR3)

Offset

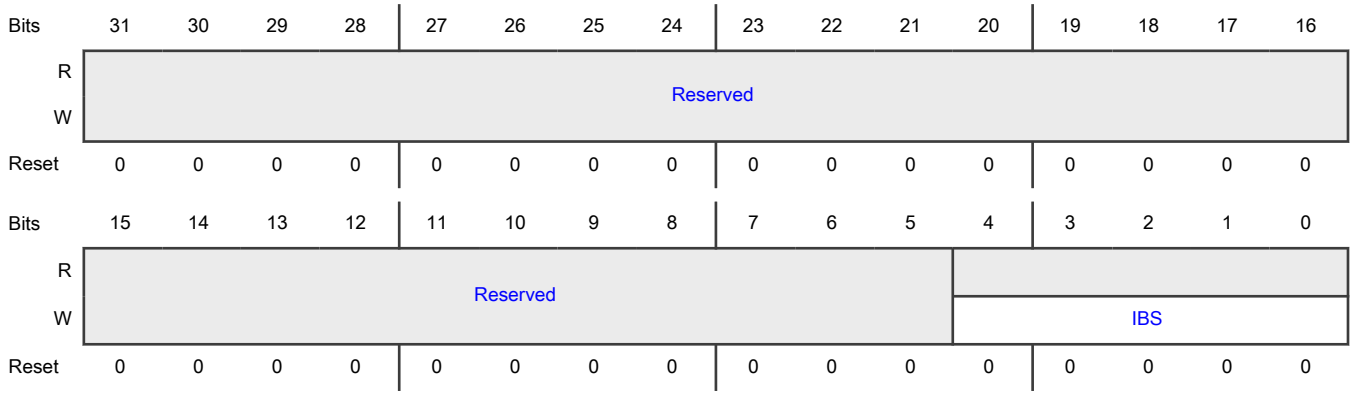
For a = 1 to 3:

Register	Offset
MSIIRa	-8h + (a × 8h)

Function

Writing the Interrupt Bit Select (IBS) field of the MSIIR register generates an interrupt to the processor and sets the corresponding bit of the associated MSIR registers.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 IBS	IBS Interrupt bit selects the bit to set in the MSIR. 00000b - Set field SH0 (bit 0) 00001b - Set field SH1 (bit 1) 00010b - Set field SH2 (bit 2) 11111b - Set field SH31 (bit 31)

17.2.3 Message Signaled Interrupt Register (MSIR1 - MSIR3)

Offset

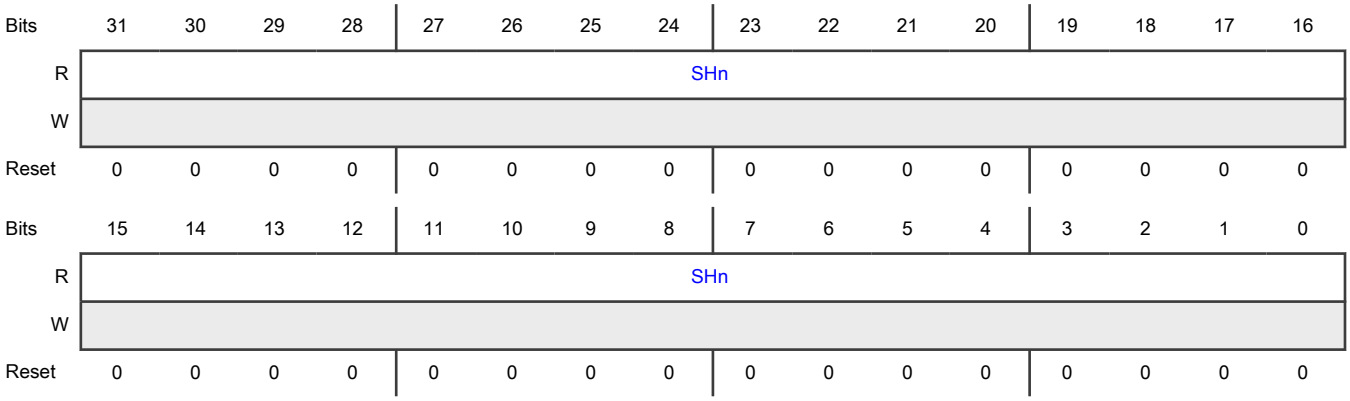
For a = 1 to 3:

Register	Offset
MSIRa	-4h + (a × 8h)

Function

The message signaled interrupt register indicates which of the up-to-32 interrupt sources sharing a message register have pending interrupts. This register is cleared when read; writes to the register have no effect.

Diagram



Fields

Field	Function
31-0 SHn	Message sharer n has a pending interrupt.

Chapter 18
PCI Express Interface Controller

18.1 The PCI Express controller as implemented on the chip

This section provides details about how the PCI Express controller is integrated into this chip.

Table 63. PCI Express controller integration

Controller	Base address	Maximum number of lanes
PCIe PF0	340_0000h	1
PCIe PF0 Controls	34C_0000h	—

18.2 Introduction

The PCI Express interface is compatible with the *PCI Express™ Base Specification, Revision 3.0* (available from http://www.pcisig.org). It is beyond the scope of this manual to document the intricacies of the PCI Express protocol. This chapter describes the PCI Express controller of this device and provides a basic description of the PCI Express protocol. The specific emphasis is directed at how the device implements the PCI Express specification. Designers of systems incorporating PCI Express devices should refer to the specification for a thorough description of PCI Express.

NOTE

Much of the available PCI Express literature refers to a 16-bit quantity as a WORD and a 32-bit quantity as a DWORD. Note that this is inconsistent with the terminology in the rest of this manual where the terms 'word' and 'double word' refer to a 32-bit and 64-bit quantity, respectively. Where necessary to avoid confusion, the precise number of bits or bytes is specified.

18.2.1 Overview

The PCI Express controller connects the internal platform to a serial interface.

As both an initiator and a target device, the PCI Express interface is capable of high-bandwidth data transfer and is designed to support next generation I/O devices. Upon coming out of reset, the PCI Express interface performs link width negotiation and exchanges flow control credits with its link partner. Once link autonegotiation is successful, the controller is in operation.

Internally, the design contains queues to keep track of inbound and outbound transactions. There is control logic that handles buffer management, bus protocol, transaction spawning and tag generation. In addition, there are memory blocks used to store inbound and outbound data.

The PCI Express controller operates as a PCI Express Endpoint (EP) device. An EP device typically denotes a peripheral or I/O device. In EP mode, a PCI Express type 0 configuration header is used.

This figure shows a high-level block diagram of the PCI Express controller.

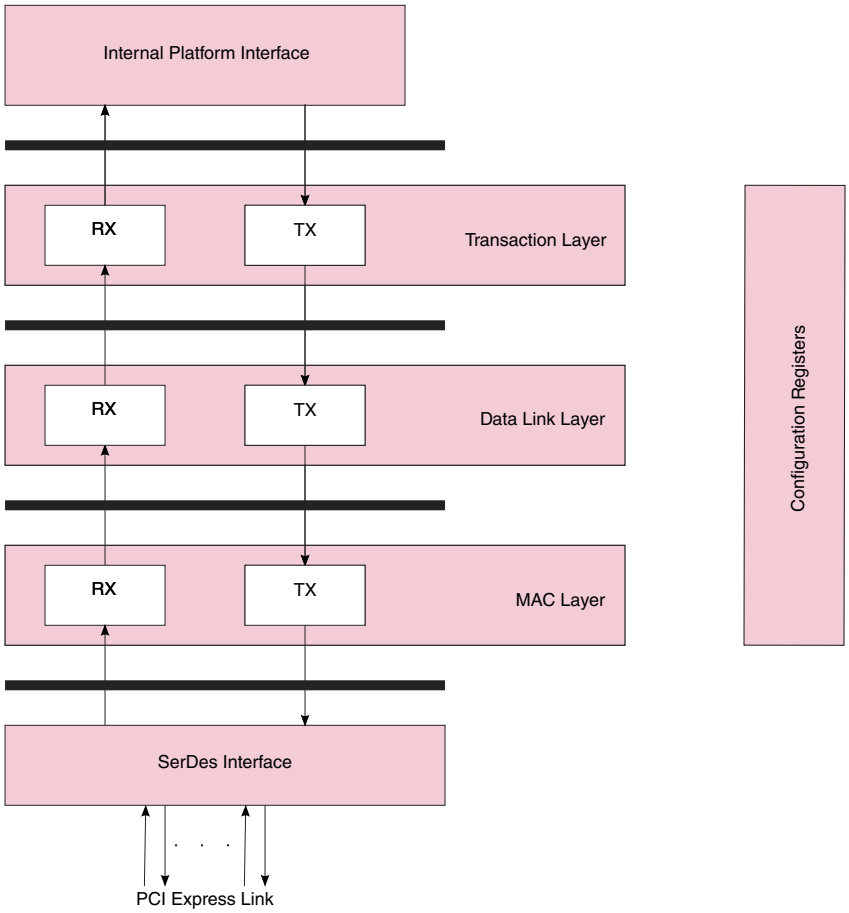


Figure 44. PCI Express Controller Block Diagram

As an initiator, the PCI Express controller supports memory read and write operations. As a target interface, the PCI Express controller accepts read and write operations to local memory space. As an EP device, the PCI Express controller accepts configuration transactions to the internal PCI Express configuration registers. Message generation and acceptance are supported. Locked transactions and inbound I/O transactions are not supported.

18.2.1.1 Outbound Transactions

Outbound internal platform transactions to PCI Express are first mapped to a translation window to determine what PCI Express transactions are to be issued. A transaction from the internal platform can become a PCI Express Memory, I/O,Message, or Configuration transaction depending on the window attributes.

A transaction may be broken up into smaller sized transactions depending on the original request size, transaction type, and either the PCI Express Device Control register [MAX_PAYLOAD_SIZE] field for write requests or the PCI Express Device Control register [MAX_READ_SIZE] field for read requests. The controller performs PCI Express ordering rule checking to determine which transaction is to be sent on the PCI Express link.

In general, transactions are serviced in the order that they are received from the internal platform . Only when there is a stalled condition does the controller apply PCI Express ordering rules to outstanding transactions. For posted write transactions, once all data has been received from the internal platform , the data is forwarded to the PCI Express link and the transaction is considered as done. For non-posted write transactions, the controller waits for the completion packets to return before considering the transaction finished. For non-posted read transactions, the controller waits for all completion packets to return and then forwards all data back to the internal platform before terminating the transaction.

Note that after reset or when recovering from a link down condition, external transactions should not be attempted until the link has successfully trained. Software can poll the LTSSM state status to check the status of link training before issuing external requests.

In EP mode, after reset or when recovering from a link down condition, the SoC must not generate any outbound memory or I/O transactions until the remote host has configured the Bus Master Enable bit in the PCI Command register.

18.2.1.2 Inbound Transactions

Inbound PCI Express transactions to internal platform are first mapped to a translation window to determine what internal platform transactions are to be issued.

A transaction may be broken up into smaller sized transactions when sending to the internal platform depending on the original request size, byte enables and starting/ending addresses. The controller performs PCI Express ordering rule checking to determine what transaction is to be sent next to the internal platform.

In general, transactions are serviced in the order that they are received from the PCI Express link. Only when there is a stalled condition does the controller apply PCI Express ordering to outstanding transactions. For posted write transactions, once all data has been received from the PCI Express link, the data is forwarded to the internal platform and the transaction is considered as done. For non-posted read transactions, the controller forwards internal platform data back to the PCI Express link.

Note that the controller splits transactions at the crossing of every 256-byte-aligned boundary when sending data back to the PCI Express link.

18.2.2 Features

The following is a list of features supported by the PCI Express controller:

- Compatible with the *PCI Express™ Base Specification, Revision 3.0*
- Supports Endpoint (EP) mode
- 32- and 64-bit PCI Express address support
- 32-bit internal platform address support
- x1 link support.
- Supports accesses to all PCI Express memory and I/O address spaces (requestor only)
- Supports posting of processor-to-PCI Express and PCI Express-to-memory writes
- Supports strong and relaxed transaction ordering rules
- Enforces outbound PCI Express ordering rules and inbound internal platform priority
- PCI Express configuration registers (type 0)
- Baseline and advanced error reporting support
- One virtual channel (VC0)
- 256-byte maximum payload size (MAX_PAYLOAD_SIZE)
- Supports 64-bit MSI interrupts
- Credit-based flow control management handled by PCI Express core.
- Supports PCI Express messages and interrupts
- Accepts up to 256-byte transactions from the internal platform
- Supports Expansion ROM.

18.3 External Signal Descriptions

The PCI Express specification defines the connection between two devices as a link, which can be composed of a single or multiple lanes. Each lane consists of a differential pair for transmitting (TX[n]_P and TX[n]_N) and a differential pair for receiving (RX[n]_P and RX[n]_N) with an embedded data clock.

Table 64. PCI Express Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
SD_RX[n]_P	I	Receive data, positive. The receive data signals carry PCI Express packet information.	
		State Meaning	Asserted/Negated—Represents data being received from the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 3.0*.
SD_RX[n]_N	I	Receive data, negative. The receive data signals carry PCI Express packet information.	
		State Meaning	Asserted/Negated—Represents the inverse of data being received from the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 3.0*.
SD_TX[n]_P	O	Transmit data, positive. The transmit data signals carry PCI Express packet information.	
		State Meaning	Asserted/Negated—Represents data being transmitted to the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 3.0*.
SD_TX[n]_N	O	Transmit data, negative. The transmit data signals carry PCI Express packet information.	
		State Meaning	Asserted/Negated—Represents the inverse of data being transmitted to the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 3.0*.

18.4 Memory map/register overview

PCI Express configuration registers

The PCI Express module supports the same configuration registers in both the internal memory map and in PCI Express configuration space. They differ only in whether they are accessed from an internal initiator or from an external initiator on the PCI Express interface. With the exception of the registers in the PEX module internal configuration space, the configuration registers are specified by the PCI Express specification for every PCI Express device. The registers in the PEX module internal configuration space are used for chip-specific functionality and are not defined by the PCI Express specification.

Table 65. PCI Express memory map

Register space	Offset	NEXT pointer	Notes
Type 0 configuration header	0x000	0x40	EP use Type 0
Power management capability structure	0x040	0x50	

Table continues on the next page...

Table 65. PCI Express memory map (continued)

Register space	Offset	NEXT pointer	Notes
MSI message capability structure	0x050	0x70	
PCI Express capability structure	0x070	0x000 (NULL)	
Advanced error reporting capability structure	0x100	0x148	
Secondary PCI Express capability structure	0x148	0x000 (NULL)	
PEX module internal configuration space	0x700	—	
BAR Mask Registers	0x1000	—	

18.4.1 PEX register descriptions

18.4.1.1 PCI_Express_Configuration_Registers Memory map

PEX base address: 340_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	PCI Express Vendor ID Register (Vendor_ID_Register)	16	RO	1957h
2h	PCI Express Device ID Register (Device_ID_Register)	16	RO	1C12h
4h	PCI Express Command Register (Command_Register)	16	RW	0000h
6h	PCI Express Status Register (Status_Register)	16	W1C	0010h
8h	PCI Express Revision ID Register (Revision_ID_Register)	8	RO	01h
9h	PCI Express Class Code Register (Class_Code_Register)	24	RO	00_0000h
Ch	PCI Express Cache Line Size Register (Cache_Line_Size_Register)	8	RW	00h
Dh	PCI Express Latency Timer Register (Latency_Timer_Register)	8	RO	00h
Eh	PCI Express Header Type Register (Header_Type_Register)	8	RO	00h
10h	PCI Express Base Address Register 0 (BAR0)	32	RW	0000_0000h
14h	PCI Express Base Address Register 1 (BAR1)	32	RW	0000_0000h
18h	PCI Express Base Address Register 2 (BAR2)	32	RW	0000_000Ch
1Ch	PCI Express Base Address Register 3 (BAR3)	32	RW	0000_0000h
2Ch	PCI Express Subsystem Vendor ID Register (Subsystem_Vendor_ID_Register)	16	RO	0000h
2Eh	PCI Express Subsystem ID Register (Subsystem_ID_Register)	16	RO	0000h
30h	PCI Express Expansion ROM Base Address Register (EP-Mode) (Expansion_ROM_BAR_Type0)	32	RW	0000_0000h
34h	Capabilities Pointer Register (Capabilities_Pointer_Register)	8	RO	40h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
3Ch	PCI Express Interrupt Line Register (Interrupt_Line_Register)	8	RW	FFh
3Dh	PCI Express Interrupt Pin Register (Interrupt_Pin_Register)	8	RO	01h
3Eh	PCI Express Minimum Grant Register (Minimum_Grant_Register)	8	RO	00h
3Fh	PCI Express Maximum Latency Register (Maximum_Latency_Register)	8	RO	00h
40h	PCI Express Power Management Capability ID Register (Power_Management_Capability_ID_Register)	8	RO	01h
42h	PCI Express Power Management Capabilities Register (Power_Management_Capabilities_Register)	16	RO	7E23h
44h	PCI Express Power Management Status and Control Register (Power_Management_Status_and_Control_Register)	16	RW	0000h
47h	PCI Express Power Management Data Register (Power_Management_Data_Register)	8	RO	00h
50h	PCI Express MSI Message Capability ID Register (MSI_Message_Capability_ID_Register)	8	RO	05h
52h	PCI Express MSI Message Control Register (MSI_Message_Control_Register)	16	RW	0088h
54h	PCI Express MSI Message Address Register (MSI_Message_Address_Register)	32	RW	0000_0000h
58h	PCI Express MSI Message Upper Address Register (MSI_Message_Upper_Address_Register)	32	RW	0000_0000h
5Ch	PCI Express MSI Message Data Register (MSI_Message_Data_Register)	16	RW	0000h
70h	PCI Express Capability ID Register (Capability_ID_Register)	8	RO	10h
72h	PCI Express Capabilities Register (Capabilities_Register)	16	RO	0002h
74h	PCI Express Device Capabilities Register (Device_Capabilities_Register)	32	RO	0000_8001h
78h	PCI Express Device Control Register (Device_Control_Register)	16	RW	2810h
7Ah	PCI Express Device Status Register (Device_Status_Register)	16	W1C	0000h
7Ch	PCI Express Link Capabilities Register (Link_Capabilities_Register)	32	RO	0043_F443h
80h	PCI Express Link Control Register (Link_Control_Register)	16	RW	0008h
82h	PCI Express Link Status Register (Link_Status_Register)	16	RO	1000h
94h	PCI Express Device Capabilities 2 Register (Device_Capabilities_2_Register)	32	RO	0000_001Fh
98h	PCI Express Device Control 2 Register (Device_Control_2_Register)	16	RW	0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
9Ch	PCI Express Link Capabilities 2 Register (Link_Capabilities_2_Register)	32	RO	0000_000Eh
A0h	PCI Express Link Control 2 Register (Link_Control_2_Register)	16	RW	0003h
A2h	PCI Express Link Status 2 Register (Link_Status_2_Register)	16	RO	0000h
100h	PCI Express Advanced Error Reporting Capability ID Register (Advanced_Error_Reporting_Capability_ID_Register)	16	RO	0001h
104h	PCI Express Uncorrectable Error Status Register (Uncorrectable_Error_Status_Register)	32	W1C	0000_0000h
108h	PCI Express Uncorrectable Error Mask Register (Uncorrectable_Error_Mask_Register)	32	RW	0000_0000h
10Ch	PCI Express Uncorrectable Error Severity Register (Uncorrectable_Error_Severity_Register)	32	RW	0046_2030h
110h	PCI Express Correctable Error Status Register (Correctable_Error_Status_Register)	32	W1C	0000_0000h
114h	PCI Express Correctable Error Mask Register (Correctable_Error_Mask_Register)	32	RW	0000_2000h
118h	PCI Express Advanced Error Capabilities and Control Register (Advanced_Error_Capabilities_and_Control_Register)	32	RW	0000_00A0h
11Ch	PCI Express Header Log Register 1 (Header_Log_Register_DWORD1)	32	RO	0000_0000h
120h	PCI Express Header Log Register 2 (Header_Log_Register_DWORD2)	32	RO	0000_0000h
124h	PCI Express Header Log Register 3 (Header_Log_Register_DWORD3)	32	RO	0000_0000h
128h	PCI Express Header Log Register 4 (Header_Log_Register_DWORD4)	32	RO	0000_0000h
134h	PCI Express Correctable Error Source ID Register (Correctable_Error_Source_ID_Register)	16	RO	0000h
136h	PCI Express Error Source ID Register (Error_Source_ID_Register)	16	RO	0000h
148h	Secondary PCI Express Extended Capability Header (SPCIE_CAP_HEADER_REG)	32	RO	0001_0019h
14Ch	Link Control 3 Register (LINK_CONTROL3_REG)	32	RW	0000_0000h
150h	Lane Error Status Register (LANE_ERR_STATUS_REG)	32	W1C	0000_0000h
154h	Lane Equalization Control Register (LANE0_EQUALIZATION_CONTROL)	16	RO	7F7Fh

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
71Ch	Symbol Timer Register and Filter Mask 1 Register (SYMBOL_TIMER_FILTER_1_OFF)	32	RW	0000_0280h
890h	Gen3 Control Register (GEN3_RELATED_OFF)	32	RW	0080_0001h
8BCh	DBI Read-only Write Enable Register (MISC_CONTROL_1_OFF)	32	RW	0000_0000h
8E0h	Coherency Control Register 1 (COHERENCY_CONTROL_1_OFF)	32	RW	0000_0000h
8E4h	Coherency Control Register 2 (COHERENCY_CONTROL_2_OFF)	32	RW	0000_0000h
8E8h	Coherency Control Register 3 (COHERENCY_CONTROL_3_OFF)	32	RU	0000_0000h
900h	iATU Index Register (IATU_VIEWPORT_OFF)	32	RW	0000_0000h
904h	iATU Region Control 1 Register (IATU_REGION_CTRL_1_OFF_INBOUND_0)	32	RW	0000_0000h
904h	iATU Region Control 1 Register (IATU_REGION_CTRL_1_OFF_OUTBOUND_0)	32	RW	0000_0000h
908h	iATU Region Control 2 Register (IATU_REGION_CTRL_2_OFF_INBOUND_0)	32	RW	0000_0000h
908h	iATU Region Control 2 Register (IATU_REGION_CTRL_2_OFF_OUTBOUND_0)	32	RW	0000_0000h
90Ch	iATU Lower Base Address Register (IATU_LWR_BASE_ADDR_OFF_INBOUND_0)	32	RW	0000_0000h
90Ch	iATU Lower Base Address Register (IATU_LWR_BASE_ADDR_OFF_OUTBOUND_0)	32	RW	0000_0000h
910h	iATU Upper Base Address Register (IATU_UPPER_BASE_ADDR_OFF_INBOUND_0)	32	RW	0000_0000h
910h	iATU Upper Base Address Register (IATU_UPPER_BASE_ADDR_OFF_OUTBOUND_0)	32	RW	0000_0000h
914h	iATU Limit Address Register (IATU_LIMIT_ADDR_OFF_INBOUND_0)	32	RW	0000_0FFFh
914h	iATU Limit Address Register (IATU_LIMIT_ADDR_OFF_OUTBOUND_0)	32	RW	0000_0FFFh
918h	iATU Region#N Lower Offset Address Register (IATU_LWR_TARGET_ADDR_OFF_INBOUND_0)	32	RW	0000_0000h
918h	iATU Outbound Region#N Lower Offset Address Register (IATU_LWR_TARGET_ADDR_OFF_OUTBOUND_0)	32	RW	0000_0000h
91Ch	iATU Upper Target Address Register (IATU_UPPER_TARGET_ADDR_OFF_INBOUND_0)	32	RW	0000_0000h
91Ch	iATU Upper Target Address Register (IATU_UPPER_TARGET_ADDR_OFF_OUTBOUND_0)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
1010h	Base Address Register 0 Mask (BAR0_MASK)	32	WO	0FFF_FFFFh
1014h	Base Address Register 1 Mask (BAR1_MASK)	32	WO	0001_FFFFh
1018h	Base Address Register 2 Mask (BAR2_MASK)	32	WO	007F_FFFFh
101Ch	Base Address Register 3 Mask (BAR3_MASK)	32	WO	FFFF_FFFFh
1030h	Expansion ROM Base Address Register Mask (EP mode) (EXP_ROM_BAR_MASK_EP)	32	WO	00FF_FFFFh

18.4.1.2 PCI Express Vendor ID Register (Vendor_ID_Register)

Offset

Register	Offset
Vendor_ID_Register	0h

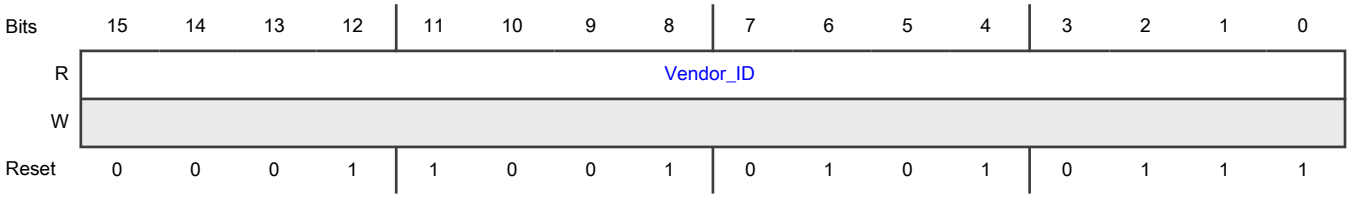
Function

The vendor ID register is used to identify the manufacturer of the device.

NOTE

This register is writeable using internal accesses, but is read-only from inbound configuration accesses by an external host.

Diagram



Fields

Field	Function
15-0 Vendor_ID	Vendor ID 0x1957 (NXP)

18.4.1.3 PCI Express Device ID Register (Device_ID_Register)

Offset

Register	Offset
Device_ID_Register	2h

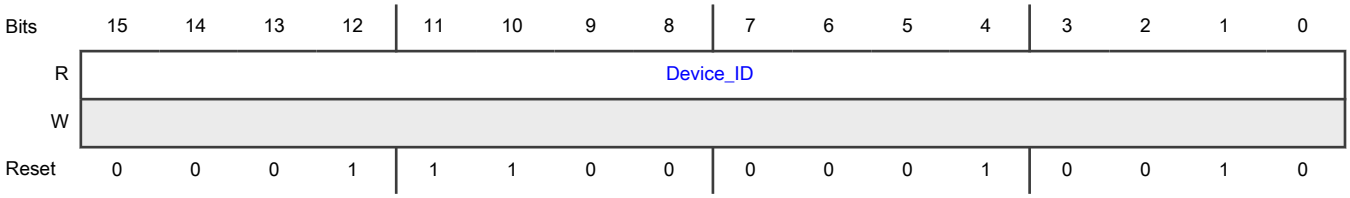
Function

The device ID register is used to identify the device.

NOTE

This register is writeable using internal accesses, but is read-only from inbound configuration accesses by an external host.

Diagram



Fields

Field	Function
15-0 Device_ID	Device ID 0001110000010010b - Semicustom Family

18.4.1.4 PCI Express Command Register (Command_Register)

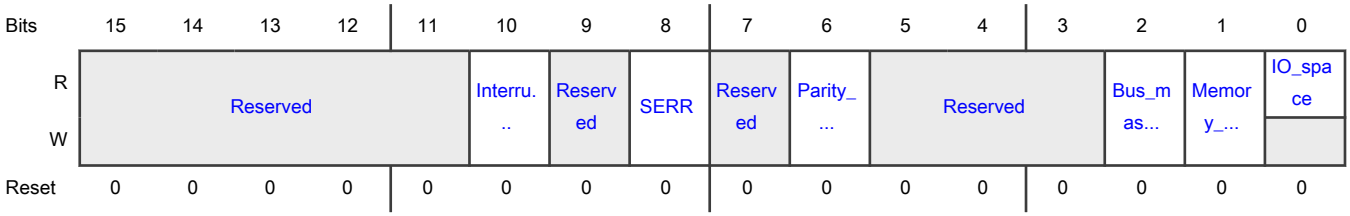
Offset

Register	Offset
Command_Register	4h

Function

The command register provides control over the ability to generate and respond to PCI Express cycles.

Diagram



Fields

Field	Function
15-11 —	Reserved
10 Interrupt_Disable	Interrupt disable Controls the ability to generate INTx interrupt messages. Any INTx emulation interrupts already asserted by this device must be deasserted when this bit is set. 0b - Enables INTx interrupt messages 1b - Disables INTx interrupt messages
9 —	Reserved
8 SERR	SERR# enable Controls the reporting of fatal and non-fatal errors detected by the device to the Root Complex. **NOTE** The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in #unique_475/unique_475_Connect_42_Device_Control_Register and the advanced error reporting registers (offsets 100h through 137h). 0b - Disables reporting 1b - Enables reporting
7 —	Reserved
6 Parity_error_response	Parity error response Controls whether this PCI Express controller responds to parity errors.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in #unique_475/unique_475_Connect_42_Device_Control_Register and the advanced error reporting registers (offsets 100h through 137h). 0b - Parity errors are ignored and normal operation continues. 1b - Parity errors cause the appropriate bit in the PCI Express status register to be set. However, note that errors are reported based on the values set in the PCI Express error enable and detection registers.
5-3 —	Reserved
2 Bus_master	Bus master enable Indicates whether this PCI Express device is configured as a master. EP mode: Clearing this bit prevent the device from issuing any memory or I/O transactions. Because MSI interrupts are effectively memory writes, clearing this bit also disables the ability of the device to issue MSI interrupts. 0b - Disables the ability to generate PCI Express accesses 1b - Enables this PCI Express controller to behave as a PCI Express bus master
1 Memory_space	Memory space enable Controls whether this PCI Express device (as a target) responds to memory accesses. EP mode: Clearing this bit prevents the device from accepting any memory transaction. 0b - This PCI Express device does not respond to PCI Express memory space accesses. 1b - This PCI Express device responds to PCI Express memory space accesses.
0 IO_space	I/O space enable EP mode: Clearing this bit prevents the device from accepting any IO transaction. Note that this bit is a don't care in EP mode since the device does not support IO transaction. 0b - This PCI Express device (as a target) does not respond to PCI Express I/O space accesses. 1b - This PCI Express device (as a target) does respond to PCI Express I/O space accesses.

18.4.1.5 PCI Express Status Register (Status_Register)

Offset

Register	Offset
Status_Register	6h

Function

The status register is used to record status information for PCI Express related events.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Detecte...	Signale...	Receive...	Receive...	Signale...	Reserved		Master_...	Reserved			Capabil...	Interru...	Reserved		
W	W1C	W1C	W1C	W1C	W1C			W1C				W1C	W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Fields

Field	Function
15 Detected_parity_error	Detected parity error Set whenever a device receives a poisoned TLP regardless of the state of bit 6 in the command register.
14 Signaled_system_error	Signaled system error Set whenever a device sends a ERR_FATAL or ERR_NONFATAL message and the SERR enable bit in the command register is set.
13 Received_master_abort	Received master abort Set whenever a requestor receives a completion with unsupported request completion status.
12 Received_target_abort	Received target abort Set whenever a device receives a completion with completer abort completion status.
11 Signaled_target_abort	Signaled target abort Set whenever a device completes a request using completer abort completion status.
10-9 —	Reserved
8 Master_data_parity_error_detected	Master data parity error Set by the requestor (primary side for Type1 headers) when either the requestor receives a completion marked poisoned or the requestor poisons a write request. Set by the EP when the EP function either receives a poisoned completion or transmits a poisoned request. Note that the parity error enable bit (bit 6) in the command register must be set for this bit to be set.
7-5 —	Reserved
4	Capabilities list

Table continues on the next page...

Table continued from the previous page...

Field	Function
Capabilities_List	All PCI Express devices are required to implement the PCI Express capability structure.
3 Interrupt_Status	Interrupt status Set when an INTx interrupt message is pending internally to the device. Note that this bit is associated with INTx messages and not Message Signaled Interrupts.
2-0 —	Reserved

18.4.1.6 PCI Express Revision ID Register (Revision_ID_Register)

Offset

Register	Offset
Revision_ID_Register	8h

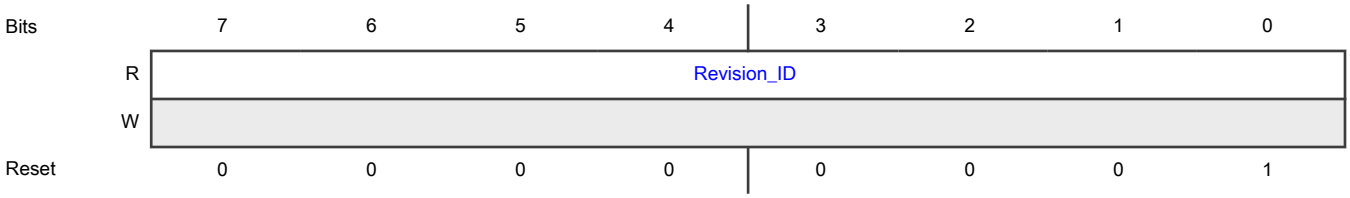
Function

The revision ID register is used to identify the revision of the device.

NOTE

This register is writeable using internal accesses, but is read-only from inbound configuration accesses by an external host.

Diagram



Fields

Field	Function
7-0 Revision_ID	Revision ID Revision specific.

18.4.1.7 PCI Express Class Code Register (Class_Code_Register)

Offset

Register	Offset
Class_Code_Register	9h

Function

The class code register is comprised of three single-byte fields—base class (offset 0x0B), sub-class (offset 0x0A), and programming interface (offset 0x09)—that indicate the basic functionality of the function.

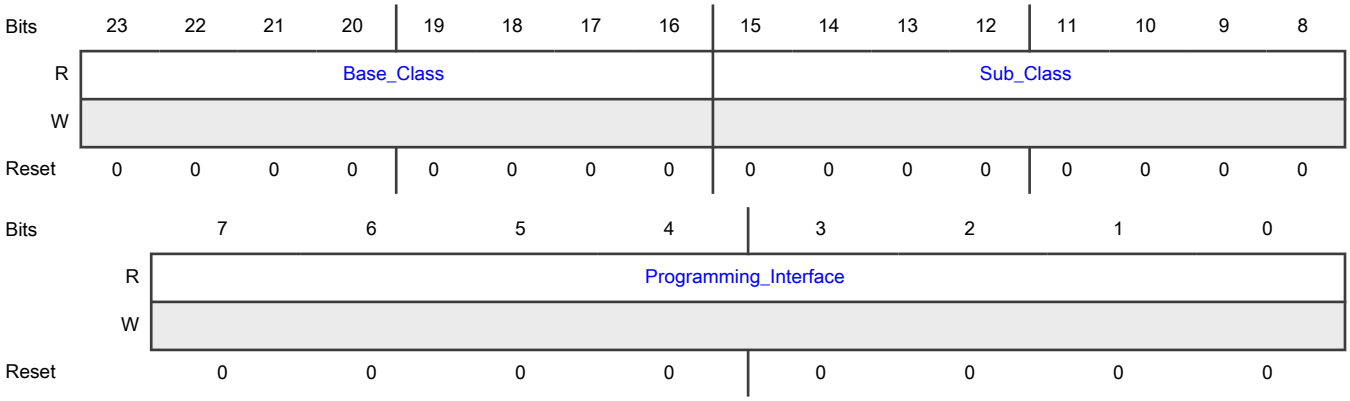
NOTE

This register is writeable using internal PEX_CONFIG_ADDR/PEX_CONFIG_DATA accesses, but is read-only from inbound configuration accesses by an external host.

NOTE

Some OSes will fail to allow proper configuration if the Base Class is 00h (Undefined) or 06h (Bridge). The Base Class value must be programmed to any value other than 00h or 06h before attempting to configure this device remotely.

Diagram



Fields

Field	Function
23-16 Base_Class	Base Class
15-8 Sub_Class	Sub-Class
7-0 Programming_Interface	Programming_Interface

18.4.1.8 PCI Express Cache Line Size Register (Cache_Line_Size_Register)

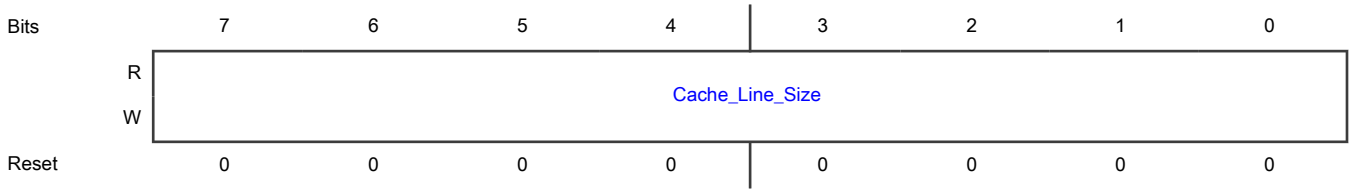
Offset

Register	Offset
Cache_Line_Size_Register	Ch

Function

The cache line size register is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

Diagram



Fields

Field	Function
7-0 Cache_Line_Size	Cache Line Size Represents the cache line size of the processor in terms of 32-bit words (8 32-bit words = 32 bytes). Note that for PCI Express operation this register is ignored.

18.4.1.9 PCI Express Latency Timer Register (Latency_Timer_Register)

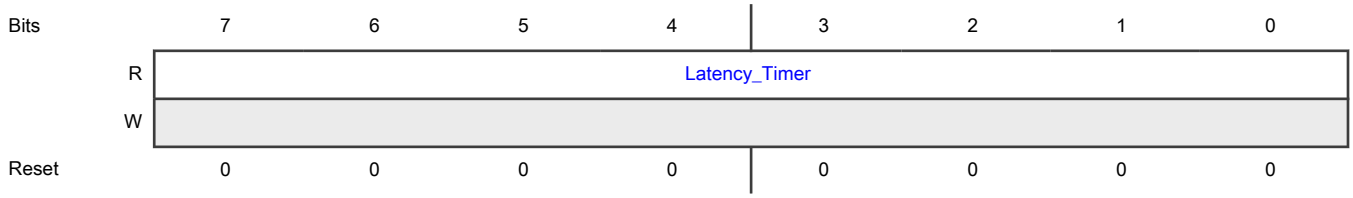
Offset

Register	Offset
Latency_Timer_Register	Dh

Function

The latency timer register is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

Diagram



Fields

Field	Function
7-0 Latency_Timer	Latency_Timer Note that for PCI Express operation this register is ignored.

18.4.1.10 PCI Express Header Type Register (Header_Type_Register)

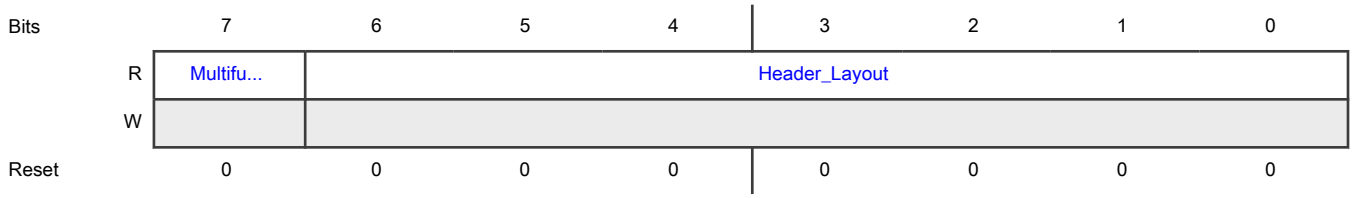
Offset

Register	Offset
Header_Type_Register	Eh

Function

The PCI Express header type register is used to identify the layout of the PCI compatible header.

Diagram



Fields

Field	Function
7 Multifunction	Multifunction Identifies whether a device supports multiple functions 0b - Single function device 1b - Multiple function device
6-0 Header_Layout	Header Layout All other encodings reserved. 0000000b - Endpoint - Type 0 layout.

18.4.1.11 PCI Express Base Address Register 0 (BAR0)

Offset

Register	Offset
BAR0	10h

Function

The PCI Express base address registers (BARs) point to the beginning of distinct address ranges which the device should claim.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	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	ADDRESS				Reserved								PREF	TYPE		MemSp
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 ADDRESS	Base address Indicates the base address of the inbound memory window 0. The default size is 256 MB.
11-4 —	Reserved
3 PREF	Prefetchable
2-1 TYPE	Type 00b - Locate anywhere in 32-bit address space.
0 MemSp	Memory space indicator Base Address registers that map to Memory Space must return a 0 in bit 0. Base Address registers that map to I/O Space must return a 1 in bit 0.

18.4.1.12 PCI Express Base Address Register 1 (BAR1)

Offset

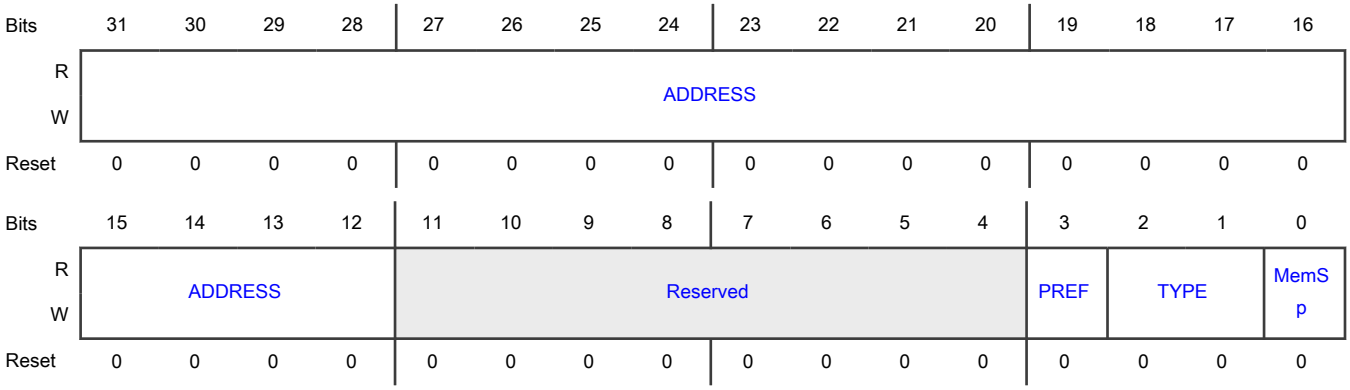
Register	Offset
BAR1	14h

Function

This register is present only in the Type 0 Header (EP mode).

Base address register 1 at offset 0x14 is used to define the inbound memory window in the 32-bit memory space.

Diagram



Fields

Field	Function
31-12 ADDRESS	Base address Indicates the base address where the inbound memory window 1 begins. The default size of this window is 128 KB.
11-4 —	Reserved. The device allows a 4 Kbyte window minimum.
3 PREF	Prefetchable
2-1 TYPE	Type 00b - Locate anywhere in 32-bit address space.
0 MemSp	Memory space indicator Base Address registers that map to Memory Space must return a 0 in bit 0. Base Address registers that map to I/O Space must return a 1 in bit 0.

18.4.1.13 PCI Express Base Address Register 2 (BAR2)

Offset

Register	Offset
BAR2	18h

Function

This register is present only in the Type 0 Header (EP mode).

Together, base address register 2 (BAR2) and base address register 3 (BAR3) define a 64-bit inbound memory window. BAR2 defines the lower portion of the memory window; BAR3 defines the upper portion of the memory window.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	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	ADDRESS				Reserved								PREF	TYPE		MemSp
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Fields

Field	Function
31-12 ADDRESS	Base address (lower portion) Indicates the lower portion of the base address where the inbound memory window begins.
11-4 —	Reserved. The device allows a 4 Kbyte window minimum.
3 PREF	Prefetchable
2-1 TYPE	Type 00b - Locate anywhere in 32-bit address space. 10b - Locate anywhere in 64-bit address space.
0 MemSp	Memory space indicator Base Address registers that map to Memory Space must return a 0 in bit 0. Base Address registers that map to I/O Space must return a 1 in bit 0.

18.4.1.14 PCI Express Base Address Register 3 (BAR3)

Offset

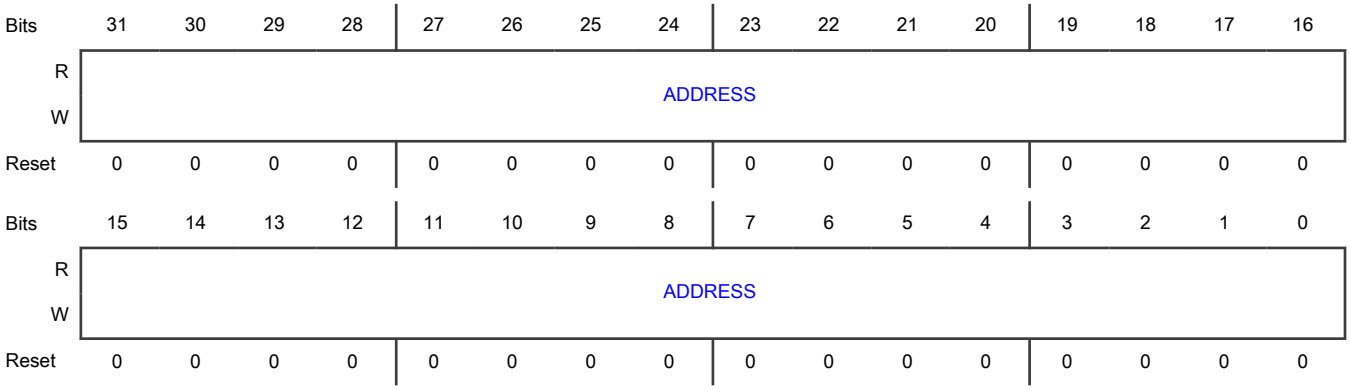
Register	Offset
BAR3	1Ch

Function

This register is present only in the Type 0 Header (EP mode).

Together, base address register 2 (BAR2) and base address register 3 (BAR3) define a 64-bit inbound memory window. BAR2 defines the lower portion of the memory window; BAR3 defines the upper portion of the memory window.

Diagram



Fields

Field	Function
31-0 ADDRESS	Base address (upper portion) Indicates the upper portion of the base address where the inbound memory window begins. If no access to local memory is to be permitted by external requesters, then all bits are programmed.

18.4.1.15 PCI Express Subsystem Vendor ID Register (Subsystem_Vendor_ID_Register)

Offset

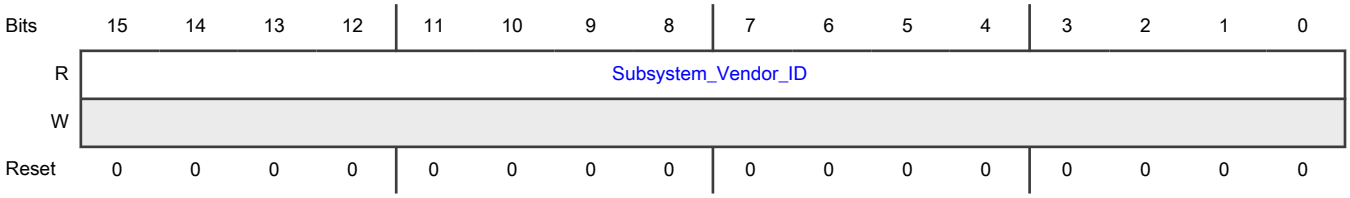
Register	Offset
Subsystem_Vendor_ID_ Register	2Ch

Function

This register is present only in the Type 0 Header (EP mode).

The PCI Express subsystem vendor ID register is used to identify the subsystem.

Diagram



Fields

Field	Function
15-0 Subsystem_Vendor_ID	Subsystem Vendor ID This register is present only in the Type 0 Header (EP mode). The PCI Express subsystem vendor ID register is used to identify the subsystem vendor.

18.4.1.16 PCI Express Subsystem ID Register (Subsystem_ID_Register)

Offset

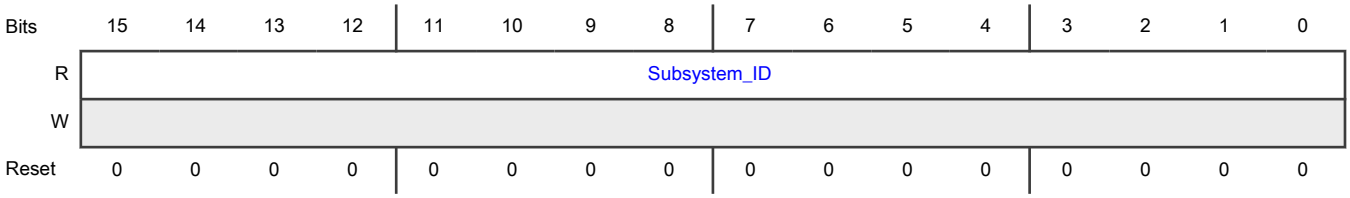
Register	Offset
Subsystem_ID_Register	2Eh

Function

This register is present only in the Type 0 Header (EP mode).

The PCI Express subsystem ID register is used to identify the subsystem.

Diagram



Fields

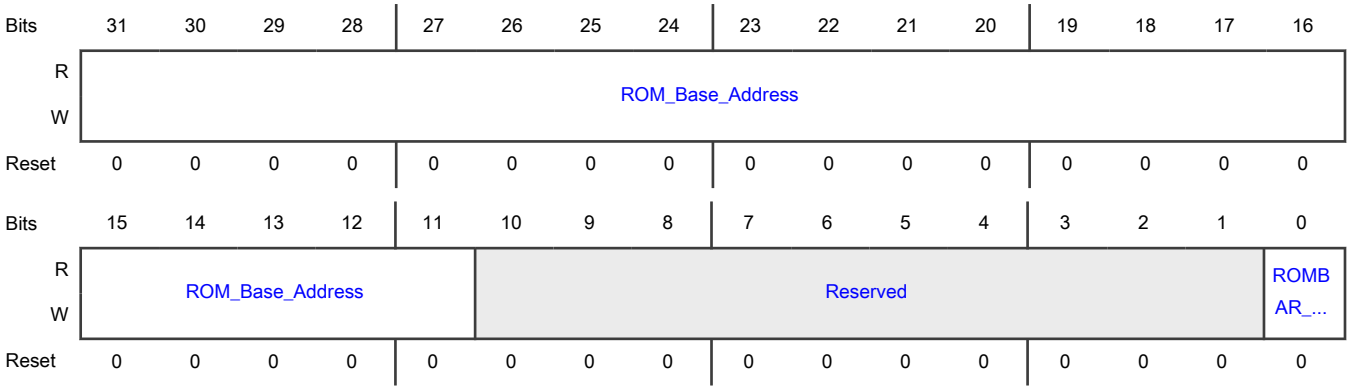
Field	Function
15-0 Subsystem_ID	Subsystem ID

18.4.1.17 PCI Express Expansion ROM Base Address Register (EP-Mode) (Expansion_ROM_BAR_Type0)

Offset

Register	Offset
Expansion_ROM_BAR_Type0	30h

Diagram



Fields

Field	Function
31-11 ROM_Base_Address	Expansion ROM base address Specifies bits 31:11 of the non-prefetchable expansion ROM space start address. Typically used for specifying memory-mapped I/O space. The default size is 16M.
10-1 —	Reserved
0 ROMBAR_EN	Expansion ROM enable This bit controls whether or not the device accepts accesses to its expansion ROM 0b - The expansion ROM address space is disabled. 1b - Address decoding is enabled.

18.4.1.18 Capabilities Pointer Register (Capabilities_Pointer_Register)

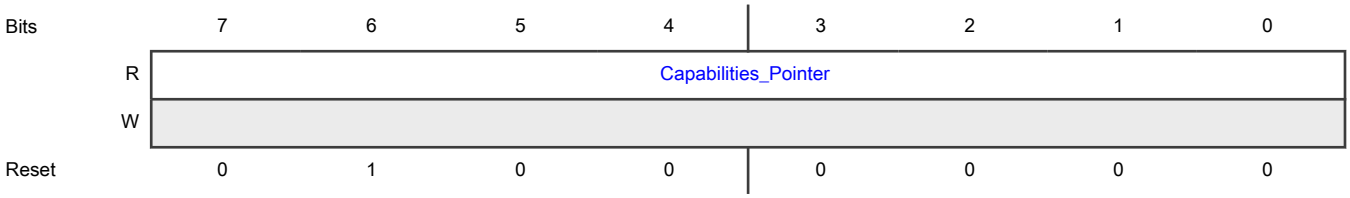
Offset

Register	Offset
Capabilities_Pointer_Register	34h

Function

The capabilities pointer identifies additional functionality supported by the device.

Diagram



Fields

Field	Function
7-0 Capabilities_Pointer	Capabilities Pointer The capabilities pointer provides the offset for additional PCI-compatible registers above the common 64-byte header.

18.4.1.19 PCI Express Interrupt Line Register (Interrupt_Line_Register)

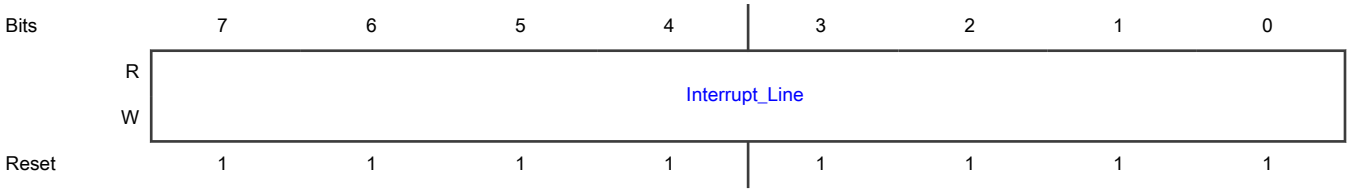
Offset

Register	Offset
Interrupt_Line_Register	3Ch

Function

The interrupt line register is used by device drivers and OS software to communicate interrupt line routing information. Values in this register are programmed by system software and are system specific.

Diagram



Fields

Field	Function
7-0 Interrupt_Line	Interrupt line Used to communicate interrupt line routing information.

18.4.1.20 PCI Express Interrupt Pin Register (Interrupt_Pin_Register)

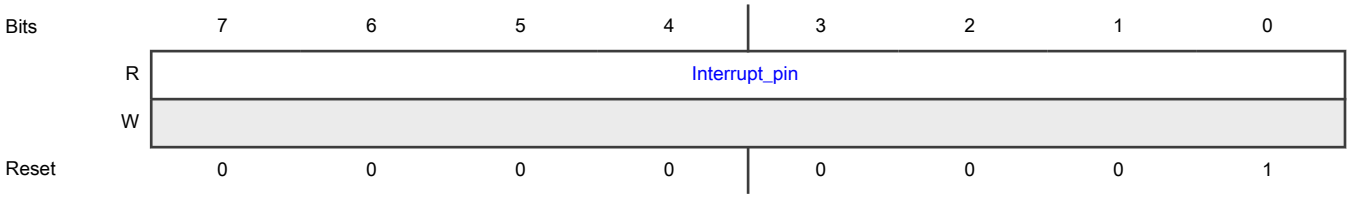
Offset

Register	Offset
Interrupt_Pin_Register	3Dh

Function

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

Diagram



Fields

Field	Function
7-0 Interrupt_pin	Interrupt pin Legacy INTx message used by this device. All other settings reserved. 00000000b - This device does not use legacy interrupt (INTx) messages. 00000001b - INTA

18.4.1.21 PCI Express Minimum Grant Register (Minimum_Grant_Register)

Offset

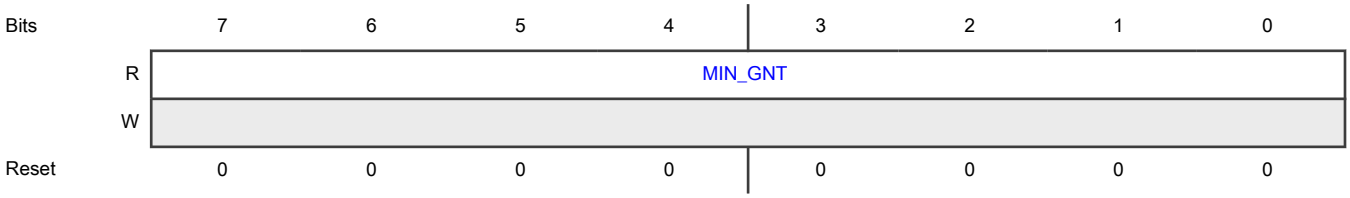
Register	Offset
Minimum_Grant_Register	3Eh

Function

This register is present only in the Type 0 Header (EP mode).

This register does not apply to PCI Express. It is present for legacy purposes.

Diagram



Fields

Field	Function
7-0 MIN_GNT	Minimum grant Does not apply for PCI Express.

18.4.1.22 PCI Express Maximum Latency Register (Maximum_Latency_Register)

Offset

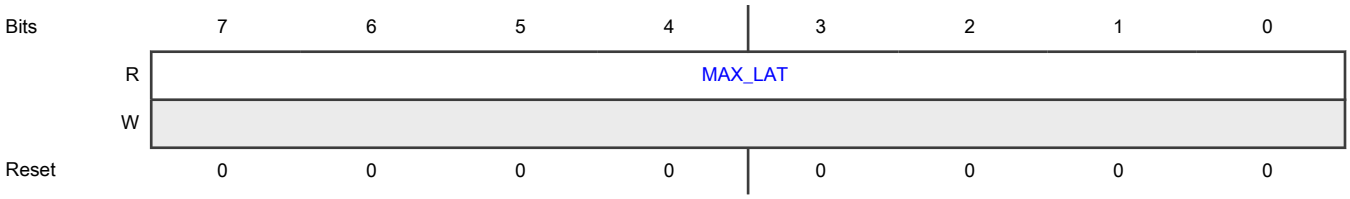
Register	Offset
Maximum_Latency_Register	3Fh

Function

This register is present only in the Type 0 Header (EP mode).

This register does not apply to PCI Express. It is present for legacy purposes.

Diagram



Fields

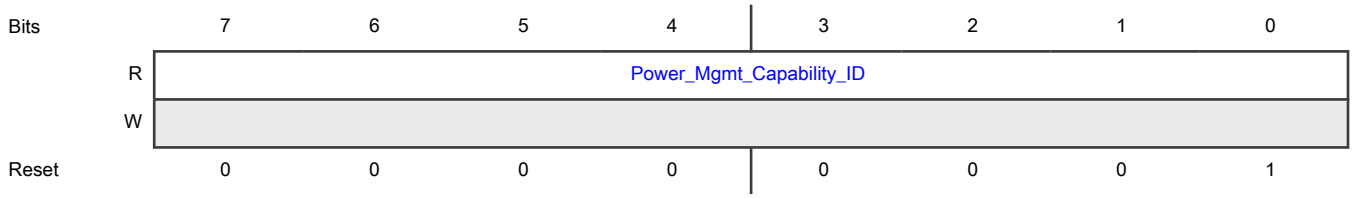
Field	Function
7-0 MAX_LAT	Maximum latency Does not apply for PCI Express.

18.4.1.23 PCI Express Power Management Capability ID Register (Power_Management_Capability_ID_Register)

Offset

Register	Offset
Power_Management_Capability_ID_Register	40h

Diagram



Fields

Field	Function
7-0 Power_Mgmt_Capability_ID	CAP_ID Power Management = 0x01

18.4.1.24 PCI Express Power Management Capabilities Register (Power_Management_Capabilities_Register)

Offset

Register	Offset
Power_Management_Capabilities_Register	42h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PME_Support					D2	D1	AUX_Curr			DSI	Reserved	PME_CLK	Version		
W																
Reset	0	1	1	1	1	1	1	0	0	0	1	0	0	0	1	1

Fields

Field	Function
15-11 PME_Support	PME support Indicates the power states that this device supports
10 D2	D2 support
9 D1	D1 support
8-6 AUX_Curr	AUX Current
5 DSI	Device Specific Initialization
4 —	Reserved
3 PME_CLK	PME clock Does not apply to PCI Express.
2-0 Version	Version

18.4.1.25 PCI Express Power Management Status and Control Register (Power_Management_Status_and_Control_Register)

Offset

Register	Offset
Power_Management_Status_and_Control_Register	44h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PME_STAT	Data_Scale			Data_Select			PME_EN	Reserved						Power_State	
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

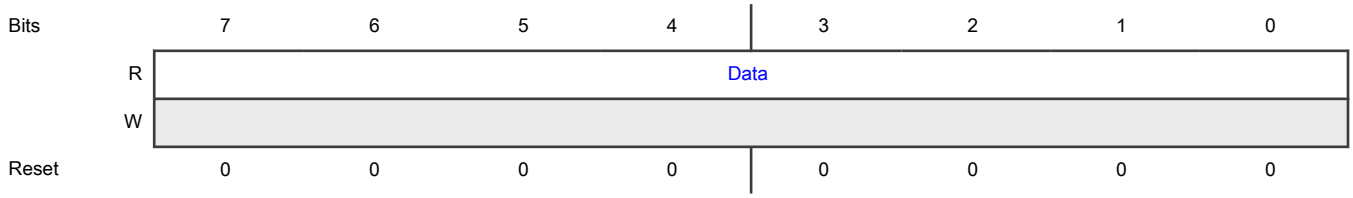
Field	Function
15 PME_STAT	PME Status
14-13 Data_Scale	Data Scale Obtained directly from the PCI Express base specification.
12-9 Data_Select	Data Select Obtained directly from the PCI Express base specification.
8 PME_EN	PME_En PME Enable. **NOTE** Bitfield access is sticky.
7-2 —	Reserved
1-0 Power_State	Power State Indicates the current power state of the function. 00b - D0 01b - D1 10b - D2 11b - D3

18.4.1.26 PCI Express Power Management Data Register (Power_Management_Data_Register)

Offset

Register	Offset
Power_Management_Data_Register	47h

Diagram



Fields

Field	Function
7-0 Data	Data Obtained from the PCI Express base specification.

18.4.1.27 PCI Express MSI Message Capability ID Register (MSI_Message_Capability_ID_Register)

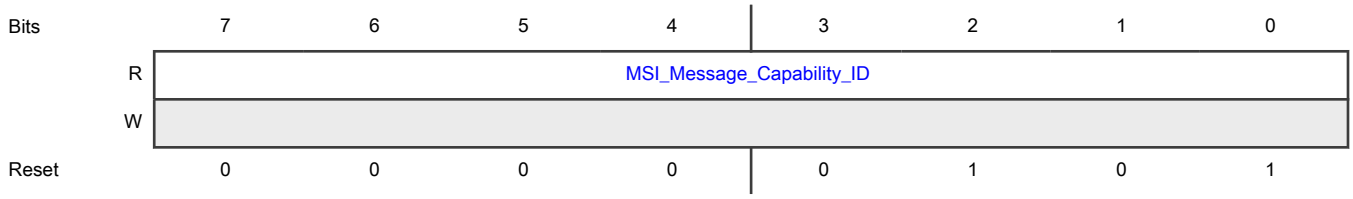
Offset

Register	Offset
MSI_Message_Capability_ID_Register	50h

Function

This register is supported only for EP mode.

Diagram



Fields

Field	Function
7-0 MSI_Message_Capability_ID	CAP_ID MSI Message = 0x05

18.4.1.28 PCI Express MSI Message Control Register (MSI_Message_Control_Register)

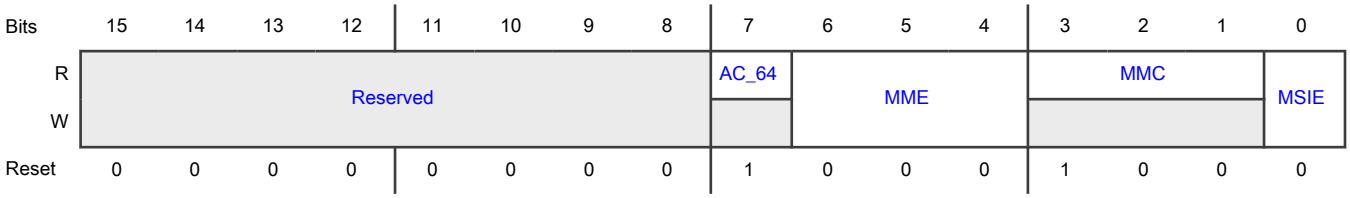
Offset

Register	Offset
MSI_Message_Control_Register	52h

Function

This register is supported only for EP mode.

Diagram



Fields

Field	Function
15-8 —	Reserved
7 AC_64	64-bit address capable
6-4 MME	Multiple message enable
3-1 MMC	Multiple message capable
0 MSIE	MSI enable

18.4.1.29 PCI Express MSI Message Address Register (MSI_Message_Address_Register)

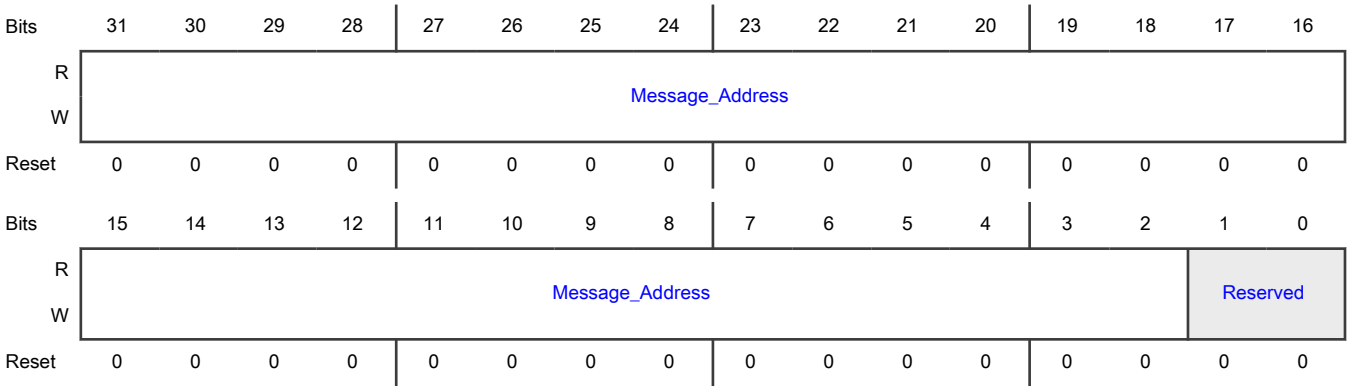
Offset

Register	Offset
MSI_Message_Address_Register	54h

Function

This register is supported only for EP mode.

Diagram



Fields

Field	Function
31-2 Message_Address	Message Address System-specified message address
1-0 —	Reserved. Always returns 00 on reads; write operations have no effect.

18.4.1.30 PCI Express MSI Message Upper Address Register (MSI_Message_Upper_Address_Register)

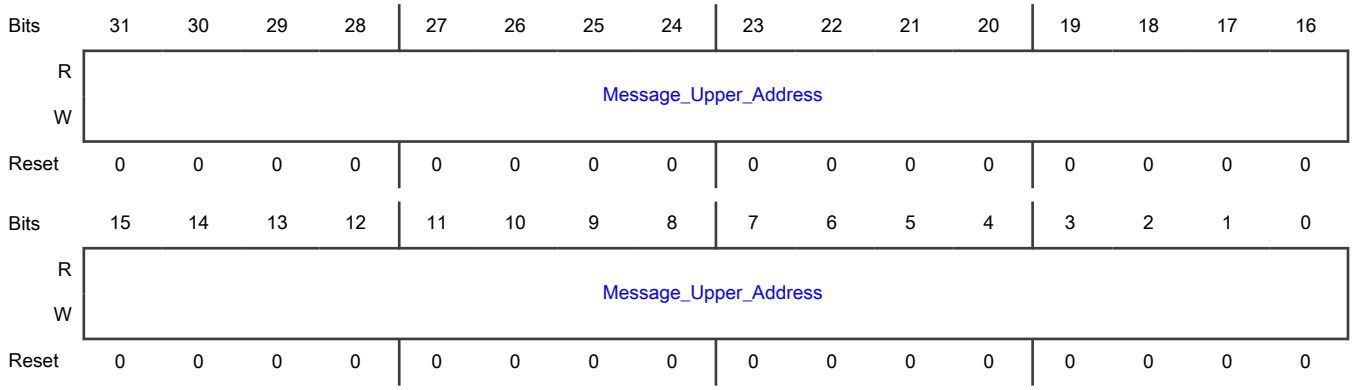
Offset

Register	Offset
MSI_Message_Upper_Address_Register	58h

Function

This register is supported only for EP mode.

Diagram



Fields

Field	Function
31-0 Message_Upper_Address	Message Address (upper portion) System-specified message upper address

18.4.1.31 PCI Express MSI Message Data Register (MSI_Message_Data_Register)

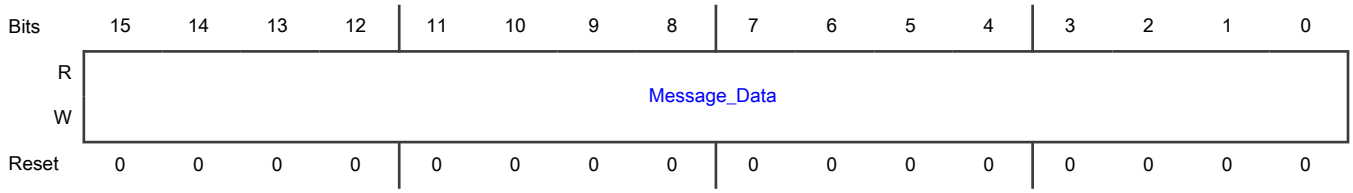
Offset

Register	Offset
MSI_Message_Data_Register	5Ch

Function

This register is supported only for EP mode.

Diagram



Fields

Field	Function
15-0 Message_Data	Data System-specified message data.

18.4.1.32 PCI Express Capability ID Register (Capability_ID_Register)

Offset

Register	Offset
Capability_ID_Register	70h

Diagram

Bits	7	6	5	4	3	2	1	0
R	PCI_Express_Capability_ID							
W								
Reset	0	0	0	1	0	0	0	0

Fields

Field	Function
7-0 PCI_Express_Capability_ID	Capability ID PCI Express = 0x10

18.4.1.33 PCI Express Capabilities Register (Capabilities_Register)

Offset

Register	Offset
Capabilities_Register	72h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		Interrupt_Message_Number					Slot	Device_Port_Type				Capability_Version			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fields

Field	Function
15-14 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-9 Interrupt_Message_Number	Interrupt Message Number If this function is allocated more than one MSI interrupt number, then this register is required to contain the offset between the base Message Data and the MSI Message that is generated when any of the status bits in either the Slot Status register or the Root Port Status register, of this capability structure, are set.
8 Slot	Slot Implemented Reserved
7-4 Device_Port_Type	Device/Port Type 0000b - (EP mode)
3-0 Capability_Version	Capability Version Indicates the defined PCI Express capability structure version number.

18.4.1.34 PCI Express Device Capabilities Register (Device_Capabilities_Register)

Offset

Register	Offset
Device_Capabilities_Register	74h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved			FLRC	CSPLS		CSPLV						Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RBER	Reserved			EP_L1_LAT			EP_L0s_LAT			ET	PHAN_FCT		MAX_PL_SIZE_SUP		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Fields

Field	Function
31-29 —	Reserved
28 FLRC	Function Level Reset Capability For non-SR-IOV capable EP mode, the reset value for this bit is 0.
27-26 CSPLS	Captured Slot Power Limit Scale
25-18 CSPLV	Captured Slot Power Limit Value For EP mode, the reset value for this field is determined by received Set_Slot_Power_Limit messages.
17-16 —	Reserved
15 RBER	Role-Based Error Reporting
14-12 —	Reserved. System software must ignore the value read from these bits. System software is permitted to write any value to these bits.
11-9 EP_L1_LAT	Endpoint L1 Acceptable Latency
8-6 EP_L0s_LAT	Endpoint L0s Acceptable Latency
5 ET	Extended Tag Field Supported
4-3 PHAN_FCT	Phantom Functions Supported
2-0 MAX_PL_SIZE_SUP	Max_Payload_Size Supported Maximum payload size supported. 001 = 256-bytes

18.4.1.35 PCI Express Device Control Register (Device_Control_Register)

Offset

Register	Offset
Device_Control_Register	78h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IFLR	MAX_READ_SIZE			ENS	APE	PFE	ETE	MAX_PAYLOAD_SIZE			RO	URR	FER	NFER	CER
Reset	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0

Fields

Field	Function
15 IFLR	Initiate Function Level Reset
14-12 MAX_READ_SIZE	Maximum_Read_Request_Size
11 ENS	Enable No Snoop
10 APE	AUX Power PM Enable
9 PFE	Phantom Functions Enable
8 ETE	Extended Tag Field Enable
7-5 MAX_PAYLOAD_SIZE	Max_Payload_Size Maximum payload size
4 RO	Enable Relaxed Ordering
3 URR	Unsupported Request Reporting Enable
2 FER	Fatal Error Reporting Enable
1 NFER	Non-Fatal Error Reporting Enable
0 CER	Correctable Error Reporting Enable

18.4.1.36 PCI Express Device Status Register (Device_Status_Register)

Offset

Register	Offset
Device_Status_Register	7Ah

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved										TP	APD	URD	FED	NFED	CED
W													W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

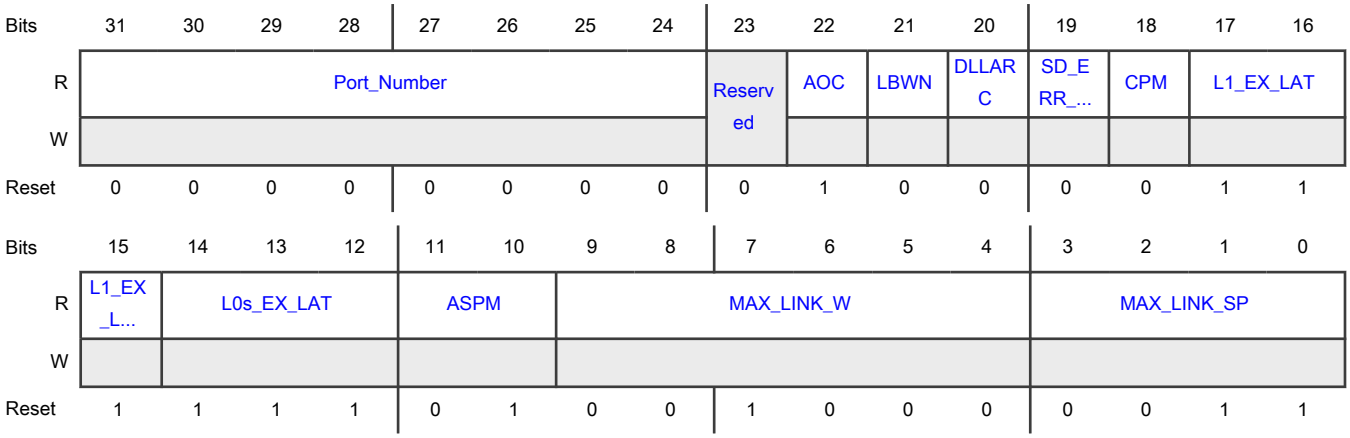
Field	Function
15-6 —	Reserved
5 TP	Transactions Pending
4 APD	AUX Power Detected
3 URD	Unsupported Request Detected
2 FED	Fatal Error Detected
1 NFED	Non-Fatal Error Detected
0 CED	Correctable Error Detected

18.4.1.37 PCI Express Link Capabilities Register (Link_Capabilities_Register)

Offset

Register	Offset
Link_Capabilities_Register	7Ch

Diagram



Fields

Field	Function
31-24 Port_Number	Port Number This field indicates the PCI Express Port number for the given PCI Express Link
23 —	Reserved
22 AOC	ASPM Optionality Compliance Software is permitted to use the value of this bit to help determine whether to enable ASPM or whether to run ASPM compliance tests.
21 LBWN	Link Bandwidth Notification Capability In EP-mode, this bit is hardwired to 0.
20 DLLARC	Data Link Layer Active Reporting Capable Set to 0 when in EP.
19 SD_ERR_RPT_CAP	Surprise Down Error Reporting Capable

Table continues on the next page...

