

# **HINOC BASE**

## **Technical Manual**

## **Version 0.0.2**

## Revision History

Version	Date	Author	Remarks
0.0.1	2013.04.18	Newsy	Preliminary version
0.0.2	2013.04.28	Newsy	<ul style="list-style-type: none"><li>- Remove section: 9.3 Frame Count</li><li>- Remove DDR_A13 / A14 due to use 512Mbits capacity only.</li><li>- Update all DW_IP offset register information</li><li>- Update Chapter 17, 18</li></ul>

# Contents

1	Introduction.....	7
1.1	Overview.....	7
1.1.1	Introduction .....	7
1.1.2	Features .....	7
1.1.3	Block Diagram Overview .....	8
1.1.4	Bus Architecture Overview .....	8
1.2	Description.....	9
1.2.1	UR0431A IO description.....	9
2	Memory Mapping.....	12
2.1	Introduction.....	12
2.2	Detailed Mapping .....	12
3	Clocks, Reset.....	13
3.1	Introduction .....	13
3.2	Resets diagram .....	14
3.3	Registers .....	14
3.3.1	Registers Summary .....	14
3.3.2	Detail Register Description .....	14
3.4	PLL control setting .....	18
3.5	Programmable Clock Structure.....	20
3.6	Programming Sequence.....	20
4	AHB.....	22
4.1	AHB Overview.....	22
4.2	AHB Mapping.....	22
4.2.1	Register Memory Maps.....	22
4.2.2	Register and Field Descriptions .....	23
5	Direct Memory Access (DMA) .....	27
5.1	Overview.....	27
5.1.1	DMA diagram.....	27
5.2	Functional Description.....	28
5.2.1	Block Flow Controller and Transfer Type .....	28
5.2.2	Basic Interface Definitions .....	28
5.2.3	Software Handshaking .....	28
5.2.4	Setting Up Transfers .....	29
5.2.5	Generating Requests for the AHB Master Bus Interface.....	29
5.3	DMA Register.....	30
5.3.1	Register Memory Maps.....	30
5.3.2	Register and Field Descriptions .....	33
6	Interrupts (DW_ICCTL).....	56
6.1	ICCTL Function Description.....	56
6.1.1	Overview.....	56
6.1.2	IRQ Interrupt Processing.....	56
6.1.3	FIQ Interrupt Processing .....	57
6.2	ICCTL Programming .....	58
6.2.1	Initialization .....	58
6.2.2	Interrupt Service .....	58
6.3	Interrupt Mapping .....	58
6.3.1	Register Memory Maps.....	59
6.3.2	Register and Field Descriptions .....	60
7	DDR2 Memory .....	66
7.1	Overview.....	66
7.2	DDR2 Function Interface.....	66
7.3	Architecture and Function Description.....	66
7.4	DDR2 Programming .....	66

8	7.4.1 DDR2 Register Manual .....	66
8	HiNoC.....	80
8.1	HiNoC Overview.....	80
8.2	HiNoC Environment.....	80
8.3	HiNoC Integration.....	80
8.4	HiNoC Function Description and Data Transfer Bandwidth.....	80
8.5	HiNoC Programming Model.....	80
8.6	HiNoC Registers .....	80
9	10/100 Ethernet Mac.....	81
9.1	10/100 Ethernet Mac Introduction.....	81
	Transmit and Receive FIFOs .....	81
	MAC .....	81
	Transaction Layer (MTL).....	82
	DMA Block .....	83
9.2	DMA controller:.....	83
9.2.1	Initialization .....	84
9.2.2	Transmission .....	85
9.2.3	Reception.....	85
9.2.4	Interrupts.....	86
9.2.5	Err Response .....	86
9.3	MAC Transaction Layer (MTL):.....	86
9.3.1	Transmit Path .....	87
9.3.2	Receive Path.....	87
9.4	MAC: .....	88
9.4.1	Transmission Path .....	88
9.4.2	Reception.....	88
9.5	MAC Management Counter .....	88
9.5.1	PMT Register Map.....	88
9.6	Power Management .....	91
9.6.1	PMT Register Map.....	91
9.6.2	Remote Wake-Up Frame Filter Register.....	91
9.7	Register .....	92
9.7.1	GMA Register Map.....	93
	Register Descriptions .....	130
10	Timers.....	214
10.1	General-Purpose Timer .....	214
10.1.1	Function Description.....	214
10.1.2	Timers Registers.....	215
10.2	Watchdog Timer .....	225
10.2.1	Features.....	225
10.2.2	Function Description.....	226
10.2.3	Function Programming .....	227
10.2.4	WDT Registers.....	227
11	UART .....	233
11.1	UART Features.....	233
11.2	UART Function Description.....	233
11.2.1	UART (RS232) Serial Protocol .....	233
11.2.2	FIFO Support .....	233
11.2.3	Back-to-Back Character Stream Transmission.....	234
11.2.4	Interrupts .....	234
11.2.5	Auto Flow Control .....	235
11.3	UART Programming.....	235
11.3.1	UART Transmit Programming Flowchart.....	235
11.3.2	UART Receive Programming Flowchart .....	236
11.4	UART Registers.....	236
11.4.1	Register Memory Maps .....	236

12	11.4.2 Register and Field Descriptions .....	238
	I2C .....	250
12.1	I2C Overview .....	250
12.2	I2C Function Description.....	250
12.2.1	I2C Bus Terms .....	250
12.2.2	I2C Behavior.....	250
12.2.3	Tx FIFO Management and START, STOP and RESTART Generation.....	252
12.2.4	Operation Modes.....	252
12.2.5	Spike Suppression.....	253
12.2.6	IC_CLK Frequency Configuration .....	253
12.2.7	SDA Hold Time .....	253
12.3	I2C Programming .....	254
12.3.1	I2C Master Flowchart.....	255
12.3.2	I2C Master in Standard or Fast Speed Mode Flowchart .....	256
12.4	I2C Register .....	256
12.4.1	Register Memory Maps .....	256
12.4.2	Registers and Field Descriptions.....	261
13	Boot SPI.....	288
13.1	Features .....	288
13.2	Dirrect Access Mode .....	288
13.2.1	Description.....	288
13.2.2	Setup Direct Access Mode .....	288
13.3	Registers.....	289
13.3.1	Registers Memory Mapping.....	289
13.3.2	Register Description.....	289
14	SSI .....	291
14.1	Function Description.....	291
14.1.1	Clock Ratio .....	291
14.1.2	Transmit and Receive FIFO Buffers .....	291
14.1.3	Interrupts .....	292
14.1.4	Transfer Modes .....	293
14.1.5	Master SPI Serial Transfers.....	293
14.2	SSI Registers .....	295
14.2.1	Register Memory Maps .....	295
14.2.2	Registers and Field Descriptions.....	302
15	GPIO.....	324
15.1	GPIO Modules.....	324
15.2	Function Description.....	324
15.2.1	Data and Control Flow .....	324
15.2.2	Interrupts .....	326
15.3	GPIO Register Manual.....	331
15.3.1	Bus Interface.....	331
15.3.2	GPIO Register Mapping.....	331
15.3.3	Register and Field Descriptions .....	333
16	Initialization .....	344
16.1	Initialization Overview .....	344
16.2	Pre-initialization .....	344
16.3	Power, Clock, and Reset Sequence on Power-up .....	344
16.4	Device Booting by ROM code .....	345
17	Configuration .....	347
17.1	Function Configuration Overview .....	347
17.2	I/O Configuration .....	350
17.3	I/O Multiplexing .....	352
17.4	Debug Module .....	352
18	Electrical Characteristics .....	353
18.1	DC Characteristics .....	353

---

18.1.1	GPIO groups DC characteristics .....	353
18.1.2	DDR IO DC Electrical characteristics.....	353
18.2	AC Electrical Characteristics.....	355
18.2.1	DDR IO groups AC characteristics .....	355
19	Package Specifications .....	357
20	Memory Usage.....	358

# 1 Introduction

## 1.1 Overview

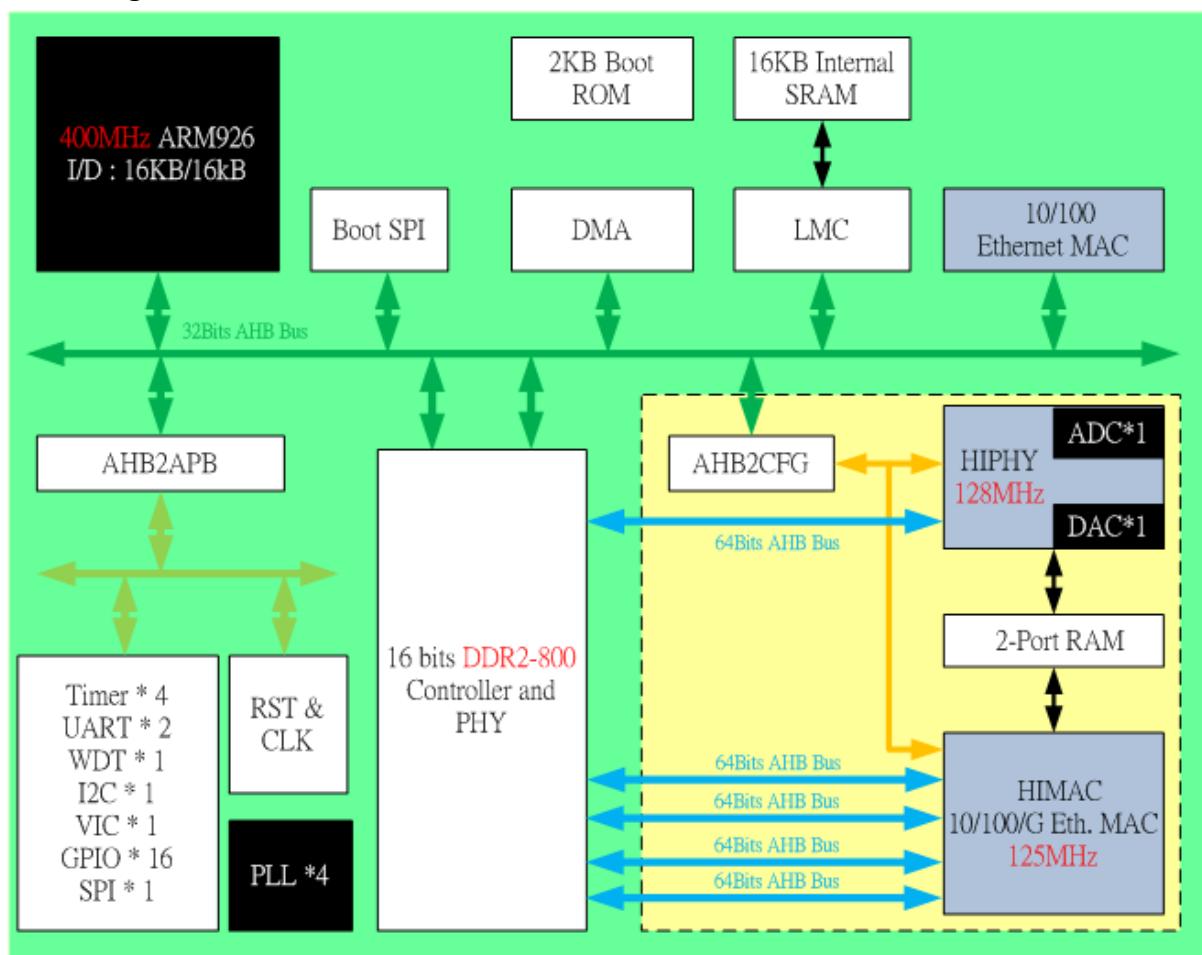
### 1.1.1 Introduction

The chip is a highly integrated ARM-based SOC solution for cable base band application with the high resolution ADC/DAC IPs. It provides various interfaces such as DDR2, MII, GMII, PCI, and IDE. Based on TSMC 65um LP technology process, the operating frequency of the embedded ARM926E core and the AHB bus is targeted at 400MHz and 200MHz respectively, and supports DDR2-800 Mbps interface.