Table continued from the previous page...

Field	Function
18 CPM	Clock Power Management
17-15 L1_EX_LAT	L1 Exit Latency
14-12 L0s_EX_LAT	L0s Exit Latency
11-10 ASPM	Active State Power Management (ASPM) Support
9-4 MAX_LINK_W	Maximum Link Width
3-0 MAX_LINK_SP	Maximum Link Speed 0001b - 2.5 GT/s link 0010b - Reserved 0011b - 8.0 GT/s

18.4.1.38 PCI Express Link Control Register (Link_Control_Register)

Offset

Register	Offset
Link_Control_Register	80h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				LABIE	LBMIE	HW_AUTO...	ECPM	EXT_SYNC	CCC	RL	LD	RCB	Reserved	ASPM_CTL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Fields

Field	Function
15-12 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 LABIE	Link Autonomous Bandwidth Interrupt Enable
10 LBMIE	Link Bandwidth Management Interrupt Enable
9 HW_AUTO_WIDTH_DIS	Hardware Autonomous Width Disable
8 ECPM	Enable Clock Power Management
7 EXT_SYNC	Extended Synch
6 CCC	Common Clock Configuration
5 RL	Retrain Link (Reserved for EP devices).
4 LD	Link Disable (Reserved for EP devices)
3 RCB	Read Completion Boundary (RCB)
2 —	Reserved
1-0 ASPM_CTL	Active State Power Management (ASPM) Control

18.4.1.39 PCI Express Link Status Register (Link_Status_Register)

Offset

Register	Offset
Link_Status_Register	82h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LABS	LBMS	Reserved	SCC	LT	Reserved	NEG_LINK_W						LINK_SP			
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Fields

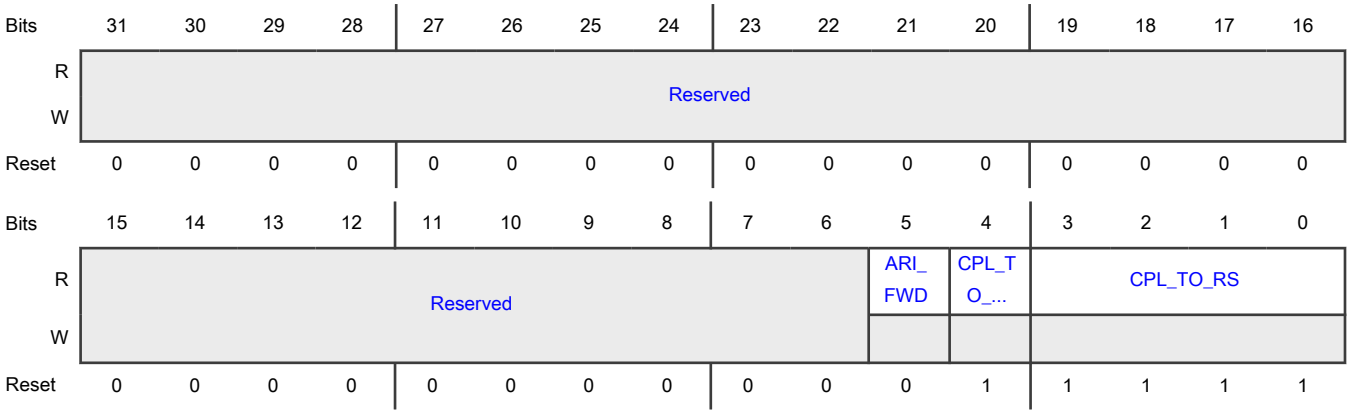
Field	Function
15 LABS	Link Autonomous Bandwidth Status
14 LBMS	Link Bandwidth Management Status
13 —	Reserved
12 SCC	Slot Clock Configuration
11 LT	Link Training This bit is reserved in the EP mode.
10 —	Reserved.
9-4 NEG_LINK_W	Negotiated link width All other encodings are reserved. The value in this field is undefined when the link is not up. 000001b - x1 000010b - x2
3-0 LINK_SP	Current Link Speed 0001b - 2.5 GT/s 0010b - Reserved 0011b - 8.0 GT/s

18.4.1.40 PCI Express Device Capabilities 2 Register (Device_Capabilities_2_Register)

Offset

Register	Offset
Device_Capabilities_2_Register	94h

Diagram



Fields

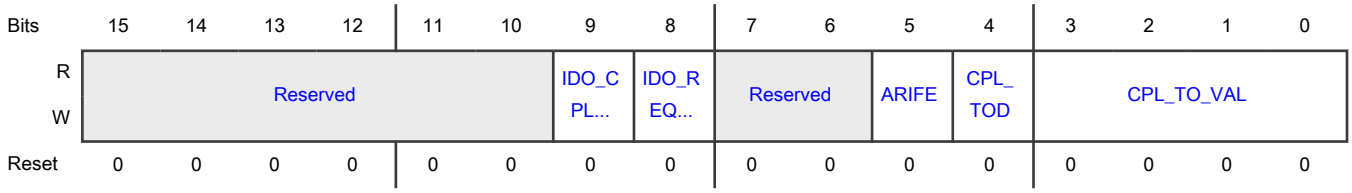
Field	Function
31-6 —	Reserved
5 ARI_FWD	ARI Forwarding Supported
4 CPL_TO_DS	Completion Timeout Disable Supported
3-0 CPL_TO_RS	Completion Timeout Ranges Supported

18.4.1.41 PCI Express Device Control 2 Register (Device_Control_2_Register)

Offset

Register	Offset
Device_Control_2_Register	98h

Diagram



Fields

Field	Function
15-10 —	Reserved
9 IDO_CPL_EN	IDO Completion Enable
8 IDO_REQ_EN	IDO Request Enable
7-6 —	Reserved
5 ARIFE	ARI Forwarding Enable Must be set when RC is communicating with ARI devices. When set will allow RC to issue configuration type 0 cycles with device number != 0.
4 CPL_TOD	Completion Timeout Disable
3-0 CPL_TO_VAL	Completion Timeout Value

18.4.1.42 PCI Express Link Capabilities 2 Register (Link_Capabilities_2_Register)

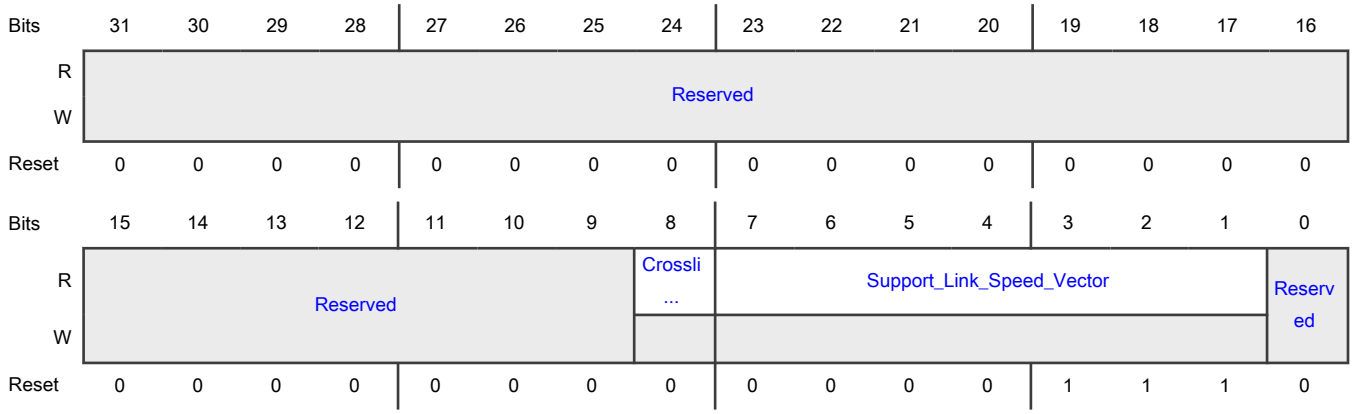
Offset

Register	Offset
Link_Capabilities_2_Register	9Ch

Function

The PCI Express link capability 2 register is shown below.

Diagram



Fields

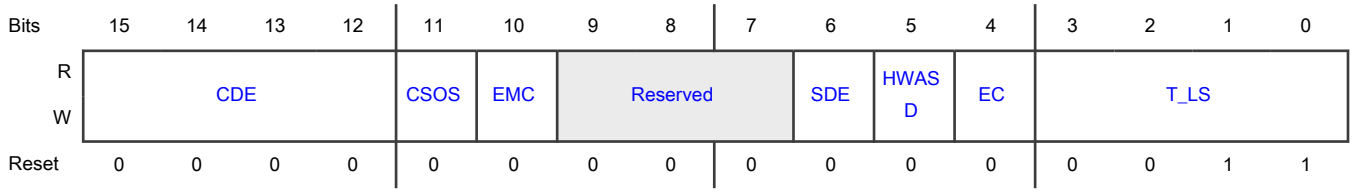
Field	Function
31-9 —	Reserved
8 Crosslink_Supported	Crosslink Supported
7-1 Support_Link_Speed_Vector	Supported Link Speeds Vector This field indicates the supported Link speed(s) of the associated Port. For each bit, a value of 1 indicates that the corresponding Link speed is supported; otherwise, the Link speed is not supported. Bit definitions are: Bit 1 2.5 GT/s Bit 2 Reserved Bit 3 8.0 GT/s Bits 7:4 Reserved
0 —	Reserved

18.4.1.43 PCI Express Link Control 2 Register (Link_Control_2_Register)

Offset

Register	Offset
Link_Control_2_Register	A0h

Diagram



Fields

Field	Function
15-12 CDE	Compliance Preset/De-emphasis
11 CSOS	Compliance SOS
10 EMC	Enter Modified Compliance
9-7 Reserved	Reserved
6 SDE	Selectable De-emphasis This bit is reserved in the EP mode
5 HWASD	Hardware Autonomous Speed Disable
4 EC	Enter Compliance
3-0 T_LS	Target Link Speed

18.4.1.44 PCI Express Link Status 2 Register (Link_Status_2_Register)

Offset

Register	Offset
Link_Status_2_Register	A2h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved										LER	EP3S	EP2S	EP1S	EC	DE_LVL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-6 —	Reserved
5 LER	Link Equalization Request
4 EP3S	Equalization Phase 3 Successful
3 EP2S	Equalization Phase 2 Successful
2 EP1S	Equalization Phase 1 Successful
1 EC	Equalization Complete
0 DE_LVL	Current De-emphasis Level

18.4.1.45 PCI Express Advanced Error Reporting Capability ID Register (Advanced_Error_Reporting_Capability_ID_Register)

Offset

Register	Offset
Advanced_Error_Reporting_Capability_ID_Register	100h

Diagram

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

Fields

Field	Function
15-0 Capability_ID	PCI Express Extended Capability ID 0x0001 = Advanced error reporting capability

18.4.1.46 PCI Express Uncorrectable Error Status Register (Uncorrectable_Error_Status_Register)

Offset

Register	Offset
Uncorrectable_Error_Status_Register	104h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved											URE	ECRCE	MTLP	RXO	UC
W												W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CA	CTO	FCPE	PTLP	Reserved							DLPE	Reserved			
W	W1C	W1C	W1C	W1C								W1C				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-21 —	Reserved
20 URE	Unsupported request error status.

Table continues on the next page...

Table continued from the previous page...

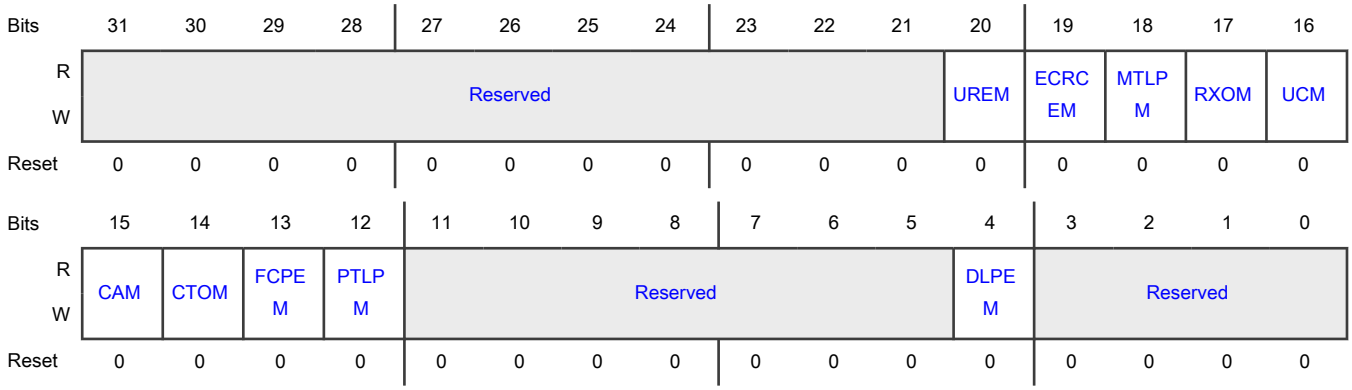
Field	Function
19 ECRCE	ECRC error status.
18 MTLP	Malformed TLP status.
17 RXO	Receiver overflow status.
16 UC	Unexpected completion status.
15 CA	Completer abort status.
14 CTO	Completion timeout status. Note that a completion timeout error is a fatal error. If a completion timeout error is detected, the system has become unstable. Hot reset is recommended to restore stability of the system.
13 FCPE	Flow control protocol error status.
12 PTLP	Poisoned TLP status.
11-5 —	Reserved
4 DLPE	Data link protocol error status.
3-0 —	Reserved

18.4.1.47 PCI Express Uncorrectable Error Mask Register (Uncorrectable_Error_Mask_Register)

Offset

Register	Offset
Uncorrectable_Error_Mask_Register	108h

Diagram



Fields

Field	Function
31-21 —	Reserved
20 UREM	Unsupported request error mask.
19 ECRCEM	ECRC error mask.
18 MTLPM	Malformed TLP mask.
17 RXOM	Receiver overflow mask.
16 UCM	Unexpected completion mask.
15 CAM	Completer abort mask.
14 CTOM	Completion timeout mask.
13 FCPEM	Flow control protocol error mask.
12 PTLPM	Poisoned TLP mask.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-5 —	Reserved
4 DLPEM	Data link protocol error mask.
3-0 —	Reserved

18.4.1.48 PCI Express Uncorrectable Error Severity Register (Uncorrectable_Error_Severity_Register)

Offset

Register	Offset
Uncorrectable_Error_ Severity_Register	10Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved											URES	ECRCES	MTLPS	RXOS	UCS
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CAS	CTOS	FCPES	PTLPS	Reserved							DLPES	Reserved			
Reset	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0

Fields

Field	Function
31-21 —	Reserved
20 URES	Unsupported request error severity.
19	ECRC error severity.

Table continues on the next page...

Table continued from the previous page...

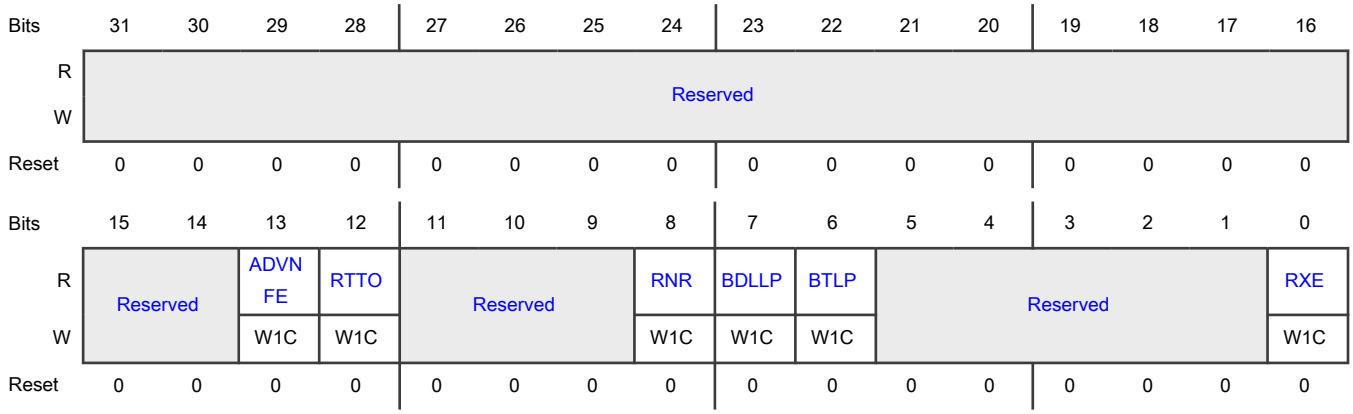
Field	Function
ECRCES	
18 MTLPS	Malformed TLP severity.
17 RXOS	Receiver overflow severity.
16 UCS	Unexpected completion severity.
15 CAS	Completer abort severity.
14 CTOS	Completion timeout severity.
13 FCPES	Flow control protocol error severity.
12 PTLPS	Poisoned TLP severity.
11-5 —	Reserved
4 DLPES	Data link protocol error severity.
3-0 —	Reserved

18.4.1.49 PCI Express Correctable Error Status Register (Correctable_Error_Status_Register)

Offset

Register	Offset
Correctable_Error_Status_Register	110h

Diagram



Fields

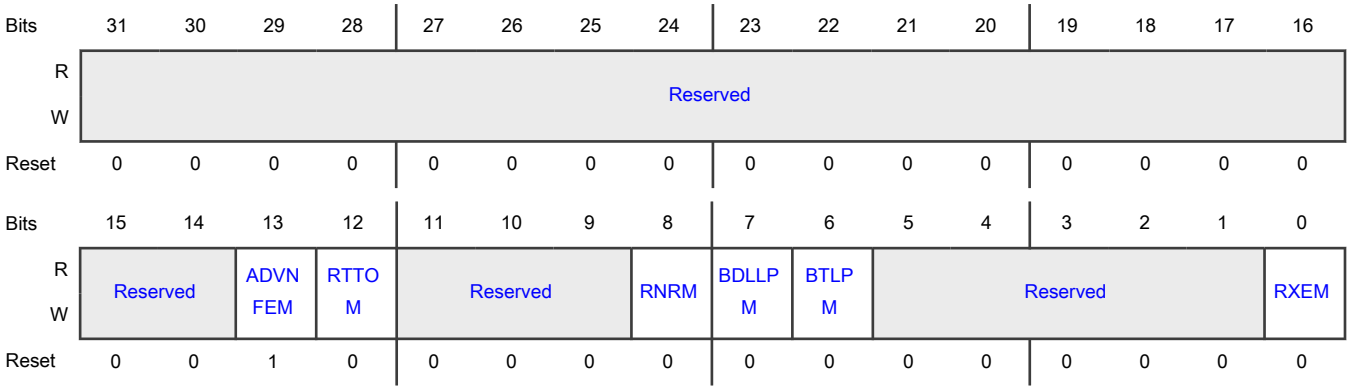
Field	Function
31-14 —	Reserved
13 ADVNFE	Advisory Non-Fatal Error Status indicates the occurrence of the advisory error, and the Advisory Non-Fatal Error Mask bit in the #unique_475/unique_475_Connect_42_Correctable_Error_Mask_Register is checked to determine whether to proceed further with logging and signaling
12 RTTO	Replay timer timeout status
11-9 —	Reserved
8 RNR	REPLAY_NUM Rollover status
7 BDLLP	Bad DLLP status
6 BTLP	Bad TLP status
5-1 —	Reserved
0 RXE	Receiver error status

18.4.1.50 PCI Express Correctable Error Mask Register (Correctable_Error_Mask_Register)

Offset

Register	Offset
Correctable_Error_Mask_Register	114h

Diagram



Fields

Field	Function
31-14 —	Reserved
13 ADVNFEM	Advisory non-fatal error mask. This bit is Set by default to enable compatibility with software that does not comprehend Role-Based Error Reporting
12 RTTOM	Replay timer timeout mask
11-9 —	Reserved
8 RNRM	REPLAY_NUM Rollover mask
7 BDLLPM	Bad DLLP mask
6 BTLPM	Bad TLP mask

Table continues on the next page...

Table continued from the previous page...

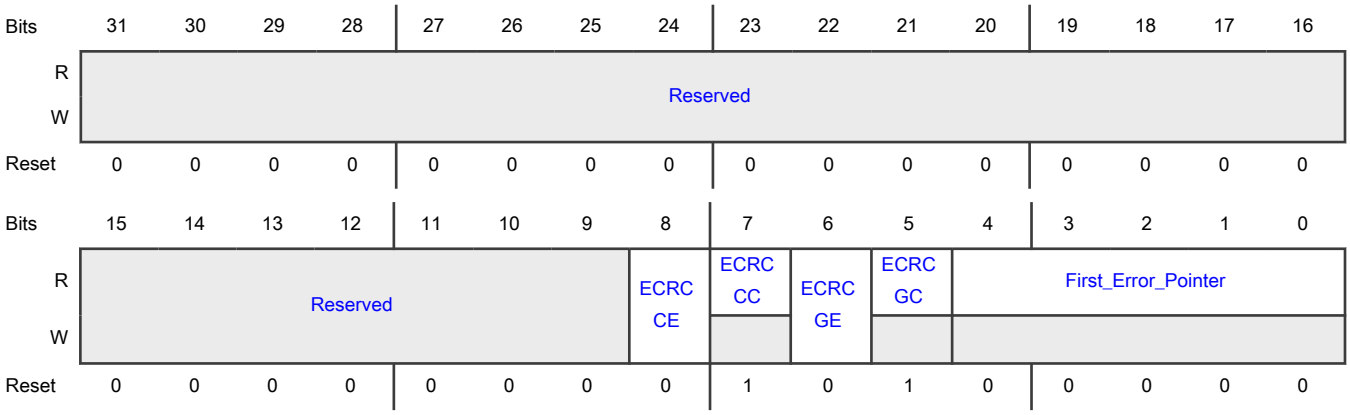
Field	Function
5-1 —	Reserved
0 RXEM	Receiver error mask

18.4.1.51 PCI Express Advanced Error Capabilities and Control Register (Advanced_Error_Capabilities_and_Control_Register)

Offset

Register	Offset
Advanced_Error_Capabilities_and_Control_Register	118h

Diagram



Fields

Field	Function
31-9 —	Reserved
8 ECRCCE	ECRC checking enable.
7 ECRCCC	ECRC checking capable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
6 ECRCGE	ECRC generation enable.
5 ECRCGC	ECRC generation capable.
4-0 First_Error_Pointer	The First Error Pointer is a read-only register that identifies the bit position of the first error reported in the Uncorrectable Error Status register.

18.4.1.52 PCI Express Header Log Register 1 (Header_Log_Register_DWORD1)

Offset

Register	Offset
Header_Log_Register_DWORD1	11Ch

Function

The first DWORD of the PCI Express header log register is shown in the figure below.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Byte_0								Byte_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	Byte_2								Byte_3							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 Byte_0	Byte n of the TLP header associated with the error.

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-16 Byte_1	Byte n of the TLP header associated with the error.
15-8 Byte_2	Byte n of the TLP header associated with the error.
7-0 Byte_3	Byte n of the TLP header associated with the error.

18.4.1.53 PCI Express Header Log Register 2 (Header_Log_Register_DWORD2)

Offset

Register	Offset
Header_Log_Register_DWORD2	120h

Function

The second DWORD of the PCI Express header log register is shown in the figure below.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Byte_4								Byte_5							
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	Byte_6								Byte_7							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 Byte_4	Byte n of the TLP header associated with the error.
23-16	Byte n of the TLP header associated with the error.

Table continues on the next page...

Table continued from the previous page...

Field	Function
Byte_5	
15-8 Byte_6	Byte n of the TLP header associated with the error.
7-0 Byte_7	Byte n of the TLP header associated with the error.

18.4.1.54 PCI Express Header Log Register 3 (Header_Log_Register_DWORD3)

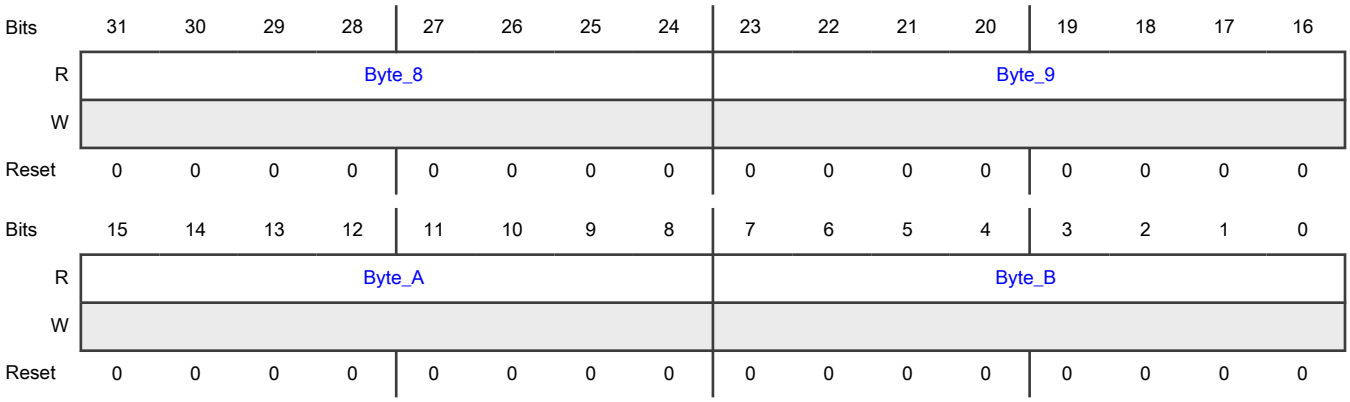
Offset

Register	Offset
Header_Log_Register_DWORD3	124h

Function

The third DWORD of the PCI Express header log register is shown in the figure below.

Diagram



Fields

Field	Function
31-24 Byte_8	Byte n of the TLP header associated with the error.
23-16 Byte_9	Byte n of the TLP header associated with the error.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-8 Byte_A	Byte n of the TLP header associated with the error.
7-0 Byte_B	Byte n of the TLP header associated with the error.

18.4.1.55 PCI Express Header Log Register 4 (Header_Log_Register_DWORD4)

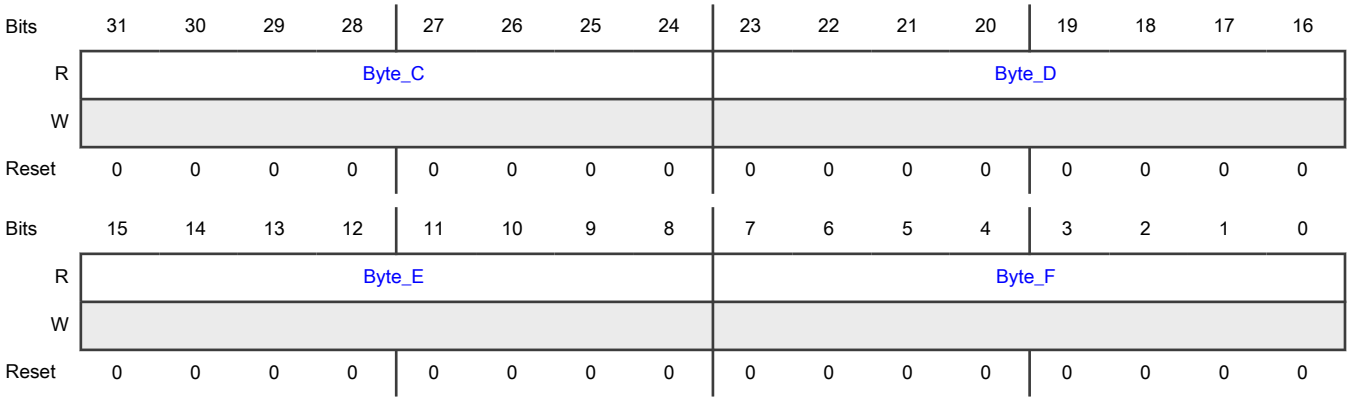
Offset

Register	Offset
Header_Log_Register_DWORD4	128h

Function

The fourth DWORD of the PCI Express header log register is shown in the figure below.

Diagram



Fields

Field	Function
31-24 Byte_C	Byte n of the TLP header associated with the error.
23-16 Byte_D	Byte n of the TLP header associated with the error.
15-8	Byte n of the TLP header associated with the error.

Table continues on the next page...

Table continued from the previous page...

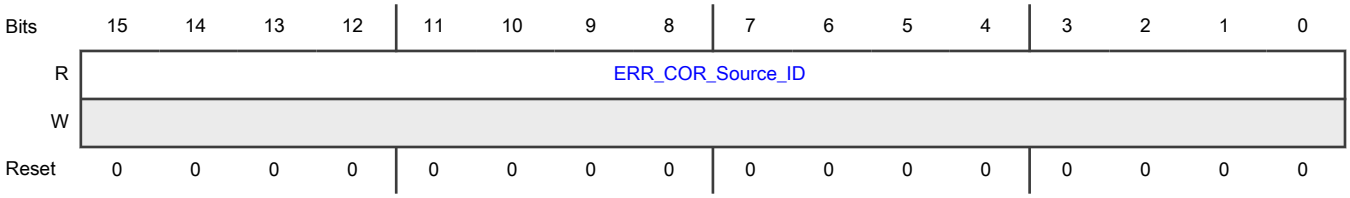
Field	Function
Byte_E	
7-0 Byte_F	Byte n of the TLP header associated with the error.

18.4.1.56 PCI Express Correctable Error Source ID Register (Correctable_Error_Source_ID_Register)

Offset

Register	Offset
Correctable_Error_Source_ID_Register	134h

Diagram



Fields

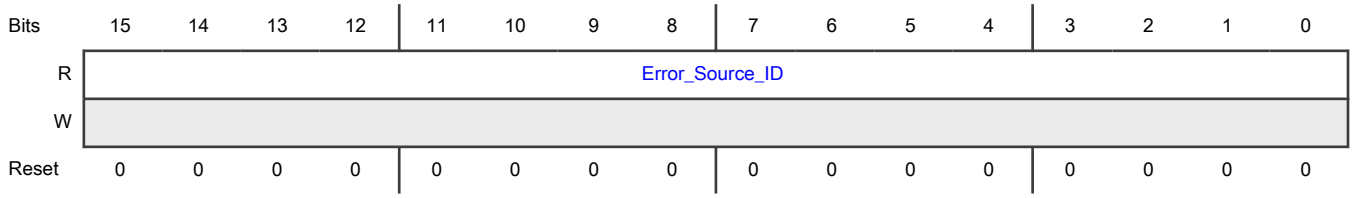
Field	Function
15-0 ERR_COR_Source_ID	Loaded with the requester ID indicated in the received ERR_COR Message when the ERR_COR Received register is not already set. Default value of this field is 0.

18.4.1.57 PCI Express Error Source ID Register (Error_Source_ID_Register)

Offset

Register	Offset
Error_Source_ID_Register	136h

Diagram



Fields

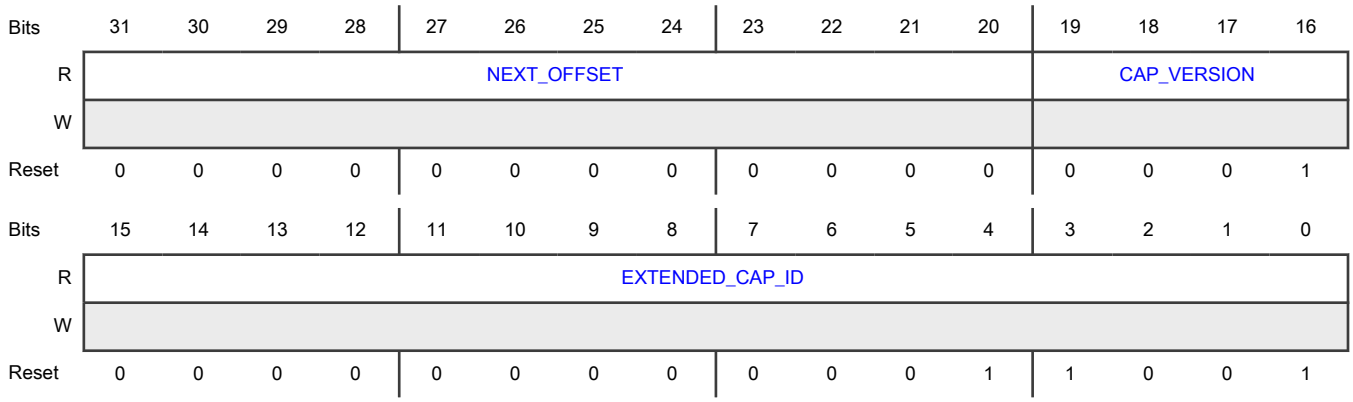
Field	Function
15-0 Error_Source_ID	ERR_FATAL/NONFATAL source ID

18.4.1.58 Secondary PCI Express Extended Capability Header (SPCIE_CAP_HEADER_REG)

Offset

Register	Offset
SPCIE_CAP_HEADER_REG	148h

Diagram



Fields

Field	Function
31-20 NEXT_OFFSET	Next Capability Offset This field is a pointer to the next capability structure.

Table continues on the next page...

Table continued from the previous page...

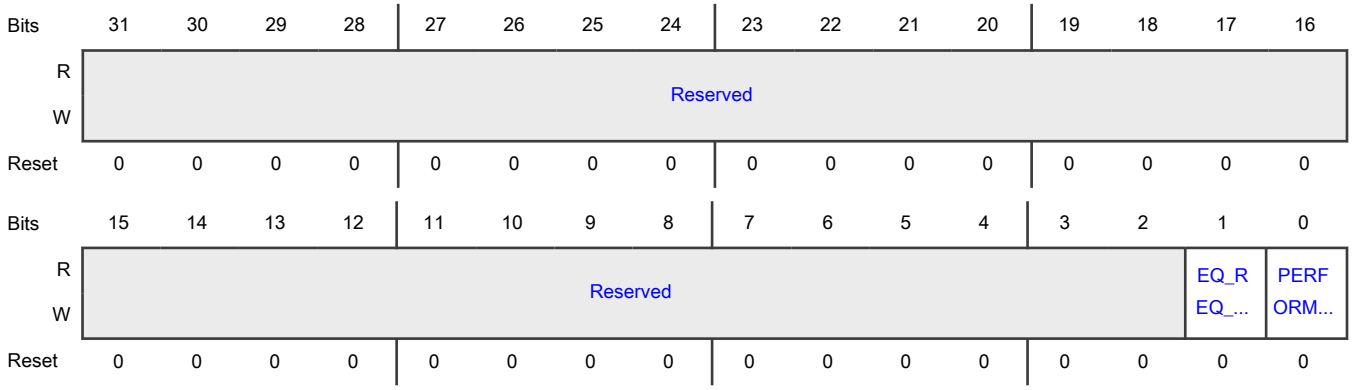
Field	Function
	NOTE The reset value of this field depends on the instantiation of the module. See PCI Express configuration registers for the specific values.
19-16 CAP_VERSION	Capability Version Indicates the version of the Capability structure present.
15-0 EXTENDED_CAP_ID	PCI Express Extended Capability ID 0019h = Secondary PCI Express extended capability

18.4.1.59 Link Control 3 Register (LINK_CONTROL3_REG)

Offset

Register	Offset
LINK_CONTROL3_REG	14Ch

Diagram



Fields

Field	Function
31-2 —	Reserved
1 EQ_REQ_INT_EN	This bit is reserved in the EP mode since the crosslink feature is not supported.

Table continues on the next page...

Table continued from the previous page...

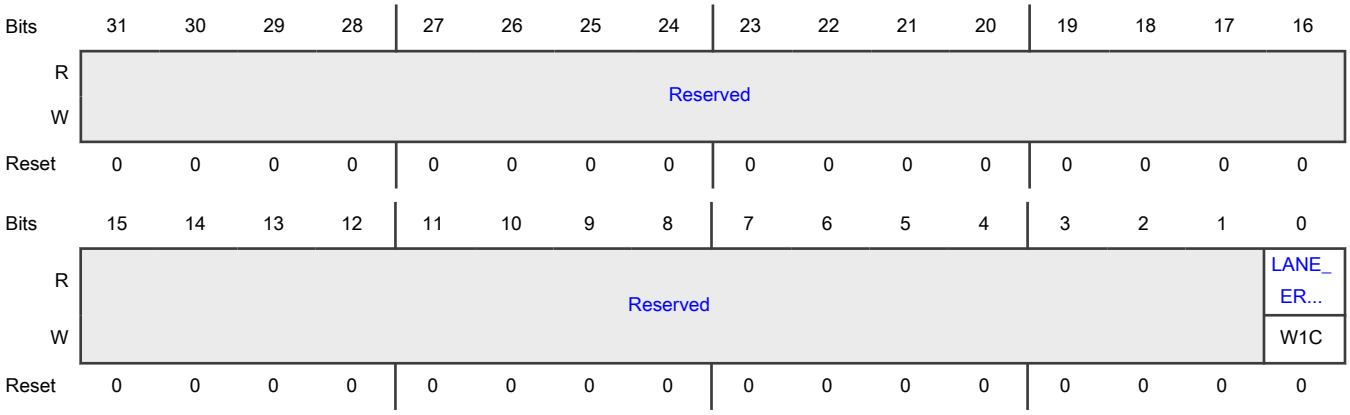
Field	Function
0 PERFORM_EQ	This bit is reserved in the EP mode since the crosslink feature is not supported.

18.4.1.60 Lane Error Status Register (LANE_ERR_STATUS_REG)

Offset

Register	Offset
LANE_ERR_STATUS_REG	150h

Diagram



Fields

Field	Function
31-1 —	Reserved
0 LANE_ERR_STATUS	Lane Error Status Each bit indicates if the corresponding Lane detected a Lane-based error.

18.4.1.61 Lane Equalization Control Register (LANE0_EQUALIZATION_CONTROL)

Offset

Register	Offset
LANE0_EQUALIZATION_CONTROL	154h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	USP_RX_PRESET_HINT			USP_TX_PRESET				Reserved	DSP_RX_PRESET_HINT			DSP_TX_PRESET			
W																
Reset	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1

Fields

Field	Function
15 —	Reserved
14-12 USP_RX_PRESET_HINT	Upstream Port Receiver Preset Hint
11-8 USP_TX_PRESET	Upstream Port Transmitter Preset
7 —	Reserved
6-4 DSP_RX_PRESET_HINT	Downstream Port Receiver Preset Hint
3-0 DSP_TX_PRESET	Downstream Port Transmitter Preset

18.4.1.62 Symbol Timer Register and Filter Mask 1 Register (SYMBOL_TIMER_FILTER_1_OFF)

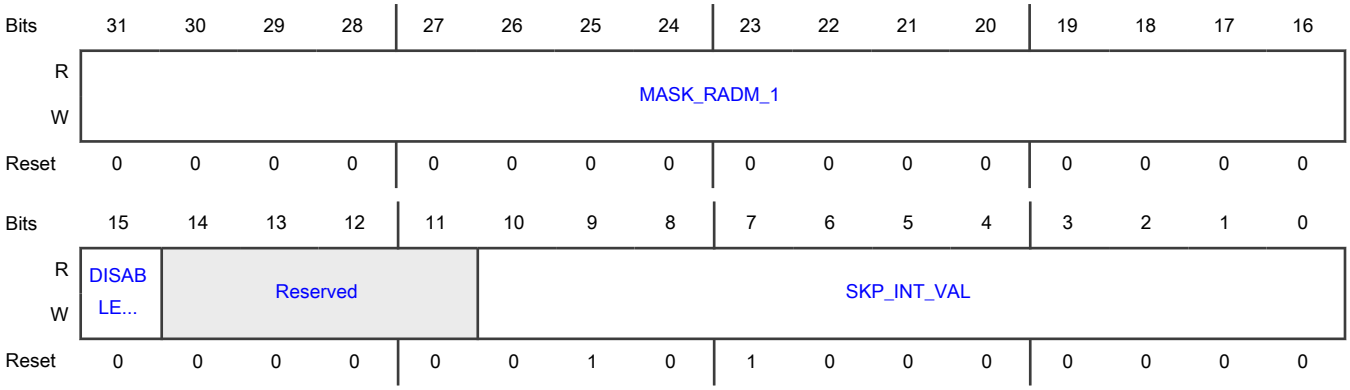
Offset

Register	Offset
SYMBOL_TIMER_FILTER_1_OFF	71Ch

Function

The Filter Mask 1 Register modifies the receive filtering and error handling rules. In each case, '0' applies the associated filtering rule and '1' masks the associated filtering rule

Diagram



Fields

Field	Function
31-16 MASK_RADM_1	Filter Mask 1 The Filter Mask 1 Register modifies the receive filtering and error handling rules. In each case, '0' applies the associated filtering rule and '1' masks the associated filtering rule. [31]: CX_FLT_MASK_RC_CFG_DISCARD 0: For RC filter to not allow CFG transaction being received 1: For RC filter to allow CFG transaction being received [30]: CX_FLT_MASK_RC_IO_DISCARD 0: For RC filter to not allow IO transaction being received 1: For RC filter to allow IO transaction being received [29]: CX_FLT_MASK_MSG_DROP 0: Drop MSG TLP (except for Vendor MSG). 1: Do not Drop MSG (except for Vendor MSG). Send message TLPs to your application. This bit only controls message TLPs other than Vendor MSGs. Vendor MSGs are controlled by Filter Mask Register 2, bits [1:0].

Table continues on the next page...

Field	Function
	NOTE The forwarding of message TLPs may be undesirable. If so, write a 0 to this bit to disable forwarding of message TLPs. [28]: CX_FLT_MASK_CPL_ECRC_DISCARD Only used when completion queue is advertized with infinite credits and is in store-and-forward mode. 0: Discard completions with ECRC errors 1: Allow completions with ECRC errors to be passed up Reserved field for SW. [27]: CX_FLT_MASK_ECRC_DISCARD 0: Discard TLPs with ECRC errors 1: Allow TLPs with ECRC errors to be passed up [26]: CX_FLT_MASK_CPL_LEN_MATCH 0: Enforce length match for completions; a violation results in cpl_abort, and possibly AER of unexp_cpl_err 1: MASK length match for completions [25]: CX_FLT_MASK_CPL_ATTR_MATCH 0: Enforce attribute match for completions; a violation results in a malformed TLP error, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca 1: Mask attribute match for completions [24]: CX_FLT_MASK_CPL_TC_MATCH 0: Enforce Traffic Class match for completions; a violation results in a malformed TLP error, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca 1: Mask Traffic Class match for completions [23]: CX_FLT_MASK_CPL_FUNC_MATCH 0: Enforce function match for completions; a violation results in cpl_abort, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca 1: Mask function match for completions [22]: CX_FLT_MASK_CPL_REQID_MATCH 0: Enforce Req. Id match for completions; a violation result in cpl_abort, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca 1: Mask Req. Id match for completions [21]: CX_FLT_MASK_CPL_TAGERR_MATCH 0: Enforce Tag Error Rules for completions; a violation result in cpl_abort, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca 1: Mask Tag Error Rules for completions [20]: CX_FLT_MASK_LOCKED_RD_AS_UR 0: Treat locked Read TLPs as UR for EP.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1: Treat locked Read TLPs as Supported for EP. [19]: CX_FLT_MASK_CFG_TYPE1_RE_AS_UR 0: Treat CFG type1 TLPs as UR for EP. 1: Treat CFG type1 TLPs as Supported for EP. [18]: CX_FLT_MASK_UR_OUTSIDE_BAR 0: Treat out-of-bar TLPs as UR 1: Do not treat out-of-bar TLPs as UR [17]: CX_FLT_MASK_UR_POIS 0: Treat poisoned request TLPs as UR 1: Do not treat poisoned request TLPs as UR The native PEX controller always passes poisoned completions to your application. [16]: CX_FLT_MASK_UR_FUNC_MISMATCH 0: Treat Function MisMatched TLPs as UR 1: Do not treat Function MisMatched TLPs as UR *Note*: This register field is sticky.
15 DISABLE_FC_WD_TIMER	Disable FC Watchdog Timer Disable FC Watchdog Timer. *Note*: This register field is sticky.
14-11 —	Reserved
10-0 SKP_INT_VAL	SKP Interval Value The number of symbol times to wait between transmitting SKP ordered sets. Note that the PEX controller actually waits the number of symbol times in this register plus 1 between transmitting SKP ordered sets. Your application must program this register accordingly. For example, if 1536 were programmed into this register (in a 250 MHz PEX controller), then the PEX controller actually transmits SKP ordered sets once every 1537 symbol times. The value programmed to this register is actually clock ticks and not symbol times. In a 125 MHz PEX controller, programming the value programmed to this register should be scaled down by a factor of 2 (because one clock tick = two symbol times in this case). *Note:*This value is not used at Gen3 speed; the skip interval is hardcoded to 370 blocks. *Note*: This register field is sticky.

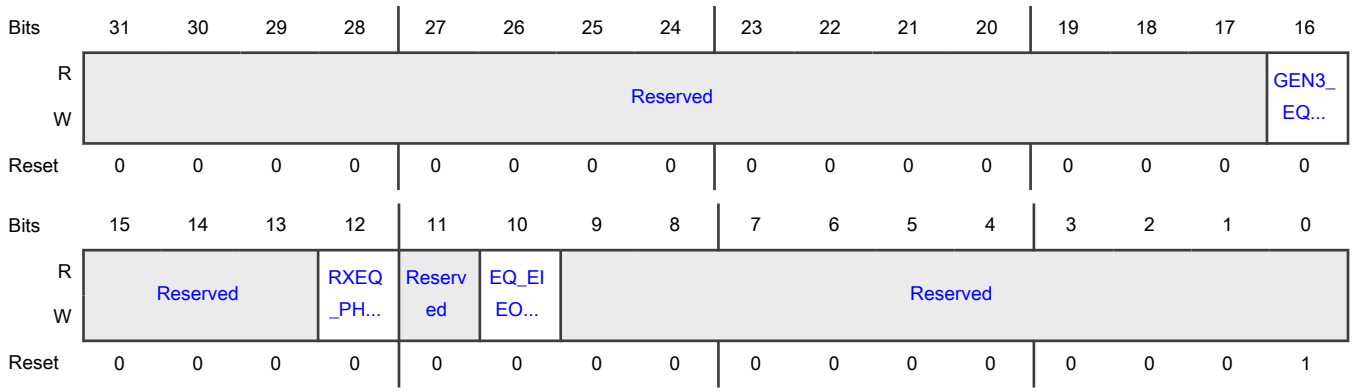
18.4.1.63 Gen3 Control Register (GEN3_RELATED_OFF)

Offset

Register	Offset
GEN3_RELATED_OFF	890h

Function

Diagram



Fields

Field	Function
31-17 —	Reserved
16 GEN3_EQUALIZATION_DISABLE	Equalization Disable Disable equalization feature. *Note*: This register field is sticky.
15-13 —	Reserved
12 RXEQ_PH01_EN	Rx Equalization Phase 0/Phase 1 Hold Enable When this bit is set the upstream port holds phase 0 (the downstream port holds phase 1) for 10ms. Holding phase 0 or phase 1 can be used to allow sufficient time for Rx Equalization to be performed by the PHY. *Note*: This register field is sticky.
11 —	Reserved
10	Equalization EIEOS Count Reset Disable

Table continues on the next page...

Table continued from the previous page...

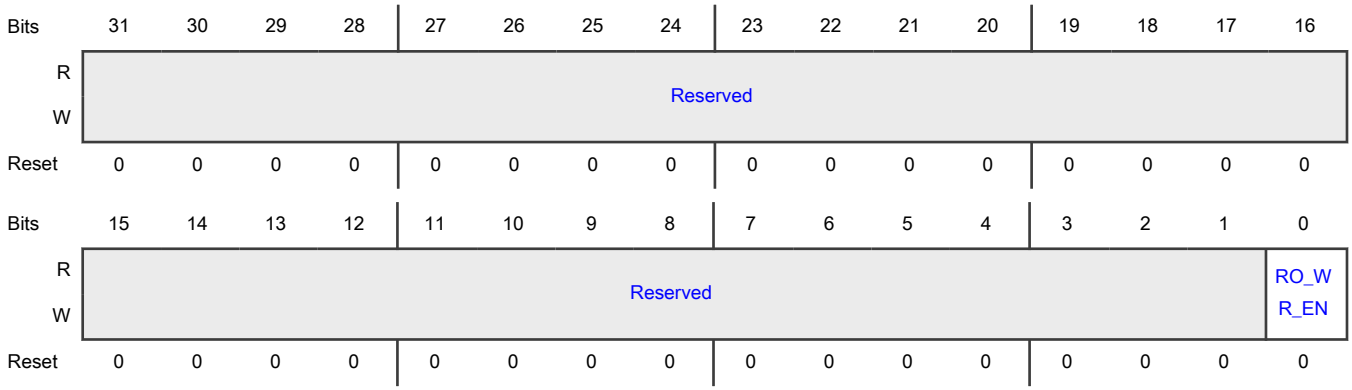
Field	Function
EQ_EIEOS_CNT	Disable requesting reset of EIEOS count during equalization. *Note*: This register field is sticky.
9-0 —	Reserved

18.4.1.64 DBI Read-only Write Enable Register (MISC_CONTROL_1_OFF)

Offset

Register	Offset
MISC_CONTROL_1_OFF	8BCh

Diagram



Fields

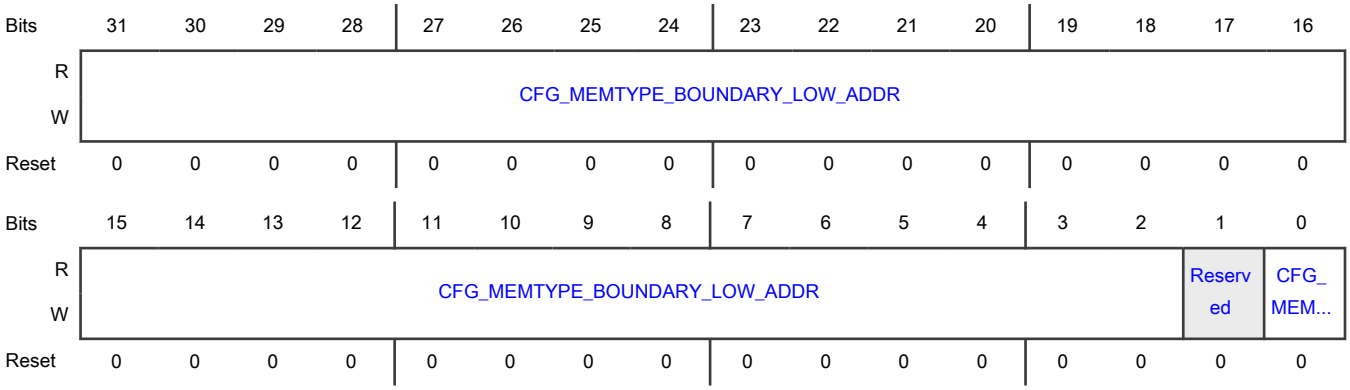
Field	Function
31-1 —	Reserved
0 RO_WR_EN	Read-only write enable This field enables writing (using internal accesses only) to some read-only and hardware-initialized bits in the configuration registers. 0b - Read-only fields are read only. 1b - Enables writing to some read-only fields

18.4.1.65 Coherency Control Register 1 (COHERENCY_CONTROL_1_OFF)

Offset

Register	Offset
COHERENCY_CONTROL_1_OFF	8E0h

Diagram



Fields

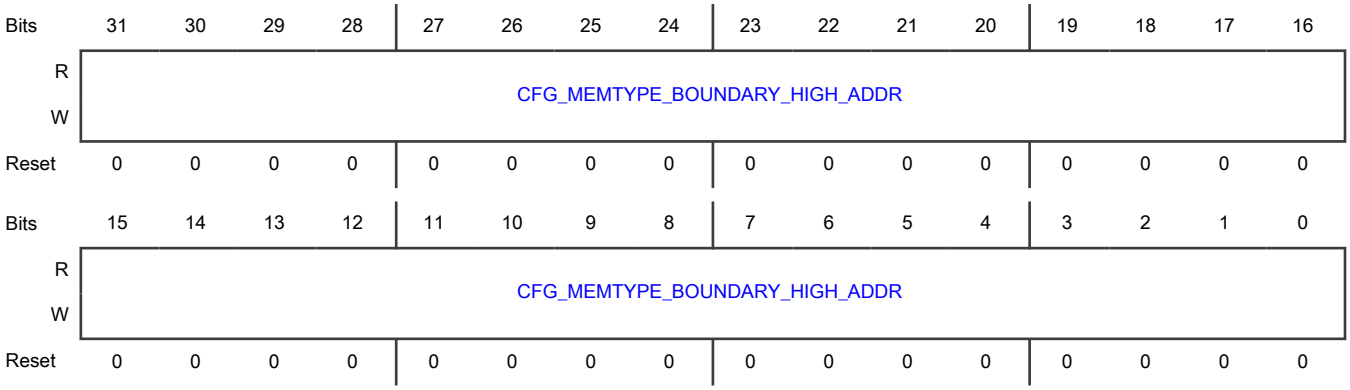
Field	Function
31-2 CFG_MEMTYPE_BOUNDARY_LOW_ADDR	Boundary lower address for memory type Bits [31:0] of dword-aligned address of the boundary for memory type. The two lower address least-significant bits must be 00. Addresses up to but not including this value are in the lower address space region; addresses equal or greater than this value are in the upper address space region. *Note*: This register field is sticky.
1 —	Reserved
0 CFG_MEMTYPE_VALUE	Memory type Sets the memory type for the lower and upper regions of the address space: *Note*: This register field is sticky. 0b - lower = CCSR; upper = Memory 1b - Reserved

18.4.1.66 Coherency Control Register 2 (COHERENCY_CONTROL_2_OFF)

Offset

Register	Offset
COHERENCY_CONTROL_2_OFF	8E4h

Diagram



Fields

Field	Function
31-0 CFG_MEMTYPE_BOUNDARY_HIGH_ADDR	Boundary upper address for memory type Bits [63:32] of the 64-bit dword-aligned address of the boundary for memory type. *Note*: This register field is sticky.

18.4.1.67 Coherency Control Register 3 (COHERENCY_CONTROL_3_OFF)

Offset

Register	Offset
COHERENCY_CONTROL_3_OFF	8E8h

Diagram

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

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

Fields

Field	Function
31-0 —	Reserved. Must be 0000_0000h.

18.4.1.68 iATU Index Register (IATU_VIEWPORT_OFF)

Offset

Register	Offset
IATU_VIEWPORT_OFF	900h

Function

The iATU registers are programmed through an indirect addressing scheme (using this index register) to reduce the address footprint in the PCI Express extended configuration space.

The iATU can remap 4 outbound and 3 inbound address regions. This index register determines the memory region to be accessed.

Diagram

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

Fields

Field	Function
31 REGION_DIR	Region Direction Defines the region being accessed as either 0: Outbound 1: Inbound
30-8 —	Reserved
7-0 REGION_INDEX	Region Index Defines which region is being accessed when writing to the control, base, limit and target registers. Must not be set to a number greater than 3 when an outbound region is being accessed. Must not be set to a value greater than 2 when an inbound region is being accessed.

18.4.1.69 iATU Region Control 1 Register (IATU_REGION_CTRL_1_OFF_INBOUND_0)

Offset

Register	Offset
IATU_REGION_CTRL_1_OFF_INBOUND_0	904h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								Reserved				CTRL_1_FUNC_NUM			
Reset	0	0	0	0	0	0	0	0	0	0	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				TYPE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-25 —	Reserved

Table continues on the next page...

Table continued from the previous page...

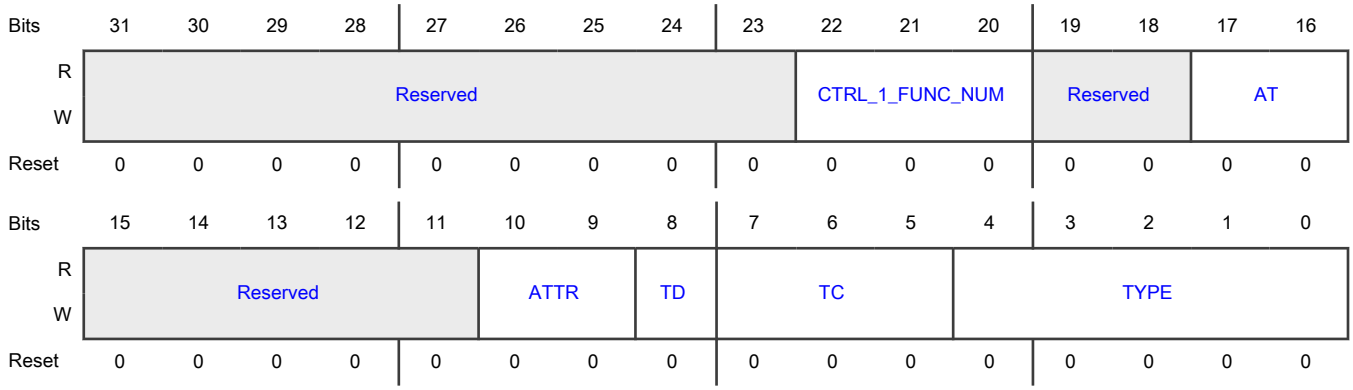
Field	Function
24-23 —	Reserved
22-20 CTRL_1_FUNC_NUM	Function Number. **MEM:** DM/EP/SW: When the Address and BAR matching logic in the PEX controller indicate that a MEM transaction matches a BAR in the function corresponding to this value, then address translation proceeds. This check is only performed if the "Function Number Match Enable" bit of the "iATU Control 2 Register" is set. RC: Reserved. You must set this bit to "0"
19-18 —	Reserved
17-16 AT	Address Translation (AT) When the TYPE field of an inbound TLP is matched to this value, then address translation proceeds (when all other enabled field-matches are successful). This check is only performed if the "AT Match Enable" bit of the iATU Control 2 Register is set.
15-11 —	Reserved
10-9 —	Reserved
8 —	Reserved
7-5 —	Reserved
4-0 TYPE	Type When the TYPE field of an inbound TLP is matched to this value, then address translation proceeds (when all other enabled field-matches are successful).

18.4.1.70 iATU Region Control 1 Register (IATU_REGION_CTRL_1_OFF_OUTBOUND_0)

Offset

Register	Offset
IATU_REGION_CTRL_1_OFF_OUTBOUND_0	904h

Diagram



Fields

Field	Function
31-23 —	Reserved
22-20 CTRL_1_FUNC_NUM	Function Number The value in this register must be 0x0. RC: Reserved. You must set this bit to "0".
19-18 —	Reserved
17-16 AT	AT When the address of an outbound TLP is matched to this region, then the AT field of the TLP is changed to the value in this register.
15-11 —	Reserved
10-9 ATTR	Attribute (Attr) When the address of an outbound TLP is matched to this region, then the ATTR field of the TLP is changed to the value in this register.
8 TD	TLP Digest (TD) When the address of an outbound TLP is matched to this region, then the TD field of the TLP is changed to the value in this register.
7-5 TC	Traffic class (TC) When the address of an outbound TLP is matched to this region, then the TC field of the TLP is changed to the value in this register.
4-0	Type

Table continues on the next page...

Table continued from the previous page...

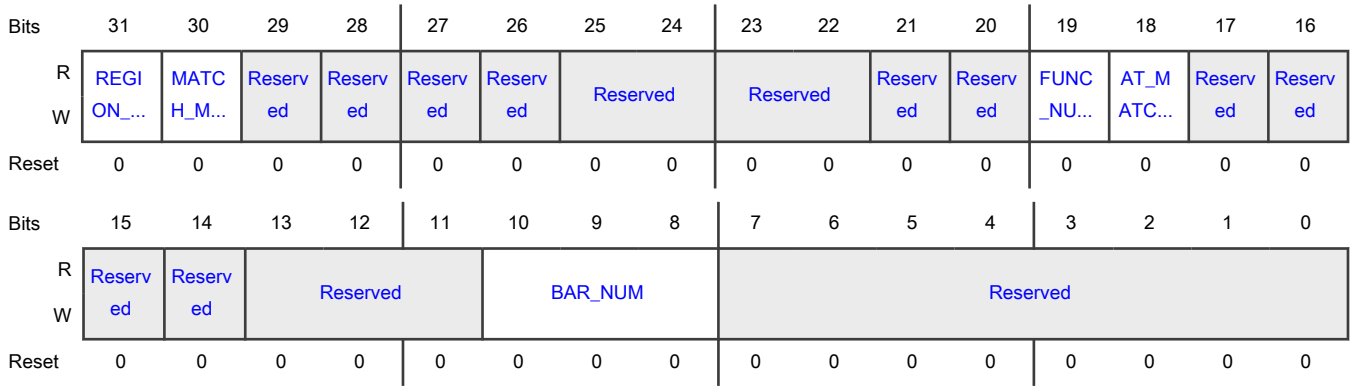
Field	Function
TYPE	When the address of an outbound TLP is matched to this region, then the TYPE field of the TLP is changed to the value in this register.

18.4.1.71 iATU Region Control 2 Register (IATU_REGION_CTRL_2_OFF_INBOUND_0)

Offset

Register	Offset
IATU_REGION_CTRL_2_OFF_INBOUND_0	908h

Diagram



Fields

Field	Function
31 REGION_EN	Region Enable This bit must be set to "1" for address translation to take place.
30 MATCH_MODE	Match Mode. Determines Inbound matching mode for TLPs. The mode depends on the type of TLP that is received as follows: For MEM-I/O TLPs, this field is interpreted as follows: 0: Address Match Mode. The iATU operates using addresses as in the outbound direction. The Region Base and Limit Registers must be setup. 1:BAR Match Mode. BAR matching is used. The "BAR Number" field is relevant. For CFG0 TLPs, this field is interpreted as follows:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0: Routing ID Match Mode. The iATU interprets the Routing ID (Bytes 8 to 11 of TLP header) as an address. This corresponds to the upper 16 bits of the address in MEM-I/O transactions. The Routing ID of the TLP must be within the base and limit of the iATU region for matching to proceed. 1: Accept Mode. The iATU accepts all CFG0 transactions as address matches. The routing ID in the CFG0 TLP is ignored. This is useful as all received CFG0 TLPs should be processed regardless of the Bus number. For MSG/MSGD TLPs, this field is interpreted as follows: 0: Address Match Mode. The iATU treats the third dword and fourth dword of the inbound MSG/MSGD TLP as an address and it is matched against the Region Base and Limit Registers. 1: Vendor ID Match Mode. This mode is relevant for ID-routed Vendor Defined Messages. The iATU ignores the Routing ID (Bus, Device, Function) in bits [31:16] of the third dword of the TLP header, but matches against the Vendor ID in bits [15:0] of the third dword of the TLP header. Bits [15:0] of the Region Upper Base register should be programmed with the required Vendor ID. The lower Base and Limit Register should be programmed to translate TLPs based on vendor specific information in the fourth dword of the TLP header.
29 —	Reserved
28 —	Reserved
27 —	Reserved
26 —	Reserved
25-24 —	Reserved
23-22 —	Reserved
21 —	Reserved
20 —	Reserved
19 FUNC_NUM_MATCH_EN	Function Number Match Enable **DM/EP:** Function Number Match Enable. Ensures that a successful Function Number TLP field comparison match (see Function Number field of the "iATU Control 1 Register") occurs (in MEM-I/O and CFG0/CFG1 transactions) for address translation to proceed.

Table continues on the next page...

Table continued from the previous page...

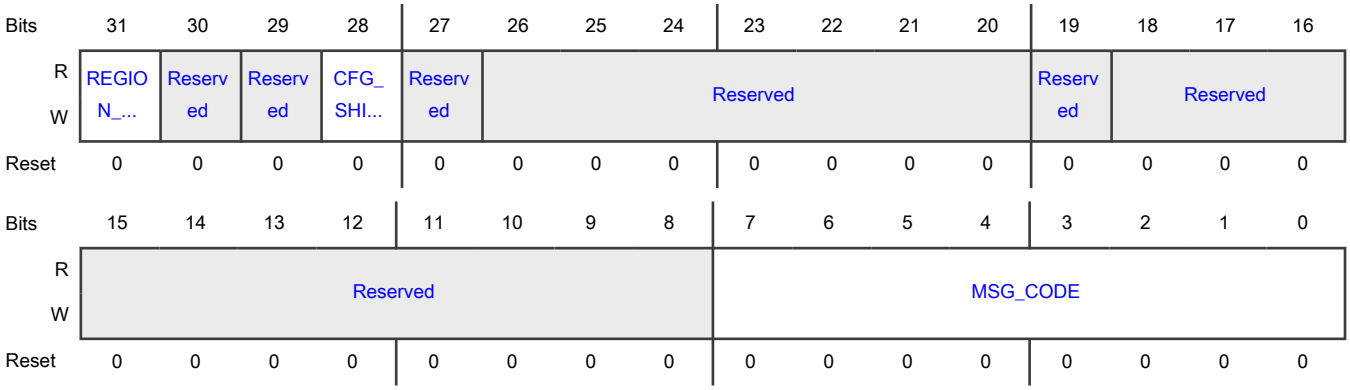
Field	Function
18 AT_MATCH_EN	AT Match Enable Ensures that a successful AT TLP field comparison match (see AT field of the "iATU Control 1 Register") occurs for address translation to proceed.
17 —	Reserved
16 —	Reserved
15 —	Reserved
14 —	Reserved
13-11 —	Reserved
10-8 BAR_NUM	BAR Number **DM/EP/SW:** When the BAR number of an inbound MEM or IO TLP that is matched by the normal internal BAR address matching mechanism is the same as this field, address translation proceeds (when all other enabled field-matches are successful). This check is only performed if the "Match Mode" bit of the "iATU Control 2 Register" is set. 000b - BAR0 001b - BAR1 010b - BAR2 011b - BAR3 100b - BAR4 101b - BAR5 110b - ROM 111b - reserved IO translation would require either 00100b or 00101b in the inbound TLP TYPE; the BAR Number set in the range 000b-101b and that BAR configured as an IO BAR.
7-0 —	Reserved

18.4.1.72 iATU Region Control 2 Register (IATU_REGION_CTRL_2_OFF_OUTBOUND_0)

Offset

Register	Offset
IATU_REGION_CTRL_2_OFF_OUTBOUND_0	908h

Diagram



Fields

Field	Function
31 REGION_EN	Region Enable This bit must be set to "1" for address translation to take place.
30 —	Reserved
29 —	Reserved
28 CFG_SHIFT_MODE	CFG Shift Mode This is useful for CFG transactions where the PCIe configuration mechanism maps bits [27:12] of the address to the bus/device and function number. This allows a CFG configuration space to be located in any 256MB window of your application memory space using a 28-bit effective address.Shifts bits [27:12] of the untranslated address to form bits [31:16] of the translated address.
27 —	Reserved
26-20 —	Reserved

Table continues on the next page...

Table continued from the previous page...

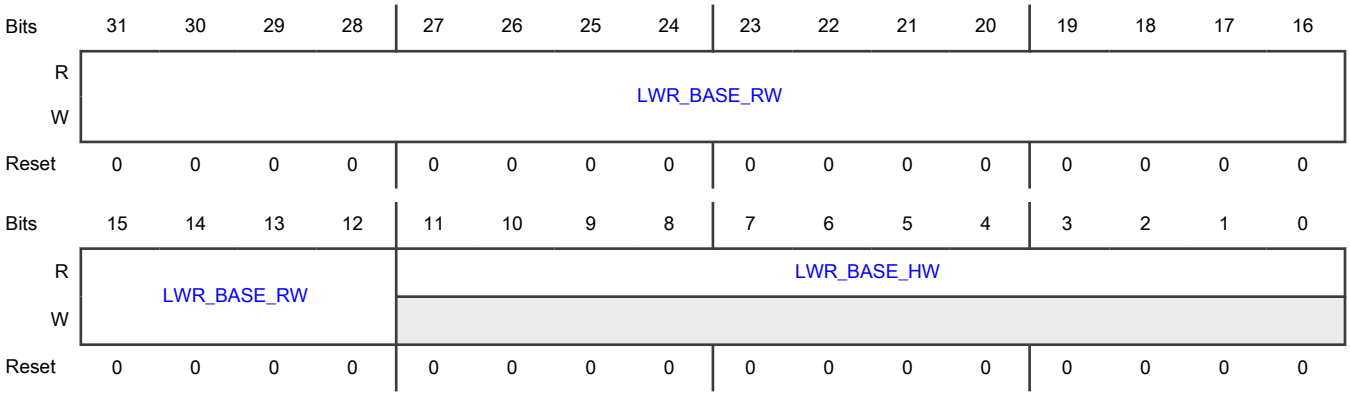
Field	Function
19 —	Reserved
18-8 —	Reserved
7-0 MSG_CODE	Message Code Message TLPs: When the address of an outbound TLP is matched to this region, and the translated TLP TYPE field is Msg or MsgD; then the message field of the TLP is changed to the value in this register.

18.4.1.73 iATU Lower Base Address Register (IATU_LWR_BASE_ADDR_OFF_INBOUND_0)

Offset

Register	Offset
IATU_LWR_BASE_ADDR_OFF_INBOUND_0	90Ch

Diagram



Fields

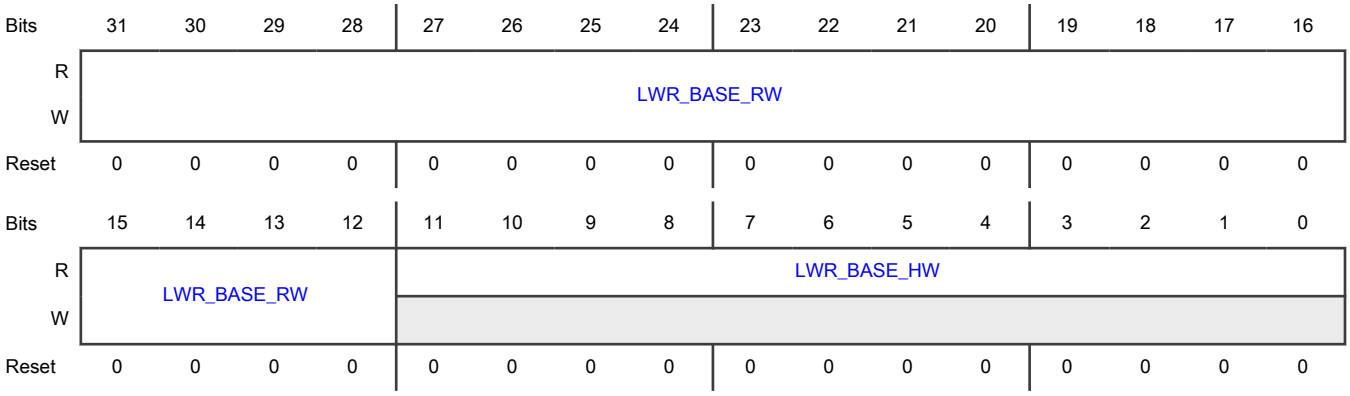
Field	Function
31-12 LWR_BASE_RW	Lower base address bits—programmable Forms bits [31:12] of the start address of the address region to be translated.
11-0 LWR_BASE_HW	Lower base address bits—hardwired Forms bits [11:0] of the start address of the address region to be translated. The start address must be aligned to a 4 KB boundary, so these bits are always 0. A write to this location is ignored by the PEX controller.

18.4.1.74 iATU Lower Base Address Register (IATU_LWR_BASE_ADDR_OFF_OUTBOUND_0)

Offset

Register	Offset
IATU_LWR_BASE_ADDR_OFF_OUTBOUND_0	90Ch

Diagram



Fields

Field	Function
31-12 LWR_BASE_RW	Lower base address bits—programmable Forms bits [31:12] of the start address of the address region to be translated.
11-0 LWR_BASE_HW	Lower base address bits—hardwired Forms bits [11:0] of the start address of the address region to be translated. The start address must be aligned to a 4 KB boundary, so these bits are always 0. A write to this location is ignored by the PEX controller.

18.4.1.75 iATU Upper Base Address Register (IATU_UPPER_BASE_ADDR_OFF_INBOUND_0)

Offset

Register	Offset
IATU_UPPER_BASE_ADDR_OFF_INBOUND_0	910h

Diagram

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

Fields

Field	Function
31-0 UPPER_BASE_RW	Upper base address bits Forms bits [63:32] of the start (and end) address of the address region to be translated.

18.4.1.76 iATU Upper Base Address Register (IATU_UPPER_BASE_ADDR_OFF_OUTBOUND_0)

Offset

Register	Offset
IATU_UPPER_BASE_ADDR_OFF_OUTBOUND_0	910h

Diagram

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

Fields

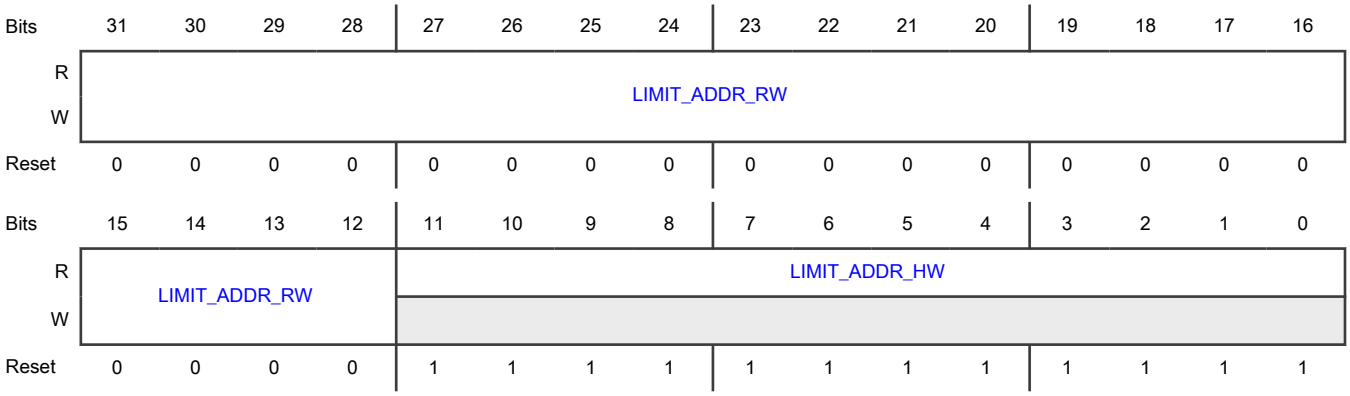
Field	Function
31-0 UPPER_BASE_RW	Upper base address bits Forms bits [63:32] of the start (and end) address of the address region to be translated. In systems with a 32-bit address space, this register is not used and therefore writing to this register has no effect.