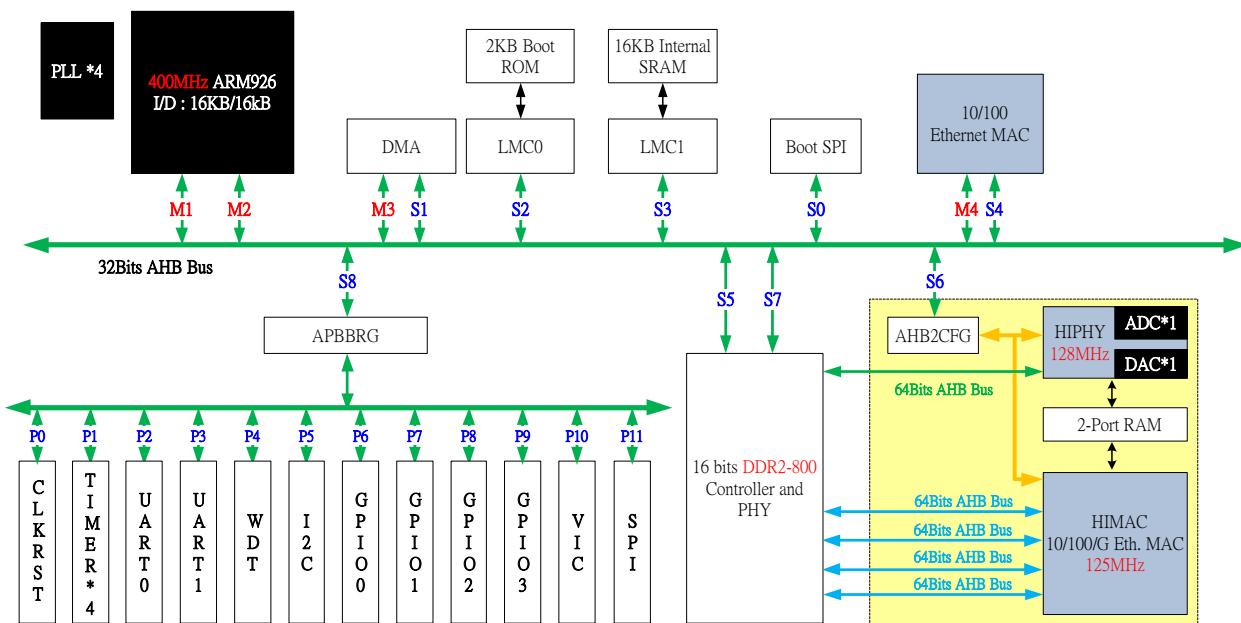
### 1.1.2 Features

- Embedded ARM926EJS Core (with 16KB/16KB I/D caches and 16k/16k I/D TCM)
- 10/100/1000M GMII 接口，数据上下行带宽总和到 160M (平滑带宽上下行综合为 320M, 峰值带宽为 2G)
- 10/100M MII 接口 (管理通道用) (改为 MII 接口)
- DDR2-800Mbps Interface, update to 4Gbyte capacity
- IGMP Snooping protocol
- SNMP protocol (HB 充当 SNMP Agent 和 SNMP 被管理设备, HM 充当 SNMP 被管理设备)
- 2 个 UART 接口、1 个 I2C 接口、1 个 SPI 接口、1 个 SPI BOOT 接口、16 个 GPIO 接口
- SPI flash memory update to 32Mbytes
- Hinoc PHY：
  - 采用正交幅度调制(QAM)，并根据同轴电缆不同的噪声、衰减、反射等情况自适应使用 QPSK、8QAM、16QAM、32QAM、64QAM、128QAM、256QAM、512QAM、1024QAM 的调制格式。
  - 为了避免由于多径效应引起的码间串扰问题，提高信道的利用率，选用了正交频分复用(OFDM)技术传输数据。
  - 在纠错码方面，采用高编码效率、低实现复杂度的 BCH 码。可以在 16MHz 的信道带宽上实现 100Mb/s 以上的物理层传输速率，其有效带宽的频率利用率可达到 7bit/s/Hz。
- Hinoc MAC：
  - 支持 QOS
  - 支持 IGMP IGMPv2 和 MLDv1。
  - 支持 HINOC 协议的接纳及链路维护功能；
  - 支持 HINOC 协议的带宽分配功能；
  - 上下行带宽支持到 160Mhz。
  - 根据 vlan 进行优先级调度，支持 8 个优先级，每个端口 4 个队列
  - HB 可以支持 32 个 HM；
  - 可以配置每个 HM 的带宽。
  - 认证 HM 合法性；
  - 可以使能/不使能节点的流量；
  - MAC 地址学习 1k 个地址，需要满足每秒学习 64 个 mac 地址。
  - 组播支持 128 个组播组。
  - 支持地址老化。老化时间可配。

### 1.1.3 Block Diagram Overview



### 1.1.4 Bus Architecture Overview



## 1.2 Description

### 1.2.1 UR0431A IO description

Signal	I/O	Bit	Voltage	Description
<b>JTAG</b>				
TCK	I	1	3.3	JTAG Test Clock In. This clock controls the updates to the TAP Controller and the shifts through the Instruction register or selected data registers. Connect to GND for normal operation.
TDI	I	1	3.3	JTAG Test Data In. Serial test data is input on this pin and is shifted into the instruction or data register. This input is sampled on the rising edge of TCK. Connect to GND for normal operation.
RTCK	O	1	3.3	JTAG Test Clock Out. This clock controls the output data. Leave unconnected for normal operation.
TDO	IO	1	3.3	JTAG Test Data Out. The AR7400 outputs serial test data on this pin from the instruction or data register. This signal changes on the rising edge of RTCK. This pin is also sampled during TRST# – if TDO is sampled high during TRST, then the top-level DFT TAP controller is selected to be connected to the JTAG I/O pads. If TDO is sampled low during TRST, then the ARMs TAP controller is connected to the JTAG I/O pads. Leave unconnected for normal operation.
TMS	I	1	3.3	JTAG Test Mode Select. This input is the control signal for the TAP Controller. It is sampled on the rising edge of TCK. Connect to GND for normal operation.
TRST_N	I	1	3.3	JTAG Test Reset. This signal is asserted asynchronously to reset the TAP Controller, instruction register. Connect to GND for normal operation.
<b>Reset</b>				
RESET_N	I	1	3.3	Chip reset pin. Active low. (From power management IC)
PHY_RESET	O	1	3.3	External PHY reset. Active low.(SW register output)
<b>Clock</b>				
OSCIN_32M	I	1	3.3	Input clock. 32MHz.
OSCIN_25M	I	1	3.3	Input clock. 25Mhz.
<b>DDR</b>				
DDR_A12~0	O	13	1.8	DDR2 address
DDR_BA1~0	O	2	1.8	DDR2 bank Select.
DDR_CS_N	O	1	1.8	DDR2 chip Select.
DDR_RAS_N	O	1	1.8	DDR2 row Address Strobe.
DDR_CAS_N	O	1	1.8	DDR2 column Address Strobe.
DDR_WE_N	O	1	1.8	DDR2 write Enable.
DDR_CKE	O	1	1.8	DDR2 clock Enable. If low, this signal indicates to the DDR2 to enter the power-down state.
DDR_DQM1~0	O	2	1.8	DDR2 data Mask. DDR_DQM0 corresponds to DDR_DQ[7:0], others [15:8]
DDR_DQ15~0	IO	16	1.8	DDR2 data bus
DDR_DQS0P	IO	1	1.8	DDR2 data Strobe. For lower byte.
DDR_DQS0N	IO	1	1.8	DDR2 data Strobe. For lower byte.
DDR_DQS1P	IO	1	1.8	DDR2 data Strobe. For upper byte.
DDR_DQS1N	IO	1	1.8	DDR2 data Strobe. For upper byte.
DDR_CLKP	O	1	1.8	DDR2 differential Clock.
DDR_CLKN	O	1	1.8	DDR2 differential Clock.
DDR_ODT	O	1	1.8	DDR2 on die termination
<b>10/100 MAC MII</b>				
CPU_TX_CLK	I	1	3.3	发送和接收同步时钟信号，是由外部的时钟源提供。
CPU_TXD3~0	O	4	3.3	MII transmitter data bus
CPU_TX_EN	O	1	3.3	MII transmitter enable signal