18.4.1.77 iATU Limit Address Register (IATU_LIMIT_ADDR_OFF_INBOUND_0)

Offset

Register	Offset
IATU_LIMIT_ADDR_OFF_INBOUND_0	914h

Diagram



Fields

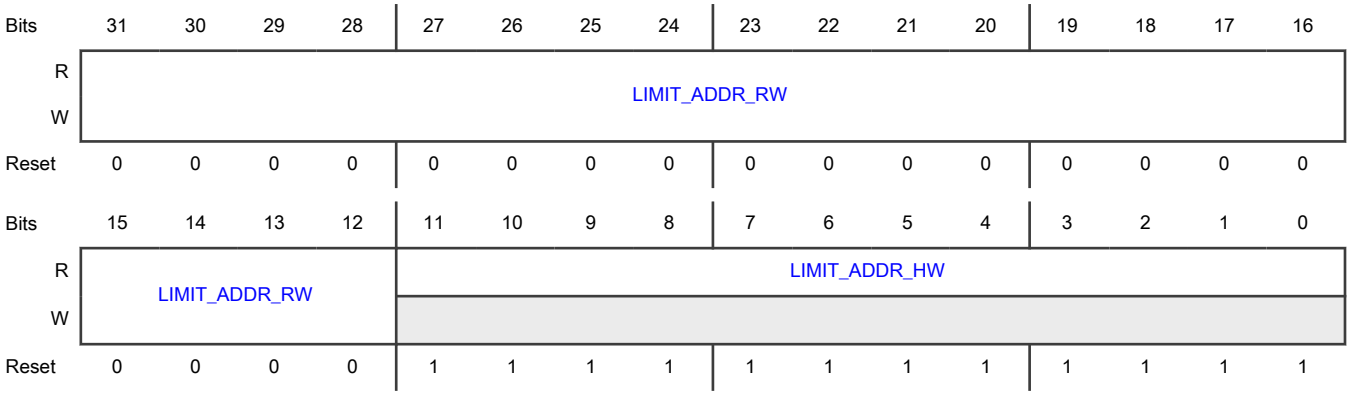
Field	Function
31-12 LIMIT_ADDR_RW	Limit address bits—programmable Forms bits [31:12] of the end address of the address region to be translated.
11-0 LIMIT_ADDR_HW	Limit address bits—hardwired Forms bits [11:0] of the end address of the address region to be translated. Address regions must be aligned to a 4 KB boundary, so these bits are always 1. A write to this location is ignored by the PEX controller.

18.4.1.78 iATU Limit Address Register (IATU_LIMIT_ADDR_OFF_OUTBOUND_0)

Offset

Register	Offset
IATU_LIMIT_ADDR_OFF_OUTBOUND_0	914h

Diagram



Fields

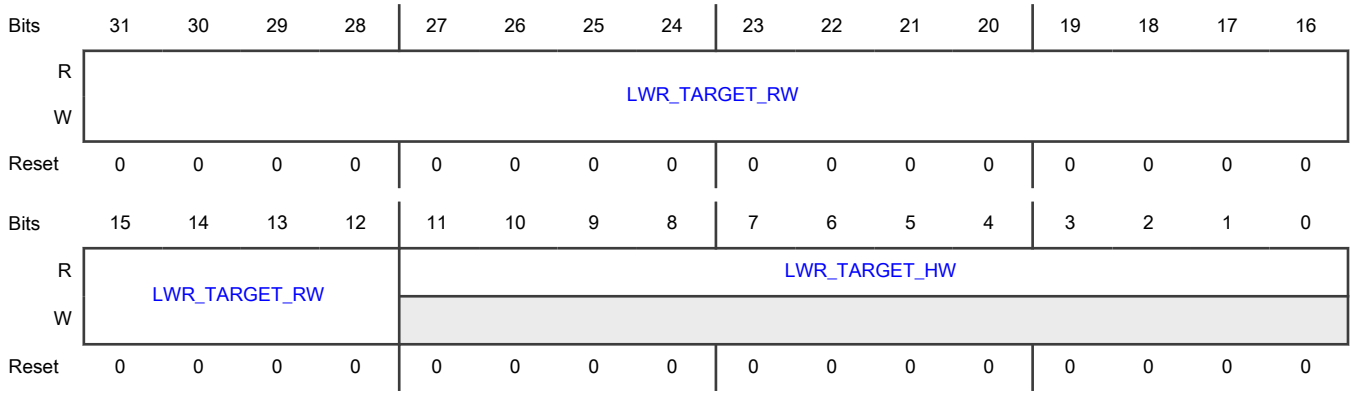
Field	Function
31-12 LIMIT_ADDR_RW	Limit address bits—programmable Forms bits [31:12] of the end address of the address region to be translated.
11-0 LIMIT_ADDR_HW	Limit address bits—hardwired Forms bits [11:0] of the end address of the address region to be translated. Address regions must be aligned to a 4 KB boundary, so these bits are always 1. A write to this location is ignored by the PEX controller.

18.4.1.79 iATU Region#N Lower Offset Address Register (IATU_LWR_TARGET_ADDR_OFF_INBOUND_0)

Offset

Register	Offset
IATU_LWR_TARGET_ADDR_OFF_INBOUND_0	918h

Diagram



Fields

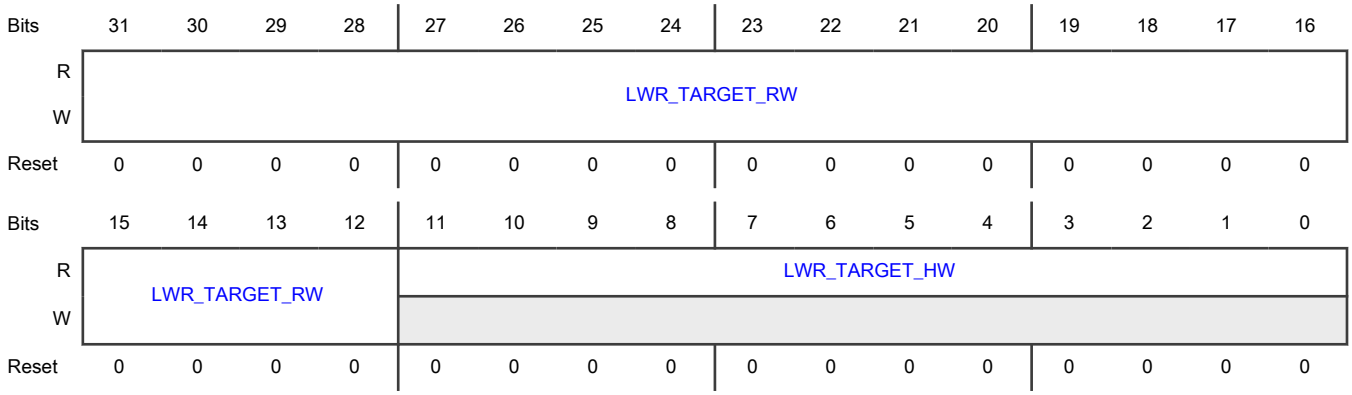
Field	Function
31-12 LWR_TARGET_RW	Lower target address bits—programmable Forms bits [31:12] of the of the new address of the translated region.
11-0 LWR_TARGET_HW	Lower target address bits—hardwired Forms bits [11:0] of the start address of the new address of the translated region. The start address must be aligned to a 4 KB boundary (in address match mode); and to the BAR size boundary (in BAR match mode) so these bits are always 0. If the BAR is smaller than the iATU region size, then the iATU target address must align to the iATU region size; otherwise it must align to the BAR size. A write to this location is ignored by the PEX controller.

18.4.1.80 iATU Outbound Region#N Lower Offset Address Register (IATU_LWR_TARGET_ADDR_OFF_OUTBOUND_0)

Offset

Register	Offset
IATU_LWR_TARGET_ADDR_OFF_OUTBOUND_0	918h

Diagram



Fields

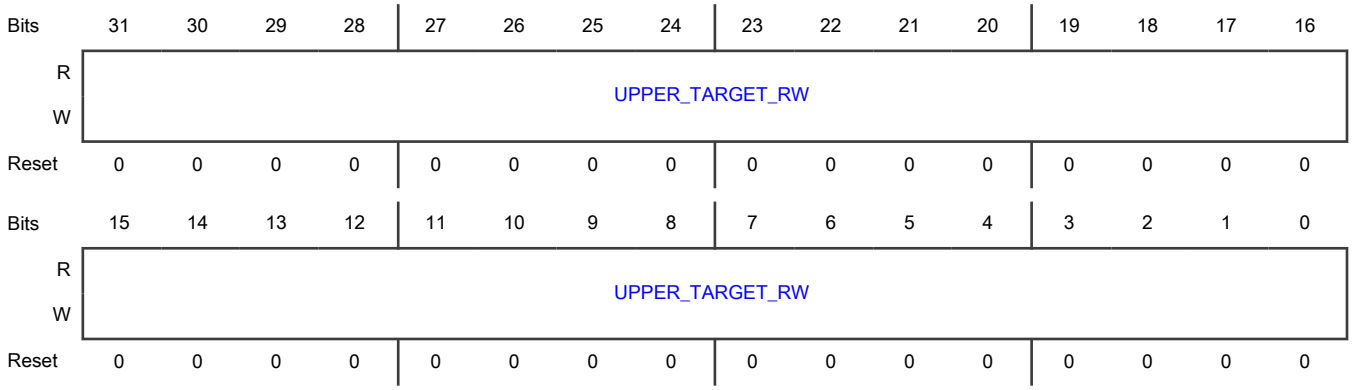
Field	Function
31-12 LWR_TARGET_RW	Lower target address bits—programmable Forms bits [31:12] of the of the new address of the translated region.
11-0 LWR_TARGET_HW	Lower target address bits—hardwired Forms bits [11:0] of the start address of the new address of the translated region. The start address must be aligned to a 4 KB boundary, so these bits are always 0. A write to this location is ignored by the PEX controller.

18.4.1.81 iATU Upper Target Address Register (IATU_UPPER_TARGET_ADDR_OFF_INBOUND_0)

Offset

Register	Offset
IATU_UPPER_TARGET_ADDR_OFF_INBOUND_0	91Ch

Diagram



Fields

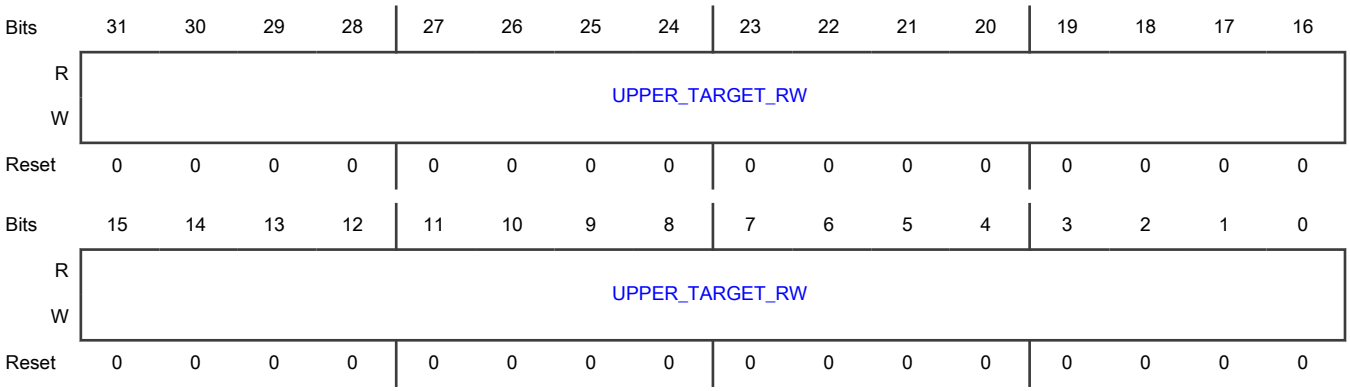
Field	Function
31-0 UPPER_TARGET_RW	Upper target address bits Forms bits [63:32] of the start address of the new address of the translated region. In systems with a 32-bit address space, this register is not used and therefore writing to this register has no effect.

18.4.1.82 iATU Upper Target Address Register (IATU_UPPER_TARGET_ADDR_OFF_OUTBOUND_0)

Offset

Register	Offset
IATU_UPPER_TARGET_ADDR_OFF_OUTBOUND_0	91Ch

Diagram



Fields

Field	Function
31-0 UPPER_TARGET_RW	Upper target address bits Forms bits [63:32] of the start address of the new address of the translated region.

18.4.1.83 Base Address Register 0 Mask (BAR0_MASK)

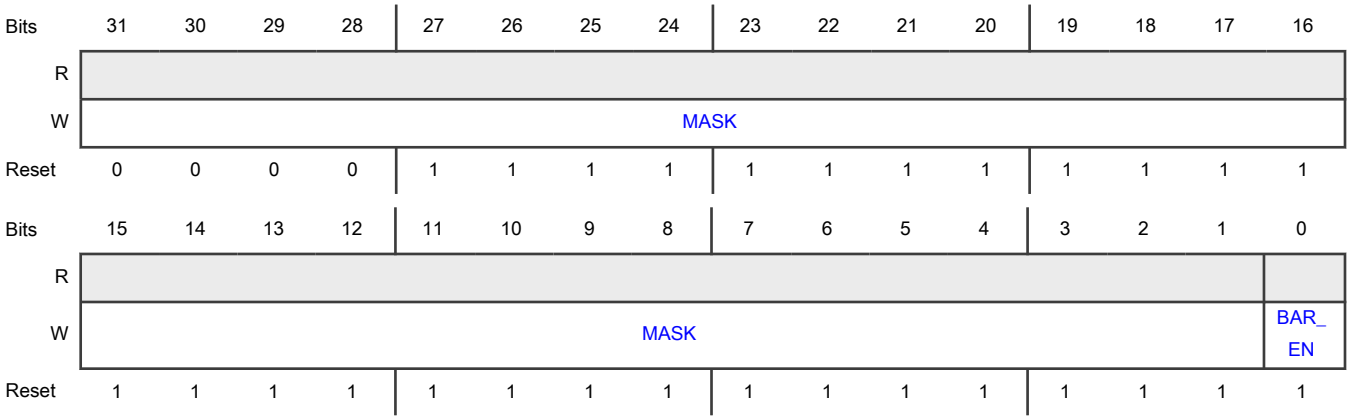
Offset

Register	Offset
BAR0_MASK	1010h

Function

The Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express base address register (BAR) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

Diagram



Fields

Field	Function
31-1 MASK	Mask
0 BAR_EN	BAR enable. 0b - Disable BAR 1b - Enable BAR

18.4.1.84 Base Address Register 1 Mask (BAR1_MASK)

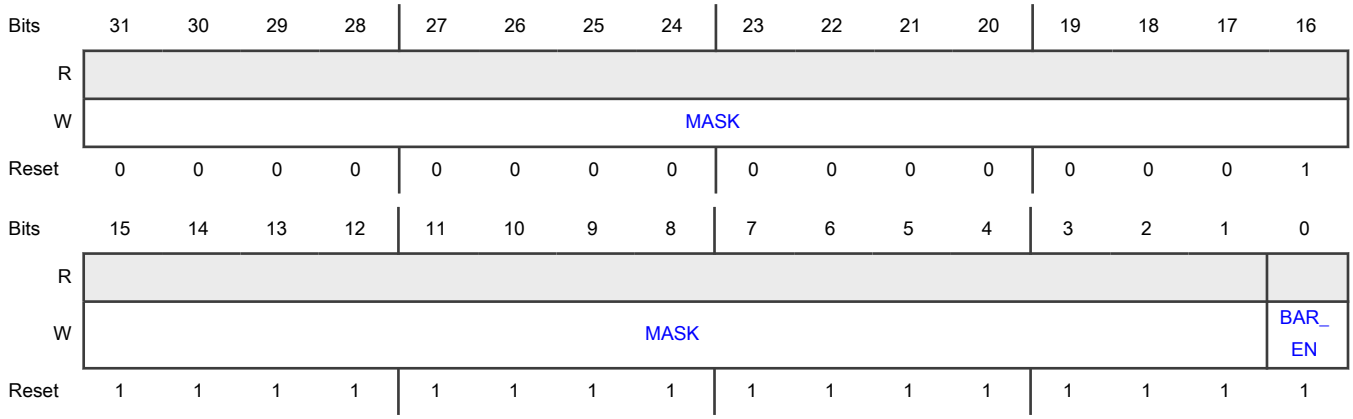
Offset

Register	Offset
BAR1_MASK	1014h

Function

The Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express base address register (BAR) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

Diagram



Fields

Field	Function
31-1 MASK	Mask
0 BAR_EN	BAR enable. 0b - Disable BAR 1b - Enable BAR

18.4.1.85 Base Address Register 2 Mask (BAR2_MASK)

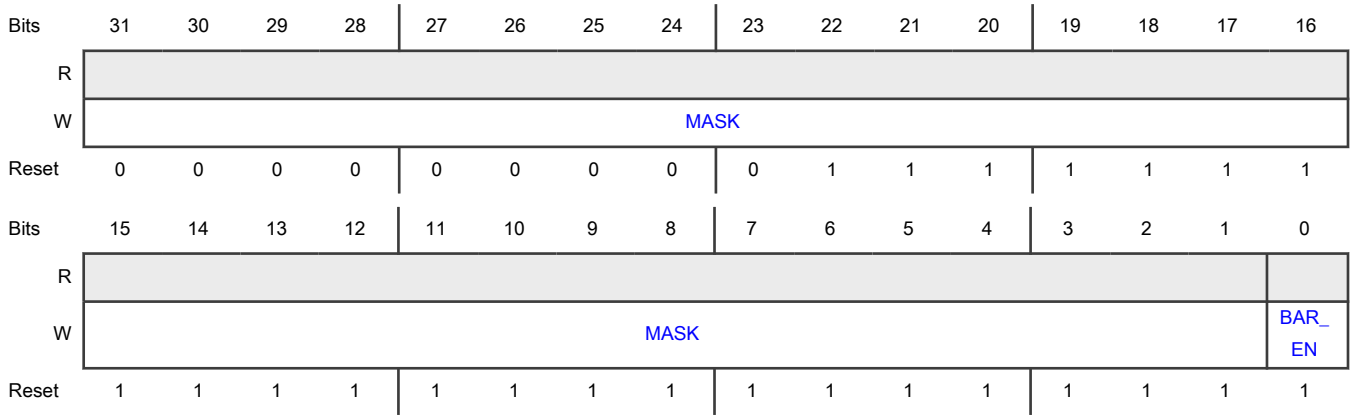
Offset

Register	Offset
BAR2_MASK	1018h

Function

The Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express base address register (BAR) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

Diagram



Fields

Field	Function
31-1 MASK	Mask
0 BAR_EN	BAR enable. 0b - Disable BAR 1b - Enable BAR

18.4.1.86 Base Address Register 3 Mask (BAR3_MASK)

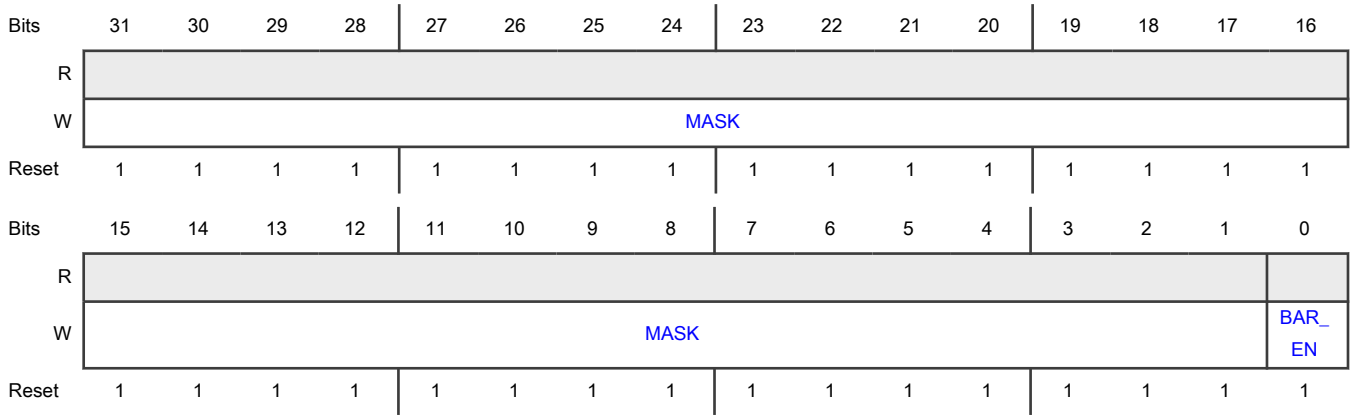
Offset

Register	Offset
BAR3_MASK	101Ch

Function

The Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express base address register (BAR) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

Diagram



Fields

Field	Function
31-1 MASK	Mask
0 BAR_EN	BAR enable. 0b - Disable BAR 1b - Enable BAR

18.4.1.87 Expansion ROM Base Address Register Mask (EP mode) (EXP_ROM_BAR_MASK_EP)

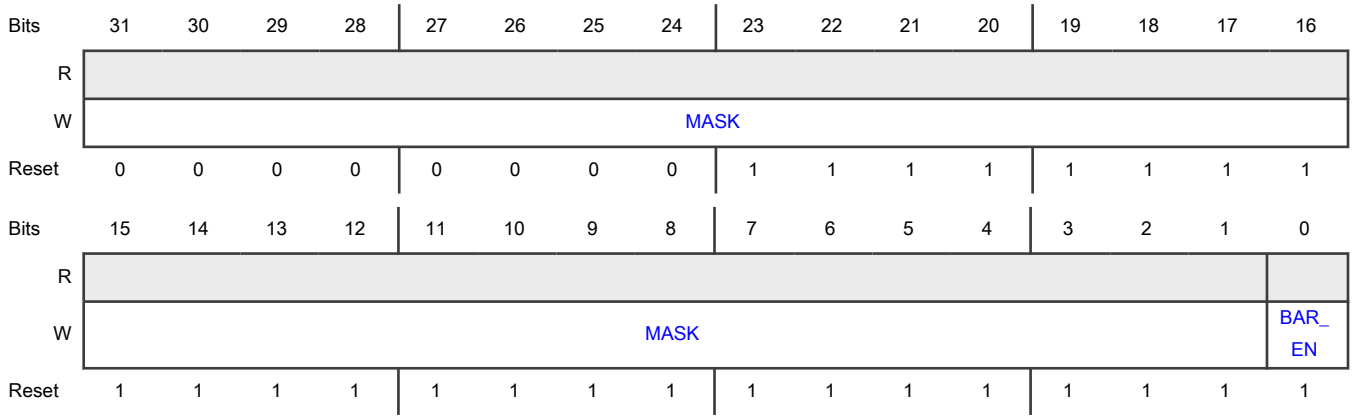
Offset

Register	Offset
EXP_ROM_BAR_MASK_EP	1030h

Function

The Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express Expansion ROM Base Address Register (EP-Mode) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

Diagram



Fields

Field	Function
31-1 MASK	Expansion ROM BAR Mask
0 BAR_EN	Expansion ROM BAR enable. 0b - Disable Expansion ROM BAR 1b - Enable Expansion ROM BAR

18.5 PEX_PF_CONTROL register descriptions

18.5.1 PEX_PF_CONTROL Memory map

PEX_PF0_CONTROL base address: 34C_0000h

Offset	Register	Width (In bits)	Access	Reset value
14h	PEX PFa Config Register (PEX_PF0_CONFIG)	32	RW	0000_0000h
18h	PEX PFa Interrupt status Register (PEX_PF0_INT_STAT)	32	RO	0000_0000h
20h	PEX PFa PCIE pme and message detect register (PEX_PF0_PME_MES_DR)	32	W1C	0000_0000h
24h	PEX PFa PCIE pme and message disable register (PEX_PF0_PME_MES_DISR)	32	RW	0000_0000h
28h	PEX PFa PCIE pme and message interrupt enable register (PEX_PF0_PME_MES_IER)	32	RW	0000_0000h
2Ch	PEX PFa PCIE message command register (PEX_PF0_MCR)	32	RW	0000_0000h
140h	PEX PFa Route By Port Address Upper register (PEX_PF0_RBP_ADDR_U)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
200h	PEX PFa PCIE error detect register (PEX_PF0_ERR_DR)	32	W1C	0000_0000h
208h	PEX PFa PCIE error interrupt enable register (PEX_PF0_ERR_EN)	32	RW	0000_0000h
210h	PEX PFa PCIE error disable register (PEX_PF0_ERR_DISR)	32	RW	0000_0000h
7FCh	PEX PF0 Debug register (PEX_PF0_DBG)	32	RW	0000_0000h

18.5.2 PEX PFa Config Register (PEX_PF0_CONFIG)

Offset

Register	Offset
PEX_PF0_CONFIG	14h

Function

The PEX PF*a* Config register is used for setting the config ready control of the PCIE device.

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			LTSSM_EN	Reserved											CFG_REA...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-13 —	Reserved
12 LTSSM_EN	LTSSM Enable: This bit is cleared by default to enable EP application software to configure the PCIe controller. Any config attempts from the Root Complex will be returned with config retry status (CRS). After application Software finishes configuring the PCIe controller it sets the LTSSM_EN bit to enable the LTSSM state machine to start running.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-1 —	Reserved
0 CFG_READY	Config Ready: This bit is cleared by default to enable EP application software configure the PCIe controller. Any config attempts from the Root Complex will be returned with config retry status (CRS). After application software finishes configuring the PCIe controller it sets the CFG_READY bit to enable accesses from the Root Complex.

18.5.3 PEX PFa Interrupt status Register (PEX_PF0_INT_STAT)

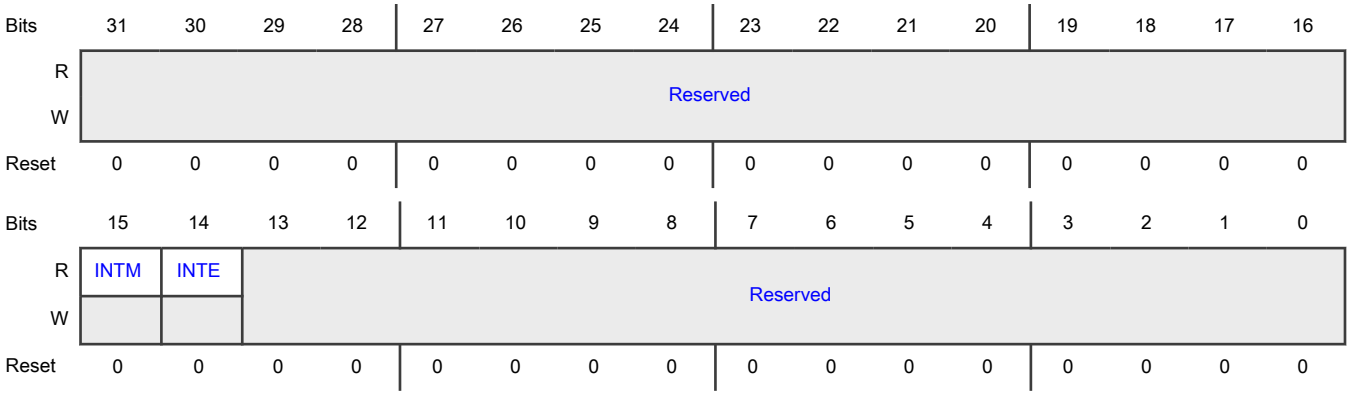
Offset

Register	Offset
PEX_PF0_INT_STAT	18h

Function

The PEX PF*a* Interrupt status register is used to indicate that an interrupt is pending.

Diagram



Fields

Field	Function
31-16 —	Reserved
15 INTM	Per PF dependent message interrupt is pending.

Table continues on the next page...

Table continued from the previous page...

Field	Function
14 INTE	Per PF dependent error interrupt is pending.
13-0 —	Reserved

18.5.4 PEX PFa PCIE pme and message detect register (PEX_PF0_PME_MES_DR)

Offset

Register	Offset
PEX_PF0_PME_MES_DR	20h

Function

The PEX PF*a* PCIE pme and message detect register is used to detect various power management and error events.

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	PTO	Reserved	ENL23	EXL23	Reserved	HRD	LDD	Reserved	LUD	Reserved						
W	W1C		W1C	W1C		W1C	W1C		W1C							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15 PTO	PTO: Indicates that PME turn off was detected. 1'b1: PME turn off was detected. 1'b0: PME turn off was not detected.
14	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
13 ENL23	ENL23: Indicates that PCIe core entered L2/L3 ready state. 1'b1: Entered L2/3 ready state was detected. 1'b0: Entered L2/3 ready state was not detected.
12 EXL23	EXL23: Indicates that PCIe core Exited L2/L3 ready state. 1'b1: Exit L2/3 ready state was detected. 1'b0: Exit L2/3 ready state was not detected.
11 —	Reserved
10 HRD	HRD: Indicates a hot reset was detected. 1'b1: hot reset was detected. 1'b0: hot reset was not detected.
9 LDD	LDD : Indicates a link down was detected. 1'b1: link down was detected. 1'b0: link down was not detected.
8 —	Reserved
7 LUD	LUD : Indicates a link up was detected. 1'b1: link up was detected. 1'b0: link up was not detected.
6-0 —	Reserved

18.5.5 PEX PFa PCIE pme and message disable register (PEX_PF0_PME_MES_DISR)

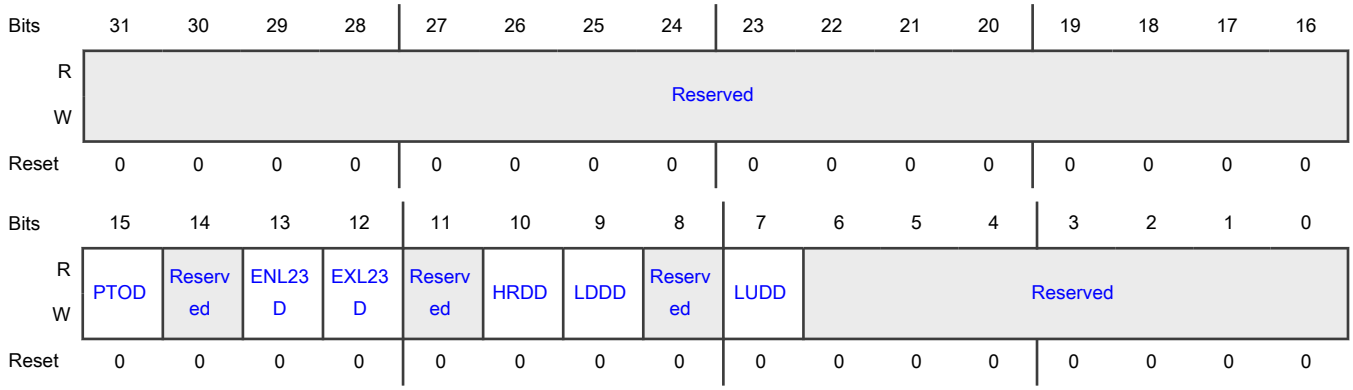
Offset

Register	Offset
PEX_PF0_PME_MES_DISR	24h

Function

The PEX PF*a* PCIE pme and message disable register is used to disable the detection of various power management and error events.

Diagram



Fields

Field	Function
31-16 —	Reserved
15 PTOD	PTO: PME turn off detect disabled. 1'b1: PME turn off detect disabled. 1'b0: PME turn off detect enabled.
14 —	Reserved
13 ENL23D	ENL23: Entered L2/L3 ready state detect disabled. 1'b1: Entered L2/L3 ready state detect disabled. 1'b0: Entered L2/L3 ready state detect enabled.
12 EXL23D	EXL23: Exited L2/L3 ready state detect disable. 1'b1: Exit L2/L3 ready state detect disabled. 1'b0: Exit L2/L3 ready state detect enabled.
11 —	Reserved
10 HRDD	HRDD: Hot reset detect disable. 1'b1: hot reset detect disabled. 1'b0: hot reset detect enabled.
9 LDDD	LDDD : Link down detect disable. 1'b1: link down detect disabled. 1'b0: link down detect enabled.
8 —	Reserved
7 LUDD	LUDD : link up detect disable. 1'b1: link up detect disabled. 1'b0: link up detect enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
6-0 —	Reserved

18.5.6 PEX PFa PCIE pme and message interrupt enable register (PEX_PF0_PME_MES_IER)

Offset

Register	Offset
PEX_PF0_PME_MES_IER	28h

Function

The PEX PF*a* PCIE pme and message disable register is used to enable interrupt generation for the detection of various power management and error events.

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	PTOIE	Reserved	ENL23IE	EXL23IE	Reserved	HRDIE	LDDIE	Reserved	LUDIE	Reserved						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15 PTOIE	PTO: PME turn off detect interrupt enable. 1'b1: PME turn off detect interrupt enabled. 1'b0: PME turn off detect interrupt disabled.
14 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 ENL23IE	ENL23IE: Entered L2/L3 ready state detect interrupt enabled. 1'b1: Entered L2/L3 ready state detect interrupt enabled. 1'b0: Entered L2/L3 ready state detect interrupt disabled.
12 EXL23IE	EXL23IE: Exited L2/L3 ready state detect interrupt enable. 1'b1: Exit L2/L3 ready state detect interrupt enabled. 1'b0: Exit L2/L3 ready state detect interrupt disabled.
11 —	Reserved
10 HRDIE	HRDD: Hot reset detect interrupt enable. 1'b1: hot reset detect interrupt enabled. 1'b0: hot reset detect interrupt disabled.
9 LDDIE	LDDIE : Link down detect interrupt enable. 1'b1: link down detect interrupt enabled. 1'b0: link down detect interrupt disabled.
8 —	Reserved
7 LUDIE	LUDIE : link up detect interrupt enable. 1'b1: link up detect interrupt enabled. 1'b0: link up detect interrupt disabled.
6-0 —	Reserved

18.5.7 PEX PFa PCIE message command register (PEX_PF0_MCR)

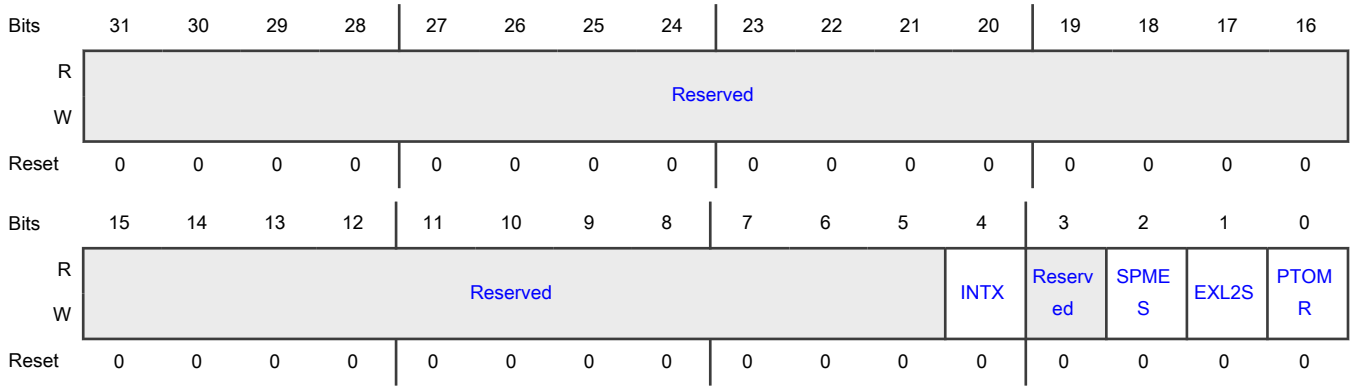
Offset

Register	Offset
PEX_PF0_MCR	2Ch

Function

The PEX PF*a* PCIE message command register enables the generation of interrupt and power management messages per PF.

Diagram



Fields

Field	Function
31-5 —	Reserved
4 INTX	INTX: Assert/de-assert intx command. When sys_int goes from low to high, the core generates an Assert_INTx Message. When sys_int goes from high to low, the core generates a Deassert_INTx Message.The Interrupt Pin register for the corresponding function determines which INTx Message the core generates (INTA).
3 —	Reserved
2 SPMES	SPME: Send PM_PME command. When set to 1 , a PM PME command is generated to wake up the the PCIE power management controller from a D1, D2 or D3 power state. The bit is self clearing. Once the bit is set , S/W needs to wait for the bit to self clear before sending a new command..
1 EXL2S	EXL2S : Exit L2 state command. When set to 1 , an L2 exit command is generated. The bit is self clearing. Once the bit is set , S/W needs to wait for the bit to self clear before sending a new command.
0 PTOMR	Reserved

18.5.8 PEX PFa Route By Port Address Upper register (PEX_PF0_RBP_ADDR_U)

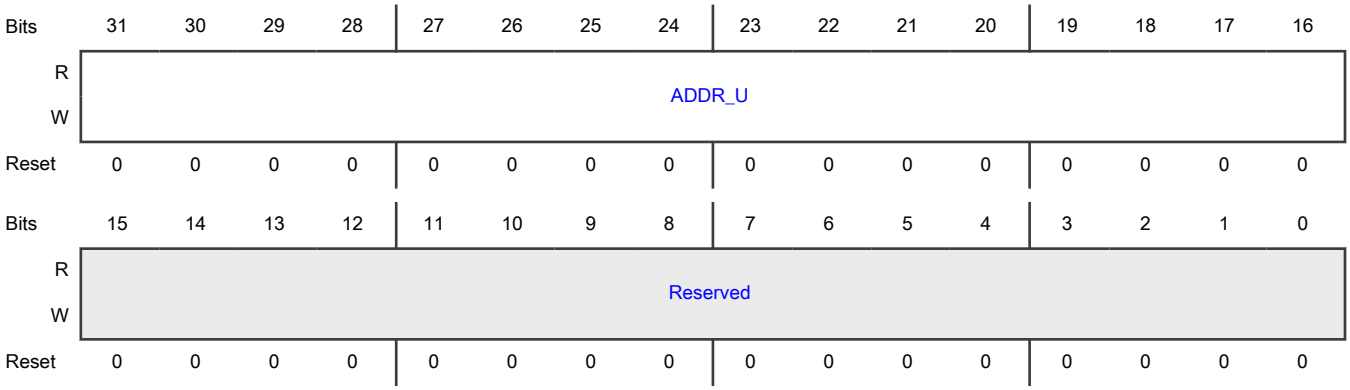
Offset

Register	Offset
PEX_PF0_RBP_ADDR_U	140h

Function

The PEX PF*a* Route By Port Address Upper register provides the upper 16 bit out of the 64 bit address of the PCIE slave device used for the outbound transactions.

Diagram



Fields

Field	Function
31-16 ADDR_U	ADDR_U : addr[63:48] . upper 16 bit of the PCIE slave device used for the outbound transactions.
15-0 —	Reserved

18.5.9 PEX PFa PCIE error detect register (PEX_PF0_ERR_DR)

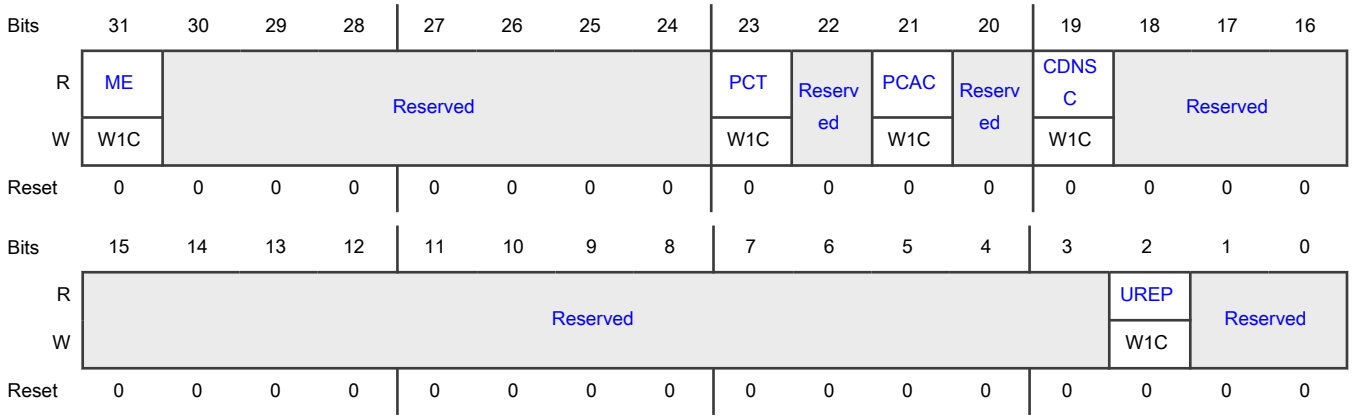
Offset

Register	Offset
PEX_PF0_ERR_DR	200h

Function

The PEX PF*a* error detect register is used for capturing various error events .

Diagram



Fields

Field	Function
31 ME	ME: Indicates Multiple errors of same type. If any of the detectable errors in PEX_ERR_DET register is detected more than one time the ME bit will be set. 1'b1: Multiple errors of same type was detected. 1'b0: Multiple errors of same type was not detected.
30-24 —	Reserved
23 PCT	PCT: Indicates completion timeout. 1'b1: Completion timeout was detected. 1'b0: Completion timeout was not detected.
22 —	Reserved
21 PCAC	PCAC: Completer abort was detected. 1'b1: Completer abort was detected. 1'b0: Completer abort was not detected.
20 —	Reserved
19 CDNSC	CDNSC : Completion with data not succsessful was detected. 1'b1: Completion with data not succsessful was detected. 1'b0: Completion with data not succsessful was not detected.
18-3 —	Reserved
2 UREP	UREP : Indicates an unsupported request completion was detected. 1'b1: Unsupported request in EP mode was detected. 1'b0: Unsupported request in EP mode not detected.
1-0 —	Reserved

18.5.10 PEX PFa PCIE error interrupt enable register (PEX_PF0_ERR_EN)

Offset

Register	Offset
PEX_PF0_ERR_EN	208h

Function

The PEX PF*a* error interrupt enable register. Enables/disables interrupt for errors detected in PEX_ERR_DET.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								PCTIE	Reserved	PCACIE	Reserved	CDNSCIE	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													UREPIE	Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23 PCTIE	PCTIE: completion timeout interrupt enable. 1'b1: Completion timeout interrupt enabled. 1'b0: Completion timeout interrupt disabled.
22 —	Reserved
21 PCACIE	PCACIE: Completer abort interrupt enable. 1'b1: Completer abort interrupt enabled. 1'b0: Completer abort interrupt disabled.
20 —	Reserved
19 CDNSCIE	CDNSCIE : Completion with data not succsessful interrupt enable. 1'b1: Completion with data not succsessful interrupt enabled. 1'b0: Completion with data not succsessful was not detected.
18-3	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
2 UREPIE	UREPIE : Unsupported request in EP mode interrupt enable. 1'b1: Unsupported request in EP mode interrupt enabled. 1'b0: Unsupported request in EP mode interrupt disabled.
1-0 —	Reserved

18.5.11 PEX PFa PCIE error disable register (PEX_PF0_ERR_DISR)

Offset

Register	Offset
PEX_PF0_ERR_DISR	210h

Function

The PEX PF*a* error disable register. disables detection of errors in PEX_ERR_DET.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MED	Reserved							PCTD	Reserved	PCACD	Reserved	CDNSCD	Reserved		
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	Reserved													UREPD	Reserved	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 MED	ME: Multiple errors of same type detection disable. 1'b1: Multiple errors of same type detection disabled. 1'b0: Multiple errors of same type detection enabled.
30-24 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
23 PCTD	PCTIE: completion detection disable. 1'b1: Completion timeout detection disabled. 1'b0: Completion timeout detection enabled.
22 —	Reserved
21 PCACD	PCACD: Completer abort detection disable. 1'b1: Completer abort detection disabled. 1'b0: Completer abort detection enabled.
20 —	Reserved
19 CDNSCD	CDNSCD : Completion with data not succsessful detection disable. 1'b1: Completion with data not succsessful detection disabled. 1'b0: Completion with data not succsessful detection enabled.
18-3 —	Reserved
2 UREPD	UREPD : Unsupported request in EP mode detection disable. 1'b1: Unsupported request in EP mode detection disabled. 1'b0: Unsupported request in EP mode detection enabled.
1-0 —	Reserved

18.5.12 PEX PF0 Debug register (PEX_PF0_DBG)

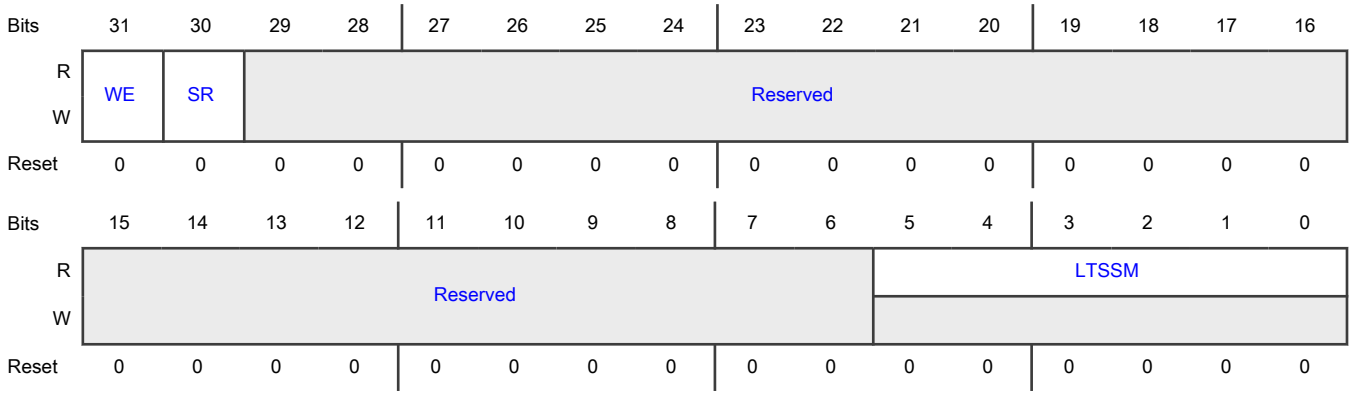
Offset

Register	Offset
PEX_PF0_DBG	7FCh

Function

By setting the WE field, the user has access to the SR field. Once set, SR will enable soft reset to the PEX module. LTSSM is a read only field that reports the LTSSM status from the PEX module.

Diagram



Fields

Field	Function
31 WE	Write Enable This bit when set allows the user to have write access to the PEXDBG[SR] for both setting and clearing the soft reset on the PEX module.
30 SR	Soft Reset When SR is set to 1, the PEX module enters soft reset. The PEX mode remains in soft reset while SR is set to 1. When SR is cleared, the PEX module exits soft reset. If SR is to be set and cleared multiple times back to back, software must wait a few seconds before changing the bit value to allow each new state to take effect.
29-6 —	Reserved
5-0 LTSSM	Link Training Status State Machine (LTSSM) status This field is read only and captures the PEX block LTSSM state at each clock cycle. These bits indicate the link training status. This field is useful for debugging link training failures. **NOTE** The status code changes while reacting to the link status. Therefore, software may need to read LTSSM multiple times to ensure the value has stabilized. 000000b - DETECT_QUIET 000001b - DETECT_ACTIVE 000010b - POLL_ACTIVE 000011b - POLL_COMPLIANCE 000100b - POLL_CONFIG 000101b - PRE_DETECT_QUIET 000110b - DETECT_WAIT 000111b - CFG_LINKWD_START

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001000b - CFG_LINKWD_ACEPT 001001b - CFG_LANENUM_WAIT 001010b - CFG_LANENUM_ACEPT 001011b - CFG_COMPLETE 001100b - CFG_IDLE 001101b - RCVRY_LOCK 001110b - RCVRY_SPEED 001111b - RCVRY_RCVRCFG 010000b - RCVRY_IDLE 010001b - L0 010010b - L0S 010011b - L123_SEND_EIDLE 010100b - L1_IDLE 010101b - L2_IDLE 010110b - L2_WAKE 010111b - DISABLED_ENTRY 011000b - DISABLED_IDLE 011001b - DISABLED 011010b - LPBK_ENTRY 011011b - LPBK_ACTIVE 011100b - LPBK_EXIT 011101b - LPBK_EXIT_TIMEOUT 011110b - HOT_RESET_ENTRY 011111b - HOT_RESET 100000b - RCVRY_EQ0 100001b - RCVRY_EQ1 100010b - RCVRY_EQ2 100011b - RCVRY_EQ3

18.6 Functional Description

The PCI Express protocol relies on a requestor/completer relationship where one device requests that some desired action be performed by some target device and the target device completes the task and responds. Usually the requests and responses occur through a network of links, but to the requester and to the completer, the intermediate components are transparent.

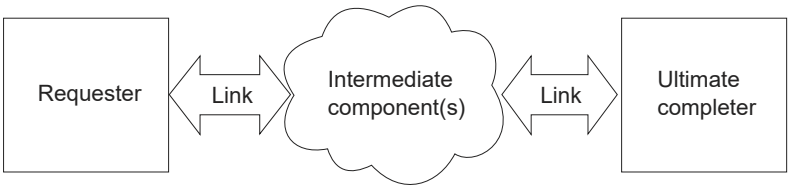


Figure 45. Requester/Completer Relationship

Each PCI Express device is divided into two halves—transmit (TX) and receive (RX), and each of these halves is further divided into three layers—transaction, data link, and physical, as shown in the following figure.

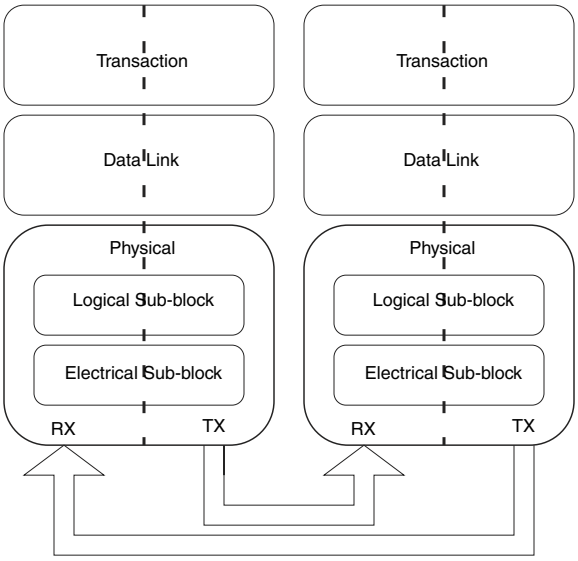


Figure 46. PCI Express High-Level Layering

Packets are formed in the transaction layer (TLPs) and data link layer (DLLPs), and each subsequent layer adds the necessary encodings and framing, as shown in the following figure. As packets are received, they are decoded and processed by the same layers but in reverse order, so they may be processed by the layer or by the device application software.

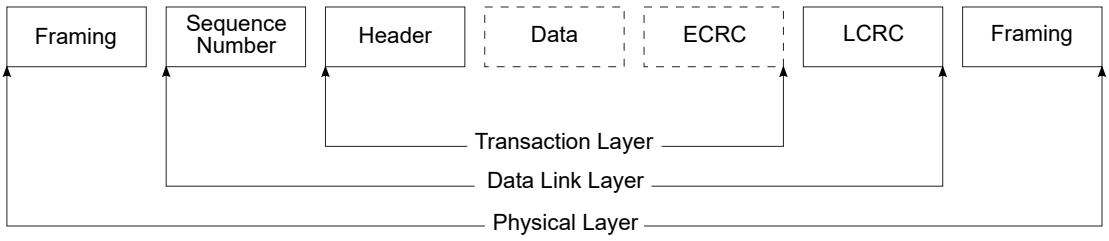


Figure 47. PCI Express Packet Flow

18.6.1 Architecture

This section describes implementation details of the PCI Express controller.

18.6.1.1 PCI Express Transactions

The following table contains the list of transactions that the PCI Express controller supports as an initiator and a target.

Table 66. PCI Express Transactions

PCI Express Transaction	Supported as an Initiator	Supported as a Target	Definition
Mrd	Yes	Yes	Memory Read Request
MRdLk	No	No	Memory Read Lock. As a target, CplLk with UR status is returned.
MWr	Yes	Yes	Memory Write Request to memory-mapped PCI-Express space
IORd	No	No	I/O Read request. As a target, Cpl with UR status is returned.
IOWr	No	No	I/O Write Request. As a target, Cpl with UR status is returned.
CfgRd0	No	Yes	Configuration Read Type 0.
CfgWr0	No	Yes	Configuration Write Type 0.
CfgRd1	No	No	Configuration Read Type 1. As a target, Cpl with UR status is returned.
CfgWr1	No	No	Configuration Write Type 1. As a target, Cpl with UR status is returned.
Msg	Yes	Yes	Message Request. Message without data is not forwarded to memory.
MsgD	No	Yes	Message Request with Data payload.
Cpl	Yes	Yes	Completion without Data
CplD	Yes	Yes	Completion with Data
CplLk	No	Yes	Completion for Locked Memory Read without Data. The only time that CplLk is returned with UR status is when the controller receives a MRdLk command.
CplDLk	No	No	Completion for Locked Memory Read with Data

18.6.1.2 Transaction ordering rules

In general, transactions are serviced in the order that they are received. However, transactions can be reordered as they are sent due to a stalled condition such as a full internal buffer. The following table describes the transaction ordering rules for this device.

Table 67. PCI Express controller TLP transaction ordering rules

Row pass column?		Posted request	Non-posted request		Completion	
		Memory write or message request (col 2)	Read request (Col 3)	I/O or Configuration write request (col 4)	Read completion (Col 5)	I/O or configuration write completion (col 6)
Posted request	Memory write or message request (row A)	a) No[1] b) No	Yes	Yes	a) Yes[2] b) N/A[3]	a) Yes[2] b) N/A[3]

Table continues on the next page...

Table 67. PCI Express controller TLP transaction ordering rules (continued)

Row pass column?		Posted request	Non-posted request		Completion	
		Memory write or message request (col 2)	Read request (Col 3)	I/O or Configuration write request (col 4)	Read completion (Col 5)	I/O or configuration write completion (col 6)
Non-posted request	Read request (row B)	No	No	No	a) No[4] b) Yes[5]	a) No[4] b)Yes[5]
	I/O or configuration write request (Row C)	No	No	No	a) No[4] b) Yes[5]	a) No[4] b) Yes[5]
Completion	Read completion (row D)	a) No[6] b) Yes[7]	Yes	Yes	a) No[8] b) No[8]	No
	I/O or configuration write completion (row E)	a) No[6] b) Yes[7]	Yes	Yes	No	No

1. Regardless of the setting of the relaxed ordering (RO) bit, a posted request cannot bypass another posted request.
2. Regardless of the setting of the relaxed ordering bit, a posted request can always bypass a completion.
3. N/A indicates that the original rules at these entries defined by the *PCI Express Base Specification* do not apply to RC or EP.
4. A non-posted request cannot bypass a completion if the relaxed ordering bit is cleared (that is, RO = 0).
5. A non-posted request can bypass a completion if the relaxed ordering bit is set (that is, RO = 1).
6. A read completion, I/O write completion, or configuration write completion cannot bypass a posted request if the relaxed ordering bit is cleared (that is, RO = 0).
7. A read completion, I/O write completion, or configuration write completion can bypass a posted request if the relaxed ordering bit is set (that is, RO = 1).
8. Regardless of the setting of the relaxed ordering bit, a read completion cannot bypass another read completion.

In general, the following points summarize the ordering rules for sending the next outstanding request:

- A posted request can bypass all other transactions except another posted request.
- A non-posted request cannot bypass posted or other non-posted requests, but it can bypass a completion if the relaxed ordering (RO) bit is set.(See Table 67).
- A completion can bypass posted requests if the relaxed ordering (RO) bit is set and can bypass non-posted transactions. However, a completion cannot bypass other completions.

18.6.1.3 Internal Address Translation Unit

The core uses the iATU to replace the TLP address and TLP header fields in the current TLP request header. This section presents the following topics:

- iATU Overview
- Programming the iATU
- Outbound iATU Operation
- Inbound iATU Operation

18.6.1.3.1 iATU Overview

Address translation is used for mapping different address ranges to different memory spaces supported by your application. A typical example maps the internal platform memory space to PCI memory space. The iATU supports type translation. Without address translation, your application address is passed to/from the PCIe TLPs directly through the internal platform . You can program the iATU to implement your own address translation scheme without the need for additional external hardware.

Inbound Features:

- Match mode operation for MEM TLPs. No translation for completions. Selectable BAR Match mode operation for MEM TLPs.

  NOTE

  BAR match mode must be used.

  - TLPs destined for configuration registers in an upstream port are not translated.
  - TLPs that are not error-free (ECRC, malformed and so on) are not translated.
- Programmable TLP header field matching.
  - TYPE/AT
- Function number
- 3 address regions programmable for location and size.
- Programmable enable/disable per region.
- Automatic FMT field translation between three DWORDs and four DWORDs for 64-bit addresses.
- ECAM Configuration Shift mode to allow a 256 MB CFG1 space to be located anywhere in the 64-bit address space.
- Supports regions from 4 kB to 4 GB in size.

Outbound Features:

- Address Match mode operation for MEM and MSG TLPs. No translation for completions.
- Supports type translation through TLP type header field replacement for MEM types to MSG/CFG types.
  - Includes posted to non-posted translation
  - No translation from completions
- Programmable TLP header field replacement.
  - TYPE/TD/TC/AT/ATTR/MSG-Code
- 4 address regions programmable for location and size.
- Programmable enable/disable per region.
- Automatic FMT field translation between three DWORDs and four DWORDs for 64-bit addresses.
- Configuration Shift mode. Optimizes the memory footprint of CFG accesses destined for the internal platform interface in a multifunction device.
- Response code which defines the completion status to return for accesses matching a region.
- Supports regions from 4 kB to 4 GB in size.

NOTE

The default behavior of the ATU when there is no address match in the outbound direction or no TLP attribute match in the inbound direction, is to pass the transaction through.

18.6.1.3.2 Programming the iATU

You can access the iATU registers through the DBI interface or through PCIe CFG accesses. The following registers are used for programming the iATU.

Table 68. iATU Register Map

Byte Offset	Description
0x900	iATU Index Register
0x904	iATU Region Control 1 Register
0x908	iATU Region Control 2 Register
0x90C	iATU Region Lower Base Address Register
0x910	iATU Region Upper Base Address Register
0x914	iATU Region Limit Address Register
0x918	iATU Region Lower Target Address Register
0x91C	iATU Region Upper Target Address Register
0x920	iATU Region Control 3 Register

The iATU registers are programmed through an indirect addressing scheme (using an index register) to reduce the address footprint in the PCI Express extended configuration space. The index register has a “Region Direction” bit to determine whether an inbound or outbound region is being accessed and a “Region Index” field to determine which region to program/read when accessing the other address translation registers.

18.6.1.3.3 Outbound iATU Operation

This section describes the processing of outbound requests by the iATU. This section presents the following topics:

- Overview (Address Match Mode)
- RID BDF Number Replacement
- iATU Outbound MSG Handling
- FMT Translation
- No Address Match Result
- Writing to a MRdLk Region
- Outbound Programming Example

18.6.1.3.3.1 Overview (Address Match Mode)

The address field of each request MEM and I/O TLP is checked to see if it falls into any of the enabled address regions defined by the “Start” and “End” addresses as defined in Figure 48.

NOTE

When the “Region Enable” bit of the Region Control 2 register is “0”, then that region is not used for address matching.

NOTE

The target(translation) address and base address of the inbound iATU window must be aligned based on the window size programmed in the corresponding BAR MASK register

When an address match is found, then the TLP address field is modified as follows: Address = Address - Base Address + Target Address and the TYPE, TD, TC, AT, and ATTR header fields are replaced with the corresponding fields in the “iATU Control 1 Register” (IATU_REGION_CTRL_1_OFF_OUTBOUND_0). When your application internal platform address field matches more than one of the 4 address regions, then the first (lowest of the numbers from 0 to 3) enabled region to be matched is used. See No Address Match Result for details on what happens when there is no address match. This operational mode (called “Address Match Mode“) is always used for outbound translation.

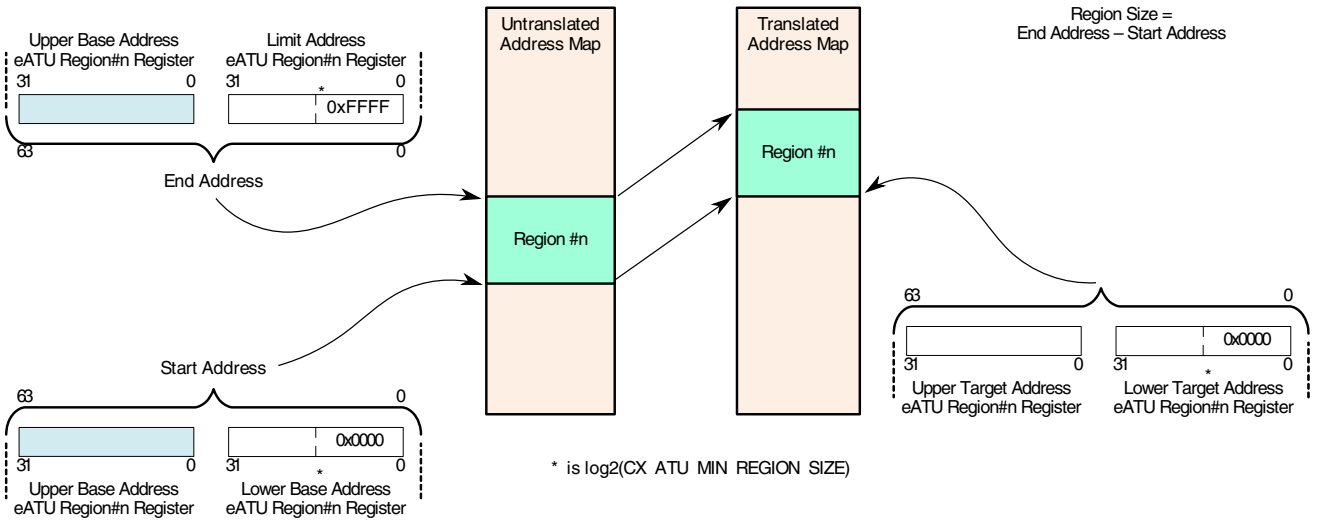


Figure 48. iATU Address Region Mapping: Outbound and Inbound (Address Match Mode), 64-bit Address

When the PCIe core is operating with 32-bit addresses, then the operation is defined as in figure below.

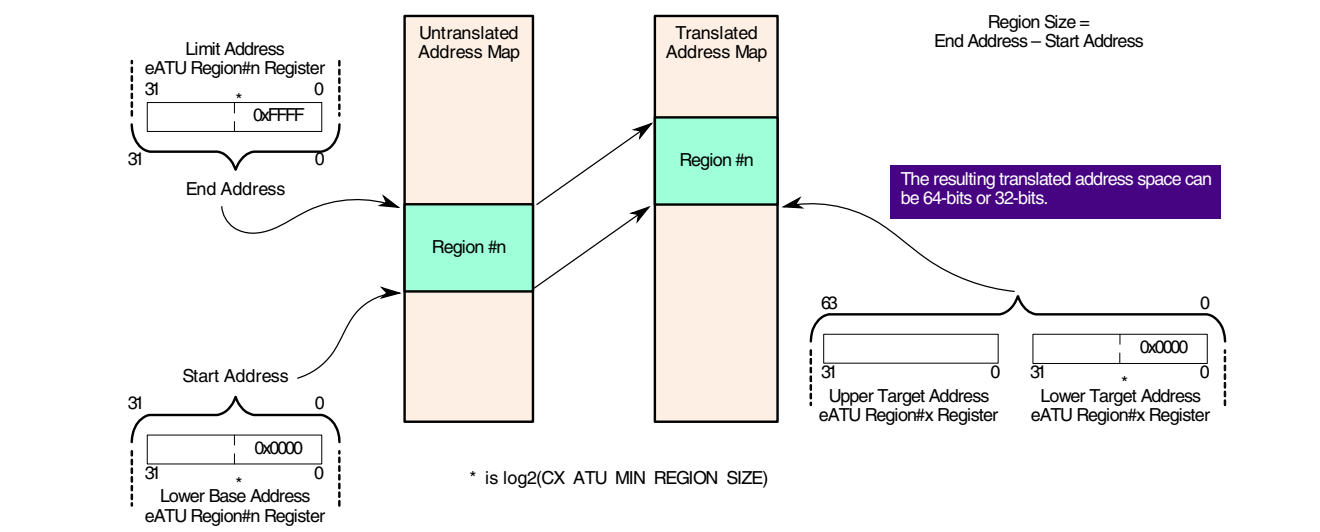


Figure 49. iATU Address Region Mapping: Outbound and Inbound (Address Match Mode), 32-bit Address

The upper 32 bits of the Target Address register always forms the upper 32 bits of the translated address because:

- The maximum region size is 4 GB.
- A region must not cross a 4 GB boundary.

In the figure, CX_ATU_MIN_REGION_SIZE specifies the minimum size of an address translation region and is 4 KB.

18.6.1.3.3.2 RID BDF Number Replacement

When there is a successful address match on an outbound TLP, then the function number used in generating the function part of the requester ID field of the TLP is taken from the 3-bit “Function Number” field of the “iATU Control 1 Register” (IATU_REGION_CTRL_1_OFF_OUTBOUND_0). The value in this field must be 0x0 unless multifunction operation is enabled (the number of PFs is greater than 1).

NOTE

The requester ID uses the 8:5:3 bit PCI Bus.Device.Function (BDF) format.

18.6.1.3.3.3 iATU Outbound MSG Handling

NOTE

The iATU only supports generating outbound messages with data (MsgD); messages without data are not supported. The MWr transactions to be translated must have a non-zero effective length.

The iATU supports TYPE translation/conversion of MEM and I/O TLPs to MsgD TLPs. This supports applications that are unable to directly generate MsgD TLPs. When there is a successful address match on an outbound MEM TLP, and the translated TLP type field is MSG (that is, the type field of the “iATU Control 1 Register” IATU_REGION_CTRL_1_OFF_OUTBOUND_0 is “10xxx”), then the message code field of the TLP is set to the value in the “Message Code” field of the “iATU Control 2 Register” (IATU_REGION_CTRL_2_OFF_OUTBOUND_0).

18.6.1.3.3.4 FMT Translation

The iATU automatically sets the TLP format field for three DWORDs when it detects all zeroes in the upper 32 bits of the translated address. Otherwise, it sets it to four DWORDs when it detects a 64-bit address (that is, when there is a “1” in the upper 32 bits of the translated address). When the original address and the translated address are of different format, the iATU ensures that the TLP header size matches the translated address format.

18.6.1.3.3.5 No Address Match Result

When there is no address match then the address is untranslated but the TLP header information comes from the relevant fields on the internal platform interface.

18.6.1.3.3.6 Writing to a MRdLk Region

When there is a successful address match for an outbound write and the type header field in the “iATU Control 1 Register” is “ 00001” indicating a locked MEM transfer, then the core sets the type field to “0000” (MEM).

18.6.1.3.3.7 Outbound Programming Example

Define outbound region 1 as a 64 kB I/O region from 0x8000_0000_D000_0000 to 0x8000_0000_D000_FFFF, to be mapped to 0x00010000 in the PCIe I/O space.

1. Setup the Index Register.

   Write 0x00000001 to Address { 0x700 + 0x200 } to set outbound region 1 as the current region

2. Setup the Region Base and Limit Address Registers.

   Write 0xd0000000 to Address {0x700 +0x20C} to set the Lower Base Address.

   Write 0x80000000 to Address {0x700 +0x210} to set the Upper Base Address.

   Write 0xd000ffff to Address {0x700 +0x214} to set the Limit Address.

3. Setup the Target Address Registers.

   Write 0x00010000 to Address {0x700 +0x218} to set the Lower Target Address.

   Write 0x00000000 to Address {0x700 +0x21C} to set the Upper Target Address.

4. Configure the region through the Region Control 1 Register.

   Write 0x00000002 to Address {0x700 +0x204} to define the type of the region to be I/O.

5. Enable the region.

   Write 0x80000000 to Address {0x700 +0x208} to enable the region.

18.6.1.3.4 Inbound iATU Operation

This section discusses how the iATU processes inbound requests. The topics are:

- Overview
- MEM Match Modes
- CFG Handling (Upstream Port)
- FMT Translation
- Inbound Programming Example

18.6.1.3.4.1 Overview

NOTE

- The main difference between inbound and outbound iATU operation is that the TLP type is never changed in the inbound direction. Instead, the type field is used for more precise matching. Other fields can also be optionally used to further refine the matching process.
- Another difference is that for MEM TLPs, inbound transactions for Endpoints use BAR matching instead of the address matching used in outbound operations.

The following translation rules and limitations apply:

- When there is no match, then the address is untranslated. In addition
- TLPs destined for the configuration registers in an upstream port are not translated.
- TLPs that are not error-free (ECRC, malformed and so on) are not translated.
- In BAR Match mode, only translation of MEM is supported.

  The setting of the “Match Mode” field in the “iATU Control 2 Register” (IATU_REGION_CTRL_2_OFF_ OUTBOUND_0) determines how iATU inbound matching is done for each TLP type.

Table 69. Determination of Match Mode By TLP Type and “Match Mode” Field of iATU Control 2 Register

	Match Mode = 0	Match Mode = 1
MEM	Address Match Mode—not supported"	BAR Match Mode

18.6.1.3.4.2 MEM Match Modes

BAR Match Mode

Looking for an address match is a two-step process.

1. The standard internal PCI Express BAR Matching Mechanism checks if the address field of any MEM request TLP falls into any address region defined by the enabled BAR addresses and masks.
2. When a matched BAR is found, then the iATU compares the BAR ID to the “BAR Number” field in the “iATU Control 2 Register” for all enabled regions.Figure 50 and Figure 51 provide more details on inbound translation in BAR Match Mode.

NOTE

BAR Match Mode can only be used for MEM transactions

NOTE

The target(translation) address and base address of the inbound iATU window must be aligned based on the window size programmed in the corresponding BAR MASK register

When the PCIe core is operating with 64-bit BARs, the operation is defined as in figure below.

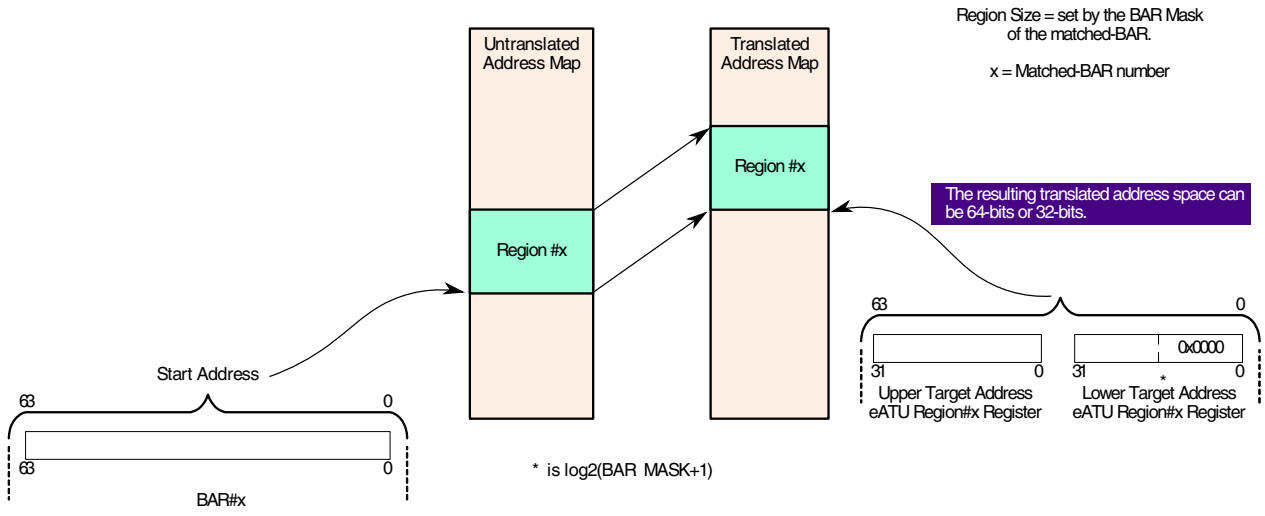


Figure 50. iATU Address Region Mapping: Inbound (BAR Match Mode), 64-bit BAR

When the address range does not match one of its BAR ranges in an upstream port, then the device rejects the request with unsupported request (UR) completion status and no translation occurs.

When the PCIe core is operating with 32-bit BARs, the operation is defined as in figure below.

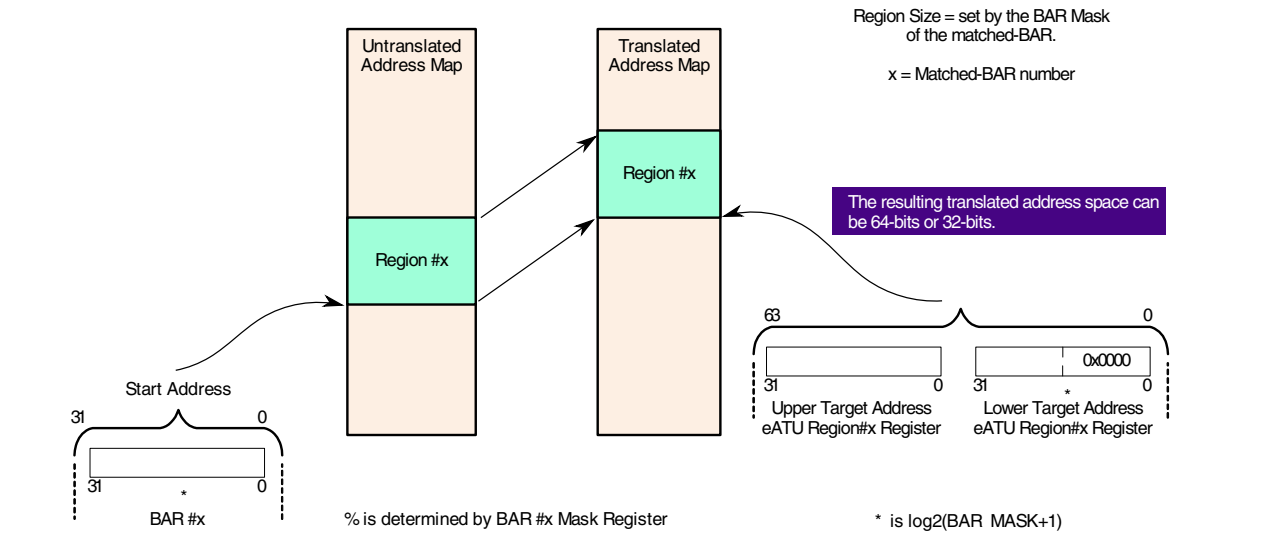


Figure 51. iATU Address Region Mapping: Inbound (BAR Match Mode), 32-bit BAR

When the address range does not match one of its BAR ranges in an upstream port, then the device rejects the request with unsupported request (UR) completion status and no translation occurs.