Signal	I/O	Bit	Voltage	Description
CPU_TX_ER	O	1	3.3	MII transmitter error signal
CPU_RXD3~0	I	4	3.3	MII receiver data bus
CPU_RX_CLK	I	1	3.3	MII receiver clock
CPU_RX_ER	I	1	3.3	MII receiver error signal
CPU_RX_DV	I	1	3.3	MII receiver enable signal
CPU_CRS	I	1	3.3	MII Carrier Sense
CPU_COL	I	1	3.3	MII Collision Detec
CPU_MDC	O	1	3.3	MII Management Data Clock. The MAC core sources MDC as the timing reference for transfer of information on the MDI/MDO signals. MDC signal has no maximum high or low times. MDC minimum high and low times are 160 ns each, and the minimum period for MDC is 400 ns.
CPU_MDIO	IO	1	3.3	MII Management Data In/Out. This is the data input signal from the PHY controller. The PHY drives the Read Data synchronously with respect to the MDC clock during the read cycles. This is also the data output signal from the MAC core that drives the control information during the Read/Write cycles to the PHY controller. The MAC core drives the MDO signal synchronously with respect to the MDC.
<b>HIMAC GMII</b>				
GMTX_GCLK	O	1	3.3	当 EMAC 工作在1000M 模式下的发送时钟，为125MHz
GMTX_CLK	I	1	3.3	当 EMAC 工作在 10/100M 模式下的发送时钟，为 25Mhz
GMTX_D7~0	O	8	3.3	RGMII transmitter data bus
GMTX_ER	O	1	3.3	RGMII transmitter error signal
GMTX_EN	O	1	3.3	RGMII transmitter enable signal
GMRX_CLK	I	1	3.3	RGMII receiver clock
GMRX_D7~0	I	8	3.3	RGMII receiver data bus
GMRX_DV	I	1	3.3	RGMII receiver enable signal
GMRX_ER	I	1	3.3	RGMII receiver error signal
GMRX_COL	I	1	3.3	RGMII Collision Detec
GMRX_CRS	I	1	3.3	RGMII Carrier Sense
<b>Nor Flash SPI</b>				
SPI_MEM_CS	O	1	3.3	SPI Chip Select. SPI_CS# is the chip select used to enable additional SPI devices.
SPI_MEM_CLK	O	1	3.3	SPI Clock.
SPI_MEM_DI	I	1	3.3	SPI Data In. The SPI_DI is used to transfer serial data into the ASIC.
SPI_MEM_DO	O	1	3.3	SPI Data Out. SPI_DO is used to transfer serial data out of the ASIC.
SPI_MEM_WP	O	1	3.3	Write protection
<b>SPI</b>				
SPI_CLK	O	1	3.3	SPI Clock.
SPI_CS	O	1	3.3	SPI Chip Select. SPI_CS# is the chip select used to enable additional SPI devices.
SPI_DI	I	1	3.3	SPI Data In. The SPI_DI is used to transfer serial data into the ASIC.
SPI_DO	O	1	3.3	SPI Data Out. SPI_DO is used to transfer serial data out of the ASIC.
<b>GPIO</b>				
GPIO15~0	IO	16	3.3	General Purpose I/O. GPIO1 could be output of Sys_Test_CLK divided by 16. GPIO2 could be output of Mem_Test_CLK divided by 16. GPIO3 could be output of HiMAC_Test_CLK divided by 4. GPIO5 could be output of HiPY_Test_CLK divided by 4. GPIO0/4/8/12 could be input for external interrupt individually.
<b>UART</b>				
UART_TX0	O	1	3.3	UART0 transmit data output
UART_RX0	I	1	3.3	UART0 receive data input

Signal	I/O	Bit	Voltage	Description
UART_TX1	O	1	3.3	UART1 transmit data output
UART_RX1	I	1	3.3	UART1 receive data input
<b>I2C</b>				
I2C_SCLK	O	1	3.3	Serial clock output of I2C bus (control RF)
I2C_SDA	IO	1	3.3	Serial data input/output of I2C bus (control RF)
<b>System Application</b>				
TESTM3	I	1	3.3	Function modes selection 4'b0000: From SPI - HM ARMJTAG 4'b0001: From SPI - HM Non ARMJTAG 4'b0010: From SPI - HB ARMJTAG 4'b0011: From SPI - HB Non ARMJTAG 4'b0100: From InternalROM - HM ARMJTAG 4'b0101: From InternalROM - HM Non ARMJTAG 4'b0110: From InternalROM - HB ARMJTAG 4'b0111: From InternalROM - HB Non ARMJTAG Others: Debugging modes
TESTM2	I	1	3.3	
TESTM1	I	1	3.3	
TESTM0	I	1	3.3	
<b>HIPHY AFE – ADC</b>				
RXQN	I	1	2.5	Q channel analog base-band data negative input for receive mode
RXQP	I	1	2.5	Q channel analog base-band data positive input for receive mode
RXIN	I	1	2.5	I channel analog base-band data negative input for receive mode
RXIP	I	1	2.5	I channel analog base-band data positive input for receive mode
AVDDREFI	I	1	1.2	I channel Positive reference
AGNDREFI	I	1	-	I channel Negative reference
AVDDREFQ	I	1	1.2	Q channel Positive reference
AGNDREFQ	I	1	-	Q channel Negative reference
VCMINREF	O	1	2.5	Output reference voltage to be taken as reference for the ADC input signal common mode level
<b>HIPHY AFE – DAC</b>				
TXQN	O	1	2.5	Q channel analog base-band data negative output for transmit mode
TXQP	O	1	2.5	Q channel analog base-band data positive output for transmit mode
TXIN	O	1	2.5	I channel analog base-band data negative output for transmit mode
TXIP	O	1	2.5	I channel analog base-band data positive output for transmit mode
Iref	IO	1	2.5	Reference Current. To be used with an External Reference Resistor.
Vdref	IO	1	2.5	Reference voltage decoupling
AFE_POWERDN	O	1	3.3	Reference or Bias for HINOC power amplifier
AFE_TXEN	O	1	3.3	TX output Port
AFE_RXEN	O	1	3.3	RX output Port
AFE_GAIN4~0	IO	5	3.3	The control port of AFE module
PD_IN	I	1	3.3	
PD_OUT	O	1	3.3	

## 2 Memory Mapping

### 2.1 Introduction

Memory map is composed of memory space (SPI Nor Flash and DDR2), register space (Internal SRAM, control register, etc.), and I/O space (Peripherals) which are shared among the chip modules. The DDR2 interface and SPI Nor Flash interface are dedicated to memory connection. Internal SRAM allows sharing of critical resource such as memories between CPU subsystem and system DMA. I/O space represents a generic address space used to access several sub modules used to control peripherals and manage their accesses.

### 2.2 Detailed Mapping

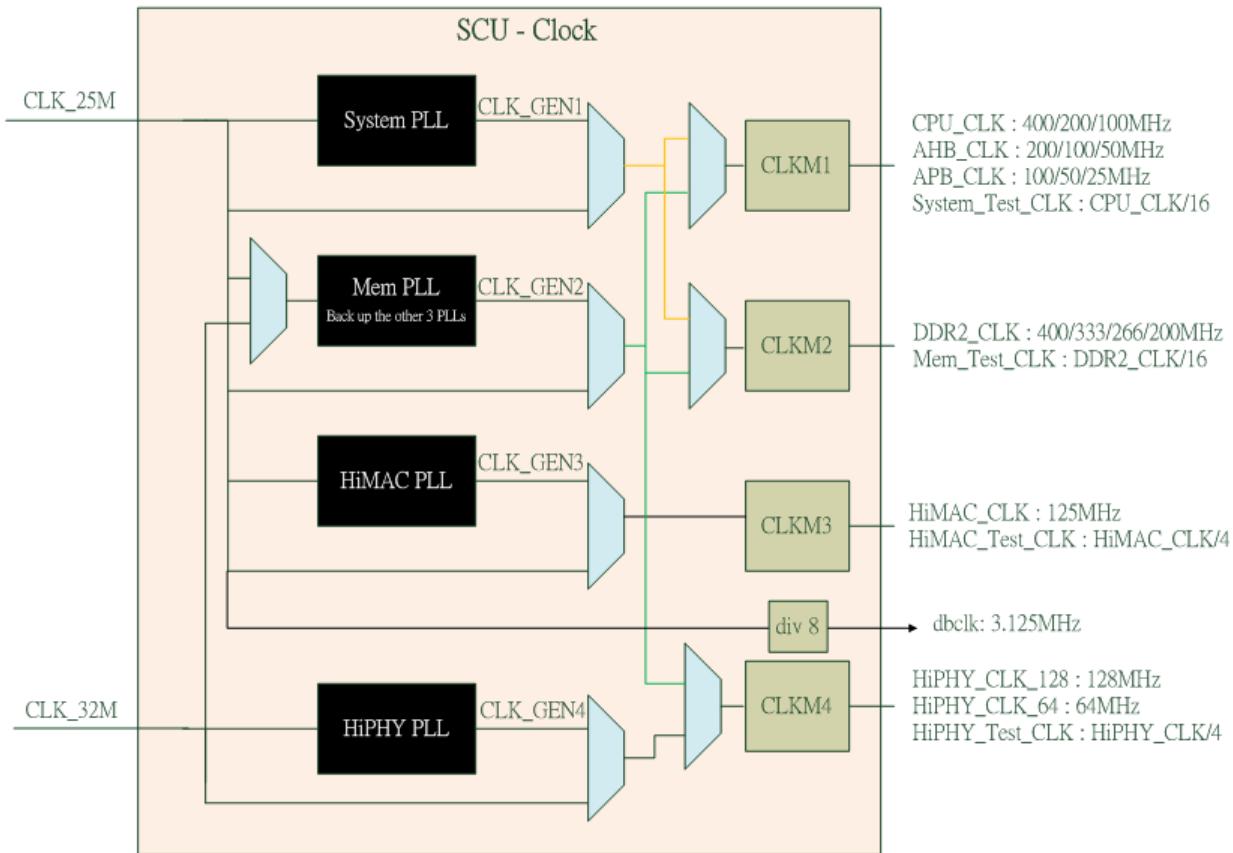
Name	Base Addr. (Normal)	Base Addr. (SPI)	Addr. Range	Description
<b>AHB IP</b>				
Internal ROM	0x0000_0000	0x0400_0000	0x0001_0000	
SPI boot	0x0400_0000	0x0000_0000	0x0400_0000	SPI NOR Flash memory, 64Mb
Internal RAM	0x0800_0000	0x0800_0000	0x0001_0000	
DDR data	0x1000_0000	0x1000_0000	0x1FFF_0000	
Arbiter	0x8000_0000	0x8000_0000	0x0001_0000	
DMA	0x8001_0000	0x8001_0000	0x0001_0000	
10/100 MAC	0x8002_0000	0x8002_0000	0x0001_0000	
DDR reg	0x8003_0000	0x8003_0000	0x0001_0000	
HiNOC	0x8004_0000	0x8004_0000	0x0002_0000	
APB Bridge	0x9000_0000	0x9000_0000	0x0010_0000	
<b>Reserved</b>				

APB IP	Start Address	Addr. Range	Description
CLKRST	0x9000_0000	0x0000_1000	PLL control and clock mux selection
TIMER	0x9001_0000	0x0000_1000	
UART0	0x9002_0000	0x0000_1000	
UART1	0x9003_0000	0x0000_1000	
WDT	0x9004_0000	0x0000_1000	
I2C	0x9005_0000	0x0000_1000	
GPIO_INS0	0x9006_0000	0x0000_1000	
GPIO_INS1	0x9007_0000	0x0000_1000	
GPIO_INS2	0x9008_0000	0x0000_1000	
GPIO_INS3	0x9009_0000	0x0000_1000	
VIC	0x900a_0000	0x0000_1000	
SPI	0x900b_0000	0x0000_1000	
<b>Reserved</b>			

## 3 Clocks, Reset

### 3.1 Introduction

The device clocks/reset/power architecture includes the functional clocks, interface clocks, reset distribution and wake-up requests for the platform, and peripherals. The functional and interface clocks are all generated from two input clocks: 25-MHz and 32-MHz. The clocks are generated



The SCU is an APB slave module that is designed for generating all of the internal and system clocks, resets of chip and power domain control. SCU generates system clock from PLL output clock or external clock source, and generate system reset from external power-on-reset or watchdog timer reset. The SCU also provide power management mechanism for system power saving and general control bit.

The SCU performs the following functions:

- \* Controls four on-chip PLLs that generate CPU\_CLK, DDR2, HiMAC\_CLK and HiPHY\_CLK.
- \* Manages the clocks and resets distributed to the platform and to certain peripherals.
- \* Performs the transition between the power modes: awake, half speed and off.