18.6.1.3.4.3 CFG Handling (Upstream Port)

The PEX controller normally routes CFG TLPs to the configuration registers without translating them. The iATU only translates CFG0 TLPs that the PEX controller has routed to the internal platform. Inbound address translation for CFG0 TLPs operates in one of two matching modes as determined by the “Inbound CFG0 Match Mode” field in the “iATU Control 2 Register”.

Routing ID Match Mode: The operation is similar to Outbound iATU Operation. The routing ID of the inbound CFG0 TLP must fall within the Base and Limit of the defined iATU region for matching to proceed. The iATU interprets the routing ID (Bytes 8-11 of TLP header) as an address. This corresponds to the upper 16 bits of the address in MEM and I/O transactions.

Accept Mode: The PEX controller always accepts CFG0 TLPs even when the CFG bus number does not match the current bus number of the device. This mode follows that behavior. The routing ID of received CFG0 TLPs are ignored when determining a match.

18.6.1.3.4.4 FMT Translation

The iATU automatically sets the TLP format field for three DWORDs when it detects all zeroes in the upper 32 bits of the translated address. Otherwise it sets it to four DWORDs when it detects a 64-bit address (when there is a “1” in the upper 32 bits of the translated address). When the original address and the translated address are of a different format then the iATU ensures that the TLP header size matches the translated address format.

18.6.1.3.4.5 Inbound Programming Example

BAR match mode

Define inbound region 2 as MEM region matching BAR4 (BAR match mode) mapping to 0x8000_0000_2000_0000 in your application memory space

1. Setup the Index Register.

   Write 0x80000002 to Address { 0x700 + 0x200 } to set inbound region 2 as the current region.

2. Setup the Target Address Registers.

   Write 0x20000000 to Address {0x700 + 0x218} to set the Lower Target Address.

   Write 0x80000000 to Address {0x700 + 0x21C} to set the Upper Target Address.

3. Configure the region through the Region Control 1 Register.

   Write 0x00000000 to Address {0x700 + 0x204} to define the type of the region to be MEM.

4. Enable the region for BAR Match Mode.

   Write 0xC0000400 to Address {0x700 + 0x208} to enable the region for BAR match mode for BAR4.

18.6.1.4 Memory Space Addressing

A PCI Express memory transaction can address a 32- or 64-bit memory space. As an initiator, the controller is capable of sending 32- or 64-bit memory packets. Any transaction from the internal platform that (after passing through the translation mechanism) has a translated address greater than 4G is sent as a 64-bit memory packet. Otherwise, a 32-bit memory packet is sent. As a target device, the controller is capable of decoding 32- or 64-bit memory packets. This is done through two 32-bit inbound windows and two 64-bit inbound windows. All inbound addresses are translated to 40-bit internal platform addresses.

Table 70. PCI Express Memory Transactions

Transaction	Type[4:0]	FMT[1]	FMT[0]
MRd (32-bit address)	00000	0	0
MRd (64-bit address)	00000	0	1
MWr (32-bit address)	00000	1	0
MWr (64-bit address)	00000	1	1

18.6.1.5 Configuration Space Addressing

The controller accepts inbound configuration transactions in EP mode only. The controller does not support generating outbound configuration transactions in EP mode.

Table 71. PCI Express Configuration Transactions

Transaction	Type	FMT[1]
CfgRd0	00100	0
CfgWr0	00100	1

Note that all configuration transactions sent on PCI Express require a response, regardless whether they are read or a write configuration transactions.

18.6.1.6 Messages

This section describes outbound message generation and inbound message reception.

18.6.1.6.1 Outbound Message Generation

Software can generate the following outbound message transactions:

Outbound Message	Generated by
Set_Slot_Power_Limit	Outbound iATU translation to MsgD with the appropriate encoding.
Set_Slot_Power_Limit	Writing to the Slot Power Limit Scale and Slot Power Limit Value fields of the Slot Capabilities Register in the PCI Express Capabilities Structure.

18.6.1.6.2 Inbound Messages

The following table provides a complete list of supported inbound messages in EP mode.

Table 72. PCI Express EP Inbound Message Handling

Name	Code[7:0]	Routing[2:0]	Action
Power Management			
PME_Turn_Off	0001 1001	011	
Slot Power Limit Support			

Table continues on the next page...

Table 72. PCI Express EP Inbound Message Handling (continued)

Name	Code[7:0]	Routing[2:0]	Action
Set_Slot_Power_Limit	0101 0000	100	Update power value in PCI Express device capability register in configuration space.

18.6.1.7 Error Handling

The PCI Express specification classifies errors as correctable and uncorrectable. Correctable errors result in degraded performance, but uncorrectable errors generally result in functional failures. As shown in the figure below, uncorrectable errors can further be classified as fatal or non-fatal.

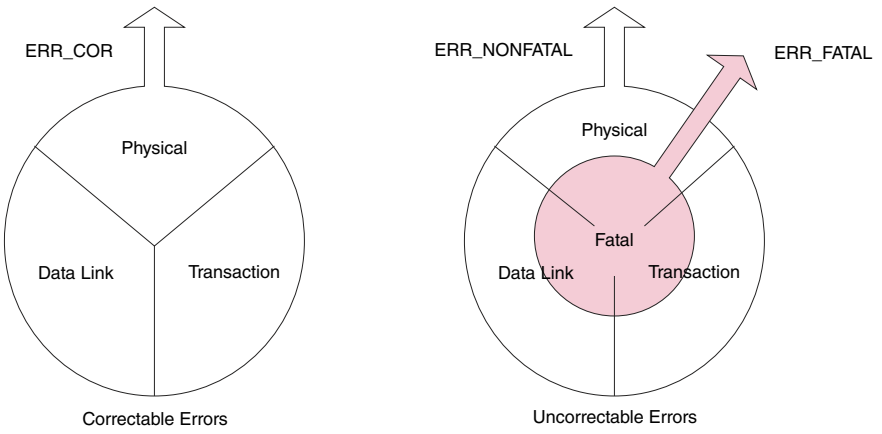


Figure 52. PCI Express Error Classification

18.6.1.7.1 PCI Express Error Logging and Signaling

The figure below shows the PCI Express-defined sequence of operations related to signaling and logging of errors detected by a device. Note that the PCI Express controller on this device supports the advanced error handling capabilities shown within the dotted lines.

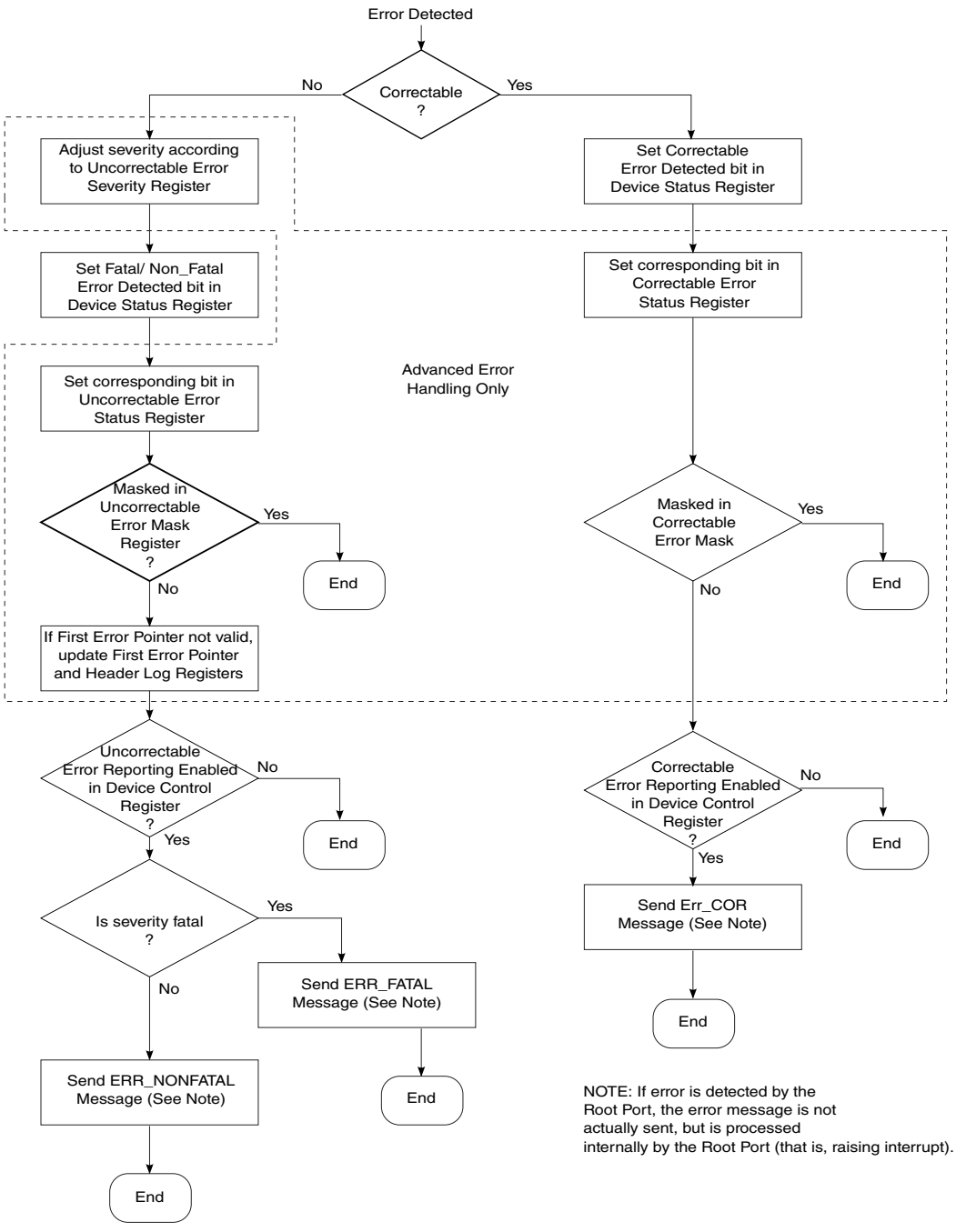


Figure 53. PCI Express Device Error Signaling Flowchart

18.6.2 Interrupts

Both message signaled interrupts (MSI) and legacy INTx are supported.

- Legacy INTx interrupts are handled by virtual-wire message transactions. See Inbound Messages.

18.6.3 Initial Credit Advertisement

To prevent overflowing of the receiver's buffers and for ordering compliance purposes, the transmitter cannot send transactions unless it has enough flow control (FC) credits to send. Each device maintains an FC credit pool. The FC information is conveyed between the two link partners by DLLPs during link training (initial credit advertisement). The transaction layer performs the FC accounting functions. One FC unit is four DWs (16-bytes) of data.

Table 73. Initial credit advertisement

Credit Type	Initial Credit Advertisement
PH (Memory Write, Message Write)	4
PD (Memory Write, Message Write)	32
NPH (Memory Read, IO Read, Cfg Read, Cfg Write)	4
NPD (IO Write, Cfg Write)	2
CPLH (Memory Read Completion, IO R/W Completion, Cfg R/W Completion)	Infinite
CPLD (Memory Read Completion, IO Read Completion, Cfg Read Completion)	Infinite

18.6.4 Power Management

All device power states are supported with the exception of D3cold. Only L0s ASPM mode is supported if enabled by configuring the Link Control register's bits 1-0 in configuration space. Note that there is no power saving in the controller when the device is put into a non-D0 state. The only power saving is the I/O drivers when the controller is put into a non-L0 link state.

Table 74. Power Management State Supported

Component D-State	Permissible Interconnect State	Action
D0	L0, L0s	In full operation.
D1	L0, L0s, L1	All outbound traffic is stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions.
D2	L0, L0s, L1	All outbound traffic is stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions.
D3hot	L0, L0s, L1, L2/L3 Ready	All outbound traffic is stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions. Note that if a transition of D3hot to D0 occurs, a reset is performed to the controller's configuration space. In addition, link training restarts.
D3cold	L3	Completely off.

18.6.4.1 L2/L3 Ready Link State

The L2/L3 Ready link state is entered after the EP device is put into a D3hot state followed by a PME_Turn_Off/PME_TO_Ack message handshake protocol. Exiting this state requires a POR reset or a WAKE_B signal from the EP device. The PCI Express controller (in EP mode) does not support the generation of beacon; therefore, as an alternative, the device can use one of the GPIO signals as an enable to an external tristate buffer to generate a WAKE_B signal, as shown in the figure below.

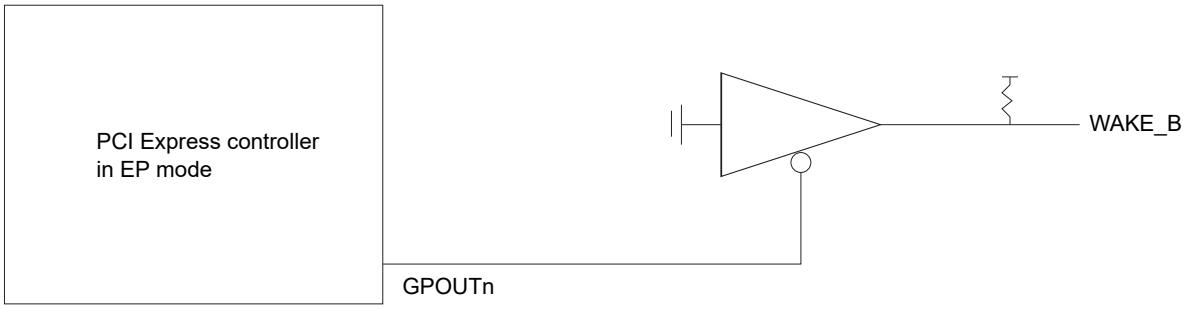


Figure 54. WAKE_B Generation Example

18.6.5 Hot Reset

When a hot reset condition occurs, the controller initiates a clean-up of all outstanding transactions and returns to an idle state. All configuration register bits that are non-sticky are reset. Link training takes place subsequently. As an EP device, it is not permitted to generate a hot reset condition; it can only detected a hot reset condition and initiates the clean-up procedure appropriately.

18.7 Initialization/Application Information

18.7.1 Gen 3 Link Equalization

Link equalization allows components to adjust the transmitter and receiver setup of each lane to improve the signal quality and meet the requirements for operating at Gen 3 data rates. However, link equalization cannot be performd in loopback mode. Before attempting link training in loopback mode link equalization must be disabled by setting the "Equalization Disable" bit in the Gen3 Control Register at PCI Express extended configuration space offset 890h.

18.7.2 Configuring the chip for inbound CCSR accesses

The chip must be configured using the following procedure prior to receiving any inbound memory transactions targeting CCSR space:

1. Program COHERENCY_CONTROL_3_OFF (offset 8E8h) = 0000_0000h
2. Program COHERENCY_CONTROL_2_OFF (offset 8E4h) = 0000_0000h
3. Program COHERENCY_CONTROL_1_OFF (offset 8E0h) = 1000_0000h

This procedure establishes a boundary between CCSR and memory at 1000_0000h in the system memory map.

NOTE

Any inbound memory transactions targeting CCSR space must set NO_SNOOP = 1 in the TLP.

18.7.3 Poisoned TLP handling

In some cases, the handling of inbound Poisoned TLPs needs special attention. Inbound Poisoned TLP requests are dropped and the Poisoned TLP flag in the Uncorrectable Error Status Register (bit 12) is set, provided it is not masked. However, for an inbound Poisoned TLP response to an outbound request, the response is forwarded to the application as a completion and the Poisoned TLP flag in the Uncorrectable Error Status Register (bit 12) is set (again, provided it is not masked). This allows the requesting transaction to terminate without stalling, although the error is flagged. In this case, the error handler should cause the response to be discarded.

Chapter 19
PHY-Timer/Comparator

19.1 PHY-Timer as implemented on the chip

The PHY Timer counter value is output as a 32-bit Timestamp value. There is no distinction between the PHY Timer counter and the Timestamp value; they are always the same.

Comparator outputs of the PHY Timer are categorized as trigger_out or xtrigger_out. xtrigger_out outputs are 1-cycle pulses. xtritter_out triggers are not used by the chip. Instead, all PHY Timer outputs used by the chip are in the trigger_out category. These outputs have configurable assertion and negation characteristics. The PHY Timer includes comparators to generate control outputs, including:

- PA Enable
- LNA Enable (3)
- Tx/Rx Switch Control (2)
- Tx Allowed controls (5)
- Rx Allowed controls (6, including Auxiliary Transmission Power Detection and RSSI)
- Pulse Per Second
- VSPA “Go” signals (2)
- Peak Value Start and Stop signals to AXIQ

Table 75.

Comparator name	Number of comparators
PA Enable	1
LNA Enable	3
Tx/Rx Switch Control	2
Tx Allowed controls	5
Rx Allowed controls (including Auxiliary Transmission Power Detection and RSSI)	6
Pulse Per Second	1
VSPA “Go” signals	2
Peak Value Start and Stop signals to AXIQ	1
Input Capture Trigger	1

The PHY Timer supports Pulse Per Second input and output. Upon receiving a PPS input, the PHY Timer latches the current Timestamp into a register that is visible to software. The PPS_IN signal will be connected to the cmp_capture_in[13] input. The latched timestamp value can then be read in the associated register of the PHY Timer. The PPS_IN signal is captured by the clock of the PHY Timer, so its required minimum pulse width is one 12.5 ns period of that clock. The PPS output signal is generated from a programmable comparator value like other PHY Timer outputs. A pulse width required by the system can be configured in the registers of the PHY Timer associated with the channel 14 output.

PHY Timer Interrupt Ouput-

Unlike primary pin outputs, the PPS_OUT interrupt signal to NVIC from PHY Timer uses a Cross Trigger Output (XTRIGGER OUT output) of the PHY Timer. This is done to make the timing characteristics of the interrupt signal independent of the

software controlled waveform of the PPS_OUT primary output, which is controlled by a TRIGGER_OUT style output. The use of the XTRIGGER_OUT mode for the PPS_OUT interrupt signal requires specific configuration of the PHY Timer to generate this output; specifically the Cross Trigger Enable bit of the associated Comparator Status and Control registers must be set (TM_PHY_TMR_C14SC[CTE]). It also implies that the NVIC interrupt controller must be configured to expect a pulse input for this interrupt signal.

The following table shows proposed alignment of Timer, DMA, and channels.

PHY Timer and VSPA DMA Assignments

PHY Timer Comparator number - output trigger	Timer Function	VSPA DMA	DCS Channel	Primary pin channel function name
—	—	0	—	High Priority SW process
0	VSPA GO[0]	—	—	—
1	Ch1 Rx_Allowed	1	RO ADC 0	RO 0
2	Ch2 Rx_Allowed	2	RO ADC 1	RO 1
3	Ch3 Rx_Allowed	3	RX ADC 0	RX 0
4	Ch4 Rx_Allowed	4	RX ADC 1	RX 1
5	Ch5 Rx_Allowed	5	AUX ADC	TX_DET
6	Ch6 Rx_Allowed	6	—	RSSI
7	—	—	—	—
8	—	—	—	—
9	—	—	—	—
10	—	—	—	—
11	Ch5 Tx_Allowed	11	DAC IQ	DAC IQ
12	VSPA GO[1]			
13	PPS_IN			
14	PPS_OUT			
15	RFCTL-0 [1]			TXRX1
16	RFCTL-1 [1]			TXRX0
17	RFCTL-2 [1]			LNA3_EN
18	RFCTL-3 [1]			LNA2_EN

Table continues on the next page...

Table continued from the previous page...

PHY Timer Comparator number - output trigger	Timer Function	VSPA DMA	DCS Channel	Primary pin channel function name
19	RFCTL-4 [1]			LNA1_EN
20	RFCTL-5 [1]			PA_EN

1. LA9310 PHY-Timer block includes 6 programmable comparators with trigger_out signals connected to external pins. These pins as a group are referred to as RFCTL and allow control of system components with precise timing. If application doesn't require utilization of PHY-Timer based control functions; corresponding pins can be used as GPIO. PMUXCR0 register allows to configure RFCTL pins individually to function as GPIO. When RFCTL pin is configured to function as GPIO, it will not reflect corresponding PHY-Timer comparator trigger_out signal.

VSPA, the PHY Timer, and AXIQ work together to control, calculate, and report peak sample values. The interactions are shown in figure below.

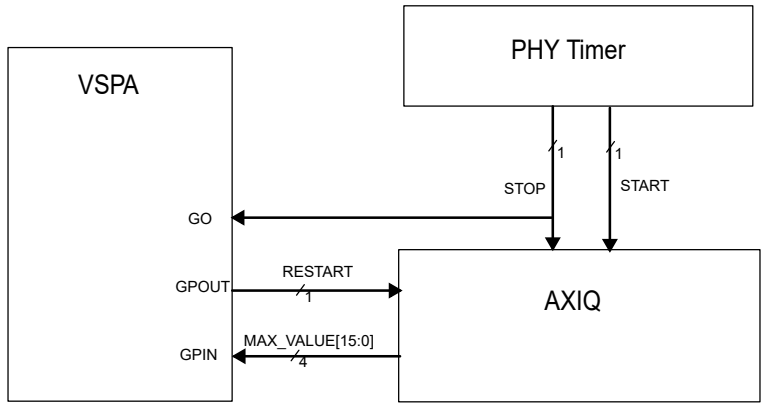


Figure 55. Peak Detection Mechanism

Peak Detection Control definitions are shown in below table.

Table 76. Peak Detection Control Signals

Signal	Source	Destination	Description
START_MAX_SEARCH	PHY Timer Trigger #21	AXIQ	Reset internal max_val Start max value search
STOP_MAX_SEARCH	PHY Timer Trigger #22	AXIQ	Save max_val to saved_max_val Stop max value search
		VSPA GO #3	Trigger VSPA to read peak value on GP Input pins
RESTART_MAX_SEARCH	VSPA GPOUT: GPOUT_7[31]	AXIQ	Save the current max_val into the 'saved_max_val' buffer. Restart the search

Table continues on the next page...

Table 76. Peak Detection Control Signals (continued)

Signal	Source	Destination	Description
SAVED_MAX_VAL	AXIQ: Ch1 - axiq_stat[79:64] Ch2 - axiq_stat[95:80] Ch3 - axiq_stat[111:96] Ch4 - axiq_stat[127:112]	VSPA GPIN: GPIN_2[15:0] GPIN_2[31:16] GPIN_3[15:0] GPIN_3[31:16]	4 16-bit Max Value signals to VSPA via the gp_in_reg vector

19.2 PHY-Timer Architecture

The PHY-Timer consists of a free-running counter, and 23 comparators which can be used to generate a single trigger output per comparator. One comparator is reserved for use as a capture trigger. The PHY timer and comparators provide a means for HRTM firmware to generate time intervals with precise relationships to PHY operations, as is needed for actions such as turning the transmitter off at the end of an outgoing PPDU, or enabling the receiver at the proper instant within SIFS so that an incoming PPDU that begins at the SIFS boundary will be detected. The PHY timer comparators compare against the value of a 32-bit, free-running PHY timer which is counting rising edges of the synchronized slower 76.75 MHz timer input.

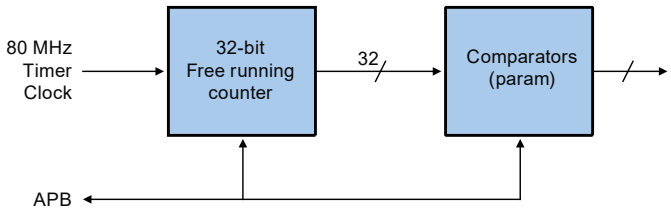


Figure 56. PHY-Timer and External Connections

Features include:

- AMBA APB bus for access to configuration and status registers (32-bit access only)
- External Timer Clock
- 32-bit Free-running Counter (counter increments on rising edge of External Timer Clock Input)
- 23 Trigger Output Signals, one is reserved for use as a capture trigger

19.3 PHY-Timer Comparators

Precise timing of MAC actions that are referenced to activity on the wireless medium, coordination of use of the wireless medium by different on-chip protocol controllers (For example: BLE), as well synchronization between MAC and PHY for activities that have sub-microsecond tolerances, is achieved using a 32-bit free-running counter, known as the PHY-Timer. Within the PHY-Timer, timed activities are generated by PHY Timer Comparators. The PHY timer counts up (modulo 2^{32}) at 76.75 MHz. The comparators can each be configured to generate triggers which can be used in a variety of ways.

The PHY timer comparator control/status and value registers are shown in the table below. The PHY timer count value cannot be read directly. Instead, the current value of the PHY Timer will be captured each clock cycle until the firmware de-asserts the capture request.

Table 77. PHY Timer Registers

Name	Type	31	16	15	0
TM_PHY_TMR_C0SC	RW	Comparator 0 Status & Control			
TM_PHY_TMR_C0V	RW	Comparator 0 Value			
TM_PHY_TMR_C1SC	RW	Comparator 1 Status & Control			
TM_PHY_TMR_C1V	RW	Comparator 1 Value			
-	RW	-			
-	RW	-			
TM_PHY_TMR_CnSC	RW	Comparator n Status & Control			
TM_PHY_TMR_CnV	RW	Comparator n Value			

Each TM_PHY_CnV register holds a 32-bit value that, if the comparator is enabled, generates a trigger request when the PHY Timer value is equal to the register contents. The comparator is enabled by a write to the TM_PHY_CnV register, and disabled when the compare equal occurs or the CMPE bit is cleared in the corresponding TM_PHY_CnV register. The TM_PHY_CnV value can be read at any time without effect on the enable state of the comparator. When using a PHY Timer Comparator the TM_PHY_CnSC register should be written to set the desired states of CIF, CTE, and CompTrig before enabling the comparator by writing TM_PHY_CnV.

Where

- TVAL is a read-only bit that returns the current state of the trigger output
- CAP_EDGE is the capture edge flag. When the cmp_capture_in signal is used, CAP_EDGE = 0 indicates the PHY_TMR value should be captured into the TM_PHY_CnV register on the rising edge of the signal. CAP_EDGE = 1 indicates the PHY_TMR value should be captured into the TM_PHY_CnV register on the falling edge of the cmp_capture_in signal.
- CIF is the comparator interrupt flag. On read, CIF = 0 means a compare equal has not occurred and CIF =1 means a compare equal has occurred. Writing 1 to this bit clears CIF, writing 0 to this bit has no effect.
- CMPE is the comparator enable flag. On read, CMPE = 0 means the comparator is disabled and CMPE = 1 means the comparator is enabled. Writing a 1 to this bit disables the comparator, writing a 0 to this bit has no effect. (The comparator is enabled by writing the compare value to the TM_PHY_CnV register.)
- CAP causes capture of the current PHY Timer value. Writing 1 to this bit causes the PHY Timer value to be loaded into the corresponding TM_PHY_CnV register each valid clock cycle until this bit is written to 0. The typical use is to set CAP = 1 and back to 0 to record the start time of a function triggered by firmware (using a non-zero DIR_TRIG value).
- DIR_TRIG permits firmware to generate a USM trigger directly. The value written to this field is interpreted as follows:
  - 00 no change to trigger output signal from this comparator
  - 01 force trigger output signal to 0
  - 10 force trigger output signal to 1
  - 11 invert the state of the trigger output signal
  - The value of DIR_TRIG should always be written as 00 when writing to TM_PHY_CnSC while the corresponding comparator is enabled. If the comparator is not known to be disabled and a direct trigger needs to be performed

the comparator should be disabled by writing CMPE=1 with DIR_TRIG=00 at least one instruction before writing a non-zero value to DIR_TRIG.

- CMP_TRIG selects the trigger output signal change upon compare equal. The value written to this field is interpreted as follows:
  - 00 no change to trigger output on compare equal
  - 01 force trigger output signal to 0 on compare equal
  - 10 force trigger output signal to 1 on compare equal
  - 11 invert the state of the trigger output signal on compare equal
  - The value of CMP_TRIG should be set before enabling the comparator, and should not be changed while the comparator is enabled.

19.4 PHY-Timer Integration

19.4.1 PHY-TIMER register descriptions

19.4.1.1 PHYTIMER Memory map

PHYTIMER base address: 102_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Comparator Control (TM_PHY_TMR_CTRL)	32	RW	0000_0000h
4h	Comparator Status and Control (TM_PHY_TMR_C0SC)	32	RW	0000_0000h
8h	Comparator Value (TM_PHY_TMR_C0V)	32	RW	0000_0000h
Ch	Comparator Status and Control (TM_PHY_TMR_C1SC)	32	RW	0000_0000h
10h	Comparator Value (TM_PHY_TMR_C1V)	32	RW	0000_0000h
14h	Comparator Status and Control (TM_PHY_TMR_C2SC)	32	RW	0000_0000h
18h	Comparator Value (TM_PHY_TMR_C2V)	32	RW	0000_0000h
1Ch	Comparator Status and Control (TM_PHY_TMR_C3SC)	32	RW	0000_0000h
20h	Comparator Value (TM_PHY_TMR_C3V)	32	RW	0000_0000h
24h	Comparator Status and Control (TM_PHY_TMR_C4SC)	32	RW	0000_0000h
28h	Comparator Value (TM_PHY_TMR_C4V)	32	RW	0000_0000h
2Ch	Comparator Status and Control (TM_PHY_TMR_C5SC)	32	RW	0000_0000h
30h	Comparator Value (TM_PHY_TMR_C5V)	32	RW	0000_0000h
34h	Comparator Status and Control (TM_PHY_TMR_C6SC)	32	RW	0000_0000h
38h	Comparator Value (TM_PHY_TMR_C6V)	32	RW	0000_0000h
3Ch	Comparator Status and Control (TM_PHY_TMR_C7SC)	32	RW	0000_0000h
40h	Comparator Value (TM_PHY_TMR_C7V)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

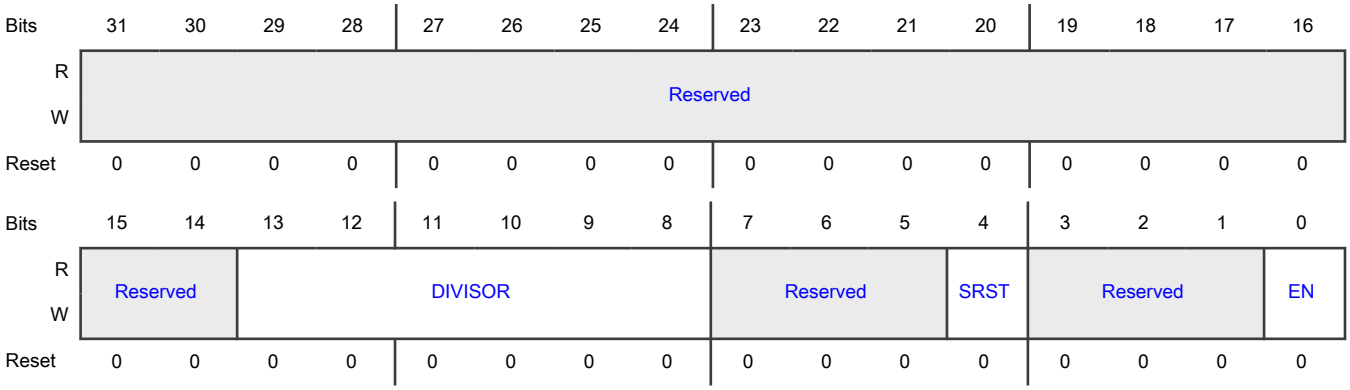
Offset	Register	Width (In bits)	Access	Reset value
44h	Comparator Status and Control (TM_PHY_TMR_C8SC)	32	RW	0000_0000h
48h	Comparator Value (TM_PHY_TMR_C8V)	32	RW	0000_0000h
4Ch	Comparator Status and Control (TM_PHY_TMR_C9SC)	32	RW	0000_0000h
50h	Comparator Value (TM_PHY_TMR_C9V)	32	RW	0000_0000h
54h	Comparator Status and Control (TM_PHY_TMR_C10SC)	32	RW	0000_0000h
58h	Comparator Value (TM_PHY_TMR_C10V)	32	RW	0000_0000h
5Ch	Comparator Status and Control (TM_PHY_TMR_C11SC)	32	RW	0000_0000h
60h	Comparator Value (TM_PHY_TMR_C11V)	32	RW	0000_0000h
64h	Comparator Status and Control (TM_PHY_TMR_C12SC)	32	RW	0000_0000h
68h	Comparator Value (TM_PHY_TMR_C12V)	32	RW	0000_0000h
6Ch	Comparator Status and Control (TM_PHY_TMR_C13SC)	32	RW	0000_0000h
70h	Comparator Value (TM_PHY_TMR_C13V)	32	RW	0000_0000h
74h	Comparator Status and Control (TM_PHY_TMR_C14SC)	32	RW	0000_0000h
78h	Comparator Value (TM_PHY_TMR_C14V)	32	RW	0000_0000h
7Ch	Comparator Status and Control (TM_PHY_TMR_C15SC)	32	RW	0000_0000h
80h	Comparator Value (TM_PHY_TMR_C15V)	32	RW	0000_0000h
84h	Comparator Status and Control (TM_PHY_TMR_C16SC)	32	RW	0000_0000h
88h	Comparator Value (TM_PHY_TMR_C16V)	32	RW	0000_0000h
8Ch	Comparator Status and Control (TM_PHY_TMR_C17SC)	32	RW	0000_0000h
90h	Comparator Value (TM_PHY_TMR_C17V)	32	RW	0000_0000h
94h	Comparator Status and Control (TM_PHY_TMR_C18SC)	32	RW	0000_0000h
98h	Comparator Value (TM_PHY_TMR_C18V)	32	RW	0000_0000h
9Ch	Comparator Status and Control (TM_PHY_TMR_C19SC)	32	RW	0000_0000h
A0h	Comparator Value (TM_PHY_TMR_C19V)	32	RW	0000_0000h
A4h	Comparator Status and Control (TM_PHY_TMR_C20SC)	32	RW	0000_0000h
A8h	Comparator Value (TM_PHY_TMR_C20V)	32	RW	0000_0000h
ACh	Comparator Status and Control (TM_PHY_TMR_C21SC)	32	RW	0000_0000h
B0h	Comparator Value (TM_PHY_TMR_C21V)	32	RW	0000_0000h
B4h	Comparator Status and Control (TM_PHY_TMR_C22SC)	32	RW	0000_0000h
B8h	Comparator Value (TM_PHY_TMR_C22V)	32	RW	0000_0000h

19.4.1.2 Comparator Control (TM_PHY_TMR_CTRL)

Offset

Register	Offset
TM_PHY_TMR_CTRL	0h

Diagram



Fields

Field	Function
31-14 —	Reserved.
13-8 DIVISOR	DIVISOR sets the PHY-TIMER clock divide. The frequency of the timer CLK can be divided by the values of 1 through 63. Please note that a DIVISOR of either 0 or 1 result in a divide by 1. All other values result in a divide by that value. This bit field does not have any enumerations.
7-5 —	Reserved.
4 SRST	SRST is a soft reset to allow reseting the PHY-TIMER count to 0. The timer is reset by setting bit to 1 and then setting bit to 0. The minimum time the reset needs to be set to 1 is one phytmr clk/ divisor value. This bit field does not have any enumerations.
3-1 —	Reserved.
0 EN	PHY-TIMER enable.

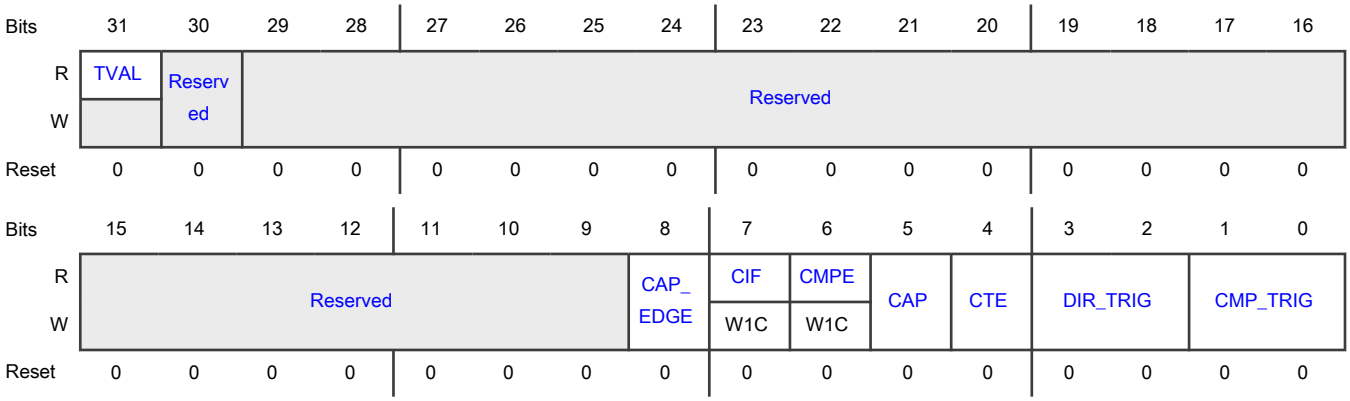
19.4.1.3 Comparator Status and Control (TM_PHY_TMR_C0SC - TM_PHY_TMR_C22SC)

Offset

For n = 0 to 22:

Register	Offset
TM_PHY_TMR_CnSC	4h + (n × 8h)

Diagram



Fields

Field	Function
31 TVAL	Returns the current value of the trigger output signal Read-only signal
30 —	Reserved.
29-9 —	Reserved.
8 CAP_EDGE	Edge indicator for hardwired capture bit to capture current PHY Timer value into CnV register Writing 0 to this bit causes the current PHY Timer value to be latched into the corresponding TM_PHY_CnV register upon rising edge of hardware cmp_capture_in bit. Writing 1 to this bit causes PHY Timer value to be caught on the falling edge of corresponding hardware cmp_capture_in bit.
7 CIF	Comparator interrupt flag Upon read, CIF = 0 means a compare equal has not occurred and CIF = 1 means a compare equal has occurred. Writing 1 to this bit clears CIF, wirting 0 to this bit has no effect.
6 CMPE	Capture of the current PHY Timer value into CnV register Writing 1 to this bit causes the current PHY Timer value to be loaded into the corresponding TM_PHY_CnV register. Bit should be set to 0 before programming the CnV with a new compare value.

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 CAP	Capture of the current PHY Timer value into CnV register Writing 1 to this bit causes the current PHY Timer value to be loaded into the corresponding TM_PHY_CnV register. Bit should be set to 0 before programming the CnV with a new compare value.
4 CTE	Cross-trigger enable This bit field does not have any enumerations. CTE = 1 enables pulse trigger generation.
3-2 DIR_TRIG	Permits firmware to generate a trigger directly This bit field has a 2-bit enumeration. 00b - No change to trigger output signal 01b - Force trigger output signal to 0 10b - Force trigger output signal to 1 11b - Invert the state of the trigger output signal
1-0 CMP_TRIG	Selects trigger output signal change up on compare equal This bit field has a 2-bit enumeration. 00b - No change to trigger output on compare equal 01b - Force trigger output signal to 0 on compare equal 10b - Force trigger output signal to 1 on compare equal 11b - Invert the state of the trigger output signal on compare equal

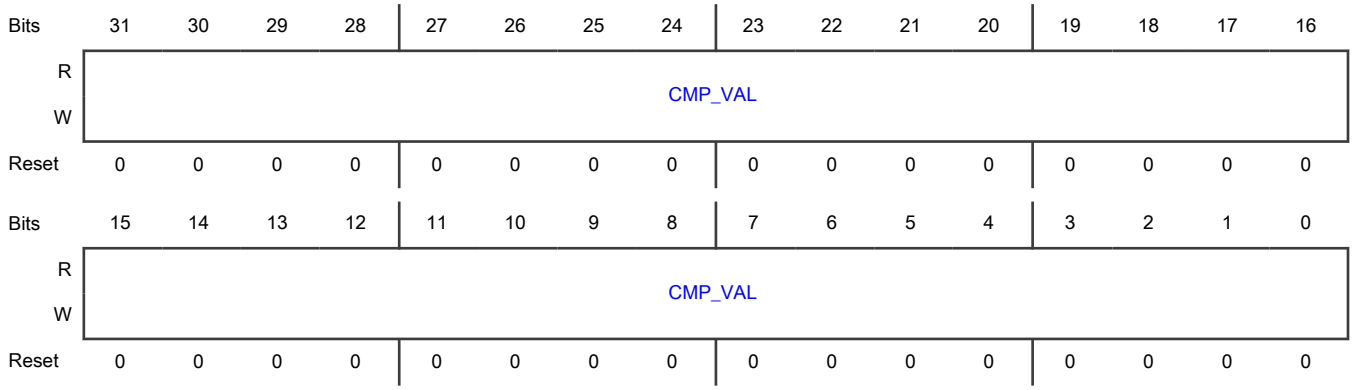
19.4.1.4 Comparator Value (TM_PHY_TMR_C0V - TM_PHY_TMR_C22V)

Offset

For n = 0 to 22:

Register	Offset
TM_PHY_TMR_CnV	8h + (n × 8h)

Diagram



Fields

Field	Function
31-0 CMP_VAL	Compare equal value This bit field has no enumeration.

Chapter 20
SerDes Module

20.1 SerDes Module as Implemented on the chip

The chip supports a single lane PCIe Gen3 SerDes option.

20.2 Overview

The SerDes module implements link serialization/deserialization and PCS functions for high speed serial interfaces.

20.2.1 Features

The SerDes module includes 1 data lanes and 1 PLL and supports the following protocols:

- PCI Express 3.0 (November, 2010) @ 2.5, 5, 8 Gbaud
- IEEE 1149.1 and 1149.6-2003 for JTAG

20.3 Modes of Operation

The SerDes supports several combinations of protocols and frequencies. See SerDes Lane Assignments and Multiplexing, for details on the combinations.

20.4 External Signals Description

The table below lists the external signals for the SerDes.

Table 78. SerDes Interface Signals

Pin Name	Description	No. of Signals	I/O
SDn_TX_P sd_*_tx_p	Transmitter serial output, positive data	1	O
SDn_TX_N sd_*_tx_n	Transmitter serial output, negative data	1	O
SDn_IMP_CAL_TX	Tx Impedance Calibration	1	I
SDn_RX_P sd_*_rx_p	Receiver serial output, positive data	1	I
SDn_RX_N sd_*_rx_n	Receiver serial output, negative data	1	I
SDn_IMP_CAL_RX	Rx Impedance Calibration	1	I
x=1			
SDn_REFx_CLK_P	Reference clock input to PLLx		I

Table continues on the next page...

Table 78. SerDes Interface Signals (continued)

Pin Name	Description	No. of Signals	I/O
sd_ref_clkx			
SDn_REFx_CLK_N sd_ref_clkx_b	Reference clock-bar input to PLLx		I

20.5 SerDes Lane Assignments and Multiplexing

20.5.1 Top-level SerDes Memory Map

The SerDes register map occupies 8K bytes of memory-mapped space, allocated as in the table below

Table 79. SerDes Memory Map Summary

Register Offset	Description
0x0000-0x001F	PLL1 Control Registers
0x0020-0x003F	Reserved
0x0040-0x007F	Reserved
0x0080-0x00FF	Process Monitor, Calibration, Test, Miscellaneous Control/Status Registers
0x0100-0x01FF	Global Protocol Status Registers
0x0200-0x02FF	Global Protocol Configuration Registers
0x0800-0x09FF	SRDS Control Registers for lanes
0x0A00-0x0FFF	Reserved
0x1000-0x1FFF	Protocol Control Registers

20.6 SerDes register descriptions

The SerDes module is programmed by control/status registers (CSRs). The CSRs are used for mode control, and to extract status information. All accesses to and from the registers must be made as 32-bit accesses. There is no support for accesses of sizes other than 32 bits. Writes to reserved register bits must always preserved the previous value; that is, the register should be programmed by reading the value, modifying appropriate fields, and writing back the value. Unless otherwise specified, the read value of unmapped registers or of reserved bits in mapped registers is not defined, and must not be assumed to be 0.

The table below lists the address, name, and a cross-reference to the complete description of each register. The offsets to the memory map table are defined for each SerDes module from 0x0000 to 0x1FFF (8 KB).

Unlisted register addresses are reserved.

- Reserved fields are always ignored for the purposes of determining access type.
- Reserved registers and fields must be preserved on writes.
- R (read only) indicates that all the non-reserved fields in a register are read-only.

- R/W (read/write) indicates all of the non-reserved fields in a register are either read/write or read-only, with at least one read/write field. The register description indicates which fields are read-write and which read-only.
- RC (read clear) indicates all of the fields in a register are cleared on read, are are not otherwise writeable.
- W1C indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description provide the details for access.

This section describes the control, status, and test registers for SerDes and Protocol PCS layers.

All reserved register fields must be preserved on register writes (read-modified-write).

20.6.1 SerDes Memory map

SerDes base address: 1EA_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	SerDes PLL1 Reset Control Register (PLL1RSTCTL)	32	RW	0000_0020h
4h	SerDes PLL1 Control Register 0 (PLL1CR0)	32	RW	8080_0008h
8h	SerDes PLL1 Control Register 1 (PLL1CR1)	32	RW	0800_0000h
90h	SerDes Transmit Calibration Control Register (TCALCR)	32	RW	0000_0000h
A0h	Receive Calibration Control Register (RCALCR)	32	RW	0000_0000h
B0h	General Control Register 0 (GR0)	32	RW	0000_0000h
100h	Lane A Protocol Select Status Register 0 (LNAPSSR0)	32	RO	0000_0000h
200h	Protocol Configuration Register 0 (PCCR0)	32	RW	0000_0000h
300h	PCIe Equalization Configuration Register (PEXEQCR)	32	RW	0000_0C10h
304h	PCIe Equalization Preset 0 Register (PEXEQP0CR)	32	RW	0000_C000h
308h	PCIe Equalization Preset 1 Register (PEXEQP1CR)	32	RW	0000_8000h
30Ch	PCIe Equalization Preset 2 Register (PEXEQP2CR)	32	RW	0000_A000h
310h	PCIe Equalization Preset 3 Register (PEXEQP3CR)	32	RW	0000_6000h
314h	PCIe Equalization Preset 4 Register (PEXEQP4CR)	32	RW	0000_0000h
318h	PCIe Equalization Preset 5 Register (PEXEQP5CR)	32	RW	0000_0005h
31Ch	PCIe Equalization Preset 6 Register (PEXEQP6CR)	32	RW	0000_0006h
320h	PCIe Equalization Preset 7 Register (PEXEQP7CR)	32	RW	0000_9005h
324h	PCIe Equalization Preset 8 Register (PEXEQP8CR)	32	RW	0000_6006h
328h	PCIe Equalization Preset 9 Register (PEXEQP9CR)	32	RW	0000_0008h
32Ch	PCIe Equalization Preset 10 Register (PEXEQP10CR)	32	RW	0000_A006h
800h	General Control Register 0 - Lane A (LNAGCR0)	32	RW	1104_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
804h	General Control Register 1 - Lane A (LNAGCR1)	32	RW	004C_4011h
80Ch	Speed Switch Control Register 0 - Lane A (LNASSCR0)	32	RW	9800_2B00h
810h	Receive Equalization Control Register 0 - Lane A (LNARECR0)	32	RW	0000_001Fh
814h	Receive Equalization Control Register 1- Lane A (LNARECR1)	32	RO	0000_0008h
818h	Transmit Equalization Control Register 0 - Lane A (LNATECR0)	32	RW	1028_3000h
81Ch	Speed Switch Control Register 1- Lane 0 (LNASSCR1)	32	RW	0000_0000h
820h	TTL Control Register 0 - Lane A (LNATTLCR0)	32	RW	0000_0400h
83Ch	Test Control/Status Register 3 - Lane A (LNATCSR3)	32	RW	0400_0000h
1000h	PEXA Protocol Control Register 0 (PEXACR0)	32	RW	2000_0000h

20.6.2 SerDes PLL1 Reset Control Register (PLL1RSTCTL)

Offset

Register	Offset
PLL1RSTCTL	0h

Function

PLL*n*RSTCTL contains control and status bits for the SerDes PLL *n* reset state machine.

This register should not be modified unless RSMCS=0111 (RST_ERR=1), 1000 (RST_DONE=1), or 1111 (waiting for RSTREQ).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSTREQ	RST_DONE	RST_ERR	Reserved	Reserved			Reserved	Reserved				Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				Reserved				PLLRST_B	SDRST_B	SDEN	Reserved	Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Fields

Field	Function
31 RSTREQ	Reset request PLL Reset Request - write 1 self-clearing. Software setting of 1 initiates a soft reset of SerDes PLL *n*, along with all lanes using PLL *n*. Hardware automatically clears this bit before reset is done. Note: This field must be 0 if PLL2 does not have a reference clock. Clearing this bit during the reset sequence has no effect. 0b - Not requesting PLL soft reset. 1b - Requesting PLL soft reset.
30 RST_DONE	Reset done PLL Reset Done from Control Block State Machine 0b - PLL reset sequence in progress. 1b - PLL reset sequence complete.
29 RST_ERR	Reset Error No PLL Lock before counter time_out 0b - Normal function 1b - PLL lock didn't happen in the expected time period
28 —	Reserved
27-25 —	Reserved
24 —	Reserved
23-20 —	Reserved
19-16 —	Reserved
15-12 —	Reserved
11-8 —	Reserved
7	PLL reset. Resets PLLn

Table continues on the next page...

Table continued from the previous page...

Field	Function
PLLRST_B	PLL reset. Resets PLLn . Default value: 0 The SerDes reset state machine overrides this field to 0 in some states. Setting PLLRST_B=0 when RST_DONE=0 forces a SerDes PLL reset regardless of the reset state machine state, and may cause the reset state machine to time out and set RST_ERR=1. 0b - Force PLL reset 1b - Application Mode, unless RSTREQ=1
6 SDRST_B	SRDS group reset. Resets all lanes operating from PLLn The SerDes reset state machine overrides this field to 0 in some states. 0b - Force SerDes group reset 1b - Application Mode, unless RSTREQ=1
5 SDEN	SerDes enable. Enable the section of the SerDes module clocked by PLLn SerDes enable. Enable the section of the SerDes module clocked by PLLn. Default value: 1 The SerDes reset state machine overrides this field to 0 in some states. Setting SDEN=0 when RST_DONE=0 forces a SerDes powerdown regardless of the reset state machine state, and may cause the reset state machine to time out and set RST_ERR=1. Note: This field must be 0 if PLL2 does not have a reference clock. 0b - Force SerDes power down 1b - Application Mode, unless RSTREQ=1
4 —	Reserved
3-0 —	Reserved

20.6.3 SerDes PLL1 Control Register 0 (PLL1CR0)

Offset

Register	Offset
PLL1CR0	4h

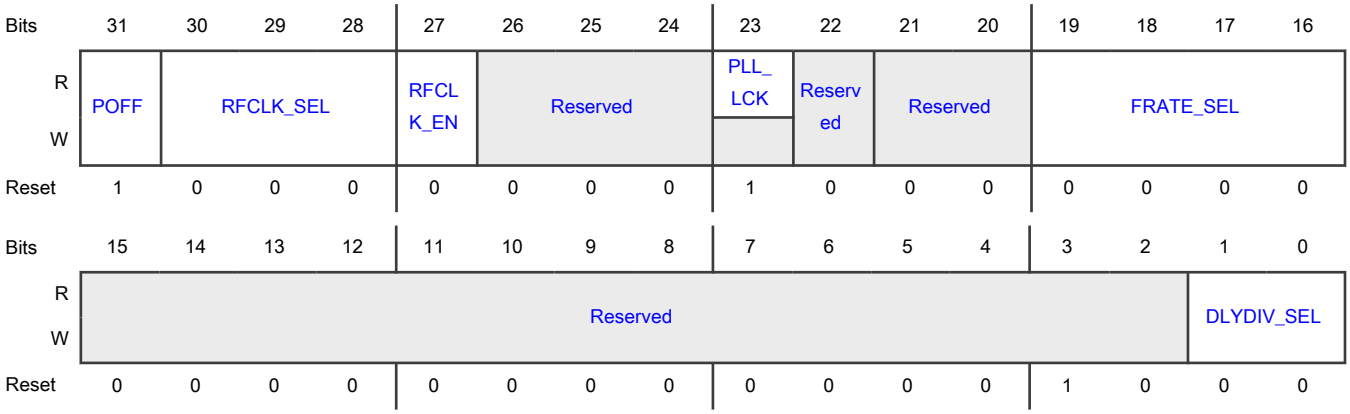
Function

PLLnCR0 contains control and status bits for SerDes PLL *n*

NOTE

This register may be automatically updated when PCI Express is active on the SerDes. Only write to this register when all associated PCI Express controllers are inactive.

Diagram



Fields

Field	Function
31 POFF	PLL off Power down an unused PLL Default value: 0 0b - PLL on 1b - PLL off. All lanes referencing this PLL must also be disabled.
30-28 RFCLK_SEL	Reference clock frequency select All others reserved Recommended setting per protocol: PCIe: 000 or 001 000b - 100 MHz 001b - 125 MHz 010b - 156.25 MHz
27 RFCLK_EN	Reference clock observe enable Default value: 0 Recommended setting: 0 0b - Disable reference clock output 1b - Enable buffered version of SD_REF_CLKn
26-24 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
23 PLL_LCK	PLL lock Indicates PLL(n) has calibrated and locked 0b - PLL is not locked 1b - PLL is locked
22 —	Reserved
21-20 —	Reserved
19-16 FRATE_SEL	Select frequency of PLL VCO. All lanes within a group must operate at a multiple of this frequency and within specified limits of the protocol(s). Recommended settings per protocol: All others are reserved 0000b - 5.00 GHz ---------- [PCIe] 0111b - 4.00 GHz ---------- [PCIe]
15-2 —	Reserved
1-0 DLYDIV_SEL	Select PLLn_ex_dly_clk divider value: All other values reserved 00 - PLLn_ex_dly_clk off

20.6.4 SerDes PLL1 Control Register 1 (PLL1CR1)

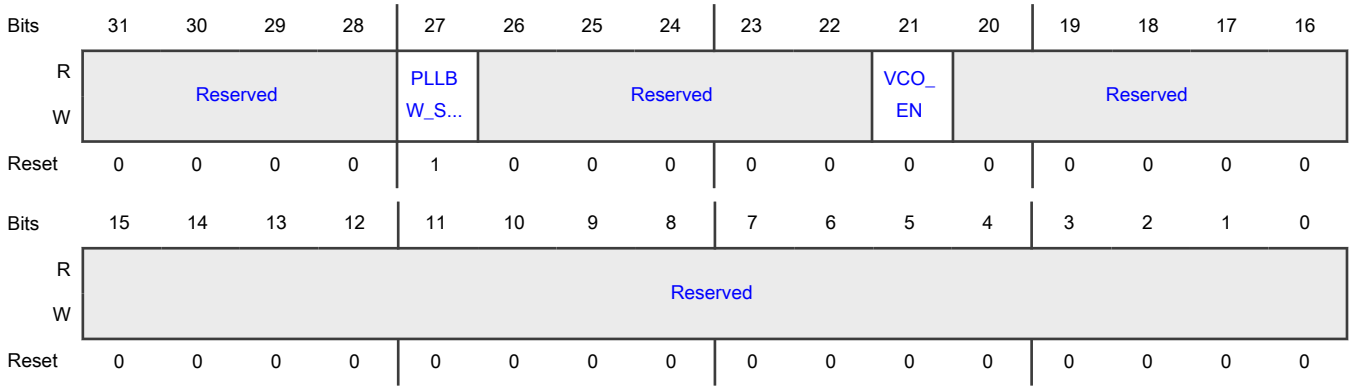
Offset

Register	Offset
PLL1CR1	8h

Function

PLLnCR1 contains control bits for SerDes PLL *n*.

Diagram



Fields

Field	Function
31-28 —	Reserved
27 PLLBW_SEL	Select Higher PLL(n) Bandwidth Recommended setting per PLL: 1 0b - Nominal PLL Bandwidth 1b - PLL Bandwidth Setting
26-22 —	Reserved
21 VCO_EN	Select VCO for use in PLLn Recommended settings per PLLnCR0[FRATE_SEL] and LNnGCR0[T/RRAT_SEL]: • 5.00 GHz: full/half speed: 0 or 1 • 4 GHz: double speed: 0 • 3.125 GHz: full speed: 1 All others reserved Note: if running two LC VCOs at the same frequency, the reference clock for the two PLLs must be from a common source. 0b - Use LC VCO 1b - Use ring VCO
20-0 —	Reserved

20.6.5 SerDes Transmit Calibration Control Register (TCALCR)

Offset

Register	Offset
TCALCR	90h

Function

TCALCR contains the control bits used for Transmit Calibration.

Diagram

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

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

Fields

Field	Function
31-28 —	Reserved
27 CALRST_B	Reset the transmit calibration During POR (warm reset sequence) it is active and is released when the first PLL comes out of reset. 0b - Reset 1b - Application Mode
26-0 —	Reserved

20.6.6 Receive Calibration Control Register (RCALCR)

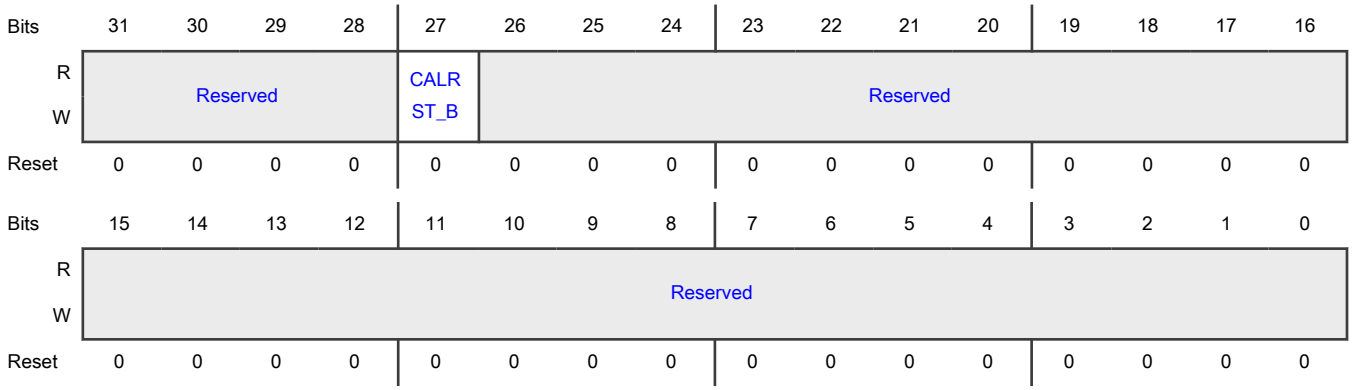
Offset

Register	Offset
RCALCR	A0h

Function

RCALCR contains the control bits for Receive Calibration.

Diagram



Fields

Field	Function
31-28 —	Reserved
27 CALRST_B	Reset the receiver calibration During POR (warm reset sequence) it is active and is released when the first PLL comes out of reset 0b - Reset 1b - Application Mode
26-0 —	Reserved

20.6.7 General Control Register 0 (GR0)

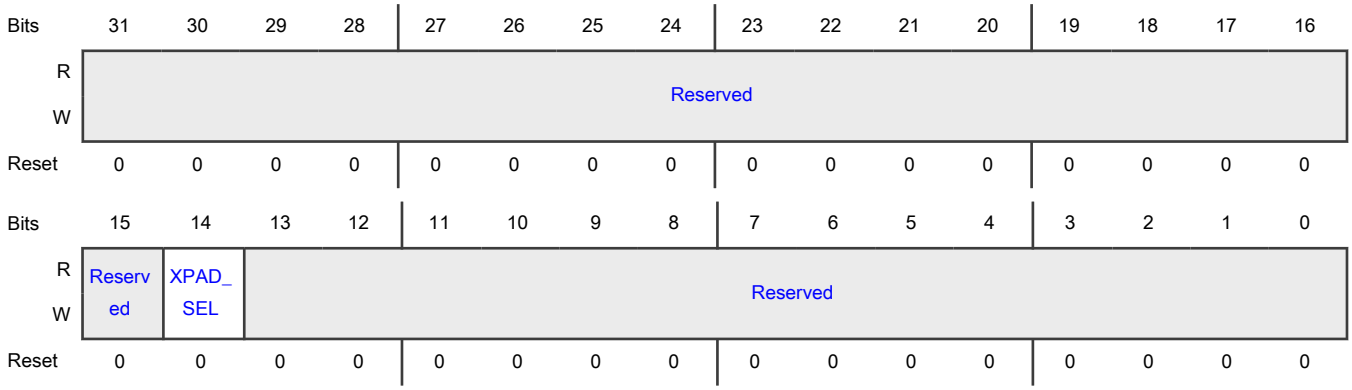
Offset

Register	Offset
GR0	B0h

Function

GR0 contains general control/status bits for the SerDes.

Diagram



Fields

Field	Function
31-15 —	Reserved
14 XPAD_SEL	Describes to SerDes module the value of the power supply being used by the Serdes I/Os: Full SerDes reset required after changing this value. **Warning**: setting XPAD_SEL to low xpadvdd supply when the supply is > 1.35V+5% may cause irreparable damage to the device. 0b - High xpadvdd supply (nominal 1.5V - see H/W spec for details) 1b - Low xpadvdd supply (nominal 1.35V - see H/W spec for details)
13-0 —	Reserved

20.6.8 Lane A Protocol Select Status Register 0 (LNAPSSR0)

Offset

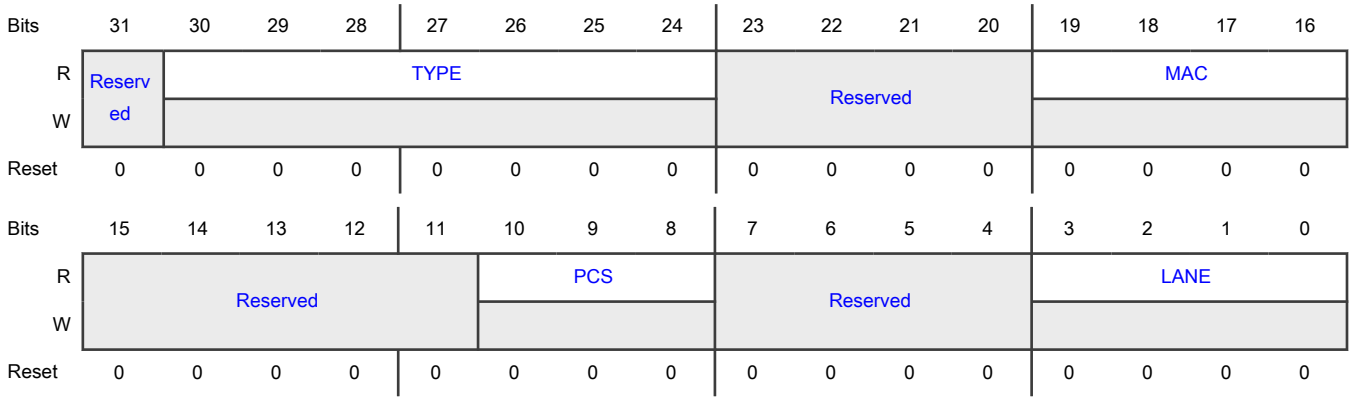
Register	Offset
LNAPSSR0	100h

Function

LNmPSSR0 contains decoded protocol select information per lane.

The primary usage of this register is to map an Ethernet link to its corresponding MAC and MDIO address space.

Diagram



Fields

Field	Function
31 —	Reserved
30-24 TYPE	Procotol Type Bits 1-5 are same as LNmGCR0[PROTS]. Bits 6-7: • Needs to be set to: 00
23-20 —	Reserved
19-16 MAC	MAC instance 0000b - MAC1 0001b - MAC2 0010b - MAC3 0011b - MAC4 0100b - MAC5 0101b - MAC6 0110b - MAC7 0111b - MAC8
15-11 —	Reserved
10-8 PCS	PCS instance of TYPE within PHY 000b - PCSa/1

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001b - PCSb/2 010b - PCSc/3
7-4 —	Reserved
3-0 LANE	Lane number within PCS Example 1: for x4 PCIe [3:0] on lanes [3:0], LANE=0 for n=0, LANE=1 for n=1, LANE=2 for n=2 and LANE=3 for n=3 Note that the lane number within PCS does not take into account lane reversal, either via auto-negotiate, or by S/W control in SerDes registers. All others reserved 0000b - lane 0 0001b - lane 1 0010b - lane 2

20.6.9 Protocol Configuration Register 0 (PCCR0)

Offset

Register	Offset
PCCR0	200h

Function

PCCR0 contains the protocol configuration for PCIe.

Diagram

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

Fields

Field	Function
31 —	Reserved
30-28 PEXA_CFG	PEXa Configuration All others reserved 000b - Disabled 011b - x1 on Lanes 0
27 —	Reserved
26-24 —	Reserved
23 —	Reserved
22-20 —	Reserved
19 —	Reserved
18-16 —	Reserved
15-0 —	Reserved

20.6.10 PCIe Equalization Configuration Register (PEXEQCR)

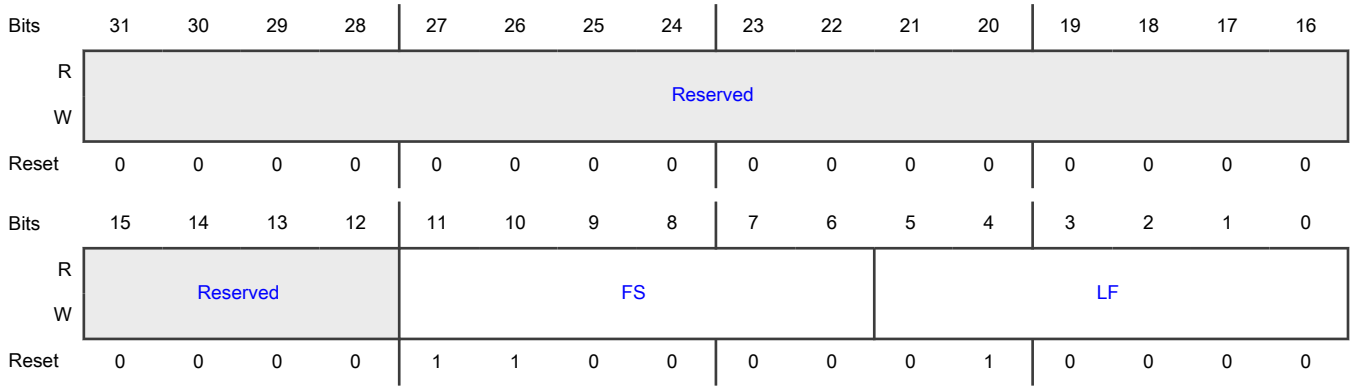
Offset

Register	Offset
PEXEQCR	300h

Function

PEXEQCR contains the PCIe gen3 Tx equalization configuration values.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-6 FS	PCI Express FS value. This field determines the FS value advertised during 8.0 GT/s link equalization phase 1, and is also used to calculate the preset C(0). Default value = 0x30 This value applies to all PCIe links on
5-0 LF	PCI Express LF value. This field determines the LF value advertised during 8.0 GT/s link equalization phase 1. Default value =0x10 This value applies to all PCIe links on

20.6.11 PCIe Equalization Preset 0 Register (PEXEQP0CR)

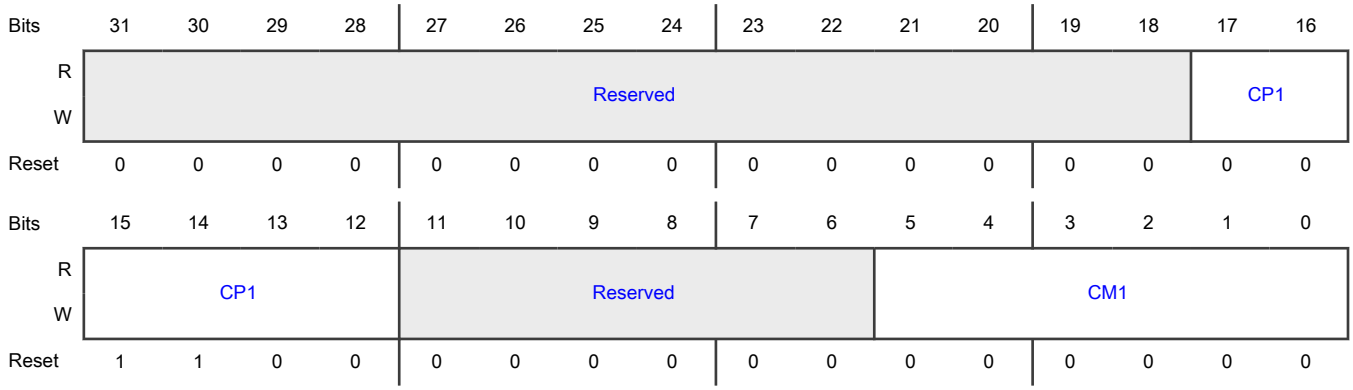
Offset

Register	Offset
PEXEQP0CR	304h

Function

PEXEQPyCR (y=0-10) contains the PCIe gen3 Tx equalization preset values. Note that C(0) is automatically calculated from C(-1), C(+1), and PEXEQCR[FS].

Diagram



Fields

Field	Function
31-18 —	Reserved
17-12 CP1	C(+1) preset value. Default values for preset a= 000000b - 6'b00_1100 000001b - 6'b00_1000 000010b - 6'b00_1010 000011b - 6'b00_0110 000100b - 6'b00_0000 000101b - 6'b00_0000 000110b - 6'b00_0000 000111b - 6'b00_1001 001000b - 6'b00_0110 001001b - 6'b00_0000 001010b - 6'b00_1010
11-6 —	Reserved
5-0 CM1	C(-1) preset value. Default values for preset a= 000000b - 6'b00_0000 000001b - 6'b00_0000 000010b - 6'b00_0000

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000011b - 6'b00_0000 000100b - 6'b00_0000 000101b - 6'b00_0101 000110b - 6'b00_0110 000111b - 6'b00_0101 001000b - 6'b00_0110 001001b - 6'b00_1000 001010b - 6'b00_0110

20.6.12 PCIe Equalization Preset 1 Register (PEXEQP1CR)

Offset

Register	Offset
PEXEQP1CR	308h

Function

PEXEQPyCR (y=0-10) contains the PCIe gen3 Tx equalization preset values. Note that C(0) is automatically calculated from C(-1), C(+1), and PEXEQCR[FS].

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved														CP1	
Reset	0	0	0	0	0	0	0	0	0	0	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	CP1				Reserved						CM1					
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-18 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
17-12 CP1	C(+1) preset value. Default values for preset a= 000000b - 6'b00_1100 000001b - 6'b00_1000 000010b - 6'b00_1010 000011b - 6'b00_0110 000100b - 6'b00_0000 000101b - 6'b00_0000 000110b - 6'b00_0000 000111b - 6'b00_1001 001000b - 6'b00_0110 001001b - 6'b00_0000 001010b - 6'b00_1010
11-6 —	Reserved
5-0 CM1	C(-1) preset value. Default values for preset a= 000000b - 6'b00_0000 000001b - 6'b00_0000 000010b - 6'b00_0000 000011b - 6'b00_0000 000100b - 6'b00_0000 000101b - 6'b00_0101 000110b - 6'b00_0110 000111b - 6'b00_0101 001000b - 6'b00_0110 001001b - 6'b00_1000 001010b - 6'b00_0110

20.6.13 PCIe Equalization Preset 2 Register (PEXEQP2CR)

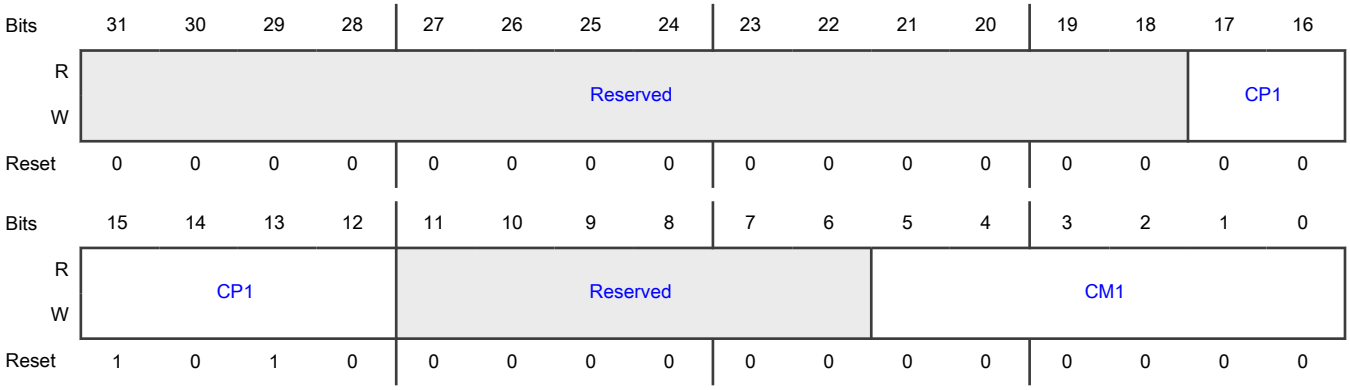
Offset

Register	Offset
PEXEQP2CR	30Ch

Function

PEXEQPyCR (y=0-10) contains the PCIe gen3 Tx equalization preset values. Note that C(0) is automatically calculated from C(-1), C(+1), and PEXEQCR[FS].

Diagram



Fields

Field	Function
31-18 —	Reserved
17-12 CP1	C(+1) preset value. Default values for preset a= 000000b - 6'b00_1100 000001b - 6'b00_1000 000010b - 6'b00_1010 000011b - 6'b00_0110 000100b - 6'b00_0000 000101b - 6'b00_0000 000110b - 6'b00_0000 000111b - 6'b00_1001 001000b - 6'b00_0110

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001001b - 6'b00_0000 001010b - 6'b00_1010
11-6 —	Reserved
5-0 CM1	C(-1) preset value. Default values for preset a= 000000b - 6'b00_0000 000001b - 6'b00_0000 000010b - 6'b00_0000 000011b - 6'b00_0000 000100b - 6'b00_0000 000101b - 6'b00_0101 000110b - 6'b00_0110 000111b - 6'b00_0101 001000b - 6'b00_0110 001001b - 6'b00_1000 001010b - 6'b00_0110

20.6.14 PCIe Equalization Preset 3 Register (PEXEQP3CR)

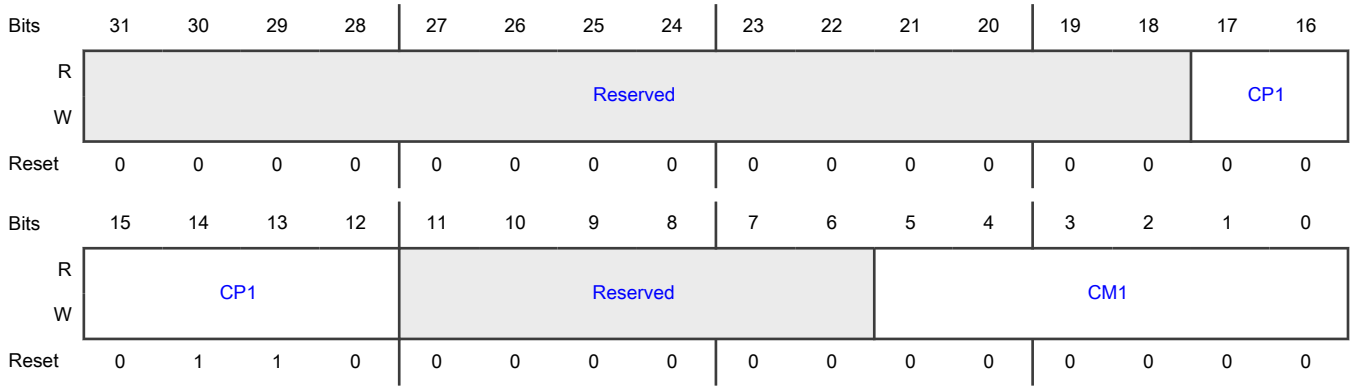
Offset

Register	Offset
PEXEQP3CR	310h

Function

PEXEQPyCR (y=0-10) contains the PCIe gen3 Tx equalization preset values. Note that C(0) is automatically calculated from C(-1), C(+1), and PEXEQCR[FS].

Diagram



Fields

Field	Function
31-18 —	Reserved
17-12 CP1	C(+1) preset value. Default values for preset a= 000000b - 6'b00_1100 000001b - 6'b00_1000 000010b - 6'b00_1010 000011b - 6'b00_0110 000100b - 6'b00_0000 000101b - 6'b00_0000 000110b - 6'b00_0000 000111b - 6'b00_1001 001000b - 6'b00_0110 001001b - 6'b00_0000 001010b - 6'b00_1010
11-6 —	Reserved
5-0 CM1	C(-1) preset value. Default values for preset a= 000000b - 6'b00_0000 000001b - 6'b00_0000 000010b - 6'b00_0000

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000011b - 6'b00_0000 000100b - 6'b00_0000 000101b - 6'b00_0101 000110b - 6'b00_0110 000111b - 6'b00_0101 001000b - 6'b00_0110 001001b - 6'b00_1000 001010b - 6'b00_0110

20.6.15 PCIe Equalization Preset 4 Register (PEXEQP4CR)

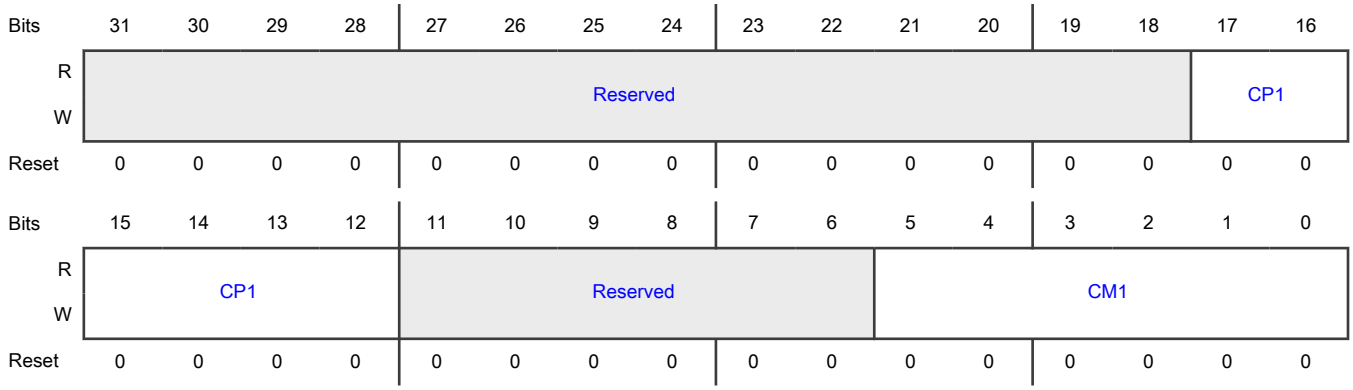
Offset

Register	Offset
PEXEQP4CR	314h

Function

PEXEQPyCR (y=0-10) contains the PCIe gen3 Tx equalization preset values. Note that C(0) is automatically calculated from C(-1), C(+1), and PEXEQCR[FS].

Diagram



Fields

Field	Function
31-18 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
17-12 CP1	C(+1) preset value. Default values for preset a= 000000b - 6'b00_1100 000001b - 6'b00_1000 000010b - 6'b00_1010 000011b - 6'b00_0110 000100b - 6'b00_0000 000101b - 6'b00_0000 000110b - 6'b00_0000 000111b - 6'b00_1001 001000b - 6'b00_0110 001001b - 6'b00_0000 001010b - 6'b00_1010
11-6 —	Reserved
5-0 CM1	C(-1) preset value. Default values for preset a= 000000b - 6'b00_0000 000001b - 6'b00_0000 000010b - 6'b00_0000 000011b - 6'b00_0000 000100b - 6'b00_0000 000101b - 6'b00_0101 000110b - 6'b00_0110 000111b - 6'b00_0101 001000b - 6'b00_0110 001001b - 6'b00_1000 001010b - 6'b00_0110

20.6.16 PCIe Equalization Preset 5 Register (PEXEQP5CR)

Offset

Register	Offset
PEXEQP5CR	318h

Function

PEXEQPyCR (y=0-10) contains the PCIe gen3 Tx equalization preset values. Note that C(0) is automatically calculated from C(-1), C(+1), and PEXEQCR[FS].

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved														CP1	
Reset	0	0	0	0	0	0	0	0	0	0	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	CP1				Reserved						CM1					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Fields

Field	Function
31-18 —	Reserved
17-12 CP1	C(+1) preset value. Default values for preset a= 000000b - 6'b00_1100 000001b - 6'b00_1000 000010b - 6'b00_1010 000011b - 6'b00_0110 000100b - 6'b00_0000 000101b - 6'b00_0000 000110b - 6'b00_0000 000111b - 6'b00_1001 001000b - 6'b00_0110

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001001b - 6'b00_0000 001010b - 6'b00_1010
11-6 —	Reserved
5-0 CM1	C(-1) preset value. Default values for preset a= 000000b - 6'b00_0000 000001b - 6'b00_0000 000010b - 6'b00_0000 000011b - 6'b00_0000 000100b - 6'b00_0000 000101b - 6'b00_0101 000110b - 6'b00_0110 000111b - 6'b00_0101 001000b - 6'b00_0110 001001b - 6'b00_1000 001010b - 6'b00_0110

20.6.17 PCIe Equalization Preset 6 Register (PEXEQP6CR)

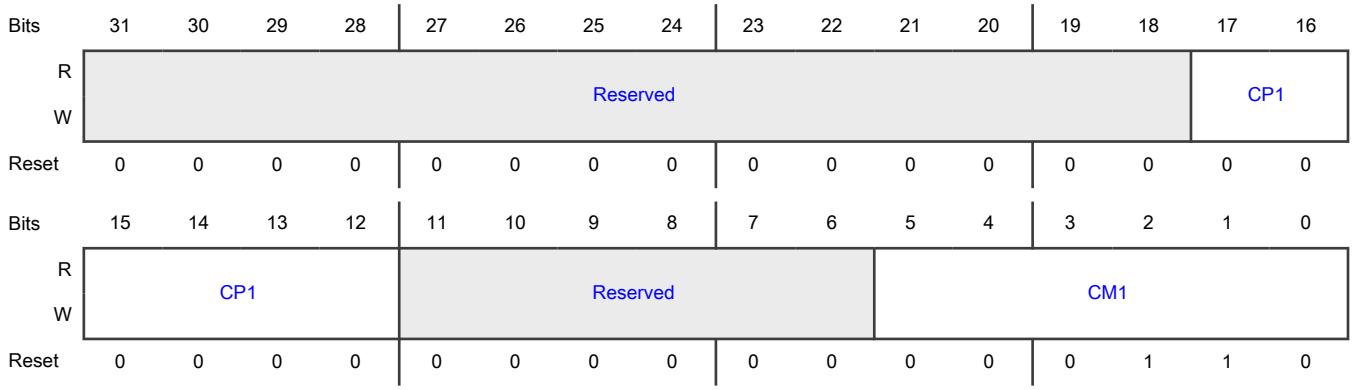
Offset

Register	Offset
PEXEQP6CR	31Ch

Function

PEXEQPyCR (y=0-10) contains the PCIe gen3 Tx equalization preset values. Note that C(0) is automatically calculated from C(-1), C(+1), and PEXEQCR[FS].

Diagram



Fields

Field	Function
31-18 —	Reserved
17-12 CP1	C(+1) preset value. Default values for preset a= 000000b - 6'b00_1100 000001b - 6'b00_1000 000010b - 6'b00_1010 000011b - 6'b00_0110 000100b - 6'b00_0000 000101b - 6'b00_0000 000110b - 6'b00_0000 000111b - 6'b00_1001 001000b - 6'b00_0110 001001b - 6'b00_0000 001010b - 6'b00_1010
11-6 —	Reserved
5-0 CM1	C(-1) preset value. Default values for preset a= 000000b - 6'b00_0000 000001b - 6'b00_0000 000010b - 6'b00_0000

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000011b - 6'b00_0000 000100b - 6'b00_0000 000101b - 6'b00_0101 000110b - 6'b00_0110 000111b - 6'b00_0101 001000b - 6'b00_0110 001001b - 6'b00_1000 001010b - 6'b00_0110

20.6.18 PCIe Equalization Preset 7 Register (PEXEQP7CR)

Offset

Register	Offset
PEXEQP7CR	320h

Function

PEXEQPyCR (y=0-10) contains the PCIe gen3 Tx equalization preset values. Note that C(0) is automatically calculated from C(-1), C(+1), and PEXEQCR[FS].

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved														CP1	
Reset	0	0	0	0	0	0	0	0	0	0	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	CP1				Reserved						CM1					
Reset	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1

Fields

Field	Function
31-18 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
17-12 CP1	C(+1) preset value. Default values for preset a= 000000b - 6'b00_1100 000001b - 6'b00_1000 000010b - 6'b00_1010 000011b - 6'b00_0110 000100b - 6'b00_0000 000101b - 6'b00_0000 000110b - 6'b00_0000 000111b - 6'b00_1001 001000b - 6'b00_0110 001001b - 6'b00_0000 001010b - 6'b00_1010
11-6 —	Reserved
5-0 CM1	C(-1) preset value. Default values for preset a= 000000b - 6'b00_0000 000001b - 6'b00_0000 000010b - 6'b00_0000 000011b - 6'b00_0000 000100b - 6'b00_0000 000101b - 6'b00_0101 000110b - 6'b00_0110 000111b - 6'b00_0101 001000b - 6'b00_0110 001001b - 6'b00_1000 001010b - 6'b00_0110

20.6.19 PCIe Equalization Preset 8 Register (PEXEQP8CR)

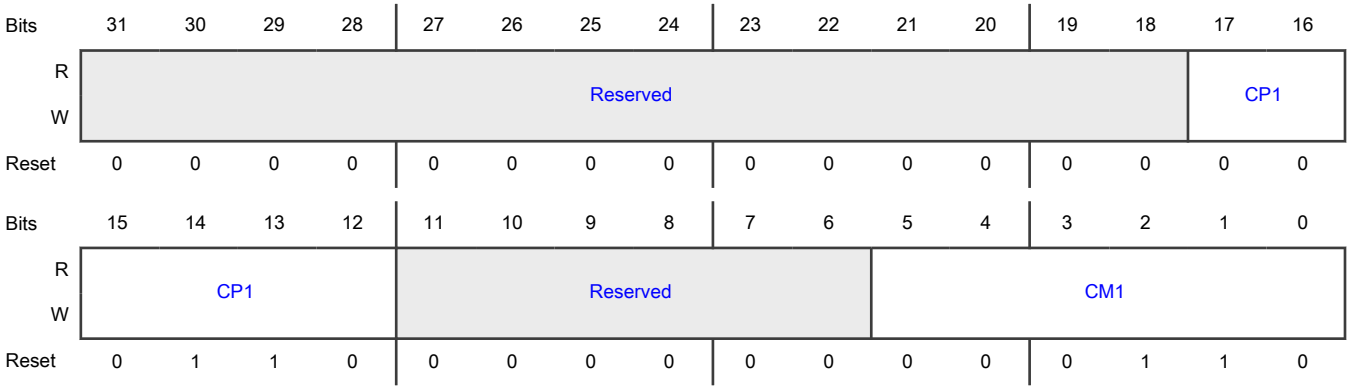
Offset

Register	Offset
PEXEQP8CR	324h

Function

PEXEQPyCR (y=0-10) contains the PCIe gen3 Tx equalization preset values. Note that C(0) is automatically calculated from C(-1), C(+1), and PEXEQCR[FS].

Diagram



Fields

Field	Function
31-18 —	Reserved
17-12 CP1	C(+1) preset value. Default values for preset a= 000000b - 6'b00_1100 000001b - 6'b00_1000 000010b - 6'b00_1010 000011b - 6'b00_0110 000100b - 6'b00_0000 000101b - 6'b00_0000 000110b - 6'b00_0000 000111b - 6'b00_1001 001000b - 6'b00_0110

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001001b - 6'b00_0000 001010b - 6'b00_1010
11-6 —	Reserved
5-0 CM1	C(-1) preset value. Default values for preset a= 000000b - 6'b00_0000 000001b - 6'b00_0000 000010b - 6'b00_0000 000011b - 6'b00_0000 000100b - 6'b00_0000 000101b - 6'b00_0101 000110b - 6'b00_0110 000111b - 6'b00_0101 001000b - 6'b00_0110 001001b - 6'b00_1000 001010b - 6'b00_0110

20.6.20 PCIe Equalization Preset 9 Register (PEXEQP9CR)

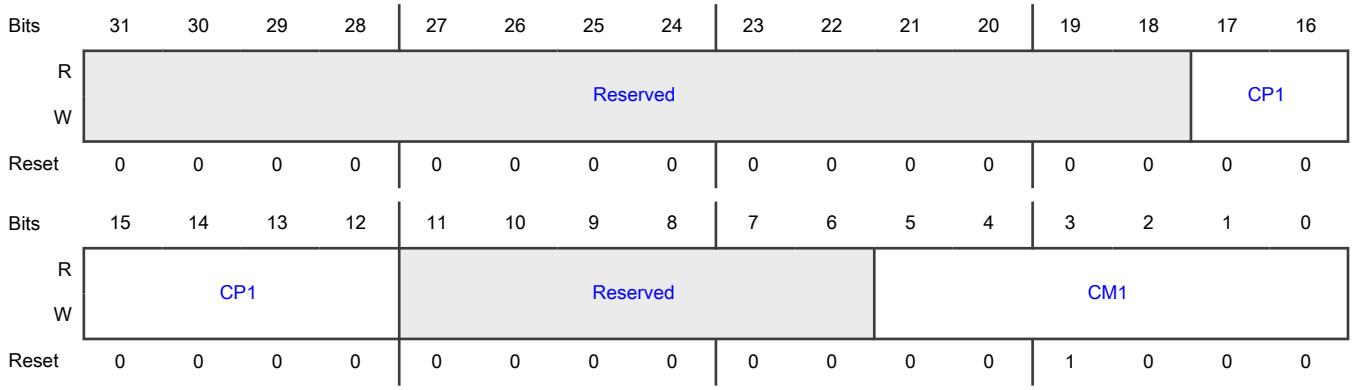
Offset

Register	Offset
PEXEQP9CR	328h

Function

PEXEQPyCR (y=0-10) contains the PCIe gen3 Tx equalization preset values. Note that C(0) is automatically calculated from C(-1), C(+1), and PEXEQCR[FS].

Diagram



Fields

Field	Function
31-18 —	Reserved
17-12 CP1	C(+1) preset value. Default values for preset a= 000000b - 6'b00_1100 000001b - 6'b00_1000 000010b - 6'b00_1010 000011b - 6'b00_0110 000100b - 6'b00_0000 000101b - 6'b00_0000 000110b - 6'b00_0000 000111b - 6'b00_1001 001000b - 6'b00_0110 001001b - 6'b00_0000 001010b - 6'b00_1010
11-6 —	Reserved
5-0 CM1	C(-1) preset value. Default values for preset a= 000000b - 6'b00_0000 000001b - 6'b00_0000 000010b - 6'b00_0000

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000011b - 6'b00_0000 000100b - 6'b00_0000 000101b - 6'b00_0101 000110b - 6'b00_0110 000111b - 6'b00_0101 001000b - 6'b00_0110 001001b - 6'b00_1000 001010b - 6'b00_0110

20.6.21 PCIe Equalization Preset 10 Register (PEXEQP10CR)

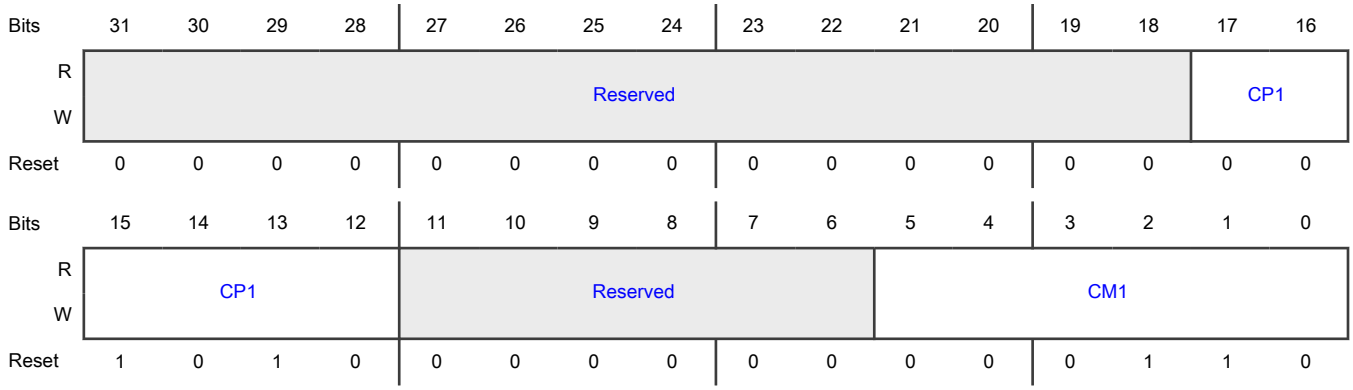
Offset

Register	Offset
PEXEQP10CR	32Ch

Function

PEXEQPyCR (y=0-10) contains the PCIe gen3 Tx equalization preset values. Note that C(0) is automatically calculated from C(-1), C(+1), and PEXEQCR[FS].

Diagram



Fields

Field	Function
31-18 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
17-12 CP1	C(+1) preset value. Default values for preset a= 000000b - 6'b00_1100 000001b - 6'b00_1000 000010b - 6'b00_1010 000011b - 6'b00_0110 000100b - 6'b00_0000 000101b - 6'b00_0000 000110b - 6'b00_0000 000111b - 6'b00_1001 001000b - 6'b00_0110 001001b - 6'b00_0000 001010b - 6'b00_1010
11-6 —	Reserved
5-0 CM1	C(-1) preset value. Default values for preset a= 000000b - 6'b00_0000 000001b - 6'b00_0000 000010b - 6'b00_0000 000011b - 6'b00_0000 000100b - 6'b00_0000 000101b - 6'b00_0101 000110b - 6'b00_0110 000111b - 6'b00_0101 001000b - 6'b00_0110 001001b - 6'b00_1000 001010b - 6'b00_0110

20.6.22 General Control Register 0 - Lane A (LNAGCR0)

Offset

Register	Offset
LNAGCR0	800h

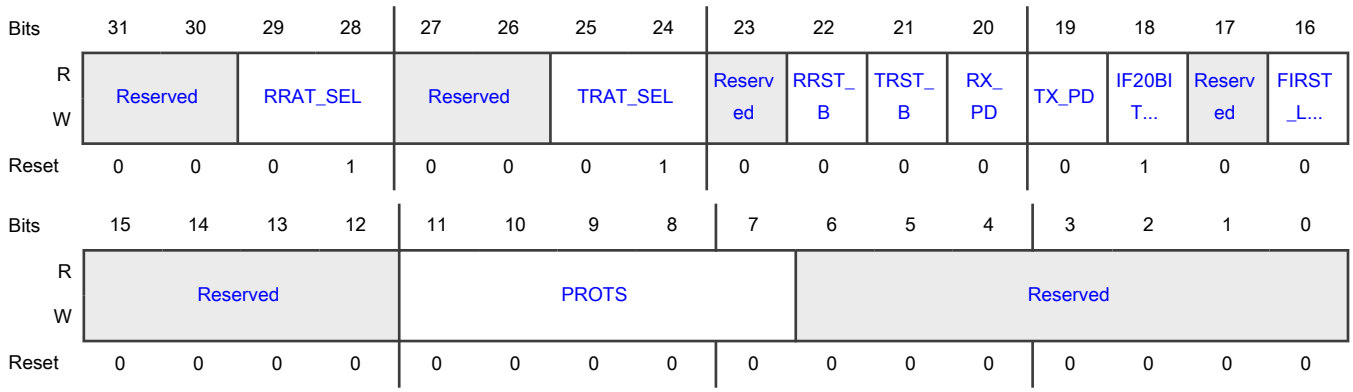
Function

LNmGCR0 contains functional control bits for SerDes lane *m*.

Special consideration must be taken when writing this register while PLLnRSTCTL[RST_DONE]=0, or if PCIe is running on this lane. See RPLL_LES, TPLL_LES, RRST_B and TRST_B fields for details.

See , for details on the sequence required to change settings.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-28 RRAT_SEL	Receiver speed selection for lane a, relative to the corresponding PLLnCR0[FRATE_SEL], n as selected by LNaGCR0[RPLL_LES]: Default value set by reset configuration. Required setting per protocol: PCIe N/A (receiver speed selection automatically selected by protocol) This register field is read-only when running in PCIe mode . Note: must be set the same as TRAT_SEL for normal function. 00b - Same as FRATE_SEL 01b - FRATE_SEL/2

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - FRATE_SEL/4 11b - FRATE_SEL*2
27-26 —	Reserved
25-24 TRAT_SEL	Transmitter speed selection for lane a, relative to the corresponding PLLnCR0[FRATE_SEL], n as selected by LNaGCR0[TPLL_LES]: Default value set by reset configuration. Required setting per protocol: see RRAT_SEL This register field is read-only when running in PCIe mode . Note: must be set the same as RRAT_SEL for normal function. 00b - Same as FRATE_SEL 01b - FRATE_SEL/2 10b - FRATE_SEL/4 11b - FRATE_SEL*2
23 —	Reserved
22 RRST_B	Resets receiver for lane a Recommended setting per protocol: 1, unless PLLnRSTCTL[RST_DONE]=0, then 0. This register value is read-only during SerDes reset (POR or PLLnRSTCTL[RSTREQ]). This register field is read-only when running in PCIe mode . Used in lane reset and reconfiguring procedures. See Lane Reset and Reconfiguration, for details. 0b - Reset 1b - Application Mode
21 TRST_B	Resets transmitter for lane a Recommended setting per protocol: 1 This register value is read-only during SerDes reset (POR or PLLnRSTCTL[RSTREQ]). This register field is read-only when running in PCIe mode . Used in lane reset and reconfiguring procedures. See Lane Reset and Reconfiguration, for details. 0b - Reset 1b - Application Mode
20 RX_PD	Lane powerdown for receiver on lane a Default value set by reset configuration.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This register field read-only during SerDes reset (POR or PLLnRSTCTL[RSTREQ]). See Lane Enable After Powerdown for details on re-enabling powered down lanes. 0b - Lane receiver active 1b - Lane receiver powered down
19 TX_PD	Lane powerdown for transmitter on lane a Default value set by reset configuration. This register value is read-only during SerDes reset (POR or PLLnRSTCTL[RSTREQ]). Note: the user must not assert TX_PD for the master clock lane (TX_CLK_1STLANE=1) of a multi-lane link during normal function. See Lane Enable After Powerdown for details on e-enabling powered down lanes. 0b - Lane transmitter active 1b - Lane transmitter powered down
18 IF20BIT_EN	20-bit interface enable Default value set by reset configuration. Required value per protocol: PCIe 1 0b - 10-bit interface 1b - 20-bit interface
17 —	Reserved
16 FIRST_LANE	Indicates this lane is the first (lane 0) of a group of lanes Default/recommended value set by reset configuration 0b - Lane a is not lane 0 of the link 1b - Lane a is lane 0 of the link or unused lane
15-12 —	Reserved
11-7 PROTS	Lane Protocol Select All others reserved 00000b - PCI EXP
6-0 —	Reserved

20.6.23 General Control Register 1 - Lane A (LNAGCR1)

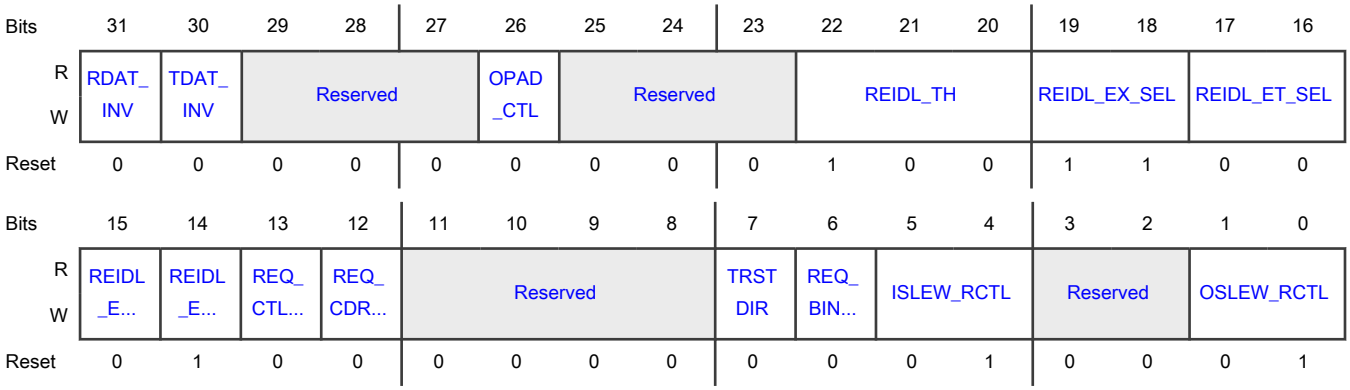
Offset

Register	Offset
LNAGCR1	804h

Function

LNmGCR1 contains functional control bits for SerDes lane a.

Diagram



Fields

Field	Function
31 RDAT_INV	Invert Rx data. Has the same effect as swapping SD_RX[m]_P and SD_RX[m]_N. Note: this field is read-only when this lane is operating as PCIe . 0b - Rx data is not inverted 1b - Rx data is inverted before it is decoded
30 TDAT_INV	Invert Tx data. Has the same effect as swapping SD_TX[m]_P and SD_TX[m]_N. 0b - Tx data is not inverted 1b - Tx data is inverted before it is transmitted
29-27 —	Reserved
26 OPAD_CTL	TX Output pad control signal for common mode Note: this field is read-only when this lane is operating as PCIe, . 0b - Transmitter Enabled 1b - Force Transmitter Output to Common Mode

Table continues on the next page...

Table continued from the previous page...

Field	Function
25-23 —	Reserved
22-20 REIDL_TH	Receiver electrical idle detection threshold control Recommended value per protocol: PCIe: 100 (2.5 Gbaud) Others: 000 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[REIDL_TH_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[REIDL_TH_1) .
19-18 REIDL_EX_SEL	Exit electrical idle filter select = {REIDL_EX_MSB,REIDL_EX_SEL} Recommended value per protocol: PCIe: 011 (2.5 Gbaud) Others: 000 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[REIDL_EX_SEL_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[REIDL_EX_SEL_1) .
17-16 REIDL_ET_SEL	Enter idle filter select = {REIDL_ET_MSB,REIDL_ET_SEL}: Recommended setting per protocol: PCIe: 111 (2.5 Gbaud) Others: 000 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[REIDL_ET_SEL_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[REIDL_ET_SEL_1) .
15 REIDL_EX_MSB	Exit idle filter select MSB. See REIDL_EX_SEL for settings. Note: PCIe 5 Gbaud setting taken from LNaSSCR0[REIDL_EX_MSB_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[REIDL_EX_MSB_1) .
14 REIDL_ET_MSB	Enter idle filter select MSB. See REIDL_ET_SEL for settings. Note: PCIe 5 Gbaud setting taken from LNaSSCR0[REIDL_ET_MSB_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[REIDL_ET_MSB_1) .
13 REQ_CTL_SNP	Initiate snapshot of RX Equalization Control Gaink2, Gaink3, and Offset Registers Recommended Setting per protocol: 0
12 REQ_CDR_SNP	Initiate snapshot of RX Clock/Data Recovery (CDR) Registers Recommended Setting per protocol: 0

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-8 —	Reserved
7 TRSTDIR	Multi-lane protocol Tx clock synchronization control Recommended setting per protocol: PCIe: 0 Others: 0
6 REQ_BIN_SNP	Initiate snapshot of RX Equalization Control Binning Registers Note: this field is read-only when this lane is operating as PCIe
5-4 ISLEW_RCTL	Slew control for Quadrature Generator Default value set by RCS configuration. Recommended setting per protocol: PCIe: 01 (2.5 Gbaud) Note: PCIe 5 Gbaud setting taken from LNaSSCR0[ISLEW_RCTL_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[ISLEW_RCTL_1) .
3-2 —	Reserved
1-0 OSLEW_RCTL	Phase Interpolator Output clock edge rate control Recommended setting per protocol: PCIe: 01 (2.5 Gbaud) Note: PCIe 5 Gbaud setting taken from LNaSSCR0[OSLEW_RCTL_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[OSLEW_RCTL_1) .

20.6.24 Speed Switch Control Register 0 - Lane A (LNASSCR0)

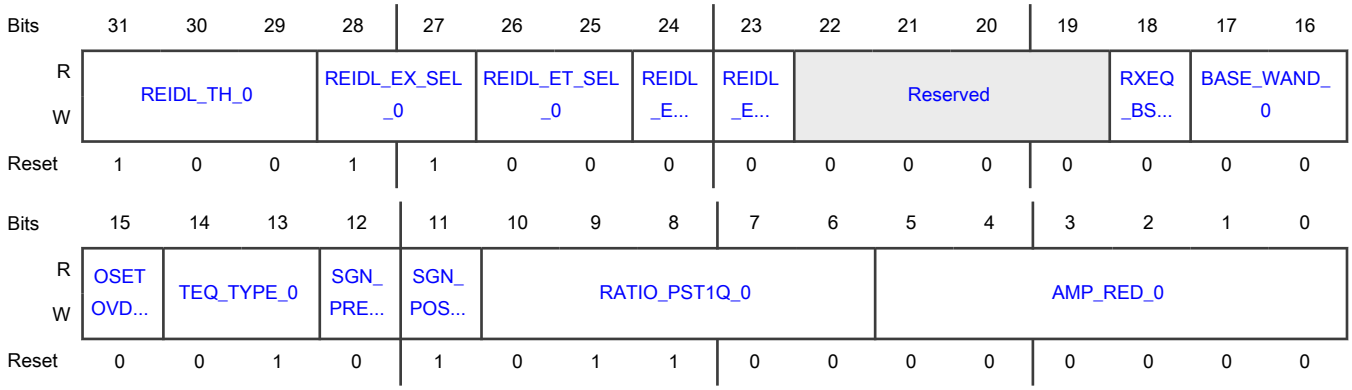
Offset

Register	Offset
LNASSCR0	80Ch

Function

LNmSSCR0 contains control bits for modifying the PCI Express 5 Gbaudbehavior of SerDes lane a.

Diagram



Fields

Field	Function
31-29 REIDL_TH_0	Receiver electrical idle detection threshold control Recommended settings per protocol: PCIe: 100 (5 Gbaud) Others: 000 For PCIe 2.5 Gbaud, see LNaGCR1[REIDL_TH]. For PCIe 8 Gbaud, see LNaSSCR1[REIDL_TH_1].
28-27 REIDL_EX_SEL_0	Exit electrical idle filter select = {REIDL_EX_MSB_0,REIDL_EX_SEL_0} for PCIe 5 Gbaud Recommended settings per protocol: PCIe: 011 (5 Gbaud) Others: 000 For PCIe 2.5 Gbaud, see LNaGCR1[REIDL_EX_SEL]. For PCIe 8 Gbaud, see LNaSSCR1[REIDL_EX_SEL_1].
26-25 REIDL_ET_SEL_0	Enter idle filter select = {REIDL_ET_MSB_0,REIDL_ET_SEL_0} for PCIe 5 Gbaud: Recommended settings per protocol: PCIe: 000 (5 Gbaud) Others: 000 For PCIe 2.5 Gbaud, see LNaGCR1[REIDL_ET_SEL]. For PCIe 8 Gbaud, see LNaSSCR1[REIDL_ET_SEL_1].
24 REIDL_EX_MSB_0	Exit idle filter select MSB. See REIDL_EX_SEL_0 for settings. For PCIe 2.5 Gbaud, see LNaGCR1[REIDL_EX_MSB]. For PCIe 8 Gbaud, see LNaSSCR1[REIDL_EX_MSB_1].
23	Enter idle filter select MSB. See REIDL_ET_SEL_0 for settings.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REIDL_ET_MSB_0	For PCIe 2.5 Gbaud, see LNaGCR1[REIDL_ET_MSB]. For PCIe 8 Gbaud, see LNaSSCR1[REIDL_ET_MSB_1].
22-19 —	Reserved
18 RXEQ_BST_0	Rx Equalization Boost Recommended value per protocol: 0 For PCIe 2.5 Gbaud, see LNaRECR0[RXEQ_BST]. For PCIe 8 Gbaud, see LNaSSCR1[RXEQ_BST_1].
17-16 BASE_WAND_0	Baseline Wander Control Select Recommended settings per protocol: 00 For PCIe 2.5 Gbaud, see LNaRECR0[BASE_WAND]. For PCIe 8 Gbaud, see LNaSSCR1[BASE_WAND_1]. 00b - off (8b10b data) 01b - default BinBLW threshold 10b - alternate BinBLW sign 11b - Use RX Eq offset as GainBLW override
15 OSETOVD6_0	Binary Decode of Lane Adaptive Equalization offset intitialization or override value. [6] = 1: Double Imposed Offset Recommended setting per protocol: PCIe: 0 (5 Gbaud) Others: 0 For PCIe 2.5 Gbaud, see LNaRECR0[OSETOVD]. For PCIe 8 Gbaud, see LNaSSCR1[OSETOVD6_1].
14-13 TEQ_TYPE_0	Lane transmit equalization. Recommended setting per protocol: PCIe: 01 (5 Gbaud) Others: 00 For PCIe 2.5 Gbaud, see LNaTECR0[TEQ_TYPE]. For PCIe 8 Gbaud, see LNaSSCR1[TEQ_TYPE_1]. 00b - No TX Equalization 01b - 2 Levels of TX Equalization (+1 post cursor)

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - 3 Levels of TX Equalization (+1 pre-cursor and +1 post-cursor) 11b - Reserved
12 SGN_PREQ_0	Precursor sign Recommended setting per protocol: PCIe: 0 (5 Gbaud) Others: 0 For PCIe 2.5 Gbaud, see LNaTECR0[SGN_PREQ]. For PCIe 8 Gbaud, see LNaSSCR1[SGN_PREQ_1]. 0b - Negative Sign (close eye) 1b - Positive Sign (open eye)
11 SGN_POST1Q_0	Post1q sign Recommended setting per protocol: PCIe: 1 (5 Gbaud) Others: 0 For PCIe 2.5 Gbaud, see LNaTECR0[SGN_POST1Q]. For PCIe 8 Gbaud, see LNaSSCR1[SGN_POST1Q_1] 0b - Negative Sign (close eye) 1b - Positive Sign (open eye)
10-6 RATIO_PST1Q_0	Ratio of full swing transition bit to first post-cursor. Recommended setting per protocol: PCIe: 0_1100 (5 Gbaud) Others: 0_0000 For PCIe 2.5 Gbaud, see LNaTECR0[RATIO_PST1Q]. For PCIe 8 Gbaud, see LNaSSCR1[RATIO_PST1Q_1]. Note: this field is read-only when this lane is operating as PCIe
5-0 AMP_RED_0	Overall TX Amplitude Reduction Recommended setting per protocol: PCIe: 00_0000 (5 Gbaud) Others: 00_0000 For PCIe 2.5 Gbaud, see LNaTECR0[AMP_RED]. For PCIe 8 Gbaud, see LNaSSCR1[AMP_RED_1]. Note: this field is read-only when this lane is operating as PCIe

20.6.25 Receive Equalization Control Register 0 - Lane A (LNARECR0)

Offset

Register	Offset
LNARECR0	810h

Function

LNmRECR0 contains functional control bits for SerDes lane a

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			RXEQ_BST	GK2OVD				Reserved				GK3OVD			
Reset	0	0	0	0	0	0	0	0	0	0	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	GK2OVD_...	GK3OVD_...	OSETOVD...	Reserved	BASE_WAND		Reserved		Reserved	OSETOVD						
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Fields

Field	Function
31-29 —	Reserved
28 RXEQ_BST	Rx Equalization Boost Recommended value per protocol: 0 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[RXEQ_BST_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[RXEQ_BST_1) .
27-24 GK2OVD	Binary decode of lane Adaptive Equalization gaink2 initialization or override value. Recommended settings per protocol: PCIe: 0000 (2.5 Gbaud) Note: PCIe 5 Gbaud setting taken from LNaSSCR0[GK2OVD_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[GK2OVD_1) .
23-20 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
19-16 GK3OVD	Binary decode of lane Adaptive Equalization gaink3 intitialization or override value. Recommended settings per protocol: PCIe: 0000 (2.5 Gbaud) Note: PCIe 5 Gbaud setting taken from LNaSSCR0[GK3OVD_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[GK3OVD_1) .
15 GK2OVD_EN	Controls source of rx equalization "gaink2" setting. Recommended settings per protocol: Others: 0 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[GK2OVD_EN_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[GK2OVD_EN_1) . 0b - Use rxeq adaption derived gaink2 1b - Fix gaink2 according to GK2OVD
14 GK3OVD_EN	Controls source of rx equalization "gaink3" setting. Recommended settings per protocol: Others: 0 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[GK3OVD_EN_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[GK3OVD_EN_1) . 0 Use rxeq adaption derived gaink3 1 Fix gaink3 according to GK3OVD
13 OSETOVD_EN	Controls source of rx equalization "offset" setting. Recommended setting per protocol: 0 0b - On release of srds_reset_b, initial rx eq offset is LNaRECR0[OSETOVD] and rx eq loop will begin adjustment at this value 1b - rx eq offset is fixed at LNaRECR0[OSETOVD]
12 —	Reserved
11-10 BASE_WAND	Baseline Wander Control Select Recommended setting per protocol: Others: 00 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[BASE_WAND_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[BASE_WAND_1) .

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - off (8b10b data) 01b - default BinBLW threshold 10b - alternate BinBLW sign 11b - Use OSETOVD[4:0] as GainBLW override
9-8 —	Reserved
7 —	Reserved
6-0 OSETOVD	Binary Decode of Lane Adaptive Equalization offset intitialization or override value. [6] 1: Double Imposed Offset [5:0] b00 0000: + Maximum Imposed Offset [5:0] b01 1111: No imposed offset [5:0] b11 1111: - Maximum Imposed Offset Note: If BASE_WAND= 11, then use OSETOVD[4:0] as GainBLW Override Recommended setting per protocol: Others: 001_1111 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[OSETOVD_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[OSETOVD_1)

20.6.26 Receive Equalization Control Register 1- Lane A (LNARECR1)

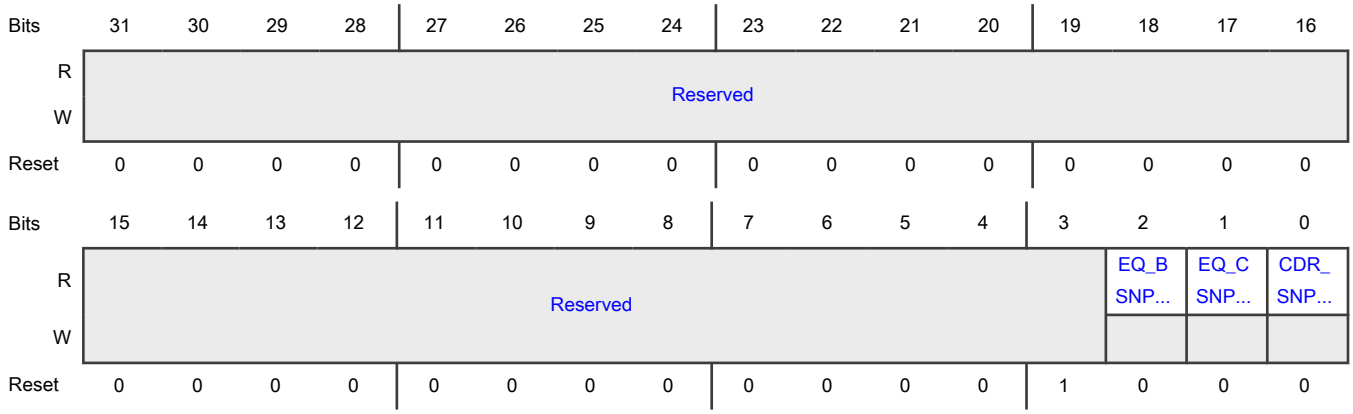
Offset

Register	Offset
LNARECR1	814h

Function

LNmRECR1 contains control and status bits for receiver equalization on lane a.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 EQ_BSNP_DN	Snapshot of RX EQ Bin Complete
1 EQ_CSNP_DN	Snapshot of RX EQ Ctrl Complete
0 CDR_SNP_DN	Snapshot of CDR Loop Complete

20.6.27 Transmit Equalization Control Register 0 - Lane A (LNATECR0)

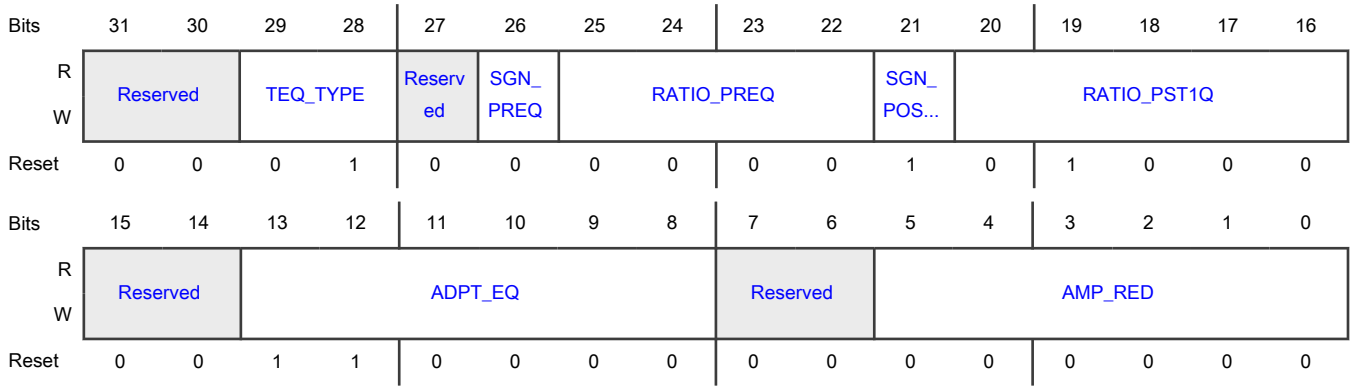
Offset

Register	Offset
LNATECR0	818h

Function

LNmTECR0 contains Tx equalization control bits for SerDes lane a.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-28 TEQ_TYPE	Selects amount/type of Transmit Equalization Recommended settings per protocol: Others: 01 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[TEQ_TYPE_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[TEQ_TYPE_1) 00b - No TX Equalization 01b - 2 Levels of TX Equalization (+1 post cursor) 10b - 3 Levels of TX Equalization (+1 pre-cursor and +1 post-cursor) 11b - Reserved
27 —	Reserved
26 SGN_PREQ	Precursor sign Recommended settings per protocol: Others: 0 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[SGN_PREQ_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[SGN_PREQ_1) 0b - Negative Sign(close eye) 1b - Positive Sign (open eye)
25-22	Ratio of full swing transition bit to pre-cursor

Table continues on the next page...

Table continued from the previous page...

Field	Function
RATIO_PREQ	Recommended settings per protocol: Others: 0000 Note: this field is read-only when this lane is operating as PCIe
21 SGN_POST1Q	Post q Sign Recommended settings per protocol: Others: 1 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[SGN_POST1Q_0) Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[SGN_POST1Q_1) . 0b - Negative Sign (close eye) 1b - Positive Sign (open eye)
20-16 RATIO_PST1Q	Ratio of full swing transition bit to first post-cursor. Recommended settings per protocol: PCIe: 0_1000 (2.5 Gbaud) Others: 0_0110 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[RATIO_PST1Q_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[RATIO_PST1Q_1) . Note: this field is read-only when this lane is operating as PCIe
15-14 —	Reserved
13-8 ADPT_EQ	Transmitter Adjustments for 8G/10G Recommended settings per protocol: Others: 11_0000 Note: this field is read-only when this lane is operating as PCIe
7-6 —	Reserved
5-0 AMP_RED	Overall TX Amplitude Reduction Recommended settings per protocol: Others: 00_0000 Note: PCIe 5 Gbaud setting taken from LNaSSCR0[AMP_RED_0) . Note: and PCIe 8 Gbaud setting taken from LNaSSCR1[AMP_RED_1)

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Note: this field is read-only when this lane is operating as PCIe

20.6.28 Speed Switch Control Register 1- Lane 0 (LNASSCR1)

Offset

Register	Offset
LNASSCR1	81Ch

Function

LNaSSCR1 contains control bits for modifying PCI Express 8 Gbaud behavior of SerDes lane a.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	REIDL_TH_1			REIDL_EX_SEL_1		REIDL_ET_SEL_1		REIDL_E...	REIDL_E...	Reserved				RXEQ_BS...	BASE_WAND_1	
Reset	0	0	0	0	0	0	0	0	0	0	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	OSET OVD...	TEQ_TYPE_1		SGN_PRE...	SGN_POS...	RATIO_PST1Q_1			AMP_RED_1							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 REIDL_TH_1	Receiver electrical idle detection threshold control Recommended settings per protocol: PCIe: 100 (8 Gbaud) Others: 000 For PCIe 2.5 Gbaud, see LNaGCR1[REIDL_TH]. For PCIe 5 Gbaud, see LNaSSCR0[REIDL_TH_0].
28-27 REIDL_EX_SEL_1	Exit electrical idle filter select = REIDL_EX_MSB_1||REIDL_EX_SEL_1 Recommended settings per protocol: PCIe: 011 (8 Gbaud)

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Others: 000 For PCIe 2.5 Gbaud, see LNaGCR1[REIDL_EX_SEL]. For PCIe 5 Gbaud, see LNaSSCR0[REIDL_EX_SEL_0].
26-25 REIDL_ET_SEL_1	Enter idle filter select = REIDL_ET_MSB_1||REIDL_ET_SEL_1: Recommended setting per protocol: PCIe: 000 (8 Gbaud) Others: 000 For PCIe 2.5 Gbaud, see LNaGCR1[REIDL_ET_SEL]. For PCIe 5 Gbaud, see LNaSSCR0[REIDL_ET_SEL_0].
24 REIDL_EX_MSB_1	Exit idle filter select MSB. See REIDL_EX_SEL_1 for settings. For PCIe 2.5 Gbaud, see LNaGCR1[REIDL_EX_MSB]. For PCIe 5 Gbaud, see LNaSSCR0[REIDL_EX_MSB_0].
23 REIDL_ET_MSB_1	Enter idle filter select MSB. See REIDL_ET_SEL_1 for settings. For PCIe 2.5 Gbaud, see LNaGCR1[REIDL_ET_MSB]. For PCIe 5 Gbaud, see LNaSSCR0[REIDL_ET_MSB_0].
22-19 —	Reserved
18 RXEQ_BST_1	Rx Equalization Boost Recommended value per protocol: 0 For PCIe 2.5 Gbaud, see LNaRECR0[RXEQ_BST]. For PCIe 5 Gbaud settings, see LNaSSCR0[RXEQ_BST_0].
17-16 BASE_WAND_1	Baseline Wander Control Select Recommended setting per protocol: PCIe: 01 (8 Gbaud) Others: 00 For PCIe 2.5 Gbaud settings, see LNaRECR0[BASE_WAND]. For PCIe 5 Gbaud settings, see LNaSSCR0[BASE_WAND_0]. 00b - off (8b10b data) 01b - default BinBLW threshold 10b - alternate BinBLW sign 11b - Use OSETOVD[4:0] as GainBLW override

Table continues on the next page...

Table continued from the previous page...

Field	Function
15 OSETOVD6_1	Binary Decode of Lane Adaptive Equalization offset intitialization or override value. [6] = 1: Double Imposed Offset Recommended setting per protocol: PCIe: 1 (8 Gbaud) Others: 0 For PCIe 2.5 Gbaud, see LNaRECR0[OSETOVD]. For PCIe 5 Gbaud, see LNaSSCR0[OSETOVD6_0].
14-13 TEQ_TYPE_1	Lane transmit equalization. Recommended setting per protocol: PCIe: 10 (8 Gbaud) Others: 00 For PCIe 2.5 Gbaud, see LNaTECR0[TEQ_TYPE]. For PCIe 5 Gbaud, see LNaSSCR0[TEQ_TYPE_0]. 00b - No TX Equalization 01b - 2 Levels of TX Equalization (+1 post cursor) 10b - 3 Levels of TX Equalization (+1 pre-cursor and +1 post-cursor) 11b - Reserved
12 SGN_PREQ_1	Precursor sign Recommended setting per protocol: PCIe: 1 (8 Gbaud) Others: 0 For PCIe 2.5 Gbaud, see LNaTECR0[SGN_PREQ]. For PCIe 5 Gbaud, see LNaSSCR0[SGN_PREQ_0]. 0b - Negative Sign (close eye) 1b - Positive Sign (open eye)
11 SGN_POST1Q_1	Post1q sign Recommended setting per protocol: PCIe: 1 (8 Gbaud) Others: 0 For PCIe 2.5 Gbaud, see LNaTECR0[SGN_POST1Q]. For PCIe 5 Gbaud, see LNaSSCR0[SGN_POST1Q_0]. 0b - Negative Sign (close eye) 1b - Positive Sign (open eye)

Table continues on the next page...

Table continued from the previous page...

Field	Function
10-6 RATIO_PST1Q_1	Ratio of full swing transition bit to first post-cursor. Recommended setting per protocol: PCIe: 0_1100 (8 Gbaud) Others: 0_0000 For PCIe 2.5 Gbaud, see LNaTECR0[AMP_RED]. For PCIe 5 Gbaud, see LNaSSCR0[AMP_RED_0]. Note: this field is read-only when this lane is operating as PCIe
5-0 AMP_RED_1	Overall TX Amplitude Reduction Recommended setting per protocol: PCIe: 00_0000 (8 Gbaud) Others: 00_0000 For PCIe 2.5 Gbaud, see LNaTECR0[AMP_RED]. For PCIe 5 Gbaud, see LNaSSCR0[AMP_RED_0]. Note: this field is read-only when this lane is operating as PCIe

20.6.29 TTL Control Register 0 - Lane A (LNATTLCR0)

Offset

Register	Offset
LNATTLCR0	820h

Function

LNmTTLCR0 contains control bits for the Transition Tracking Loop (TTL) on SerDes lane a.

Diagram

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

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

Fields

Field	Function
31-30 —	Reserved
29-24 FLT_SEL	Selects the gain 'Kfr', 'Kph' and TTL Edge Counting Window Widths in the CDR Loop for the Lane. Recommended values per protocol: Others: 00_0000
23-0 —	Reserved

20.6.30 Test Control/Status Register 3 - Lane A (LNATCSR3)

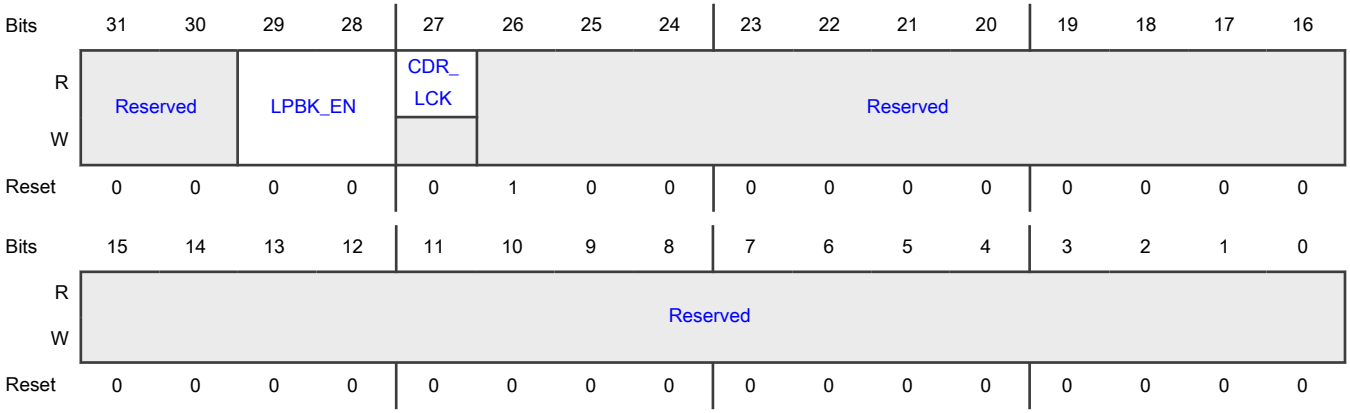
Offset

Register	Offset
LNATCSR3	83Ch

Function

LNmTCSR3 contains test control and status bits.

Diagram



Fields

Field	Function
31-30 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
29-28 LPBK_EN	Loopback data from TX to RX All others reserved Note: Loopback using PCI-Express requires Root Complex configuration. PCI Express End point loopback is not supported. 00b - Application Mode 01b - Loopback Mode
27 CDR_LCK	When asserted, CDR loop has acquired a valid Rx clock
26-0 —	Reserved

20.6.31 PEXA Protocol Control Register 0 (PEXACR0)

Offset

Register	Offset
PEXACR0	1000h

Function

The PCIe Protocol Control and Status Registers support the control and status bits related to the PCIe PCS layers and PHY control.

PEXnCR0 contains control bits used for the PEXn protocol.

Diagram

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

Fields

Field	Function
31-28 —	Reserved
27 RD_SW	Reduced Swing operation for 8 GT/s Note: full versus reduced swing for 2.5 and 5.0 GT/s is selected in the PCIe controller 0b - Full swing operation at 8 GT/s 1b - Reduced swing operation at 8 GT/s
26-0 —	Reserved

20.7 Functional Description

20.7.1 Combination Configuration

20.7.2 Error Handling

20.8 Initialization/Application Information

20.8.1 Initialization

All initialization necessary for the SerDes module to start operation is done automatically.

20.8.2 Unused Lanes

Unused lanes should be powered down to save power and avoid noise on adjacent lanes.

Power down the Rx portion of the lanes by setting LNmGCR0[RX_PD]=1 and LNmGCR0[RRST_B]=0.

Power down the Tx portion of the lanes by setting LNmGCR0[TX_PD]=1 and LNmGCR0[TRST_B]=0.

The Rx and Tx portions of the lanes may be independently powered down for uni-directional lanes or links.

20.8.3 Frequency Negotiation

PCI Express auto-negotiate frequency in hardware. For PCIe 2.5/5 Gbaud, that auto-negotiation involves overriding the by-n rate select function of the SerDes lane.

For PCIe 8 Gbaud, the auto-negotiation also involves a combination of switching PLL selects, and/or reconfiguring a PLL. There are two scenarios for PLL reconfiguration:

- PCIe starts on any PLL which runs at 5GHz, and the other PLL is off. PHY wrapper will switch to the other PLL as part of Gen 3 speed switch. In this case, both PLLs must be supplied with valid reference clocks.
- PCIe starts on any PLL which runs at 5GHz, and the other PLL is being used for other protocols. PHY wrapper will reconfigure the current 5GHz PLL as part of Gen 3 speed switch. PCIe stays on the same PLL in this case. This also applies to the scenario where one PCIe port runs on all lanes.

The current state of the by-n rate selects and PLL select per lane are reflected in SRDSxLNnGCR0, as defined in General Control Register 0 - Lane A (LNAGCR0). The current state of the PLL full-rate frequency select is reflected in SRDSxPLL, as defined in SerDes PLL1 Control Register 0 (PLL1CR0).

Other protocols either have a static frequency configuration, or change frequencies via a software reconfiguring sequence. See Lane Reset and Reconfiguration, and PLL Reset and Reconfiguration, for details on the software reconfiguring sequences.

20.8.4 Soft Reset and Reconfiguring Procedures

20.8.4.1 Lane Reset and Reconfiguration

To reconfigure a lane (change settings including clock divider or PLL select), perform the following sequence:

1. Put the lane(s) into reset by setting LNmGCR0[TRST_B]=0 and LNmGCR0[RRST_B]=0.
2. Wait at least 50 ns
3. Change the desired per-lane settings
4. Wait at least 120 ns
5. Take the lane(s) out of reset by setting LNmGCR0[TRST_B]=1 and LNmGCR0[RRST_B]=1

Note that if the lanes being reconfigured are grouped for multi-lane or synchronous mode, then the master source clock lane for the group must have TRST_B set to 1 after all the other lanes in the group. The master source clock lane of the group is indicated by LNmGCR0[1STLANE]=1. All lanes p<m (if LNmGCR1[TRSTDIR]=0) or p>m (if LNmGCR1[TRSTDIR]=1) of the master source clock lane until the next lane with LNmGCR0[1STLANE]=1, or the end of the SerDes, are grouped with the master source clock lane. All grouped lanes must have the same setting of TRSTDIR.

It is recommended to disable any controller connected to a lane being reconfigured prior to starting the reconfiguration sequence.

20.8.4.2 Lane Enable After Powerdown

To enable a previously powered down lane, set LNmGCR0[nX_PD]=0 (n=R or T), wait 15 us, then set LNmGCR0[nRST]=1.

Note that if a Tx lane m with LNmGCR0[1STLANE]=1 (master Tx clock lane) is powered down or reset, all Tx lanes p<m (if LNmGCR1[TRSTDIR]=0), or all Tx lanes p>m (if LNmGCR1[TRSTDIR]=1), cannot be used in normal function.

20.8.4.3 PLL Reset and Reconfiguration

To reconfigure a PLL, perform the following sequence:

1. Disable all lanes using the PLL to be reconfigured by setting PLLnRSTCTL[SDRST_B]=0
2. Wait at least 50 ns
3. Disable the PLL by setting PLLnRSTCTL[SDEN]=0 and PLLnRSTCTL[PLLRST_B]=0
4. Wait at least 100 ns
5. Change the desired per-PLL settings (VCO frequency, refclk frequency, etc.).
6. Reset the PLL by setting PLLnRSTCTL[RSTREQ]=1
7. Set PLLnRSTCTL[SDEN]=1, PLLnRSTCTL[PLLRST_B]=1, and PLLnRSTCTL[SDRST_B]=1
    - Note this step is not to be combined with setting RSTREQ=1

Note: lane reconfiguration may also be performed while the PLL is disabled by following the first three steps of the lane reset sequence in Lane Reset and Reconfiguration after changing the per-PLL settings, before setting RSTREQ=1, and the last step of the lane reset sequence after the last step of the above PLL reset sequence.

It is recommended to disable any controller connected to a lane selecting the PLL being reconfigured prior to starting the reconfiguration sequence. In the case of PCI Express, the controllers must be disabled to prevent auto-negotiation to 8Gbaud, which can include autonomous PLL reset and reconfiguration.

20.8.5 Debug Mode Procedures

20.8.5.1 SerDes Loopback

The SerDes module supports several internal loopback modes.

Note: To run SerDes internal loopback in PCIe application mode, either the SerDes needs to be connected to itself externally via a connector, or PEXnCR0[BYPTRN] must be set to 1 to bypass the Tx detector of receiver and electrical idle analog functions of Lynx.

20.8.6 Quiesce Sequences for System Sleep

To quiece the SerDes module in preparation for System Sleep, the user must first quiesce all controllers and disable transmission/reception of packets.

Chapter 21
Serial Peripheral Interface

21.1 Introduction

The serial peripheral interface (SPI) module provides a synchronous serial bus for communication between a chip and an external peripheral device.

21.1.1 Block Diagram

The block diagram of this module is as follows:

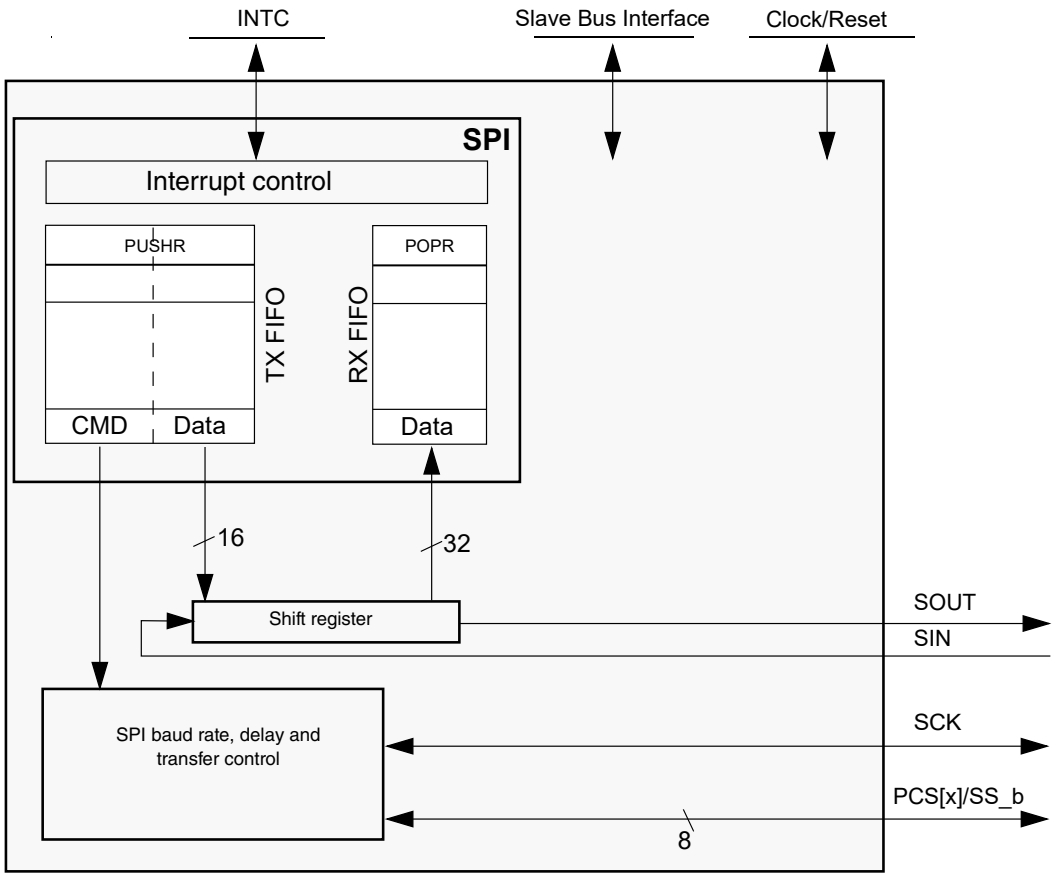


Figure 57. SPI Block Diagram

21.1.2 Features

The module supports the following features:

- Full-duplex, three-wire synchronous transfers
- Master mode
- Buffered transmit operation using the transmit first in first out (TX FIFO) with depth of 4 entries
- Support for 8/16-bit accesses to the PUSH TX FIFO Register Data Field
- Buffered receive operation using the receive FIFO (RX FIFO) with depth of 4 entries
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues

- Visibility into TX and RX FIFOs for ease of debugging
- Programmable transfer attributes on a per-frame basis:
  - 2 transfer attribute registers
  - 2 extended transfer attribute registers
  - Serial clock (SCK) with programmable polarity and phase
  - Various programmable delays
  - Programmable serial frame size: 4 to 32
    - SPI frames longer than 32 bits can be supported using the continuous selection format.
  - Continuously held chip select capability
  - Parity control
- 4 peripheral chip selects (PCSes), expandable to 2 ^ (4 – 1) with external demultiplexer
- Deglitching support for up to 4 peripheral chip selects (PCSes) with external demultiplexer
- Interrupt conditions:
  - End of Queue reached (EOQF)
  - TX FIFO is not full (TFFF)
  - CMD FIFO is not full (CMDFFF)
  - Transfer of current frame complete (TCF)
  - Transfers due from current command frame complete (CMDTCF)
  - Attempt to transmit with an empty Transmit FIFO (TFUF)
  - RX FIFO is not empty (RFDF)
  - Frame received while Receive FIFO is full (RFOF)
  - SPI Parity Error (SPEF)
  - Data present in TX FIFO while CMD FIFO is empty (TFIWF)
- Global interrupt request line
- Power-saving architectural features:
  - Support for Stop mode

21.1.3 Interface configurations

21.1.3.1 SPI configuration

The Serial Peripheral Interface (SPI) configuration allows the module to send and receive serial data. This configuration allows the module to operate as a basic SPI block with internal FIFOs supporting external queue operation. Transmitted data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the Receive FIFO and write transmit data to the Transmit FIFO.

For queued operations, the SPI queues can reside in system RAM, external to the module. Data transfers between the queues and the module FIFOs are accomplished by a DMA controller or host CPU. The following figure shows a system example with DMA, SPI, and external queues in system RAM.

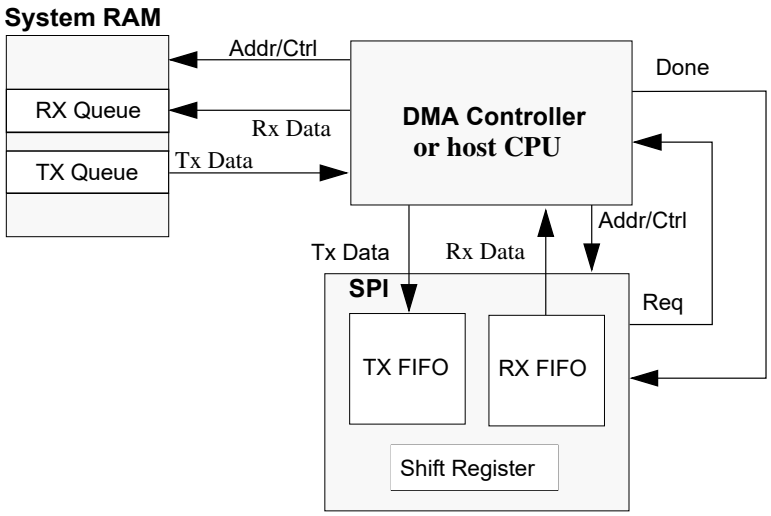


Figure 58. SPI with queues and DMA

21.1.4 Modes of operation

The module supports the following modes of operation that can be divided into two categories:

- Module-specific modes:
  - Master mode
  - Module Disable mode
- Chip-specific modes:
  - External Stop mode
  - Debug mode

The module enters module-specific modes when the host writes to a module register. The signals external to the module control the chip-specific modes. The chip may enter these modes in parallel to the module-specific modes.

21.1.4.1 Master Mode

Master mode allows the module to initiate and control serial communication. In this mode, these signals are controlled by the module and configured as outputs:

- SCK
- SOUT
- PCS[*x*]

21.1.4.2 Module Disable Mode

The Module Disable mode can be used for chip power management. The clock to the non-memory mapped logic in the module can be stopped while in the Module Disable mode.

21.1.4.3 External Stop Mode

External Stop mode is used for chip power management. The module supports the Peripheral Bus Stop mode mechanism. When a request is made to enter External Stop mode, it acknowledges the request and completes the transfer that is in progress. When the module reaches the frame boundary, it signals that the protocol clock to the module may be shut off.

21.1.4.4 Debug Mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls module behavior in the Debug mode:

- If the bit is set, the module stops all serial transfers, when the chip is in debug mode.
- If the bit is cleared, the chip debug mode has no effect on the module.

21.2 Module signal descriptions

This table describes the signals on the boundary of the module that may connect off chip (in alphabetical order).

Table 80. Module signal descriptions

Signal	Master mode	I/O
HT	Hardware Trigger	I
PCS0	Peripheral Chip Select 0 (O)	I/O
PCS[6:7]	Peripheral Chip Selects 6–7	O
SCK	Serial Clock (O)	I/O
SIN	Serial Data In	I
SOUT	Serial Data Out	O

21.2.1 HT—Hardware Trigger

Master mode: Hardware Trigger (I)—Honored only when the module is configured for the DSI or CSI configuration (MCR[DCONF]), hardware-trigger reception is enabled (DSICR0[TRRE] or TRIG[TRRE]), and the module is in the Running state (SR[TXRXS]). Receives hardware triggers that initiate data transfers. DSICR0[TPOL] specifies whether the module uses rising or falling edges of HT as hardware triggers.

21.2.2 PCS0—Peripheral Chip Select

Master mode: Peripheral Chip Select 0 (O)—Selects an SPI slave to receive data transmitted from the module.

NOTE

Do not tie the SPI slave select pin to ground. Otherwise, SPI cannot function properly.

21.2.3 PCS1–PCS3—Peripheral Chip Selects 1–3

Master mode: Peripheral Chip Selects 1–3 (O)—Select an SPI slave to receive data transmitted by the module.

21.2.4 SCK—Serial Clock

Master mode: Serial Clock (O)—Supplies a clock signal from the module to SPI slaves.

21.2.5 SIN—Serial Input

Master mode: Serial Input (I)—Receives serial data.

21.2.6 SOUT—Serial Output

Master mode: Serial Output (O)—Transmits serial data.

21.3 SPI register descriptions

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Any Write access to the POPR and RXFRn also results in a transfer error.

NOTE

While the module is in the running state, do not write to:

- CTAREn

21.3.1 SPI Memory map

SPI base address: 210_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Module Configuration Register (MCR)	32	RW	0000_4001h
8h	Transfer Count Register (TCR)	32	RW	0000_0000h
Ch - 10h	Clock and Transfer Attributes Register (In Master Mode) (CTAR0 - CTAR1)	32	RW	7800_0000h
2Ch	Status Register (SR)	32	W1C	0201_0000h
30h	DMA/Interrupt Request Select and Enable Register (RSER)	32	RW	0000_0000h
34h	PUSH TX FIFO Register In Master Mode (PUSHR)	32	RW	0000_0000h
38h	POP RX FIFO Register (POPR)	32	RO	0000_0000h
3Ch - 48h	Transmit FIFO Registers (TXFR0 - TXFR3)	32	RO	0000_0000h
7Ch - 88h	Receive FIFO Registers (RXFR0 - RXFR3)	32	RO	0000_0000h
11Ch - 120h	Clock and Transfer Attributes Register Extended (CTARE0 - CTARE1)	32	RW	0000_0001h
13Ch	Status Register Extended (SREX)	32	RO	0000_0000h

21.3.2 Module Configuration Register (MCR)

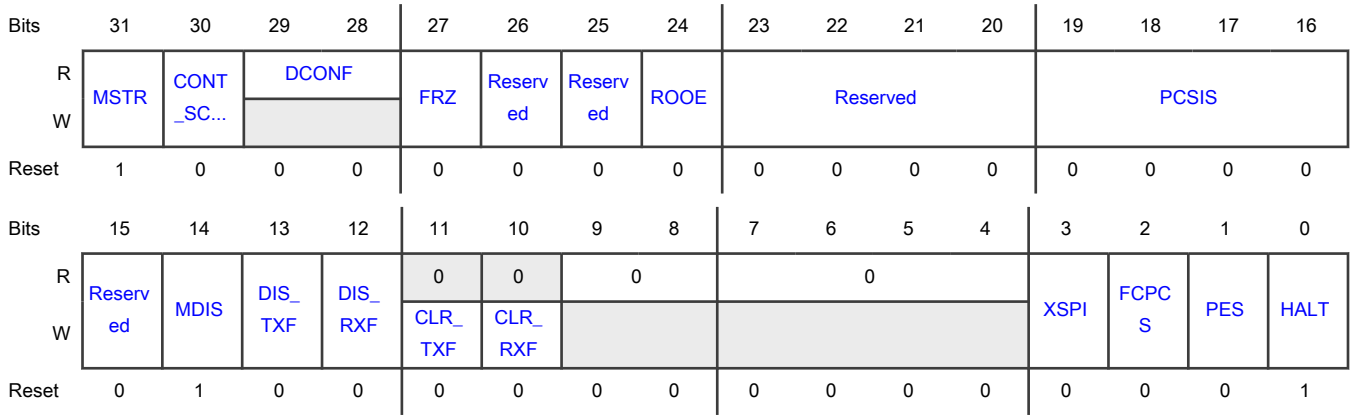
Offset

Register	Offset
MCR	0h

Function

Contains bits to configure various attributes associated with the module operations. The HALT and MDIS bits can be changed at any time, but the effect takes place only on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed, while the module is in the Running state.