The System PLL is located in the platform; it converts the input clock (25 MHz) to a high-frequency clock (maximum 800 MHz/1000MHz), which is then distributed within the platform and to the various on-chip peripherals.

The Mem PLL is located in the platform; it converts the input clock (25 MHz) to a high-frequency clock (maximum 800 MHz/1000MHz) or converts the input clock (32 MHz) to a high-frequency clock (maximum 256MHz), which is then distributed within the various on-chip peripherals or HiPHY PLL.

The HiMAC PLL is located in the platform; it converts the input clock (25 MHz) to a high-frequency clock (maximum 250 MHz), which is then distributed within the platform and to HiNoC peripherals.

The HiPHY PLL is located in the platform; it converts the input clock (32 MHz) to a high-frequency clock (maximum 256 MHz), which is then distributed within the platform and to HiNoC peripherals.

### 3.2 Resets diagram

#### Reset overview

Will update soon

### 3.3 Registers

This section describes the control/status registers of the design.

#### 3.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SCU_SYSPLL_CON	0x00	32-bit	0x000185f8	SYS PLL output frequency control register
SCU_MEMPLL_CON	0x04	32-bit	0x000185f8	MEM PLL output frequency control register
SCU_HIMACPLL_CON	0x08	32-bit	0x0001853a	HiMAC PLL output frequency control register
SCU_HIPHYPLL_CON	0x0C	32-bit	0x00304782	HiPHY PLL output frequency control register
SCU_CLKM0_CON	0x10	32-bit	0x0	CLKM0 clock divider and select control register
SCU_CLKM1_CON	0x14	32-bit	0x0	CLKM1 clock divider and select control register
SCU_CLKM2_CON	0x18	32-bit	0x0	CLKM2 clock divider and select control register
SCU_CLKM3_CON	0x1C	32-bit	0x0	CLKM3 clock divider and select control register
SCU_SFTRST_CON	0x20	32-bit	0x0	Soft reset control register
SCU_DEBUG_CON	0x24	32-bit	0x0	PLL debug and misc. control register

#### 3.3.2 Detail Register Description

##### 3.3.2.1 SCU\_SYSPLL\_CON

Address: Base Addr + 0x00

SYS PLL output frequency control register

Bit	Attr	Reset Value	Description
31:18	-	-	Reserved
17	RO	0x0	SYS PLL lock status 1: locked 0: busy
16	RW	0x1	SYS PLL band selection 1: high band 0: low band
15	RW	0x1	SYS PLL power down control 1: power down
14:10	RW	0x1	SYS PLL CLKR factor control
9:3	RW	0x3F	SYS PLL CLKF factor control
2:1	RW	0x0	SYS PLL CLKOD factor control
0	RW	0x0	SYS PLL BP mode control 1: bypass mode

##### 3.3.2.2 SCU\_MEMPLL\_CON

Address: Base Addr + 0x04

MEM PLL output frequency control register

Bit	Attr	Reset Value	Description
31	RW	0x0	MEM PLL ref clock source select 0: CLK_25M 1: CLK_32M
30:18	-	-	Reserved
17	RO	0x0	MEM PLL lock status 1: locked 0: busy

16	RW	0x1	MEM PLL band selection 1: high band 0: low band
15	RW	0x1	MEM PLL power down control 1: power down
14:10	RW	0x1	MEM PLL CLKRF factor control
9:3	RW	0x3F	MEM PLL CLKF factor control
2:1	RW	0x0	MEM PLL CLKOD factor control
0	RW	0x0	MEM PLL BP mode control 1: bypass mode

**3.3.2.3 SCU\_HIMACPLL\_CON**

Address: Base Addr + 0x08

HIMAC PLL output frequency control register

Bit	Attr	Reset Value	Description
31:18	-	-	Reserved
17	RO	0x0	HIMAC PLL lock status 1: locked 0: busy
16	RW	0x1	HIMAC PLL band selection 1: high band 0: low band
15	RW	0x1	HIMAC PLL power down control 1: power down
14:10	RW	0x1	HIMAC PLL CLKRF factor control
9:3	RW	0x27	HIMAC PLL CLKF factor control
2:1	RW	0x1	HIMAC PLL CLKOD factor control
0	RW	0x0	HIMAC PLL BP mode control 1: bypass mode

**3.3.2.4 SCU\_HIPHYPLL\_CON**

Address: Base Addr + 0x0C

HIPHY PLL output frequency control register

Bit	Attr	Reset Value	Description
31:23	-	-	Reserved
22	RO	0x0	HIPHY PLL lock status 1: locked 0: busy
21	RW	0x1	HIPHY PLL band selection (VCORANGE) 1: high band 0: low band
20	RW	0x1	HIPHY PLL power down control 1: power down
19:14	RW	0x1	HIPHY PLL input frequency division factor (N) control
13:6	RW	0x1e	HIPHY PLL feedback multiplication factor (M) control
5:1	RW	0x1	HIPHY PLL post-divider division factor (P) control
0	RW	0x0	HIPHY PLL BP mode control 1: bypass mode

**3.3.2.5 SCU\_CLKM0\_CON**

Address: Base Addr + 0x10

CLKM0 clock divider and select control register

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:6	RW	0x0	Control APB clock frequency

			00: ahb_clk:apb_clk = 1:1 01: ahb_clk:apb_clk = 2:1 10: ahb_clk:apb_clk = 4:1 11: ahb_clk:apb_clk = 8:1
5:4	RW	0x0	Control AHB clock frequency 00: arm_clk:ahb_clk = 1:1 01: arm_clk:ahb_clk = 2:1 10: arm_clk:ahb_clk = 4:1 11: arm_clk:ahb_clk = 8:1
3:2	RW	0x0	Control ARM clock frequency 00: M0_clk:arm_clk = 1:1 01: M0_clk:arm_clk = 2:1 10: M0_clk:arm_clk = 4:1 11: M0_clk:arm_clk = 8:1
1	RW	0x0	Control M0_clk source 0: SYSPLL_clk 1: MEMPLL_clk
0	RW	0x0	SYSPLL_clk source control 0: CLK_25M 1: SYSPLL output (CLK_GEN0)

### 3.3.2.6 SCU\_CLKM1\_CON

Address: Base Addr + 0x14

CLKM1 clock divider and select control register

Bit	Attr	Reset Value	Description
31:4	-	-	Reserved
3:2	RW	0x0	Control DDR2 clock frequency 0: M1_clk:ddr2_clk = 1:1 1: M1_clk:ddr2_clk = 2:1 2: M1_clk:ddr2_clk = 4:1
1	RW	0x0	Control M1_clk source 0: SYSPLL_clk 1: MEMPLL_clk
0	RW	0x0	MEMPLL_clk source control 0: CLK_25M 1: MEMPLL output (CLK_GEN1)

### 3.3.2.7 SCU\_CLKM2\_CON

Address: Base Addr + 0x18

CLKM2 clock divider and select control register

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved
1	RW	0x0	Control HIMAC clock frequency 0: M2_clk:himac_clk = 1:1 1: M2_clk:himac_clk = 2:1
0	RW	0x0	M2_clk source control 0: CLK_25M 1: HIMACPLL output (CLK_GEN2)

### 3.3.2.8 SCU\_CLKM3\_CON

Address: Base Addr + 0x1C

CLKM3 clock divider and select control register

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	Control HIPHY clock frequency

			0: M3_clk:hiphy_clk = 1:1 1: M3_clk:hiphy_clk = 2:1
1	RW	0x0	Control M3_clk source 0: HIPHYPLL_clk 1: MEMPLL_clk
0	RW	0x0	HIPHYPLL_clk source control 0: CLK_32M 1: HIPHYPLL output (CLK_GEN3)

### 3.3.2.9 SCU\_SOFTRST\_CON

Address: Base Addr + 0x20

Internal soft reset control register

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved
25	RW	0x0	DMA controller reset request. 0: soft reset in-active 1: soft reset active
24	RW	0x0	ROM controller reset request 0: soft reset in-active 1: soft reset active
23	RW	0x0	SRAM controller reset request. 0: soft reset in-active 1: soft reset active
22	RW	0x0	Boot SPI controller reset request 0: soft reset in-active 1: soft reset active
21	RW	0x0	10/100 MAC controller reset request. 0: soft reset in-active 1: soft reset active
20	RW	0x0	HINOC reset request 0: soft reset in-active 1: soft reset active
19	RW	0x0	DDR2 controller register port reset request. 0: soft reset in-active 1: soft reset active
18	RW	0x0	DDR2 controller reset request 0: soft reset in-active 1: soft reset active
17	RW	0x0	HIMAC reset request. 0: soft reset in-active 1: soft reset active
16	RW	0x0	HIPHY controller reset request 0: soft reset in-active 1: soft reset active
15	RW	0x0	Interrupt controller reset request. 0: soft reset in-active 1: soft reset active
14	RW	0x0	GPIO0 controller reset request 0: soft reset in-active 1: soft reset active
13	RW	0x0	GPIO1 controller reset request. 0: soft reset in-active 1: soft reset active
12	RW	0x0	GPIO2 controller reset request 0: soft reset in-active

			1: soft reset active
11	RW	0x0	GPIO3 controller reset request. 0: soft reset in-active 1: soft reset active
10	RW	0x0	I2C controller reset request 0: soft reset in-active 1: soft reset active
9	RW	0x0	SPI controller reset request. 0: soft reset in-active 1: soft reset active
8	RW	0x0	UART0 controller reset request 0: soft reset in-active 1: soft reset active
7	RW	0x0	UART1 controller reset request. 0: soft reset in-active 1: soft reset active
6	RW	0x0	Watchdog controller reset request 0: soft reset in-active 1: soft reset active
5	RW	0x0	Timer controller bus reset request. 0: soft reset in-active 1: soft reset active
4	RW	0x0	Timer1 counter reset request 0: soft reset in-active 1: soft reset active
3	RW	0x0	Timer2 counter reset request. 0: soft reset in-active 1: soft reset active
2	RW	0x0	Timer3 counter reset request 0: soft reset in-active 1: soft reset active
1	RW	0x0	Timer4 controller reset request. 0: soft reset in-active 1: soft reset active
0	RW	0x0	PHY_RESET output reset request 0: soft reset in-active 1: soft reset active

### 3.3.2.10 SCU\_DEBUG\_CON

Address: Base Addr + 0x24

PLL debug and misc. control register

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved
5:2	RO	0x0	TESTM3- TESTM0 input pin values
1	RW	0x0	ARM debug synchronize logic bypass control 0: disable 1: enable
0	RW	0x0	PLL test clock output control 0: disable 1: enable

## 3.4 PLL control setting

### 3.4.1.1 System PLL / Mem PLL / HiMAC PLL control setting

For **System PLL**, **Mem PLL** and **HiMAC PLL**, the output frequency is calculated according to following table and formulas:

Parameter	SYM	Condition	min(MHz)	max(MHz)
Comparison Freq	Fref	Fref = Fin / NR	10	50
Input clock Freq	Fin	Fin = NR * Fref	10	400
Output clock Freq	Fout	Fout = Fvco / OD	High-band	62.5
			Low-band	37.5
VCO operating Range	Fvco	Fvco = Fref * NF	High-band	500
			Low-band	300
				600

PD	BP	OD	Description	Fout	BS	Description
0	0	0	Normal mode	Fref * NF	0	low band
		1		Fref * NF / 2	1	High band
		2		Fref * NF / 4		
		3		Fref * NF / 8		
0	1	X	Bypass	Fin		
1	x	X	Power down	0		

For example, to obtain a 800MHz output with an input frequency of 25MHz:

$$\text{Fin} = 25\text{MHz}$$

Select normal mode in high band: BS = 1

$$\text{Fref} = \text{Fin} / \text{NR} = 25\text{MHz} / 1 = 25\text{MHz}$$

$$\text{Fout} = \text{Fvco} / \text{OD} \Rightarrow 800\text{MHz} = \text{Fvco}/1 \Rightarrow \text{Fvco} = 800\text{MHz}.$$

$$\text{Fvco} = 800\text{MHz} = \text{Fref} * \text{NF} \Rightarrow 800\text{MHz} = 25\text{MHz} * \text{NF} \Rightarrow \text{NF} = 64$$

SYS/MEM/HIMAC PLL Configuration								Ref	Fvco
Fin (MHz)	Fout (MHz)	BS	BP	PD	OD[1:0]	F[6:0]	R[4:0]		
25	800	1	0	0	0	63	1	12.5	800
25	400	1	0	0	1	63	1	12.5	800
25	200	1	0	0	2	63	1	12.5	800
25	250	1	0	0	1	39	1	12.5	500

### 3.4.1.2 HiPHY PLL control setting

The VCO oscillating frequency is a function of the input reference frequency and of the multiplication /division ratios. It can be determined in the following way:

$$\text{Fvco} = ((M+2)/(N+1)) * \text{Fclkin}$$

where M is the Feedback Multiplication Ratio and N is the Input Frequency Division Ratio.  
However, some limits apply:

$$60\text{MHz} \geq (\text{Fclkin}/(N+1)) \geq 5\text{MHz}$$

For VCORANGE = 0

$$200\text{MHz} \leq \text{Fvco} (= \text{Fclkin} * (M+2)/(N+1)) \leq 500\text{MHz}$$

For VCORANGE = 1

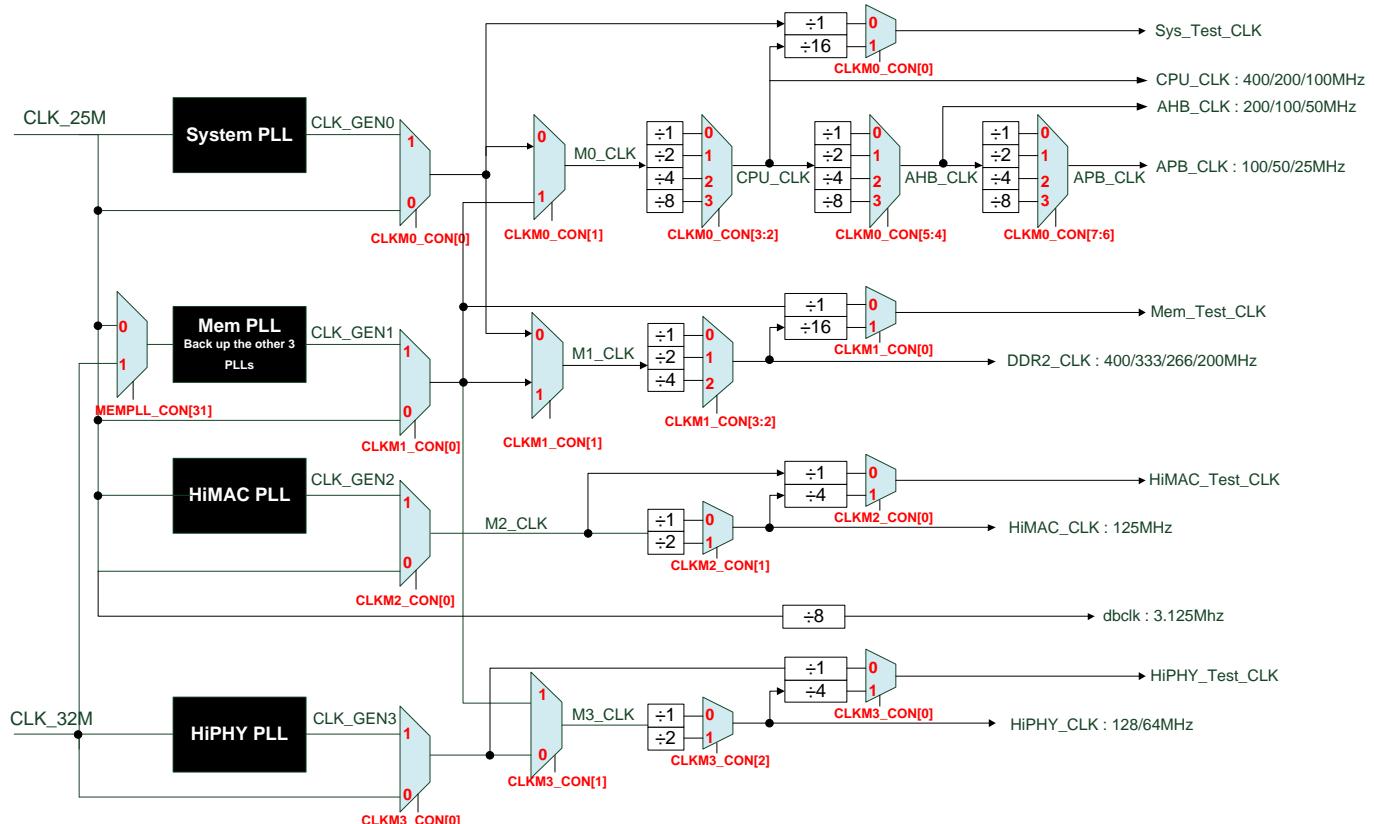
$500\text{MHz} \leq F_{vco} (=F_{clkin} \cdot (M+2)/(N+1)) \leq 1000\text{MHz}$

Finally, the output clock frequency is derived from a programmable division of the VCO frequency:

$$F_{clkout1} = (1/(P+1)) * F_{vco}$$

HIPHY PLL Configuration								Fref	Fvco
Fin (MHz)	Fout (MHz)	VCORANGE	BP	PD	P[4:0]	M[7:0]	N[5:0]		
32	256	1	0	0	1	30	1	16	512

### 3.5 Programmable Clock Structure



### 3.6 Programming Sequence

After system power-on, SCU is in bypass mode, i.e., all PLLs are powered-down (corresponding SCU\_PLL\_CON power down bit set to "0x1"), clocks outputs are coming from input oscillators (CLK\_25M & CLK\_32M). This section provides a programming sequence example for System to switch from power-on "bypass mode" to "normal mode".

#### 1. Power up PLLs

- Configure the SCU\_SYSPLL\_CON registers (band/CLKR/CLKF/CLKOD/PLLBP) to set proper values for desired PLL\_GEN0 output frequency (800Mhz), PLL power down bit should be set to "0x1".
- Configure the SCU\_MEMPLL\_CON registers (band/CLKR/CLKF/CLKOD/PLLBP) to set proper values for desired PLL\_GEN1 output frequency(800Mhz), PLL power down bit should be set to "0x1".
- Configure the SCU\_HIMACPLL\_CON registers (band/CLKR/CLKF/CLKOD/PLLBP) to set proper values for desired PLL\_GEN2 output frequency (250Mhz), PLL power down bit should be set to "0x1".
- Configure the SCU\_HIPHYPLL\_CON registers (VCORANGE/M/N/P/BP) to set proper values for desired PLL\_GEN3 output frequency (256Mhz), PLL power down bit should be set to "0x1".
- Power-up SYSPLL by clear PLL power down bit (SCU\_SYSPLL\_CON[15] = 0).
- Power-up MEMPLL by clear PLL power down bit (SCU\_MEMPLL\_CON[15] = 0).
- Power-up HIMACPLL by clear PLL power down bit (SCU\_HIMACPLL\_CON[15] = 0).
- Power-up HIPHYPLL by clear PLL power down bit (SCU\_HIPHYPLL\_CON[20] = 0).

- 
- i. Polling SYSPLL lock status (SCU\_SYSPLL\_CON[17] == 1).
  - j. Polling MEMPLL lock status (SCU\_MEMPLL\_CON[17] == 1).
  - k. Polling HIMACPLL lock status (SCU\_HIMACPLL\_CON[17] == 1).
  - l. Polling HIPHYPLL lock status (SCU\_HIPHYPLL\_CON[22] == 1).
2. Setting CPU/AHB/APB clocks (400/200/100Mhz)
- a. SCU\_CLKM0\_CON[7:6] = 0x1 (ahb\_clk:apb\_clk = 2:1)
  - b. SCU\_CLKM0\_CON[5:4] = 0x1 (arm\_clk:ahb\_clk = 2:1)
  - c. SCU\_CLKM0\_CON[3:2] = 0x1 (M0\_clk:arm\_clk = 2:1)
  - d. SCU\_CLKM0\_CON[1] = 0x0 (SYSPLL\_CLK source)
  - e. SCU\_CLKM0\_CON[0] = 0x1 (CLK\_GEN0).

Note: To avoid unexpected timing problem, the best practice is setting all above register fields in one single register write operation. For example: write SCU\_CLKM0\_CON with value “0x55”.

3. Setting DDR2 clocks (400Mhz)
- a. SCU\_CLKM1\_CON[3:2] = 0x1 (M1\_clk:ddr2\_clk = 2:1)
  - b. SCU\_CLKM1\_CON[1] = 0x1 (MEMPLL\_CLK source)
  - c. SCU\_CLKM1\_CON[0] = 0x1 (CLK\_GEN1).

Note: To avoid unexpected timing problem, the best practice is setting all above register fields in one single register write operation. For example: write SCU\_CLKM1\_CON with value 0x7.

4. Setting HIMAC clocks (125Mhz)
- a. SCU\_CLKM2\_CON[1] = 0x1 (M2\_clk:himac\_clk = 2:1 )
  - b. SCU\_CLKM2\_CON[0] = 0x1 (CLK\_GEN2).

Note: To avoid unexpected timing problem, the best practice is setting all above register fields in one single register write operation. For example: write SCU\_CLKM2\_CON with value 0x3.

5. Setting HIPHY clocks (128Mhz)
- a. SCU\_CLKM3\_CON[2] = 0x1 (M3\_clk:hiphy\_clk = 2:1)
  - b. SCU\_CLKM3\_CON[1] = 0x0 (HIPHYPLL\_CLK source)
  - c. SCU\_CLKM3\_CON[0] = 0x1 (CLK\_GEN3).

Note: To avoid unexpected timing problem, the best practice is setting all above register fields in one single register write operation. For example: write SCU\_CLKM3\_CON with value 0x5.