Diagram



Fields

Field	Function
31 MSTR	Master/Slave Mode Select Enables either Master mode (if supported) or Slave mode (if supported) operation. 0b - Reserved 1b - Enables Master mode
30 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously. 0b - Continuous SCK disabled. 1b - Continuous SCK enabled.
29-28 DCONF	SPI Configuration. Selects among the different configurations of the module. 00b - SPI 01b - Reserved 10b - Reserved 11b - Reserved
27 FRZ	Freeze Enables transfers to be stopped on the next frame boundary when the device enters Debug mode. 0b - Do not halt serial transfers in Debug mode. 1b - Halt serial transfers in Debug mode.
26 —	Reserved
25	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
24 ROOE	Receive FIFO Overflow Overwrite Enable In the RX FIFO overflow condition, configures the module to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer, generating the overflow, is ignored or shifted into the shift register. 0b - Incoming data is ignored. 1b - Incoming data is shifted into the shift register.
23-20 —	Always write the reset value to this field.
19-16 PCSIS	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx. **NOTE** The effect of this bit only takes place when module is enabled. Ensure that this bit is configured correctly before enabling the SPI interface. 0000b - The inactive state of PCSx is low. 0001b - The inactive state of PCSx is high.
15 —	Reserved
14 MDIS	Module Disable Allows the clock to be stopped to the non-memory mapped logic in the module effectively putting it in a software-controlled power-saving state. The reset value of the MDIS bit is parameterized, with a default reset value of 1. 0b - Enables the module clocks. 1b - Allows external logic to disable the module clocks.
13 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the module operates as a simplified double-buffered SPI. This bit can be written only when the MDIS bit is cleared. 0b - TX FIFO is enabled. 1b - TX FIFO is disabled.
12 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the module operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0b - RX FIFO is enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - RX FIFO is disabled.
11 CLR_TXF	Clear TX FIFO Flushes the TX FIFO. Writing a 1 to CLR_TXF clears the TX FIFO Counter. The CLR_TXF bit is always read as zero. 0b - Do not clear the TX FIFO counter. 1b - Clear the TX FIFO counter.
10 CLR_RXF	CLR_RXF Flushes the RX FIFO. Writing a 1 to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. **NOTE** After every RX FIFO clear operation (MCR [CLR_RXF] = 0b1) following a RX FIFO overflow (SR [RFOF] = 0b1) scenario, immediately perform a single POP from the RX FIFO and discard the read data. The POP and discard operation should be completed before the reception of new incoming frame. 0b - Do not clear the RX FIFO counter. 1b - Clear the RX FIFO counter.
9-8 —	Reserved
7-4 —	Reserved
3 XSPI	Extended SPI Mode This bit enables usage of CTARE (Command and Transfer Attribute Register Extended) Registers. CTARE registers allow the user to send up to 32 bit SPI frames. Command Cycling is also enabled which allows the user to send multiple Data Frames using a single Command Frame. When MCR[DIS_TXF] is asserted, the Extended SPI Mode cannot be used to transmit SPI frames which are more than 16 bits in size. 0b - Normal SPI Mode. Frame size can be up to 16 bits. Command Cycling is not available in this mode. 1b - Extended SPI Mode. Up to 32 bit SPI Frames along with Command Cycling is Enabled.
2 FCPCS	Fast Continuous PCS Mode. This bit enables the masking of “After SCK (t_{ASC})” and “PCS to SCK (t_{CSC})” delays when operating in Continuous PCS mode. This masking is not available if Continuous SCK mode is enabled. The individual delay masks are selected via bits MASC and MCSC of the PUSHR register. The firmware should select appropriate masks when providing continuous frames via the PUSHR register. 0b - Normal or Slow Continuous PCS mode. Masking of delays is disabled. 1b - Fast Continuous PCS mode. Delays masked via control bits in PUSHR register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 PES	Parity Error Stop Controls SPI operation when a parity error is detected in a received SPI frame. 0b - SPI frame transmission continues. 1b - SPI frame transmission stops.
0 HALT	Halt The HALT bit starts and stops frame transfers. See Start and Stop of Module transfers 0b - Start transfers. 1b - Stop transfers.

21.3.3 Transfer Count Register (TCR)

Offset

Register	Offset
TCR	8h

Function

TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. Do not write the TCR when the module is in the Running state.

Diagram

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

Fields

Field	Function
31-16	SPI Transfer Counter

Table continues on the next page...

Table continued from the previous page...

Field	Function
SPI_TCNT	Counts the number of SPI transfers the module makes. The SPI_TCNT field increments every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The Transfer Counter wraps around; incrementing the counter past 65535 resets the counter to zero.
15-0 —	Reserved

21.3.4 Clock and Transfer Attributes Register (In Master Mode) (CTAR0 - CTAR1)

Offset

For a = 0 to 1:

Register	Offset
CTARa	Ch + (a × 4h)

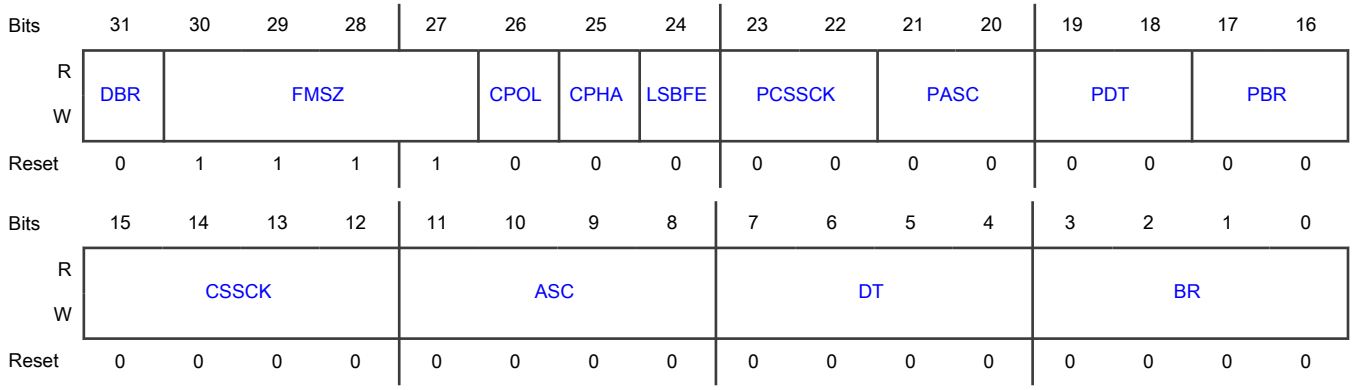
Function

CTAR registers are used to define different transfer attributes. Do not write to the CTAR registers while the module is in the Running state.

In Master mode, the CTAR registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays.

When the module is configured as a SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the CTAR registers is used.

Diagram



Fields

Field	Function
31	Double Baud Rate

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>DBR</td><td>Doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in master mode. It effectively halves the Baud Rate division ratio, supporting faster frequencies, and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in the following table. See the BR field description for details on how to compute the baud rate.

Table 81. SPI SCK Duty Cycle
<table><tr><th>DBR</th><th>CPHA</th><th>PBR</th><th>SCK Duty Cycle</th></tr><tr><td>0</td><td>any</td><td>any</td><td>50/50</td></tr><tr><td>1</td><td>0</td><td>00</td><td>50/50</td></tr><tr><td>1</td><td>0</td><td>01</td><td>33/66</td></tr><tr><td>1</td><td>0</td><td>10</td><td>40/60</td></tr><tr><td>1</td><td>0</td><td>11</td><td>43/57</td></tr><tr><td>1</td><td>1</td><td>00</td><td>50/50</td></tr><tr><td>1</td><td>1</td><td>01</td><td>66/33</td></tr><tr><td>1</td><td>1</td><td>10</td><td>60/40</td></tr><tr><td>1</td><td>1</td><td>11</td><td>57/43</td></tr></table>
0b - The baud rate is computed normally with a 50/50 duty cycle.

1b - The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.</td></tr>
<tr><td>30-27

FMSZ</td><td>Frame Size

The number of bits transferred per frame is equal to the FMSZ value plus 1. Regardless of the transmission mode, the minimum valid frame size value is 4.</td></tr>
<tr><td>26

CPOL</td><td>Clock Polarity

Selects the inactive state of the Serial Communications Clock (SCK). For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the module can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge.

NOTE
In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranted.

0b - The inactive state value of SCK is low.

1b - The inactive state value of SCK is high.</td></tr>
<tr><td>25

CPHA</td><td>Clock Phase

Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1.</td></tr>
</table>

Table continues on the next page...

Field	Function
	0b - Data is captured on the leading edge of SCK and changed on the following edge. 1b - Data is changed on the leading edge of SCK and captured on the following edge.
24 LSBFE	LSB First Specifies whether the LSB or MSB of the frame is transferred first. 0b - Data is transferred MSB first. 1b - Data is transferred LSB first.
23-22 PCSSCK	PCS to SCK Delay Prescaler Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description for information on how to compute the PCS to SCK Delay. Refer PCS to SCK Delay (t_{CSC}) for more details. 00b - PCS to SCK Prescaler value is 1. 01b - PCS to SCK Prescaler value is 3. 10b - PCS to SCK Prescaler value is 5. 11b - PCS to SCK Prescaler value is 7.
21-20 PASC	After SCK Delay Prescaler Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description for information on how to compute the After SCK Delay. Refer After SCK Delay (t_{ASC}) for more details. 00b - Delay after Transfer Prescaler value is 1. 01b - Delay after Transfer Prescaler value is 3. 10b - Delay after Transfer Prescaler value is 5. 11b - Delay after Transfer Prescaler value is 7.
19-18 PDT	Delay after Transfer Prescaler Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. See the DT field description for details on how to compute the Delay after Transfer. Refer Delay after Transfer (t_{DT}) for more details. 00b - Delay after Transfer Prescaler value is 1. 01b - Delay after Transfer Prescaler value is 3. 10b - Delay after Transfer Prescaler value is 5. 11b - Delay after Transfer Prescaler value is 7.
17-16 PBR	Baud Rate Prescaler Selects the prescaler value for the baud rate. This field is used only in master mode. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for details on how to compute the baud rate. 00b - Baud Rate Prescaler value is 2.

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>01b - Baud Rate Prescaler value is 3.
10b - Baud Rate Prescaler value is 5.
11b - Baud Rate Prescaler value is 7.</td></tr>
<tr><td>15-12
CSSCK</td><td>PCS to SCK Delay Scaler
Selects the scaler value for the PCS to SCK delay. This field is used only in master mode. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period, and it is computed according to the following equation:
$t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK$.
The following table lists the delay scaler values.
Table 82. Delay Scaler Encoding
<table>
<tr><th>Field Value</th><th>Delay Scaler Value</th></tr>
<tr><td>0000</td><td>2</td></tr>
<tr><td>0001</td><td>4</td></tr>
<tr><td>0010</td><td>8</td></tr>
<tr><td>0011</td><td>16</td></tr>
<tr><td>0100</td><td>32</td></tr>
<tr><td>0101</td><td>64</td></tr>
<tr><td>0110</td><td>128</td></tr>
<tr><td>0111</td><td>256</td></tr>
<tr><td>1000</td><td>512</td></tr>
<tr><td>1001</td><td>1024</td></tr>
<tr><td>1010</td><td>2048</td></tr>
<tr><td>1011</td><td>4096</td></tr>
<tr><td>1100</td><td>8192</td></tr>
<tr><td>1101</td><td>16384</td></tr>
<tr><td>1110</td><td>32768</td></tr>
<tr><td>1111</td><td>65536</td></tr>
</table>
Refer PCS to SCK Delay (t_{CSC}) for more details.</td></tr>
<tr><td>11-8
ASC</td><td>After SCK Delay Scaler
Selects the scaler value for the After SCK Delay. This field is used only in master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period, and it is computed according to the following equation:
$t_{ASC} = (1/f_P) \times PASC \times ASC$</td></tr>
</table>

Table continues on the next page...

Field	Function
	See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer After SCK Delay (t_{ASC}) for more details.
7-4 DT	Delay After Transfer Scaler Selects the Delay after Transfer Scaler. This field is used only in master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period, The Delay after Transfer is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{DT} = (1/f_P) \times PDT \times DT$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values.
3-0 BR	Baud Rate Scaler Selects the scaler value for the baud rate. This field is used only in master mode. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is computed according to the following equation: SCK baud rate = (f_P /PBR) x [(1+DBR)/BR] The following table lists the baud rate scaler values.

Table 83. Baud Rate Scaler

CTARn[BR]	Baud Rate Scaler Value
0000	2
0001	4
0010	6
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

21.3.5 Status Register (SR)

Offset

Register	Offset
SR	2Ch

Function

SR contains status and flag bits. The bits reflect the status of the module and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the SR by writing a 1 to them. Writing a 0 to a flag bit has no effect. This register may not be writable in Module Disable mode due to the use of power saving mechanisms.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TCF	TXRXS	0	EOQF	0	0	TFFF	BSYF	CMDTCF	Reserved	SPEF	0	RFOF	TFIWF	RFDF	CMDFFF
W	W1C			W1C			W1C		W1C		W1C		W1C	W1C		W1C
Reset	0	0	0	0	0	0	1	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	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 TCF	Transfer Complete Flag Indicates that all bits in a frame have been shifted out. TCF remains set until it is cleared by writing a 1 to it. 0b - Transfer not complete. 1b - Transfer complete.
30 TXRXS	TX and RX Status Reflects the run status of the module. 0b - Transmit and receive operations are disabled (The module is in Stopped state). 1b - Transmit and receive operations are enabled (The module is in Running state).
29 —	Reserved
28	End of Queue Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
EOQF	Indicates that the last entry in a queue has been transmitted when the module is in Master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. The EOQF bit remains set until cleared by writing a 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0b - EOQ is not set in the executing command. 1b - EOQ is set in the executing SPI command.
27 —	Reserved
26 —	Reserved
25 TFFF	Transmit FIFO Fill Flag Indicates whether there is an available location to be filled in the FIFO. Either DMA or an interrupt can be used to add another entry to the FIFO. Note that this bit is set if at least one location is free in the FIFO. The TFFF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller to the TX FIFO full request, when the TX FIFO is full. If FIFO is filled manually, and not by DMA, see Transmit FIFO Fill Interrupt or DMA Request **NOTE** The reset value of this bit is 0 when the module is disabled,(MCR[MDIS]=1). 0b - TX FIFO is full. 1b - TX FIFO is not full.
24 BSYF	Busy Flag. This bit is valid only when MCR[XSPI] is enabled. Indicates that the current Command Frame is being used for transmiting multiple data frames. This bit is not set for the last Data Frame of a Cyclic command Transfer or when CTARE[DTCP] = 1. Refer Command First In First Out (CMD FIFO) Buffering Mechanism for more details. 0b - No Cyclic Command Transfer in Progress. 1b - Cyclic Command Transfer is in progress. Current Data Frame is not the last data frame for on-going cyclic command transfer..
23 CMDTCF	Command Transfer Complete Flag. Indicates that the last Data frame for the current Cyclic Command has been transmitted. Hence this bit is set only for the last Data Frame of a Cyclic Command Transfer or when CTARE[DTCP] = 1. The bit remains set until it is cleared by writing a ‘1’ to it. 0b - Data Transfer by current Command not complete. 1b - Data Transfer by current Command is complete.
22	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
21 SPEF	SPI Parity Error Flag Indicates that a SPI frame with parity error had been received. The bit remains set until it is cleared by writing a 1 to it. 0b - No parity error. 1b - Parity error has occurred.
20 —	Reserved
19 RFOF	Receive FIFO Overflow Flag Indicates an overflow condition in the RX FIFO. The field is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until it is cleared by writing a 1 to it. 0b - No Rx FIFO overflow. 1b - Rx FIFO overflow has occurred.
18 TFIWF	Tranmit FIFO Invalid Write Flag Indicates Data Write on TX FIFO while CMD FIFO is empty. Without a Command, the Data entries present in TXFIFO are invalid. This bit remains set until it is cleared by writing a ‘1’ to it. 0b - No Invalid Data present in TX FIFO. 1b - Invalid Data present in TX FIFO since CMD FIFO is empty.
17 RFDF	Receive FIFO Drain Flag Provides a method for the module to request that entries be removed from the RX FIFO. Note that this bit is set if at least one location can be read from the FIFO. The RFDF bit can be cleared by acknowledgement from the DMA controller when the RX FIFO is empty. 0b - RX FIFO is empty. 1b - This bit auto-clears on every RXFR read performed.
16 CMDFFF	Command FIFO Fill Flag Indicates whether there is an available location to be filled in the FIFO. Either a DMA request or an interrupt indication can be used to add another entry to the FIFO. Note that this field is set if at least one location is free in the FIFO. The CMDFFF is cleared by writing a 1 to it or by acknowledgement from the DMA controller to the CMD FIFO full request. **NOTE** The reset value of this field is 0 when the module is disabled (MCR[MDIS] = 1). 0b - CMD FIFO is full. 1b - CMD FIFO is not full.
15-12	TX FIFO Counter

Table continues on the next page...

Table continued from the previous page...

Field	Function
TXCTR	Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the PUSHR is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
11-8 TXNXTPTR	Transmit Next Pointer Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register. **NOTE** When TX FIFO is cleared by setting MCR[CLR_TXF] to 1, this field does not update immediately to 0. Only when the next transfer starts, this field reflects the latest value TXNXTPTR =1.
7-4 RXCTR	RX FIFO Counter Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
3-0 POPNXTPTR	Pop Next Pointer Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

21.3.6 DMA/Interrupt Request Select and Enable Register (RSER)

Offset

Register	Offset
RSER	30h

Function

RSER controls DMA and interrupt requests. Do not write to the RSER while the module is in the Running state.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TCF_RE	CMDFF_...	Reserved	EOQF_RE	Reserved	Reserved	TFFF_RE	TFFF_DI...	CMDTCF_...	Reserved	SPEF_RE	Reserved	RFOF_RE	TFIWF_RE	RFDF_RE	RFDF_DI...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Fields

Field	Function
31 TCF_RE	Transmission Complete Request Enable Enables TCF flag in the SR to generate an interrupt request. 0b - TCF interrupt requests are disabled. 1b - TCF interrupt requests are enabled.
30 CMDFFF_RE	Command FIFO Fill Flag Request Enable. Enables the CMDFFF flag in the SR to generate a request. The CMDFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0b - CMDFFF interrupts or DMA requests are disabled. 1b - CMDFFF interrupts or DMA requests are enabled.
29 —	Always write the reset value to this field.
28 EOQF_RE	Finished Request Enable Enables the EOQF flag in the SR to generate an interrupt request. 0b - EOQF interrupt requests are disabled. 1b - EOQF interrupt requests are enabled.
27 —	Reserved
26 —	Always write the reset value to this field.
25 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0b - TFFF interrupts or DMA requests are disabled. 1b - TFFF interrupts or DMA requests are enabled.
24 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When SR[TFFF] and RSER[TFFF_RE] are set, this field selects between generating an interrupt request or a DMA request. 0b - TFFF flag generates interrupt requests. 1b - TFFF flag generates DMA requests.
23 CMDTCF_RE	Command Transmission Complete Request Enable. The CMDTCF_RE bit enables CMDTCF flag in the SR to generate an interrupt request.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - CMDTCF interrupt requests are disabled. 1b - CMDTCF interrupt requests are enabled.
22 —	Always write the reset value to this field.
21 SPEF_RE	SPI Parity Error Request Enable Enables the SPEF flag in the SR to generate an interrupt request. 0b - SPEF interrupt requests are disabled. 1b - SPEF interrupt requests are enabled.
20 —	Always write the reset value to this field.
19 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the SR to generate an interrupt request. 0b - RFOF interrupt requests are disabled. 1b - RFOF interrupt requests are enabled.
18 TFIWF_RE	Transmit FIFO Invalid Write Request Enable. Enables the TFIWF flag in the SR to generate an interrupt request. 0b - TFIWF interrupt requests are disabled. 1b - TFIWF interrupt requests are enabled.
17 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0b - RFDF interrupt or DMA requests are disabled. 1b - RFDF interrupt or DMA requests are enabled.
16 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the SR is set, and the RFDF_RE bit in the RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0b - Interrupt request. 1b - DMA request.
15 CMDFFF_DIRS	Command FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the CMDFFF flag bit in the SR is set, and the CMDFFF_RE bit in the RSER is set, the CMDFFF_DIRS bit selects between generating an interrupt request or a DMA request.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - CMDFFF flag generates interrupt requests. 1b - CMDFFF flag generates DMA requests.
14 —	Always write the reset value to this field.
13-0 —	Reserved

21.3.7 PUSH TX FIFO Register In Master Mode (PUSHR)

Offset

Register	Offset
PUSHR	34h

Function

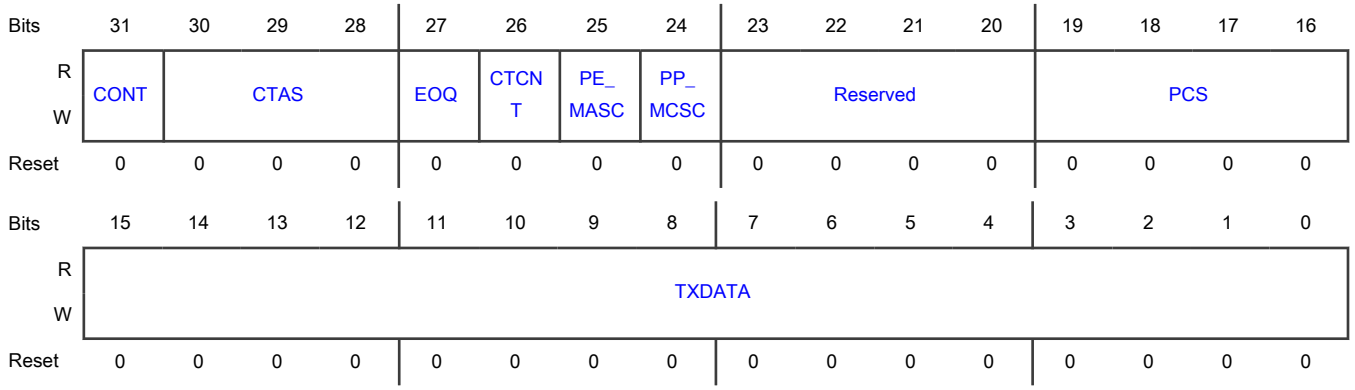
Specifies data to be transferred to the TX FIFO and CMD FIFO. User must write 16-bits data into TXDATA field. An 8- or 16-bit write access to the TXDATA field transfers 16 bits of data bus to the TX FIFO. A write access to the command fields transfers the 16 bits of command information to the CMD FIFO. In Master mode, the register transfers 16 bits of data to the TX FIFO and 16 bits of command information to the CMD FIFO.

If Extended SPI Mode is disabled (MCR[XSPI]), the TX FIFO and CMD FIFO must be filled simultaneously. In other words, you must perform write accesses to both the data and command fields for every PUSHR operation. With Extended SPI Mode disabled and both the TX FIFO and CMD FIFO are written to and read from simultaneously, they behave as a single 32 bit FIFO. When Extended SPI mode is enabled (MCR[XSPI]), the TX FIFO and CMD FIFO can be written to independently.

A read access of PUSHR returns the topmost TX FIFO and CMD FIFO entries concatenated.

When the module is disabled, writing to this register does not update the FIFO. Therefore, any reads performed while the module is disabled return the last PUSHR write performed while the module was still enabled.

Diagram



Fields

Field	Function
31 CONT	Continuous Peripheral Chip Select Enable Selects a continuous selection format. The bit is used in SPI Master mode. The bit enables the selected PCS signals to remain asserted between transfers. 0b - Return PCSn signals to their inactive state between transfers. 1b - Keep PCSn signals asserted between transfers.
30-28 CTAS	Clock and Transfer Attributes Select Selects which CTAR to use in master mode to specify the transfer attributes for the associated SPI frame. See the chip specific section for details to determine how many CTARs this device has. You should not program a value in this field for a register that is not present. 000b - CTAR0 001b - CTAR1 010b - Reserved 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - Reserved
27 EOQ	End Of Queue Host software uses this bit to signal to the module that the current SPI transfer is the last in a queue. At the end of the transfer, the EOQF bit in the SR is set. 0b - The SPI data is not the last data to transfer. 1b - The SPI data is the last data to transfer.
26 CTCNT	Clear Transfer Counter Clears the TCNT field in the TCR register. The TCNT field is cleared before the module starts transmitting the current SPI frame. 0b - Do not clear the TCR[TCNT] field. 1b - Clear the TCR[TCNT] field.
25 PE_MASC	Parity Enable or Mask T ASC delay in the current frame PE – This bit enables parity bit transmission and parity reception check for the SPI frame. MASC - The current frame has the “after SCK” delay masked if this bit is asserted. See Fast Continuous Selection Format for more details. **NOTE** This bit is used as Mask T_{ASC} in the Fast Continuous PCS mode when MCR[FCPCS] is set.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - PE - No parity bit included/checked. MASC - T_{ASC} delay is not masked and the current frame has the after SCK delay. 1b - PE - Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame. MASC - T_{ASC} delay is masked in the current frame.
24 PP_MCSC	Parity Polarity or Mask T CSC delay in the next frame PP - It controls the polarity of the parity bit transmitted and checked. MCSC - The next frame has the “PCS to SCK” delay masked if this bit is asserted. See Fast Continuous Selection Format for more details. **NOTE** This bit is used as Mask T_{CSC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0b - PP - Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. MCSC - T_{CSC} delay is not masked and the next frame has the PCS to SCK delay. 1b - PP - Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame. MCSC - T_{CSC} delay is masked in the next frame.
23-20 —	Always write the reset value to this field.
19-16 PCS	PCS Select which PCS signals are to be asserted for the transfer. 0000b - Negate the PCS[x] signal. 0001b - Assert the PCS[x] signal.
15-0 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

21.3.8 POP RX FIFO Register (POPR)

Offset

Register	Offset
POPR	38h

Function

POPR is used to read the RX FIFO. Eight- or sixteen-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register will generate a Transfer Error.

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

Fields

Field	Function
31-0 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

21.3.9 Transmit FIFO Registers (TXFR0 - TXFR3)

Offset

For a = 0 to 3:

Register	Offset
TXFRa	3Ch + (a × 4h)

Function

TXFRn registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the TXFRx registers does not alter the state of the TX FIFO. When the module is operating in Extended SPI mode, reading TXFRn registers is invalid.

Diagram

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

Fields

Field	Function
31-16 TXCMD_TXDATA	Transmit Command or Transmit Data In Master mode the TXCMD field contains the command that sets the transfer attributes for the SPI data.
15-0 TXDATA	Transmit Data Contains the SPI data to be shifted out.

21.3.10 Receive FIFO Registers (RXFR0 - RXFR3)

Offset

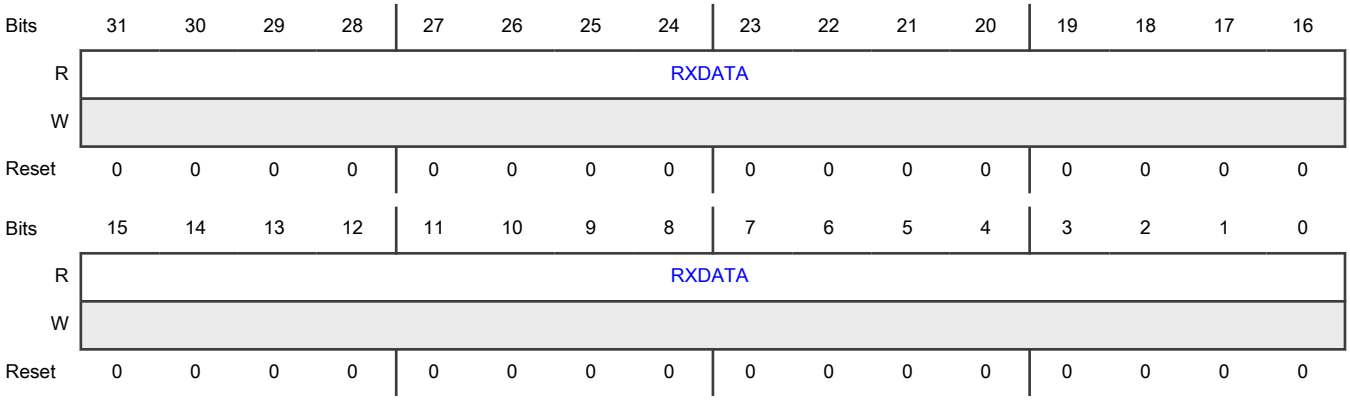
For a = 0 to 3:

Register	Offset
RXFRa	7Ch + (a × 4h)

Function

RXFRn provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The RXFR registers are read-only. Reading the RXFRx registers does not alter the state of the RX FIFO. The field MCR[MDIS] must be 0 when RXFR is read.

Diagram



Fields

Field	Function
31-0 RXDATA	Receive Data Contains the received SPI data.

21.3.11 Clock and Transfer Attributes Register Extended (CTARE0 - CTARE1)

Offset

For a = 0 to 1:

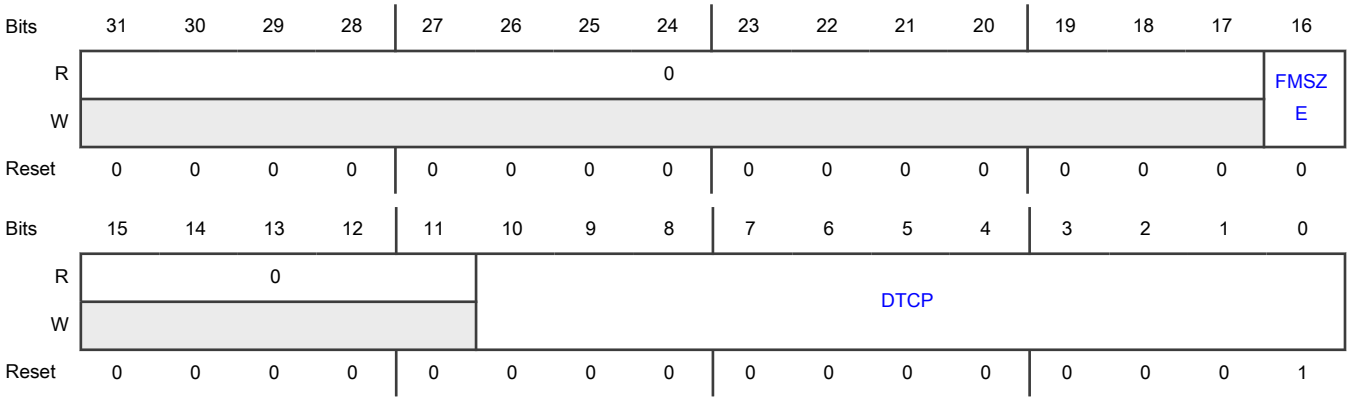
Register	Offset
CTAREa	11Ch + (a × 4h)

Function

CTARE registers are used to define the extended transfer attributes for an SPI frame. These registers are valid only when Extended SPI mode is enabled (MCR[XSPI]).

When the module is configured as a SPI master, the CTAS field in CMD FIFO entry selects which of the CTARE registers is used.

Diagram



Fields

Field	Function
31-17 —	Reserved
16 FMSZE	Frame Size Extended This field is valid only when MCR[XSPI] is set. This field concatenated with CTAR[FMSZ] defines the Frame size of the SPI frames to be transmitted. Effective frame size would be the concatenation of {CTARE[FMSZE], CTAR[FMSZ]} plus 1. 0b - Default Mode. Up to 16 bit SPI frames can be transfered. 1b - Up to 32 bit SPI frames can be transfered. Each Frame transfer will be a result of 2 simulataneous TX FIFO Pop Operation.
15-11 —	Reserved
10-0	Data Transfer Count Preload

Table continues on the next page...

Table continued from the previous page...

Field	Function
DTCP	This field is valid only when SPIx_MCR[XSPI] is set. This field defines the number of data frames (whose size is defined by CTARE[FMSZE] and CTAR[FMSZ]) to be transmitted using the Command frame which selected this SPIx_CTARE register. The value 0 is reserved and should not be written in this field. The default value of this field is 1.

21.3.12 Status Register Extended (SREX)

Offset

Register	Offset
SREX	13Ch

Function

The register contains status fields. The fields reflect the status of the module and indicate the occurrence of events. This register is not writable.

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	TXCTR4	0		RXCTR4	0			CMDCTR				CMDNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-15 —	Reserved
14 TXCTR4	TX FIFO Counter[4] This bit is an extension of SR[TXCTR]. The concatenated field {TXCTR4, TXCTR} indicates the number of valid entries in the TX FIFO. This field is incremented every time the PUSHR is written. And this field is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 —	Reserved
11 RXCTR4	RX FIFO Counter[4] This bit is an extension of SR[RXCTR]. The concatenated field {RXCTR4, RXCTR} indicates the number of entries in the RX FIFO. This field is decremented every time the POPR is read. And this field is incremented every time data is transferred from the shift register to the RX FIFO.
10-9 —	Reserved
8-4 CMDCTR	CMD FIFO Counter Indicates the number of entries in the CMD FIFO. The CMDCTR is incremented every time the command part of PUSHR is written. The CMDCTR is decremented every time a SPI command is executed (all data frames due to current command frame have been transmitted).
3-0 CMDNXTPTR	Command Next Pointer Indicates which CMD FIFO Entry is used during the next transfer. The CMDNXTPTR field is updated every time SPI data due to current command have been transmitted.

21.4 Functional description

The module supports full-duplex, synchronous serial communications between chips and peripheral devices.The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes.

The module has the following configuration

- The SPI Configuration in which the module operates as a basic SPI or a queued SPI.

The DCONF field in the Module Configuration Register (MCR) determines the module Configuration. SPI configuration is selected when DCONF within SPIx_MCR is 0b00.

The CTARn registers hold clock and transfer attributes. The SPI configuration allows to select which CTAR to use on a frame by frame basis by setting a field in the SPI command. The Extended SPI Mode (SPIx_MCR[XSPI]) further allows the usage CTAREn (CTARn Extended) registers which allows the user to send multiple data frames using a single command frame.

For more information, see the fields of CTAR registers. See CTAREx registers for information on the fields of CTARE registers.

Typical master to slave connections are shown in the following figure. When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave. The data that was in the master shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the Transfer Control Flag(TCF) bit in the Shift Register(SR) is set to indicate a completed frame transfer.

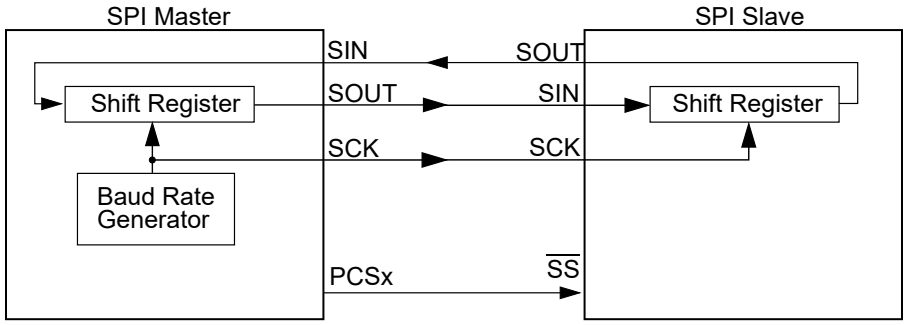


Figure 59. Serial protocol overview

Generally, more than one slave device can be connected to the module master. 4 Peripheral Chip Select (PCS) signals of the module masters can be used to select which of the slaves to communicate with.

The SPI configuration shares transfer protocol and timing properties which are described independently of the configuration in Transfer formats. The transfer rate and delay settings are described in Module baud rate and clock delay generation.

21.4.1 Starting and stopping module transfers

The module has two operating states: Stopped and Running. Both the states are independent of their configuration. The default state of the module is Stopped. In this state, the module initiates no serial transfers in the Master mode. The Stopped state is also a safe state for writing to the various configuration registers of the module, without causing undetermined results. In the Running state, serial transfers take place.

SR[TXRXS] indicates the state of the module. The value of this field is 1 if the module is in the Running state.

The module starts or transitions to the Running state when all of the following conditions are true:

- The value of SR[EOQF] = 0.
- The chip is not in the Debug mode or the value of MCR[FRZ] = 0.
- The value of MCR[HALT] = 0.

The module stops or transitions from the Running state to the Stopped state after the current frame, when any one of the following conditions exist:

- The value of SR[EOQF] = 1.
- The chip is in Debug mode and the value of MCR[FRZ] = 1.
- The value of MCR[HALT] = 1.

State transitions from running to stopped occur on the next frame boundary if a transfer is in progress, and occur immediately if no transfers are in progress.

21.4.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The module is in SPI configuration when the DCONF field in the MCR is 0b00. The SPI frames can be 32 bits long. The host CPU or a DMA controller transfers the SPI data from the external to the module RAM queues to a TX FIFO buffer. The received data is stored in entries in the RX FIFO buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the module. The operation of FIFO buffers is described in the following sections:

- Transmit First In First Out (TX FIFO) buffering mechanism
- Command First In First Out (CMD FIFO) Buffering Mechanism
- Receive First In First Out (RX FIFO) buffering mechanism

The interrupt and DMA request conditions are described in Interrupts/DMA requests.

In Master mode the module initiates and controls the transfer according to the fields of the executing SPI Command.

21.4.2.1 Master mode

In SPI Master mode, the module initiates the serial transfers by controlling the SCK and the PCS signals. The executing SPI Command determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See PUSH TX FIFO Register In Master Mode (PUSHR) for details on the SPI command fields. The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin. In SPI Master mode, each SPI frame to be transmitted has a command associated with it, allowing for transfer attribute control on a frame by frame basis.In Extended SPI Master Mode, multiple SPI frames can have a single command associated with them allowing for efficient SPI frame transfers requiring common transfer attributes. In addition, the Extended SPI Mode allows for larger frame sizes of up to 32 bits.

21.4.2.2 FIFO disable operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO, CMD FIFO or RX FIFO. The module operates as a double-buffered simplified SPI when the FIFOs are disabled. The Transmit and Receive side of the FIFOs are disabled separately. Setting the MCR[DIS_TXF] bit disables the TX FIFO and CMD FIFO, and setting the MCR[DIS_RXF] bit disables the RX FIFO.

The FIFO disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the PUSHR and received data is read from the POPR.

When the TX FIFO and CMD FIFO are disabled:

- SR[TFFF], SR[TFUF], SREX[CMDCTR] and SR[TXCTR] behave as if there is a one-entry FIFO
- The contents of TXFRs, SR[TXNXTPTR] and SREX[CMDNXTPTR] are undefined

Similarly, when the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the SR behave as if there is a one-entry FIFO, but the contents of the RXFR registers and POPNXTPTR are undefined.

21.4.2.3 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission. The TX FIFO holds 4 words, each consisting of SPI data. The number of entries in the TX FIFO is device-specific. SPI data is added to the TX FIFO by writing to the Data Field of module PUSH FIFO Register (PUSHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the module Status Register (SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time a 8- or 16-bit write takes place to PUSHR[TXDATA] or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates the TX FIFO Entry that will be transmitted during the next transfer. The TXFRn Registers are invalid in the Extended SPI Mode, since the TX FIFO and CMD FIFO can be used independently. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR number and it rolls over after reaching the maximum.

21.4.2.3.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO and CMD FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set. The TFFF bit is cleared when TX FIFO is full and the DMA controller indicates that a write to PUSHR is complete. Writing a '1' to the TFFF bit also clears it. The TFFF can generate a DMA request or an interrupt request. See Transmit FIFO Fill Interrupt or DMA Request for details.

The module ignores attempts to push data to a full TX FIFO, and the state of the TX FIFO does not change and no error condition is indicated.

21.4.2.3.2 Draining the TX FIFO

The module removes (drains) the TX FIFO entries by shifting SPI data out through the Shift register. It transfers entries from the TX FIFO to the Shift register, and the entries are shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the Shift register, the TX FIFO counter decrements by 1. When the value of MCR[XSPI] is 1,

and if the frame size of SPI data to be transmitted is more than 16 bits, then it causes two data entries to be popped from TX FIFO simultaneously. These entries are transferred to the Shift register. The first of the two popped entries forms the 16 least significant bits of the SPI frame to be transmitted. Such an operation also causes the TX FIFO counter to decrement by 2. At the end of a transfer, the value of SR[TCF] is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a 1 to MCR[CLR_TXF].

21.4.2.4 Command First In First Out (CMD FIFO) Buffering Mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission. When Extended SPI Mode (MCR[XSPI]) is disabled, the TX FIFO and CMD FIFO must be filled together, i.e. write enables should be given for both the Data and Command fields while performing a PUSHR operation.When Extended SPI Mode (MCR[XSPI]) is enabled, the TX FIFO and CMD FIFO can be filled independently.

The CMD FIFO holds 4 words, each representing SPI command fields. The number of entries in the CMD FIFO is chip-specific. SPI command is added to the CMD FIFO by writing to the command field of PUSHR. CMD FIFO entries can only be removed from the CMD FIFO by being shifted out (to help transmit SPI data) or by flushing the CMD FIFO.

When Extended SPI Mode (MCR[XSPI]) is disabled, every CMD FIFO entry has a corresponding single TX FIFO entry attached to it because both these FIFO's are filled simultaneously.

When Extended SPI Mode (MCR[XSPI]) is enabled, every CMD FIFO entry can have multiple TX FIFO entries attached to it. Thus a single CMD FIFO entry can be used to transmit multiple TX FIFO entries. The CTARE[DTCP] field decides the number of SPI Data Frames having frame size as {FMSZE, FMSZ} to be transmitted using the current Command Entry. The CTAR/CTARE registers pointed by the CTAS field in the Command frame gives the FMSZ and FMSZE fields respectively. The time for which a command entry is in use is known as a Command Cycle. The Busy Flag SR[BSYF] is asserted for the duration of the Command Cycle except for the last SPI frame in the Command Cycle.

The CMD FIFO Counter field (CMDCTR) in the SPI Status Register extended(SREX) indicates the number of valid entries in the CMD FIFO. The CMDCTR field is updated every time a 8- or 16-bit write takes place on the lower half of SPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The TXFRn Registers are invalid in the Extended SPI Mode, since the TX FIFO and CMD FIFO can be used independently. The CMDNXTPTR field indicates which CMD FIFO Entry will be used during the next command cycle. The CMDNXTPTR field is incremented every time the last SPI data in the command cycle is transferred from the TX FIFO to the shift register and it rolls over after reaching the maximum.

21.4.2.5 Receive First In First Out (RX FIFO) buffering mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds 4 received SPI data frames. The number of entries in the RX FIFO is device-specific. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data are removed (popped) from the RX FIFO by reading the module POP RX FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the module's Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the SR points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in a number of 32-bit registers. For example, POPNXTPTR equal to two means that the RXFR2 contains the received SPI data that will be returned when the POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR number and it rolls over after reaching the maximum.

21.4.2.5.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO, the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or

shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

21.4.2.5.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the module POP RX FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data, read from the empty RX FIFO, is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete or by writing a 1 to it.

21.4.3 Module baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the system clock frequency by a prescaler and a scaler with the option for doubling the baud rate. The following figure shows conceptually how the SCK signal is generated.

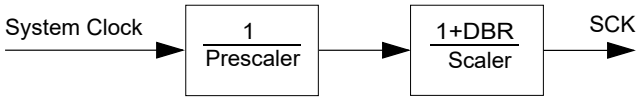


Figure 60. Communications clock prescalers and scalers

21.4.3.1 Baud rate generator

The baud rate is the frequency of the SCK. The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR, and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. The following table shows an example of how to compute the baud rate.

Table 84. Baud rate computation example

f_P	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

NOTE

The clock frequencies mentioned in the preceding table are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

21.4.3.2 PCS to SCK Delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See Figure 61 for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTAR*x* registers select the PCS to SCK delay by the formula in the CSSCK field description. The following table shows an example of how to compute the PCS to SCK delay.

Table 85. PCS to SCK delay computation example

f_{SYS}	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

21.4.3.3 After SCK Delay (t_{ASC})

The After SCK Delay is the length of time between the last edge of SCK and the negation of PCS. See Figure 61 and Figure 62 for illustrations of the After SCK delay. The PASC and ASC fields in the CTAR*x* registers select the After SCK Delay by the formula in the ASC field description. The following table shows an example of how to compute the After SCK delay.

Table 86. After SCK Delay computation example

f_P	PASC	Prescaler	ASC	Scaler	After SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

21.4.3.4 Delay after Transfer (t_{DT})

The Delay after Transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See Figure 61 for an illustration of the Delay after Transfer. The PDT and DT fields in the CTAR*x* registers select the Delay after Transfer by the formula in the DT field description. The following table shows an example of how to compute the Delay after Transfer.

Table 87. Delay after Transfer computation example

f_P	PDT	Prescaler	DT	Scaler	Delay after Transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

When in Non-Continuous Clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] field description. When in Continuous Clock mode, the delay is fixed at 1 SCK period.

21.4.4 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In Master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTARn) select the polarity and phase of the serial clock, SCK.

- CPOL - Selects the idle state polarity of the SCK
- CPHA - Selects if the data on SOUT is valid before or on the first SCK edge

The module supports two different transfer formats:

- Classic SPI with CPHA=0
- Classic SPI with CPHA=1

In the interface configurations, the module provides the option of keeping the PCS signals asserted between frames. See Continuous selection format for details.

21.4.4.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in following figure is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

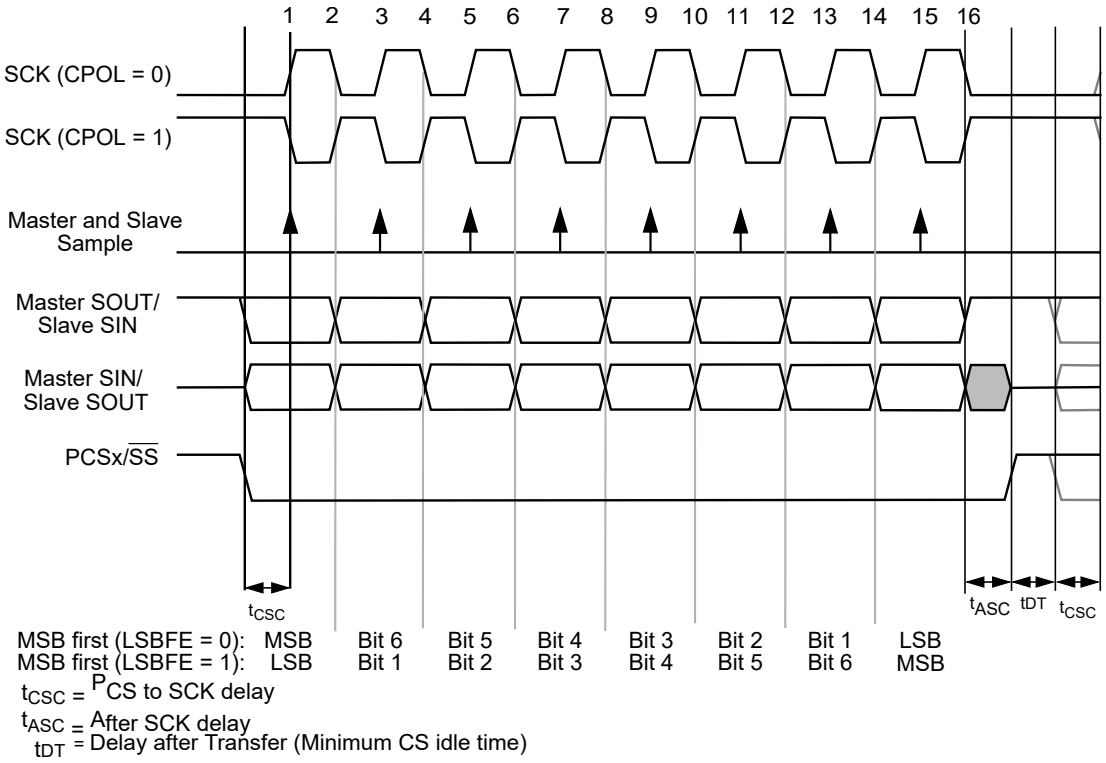


Figure 61. Module transfer timing diagram (CPHA=0, FMSZ=8)

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals. At the second edge of the SCK, the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

21.4.4.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in the following figure is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format, the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges.

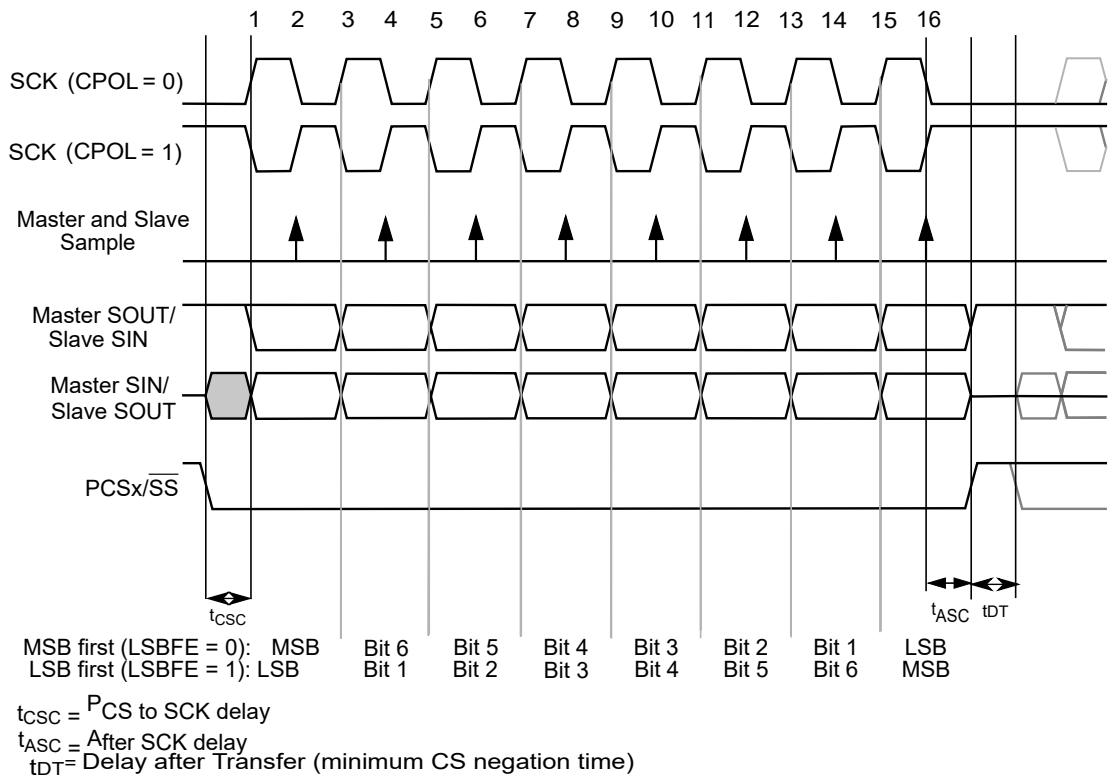


Figure 62. Module transfer timing diagram (CPHA=1, FMSZ=8)

The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

21.4.4.3 Continuous selection format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The continuous selection format provides the flexibility to handle the following case. The format is enabled for the SPI configuration by setting the CONT bit in the SPI command.

When the value of CONT is 0, the module drives the asserted Chip Select signals to their idle states in between frames. The idle states of these signals are selected by the PCSIS*n* fields in the MCR. The following figure shows the timing diagram for two 4-bit transfers with CPHA = 1 and CONT = 1.

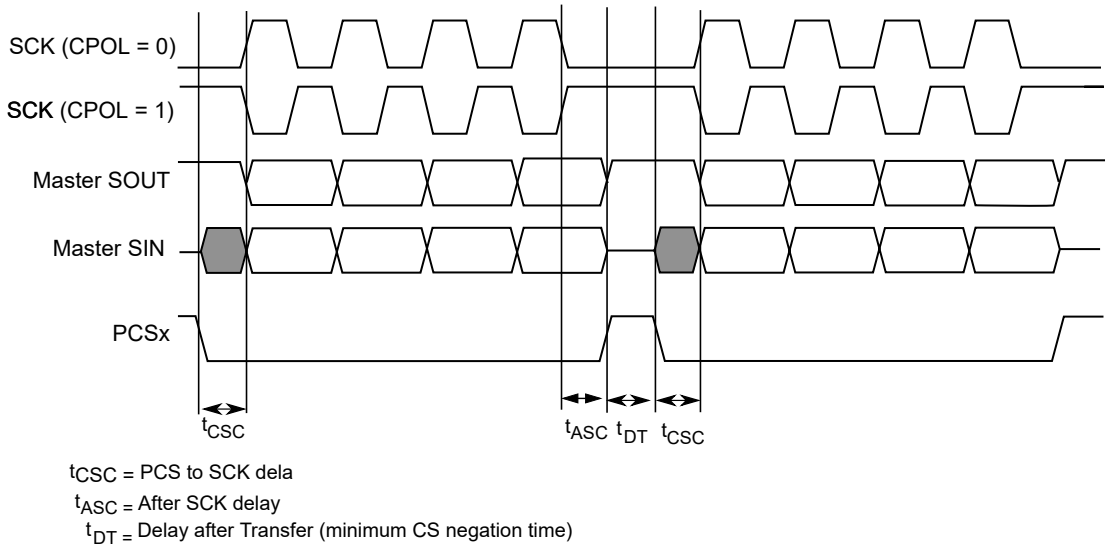


Figure 63. Example of non-continuous format (CPHA=1, CONT=0)

When CONT = 1, the PCS signal remains asserted for the duration of the two transfers. The "Delay between Transfers" (t_{DT}) is not inserted between the transfers. The following figure shows the timing diagram for two 4-bit transfers with CPHA = 1 and CONT = 1.

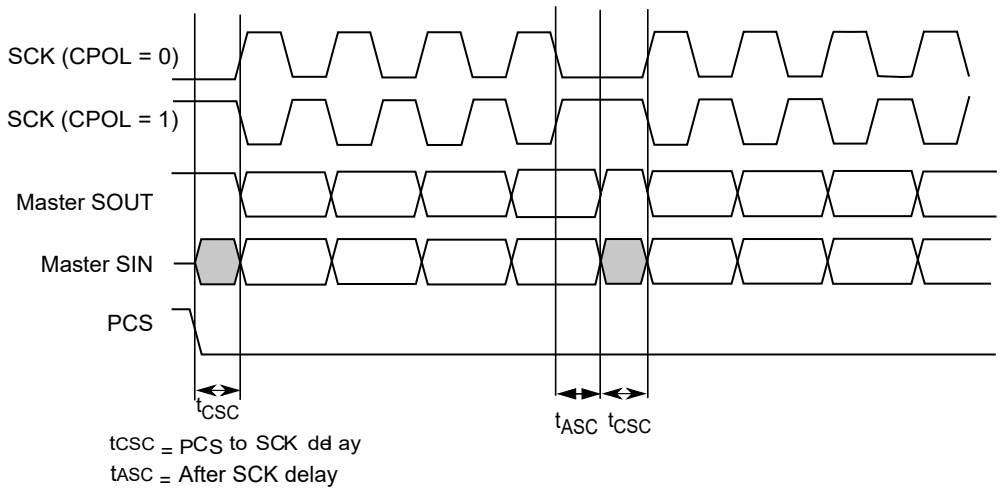


Figure 64. Example of continuous transfer (CPHA=1, CONT=1)

When using the module with continuous selection, follow these rules:

- All transmit commands must have the same PCS*n* field programming.
- The CTARs, selected by transmit commands, must be programmed with the same transfer attributes. Only the FMSZ field can be programmed differently in these CTARs.
- When transmitting multiple frames in this mode, the software must ensure that the last frame has USHR[CONT] deasserted in Master mode and the software must provide sufficient frames in the TX_FIFO to be sent out in Slave mode and the master deasserts the PCSn at end of transmission of the last frame.
- PUSHR[CONT] must be deasserted before asserting MCR[HALT] in master mode. This ensures that the PCSn signals are deasserted. Asserting MCR[HALT] during continuous transfer causes the PCSn signals to be in unknown state and hence the slave device cannot transition from Running to Stopped state.

NOTE

You must fill the TX FIFO with the number of entries that are concatenated together under one PCS assertion for both master before the TX FIFO is empty.

21.4.4.4 Fast continuous selection format

The fast continuous selection format has a functioning similar to the continuous selection format except that the inter command delays, t_{ASC} and t_{CSC}, can be masked out and are not inserted by the hardware.

NOTE

The fast continuous selection format is available in the SPI configuration only when the Continuous SCK mode is disabled. Masking of delays is not allowed if the transfer is non-continuous.

The fast continuous selection format is enabled by writing a 1 to MCR[FCPCS]. When this field is asserted, the MASC and MCSC fields of the PUSHR perform the function of mask bits for the transmit frame. These fields individually mask the t_{ASC} and t_{CSC} delays as programmed by the software. A normal continuous selection format includes these two delays for each frame, which is transmitted with the CONT field asserted. To avoid these delays and to speed up the transfer process, the software can mask these delays while programming the command in the PUSHR.

While masking the delays, the software must follow these masking rules, otherwise, correct operation is not guaranteed.

- The MASC field masks the “After SCK” delay for the current frame.
- The MCSC field masks the “PCS to SCK” delay for the next frame.
- The t_{ASC} delay must not be masked when the current frame is the last frame in the continuous selection format.
- The "PCS to SCK" delay for the first frame in the continuous selection format cannot be masked.
- Masking of only t_{ASC} is not allowed. If it is masked, then t_{CSC} must be masked too.
- Masking of both t_{ASC} and t_{CSC} delays is allowed. In this case, the delay between two frames is equal to half the baud rate (baud rate is 1 SCK cycle duration) configured by the software.
- Masking of only t_{CSC} is allowed. In this case, the delay between two frames is equal to the t_{ASC} time and thus the software must ensure that the t_{ASC} time is greater than the baud rate.
- The software must not mask these delays if the continuous selection format is not used and MCR[FCPCS] is asserted.
- Rules applicable to the continuous selection format are applicable here too.

The following figure shows the timing for a fast continuous selection format transfer. Here, seven frames are transferred with both t_{ASC} and t_{CSC} delays masked except for the last frame that terminated the transfer. The last frame has t_{ASC} delay at its end.

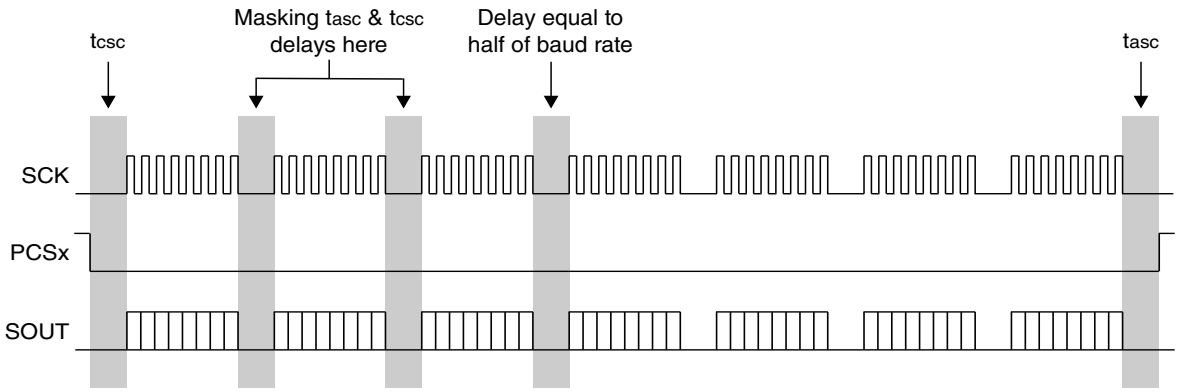


Figure 65. Example of fast continuous selection format

In case any chip select is to be changed, first the fast continuous selection format should be terminated and appropriate delays must be introduced.

21.4.5 Continuous Serial Communications Clock

The module provides the option of generating a Continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous SCK only if MCR[HALT] bit is low. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA=1. Clearing CPHA is ignored if the CONT_SCKE bit is set. .

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the module is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the module is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle. The following figure is the timing diagram for Continuous SCK format with Continuous Selection disabled.

NOTE

In Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TX FIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

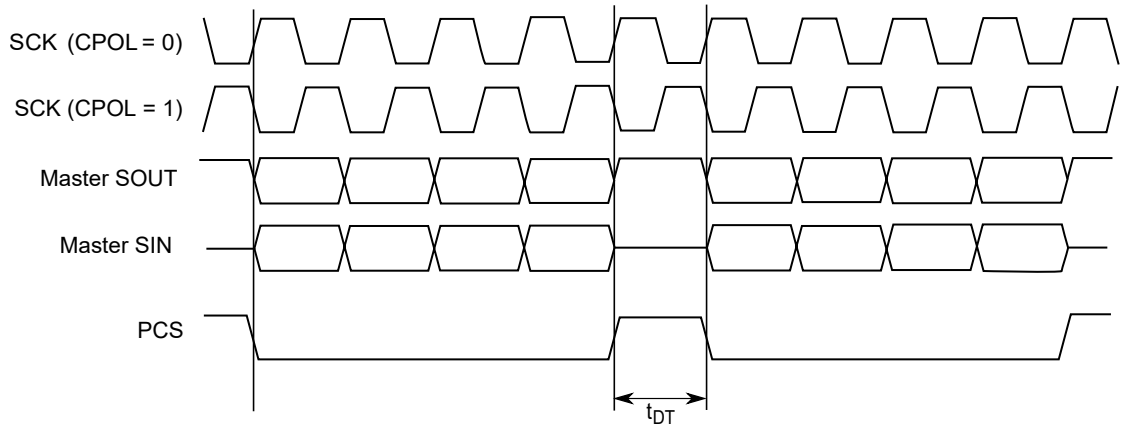


Figure 66. Continuous SCK Timing Diagram (CONT=0)

If the CONT bit in the TX FIFO entry is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT, that is, SOUT pulled high. This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the TX FIFO.
- Continuous SCK with CONT bit set and entering Stopped state (refer to Starting and stopping module transfers).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

The following figure shows timing diagram for Continuous SCK format with Continuous Selection enabled.

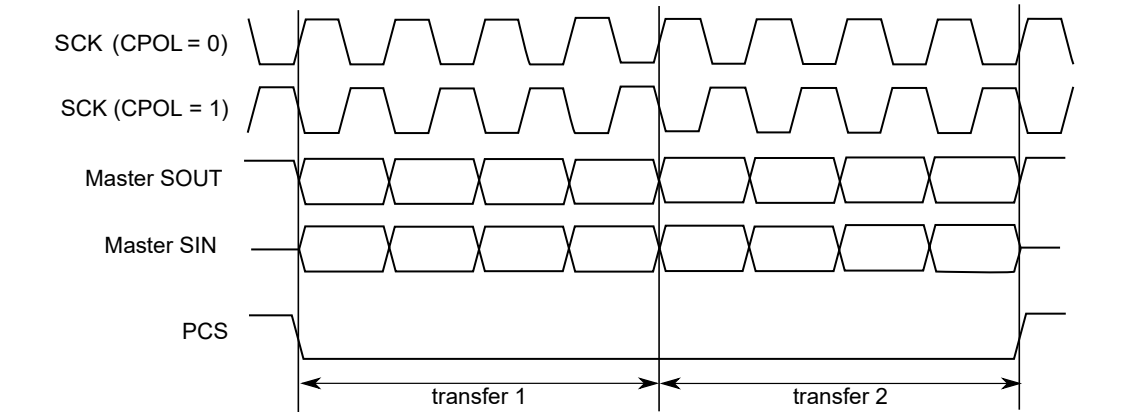


Figure 67. Continuous SCK timing diagram (CONT=1)

21.4.6 Parity Generation and Check

The module can generate and check parity in the serial frame. The parity bit replaces the last transmitted bit in the frame. The parity is calculated for all transmitted data bits in frame, not including the last data bit that would be transmitted. The parity generation/control is done on frame basis. The registers field setting frame size defines the total number of bits in the frame, including the parity bit. Thus, to transmit/receive the same number of data bits with parity check, increase the frame size by one versus the same data size frame without the parity check.

Parity can be selected as odd or even. Parity Errors in the received frame set Parity Error flags in the Status register. The Parity Error Interrupt Requests are generated if enabled. The module can be programmed to stop frame transmission in case of a frame reception with parity error.

21.4.6.1 Parity for SPI Frames

When the module is in the master mode the parity generation is controlled by PE and PP bits of the CMD FIFO entries (PUSHR). Setting the PE bit enables parity generation for transmitted SPI frames and parity check for received frames. PP bit defines polarity of the parity bit.

When continuous PCS selection is used to transmit SPI data, two parity generation scenarios are available:

- Generate/check parity for the whole frame
- Generate/check parity for each sub-frame separately.

To generate/check parity for the whole frame set PE bit only in the last command/TX FIFO entry, forming this frame (with the PUSHR register).

To generate/check parity for each sub-frame set PE bit in each command/TX FIFO entry, forming this frame.

If the parity error occurs for received SPI frame, the SR[SPEF] bit is set. If MCR[PES] bit is set, the module stops SPI frames transmission. To resume SPI operation clear the SR[SPEF] or the MCR[PES] bits.

21.4.7 Interrupts/DMA requests

The module has several conditions that can generate only interrupt requests and two conditions that can generate interrupt or DMA requests. The following table lists these conditions.

Table 88. Interrupt and DMA request conditions

Condition	Flag	Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	-
TX FIFO Fill	TFFF	Yes	Yes
CMD FIFO Fill	CMDFFF	Yes	Yes
TX FIFO Invalid Write	TFIWF	Yes	-
Transfer Complete	TCF	Yes	-
CMD Transfer Complete	CMDTCF	Yes	-
TX FIFO Underflow	TFUF	Yes	-
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	-
SPI Parity Error	SPEF	Yes	-

Each condition has a flag bit in the module Status Register (SR) and a Request Enable bit in the DMA/Interrupt Request Select and Enable Register (RSER). Certain flags (as shown in above table) generate interrupt requests or DMA requests depending on configuration of RSER register.

The module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

21.4.7.1 End Of Queue interrupt request

The End Of Queue (EOQ) interrupt request indicates that the end of a transmit queue is reached. The module generates the interrupt request when EOQ interrupt requests are enabled (RSER[EOQF_RE]) and the EOQ bit in the executing SPI command is 1.

When Extended SPI mode is enabled (MCR[XSPI]) and the EOQ bit in the executing SPI command is 1, the module generates the EOQ interrupt request when the last bit of the last data frame in the command cycle has been transmitted.

When Extended SPI mode is disabled (MCR[XSPI]), the module generates the interrupt request when the last bit of the SPI frame with EOQ bit set is transmitted.

21.4.7.2 Transmit FIFO Fill Interrupt or DMA Request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

NOTE

TFFF flag clears automatically when DMA is used to fill TX FIFO. Configure the DMA to fill only one FIFO location per transfer.

To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TX FIFO:

1. Wait until TFFF = 1.
2. Write data to PUSHR using CPU.
3. Clear TFFF by writing a 1 to its location. If TX FIFO is not full, this flag will not clear.

21.4.7.3 Command FIFO Fill Interrupt or DMA Request

The Command FIFO Fill Request indicates that the CMD FIFO is not full. The Command FIFO Fill Request is generated when the number of entries in the CMD FIFO is less than the maximum number of possible entries, and the CMDFFF_RE bit in the RSER is set. The CMDFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

This Request is useful when MCR[XSPI] is enabled, since TX FIFO and CMD FIFO can be filled independantly. If MCR[XSPI] is disabled, then ‘TX FIFO Fill Interrupt or DMA Request’ will suffice to fill both FIFO’s since both FIFO’s must be filled simultaneously.

NOTE

CMDFFF flag clears automatically when DMA is used to fill CMDFIFO.

To clear CMDFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill CMDFIFO:

1. Wait until CMDFFF = 1
2. Write data to PUSHR using CPU.
3. Clear CMDFFF by writing a 1 to its location. If CMD FIFO is not full, this flag will not clear.

21.4.7.4 Transmit FIFO Invalid Write Interrupt Request

The Transmit FIFO Invalid Write Request is valid only when MCR[XSPI] is enabled. This Request indicates that Data exists in the TX FIFO while the CMD FIFO is empty. Since no Command Fields are associated with the Data present in TX FIFO, this data is considered invalid until a Command Entry becomes available. The Transmit FIFO Invalid Write Request is generated for the above condition when TFIWF_RE bit is set in the RSER.

21.4.7.5 Transfer Complete Interrupt Request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

21.4.7.6 Command Transfer Complete Interrupt Request

The Command Transfer Complete Request indicates the end of transfer of the last SPI frame in a Command Cycle. The Transfer Complete Request is generated for the above condition when the CMDTCF_RE bit is set in the RSER.

21.4.7.7 Receive FIFO Drain Interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated. Configure the DMA to drain only one FIFO location per transfer.

21.4.7.8 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

21.4.7.9 SPI Frame Parity Error Interrupt Request

The SPI Frame Parity Error Flag indicates that a SPI frame with parity error had been received. The SPEF_RE bit in the RSER must be set for the interrupt request to be generated.

21.4.8 Power saving features

The module supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory mapped logic

21.4.8.1 Stop mode (External Stop mode)

This module supports the Stop mode protocol. When a request is made to enter External Stop mode, the module acknowledges the request . If a serial transfer is in progress, then this module waits until it reaches the frame boundary before it is ready to have its clocks shut off . While the clocks are shut off, this module's memory-mapped logic is not accessible. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

21.4.8.2 Module Disable mode

Module Disable mode is a block-specific mode that the module can enter to save power. Host CPU can initiate the Module Disable mode by setting the MDIS bit in the MCR. The Module Disable mode can also be initiated by hardware.

When the MDIS bit is set, the module negates the Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, it is said to have entered Module Disable Mode. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state.If implemented, the Clock Enable signal can stop the clock to the non-memory mapped logic. When Clock Enable is negated, the module is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different effect when the module is in the Module Disable mode. Reading the RX FIFO Pop Register does not change the state of the RX FIFO. Similarly, writing to the PUSHR Register does not change the state of the TX FIFO or CMD FIFO. Clearing either of the FIFOs has no effect in the Module Disable mode. Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect in the Module Disable mode. In the Module Disable mode, all status bits and register flags in the module return the correct values when read, but writing to them has no effect. Writing to the TCR during Module Disable mode has no effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable mode.

21.5 Initialization/application information

This section describes how to initialize the module.

21.5.1 How to manage queues

The queues are not part of the module, but it includes features in support of queue management. Queues are primarily supported in SPI configuration.

1. When module executes last command word from a queue, the EOQ bit in the command word is set to indicate it that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the module in the Stopped state. The TXRXS bit is cleared to indicate the Stopped state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO (and CMD FIFO) and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO (and CMD FIFO) by writing a 1 to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a '1' to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the module TX FIFO, (and CMD FIFO) and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

21.5.2 Initializing Module in Master Mode

Once the appropriate mode in MCR[MSTR] is configured, the module is enabled by clearing MCR[HALT]. It should be ensured that module Slave is enabled before enabling it's Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

21.5.3 Baud rate settings

The following table shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is cleared.

NOTE

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

Table 89. Baud rate values (bps)

		Baud rate divider prescaler values			
		2	3	5	7
Baud Rate Scaler Values	2	25.0M	16.7M	10.0M	7.14M
	4	12.5M	8.33M	5.00M	3.57M
	6	8.33M	5.56M	3.33M	2.38M
	8	6.25M	4.17M	2.50M	1.79M
	16	3.12M	2.08M	1.25M	893k
	32	1.56M	1.04M	625k	446k
	64	781k	521k	312k	223k
	128	391k	260k	156k	112k
	256	195k	130k	78.1k	55.8k
	512	97.7k	65.1k	39.1k	27.9k
	1024	48.8k	32.6k	19.5k	14.0k
	2048	24.4k	16.3k	9.77k	6.98k
	4096	12.2k	8.14k	4.88k	3.49k
	8192	6.10k	4.07k	2.44k	1.74k
	16384	3.05k	2.04k	1.22k	872
	32768	1.53k	1.02k	610	436

21.5.4 Delay settings

The following table shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (T_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

NOTE

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

Table 90. Delay values

		Delay scaler values (CTAR[PDT])			
		1	3	5	7
Delay scaler values (CTAR[DT])	2	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 µs
	32	320.0 ns	960.0 ns	1.6 µs	2.2 µs
	64	640.0 ns	1.9 µs	3.2 µs	4.5 µs
	128	1.3 µs	3.8 µs	6.4 µs	9.0 µs
	256	2.6 µs	7.7 µs	12.8 µs	17.9 µs
	512	5.1 µs	15.4 µs	25.6 µs	35.8 µs
	1024	10.2 µs	30.7 µs	51.2 µs	71.7 µs
	2048	20.5 µs	61.4 µs	102.4 µs	143.4 µs
	4096	41.0 µs	122.9 µs	204.8 µs	286.7 µs
	8192	81.9 µs	245.8 µs	409.6 µs	573.4 µs
	16384	163.8 µs	491.5 µs	819.2 µs	1.1 ms
	32768	327.7 µs	983.0 µs	1.6 ms	2.3 ms
	65536	655.4 µs	2.0 ms	3.3 ms	4.6 ms

21.5.5 Calculation of FIFO pointer addresses

Complete visibility of the FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the CMD FIFO the first-in pointer is the Command Next Pointer (CMDNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). The following figure illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over. See Transmit First In First Out (TX FIFO) buffering mechanism, Command First In First Out (CMD FIFO) Buffering Mechanism and Receive First In First Out (RX FIFO) buffering mechanism for details on the FIFO operation.