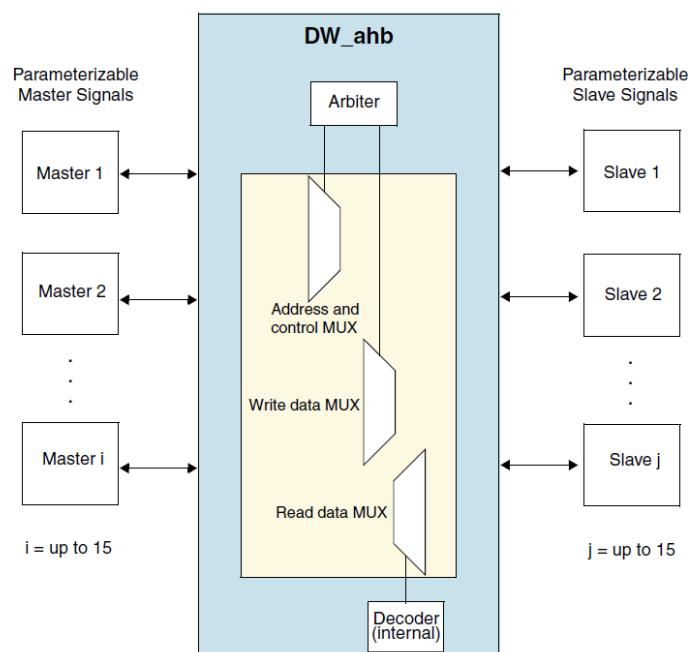
## 4 AHB

### 4.1 AHB Overview

The DW\_ahb consists of the following components:

- “Arbiter” – The bus arbiter ensures that only one bus master at a time is allowed to initiate data transfers. The arbiter contains an optional programmable slave interface for allocating priorities, selecting the default master, allocating tokens to masters, and enabling early burst termination. If the slave interface is *not* included in the design, the following will occur:
  - Weighted-token arbitration will be excluded
  - Early burst termination will be disabled
  - Priorities will be fixed and hardcoded
- “Optional Internal Decoder” – You can choose to include an AHB internal decoder, which is used to decode the address of each transfer and to generate a select signal for the slave that is involved in that transfer. By having the internal decoder, the DW\_ahb needs to supply the addresses. Overlapping regions are checked when the decoder is being configured.
- “Optional External Decoder” – You can choose to use an external decoder. By having the decoder external to the DW\_ahb, users can connect any decoder with any number of remap options.
- “Multiplexer” – All addresses, data, and control signals from each master are multiplexed.

DW\_ahb Block Diagram



### 4.2 AHB Mapping

#### 4.2.1 Register Memory Maps

Register	Offset	Size	Memory Access	Description
DW_ahb address block				
AHB_PL1	0x80000000	32 bits	R/W	<b>Value After Reset:</b> 0x1 Name: Arbitration Priority Master 1 Register Size: 4 bits Address Offset: 0x00 Read/Write Access: Read/Write
AHB_PL2	0x80000004	32	R/W	<b>Value After Reset:</b> 0x2

Register	Offset	Size	Memory Access	Description
		bits		Name: Arbitration Priority Master 2 Register Size: 4 bits Address Offset: 4 Read/Write Access: Read/Write
<a href="#">AHB_PL3</a>	0x80000008	32 bits	R/W	<b>Value After Reset:</b> 0x3 Name: Arbitration Priority Master 3 Register Size: 4 bits Address Offset: 8 Read/Write Access: Read/Write
<a href="#">AHB_PL4</a>	0x8000000c	32 bits	R/W	<b>Value After Reset:</b> 0x4 Name: Arbitration Priority Master 4 Register Size: 4 bits Address Offset: 12 Read/Write Access: Read/Write
<a href="#">AHB_EBTCOUNT</a>	0x8000003c	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Early Burst Termination Count Register Size: 10 bits Address Offset: 0x3c Read/Write Access: Read/Write
<a href="#">AHB_EBT_EN</a>	0x80000040	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Early Burst Termination Enable Register Size: 1 bit Address Offset: 0x40 Read/Write Access: Read/Write
<a href="#">AHB_EBT</a>	0x80000044	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Early Burst Termination Register Size: 1 bit Address Offset: 0x44 Read/Write Access: Read
<a href="#">AHB_DFLT_MASTER</a>	0x80000048	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Default Master ID Number Register Size: 4 bits Address Offset: 0x48 Read/Write Access: Read/Write
<a href="#">AHB_VERSION_ID</a>	0x80000090	32 bits	R	<b>Value After Reset:</b> 0x3231302a Name: Component Version ID Register Size: 32 bits Address Offset: 0x90 Read/Write Access: Read

#### 4.2.2 Register and Field Descriptions

Following is a description of the individual registers of component DW\_ahb

##### 4.2.2.1 AHB\_PL1

**Size:** 32 bits

**Offset:** 0x80000000

**Memory Access:** R/W

**Value After Reset:** 0x1

31:4	3:0
(undef)	AHB_PL1

Name: Arbitration Priority Master 1 Register Size: 4 bits Address Offset: 0x00 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:4			Reserved for future use.
3:0	AHB_PL1	R/W	Arbitration priority for master 1.

##### 4.2.2.2 AHB\_PL2

**Size:** 32 bits

**Offset:** 0x80000004

**Memory Access:** R/W

**Value After Reset:** 0x2

31:4	3:0
(undef)	AHB_PL2

Name: Arbitration Priority Master 2 Register Size: 4 bits Address Offset: 4 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:4			Reserved for future use.
3:0	AHB_PL2	R/W	Arbitration priority for master 2.

**4.2.2.3 AHB\_PL3****Size:** 32 bits**Offset:** 0x80000008**Memory Access:** R/W**Value After Reset:** 0x3

31:4	3:0
(undef)	AHB_PL3

Name: Arbitration Priority Master 3 Register Size: 4 bits Address Offset: 8 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:4			Reserved for future use.
3:0	AHB_PL3	R/W	Arbitration priority for master 3.

**4.2.2.4 AHB\_PL4****Size:** 32 bits**Offset:** 0x8000000c**Memory Access:** R/W**Value After Reset:** 0x4

31:4	3:0
(undef)	AHB_PL4

Name: Arbitration Priority Master 4 Register Size: 4 bits Address Offset: 12 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:4			Reserved for future use.
3:0	AHB_PL4	R/W	Arbitration priority for master 4.

**4.2.2.5 AHB\_EBTCOUNT****Size:** 32 bits**Offset:** 0x8000003c**Memory Access:** R/W**Value After Reset:** 0x0

31:10	9:0
(undef)	AHB_EBTCOUNT

Name: Early Burst Termination Count Register Size: 10 bits Address Offset: 0x3c Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:10			Reserved for future use.
9:0	AHB_EBTCOUNT	R/W	Early burst termination count register. Maximum number of cycles a transfer can take before being subject to an early burst termination.

**4.2.2.6 AHB\_EBT\_EN****Size:** 32 bits**Offset:** 0x80000040**Memory Access:** R/W**Value After Reset:** 0x0

31:1	0
(undef)	AHB_EBT_EN

Name: Early Burst Termination Enable Size: 1 bit Address Offset: 0x40 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	AHB_EBT_EN	R/W	Early burst termination count register.

**4.2.2.7 AHB\_EBT****Size:** 32 bits**Offset:** 0x80000044**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	AHB_EBT

Name: Early Burst Termination Register Size: 1 bit Address Offset: 0x44 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	AHB_EBT	R	Early burst termination register. Set when an Early Burst Termination takes place. The register is cleared when read by the processor.

**4.2.2.8 AHB\_DFLT\_MASTER****Size:** 32 bits**Offset:** 0x80000048**Memory Access:** R/W**Value After Reset:** 0x0

31:4	3:0
(undef)	AHB_DFLT_MASTER

Name: Default Master ID Number Register Size: 4 bits Address Offset: 0x48 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description

Bits	Name	Memory Access	Description
31:4			Reserved for future use.
3:0	AHB_DFLT_MASTER	R/W	Default master ID number register. The default master is the master that is granted by the bus when no master has requested ownership.

**4.2.2.9 AHB\_VERSION\_ID****Size:** 32 bits**Offset:** 0x80000090**Memory Access:** R**Value After Reset:** 0x3231302a

31:0
AHB_VERSION_ID

Name: Component Version ID Register Size: 32 bits Address Offset: 0x90 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:0	AHB_VERSION_ID	R	ASCII value for each number in the version.

## 5 Direct Memory Access (DMA)

### 5.1 Overview

#### 5.1.1 DMA diagram

Figure 4-1 shows the following functional groupings of the main interfaces to the DW\_ahb\_dmac block:

- DMA hardware request interface
- Up to eight channels
- FIFO per channel for source and destination
- Arbiter
- AHB master interface
- AHB slave interface

DW\_ahb\_dmac Block Diagram

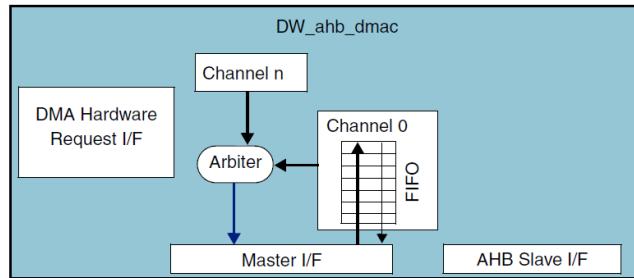


Figure 4-1 DMA Diagram

One channel of the DW\_ahb\_dmac is required for each source/destination pair. The master interface reads the data from a source peripheral (A) and writes it to a destination peripheral (B). Two AHB transfers are required for each DMA data transfer; this is also known as a dual-access transfer.

Figure 4-2 illustrates a peripheral-to-peripheral DMA transfer, where peripheral A (source) uses a hardware handshaking interface, and peripheral B (destination) uses a software handshaking interface. For example, the request to send data to peripheral B is originated by the CPU, while writing to peripheral B is handled by the DW\_ahb\_dmac. The channel source and destination arbitrate independently for the AHB master interface, along with other channels.

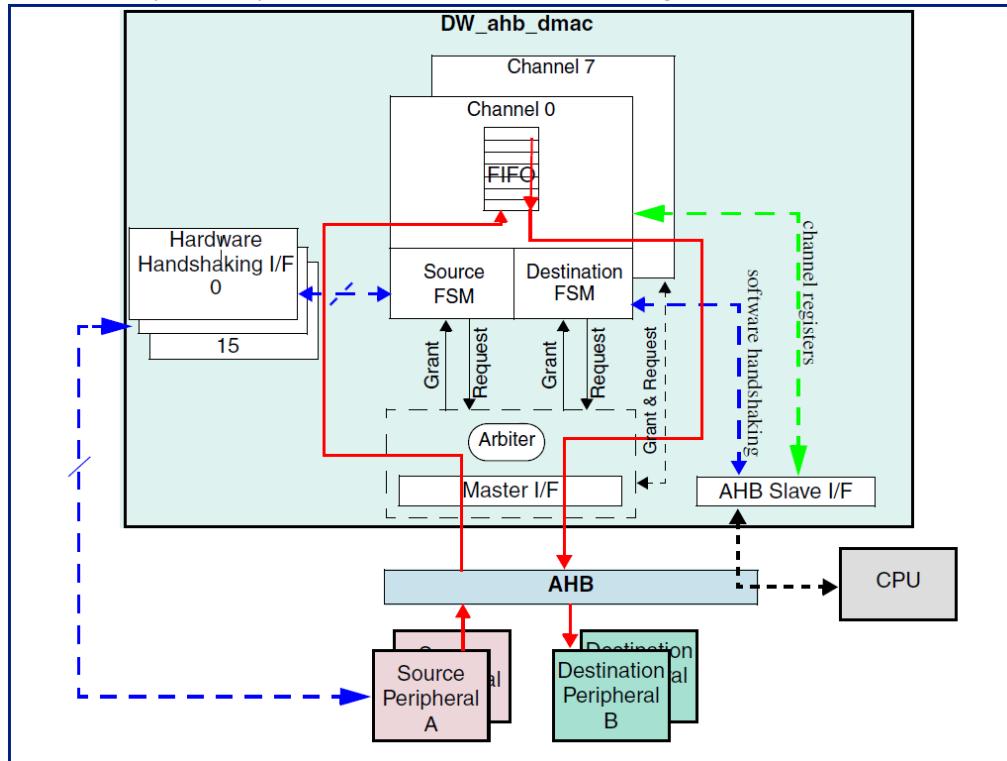


Figure 4-2 Peripheral-to-Peripheral DMA transfer on Same AHB Layer

## 5.2 Functional Description

### 5.2.1 Block Flow Controller and Transfer Type

The device that controls the length of a block is known as the flow controller. Either the DW\_ahb\_dmac, the source peripheral, or the destination peripheral must be assigned as the flow controller.

- If the block size is known prior to when the channel is enabled, then the DW\_ahb\_dmac should be programmed as the flow controller. The block size should be programmed into the CTLx.BLOCK\_TS field.

As an example, the DW\_ahb\_dmac can be programmed as the flow controller when a DMA block must be transferred from a receive DW\_apb\_ssi peripheral to memory. In a block transfer, software programs the DW\_apb\_ssi register – CTRLR1.NDF – with the number of source data items minus 1. Software then programs the CTLx.BLOCK\_TS register with the same value and programs the DW\_ahb\_dmac as the flow controller.

### 5.2.2 Basic Interface Definitions

In this chapter and the following equations, references to CTLx.SRC\_MSIZEx, CTLx.DEST\_MSIZEx, CTLx.SRC\_TR\_WIDTH, and CTLx.DST\_TR\_WIDTH refer to the decoded values of the parameters; for example, CTLx.SRC\_MSIZEx = 3'b001 decodes to 4, and CTLx.SRC\_TR\_WIDTH = 3'b010 decodes to 32 bits.

The following definitions are used in this chapter:

- Source single transaction size in bytes

$$\text{src\_single\_size\_bytes} = \text{CTLx.SRC\_TR\_WIDTH}/8 \quad (1)$$

- Source burst transaction size in bytes

$$\text{src\_burst\_size\_bytes} = \text{CTLx.SRC\_MSIZEx} * \text{src\_single\_size\_bytes} \quad (2)$$

- Destination single transaction size in bytes

$$\text{dst\_single\_size\_bytes} = \text{CTLx.DST\_TR\_WIDTH}/8 \quad (3)$$

- Destination burst transaction size in bytes

$$\text{dst\_burst\_size\_bytes} = \text{CTLx.DEST\_MSIZEx} * \text{dst\_single\_size\_bytes} \quad (4)$$

- Block size in bytes:

- DW\_ahb\_dmac is flow controller – With the DW\_ahb\_dmac as the flow controller, the processor programs the DW\_ahb\_dmac with the number of data items (block size) of source transfer width (CTLx.SRC\_TR\_WIDTH) to be transferred by the DW\_ahb\_dmac in a block transfer; this is programmed into the CTLx.BLOCK\_TS field. Therefore, the total number of bytes to be transferred in a block is:  $\text{blk\_size\_bytes\_dma} = \text{CTLx.BLOCK\_TS} * \text{src\_single\_size\_bytes}$  (5)

### 5.2.3 Software Handshaking

When the slave peripheral requires the DW\_ahb\_dmac to perform a DMA transaction, it communicates this request by sending an interrupt to the CPU or interrupt controller. The interrupt service routine then uses the software registers detailed in “Software Handshaking Registers” to initiate and control a DMA transaction. This group of software registers is used to implement the software handshaking interface.

The HS\_SEL\_SRC/HS\_SEL\_DST bit in the CFGx channel configuration register must be set to enable software handshaking.

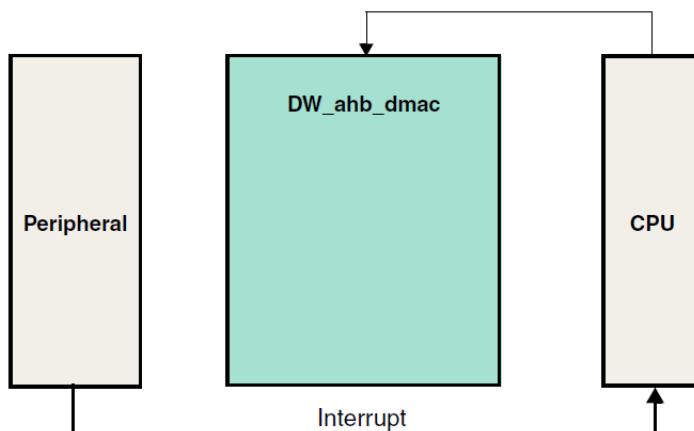


Figure 4-3 Software Controlled DMA transfers

- Program and enable channel through “Channel Registers”
- After interrupt, initiate and control DMA transaction between peripheral(s DW\_ahb\_dmac through “Software Handshaking Registers”

The software handshaking registers are:

- ReqSrcReg – source software transaction request
- ReqDstReg – destination software transaction request
- SglReqSrcReg – single source transaction request
- SglReqDstReg – single destination transaction request
- LstSrcReg – last source transaction request
- LstDstReg – last destination transaction request

#### 5.2.4 Setting Up Transfers

Transfers are set up by programming fields of the CTLx and CFGx registers for that channel.

Following Table lists the parameters that are investigated in the following examples. The effects of these parameters on the flow of the block transfer are highlighted. In addition to the software parameters, it includes the channel FIFO depth, DMAH\_CHx\_FIFO\_DEPTH, which is configurable only in coreConsultant.

**Table 3-4 Parameters Used in Transfer Examples**

Parameter	Description
DMAH_CHx_FIFO_DEPTH	Channel x FIFO depth in bytes
CTLx.TT_FC	Transfer type and flow control
CTLx.BLOCK_TS	Block transfer size
CTLx.SRC_TR_WIDTH	Source transfer width
CTLx.DST_TR_WIDTH	Destination transfer width
CTLx.SRC_MSIZE	Source burst transaction length
CTLx.DEST_MSIZE	Destination burst transaction length
CFGx.MAX_ABRST	Maximum AMBA burst length
CFGx.FIFO_MODE	FIFO mode select
CFGx.FCMODE	Flow-control mode

#### 5.2.5 Generating Requests for the AHB Master Bus Interface

Each channel has a source state machine and destination state machine running in parallel. These state machines generate the request inputs to the arbiter, which arbitrates for the master bus interface (one arbiter per master bus interface).

When the source/destination state machine is granted control of the master bus interface, and when the master bus interface is granted control of the external AHB bus, then AHB transfers between the peripheral and the DW\_ahb\_dmac (on behalf of the granted state machine) can take place.

AHB transfers from the source peripheral or to the destination peripheral cannot proceed until the channel FIFO is ready. For burst transaction requests and for transfers involving memory peripherals, the criterion for “FIFO readiness” is controlled by the FIFO\_MODE field of the CFGx register.

The definition of FIFO readiness is the same for:

- Single transactions
- Burst transactions, where CFGx.FIFO\_MODE = 0
- Transfers involving memory peripherals, where CFGx.FIFO\_MODE = 0

The channel FIFO is deemed ready when the space/data available is sufficient to complete a single AHB transfer of the specified transfer width. FIFO readiness for source transfers occurs when the channel FIFO contains enough room to accept at least a single transfer of CTLx.SRC\_TR\_WIDTH width. FIFO readiness for destination transfers occurs when the channel FIFO contains data to form at least a single transfer of CTLx.DST\_TR\_WIDTH width.

An exception to FIFO readiness for destination transfers occurs in “FIFO flush mode.” In this mode, FIFO readiness for destination transfers occurs when the channel FIFO contains data to form at least a single transfer of CTLx.SRC\_TR\_WIDTH width (and not CTLx.DST\_TR\_WIDTH width, as is the normal case).

When CFGx.FIFO\_MODE = 1, then the criteria for FIFO readiness for burst transaction requests and transfers involving memory peripherals are as follows:

- A FIFO is ready for a source burst transfer when the FIFO is less than half empty.
- A FIFO is ready for a destination burst transfer when the FIFO is greater than or equal to half full.

Exceptions to this “readiness” occur. During these exceptions, a value of CTLx.FIFO\_MODE = 0 is assumed. The following are the exceptions:

- Near the end of a burst transaction or block transfer – The channel source state machine does not wait for the channel FIFO to be less than half empty if the number of source data items left to complete the source burst transaction or source block transfer is less than DMAH\_CHx\_FIFO\_DEPTH/2. Similarly, the channel destination state machine does not wait for the channel FIFO to be greater than or equal to half full, if the number of destination data items left to complete the destination burst transaction or destination block transfer is less than DMAH\_CHx\_FIFO\_DEPTH/2.
- In FIFO flush mode – For an explanation of FIFO flush mode, refer to “Example 5” on page 64.
- When a channel is suspended – The destination state machine does not wait for the FIFO to become half empty to flush the FIFO, regardless of the value of the FIFO\_MODE field.
- After receipt of a split/retry response from a source or destination – The AMBA protocol requires that after an AHB master receives a split/retry response, it must re-issue the transfer that received the split/retry before attempting any other transfer. Therefore, a transfer is re-issued to the same address that returned the split/retry, regardless of FIFO\_MODE, when the DW\_ahb\_dmac is next granted the AHB bus. This is repeated until an OKAY response is received on the AHB hresp bus.

When the source/destination peripheral is not memory, the source/destination state machine waits for a single/burst transaction request. Upon receipt of a transaction request and only if the channel FIFO is “ready” for source/destination AHB transfers, a request for the master bus interface is made by the source/destination state machine.

There is one exception to this, which occurs when the destination peripheral is the flow controller and CFGx.FCMODE = 1 (data pre-fetching is disabled). Then the source state machine does not generate a request for the master bus interface (even if the FIFO is “ready” for source transfers and has received a source transaction request) until the destination requests new data.