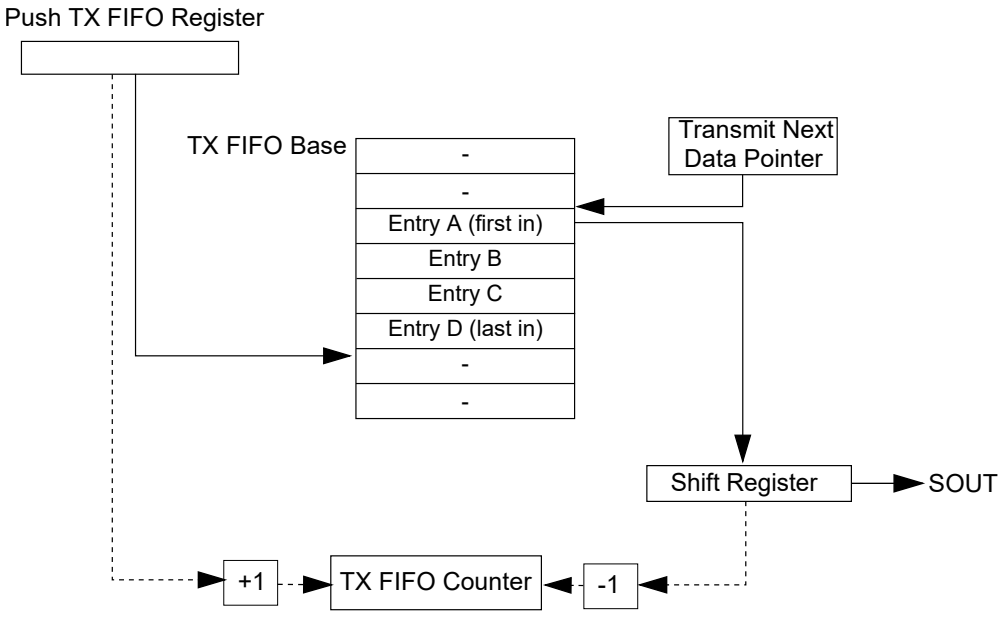


Figure 68. TX FIFO pointers and counter

21.5.5.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + \left(4 \times \text{TXNXTPTR}\right)$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times \left(\text{TXCTR} + \text{TXNXTPTR} - 1\right) \text{mod} \left(\text{TXFIFOdepth}\right)$$

TX FIFO Base - Base address of TX FIFO

TXCTR - TX FIFO Counter

TXNXTPTR - Transmit Next Pointer

TX FIFO Depth - Transmit FIFO depth, implementation specific

21.5.5.2 Address Calculation for the First-in Entry and Last-in Entry in the CMD FIFO

Use the following equation to compute the memory address of the first-in entry in the CMD FIFO:

- First-in EntryAddress = CMDFIFOBase+[4xCMDNXTPTR]

Use the following equation to compute the memory address of the last-in entry in the CMD FIFO:

- Last-in EntryAddress = CMDFIFOBase+4x[CMDCTR+CMDNXTPTR - 1]mod[CMDFIFOdepth]

Here is an explanation of the different components of these equations:

- CMD FIFO Base: Base address of CMD FIFO
- CMDCTR: CMD FIFO Counter
- CMDNXTPTR: Command Next Pointer
- CMD FIFO Depth: Command FIFO depth, implementation specific

21.5.5.3 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{RX FIFOBase} + \left(4 \times \text{POPNXTPTR}\right)$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{RX FIFO Base} + 4 \times \left(\text{RXCTR} + \text{POPNXTPTR} - 1\right) \text{mod} \left(\text{RXFIFOdepth}\right)$$

RX FIFO Base - Base address of RX FIFO

RXCTR - RX FIFO counter

POPNXTPTR - Pop Next Pointer

RX FIFO Depth - Receive FIFO depth, implementation specific

Chapter 22
Thermal Monitoring Unit (TMU)

22.1 TMU as implemented on the chip

The chip includes a thermal diode as well as a thermal monitor capable of a resolution of 4 degrees Celsius.

The table below shows the placement of the temperature sensors.

Table 91. Local Temperature Sensor Placement in Chip

Temperature Sensor ID	Placement
0	Next to VSPA
1	Near DCS (ADC-DAC)

22.2 TMU Calibration

The TMU calculates temperature by reading one of the temperature sensor sites on-chip. The temperature is calculated from a table of calibration points, programmed by the preboot loader or software from data supplied by NXP.

Table 92. TMU Calibration data

Register	Value	Comment
TTR0CR	**0x000B_0000**	**12 points @ 0°C**
TTCFGR	0x0000_0000	Point 0 @ 0°C
TSCFGR	0x0000_0025	
TTCFGR	0x0000_0001	Point 1 @ 4°C
TSCFGR	0x0000_002B	
TTCFGR	0x0000_0002	Point 2 @ 8°C
TSCFGR	0x0000_0032	
TTCFGR	0x0000_0003	Point 3 @ 12°C
TSCFGR	0x0000_0038	
TTCFGR	0x0000_0004	Point 4 @ 16°C
TSCFGR	0x0000_003F	
TTCFGR	0x0000_0005	Point 5 @ 20°C
TSCFGR	0x0000_0045	
TTCFGR	0x0000_0006	Point 6 @ 24°C
TSCFGR	0x0000_004C	
TTCFGR	0x0000_0007	Point 7 @ 28°C
TSCFGR	0x0000_0052	
TTCFGR	0x0000_0008	Point 8 @ 32°C
TSCFGR	0x0000_0059	

Table continues on the next page...

Table 92. TMU Calibration data (continued)

Register	Value	Comment
TTCFGR	0x0000_0009	Point 9 @ 36°C
TSCFGR	0x0000_005F	
TTCFGR	0x0000_000A	Point 10 @ 40°C
TSCFGR	0x0000_0066	
TTCFGR	0x0000_000B	Point 11 @ 44°C
TSCFGR	0x0000_006C	
TTR1CR	**0x000A_0026**	**11 points @ 38°C**
TTCFGR	0x0001_0000	Point 0 @ 38°C
TSCFGR	0x0000_001C	
TTCFGR	0x0001_0001	Point 1 @ 42°C
TSCFGR	0x0000_0024	
TTCFGR	0x0001_0002	Point 2 @ 46°C
TSCFGR	0x0000_002C	
TTCFGR	0x0001_0003	Point 3 @ 50°C
TSCFGR	0x0000_0034	
TTCFGR	0x0001_0004	Point 4 @ 54°C
TSCFGR	0x0000_003C	
TTCFGR	0x0001_0005	Point 5 @ 58°C
TSCFGR	0x0000_0044	
TTCFGR	0x0001_0006	Point 6 @ 62°C
TSCFGR	0x0000_004C	
TTCFGR	0x0001_0007	Point 7 @ 66°C
TSCFGR	0x0000_0054	
TTCFGR	0x0001_0008	Point 8 @ 70°C
TSCFGR	0x0000_005C	
TTCFGR	0x0001_0009	Point 9 @ 74°C
TSCFGR	0x0000_0064	
TTCFGR	0x0001_000A	Point 10 @ 78°C
TSCFGR	0x0000_006C	
TTR2CR	**0x0008_0048**	**9 points @ 72°C**
TTCFGR	0x0001_0000	Point 0 @ 38°C
TSCFGR	0x0000_001C	

Table continues on the next page...

Table 92. TMU Calibration data (continued)

Register	Value	Comment
TTCFGR	0x0001_0001	Point 1 @ 42°C
TSCFGR	0x0000_0024	
TTCFGR	0x0001_0002	Point 2 @ 46°C
TSCFGR	0x0000_002C	
TTCFGR	0x0001_0003	Point 3 @ 50°C
TSCFGR	0x0000_0034	
TTCFGR	0x0001_0004	Point 4 @ 54°C
TSCFGR	0x0000_003C	
TTCFGR	0x0001_0005	Point 5 @ 58°C
TSCFGR	0x0000_0044	
TTCFGR	0x0001_0006	Point 6 @ 62°C
TSCFGR	0x0000_004C	
TTCFGR	0x0001_0007	Point 7 @ 66°C
TSCFGR	0x0000_0054	
TTCFGR	0x0001_0008	Point 8 @ 70°C
TSCFGR	0x0000_005C	
TTCFGR	0x0001_0009	Point 9 @ 74°C
TSCFGR	0x0000_0064	
TTCFGR	0x0001_000A	Point 10 @ 78°C
TSCFGR	0x0000_006C	
TTR3CR	**0x0007_0061**	**8 points @ 97°C**
TTCFGR	0x0003_0000	Point 0 @ 97C
TSCFGR	0x0000_000E	
TTCFGR	0x0003_0001	Point 1 @ 101°C
TSCFGR	0x0000_001A	
TTCFGR	0x0003_0002	Point 2 @ 105°C
TSCFGR	0x0000_0026	
TTCFGR	0x0003_0003	Point 3 @ 109°C
TSCFGR	0x0000_0032	
TTCFGR	0x0003_0004	Point 4 @ 113°C
TSCFGR	0x0000_003E	
TTCFGR	0x0003_0005	Point 5 @ 117°C

Table continues on the next page...

Table 92. TMU Calibration data (continued)

Register	Value	Comment
TSCFGR	0x0000_004A	
TTCFGR	0x0003_0006	Point 6 @ 121°C
TSCFGR	0x0000_0056	
TTCFGR	0x0003_0007	Point 7 @ 125°C
TSCFGR	0x0000_0063	

22.3 Thermal Monitoring Unit Introduction

The Thermal Monitoring Unit (TMU) monitors and reports the temperature from one or more remote temperature measurement sites located on chip.

22.3.1 TMU Overview

The TMU has access to multiple temperature measurement sites strategically located on the chip. It monitors these sites and can signal an alarm if a programmed threshold is ever exceeded. The upper and lower temperature range is continuously captured. A set of reporting registers allow for reading the current temperature at monitored sites.

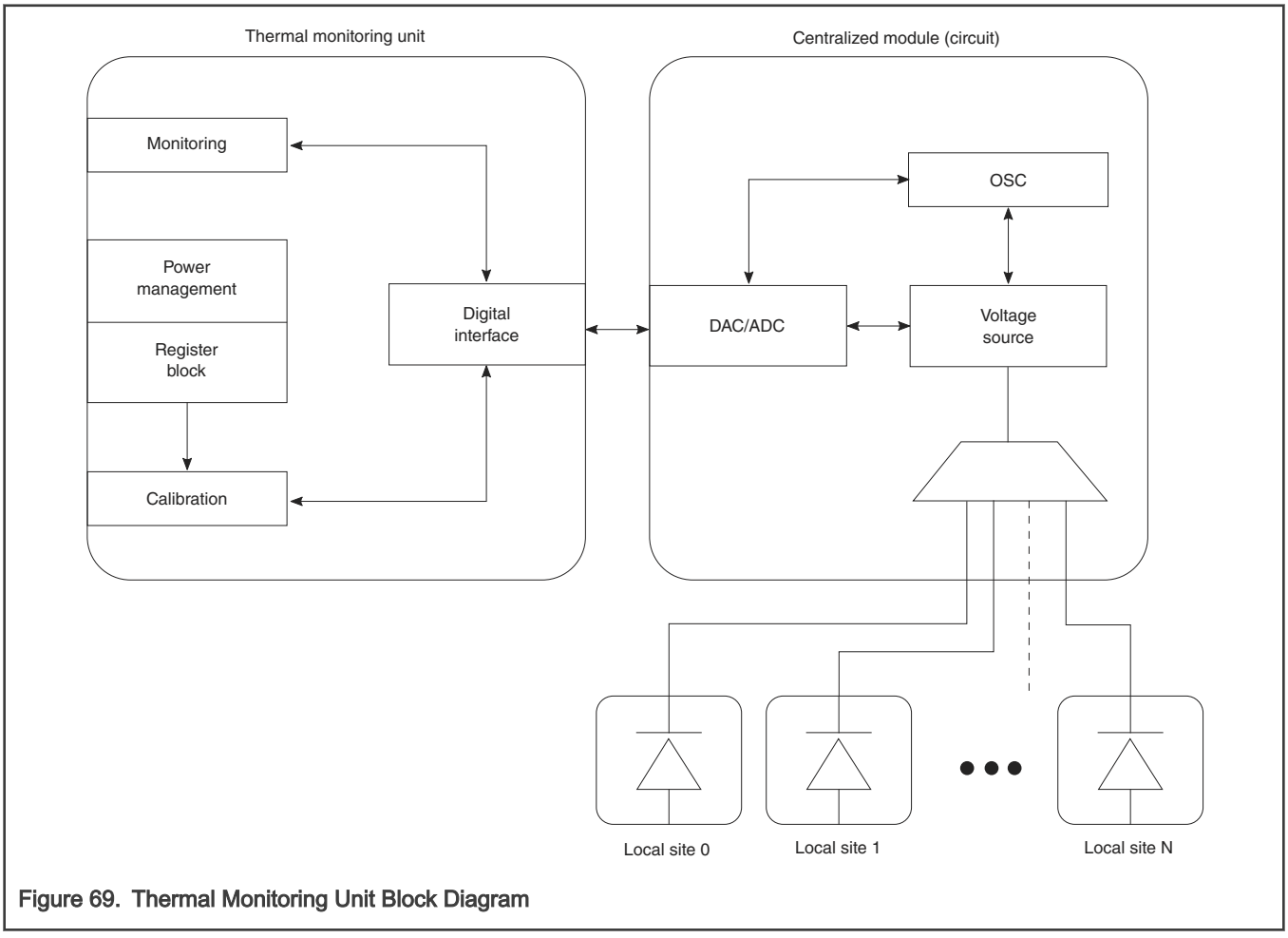


Figure 69. Thermal Monitoring Unit Block Diagram

22.3.2 Features

The temperature management unit features:

- Temperature measurement range 0-125°C.
- Calibration
  - Calibration table loaded from boot code ROM using pre-boot loader *or* initialization of calibration table through software
- Monitoring
  - Single-, or multi-site monitoring
  - Programmable monitoring interval
  - Out-of-range indication
  - High/low temperature range monitoring
  - Immediate and average temperature monitoring
  - Average temperature monitoring programmable low-pass filtering
  - Programmable monitoring thresholds for normal and critical alarm
- Reporting
  - Immediate and average temperature reporting for all monitor sites

22.3.3 Modes of Operation

The TMU has one mode of operation:

- Monitoring

The mode register monitoring enable bit, TMR[ME], determines if the unit is in active monitoring mode or in power saving mode.

The table below describes bit settings required for each TMU mode of operation.

Table 93. TMU Mode Bit Setting

Modes with Features	TMR[ME]
Monitoring mode disabled	0
Monitoring mode enabled	1

22.4 TMU register descriptions

The table shows the memory map for management of the TMU resources.

22.4.1 TMU Memory map

TMU base address: 1F8_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	TMU mode register (TMR)	32	RW	0000_0000h
4h	TMU status register (TSR)	32	RO	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
8h	TMU monitor temperature measurement interval register (TMTMIR)	32	RW	0000_0000h
20h	TMU interrupt enable register (TIER)	32	RW	0000_0000h
24h	TMU interrupt detect register (TIDR)	32	W1C	0000_0000h
28h	TMU interrupt site capture register (TISCR)	32	RW	0000_0000h
2Ch	TMU interrupt critical site capture register (TICSCR)	32	RW	0000_0000h
40h	TMU monitor high temperature capture register (TMHTCR)	32	RO	0000_0000h
44h	TMU monitor low temperature capture register (TMLTCR)	32	RO	0000_0000h
50h	TMU monitor high temperature immediate threshold register (TMHTITR)	32	RW	0000_0000h
54h	TMU monitor high temperature average threshold register (TMHTATR)	32	RW	0000_0000h
58h	TMU monitor high temperature average critical threshold register (TMHTACTR)	32	RW	0000_0000h
80h	TMU temperature configuration register (TTCFGR)	32	RW	0000_0000h
84h	TMU sensor configuration register (TSCFGR)	32	RW	0000_0000h
100h	TMU report immediate temperature site register 0 (TRITSR0)	32	RO	0000_0000h
104h	TMU report average temperature site register 0 (TRATSR0)	32	RO	0000_0000h
110h	TMU report immediate temperature site register 1 (TRITSR1)	32	RO	0000_0000h
114h	TMU report average temperature site register 1 (TRATSR1)	32	RO	0000_0000h
F10h	TMU temperature range 0 control register (TTR0CR)	32	RW	000B_0000h
F14h	TMU temperature range 1 control register (TTR1CR)	32	RW	000A_0026h
F18h	TMU temperature range 2 control register (TTR2CR)	32	RW	0008_0048h
F1Ch	TMU temperature range 3 control register (TTR3CR)	32	RW	0007_0061h

22.4.2 TMU mode register (TMR)

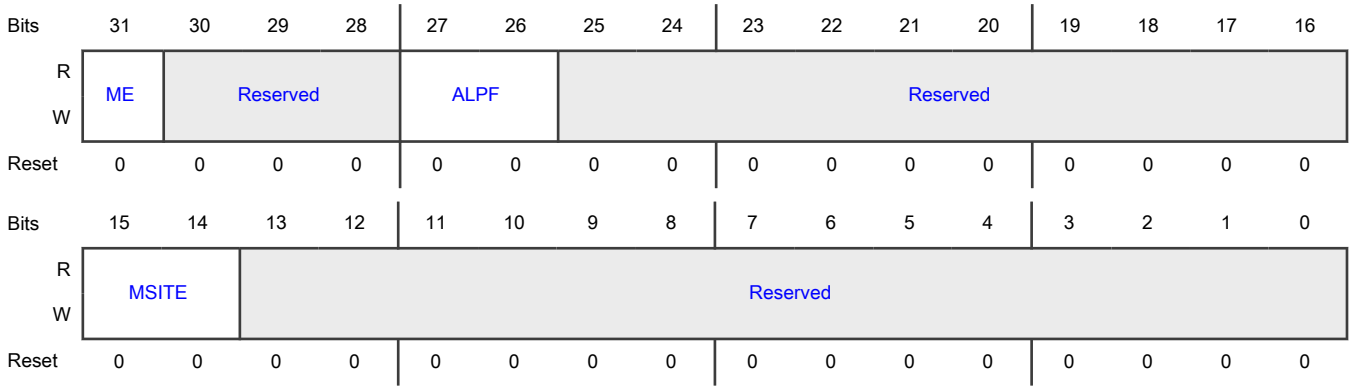
Offset

Register	Offset
TMR	0h

Function

The TMU mode register allows software to control the operation of the thermal monitoring.

Diagram



Fields

Field	Function
31 ME	Monitoring mode enable. 0 No monitoring. 1 Monitoring of sites as defined by field MSITE. Before enabling the TMU for monitoring, the TMU must be configured, see section Initialization Information. Failure to properly initialize the configuration table may result in boundedly undefined behavior.
30-28 —	Reserved.
27-26 ALPF	Average low pass filter setting. 00 1.0 01 0.5 10 0.25 11 0.125 The average temperature is calculated as: ALPF x Current_Temp + (1 - ALPF) x Average_Temp. If no previous (average) temperature is valid, current temperature is used. For proper operation, this field should only change when monitoring is disabled.
25-16 —	Reserved.
15-14 MSITE	Monitoring site select 0 - 1. By setting the select bit for a temperature sensor site, it is enabled and included in all monitoring functions. For proper operation, this field should only change when monitoring is disabled. If no site is selected, site 0 is monitored by default.
13-0 —	Reserved.

22.4.3 TMU status register (TSR)

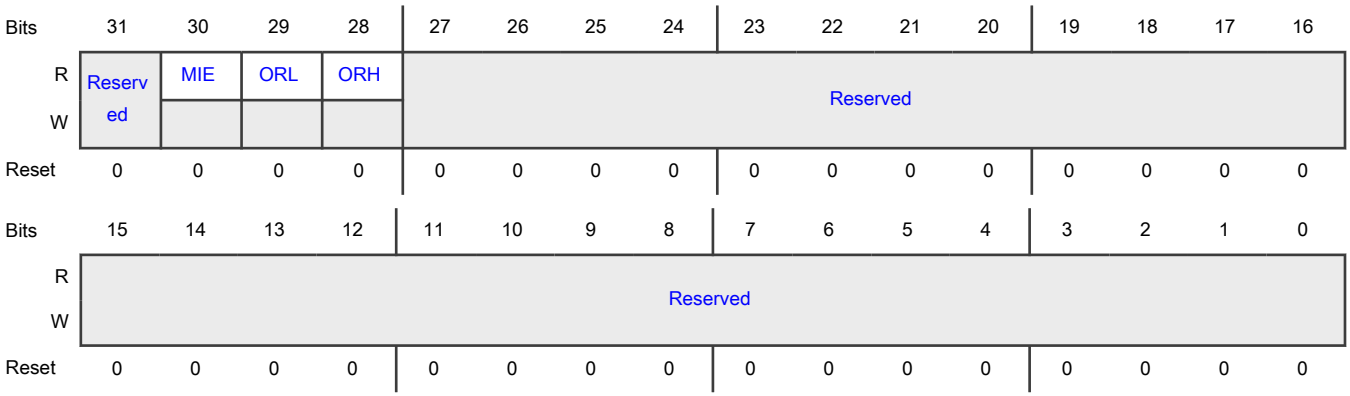
Offset

Register	Offset
TSR	4h

Function

The TMU status register reports the monitoring and calibration status during operation.

Diagram



Fields

Field	Function
31 —	Reserved.
30 MIE	Monitoring interval exceeded. 0 Monitoring interval not exceeded. 1 Monitoring interval exceeded. The time required to perform measurement of all monitored sites has exceeded the monitoring interval as defined by TMTMIR. This bit will clear automatically when TMU monitoring is (re-)enabled or the monitoring interval register, TMTMIR, is written.
29 ORL	Out-of-range low temperature measurement detected. A temperature sensor detected a temperature reading below the lowest measurable temperature of 0 degrees Celsius. This bit will clear automatically when TMU monitoring is (re-)enabled.
28 ORH	Out-of-range high temperature measurement detected. A temperature sensor detected a temperature reading above the highest measurable temperature of 125 °C. This bit will clear automatically when TMU monitoring is (re-)enabled.
27-0 —	Reserved.

22.4.4 TMU monitor temperature measurement interval register (TMTMIR)

Offset

Register	Offset
TMTMIR	8h

Function

The TMU monitor temperature measurement interval register determines at what frequency temperature sensors are read. All enabled monitored sites are read once in the duration of the time interval. The status bit TSR[MIE] will be set if the temperature measument takes longer than the set interval. Software should consider increasing the interval or reducing the number of active sites if the interval is exceeded. Disabling the interval allows for continuous monitoring.

NOTE

The time it takes for one temperature measurement is dependent on the temperatures measured and mode settings, thus there is no fixed time advertised for a single temperature measurements.

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												TMI			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-4 —	Reserved.
3-0 TMI	Temperature monitoring interval in seconds. For proper operation, this field should only change when monitoring is disabled, TMR[ME]=0. For lower platform speeds, the time increases proportionally, while the opposite is true for higher platform speeds.

Setting	Platform Clock Frequency		
	333 MHz	400 MHz	667 MHz
0000	0.03 s	0.02 s	0.015 s
0001	0.06 s	0.04 s	0.03 s
0010	0.10 s	0.08 s	0.05 s

Field	Function

Setting	Platform Clock Frequency		
0011	0.20 s	0.17 s	0.10 s
0100	0.40 s	0.34 s	0.20 s
0101	0.80 s	0.67 s	0.40 s
0110	1.60 s	1.34 s	0.80 s
0111	3.2 s	2.7 s	1.6 s
1000	6.4 s	5.4 s	3.2 s
1001	12.8 s	10.7 s	6.4 s
1010	25.8 s	21.5 s	12.9 s
1011	51.6 s	42.9 s	25.8 s
1100	103 s	85.9 s	51.5 s
1101	206.0 s	171.8 s	103.0 s
1110	412.2 s	343.6 s	206.1 s
1111	Disabled		

22.4.5 TMU interrupt enable register (TIER)

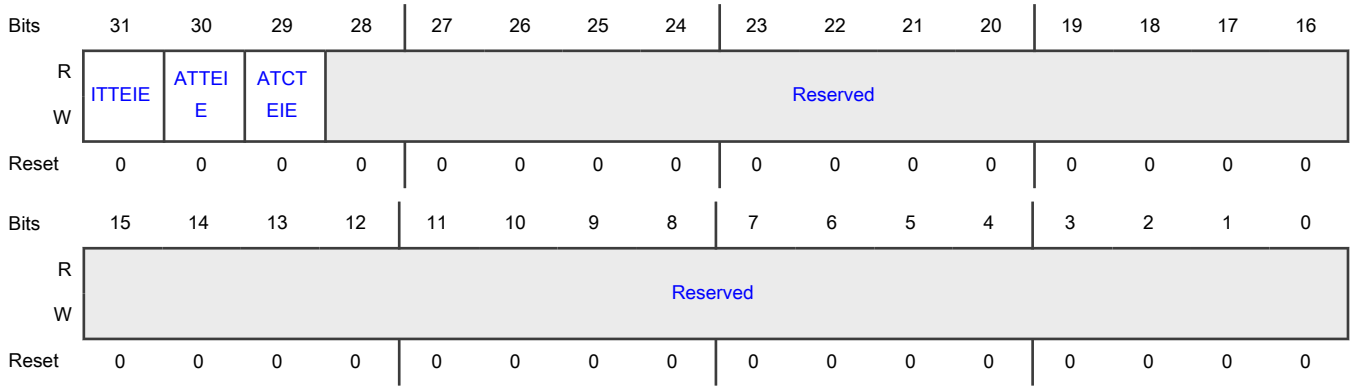
Offset

Register	Offset
TIER	20h

Function

The TMU interrupt enable register determines if a detected status condition should cause a system interrupt. A system interrupt occurs if a bit in this register is set and the corresponding bit in the interrupt detect register is also set. To clear the interrupt, write a 1 to the interrupt detect register.

Diagram



Fields

Field	Function
31 ITTEIE	Immediate temperature threshold exceeded interrupt enable. 0 Disabled. 1 Interrupt enabled. Generate an interrupt if TIDR[ITTE] is set.
30 ATTEIE	Average temperature threshold exceeded interrupt enable. 0 Disabled. 1 Interrupt enabled. Generate an interrupt if TIDR[ATTE] is set.
29 ATCTEIE	Average temperature critical threshold exceeded interrupt enable. 0 Disabled. 1 Interrupt enabled. Generate an interrupt if TIDR[ATCTE] is set.
28-0 —	Reserved.

22.4.6 TMU interrupt detect register (TIDR)

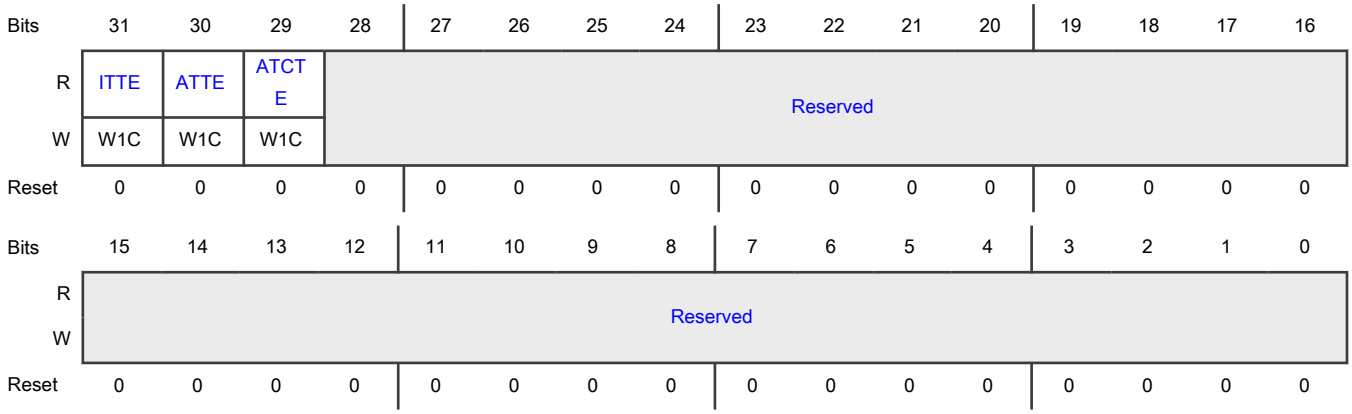
Offset

Register	Offset
TIDR	24h

Function

The TMU interrupt detect register indicates if an status condition was detected that could generate an interrupt. Write 1 to clear the detected condition and the interrupt, if enabled.

Diagram



Fields

Field	Function
31 ITTE	Immediate temperature threshold exceeded. Write 1 to clear. 0 No threshold exceeded. 1 Immediate temperature threshold, as defined by TMHTITR, has been exceeded by one or more monitored sites. This includes an out-of-range measured temperature above 125 °C. The sites which has exceeded the threshold are captured in TISCR[ISITE].
30 ATTE	Average temperature threshold exceeded. Write 1 to clear. 0 No threshold exceeded. 1 Average temperature threshold, as defined by TMHTATR, has been exceeded by one or more monitored sites. The sites which has exceeded the threshold are captured in TISCR[ASITE].
29 ATCTE	Average temperature critical threshold exceeded. Write 1 to clear. 0 No threshold exceeded. 1 Average temperature critical threshold, as defined by TMHTACTR, has been exceeded by one or more monitored sites. The sites which has exceeded the threshold are captured in TICSCR[CASITE].
28-0 —	Reserved.

22.4.7 TMU interrupt site capture register (TISCR)

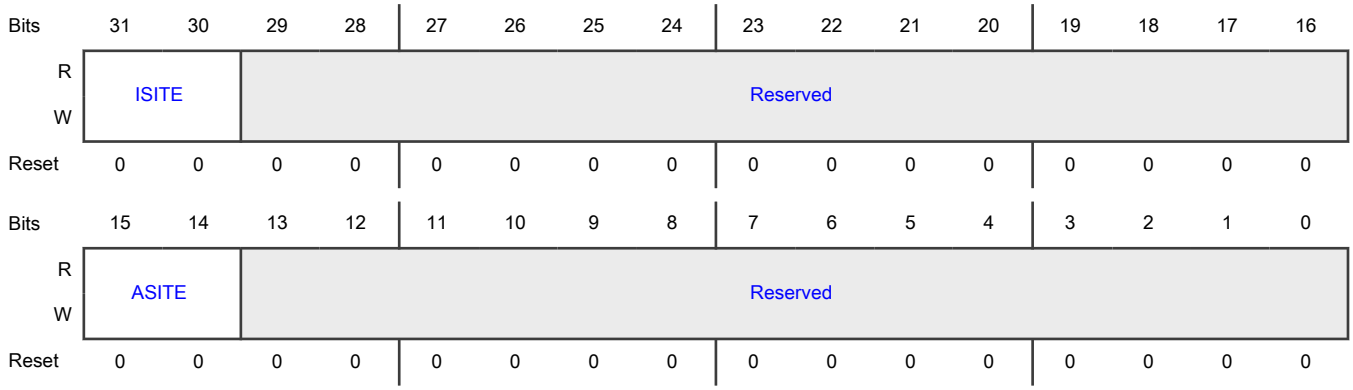
Offset

Register	Offset
TISCR	28h

Function

The TMU interrupt site capture register holds information about the temperature sensor site associated with a detected interrupt event.

Diagram



Fields

Field	Function
31-30 ISITE	Temperature sensor site associated with the setting of TIDR[ITTE]. This field has the same bit representation as TMR[MSITE]. Software should clear this field after handling the detected interrupt events ITTE.
29-16 —	Reserved.
15-14 ASITE	Temperature sensor site associated with the setting of TIDR[ATTE] . This field has the same bit representation as TMR[MSITE]. Software should clear this field after handling the detected interrupt event ATTE.
13-0 —	Reserved.

22.4.8 TMU interrupt critical site capture register (TICSCR)

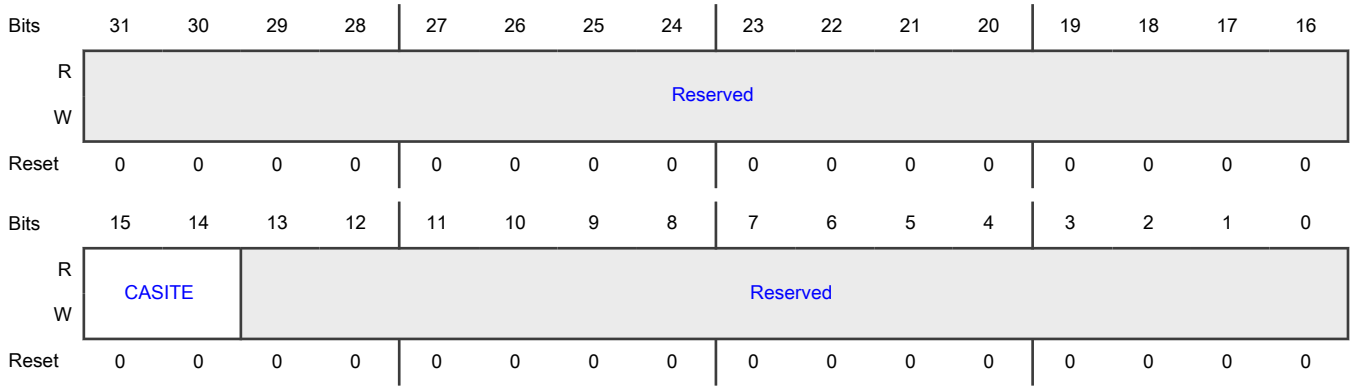
Offset

Register	Offset
TICSCR	2Ch

Function

The TMU interrupt critical site capture register holds information about the temperature sensor site associated with a detected critical interrupt event.

Diagram



Fields

Field	Function
31-16 —	Reserved.
15-14 CASITE	Temperature sensor site associated with the setting of TIDR[ATCTE] . This field has the same bit representation as TMR[MSITE]. Software should clear this field after handling the detected critical interrupt event ATCTE.
13-0 —	Reserved.

22.4.9 TMU monitor high temperature capture register (TMHTCR)

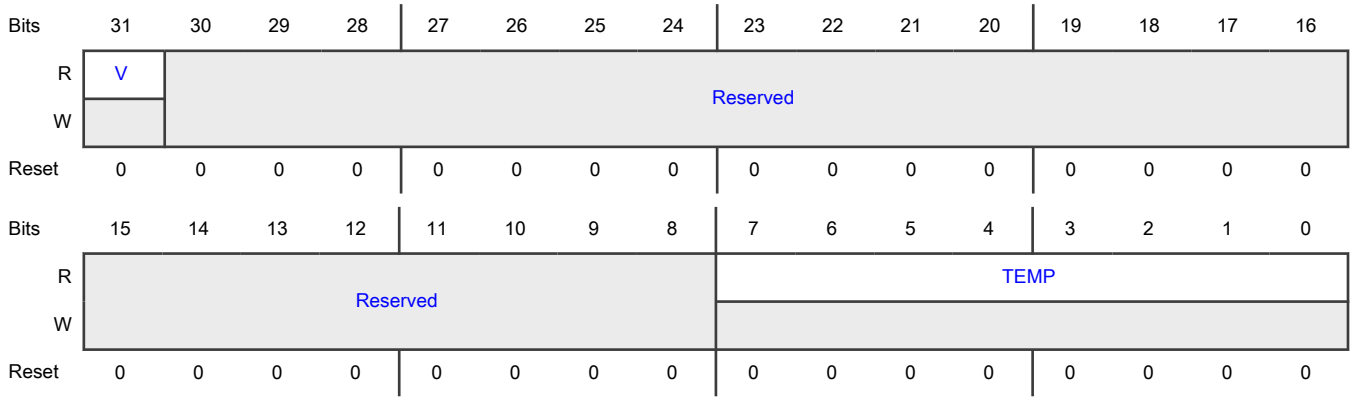
Offset

Register	Offset
TMHTCR	40h

Function

This TMU monitor register captures and record the highest temperature reached for any one enabled monitored site within temperature sensor range.

Diagram



Fields

Field	Function
31 V	Valid reading. 0 Temperature reading is not valid due to no measured temperature within the sensor range of 0-125 °C for an enabled monitored site. 1 Temperature reading is valid. (Re-)enabling the TMU will automatically clear this bit and start a new search.
30-8 —	Reserved.
7-0 TEMP	Highest temperature recorded in degrees Celcius by any enabled monitored site. Valid when V=1. 0-125 °C Sensor range 126-255 °C Reserved

22.4.10 TMU monitor low temperature capture register (TMLTCR)

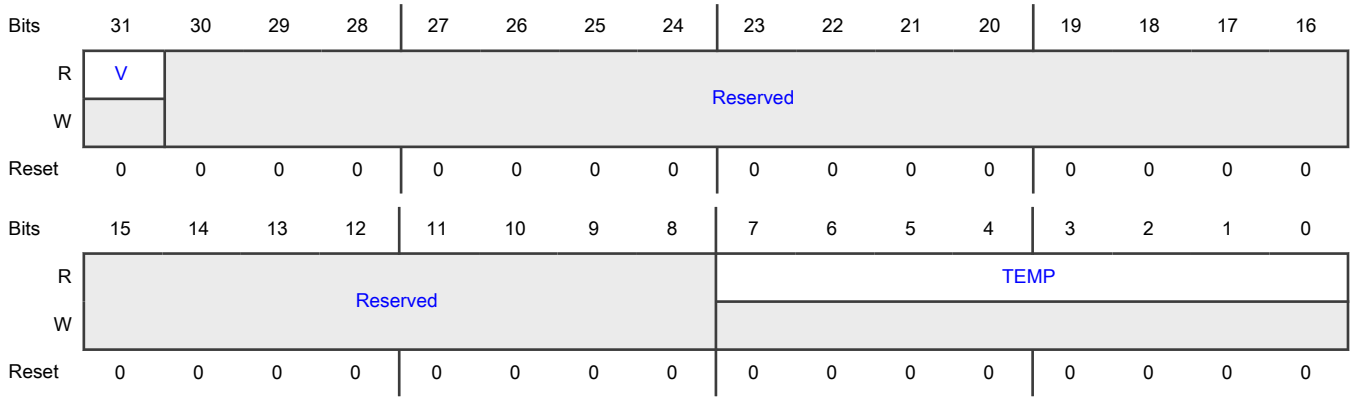
Offset

Register	Offset
TMLTCR	44h

Function

This TMU monitor register captures and record the lowest temperature reached for any one enabled monitored site within temperature sensor range.

Diagram



Fields

Field	Function
31 V	Valid reading. 0 Temperature reading is not valid due to no measured temperature within the sensor range of 0-125 °C for an enabled monitored site. 1 Temperature reading is valid. (Re-)enabling the TMU will automatically clear this bit and start a new search.
30-8 —	Reserved.
7-0 TEMP	Lowest temperature recorded in degrees Celcius by any enabled monitored site. Valid when V=1. 0-125 °C Sensor range 126-255 °C Reserved

22.4.11 TMU monitor high temperature immediate threshold register (TMHTITR)

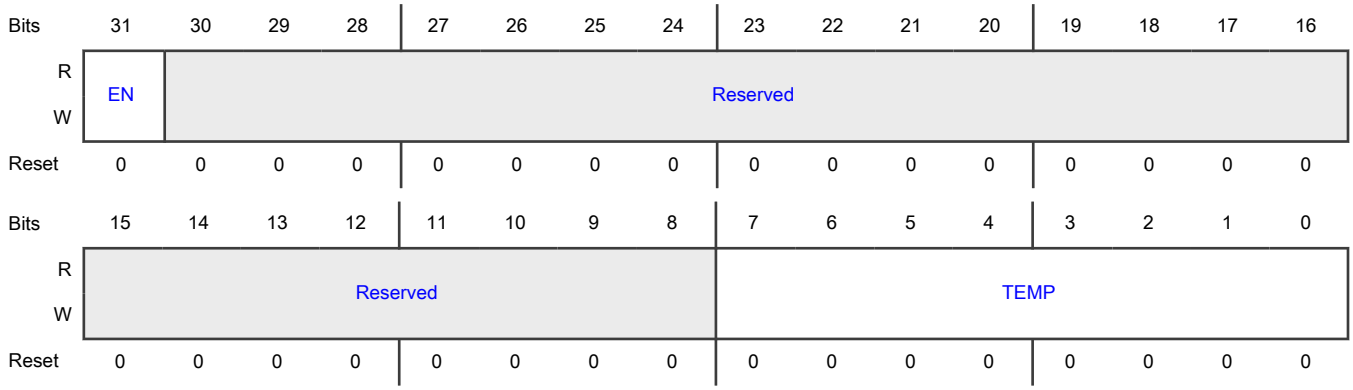
Offset

Register	Offset
TMHTITR	50h

Function

This TMU monitor register determines the high current temperature threshold for generating the TIDR[ITTE] event.

Diagram



Fields

Field	Function
31 EN	Enable threshold. 0 Disabled. 1 Threshold enabled.
30-8 —	Reserved.
7-0 TEMP	High temperature immediate threshold value. Determines the current upper temperature threshold, for any enabled monitored site, that if exceeded will cause TIDR[ITTE] to be set when EN=1. 0-125 °C Sensor range 126-255 °C Reserved

22.4.12 TMU monitor high temperature average threshold register (TMHTATR)

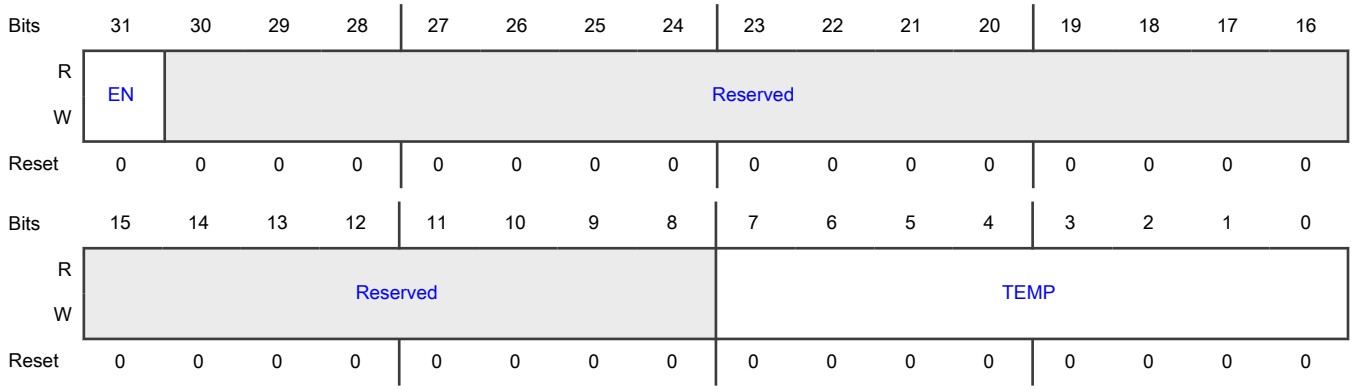
Offset

Register	Offset
TMHTATR	54h

Function

This TMU monitor register determines the high average temperature threshold for generating the TIDR[ATTE] event. The low-pass filter setting, TMR[ALPF], determines the function for calculating average temperature.

Diagram



Fields

Field	Function
31 EN	Enable threshold. 0 Disabled. 1 Threshold enabled.
30-8 —	Reserved.
7-0 TEMP	High temperature average threshold value. Determines the average upper temperature threshold, for any enabled monitor site, that if exceeded will cause TIDR[ATTE] to be set when EN=1. 0-125 °C Sensor range 126-255 °C Reserved

22.4.13 TMU monitor high temperature average critical threshold register (TMHTACTR)

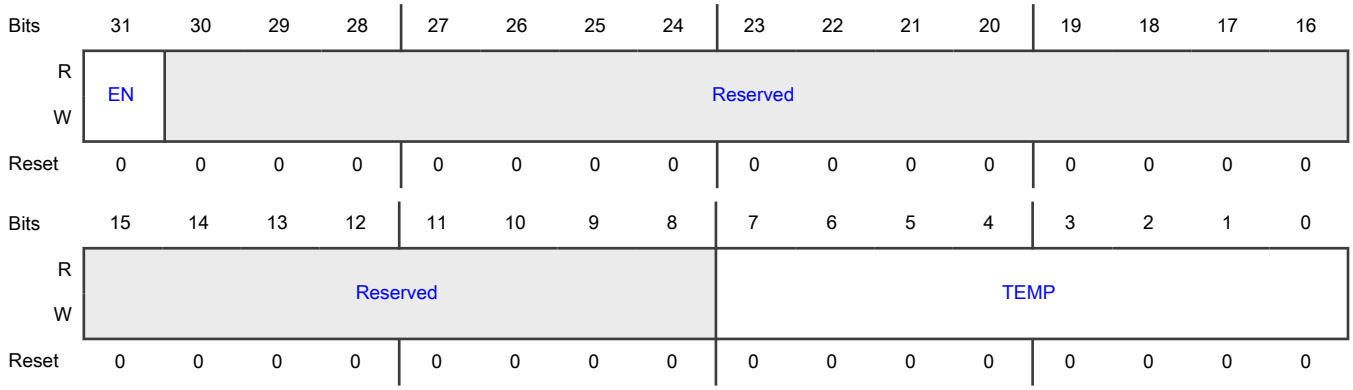
Offset

Register	Offset
TMHTACTR	58h

Function

This TMU monitor register determines the high average critical temperature threshold for generating the TIDR[ATCTE] event. The low-pass filter setting, TMR[ALPF], determines the function for calculating average temperature.

Diagram



Fields

Field	Function
31 EN	Enable threshold. 0 Disabled. 1 Threshold enabled.
30-8 —	Reserved.
7-0 TEMP	High temperature average critical threshold value. Determines the average upper critical temperature threshold, for any enabled monitored site, that if exceeded will cause TIDR[ATCTE] to be set when EN=1. 0-125 °C Sensor range 126-255 °C Reserved

22.4.14 TMU temperature configuration register (TTCFGR)

Offset

Register	Offset
TTCFGR	80h

Function

The TMU temperature configuration register, in conjunction with the sensor configuration register, is used to initialize the internal sensor translation table used during monitoring. This register pair defines indirect access to the table. See the section Initialization Information.

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

Fields

Field	Function
31-0 DATA	Sensor data.

22.4.15 TMU sensor configuration register (TSCFGR)

Offset

Register	Offset
TSCFGR	84h

Function

The TMU sensor configuration register, in conjunction with the temperature configuration register, is used to initialize the internal sensor translation table used during monitoring. This register pair defines indirect access to the table. Reading this register will return the data from the translation table as defined by TTCFGR. See the section Initialization Information.

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

Fields

Field	Function
31-0 DATA	Sensor data.

22.4.16 TMU report immediate temperature site register a (TRITSR0 - TRITSR1)

Offset

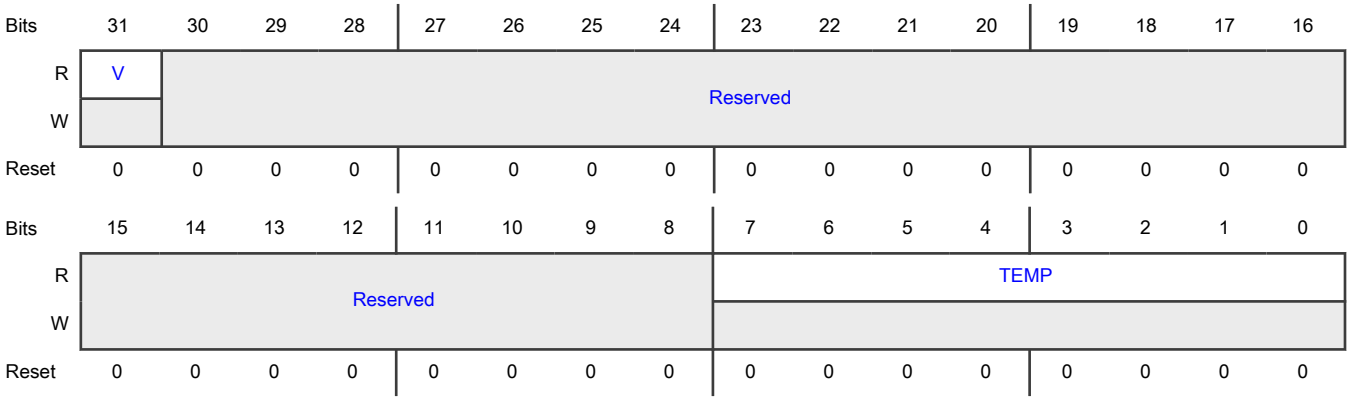
For a = 0 to 1:

Register	Offset
TRITSRa	100h + (a × 10h)

Function

This TMU report register returns the last measured temperature at site. The site must be part of the list of enabled monitored sites as defined by TMR[MSITE].

Diagram



Fields

Field	Function
31 V	Valid measured temperature. 0 Not valid. Temperature out of sensor range or first measurement still pending. 1 Valid.
30-8 —	Reserved.
7-0 TEMP	Last temperature reading at site when V=1.

22.4.17 TMU report average temperature site register a (TRATSR0 - TRATSR1)

Offset

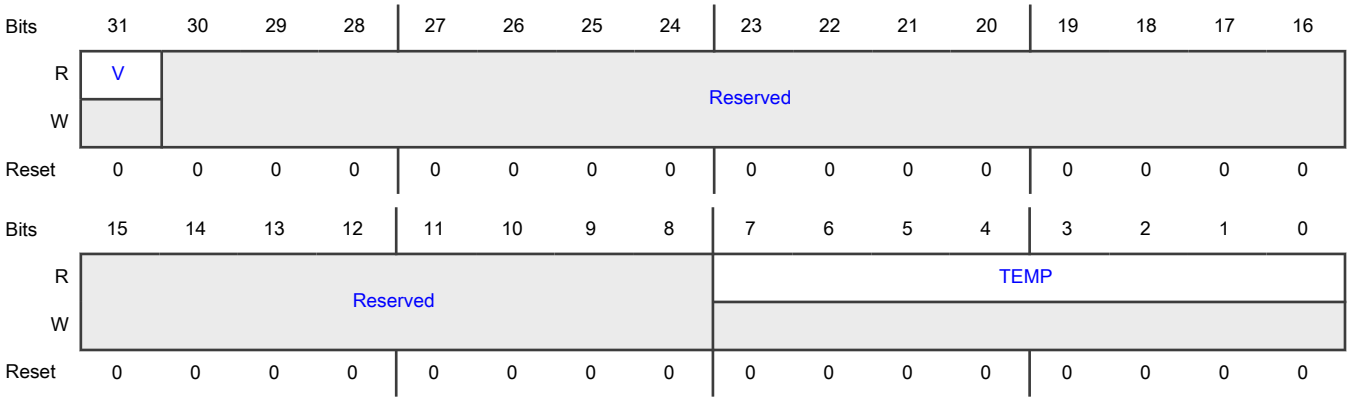
For a = 0 to 1:

Register	Offset
TRATSRa	104h + (a × 10h)

Function

This TMU report register returns the average measured temperature at site. The site must be part of the list of enabled monitored sites as defined by TMR[MSITE].

Diagram



Fields

Field	Function
31 V	Valid measured temperature. 0 Not valid. Temperature out of sensor range or first measurement still pending. 1 Valid.
30-8 —	Reserved.
7-0 TEMP	Average temperature reading at site when V=1.

22.4.18 TMU temperature range 0 control register (TTR0CR)

Offset

Register	Offset
TTR0CR	F10h

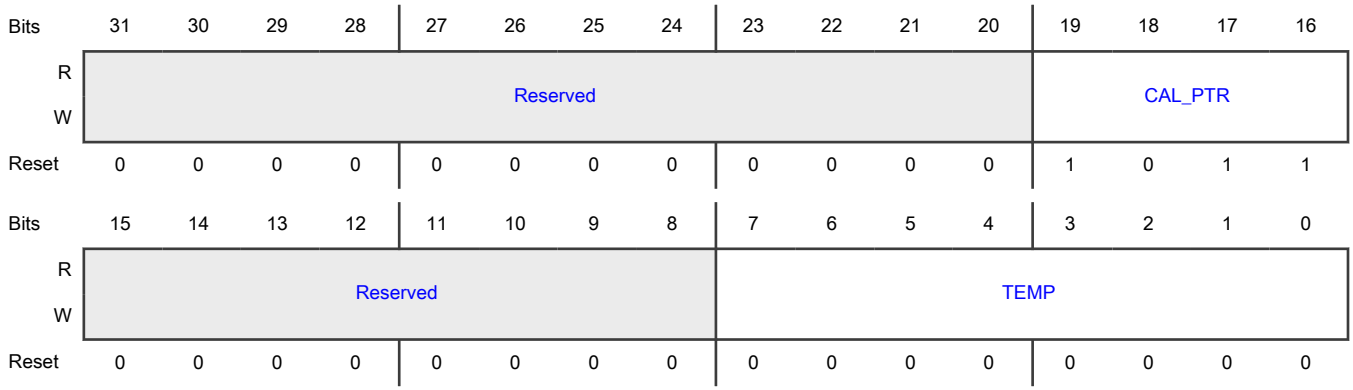
Function

The TMU temperature range control registers allow for defining the four temperature ranges that each sensor covers. A starting temperature range and number of calibration points, up to 16, is given. The temperature ranges may overlap to remove possible sample error, when searching for an accurate sensor reading. The default temperature calibration points are shown below.

Table 94. Default Temperature Configuration Points

Configuration Range	Starting Temperature (Celsius)	Temperature Configuration Points (Celsius)	Number of Points
0	0	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44	12
1	38	38, 42, 46, 50, 54, 58, 62, 66, 70, 74, 78	11
2	72	72, 76, 80, 84, 88, 92, 96, 100, 104	9
3	97	97, 101, 105, 109, 113, 117, 121, 125	8

Diagram



Fields

Field	Function
31-20 —	Reserved.
19-16 CAL_PTR	Temperature configuration points. 0000 Reserved 0001 2 points 0010 3 points ... 1111 16 points
15-8 —	Reserved.
7-0 TEMP	Starting temperature in Celsius for range.

22.4.19 TMU temperature range 1 control register (TTR1CR)

Offset

Register	Offset
TTR1CR	F14h

Function

The TMU temperature range control registers allow for defining the four temperature ranges that each sensor covers. A starting temperature range and number of calibration points, up to 16, is given. The temperature ranges may overlap to remove possible sample error, when searching for an accurate sensor reading. The default temperature calibration points are shown below.

Table 95. Default Temperature Configuration Points

Configuration Range	Starting Temperature (Celsius)	Temperature Configuration Points (Celsius)	Number of Points
0	0	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44	12
1	38	38, 42, 46, 50, 54, 58, 62, 66, 70, 74, 78	11
2	72	72, 76, 80, 84, 88, 92, 96, 100, 104	9
3	97	97, 101, 105, 109, 113, 117, 121, 125	8

Diagram

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

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

Fields

Field	Function
31-20 —	Reserved.
19-16 CAL_PTR	Temperature configuration points. 0000 Reserved 0001 2 points 0010 3 points

Table continues on the next page...

Table continued from the previous page...

Field	Function
	... 1111 16 points
15-8 —	Reserved.
7-0 TEMP	Starting temperature in Celsius for range.

22.4.20 TMU temperature range 2 control register (TTR2CR)

Offset

Register	Offset
TTR2CR	F18h

Function

The TMU temperature range control registers allow for defining the four temperature ranges that each sensor covers. A starting temperature range and number of calibration points, up to 16, is given. The temperature ranges may overlap to remove possible sample error, when searching for an accurate sensor reading. The default temperature calibration points are shown below.

Table 96. Default Temperature Configuration Points

Configuration Range	Starting Temperature (Celsius)	Temperature Configuration Points (Celsius)	Number of Points
0	0	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44	12
1	38	38, 42, 46, 50, 54, 58, 62, 66, 70, 74, 78	11
2	72	72, 76, 80, 84, 88, 92, 96, 100, 104	9
3	97	97, 101, 105, 109, 113, 117, 121, 125	8

Diagram

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

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

Fields

Field	Function
31-20 —	Reserved.
19-16 CAL_PTR	Temperature configuration points. 0000 Reserved 0001 2 points 0010 3 points ... 1111 16 points
15-8 —	Reserved.
7-0 TEMP	Starting temperature in Celsius for range.

22.4.21 TMU temperature range 3 control register (TTR3CR)

Offset

Register	Offset
TTR3CR	F1Ch

Function

The TMU temperature range control registers allow for defining the four temperature ranges that each sensor covers. A starting temperature range and number of calibration points, up to 16, is given. The temperature ranges may overlap to remove possible sample error, when searching for an accurate sensor reading. The default temperature calibration points are shown below.

Table 97. Default Temperature Configuration Points

Configuration Range	Starting Temperature (Celsius)	Temperature Configuration Points (Celsius)	Number of Points
0	0	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44	12
1	38	38, 42, 46, 50, 54, 58, 62, 66, 70, 74, 78	11
2	72	72, 76, 80, 84, 88, 92, 96, 100, 104	9
3	97	97, 101, 105, 109, 113, 117, 121, 125	8

Diagram

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

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

Fields

Field	Function
31-20 —	Reserved.
19-16 CAL_PTR	Temperature configuration points. 0000 Reserved 0001 2 points 0010 3 points ... 1111 16 points
15-8 —	Reserved.
7-0 TEMP	Starting temperature in Celsius for range.

22.5 Functional Description

The following sections describe the functionality of the TMU in details.

22.5.1 Monitoring

Monitoring is the process of reading enabled temperature sensor sites on-chip at regular intervals and taking appropriate actions, such as alarming the user when the temperature exceeds a programmed temperature threshold.

During monitoring, all enabled temperature sensor sites are periodically measured for temperature starting with site 0 and ending with site 1. The interval at which the sensors are read is set in TMTMIR and should reflect the maximum interval time required to accurately capture temperature changes. If the measurement interval has not expired after the last active site has been read, the sensor logic enters low-power mode. If the interval has expired when the last active site is read, the measurement interval exceeded bit, TSR[MIE], is set and the next active site is read immediately. If the interval has been exceeded, user may opt to reduce number of sites monitored or increase the interval, if possible.

For each site the current and average temperature is logged. The average temperature is calculated based on the low-pass filter function in the mode register. If any of the set temperature threshold registers are exceeded, the corresponding interrupt detect bit is set in TIDR. Interrupts are enabled through the interrupt enable register, TIER.

Process for enabling monitoring mode:

1. Clear the interrupt detect register, TIDR.
2. Clear interrupt site capture register (TISCR) and interrupt critical site capture register (TICSCR).
3. Enable interrupt handling by setting the appropriate bits in TIER.
4. Set the temperature threshold registers TMHTITR, TMHTATR and TMHTACTR.
5. Set the monitoring interval register, TMTMIR.
6. Enable monitor mode by setting TMR[ME]=1. Sites to monitor are controlled by setting TMR[MSITE]. There should be at least one active site enabled. Set other mode control bits as needed.
7. If the monitoring interval is too short as indicated by TSR[MIE], the interval may need to be increased or number of sites reduced.

22.5.2 Reporting

The TMU can directly report the current and average temperature for a particular temperature sensor site during monitoring mode by reading one of the report registers per site. The report uses the last measurement done by the monitoring process and requires a site to be actively monitored for accurate reading. Reading a site which has last measured an invalid temperature outside the sensor range of 0-125 degrees Celsius, will have the valid bit cleared.

If monitoring is disabled, the last temperature measurement remains for the site(s) previously monitored and can still be read using the report registers knowing that the temperature reported is no longer accurate. This method can be used to capture the temperature at multiple sites in time, but does not allow for continuous monitoring.

Chapter 23
UART

23.1 The UART module as implemented on the chip

The chip implements a 62550 compatible 2-pin UART for debug console interface. The Serial Input and Serial Output signals are multiplexed on SPI chip selects 1 and 2 respectively.

23.2 Overview

This chapter describes the universal asynchronous receiver/transmitters (UART). It describes the functional operation, the initialization sequence, and the programming details for the UART registers and features.The UART is clocked by the platform clock. The UART programming model is compatible with the PC16552D.

The UART interface is point to point, meaning that only two UART devicedsscs are attached to the connecting signals. As shown in the figure below, each UART module consists of the following:

- Receive and transmit buffers
- 16-bit counter for baud rate generation
- Interrupt control logic

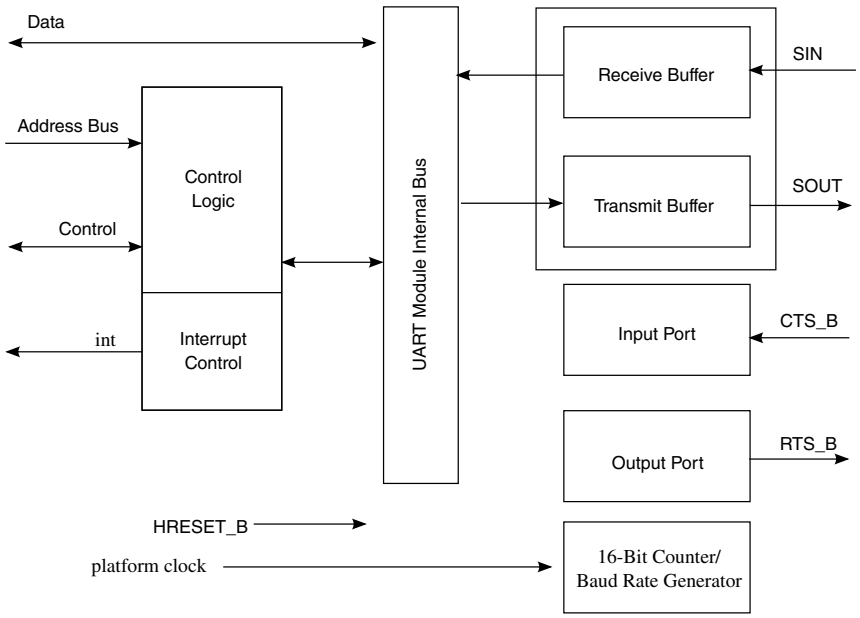


Figure 70. UART block diagram

23.2.1 Features

The UART includes these distinctive features:

- Programming model compatible with original PC16450 UART and PC16550D (improved version of PC16450 that also operates in FIFO mode)
- PC16450 register reset values
- Configurable FIFO mode for both transmitter and receiver, providing 64-byte FIFOs

- Serial data encapsulation and decapsulation with standard asynchronous communication bits (START, STOP, and parity)
- Maskable transmit, receive, line status, and modem status interrupts
- Software-programmable baud generators that divide the platform clock by 1 to (2^{16} - 1) and generate a 16x clock for the transmitter and receiver engines
- Auto flow for clear to send (CTS_B) and ready to send (RTS_B) modem control functions
- Software-selectable serial interface data format (data length, parity, 1/1.5/2 STOP bit, baud rate)
- Line and modem status registers
- Line-break detection and generation
- Internal diagnostic support and break functions
- Prioritized interrupt reporting
- Overrun, parity, and framing error detection

23.2.2 Modes of operation

The communication channel provides a asynchronous receiver and transmitter using an operating frequency derived from the platform clock.

The transmitter accepts parallel data from a write to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register of the transmitter FIFO. The transmitter converts the data to a serial bit stream inserting the appropriate start, stop, and optional parity bits. Finally, it outputs a composite serial data stream on the channel transmitter serial data output signal (SOUT). The transmitter status may be polled or interrupt driven.

The receiver accepts serial data bits on the channel receiver serial data input signal (SIN), converts it to parallel format, checks for a start bit, parity (if any), stop bits, and transfers the assembled character (with start, stop, parity bits removed) from the receiver buffer (or FIFO) in response to a read of the UART's receiver buffer register (URBR). The receiver status may be polled or interrupt driven.

23.3 UART external signal descriptions

The UART signals are described in the table below.

NOTE

Although the actual device signal names are prepended with the UART_ prefix as shown in the table, the abbreviated signal names are often used throughout this chapter.

Table 98. UART signals-detailed signal descriptions

Signal	I/O	Description	
UART*n*_SIN	I	Serial data in. Data is received on the receivers of UART*n* through the respective serial data input signal, with the least-significant bit received first.	
		State meaning	Asserted/Negated-Represents the data being received on the UART interface.
		Timing	Assertion/Negation-An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to sample the data on SIN.
UART*n*_SOUT	O	Serial data out. The serial data output signals for the UART*n* are set ('mark' condition) when the transmitter is disabled or idle. Data is shifted out on these signals, with the least significant bit transmitted first.	

Table continues on the next page...

Table 98. UART signals-detailed signal descriptions (continued)

Signal	I/O	Description	
		State meaning	Asserted/Negated-Represents the data being transmitted on the respective UART interface.
		Timing	Assertion/Negation- An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to update and drive the data on SOUT.

23.4 UART register descriptions

The table below lists the UART registers and their offsets. It lists the address, name, and a cross-reference to the complete description of each register.

All the UART registers are one-byte wide. Reads and writes to these registers must be byte-wide operations. The table below provides a register summary with references to the section and page that contains detailed information about each register. Undefined byte address spaces within offset 0x000-0xFFF are reserved.

23.4.1 UART Memory map

UART base address: 21C_0000h

Offset	Register	Width (In bits)	Access	Reset value
500h	UART divisor least significant byte register (UDLB1)	8	RW	00h
500h	UART receiver buffer register (URBR1)	8	RO	00h
500h	UART transmitter holding register (UTHR1)	8	WO	00h
501h	UART divisor most significant byte register (UDMB1)	8	RW	00h
501h	UART interrupt enable register (UIER1)	8	RW	00h
502h	UART alternate function register (UAFR1)	8	RW	00h
502h	UART FIFO control register (UFCR1)	8	WO	00h
502h	UART interrupt ID register (UIIR1)	8	RO	01h
503h	UART line control register (ULCR1)	8	RW	00h
504h	UART modem control register 1 (UMCR1)	8	RW	00h
505h	UART line status register (ULSR1)	8	RO	60h
507h	UART scratch register (USCR1)	8	RW	00h
510h	UART DMA status register (UDSR1)	8	RO	01h

23.4.2 UART divisor least significant byte register (UDLB1)

Offset

Register	Offset
UDLB1	500h

Function

This register is accessible when ULCR[DLAB] = 1.

The divisor least significant byte register (UDLB) is concatenated with the divisor most significant byte register (UDMB) to create the divisor used to divide the input clock into the UART. The output frequency of the baud generator is 16 times the baud rate; therefore the desired baud rate = platform clock frequency ÷ (16 x [UDMB||UDLB]). Equivalently, [UDMB||UDLB:0b0000] = platform clock frequency ÷ desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in the table below.

The following table shows examples of baud rate generation based on common input clock frequencies. Many other target baud rates are also possible.

NOTE

Because only integer values can be used as divisors, the actual baud rate differs slightly from the desired (target) baud rate; for this reason, both target and actual baud rates are given, along with the percentage of error.

Table 99. Baud Rate Examples

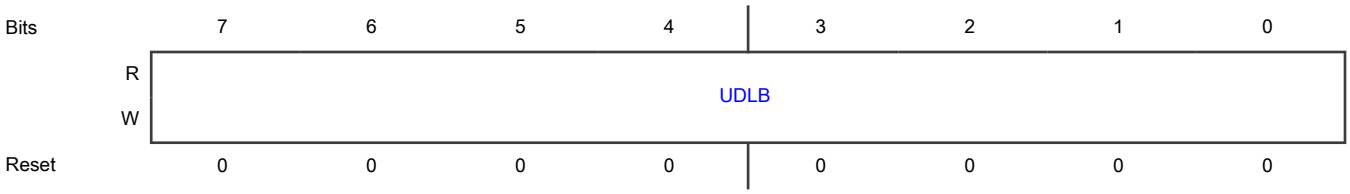
Target Baud Rate (Decimal)	Divisor		Platform Clock Frequency (MHz)	Actual Baud Rate (Decimal)	Percent Error (Decimal)
	Decimal	Hex			
9,600	1736	6C8	266	9600.61444	0.0064
19,200	868	364	266	19,201.22888	0.0064
38,400	434	1B2	266	38,402.45776	0.0064
57,600	289	121	266	57,670.12673	0.1217
115,200	145	91	266	114,942.52844	0.2235
230,400	72	48	266	231,481.48090	0.4694
9,600	2170	87A	333	9600.61444	0.0064
19,200	1085	43D	333	19,201.22888	0.0064
38,400	543	21F	333	38,367.09638	0.0858
57,600	362	16A	333	57,550.64457	0.0857
115,200	181	B5	333	115,101.28913	0.0857
230,400	90	5A	333	231,481.48148	0.4694

Table continues on the next page...

Table 99. Baud Rate Examples (continued)

Target Baud Rate (Decimal)	Divisor		Platform Clock Frequency (MHz)	Actual Baud Rate (Decimal)	Percent Error (Decimal)
	Decimal	Hex			
9,600	3472	D90	533	9600.61444	0.0064
19,200	1736	6C8	533	19,201.22888	0.0064
38,400	868	364	533	38,402.45776	0.0064
57,600	579	243	533	57,570.52389	0.0512
115,200	289	121	533	115,340.25375	0.1217
230,400	145	91	533	229,885.05747	0.2235

Diagram



Fields

Field	Function
7-0 UDLB	Divisor least significant byte. This is concatenated with UDMB.

23.4.3 UART receiver buffer register (URBR1)

Offset

Register	Offset
URBR1	500h

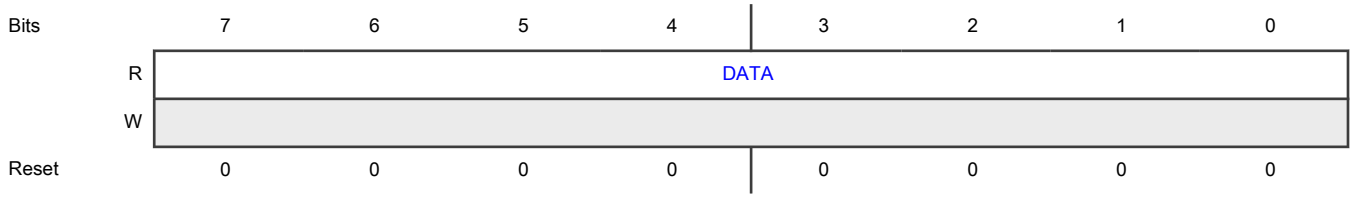
Function

This register is accessible when ULCR[DLAB] = 0.

These registers contain the data received from the transmitter on the UART buses. In FIFO mode, when read, they return the first byte received. For FIFO status information, refer to the UDSR[RXRDY] description.

Except for the case when there is an overrun, URBR returns the data in the order it was received from the transmitter. Refer to the ULSR[OE] description, #unique_788/unique_788_Connect_42_ULSR1 . Note that these registers have same offset as the UTHRs.

Diagram



Fields

Field	Function
7-0 DATA	Data received from the transmitter on the UART bus (read only)

23.4.4 UART transmitter holding register (UTHR1)

Offset

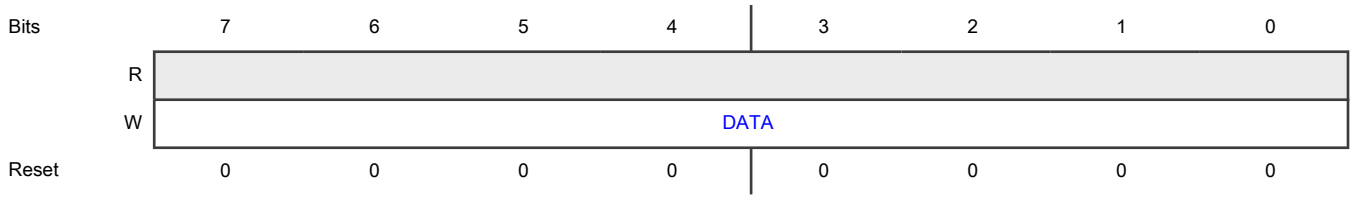
Register	Offset
UTHR1	500h

Function

This register is accessible when ULCR[DLAB] = 0.

A write to these 8-bit registers causes the UART devices to transfer 5-8 data bits on the UART bus in the format set up in the ULCR (line control register). In FIFO mode, data written to UTHR is placed into the FIFO. The data written to UTHR is the data sent onto the UART bus, and the first byte written to UTHR is the first byte onto the bus. UDSR[TXRDY_B] indicates when the FIFO is full. See #unique_788/unique_788_Connect_42_UDSR1 for more details.

Diagram



Fields

Field	Function
7-0 DATA	Data that is written to UTHR (write only)

23.4.5 UART divisor most significant byte register (UDMB1)

Offset

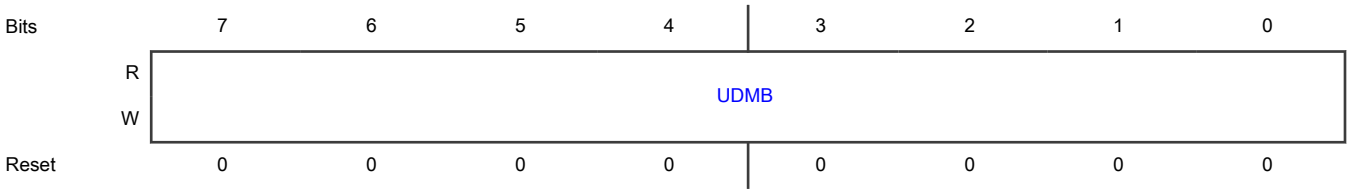
Register	Offset
UDMB1	501h

Function

This register is accessible when ULCR[DLAB] = 1.

The divisor least significant byte register (UDLB) is concatenated with the divisor most significant byte register (UDMB) to create the divisor used to divide the input clock into the UART. The output frequency of the baud generator is 16 times the baud rate; therefore the desired baud rate = platform clock frequency ÷ (16 x [UDMB||UDLB]). Equivalently, [UDMB||UDLB:0b0000] = platform clock frequency ÷ desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in #unique_788/unique_788_Connect_42_UDLB1.

Diagram



Fields

Field	Function
7-0 UDMB	Divisor most significant byte

23.4.6 UART interrupt enable register (UIER1)

Offset

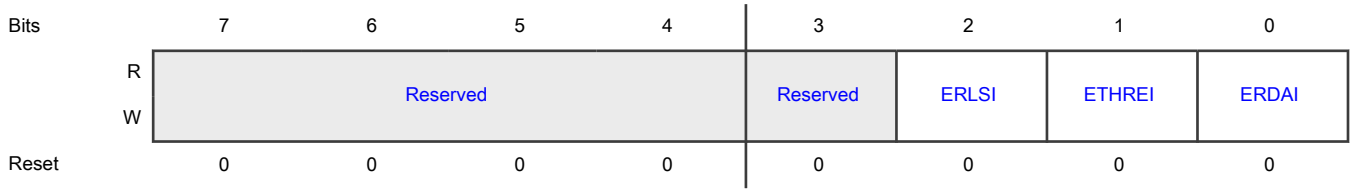
Register	Offset
UIER1	501h

Function

This register is accessible when ULCR[DLAB] = 0.

The UIER gives the user the ability to mask specific UART interrupts to the programmable interrupt controller (PIC).

Diagram



Fields

Field	Function
7-4 —	Reserved.
3 Resreved	Reserved
2 ERLSI	Enable receiver line status interrupt. 0b - Mask interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set 1b - Enable and assert interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set
1 ETHREI	Enable transmitter holding register empty interrupt. 0b - Mask interrupt when ULSR[THRE] is set 1b - Enable and assert interrupts when ULSR[THRE] is set
0 ERDAI	Enable received data available interrupt. 0b - Mask interrupt when new receive data is available or receive data time out has occurred 1b - Enable and assert interrupts when a new data character is received from the external device and/or a time-out interrupt occurs in the FIFO mode

23.4.7 UART alternate function register (UAFR1)

Offset

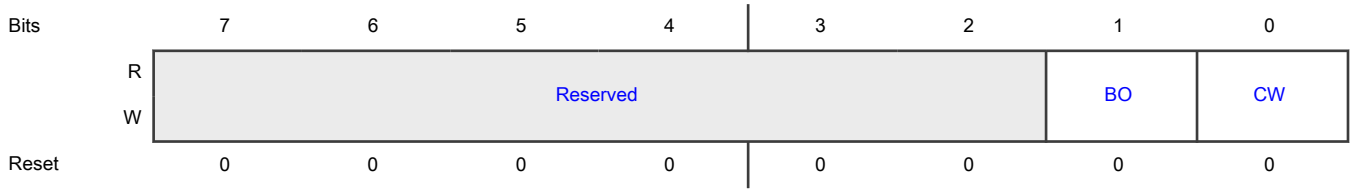
Register	Offset
UAFR1	502h

Function

This register is accessible when ULCR[DLAB] = 1.

The UAFRs give software the ability to gate off the baud clock and write to each UARTn registers simultaneously with the same write operation.

Diagram



Fields

Field	Function
7-2 —	Reserved.
1 BO	Baud clock select. 0b - The baud clock is not gated off. 1b - The baud clock is gated off.
0 CW	Concurrent write enable. 0b - Disables writing to each UART*n* 1b - Enables concurrent writes to corresponding UART registers. A write to a register in UART*n* is also a write to the corresponding register in UART*n+1* and vice versa for each UART where *n* refers to the number of UART controller. The user needs to ensure that the ULCR[DLAB] of each UART is in the same state before executing a concurrent write to register addresses 0x*m*00, 0x*m*01 and 0x*m*02, where *m* is the base address of the corresponding UART.

23.4.8 UART FIFO control register (UFCR1)

Offset

Register	Offset
UFCR1	502h

Function

This register is accessible when ULCR[DLAB] = 0.

The UFCR, a write-only register, is used to enable and clear the receiver and transmitter FIFOs, set a receiver FIFO trigger level to control the received data available interrupt, and select the type of DMA signaling.

When the UFCR bits are written, the FIFO enable bit must also be set or else the UFCR bits are not programmed. When changing from FIFO mode to 16450 mode (non-FIFO mode) and vice versa, data is automatically cleared from the FIFOs.

After all the bytes in the receiver FIFO are cleared, the receiver internal shift register is not cleared. Similarly, the bytes are cleared in the transmitter FIFO, but the transmitter internal shift register is not cleared. Both TFR and RFR are self-clearing bits.

Diagram

Bits	7	6	5	4	3	2	1	0
R				Reserved				
W	RTL		EN64		DMS	TFR	RFR	FEN
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7-6 RTL	Receiver trigger level. A received data available interrupt occurs when UIER[ERDAI] is set and the number of bytes in the receiver FIFO equals the designated interrupt trigger level as follows: 00b - 1 byte, if EN64 1 byte 01b - 4 bytes, if EN64 16 bytes 10b - 8 bytes, if EN64 32 bytes 11b - 14 bytes, if EN64 56 bytes
5 EN64	Enable 64-byte FIFO 0b - Disables the 64-byte FIFOs 1b - Enables the 64-byte FIFOs
4 —	Reserved
3 DMS	DMA mode select. 0b - UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 0. 1b - UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 1 if UFCR[FEN] = 1.
2 TFR	Transmitter FIFO reset 0b - No action 1b - Clears all bytes in the transmitter FIFO and resets the FIFO counter/pointer to 0
1 RFR	Receiver FIFO reset 0b - No action 1b - Clears all bytes in the receiver FIFO and resets the FIFO counter/pointer to 0
0 FEN	FIFO enable 0b - FIFOs are disabled and cleared 1b - Enables the transmitter and receiver FIFOs

23.4.9 UART interrupt ID register (UIIR1)

Offset

Register	Offset
UIIR1	502h

Function

This register is accessible when ULCR[DLAB] = 0.

The UIIRs indicate when an interrupt is pending from the corresponding UART and what type of interrupt is active. They also indicate if the FIFOs are enabled.

The UART prioritizes interrupts into four levels and records these in the corresponding UIIR. The four levels of interrupt conditions in order of priority are:

1. Receiver line status
2. Received data ready/character time-out
3. Transmitter holding register empty
4. Modem status

When the UIIR is read, the associated UART serial channel freezes all interrupts and indicates the highest priority pending interrupt. While this read transaction is occurring, the associated UART serial channel records new interrupts, but does not change the contents of UIIR until the read access is complete.

Table 100. UIIR IID Bits Summary

IID Bits IID[3-0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0b0001	-	-	-	-
0b0110	Highest	Receiver line status	Overrun error, parity error, framing error, or break interrupt	Read the line status register.
0b0100	Second	Received data available	Receiver data available or trigger level reached in FIFO mode	Read the receiver buffer register or interrupt is automatically reset if the number of bytes in the receiver FIFO drops below the trigger level.
0b1100	Second	Character time-out	No characters have been removed from or input to the receiver FIFO during the last 4 character times and there is at least one character in the receiver FIFO during this time.	Read the receiver buffer register.
0b0010	Third	UTHR empty	Transmitter holding register is empty	Read the UIIR or write to the UTHR.
0b0000	Fourth	Modem status	CTS_B input value changed since last read of UMSR	Read the UMSR.

Diagram

Bits	7	6	5	4	3	2	1	0
R	FE		FE64	Reserved	IID3	IID2	IID1	IID0
W								
Reset	0	0	0	0	0	0	0	1

Fields

Field	Function
7-6 FE	FIFOs enabled. Reflects the setting of UFCR[FEN]
5 FE64	64-byte FIFOs enabled. Reflects the setting of UFCR[EN64]. 0b - 64-byte FIFOs disabled 1b - 64-byte FIFOs enabled
4 —	Reserved
3 IID3	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in the table above. IID3 is set along with IID2 only when a timeout interrupt is pending for FIFO mode.
2 IID2	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in the table above.
1 IID1	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in the table above.
0 IID0	IID0 indicates when an interrupt is pending. 0b - The UART has an active interrupt ready to be serviced. 1b - No interrupt is pending.

23.4.10 UART line control register (ULCR1)

Offset

Register	Offset
ULCR1	503h

Function

This register is accessible when ULCR[DLAB] = x.

The ULCRs specify the data format for the UART bus and set the divisor latch access bit ULCR[DLAB], which controls the ability to access the divisor latch least and most significant bit registers and the alternate function register.

After initializing the ULCR, the software should not re-write the ULCR when valid transfers on the UART bus are active. The software should not re-write the ULCR until the last STOP bit has been received and there are no new characters being transferred on the bus.

The stick parity bit, ULCR[SP], assigns a set parity value for the parity bit time slot sent on the UART bus. The set value is defined as mark parity (logic 1) or space parity (logic 0). ULCR[PEN] and ULCR[EPS] help determine the set parity value. See the table below for more information. ULCR[NSTB], defines the number of STOP bits to be sent at the end of the data transfer. The receiver only checks the first STOP bit, regardless of the number of STOP bits selected. The word length select bits (1 and 0) define the number of data bits that are transmitted or received as a serial character. The word length does not include START, parity, and STOP bits.

Table 101. Parity Selection Using ULCR[PEN], ULCR[SP], and ULCR[EPS]

PEN	SP	EPS	Parity Selected
0	0	0	No parity
0	0	1	No parity
0	1	0	No parity
0	1	1	No parity
1	0	0	Odd parity
1	0	1	Even parity
1	1	0	Mark parity
1	1	1	Space parity

Diagram

Bits	7	6	5	4	3	2	1	0
R W	DLAB	SB	SP	EPS	PEN	NSTB	WLS	
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 DLAB	Divisor latch access bit. 0b - Access to all registers except UDLB, UAFR, and UDMB 1b - Ability to access divisor latch least and most significant byte registers and alternate function register (UAFR)
6 SB	Set break. 0b - Send normal UTHR data onto the serial output (SOUT) signal 1b - Force logic 0 to be on the SOUT signal. Data in the UTHR is not affected

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 SP	Stick parity. 0b - Stick parity is disabled. 1b - If PEN = 1 and EPS = 1, space parity is selected. And if PEN = 1 and EPS = 0, mark parity is selected.
4 EPS	Even parity select. See the table above for more information. 0b - If PEN = 1 and SP = 0, odd parity is selected. 1b - If PEN = 1 and SP = 0, even parity is selected.
3 PEN	Parity enable. 0b - No parity generation and checking 1b - Generate parity bit as a transmitter, and check parity as a receiver
2 NSTB	Number of STOP bits. 0b - One STOP bit is generated in the transmitted data. 1b - When a 5-bit data length is selected, 1½ STOP bits are generated. When either a 6-, 7-, or 8-bit word length is selected, two STOP bits are generated.
1-0 WLS	Word length select. Number of bits that comprise the character length. The word length select values are as follows: 00b - 5 bits 01b - 6 bits 10b - 7 bits 11b - 8 bits

23.4.11 UART modem control register 1 (UMCR1)

Offset

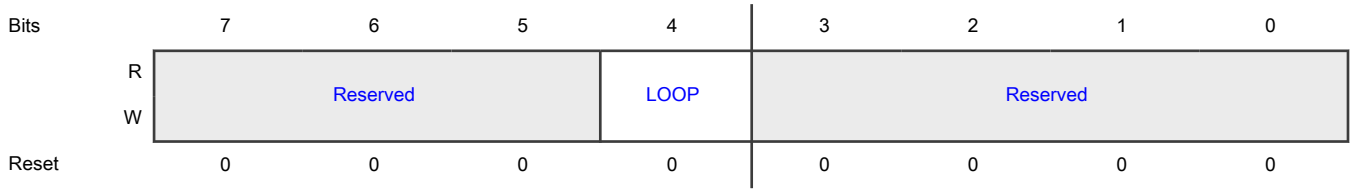
Register	Offset
UMCR1	504h

Function

This register is accessible when ULCR[DLAB] = x.

The UMCRs control the interface with the external peripheral device on the UART bus.

Diagram



Fields

Field	Function
7-5 —	Reserved.
4 LOOP	Local loopback mode. 0b - Normal operation 1b - Functionally, the data written to UTHR can be read from URBR of the same UART.
3-0 —	Reserved.

23.4.12 UART line status register (ULSR1)

Offset

Register	Offset
ULSR1	505h

Function

This register is accessible when ULCR[DLAB] = x.

The ULSRs are read-only registers that monitor the status of the data transfer on the UART buses. To isolate the status bits from the proper character received through the UART bus, software should read the ULSR and then the URBR.

Diagram

Bits	7	6	5	4	3	2	1	0
R	RFE	TEMT	THRE	BI	FE	PE	OE	DR
W								
Reset	0	1	1	0	0	0	0	0

Fields

Field	Function
7 RFE	Receiver FIFO error. 0b - This bit is cleared when there are no errors in the receiver FIFO or on a read of the ULSR with no remaining receiver FIFO errors. 1b - Set to one when one of the characters in the receiver FIFO encounters an error (framing, parity, or break interrupt)
6 TEMT	Transmitter empty. 0b - Either or both the UTHR or the internal transmitter shift register has a data character. In FIFO mode, a data character is in the transmitter FIFO or the internal transmitter shift register. 1b - Both the UTHR and the internal transmitter shift register are empty. In FIFO mode, both the transmitter FIFO and the internal transmitter shift register are empty.
5 THRE	Transmitter holding register empty. 0b - The UTHR is not empty. 1b - A data character has transferred from the UTHR into the internal transmitter shift register. In FIFO mode, the transmitter FIFO contains no data character.
4 BI	Break interrupt. **NOTE** For a single break signal, BI and DR are set multiple times, approximately once every character period. The BI and DR bits continue to be set each character period after they are cleared. This continues for the entire duration of the break signal. To accommodate this behavior, read URBR, which returns zeros and clears DR. Then delay one character period and read URBR again. Note that at the end of the break signal, a random character may be falsely detected and received in the URBR, with ULSR[DR] being set. 0b - This bit is cleared when the ULSR is read or when a valid data transfer is detected (that is, STOP bit is received). 1b - Received data of logic 0 for more than START bit + Data bits + Parity bit + one STOP bits length of time. A new character is not loaded until SIN returns to the mark state (logic 1) and a valid START is detected. In FIFO mode, a zero character is encountered in the FIFO (the zero character is at the top of the FIFO). In FIFO mode, only one zero character is stored.
3 FE	Framing error. 0b - This bit is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Invalid STOP bit for receive data (only the first STOP bit is checked). In FIFO mode, this bit is set when the character that detected a framing error is encountered in the FIFO (that is the character at the top of the FIFO). An attempt to resynchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit, so it assumes this logic 0 sample is a true START bit and then receives the following new data.
2 PE	Parity error. 0b - This bit is cleared when ULSR is read or when a new character is loaded into the URBR. 1b - Unexpected parity value encountered when receiving data. In FIFO mode, the character with the error is at the top of the FIFO .
1 OE	Overrun error. 0b - This bit is cleared when ULSR is read. 1b - Before the URBR is read, the URBR was overwritten with a new character. The old character is loss. In FIFO mode, the receiver FIFO is full (regardless of the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. The old character was overwritten by the new character. Data in the receiver FIFO was not overwritten.
0 DR	Data ready. 0b - This bit is cleared when URBR is read or when all of the data in the receiver FIFO is read. 1b - A character has been received in the URBR or the receiver FIFO.

23.4.13 UART scratch register (USCR1)

Offset

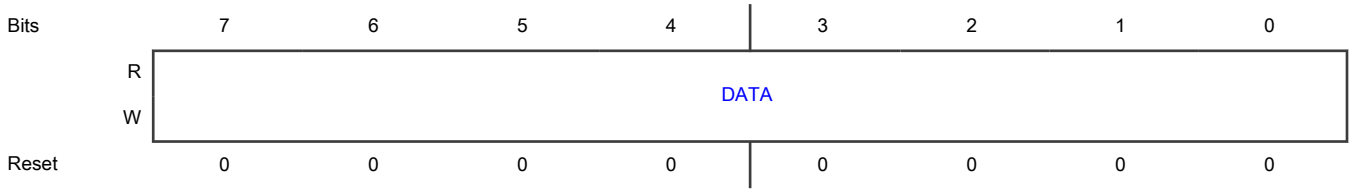
Register	Offset
USCR1	507h

Function

This register is accessible when ULCR[DLAB] = x.

The USCR registers are for debugging software or the UART hardware. The USCRs do not affect the operation of the UART.

Diagram



Fields

Field	Function
7-0 DATA	Data

23.4.14 UART DMA status register (UDSR1)

Offset

Register	Offset
UDSR1	510h

Function

This register is accessible when ULCR[DLAB] = x.

The DMA status registers (UDSRs) are read-only registers that return transmitter and receiver FIFO status. UDSRs also provide the ability to assist DMA data operations to and from the FIFOs.

Table 102. UDSR[TXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is set after the first character is loaded into the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is set when the transmitter FIFO is full.

Table 103. UDSR[TXRDY] Cleared Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR.
0	1	0	
1	0	0	

Table continues on the next page...

Table 103. UDSR[TXRDY] Cleared Conditions (continued)

DMS	FEN	DMA Mode	Meaning
1	1	1	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR. TXRDY remains clear when the transmitter FIFO is not yet full.

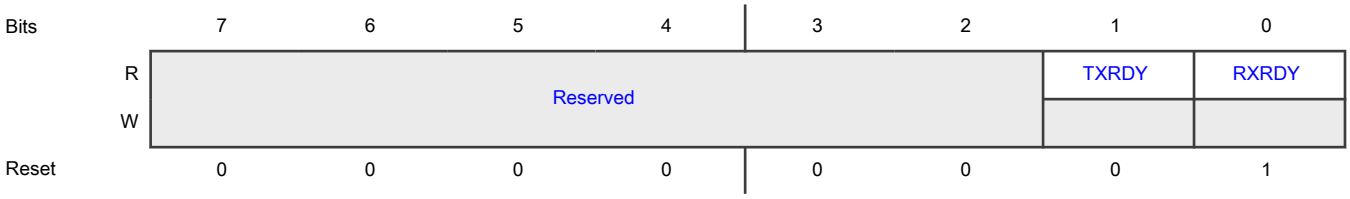
Table 104. UDSR[RXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is set when there are no characters in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is set when the trigger level has not been reached and there has been no time out.

Table 105. UDSR[RXRDY] Cleared Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is cleared when there is at least one character in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is cleared when the trigger level or a time-out has been reached. RXRDY remains cleared until the receiver FIFO is empty.

Diagram



Fields

Field	Function
7-2 —	Reserved
1 TXRDY	Transmitter ready. This read-only bit reflects the status of the transmitter FIFO or the UTHR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR.

Table continues on the next page...

Table continued from the previous page...

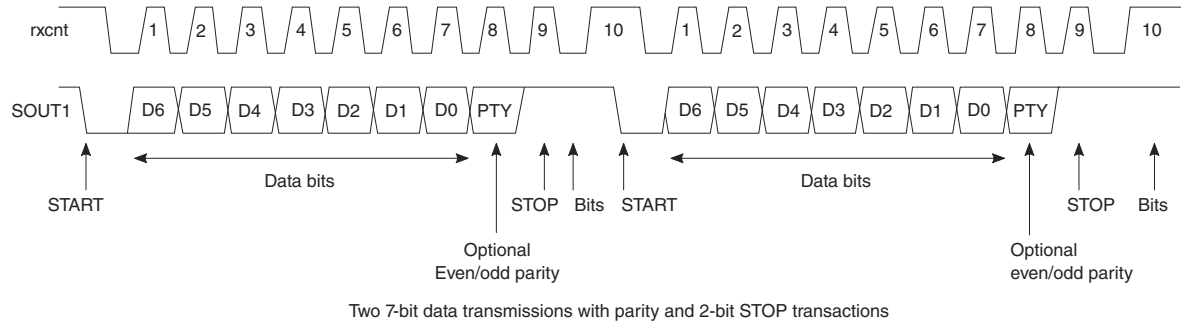
Field	Function
	0b - This bit is cleared, as shown in #unique_788/unique_788_Connect_42_UDSR1/tb739488 . 1b - This bit is set, as shown in #unique_788/unique_788_Connect_42_UDSR1/tb197655 .
0 RXRDY	Receiver ready. This read-only bit reflects the status of the receiver FIFO or URBR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR. 0b - This bit is cleared, as shown in #unique_788/unique_788_Connect_42_UDSR1/tb738939 . 1b - This bit is set, as shown in #unique_788/unique_788_Connect_42_UDSR1/tb738592 .

23.5 Functional description

The following sections provide the function of the UART controller.

23.5.1 Serial interface

The UART bus is a serial, point-to-point bus as shown in the following figure. Therefore, only two devices are attached to the same signals and there is no need for address or arbitration bus cycles.



Two 7-bit data transmissions with parity and 2-bit STOP transactions

Figure 71. UART bus interface transaction protocol example

A standard UART bus transfer is composed of either three or four parts:

- START bit
- Data transfer bits (least-significant bit is first data bit on the bus)
- Parity bit (optional)
- STOP bits

An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to drive the bits on SOUT.

The following sections describe the four components of the serial interface, the baud-rate generator, different errors, and FIFO mode.

23.5.1.1 START bit

A write to the transmitter holding register (UTHR) generates a START bit on the SOUT signal. Figure 71 the figure shows that the START bit is defined as a logic 0. The START bit denotes the beginning of a new data transfer which is limited to the bit length programmed in the UART line control register (ULCR). When the bus is idle, SOUT is high.

23.5.1.2 Data transfer

Each data transfer contains 5-8 bits of data. The ULCR data bit length for the transmitter and receiver UART devices must agree before a transfer begins; otherwise, a parity or framing error may occur. A transfer begins when UTHR is written. At that time a START bit is generated followed by 5-8 of the data bits previously written to the UTHR. The data bits are driven from the least significant to the most significant bits. After the parity and STOP bits, a new data transfer can begin if new data is written to the UTHR.

23.5.1.3 Parity bit

The user has the option of using even, odd, no parity, or stick parity. See UART line control register (ULCR1). Both the receiver and transmitter parity definition must agree before attempting to transfer data. When receiving data, a parity error can occur if an unexpected parity value is detected. See UART line status register (ULSR1).

23.5.1.4 STOP bit

The transmitter device ends the write transfer by generating a STOP bit. The STOP bit is always high. The length of the STOP bit(s) can be programmed in the ULCR. Both the receiver and transmitter STOP bit length must agree before attempting to transfer data. A framing error can occur if an invalid STOP bit is detected.

23.5.2 Baud-rate generator logic

Each UART contains an independent programmable baud-rate generator, that is capable of taking the platform clock frequencyas input and dividing the input by any divisor from 1 to 2^{16} - 1.

The baud rate is defined as the number of bits per second that can be sent over the UART bus. The formula for calculating baud rate is as follows:

Baud rate = (1/16) x (platform clock frequency ÷ divisor value)

Therefore, the output frequency of the baud-rate generator is 16 times the baud rate.

The divisor value is determined by the following two 8-bit registers to form a 16-bit binary number:

- UART divisor most significant byte register (UDMB)
- UART divisor least significant byte register (UDLB)

Upon loading either of the divisor latches, a 16-bit baud-rate counter is loaded.

The divisor latches must be loaded during initialization to ensure proper operation of the baud-rate generator. Both UART devices on the same bus must be programmed for the same baud-rate before starting a transfer.

The baud clock can be passed to the performance monitor by enabling the UAFR[BO] bit. This can be used to determine baud rate errors.

23.5.3 Errors

The following sections describe framing, parity, and overrun errors which may occur while data is transferred on the UART bus. Each of the error bits are usually cleared, as described below, when the line status register (ULSR) is read.

23.5.3.1 Framing error

When an invalid STOP bit is detected, a framing error occurs and ULSR[FE] is set. Note that only the first STOP bit is checked. In FIFO mode, ULSR[FE] is set when the character at the top of the FIFO detects a framing error. An attempt to re-synchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit. ULSR[FE] is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register.

23.5.3.2 Parity error

A parity error occurs, and ULSR[PE] is set, when unexpected parity values are encountered while receiving data. In FIFO mode, ULSR[PE] is set when the character with the error is at the top of the FIFO. ULSR[PE] is cleared when ULSR is read or when a new character is loaded into the URBR.

23.5.3.3 Overrun error

When a new (overwriting character) STOP bit is detected and the old character is lost, an overrun error occurs and ULSR[OE] is set. In FIFO mode, ULSR[OE] is set after the receiver FIFO is full (despite the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. Data in the FIFO is not overwritten; only the shift register data is overwritten. Therefore, the interrupt occurs immediately. ULSR[OE] is cleared when ULSR is read.

23.5.4 FIFO mode

The UARTs use an alternate mode (FIFO mode) to relieve the processor core from excessive software overhead. The FIFO control register (UFCR) is used to enable and clear the receiver and transmitter FIFOs and set the FIFO receiver trigger level UFCR[RTL] to control the received data available interrupt UIER[ERDAI].

The UFCR also selects the type of DMA signaling. The UDSR[RXRDY] indicates the status of the receiver FIFO. The DMA status registers (UDSR[TXRDY]) indicate when the transmitter FIFO is full. When in FIFO mode, data written to UTHR is placed into the transmitter FIFO. The first byte written to UTHR is the first byte onto the UART bus.

23.5.4.1 FIFO interrupts

In FIFO mode, the UIER[ERDAI] is set when a time-out interrupt occurs. When a receive data time-out occurs there is a maskable interrupt condition (through UIER[ERDAI]). See UART interrupt enable register (UIER1) for more details on interrupt enables.

The interrupt ID register (UIIR) indicates if the FIFOs are enabled. The interrupt ID3 bit UIIR[IID3] is only set for FIFO mode interrupts. The character time-out interrupt occurs when no characters have been removed from or input to the receiver FIFO during the last four character times and there is at least one character in the receiver FIFO during this time. The character time-out interrupt (controlled by UIIR[IID*n*]) is cleared when the URBR is read. See UART interrupt ID register (UIIR1) for more information.

The UIIR[FE] bit indicates if FIFO mode is enabled.

23.5.4.2 Interrupt control logic

An interrupt is active when UART interrupt ID register bit 7 (UIIR[IID0]), is cleared. The interrupt enable register (UIER) is used to mask specific interrupt types. See UART interrupt enable register (UIER1) for more details.

When the interrupts are disabled in UIER, polling software cannot use UIIR[IID0] to determine whether the UART is ready for service. The software must monitor the appropriate bits in the line status (ULSR) and/or the modem status registers (UMSR). UIIR[IID0] can be used for polling if the interrupts are enabled in UIER.

23.6 Initialization/Application information

The following requirement must be met for UART accesses:

- All UART registers are 1 byte wide. Reads and writes to these registers must be byte-wide operations.

A system reset puts the UART registers to a default state. Before the interface can transfer serial data, the following initialization steps are recommended:

1. Update the programmable interrupt controller (PIC) UART channel interrupt vector source registers.
2. Set data attributes and control bits in the ULCR, UFCR, UAFR, UMCR, UDLB, and UDMB.
3. Set the data attributes and control bits of the external modem or peripheral device.
4. Set the interrupt enable register (UIER).
5. To start a write transfer, write to the UTHR.

6. Poll UIIR, if the interrupts generated by the UART are masked.

Chapter 24
VSPA Architecture Overview

24.1 Overview

This chapter gives a brief overview of the VSPA implementation on the chip. For more details please refer "VSPA-16SP ISA-v2.0 Instruction Set Manual".

24.2 VSPA Architecture Overview

24.2.1 VSPA architecture introduction

VSPA is a signal processing platform which leverages a Single Instruction Multiple Data (SIMD) data path and VLIW control plane to provide extremely high compute capability per milli-watt and/or silicon area. The instruction set and scalable data path are optimized for a broad range of applications. Some of these include low-complexity modems such as Bluetooth or AM/FM car radio, multi-channel audio, and the antenna signal processing used in highly complex 5G massive-MIMO cellular base stations. The architecture employs a very simple control plane so it is not optimum for control dominant or packet switching and layer 2 applications. VSPA is typically used in SOC's with a more traditional general purpose host controller such as an ARM which provide access to general peripherals. The architecture is also very suitable as a math-co processor for more general compute platforms.

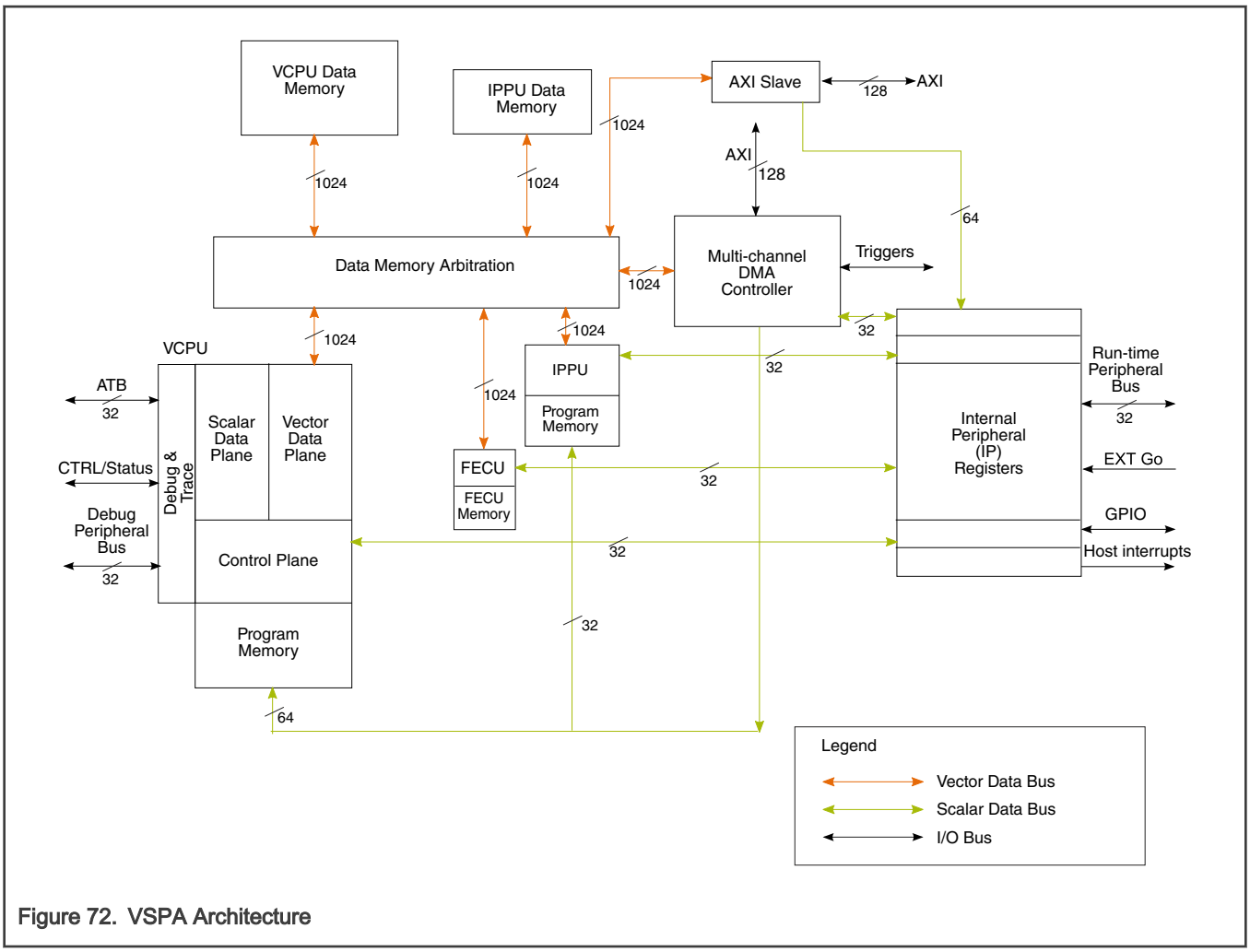


Figure 72. VSPA Architecture

The architecture consists of the following functional units:

24.2.1.1 VCPU introduction

VCPU stands for Vector Central Processing Unit.

The VSPA architecture consists of control and data planes.

- Control plane: The control plane is SIMD and can execute several operations on the data plane in each cycle. It uses a single issue pipeline and is optimized for efficient data operations. Software threads run to completion and begin with a 'go' or wake-up event initiated by an external event or DMA transfer completion. The 'done' instruction is the last instruction executed in a thread and puts the machine in a low power state.
- Data plane: All vector data math operations are implemented in software on the VCPU. The VCPU data path is organized into *vector* data plane and *scalar* data plane.

The Intervector Permutation Processing Unit and Vector Central Processing Unit are programmable machines supported by the tool chain.

Table 106. Vector/Scalar data plane features

Vector data plane	Scalar data plane
• 1024-bit vector data bus • 64 SP RMAC/RMAD operations per clock cycle • Vector sqrt(x), 1/x, & 1/sqrt(x) operations • Vector NCO operations • Vector spreading/scrambling code generator for WCDMA • Independent vector compare engine • Multi-port, 8-line vector register array • Transparent intra-vector permutations on each VRA port • 2 vector rotate units • 256-bit vector sign register	• 32-bit data plane • Full-featured arithmetic and logic unit • 12 scalar registers • Stack pointer • 20 memory pointer registers with modulo buffer and re-ordering algorithm support

24.2.1.2 IPPU introduction

The Intervector Permutation Processing Unit (IPPU) is an independent programmable machine which is responsible for reordering buffers for efficient utilization of the vector data path on the VCPU. It has independent program and data memories so it can run in parallel with the VCPU. This allows buffers to be pipelined, allowing vector math of buffer N to be processed on the VCPU while buffer N+1 is reordered on the IPPU. The IPPU has a limited instruction set optimized for flexible construction of vectors from scalars of various data types. The IPPU and VCPU data memories are visible to the other core through an arbitor to allow efficient transfer of buffers.

24.2.1.3 DMA controller

The Direct Memory Access (DMA) Controller is an independent state machine which is responsible for movement of data buffers between VSPA core and other cores, on-chip peripherals, memories and any other memory mapped component. It is compliant with the AMBA AXI3 bus protocol with independent read and write busses and a configurable size depending on the target SOC. The module supports up to 32 independent channels (active simultaneous transfers) serviced in a round-robin manner on a 16-beat burst interval. Data transfers can target both IPPU and VCPU data memories. Program images can also be loaded using the DMA. Each DMA channel can be configured to start execution of a software thread when the transfer completes. The thread can reside on the IPPU or VCPU. The DMA control resides in the IP register map and is visible to both the host (ARM) and VCPU. The boot-up sequence requires the host to configure one of the DMA channels to download the inital VCPU program image.

24.2.1.4 IP registers

Control of the IPPU, DMA, GPIO, external go events and other miscellaneous peripherals is configured with a set of memory-mapped 32-bit internal peripheral (IP) bus registers. Both the host and VCPU have access to these registers, although the physical addresses are different as viewed from each core.

24.2.1.5 Vector data memory

Data is organized into 1024-bit lines, although the physical structure may consist of multiple parallel instantiations of smaller memory blocks. Independent tightly coupled data memories exist for both the IPPU and VCPU to allow simultaneous execution of programs and data memory accesses. Both memories are visible to the DMA, VCPU and IPPU machines through a data arbiter module.

24.2.1.6 Debug and trace unit

This unit is responsible for invasive debug (break, stop, step, go, etc.) and non-invasive program and data trace of the VCPU core.

24.2.2 Variants of VSPA

VSPA can be classified based on the number of arithmetic units (AU). LA9310 has 1 instance of VSPA 16AU single precision (SP). This means that, VSPA core for LA9310 supports:

- 16 complex MAC operations per clock cycle
- Single Precision math

Chapter 25
Watchdog Timer Unit

25.1 The WDOG Timer module as implemented on the chip

The WatchDog Timer on the chip is enabled by writing a 1 to PCTBENR at CCSR base address 1E308A0h described below. PCTBENR provides a mechanism for enabling clock to the core timebase on the device.

Table 107. PCTBENR

Offset	Absolute Address	Register	Width	Access	Reset Value
8A0h	1E308A0h	Physical Core Time Base Enable Register (PCTBENR)	32	RW	00000000h

Bits	Name	Description
31-1	—	Reserved
0	PCTBEN0	Core Timebase Enable. 1’b0 - Selected Core Timebase is disabled 1’b1 - Selected Core Timebase is enabled

25.2 Watchdog unit

The Watchdog module is intended to provide a way of recovering from software crashes. The module is based around a 32-bit down counter that is initialized from the Watchdog Load Register, WDOGLOAD. It can be enabled or disabled as required. The watchdog clock generates a regular interrupt, WDOGINT, depending on a programmed value. The counter decrements by one on each positive clock edge of WDOGCLK when WDOGCLKEN is HIGH. Asserting WDOGCLKEN is done by configuring PMU register: Physical Core Time Base Enable (PCTBENR). The watchdog monitors the interrupt and asserts a reset WDOGRES signal when the counter reaches zero and the counter is stopped. On the next enabled WDOGCLK clock edge the counter is reloaded from the WDOGLOAD Register and the count down sequence continues. If the interrupt is not cleared by the time that the counter next reaches zero then the Watchdog module reasserts the reset signal. The following figure shows the watchdog block diagram.

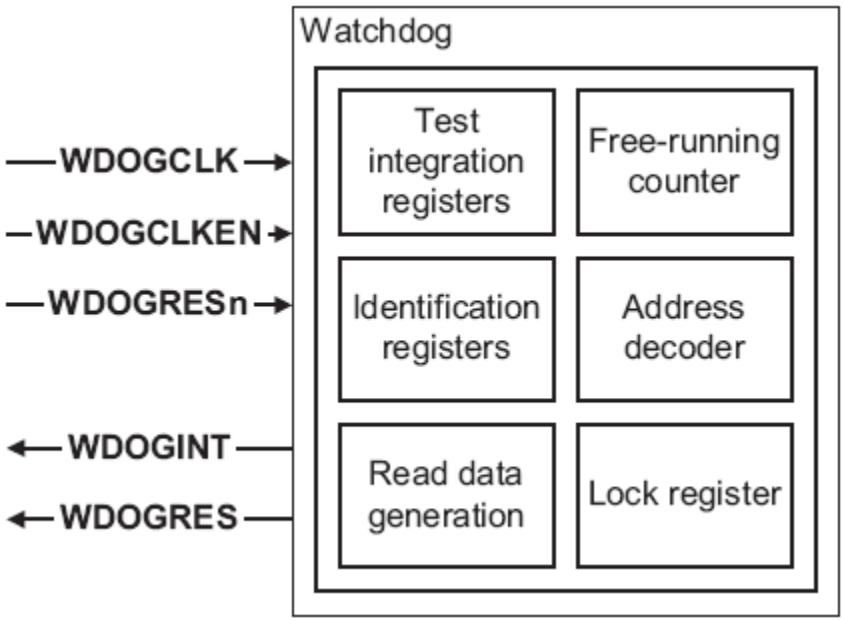


Figure 73. Watchdog components

25.3 Peripheral Identification Registers

Peripheral Identification Registers:

The WDOGPERIPHID0-3 registers are four 8-bit registers that span address locations `0xFE0-0xFEC`. The read-only registers provide the following options for the Watchdog Unit (peripheral):

Part number [11:0] This identifies the peripheral. The three-digit product code `0x805` is used for the watchdog unit.

Designer [19:12] This is the identification of the designer. Arm Limited is `0x41` (ASCII A).

Revision number [23:20] This is the revision number of the peripheral. The revision number starts from 0.

Configuration [31:24] This is the configuration option of the peripheral. The configuration value is 0.

The figure below shows the register bit assignments.

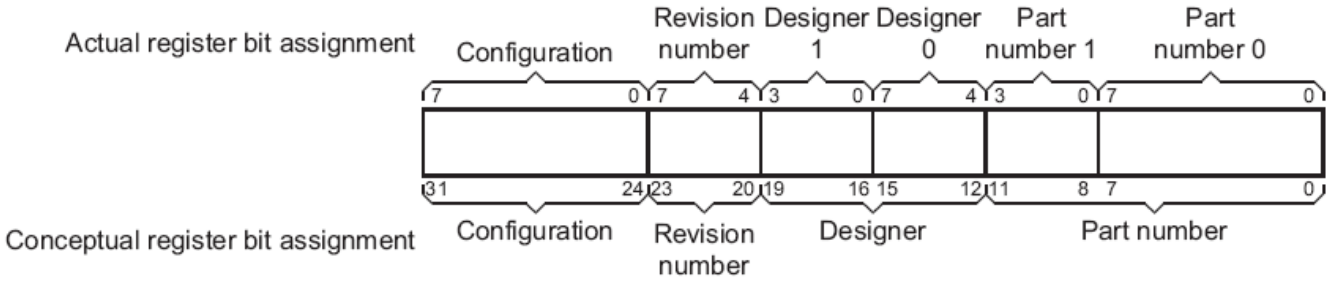


Figure 74. WDOGPERIPHID0-3 Register bit assignments

NOTE

All memory accesses to the peripheral identification registers must be 32-bit, using the LDR and STR instructions.

The four 8-bit peripheral identification registers are described in the following subsections:

- Peripheral Identification Register 0 (WDOGPERIPHID0)
- Peripheral Identification Register 1 (WDOGPERIPHID1)
- Peripheral Identification Register 2 (WDOGPERIPHID2)
- Peripheral Identification Register 3 (WDOGPERIPHID3)

25.4 PrimeCell Identification Registers

PrimeCell Identification Registers :

The WDOGPCELLID0-3 Registers are four 8-bit wide registers that span address locations `0xFF0-0xFFC`. The registers can conceptually be treated as a 32-bit register. The register is used as a standard cross-peripheral identification system. The WDOGPCELLID Register is set to `0xB105F00D`. The figure below shows the register bit assignments.

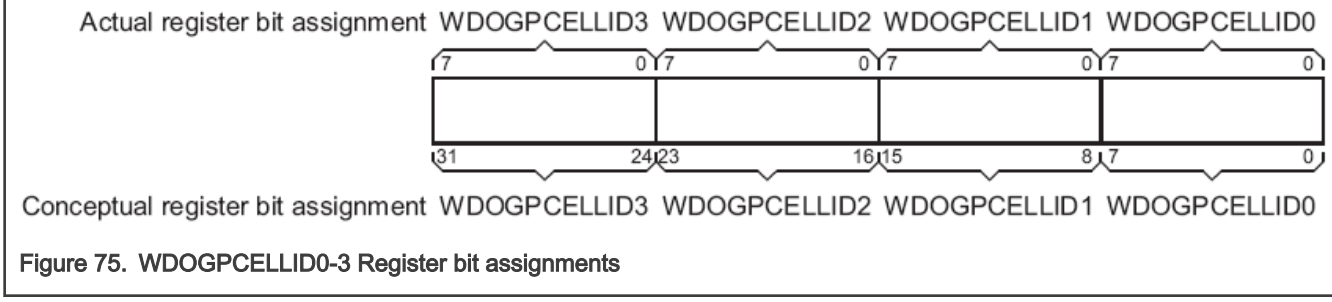


Figure 75. WDOGPCELLID0-3 Register bit assignments

The four, 8-bit PrimeCell identification registers are described in the following subsections:

- PrimeCell Identification Register 0 (WDOGPCELLID0)
- PrimeCell Identification Register 1 (WDOGPCELLID1)
- PrimeCell Identification Register 2 (WDOGPCELLID2)
- PrimeCell Identification Register 3 (WDOGPCELLID3)

25.5 WDOG register descriptions

This section describes the Watchdog registers.The base address included is for the first instance of Watchdog Timer on the chip. For more information on the base addresses of other instances of Watchdog Timers supported on the chip, please refer the CCSR Address Map in the Memory Map chapter.

25.5.1 WDOG memory map

WDOG base address: 23C_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Watchdog Load Register (WDOGLOAD)	32	RW	FFFF_FFFFh
4h	Watchdog Value Register (WDOGVALUE)	32	RW	FFFF_FFFFh
8h	Watchdog Control Register (WDOGCONTROL)	32	RW	0000_0000h
Ch	Watchdog Clear Interrupt Register (WDOGINTCLR)	32	WO	See description.

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
10h	Watchdog Raw Interrupt Status Register (WDOGRIS)	32	RO	0000_0000h
14h	Watchdog Interrupt Status Register (WDOGMIS)	32	RO	0000_0000h
C00h	Watchdog Lock Register (WDOGLOCK)	32	RW	0000_0000h
F00h	Watchdog Integration Test Control Register (WDOGITCR)	32	RW	0000_0000h
F04h	Watchdog Integration Test Output Set Register (WDOGITOP)	32	WO	0000_0005h
FE0h	Peripheral Identification Register 0 (WDOGPERIPHID0)	32	RO	0000_0005h
FE4h	Peripheral Identification Register 1 (WDOGPERIPHID1)	32	RO	0000_0018h
FE8h	Peripheral Identification Register 2 (WDOGPERIPHID2)	32	RO	0000_0014h
FECh	Peripheral Identification Register 3 (WDOGPERIPHID3)	32	RO	0000_0000h
FF0h	PrimeCell Identification Register 0 (WDOGPCELLID0)	32	RO	0000_000Dh
FF4h	PrimeCell Identification Register 1 (WDOGPCELLID1)	32	RO	0000_00F0h
FF8h	PrimeCell Identification Register 2 (WDOGPCELLID2)	32	RO	0000_0005h
FFCh	PrimeCell Identification Register 3 (WDOGPCELLID3)	32	RO	0000_00B1h

25.5.2 Watchdog Load Register (WDOGLOAD)

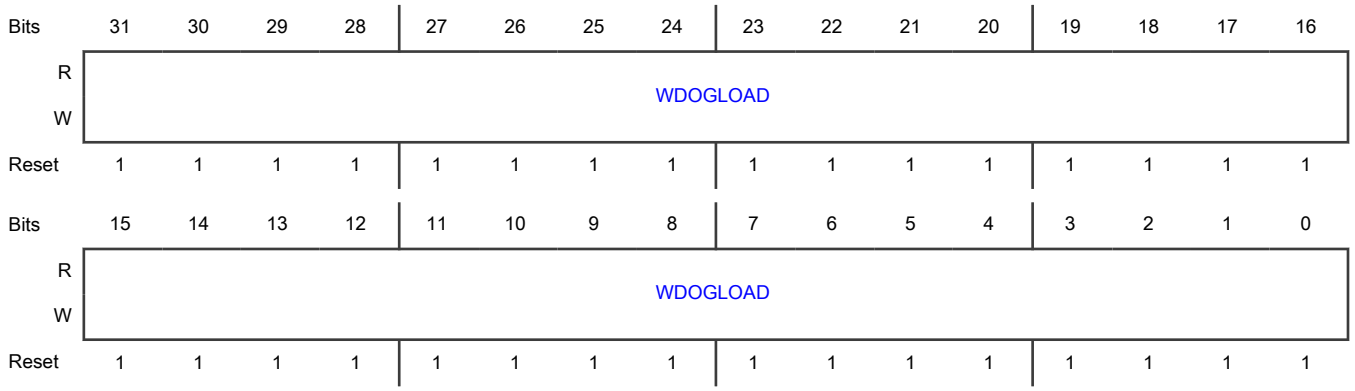
Offset

Register	Offset
WDOGLOAD	0h

Function

The WDOGLOAD Register is a 32-bit register containing the value from which the counter is to decrement. When this register is written to, the count is immediately restarted from the new value. The minimum valid value for WDOGLOAD is 1.

Diagram



Fields

Field	Function
31-0 WDOGLOAD	WDOGLOAD The value from which the counter is to decrement.

25.5.3 Watchdog Value Register (WDOGVALUE)

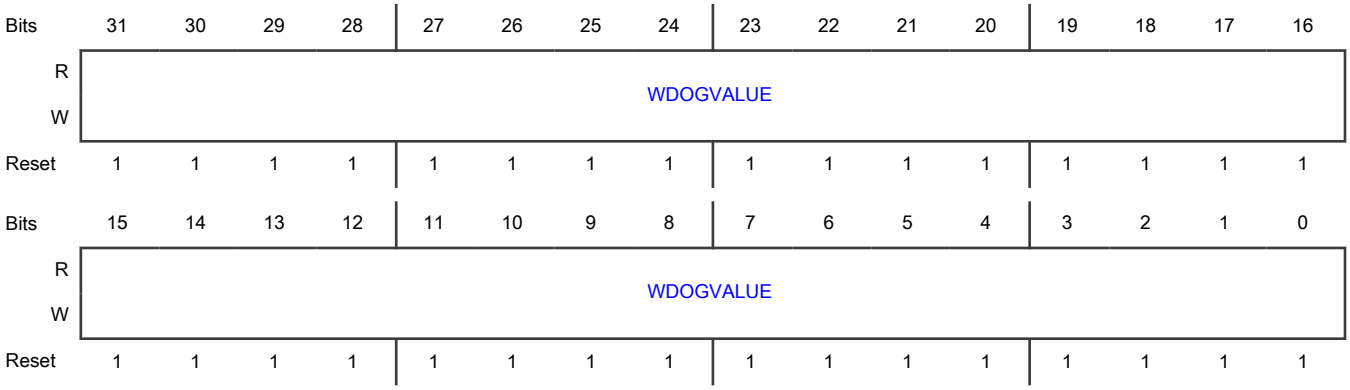
Offset

Register	Offset
WDOGVALUE	4h

Function

The WDOGVALUE Register gives the current value of the decrementing counter.

Diagram



Fields

Field	Function
31-0 WDOGVALUE	WDOGVALUE The current value of the decrementing counter.

25.5.4 Watchdog Control Register (WDOGCONTROL)

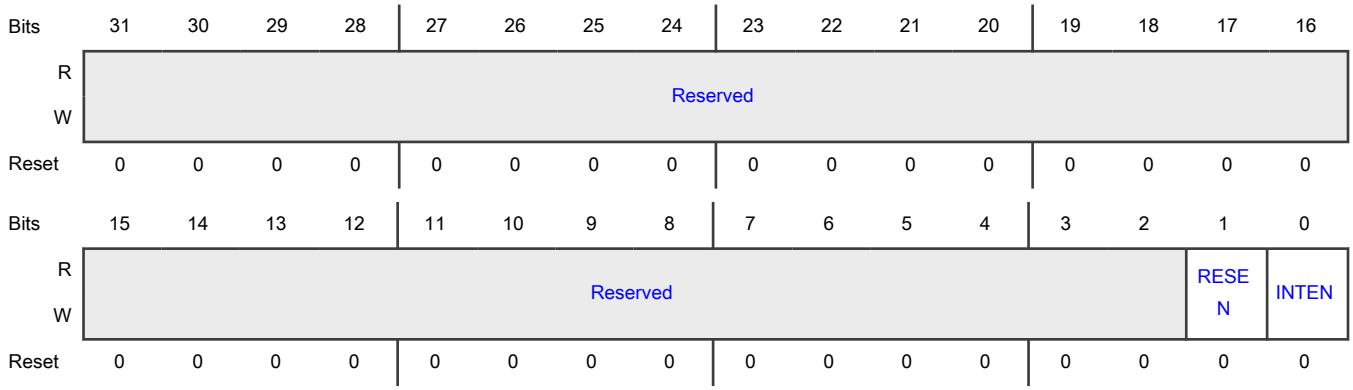
Offset

Register	Offset
WDOGCONTROL	8h

Function

The WDOGCONTROL Register is a read/write register that enables the software to control the watchdog unit.

Diagram



Fields

Field	Function
31-2 —	Reserved
1 RESEN	RESEN Enable Watchdog reset output (*WDOGRES*). Acts as a mask for the reset output. Set to 1 to enable the reset, and to 0 to disable the reset.
0 INTEN	INTEN Enable the interrupt event (*WDOGINT*). Set to 1 to enable the counter and the interrupt, and set to 0 to disable the counter and interrupt. Reloads the counter from the value in WDOGLOAD when the interrupt is enabled, and was previously disabled.

25.5.5 Watchdog Clear Interrupt Register (WDOGINTCLR)

Offset

Register	Offset
WDOGINTCLR	Ch

Function

A write of any value to the WDOGINTCLR Register clears the watchdog interrupt and reloads the counter from the value in WDOGLOAD.

Diagram

Fields

Field	Function
31-1 —	Reserved
0 Raw_Watchdog_Interrupt	Raw_Watchdog_Interrupt Raw interrupt status from the counter

25.5.6 Watchdog Raw Interrupt Status Register (WDOGRIS)

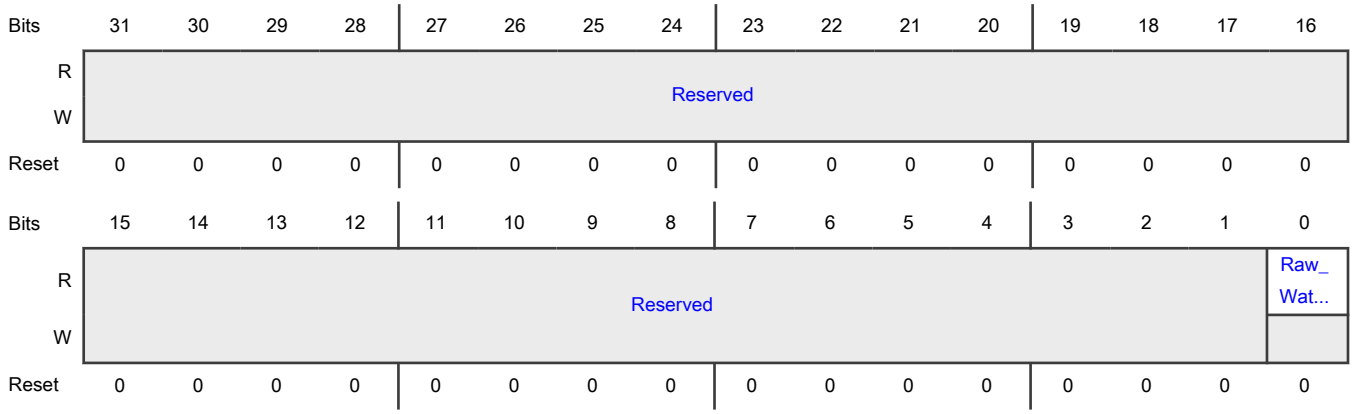
Offset

Register	Offset
WDOGRIS	10h

Function

The WDOGRIS Register is read-only. It indicates the raw interrupt status from the counter. This value is ANDed with the interrupt enable bit from the control register to create the masked interrupt, which is passed to the interrupt output pin.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 Raw_Watchdog_Interrupt	Raw_Watchdog_Interrupt Raw interrupt status from the counter

25.5.7 Watchdog Interrupt Status Register (WDOGMIS)

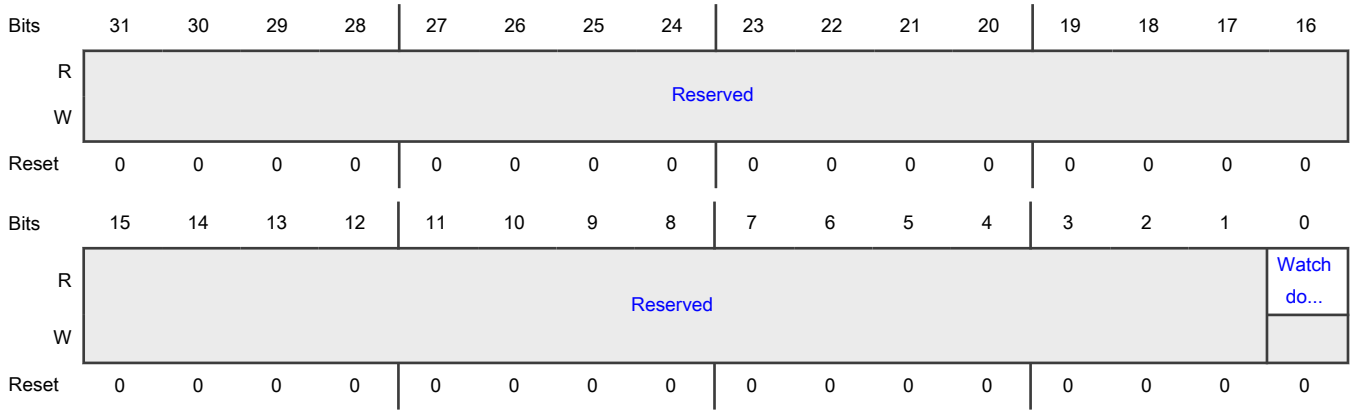
Offset

Register	Offset
WDOGMIS	14h

Function

The WDOGMIS Register is read-only. It indicates the masked interrupt status from the counter. This value is the logical AND of the raw interrupt status with the INTEN bit from the control register, and is the same value which is passed to the interrupt output pin.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 Watchdog_Interrupt	Watchdog_Interrupt Enabled interrupt status from the counter

25.5.8 Watchdog Lock Register (WDOGLOCK)

Offset

Register	Offset
WDOGLOCK	C00h

Function

The WDOGLOCK Register is write-only. Use of this register causes write-access to all other registers to be disabled. This is to prevent rogue software from disabling the watchdog functionality. Writing a value of `0x1ACCE551` enables write access to all other registers. Writing any other value disables write accesses. A read from this register returns only the bottom bit:

- 0 indicates that write access is enabled (not locked)
- 1 indicates that write access is disabled (locked).

Diagram

Fields

Field	Function
31-1 Enable_register_writes	Enable_register_writes Enable write access to all other registers by writing 0x1ACCE551. Disable write access by writing any other value.
0 Register_write_enable_status	Register_write_enable_status Register_write_enable_status 0b - write access to all other registers is enabled (default) 1b - write access to all other registers is disabled

25.5.9 Watchdog Integration Test Control Register (WDOGITCR)

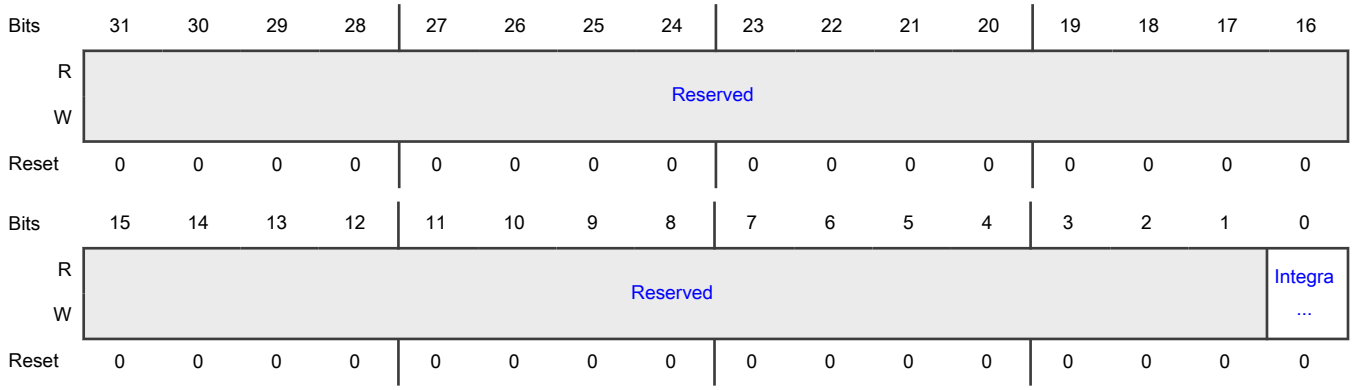
Offset

Register	Offset
WDOGITCR	F00h

Function

The WDOGITCR Register is read/write. It is a single-bit register that enables integration test mode. When in this mode, the masked interrupt output, WDOGINT, and reset output, WDOGRES, are directly controlled by the test output set register.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 Integration_Test_Mode_Enable	Integration_Test_Mode_Enable When set to 1, places the Watchdog into integration test mode

25.5.10 Watchdog Integration Test Output Set Register (WDOGITOP)

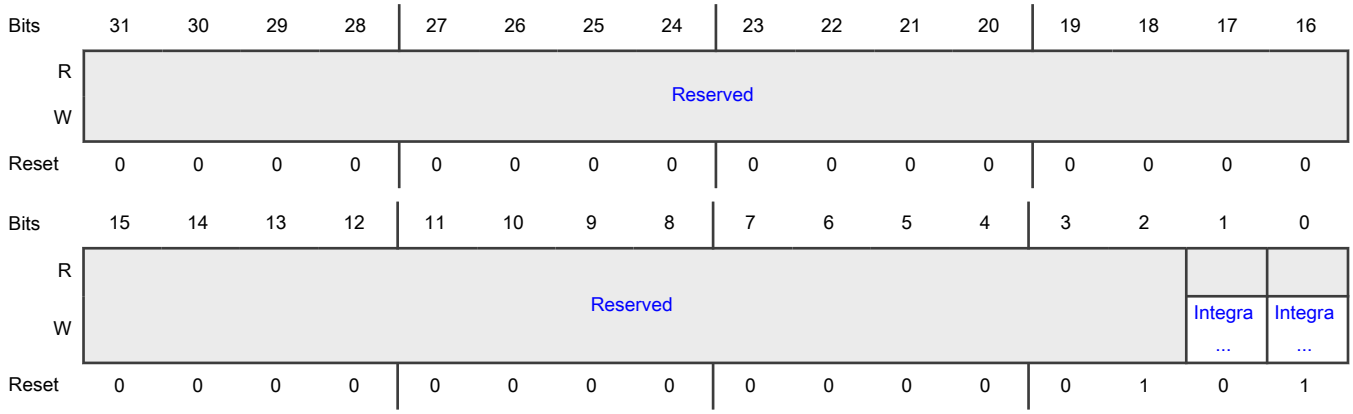
Offset

Register	Offset
WDOGITOP	F04h

Function

The WDOGITOP Register is write-only. When in integration test mode, the enabled interrupt output and reset output are driven directly from the values in this register.

Diagram



Fields

Field	Function
31-2 —	Reserved
1 Integration_Test_WDOGINT_value	Integration_Test_WDOGINT_value Value output on WDOGINT when in Integration Test Mode
0 Integration_Test_WDOGRES_value	Integration_Test_WDOGRES_value Value output on WDOGRES when in Integration Test Mode

25.5.11 Peripheral Identification Register 0 (WDOGPERIPHID0)

Offset

Register	Offset
WDOGPERIPHID0	FE0h

Function

The WDOGPERIPHID0 Register is hard-coded and the fields within the register determine the reset value.

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								PartNumber0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Fields

Field	Function
31-8 —	Reserved
7-0 PartNumber0	PartNumber0 These bits read back as 0x05.

25.5.12 Peripheral Identification Register 1 (WDOGPERIPHID1)

Offset

Register	Offset
WDOGPERIPHID1	FE4h

Function

The WDOGPERIPHID1 Register is hard-coded and the fields within the register determine the reset value.

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								Designer0				PartNumber1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Fields

Field	Function
31-8 —	Reserved
7-4 Designer0	Designer0 These bits read back as `0x1`
3-0 PartNumber1	PartNumber1 These bits read back as `0x08`

25.5.13 Peripheral Identification Register 2 (WDOGPERIPHID2)

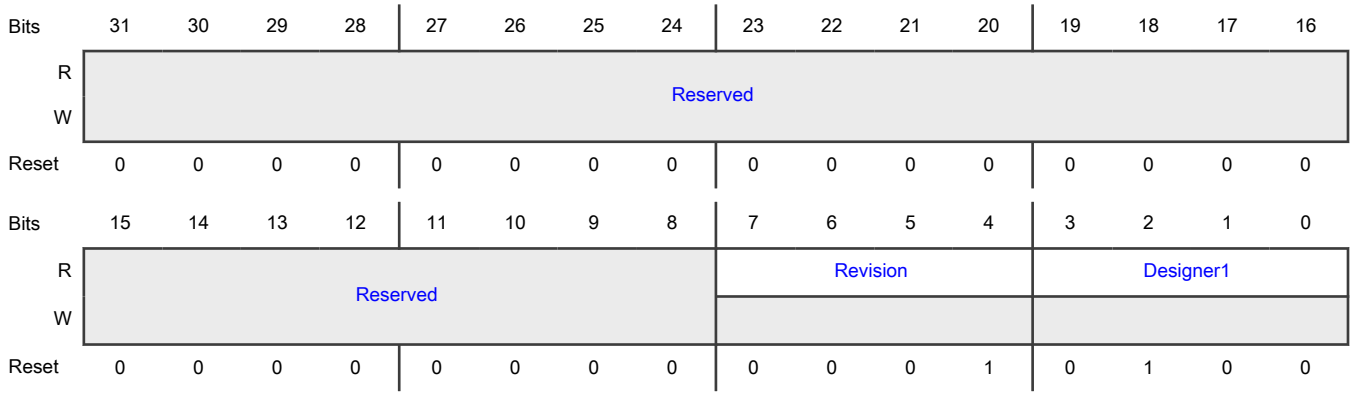
Offset

Register	Offset
WDOGPERIPHID2	FE8h

Function

The WDOGPERIPHID2 Register is hard-coded and the fields within the register determine the reset value.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-4	Revision

Table continues on the next page...

Table continued from the previous page...

Field	Function
Revision	These bits read back as `0x1`
3-0 Designer1	Designer1 These bits read back as `0x4`

25.5.14 Peripheral Identification Register 3 (WDOGPERIPHID3)

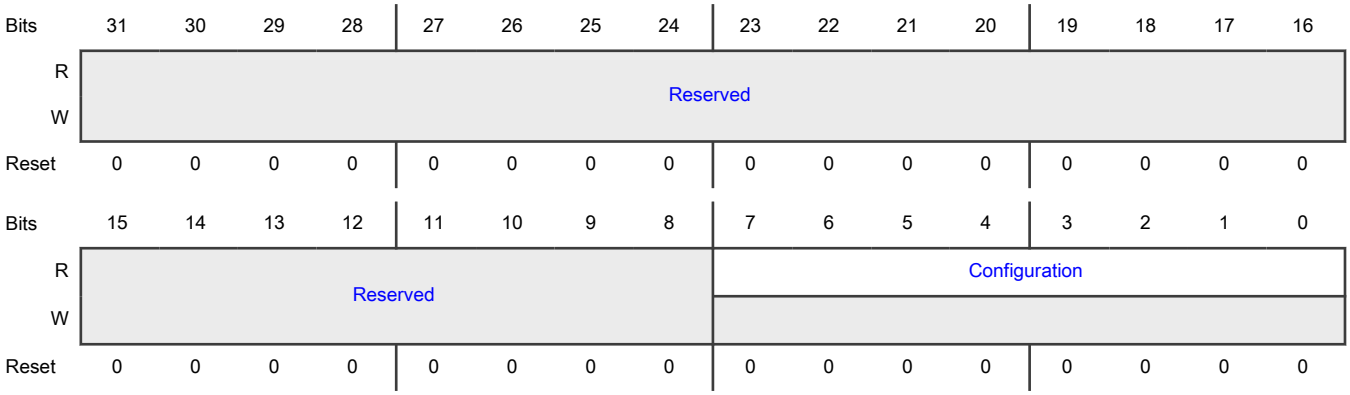
Offset

Register	Offset
WDOGPERIPHID3	FECh

Function

The WDOGPERIPHID3 Register is hard-coded and the fields within the register determine the reset value.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 Configuration	Configuration These bits read back as `0x00`

25.5.15 PrimeCell Identification Register 0 (WDOGPCELLID0)

Offset

Register	Offset
WDOGPCELLID0	FF0h

Function

The WDOGPCELLID0 Register is hard-coded and the fields within the register determine the reset value.

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								WdogPCellID0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Fields

Field	Function
31-8 —	Reserved
7-0 WdogPCellID0	WdogPCellID0 These bits read back as `0x0D`

25.5.16 PrimeCell Identification Register 1 (WDOGPCELLID1)

Offset

Register	Offset
WDOGPCELLID1	FF4h

Function

The WDOGPCELLID1 Register is hard-coded and the fields within the register determine the reset value.

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								WdogPCellID1							
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

Fields

Field	Function
31-8 —	Reserved
7-0 WdogPCellID1	WdogPCellID1 These bits read back as `0xF0`

25.5.17 PrimeCell Identification Register 2 (WDOGPCELLID2)

Offset

Register	Offset
WDOGPCELLID2	FF8h

Function

The WDOGPCELLID2 Register is hard-coded and the fields within the register determine the reset value.

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								WdogPCellID2							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Fields

Field	Function
31-8 —	Reserved
7-0 WdogPCellID2	WdogPCellID2 These bits read back as 0x05

25.5.18 PrimeCell Identification Register 3 (WDOGPCELLID3)

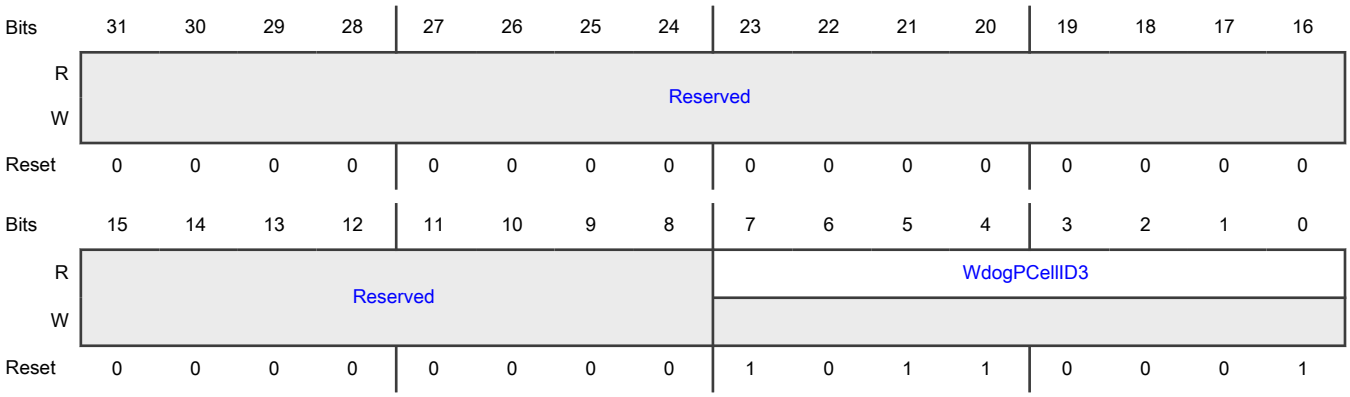
Offset

Register	Offset
WDOGPCELLID3	FFCh

Function

The WDOGPCELLID3 register is hard-coded and the fields within the register determine the reset value.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 WdogPCellID3	WdogPCellID3 These bits read back as 0xB1

Appendix A
Terminology Conventions and Resources

A.1 About this content

The primary objective of this document is to define the functionality of the chip.

NOTE

The diagrams in this content are provided to aid in understanding the overall functionality and features of this product. They do not depict the implementation details of the product, which are subject to change.

A.2 Audience

It is assumed that the reader understands operating systems, microprocessor system design, and the basic principles of RISC processing.

A.3 Acronyms and abbreviations

This table describes commonly-used acronyms and abbreviations used in this document.

Table 108. Acronyms and abbreviations

Acronym/ Abbreviation	Meaning
8b/10b	8-bit/10-bit encoding
BIST	Built-in self test
CAM	Content-addressable memory
CCSR	Configuration control and status register
CLASS	Chip-level arbitration and switching system
CRC	Cyclic redundancy check
ECC	Error checking and correction
FIFO	First in, first out
GPIO	General-purpose I/O
GPR	General-purpose register
I2C	Inter-integrated circuit
LSB	Least significant byte
lsb	Least significant bit
MAC	Multiply accumulate, media access control

Table continues on the next page...

Table 108. Acronyms and abbreviations (continued)

Acronym/ Abbreviation	Meaning
MMU	Memory management unit
MSB	Most significant byte
msb	Most significant bit
PCS	Physical coding sublayer
PIC	Programmable interrupt controller
PLL	Phase-locked loop
POR	Power-on reset
RSSI	Receive signal strength indicator
Rx	Receiver
SerDes	Serializer/Deserializer
SI	Serial interface
SPI	Serial peripheral interface
Tx	Transmitter
UART	Universal asynchronous receiver/transmitter

A.4 Notational conventions

This table shows notational conventions used in this content.

Table 109. Notational conventions

Convention	Definition
Cleared	When a bit takes the value zero, it is said to be cleared.
Set	When a bit takes the value one, it is said to be set.
mnemonics	Instruction mnemonics are shown in lowercase bold.
italics	Italics can indicate the following: • Variable command parameters, for example, **bcctr** *x* • Titles of publications • Internal signals, for example, *core int_B*
0x	Prefix to denote hexadecimal number

Table continues on the next page...

Table 109. Notational conventions (continued)

Convention	Definition
h	Suffix to denote hexadecimal number
0b	Prefix to denote binary number
b	Suffix to denote binary number
rA, **r**B	Instruction syntax used to identify a source GPR
rD	Instruction syntax used to identify a destination GPR
REGISTER[FIELD]	Abbreviations for registers are shown in uppercase text. Specific bits, fields, or ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In some contexts, such as signal encodings, an unitalicized x indicates a don't care.
x	An italicized *x* indicates an alphanumeric variable
n	An italicized *n* indicates either: • An integer variable • A general-purpose bitfield unknown
Â¬	NOT logical operator
&	AND logical operator
|	OR logical operator
||	Concatenation, for example, TCR[WPEXT] || TCR[WP]
Signals	
_B	An overbar indicates that a signal is active-low.
lowercase_italics	Lowercase italics is used to indicate internal signals
lowercase_plaintext	Lowercase plain text is used to indicate signals that are used for configuration.
Register access	
Reserved	Ignored for the purposes of determining access type
R/W	Indicates that all non-reserved fields in a register are read/write
R	Indicates that all non-reserved fields in a register are read only
W	Indicates that all non-reserved fields in a register are write only
w1c	Indicates that all non-reserved fields in a register are cleared by writing ones to them

A.5 Related resources

This table shows related resources that may be helpful. Additional literature is published as new processors become available. For current literature, visit www.NXP.com or contact your local FAE.

Table 110. Related resources

Resource	Purpose
Data Sheet	Provides specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations
Reference manual	Provides details about individual implementations
VSPA-4SP ISA-v2.0 Instruction Set Manual	Provides details about Vector Signal Processing Acceleration platform
Chip errata	Provides additional or corrective information for a particular device mask set. Individual errata items are published cumulatively.
Application note	Addresses specific design issues useful to programmers and engineers working with NXP processors

For the ARM modules, refer the following resources:

Table 111. Related resources from ARM

Resource	IP Revision
ARM® Cortex™-M4 Processor Technical Reference Manual	r0p1
CoreLink™ Network Interconnect NIC-301 Technical Reference Manual	r2p3

Appendix B
Revision History

B.1 Substantive changes from revision 0 to revision 1

Substantive changes from revision 0 to revision 1 are as follows:

B.1.2 Memory Map Revision History

Reference	Description
System Memory Map	Updated the System Memory Map table, added row "Private Peripherals"
Endianness	Added text "The Cortex M4 core will execute in Little Endian mode. The VSPA core supports Little Endian exclusively. All registers operate in little endian mode." to this section.

B.1.4 Reset Clocking and Initialization Revision History

Reference	Description
Power-On Reset Sequence Detailed Description	Updated the Power-On Reset Sequence table
Power-On Reset Configuration	Updated the Table.
Reset register descriptions	Added RCW_COMPLETIONR Register.
Core Cluster a Clock Control/Status Register (CLKC1CSR - CLKC2CSR)	In the description of CLKSEL field- Added the bitfield 0111b - Reserved.

B.1.5 Boot Overview Revision History

Reference	Description
	Added this chapter.

B.1.6 Interrupt Assignments Revision History

No substantive changes

B.1.7 ARM Modules Revision History

No substantive changes

B.1.8 Device Configuration and Pin Control Revision History

Reference	Description
Pin Muxing	Changed TX_ACTIVE[1:0] signals to Reserved.

Table continues on the next page...

Table continued from the previous page...

Reference	Description
Pins register descriptions	In PMUXCR0 (PMUXCR0) register- Updated fields GPIO5, GPIO6, GPIO7, and GPIO8.
AXIQ Loopback Capability	Added this new section.

B.1.9 ADC DAC Data Conversion System Revision History

No substantive changes

B.1.10 AXIQ Revision History

Reference	Description
Transmit Interface	Updated the text from "The transmit interface uses a ready signal and a valid signal (only on channels 2-5)." to "The transmit interface uses a ready signal and a valid signal (only on channel 5)."
Test Modes and Scan Enable Mode	Deleted these sections.
Resets	Updated the section- Deleted the text "See the SoC clock controller documentation for more about how to assert these resets after power-up."
Interrupts	Updated the section- Changed AXIQ_R6T5 to AXIQ and VSPA's ext_ruby_go to VSPA's ext_go
Simulation Strategy and Targets	Deleted this section.
Interface Specification	Added this new section.

B.1.11 Access Error Management Revision History

No substantive changes

B.1.12 eDMA Revision History

No substantive changes

B.1.13 Forward Error Correction Unit Revision History

No substantive changes

B.1.14 GPIO Revision History

No substantive changes

B.1.15 I^2C Revision History

No substantive changes

B.1.16 LLCP Revision History

No substantive changes

B.1.17 Message Unit Revision History

No substantive changes

B.1.18 PCI Express Revision History

No substantive changes

B.1.19 SerDes Revision History

Reference	Description
External Signals Description	In Table SerDes Interface Signals- Deleted pin SD_PLLx_TPD sd_pllx_tpd
Top-level SerDes Memory Map	In Table SerDes Memory Map Summary- Updated the description of 0x0020-0x003F to Reserved.

B.1.20 SPI Revision History

No substantive changes

B.1.21 TMU Revision History

No substantive changes

B.1.22 UART Revision History

No substantive changes

B.1.23 VSPA Revision History

No substantive changes

For more information, please visit: https://www.nxp.com

Date of release: 23/01/2025
Document identifier: LA9310RM