When the source/destination peripheral is memory, the source/destination state machine must wait until the channel FIFO is “ready.” A request is then made for the master bus interface. There is no handshaking mechanism employed between a memory peripheral and the DW\_ahb\_dmac.

## 5.3 DMA Register

### 5.3.1 Register Memory Maps

Register	Offset	Size	Memory Access	Description
Channel registers				
<a href="#">SAR0</a>	0x0	64 bits	R/W	<b>Value After Reset:</b> 0x0 Channel source address
<a href="#">DAR0</a>	0x8	64 bits	R/W	<b>Value After Reset:</b> 0x0 Channel destination address
<a href="#">CTL0</a>	0x18	64 bits	R/W	<b>Value After Reset:</b> 0x200304825 Channel control

Register	Offset	Size	Memory Access	Description
<a href="#">CFG0</a>	0x40	64 bits	R/W	<b>Value After Reset:</b> 0x400000e00 Channel configuration
Channel registers				
<a href="#">SAR1</a>	0x58	64 bits	R/W	<b>Value After Reset:</b> 0x0 Channel source address
<a href="#">DAR1</a>	0x60	64 bits	R/W	<b>Value After Reset:</b> 0x0 Channel destination address
<a href="#">CTL1</a>	0x70	64 bits	R/W	<b>Value After Reset:</b> 0x200304825 Channel control
<a href="#">CFG1</a>	0x98	64 bits	R/W	<b>Value After Reset:</b> 0x400000e20 Channel configuration
Interrupt registers				
<a href="#">RawTfr</a>	0x2c0	64 bits	R/W	<b>Value After Reset:</b> 0x0 Raw Status for IntTfr Interrupt
<a href="#">RawBlock</a>	0x2c8	64 bits	R/W	<b>Value After Reset:</b> 0x0 Raw Status for IntBlock Interrupt
<a href="#">RawSrcTran</a>	0x2d0	64 bits	R/W	<b>Value After Reset:</b> 0x0 Raw Status for IntSrcTran Interrupt
<a href="#">RawDstTran</a>	0x2d8	64 bits	R/W	<b>Value After Reset:</b> 0x0 Raw Status for IntDstTran Interrupt
<a href="#">RawErr</a>	0x2e0	64 bits	R/W	<b>Value After Reset:</b> 0x0 Raw Status for IntErr Interrupt
<a href="#">StatusTfr</a>	0x2e8	64 bits	R	<b>Value After Reset:</b> 0x0 Status for IntTfr Interrupt
<a href="#">StatusBlock</a>	0x2f0	64 bits	R	<b>Value After Reset:</b> 0x0 Status for IntBlock Interrupt
<a href="#">StatusSrcTran</a>	0x2f8	64 bits	R	<b>Value After Reset:</b> 0x0 Status for IntSrcTran Interrupt
<a href="#">StatusDstTran</a>	0x300	64 bits	R	<b>Value After Reset:</b> 0x0 Status for IntDstTran Interrupt
<a href="#">StatusErr</a>	0x308	64 bits	R	<b>Value After Reset:</b> 0x0 Status for IntErr Interrupt
<a href="#">MaskTfr</a>	0x310	64 bits	R/W	<b>Value After Reset:</b> 0x0 Mask for IntTfr Interrupt
<a href="#">MaskBlock</a>	0x318	64 bits	R/W	<b>Value After Reset:</b> 0x0 Mask for IntBlock Interrupt
<a href="#">MaskSrcTran</a>	0x320	64 bits	R/W	<b>Value After Reset:</b> 0x0 Mask for IntSrcTran Interrupt
<a href="#">MaskDstTran</a>	0x328	64 bits	R/W	<b>Value After Reset:</b> 0x0

Register	Offset	Size	Memory Access	Description
				Mask for IntDstTran Interrupt
<a href="#">MaskErr</a>	0x330	64 bits	R/W	<b>Value After Reset:</b> 0x0 Mask for IntErr Interrupt
<a href="#">ClearTfr</a>	0x338	64 bits	W	Clear for IntTfr Interrupt
<a href="#">ClearBlock</a>	0x340	64 bits	W	Clear for IntBlock Interrupt
<a href="#">ClearSrcTran</a>	0x348	64 bits	W	Clear for IntSrcTran Interrupt
<a href="#">ClearDstTran</a>	0x350	64 bits	W	Clear for IntDstTran Interrupt
<a href="#">ClearErr</a>	0x358	64 bits	W	Clear for IntErr Interrupt
<a href="#">StatusInt</a>	0x360	64 bits	R	<b>Value After Reset:</b> 0x0 Status for each Interrupt type
Software Handshaking Registers				
<a href="#">ReqSrcReg</a>	0x368	64 bits	R/W	<b>Value After Reset:</b> 0x0 Source Software Transaction Request register
<a href="#">ReqDstReg</a>	0x370	64 bits	R/W	<b>Value After Reset:</b> 0x0 Destination Software Transaction Request register
<a href="#">SglRqSrcReg</a>	0x378	64 bits	R/W	<b>Value After Reset:</b> 0x0 Source Single Transaction Request register
<a href="#">SglRqDstReg</a>	0x380	64 bits	R/W	<b>Value After Reset:</b> 0x0 Destination Single Transaction Request register
<a href="#">LstSrcReg</a>	0x388	64 bits	R/W	<b>Value After Reset:</b> 0x0 Source Last Transaction Request register
<a href="#">LstDstReg</a>	0x390	64 bits	R/W	<b>Value After Reset:</b> 0x0 Source Last Transaction Request register
Miscellaneous Registers				
<a href="#">DmaCfgReg</a>	0x398	64 bits	R/W	DMA Configuration Register
<a href="#">ChEnReg</a>	0x3a0	64 bits	R/W	Channel enable register
<a href="#">DmaldReg</a>	0x3a8	64 bits	R	<b>Value After Reset:</b> 0x0 DMA ID register
<a href="#">DmaTestReg</a>	0x3b0	64 bits	R/W	<b>Value After Reset:</b> 0x0 DMA Test register
<a href="#">DMA_COMP_PARAMS_6</a>	0x3c8	64 bits	R	<b>Value After Reset:</b> 0x1001201200000000 Component Parameters registers
<a href="#">DMA_COMP_PARAMS_5</a>	0x3d0	64 bits	R	<b>Value After Reset:</b> 0x1001201210012012 Component Parameters registers
<a href="#">DMA_COMP_PARAMS_4</a>	0x3d8	64 bits	R	<b>Value After Reset:</b> 0x1001201210012012 Component Parameters registers
<a href="#">DMA_COMP_PARAMS_3</a>	0x3e0	64 bits	R	<b>Value After Reset:</b> 0x5001201210012012

Register	Offset	Size	Memory Access	Description
				Component Parameters registers
<a href="#">DMA_COMP_PARAMS_2</a>	0x3e8	64 bits	R	<b>Value After Reset:</b> 0x50012012 Component Parameters registers
<a href="#">DMA_COMP_PARAMS_1</a>	0x3f0	64 bits	R	<b>Value After Reset:</b> 0x3000010433333388 Component Parameters registers
<a href="#">DmaCompsID</a>	0x3f8	64 bits	R	<b>Value After Reset:</b> 0x3231372a44571110 DMA Component ID register

### 5.3.2 Register and Field Descriptions

Following is a description of the individual registers of component DW\_ahb\_dmac

#### 5.3.2.1 Channel 0 registers

##### 5.3.2.1.1 SAR0

**Size:** 64 bits

**Offset:** 0x0

**Memory Access:** R/W

**Value After Reset:** 0x0

63:32	31:0
(undef)	SAR

Channel source address

Bits	Name	Memory Access	Description
63:32			Reserved for future use.
31:0	SAR	R/W	Channel source address

##### 5.3.2.1.2 DAR0

**Size:** 64 bits

**Offset:** 0x8

**Memory Access:** R/W

**Value After Reset:** 0x0

63:32	31:0
(undef)	DAR

Channel destination address

Bits	Name	Memory Access	Description
63:32			Reserved for future use.
31:0	DAR	R/W	Channel destination address

##### 5.3.2.1.3 CTL0

**Size:** 64 bits

**Offset:** 0x18

**Memory Access:** R/W

**Value After Reset:** 0x200304825

63:45	44	43:42	41:32	31:23	22:20	19:17	16:14	13:11	10:9	8:7	6:1	0
(undef )	DONE	(undef )	BLOCK_TS	(undef )	TT_FC	(undef )	SRC_MSIZE	DEST_MSIZE	SINC	DINC	(undef )	INT_EN

## Channel control

Bits	Name	Memory Access	Description
63:45			Reserved for future use.
44	DONE	R/W	Done Bit
43:42			Reserved for future use.
41:32	BLOCK_TS	R/W	Block length
31:23			Reserved for future use.
22:20	TT_FC	R/W	Flow control
19:17			Reserved for future use.
16:14	SRC_MSIZE	R/W	Source transaction burst length
13:11	DEST_MSIZE	R/W	Destination transaction burst length
10:9	SINC	R/W	Source address direction control
8:7	DINC	R/W	Destination address direction control
6:1			Reserved for future use.
0	INT_EN	R/W	Interrupt enable

## 5.3.2.1.4 CFG0

**Size:** 64 bits**Offset:** 0x40**Memory Access:** R/W**Value After Reset:** 0x400000e00

63:3 7	36:34	33	32	31:2 0	19	18	17:1 2	11	10	9	8	7:5	4:0
(und ef)	PROT CTL	FIFO_M ODE	FCM ODE	(und ef)	SRC_HS _POL	DST_HS _POL	(und ef)	HS_SEL _SRC	HS_SEL _DST	FIFO_E MPTY	CH_S USP	CH_PR IOR	(und ef)

## Channel configuration

Bits	Name	Memory Access	Description
63:37			Reserved for future use.
36:34	PROTCTL	R/W	AHB bus protctl bus control
33	FIFO_MODE	R/W	Channel FIFO mode control
32	FCMODE	R/W	Channel flow control mode
31:20			Reserved for future use.
19	SRC_HS_POL	R/W	Source handshake polarity
18	DST_HS_POL	R/W	Destination handshake polarity

Bits	Name	Memory Access	Description
17:12			Reserved for future use.
11	HS_SEL_SRC	R/W	Source handshake select
10	HS_SEL_DST	R/W	Destination handshake select
9	FIFO_EMPTY	R	Channel FIFO empty status
8	CH_SUSP	R/W	Channel Suspend control
7:5	CH_PRIOR	R/W	Channel Priority
4:0			Reserved for future use.

### 5.3.2.2 Channel 1 registers

#### 5.3.2.2.1 SAR1

**Size:** 64 bits

**Offset:** 0x58

**Memory Access:** R/W

**Value After Reset:** 0x0

63:32	31:0
(undef)	SAR

Channel source address

Bits	Name	Memory Access	Description
63:32			Reserved for future use.
31:0	SAR	R/W	Channel source address

#### 5.3.2.2.2 DAR1

**Size:** 64 bits

**Offset:** 0x60

**Memory Access:** R/W

**Value After Reset:** 0x0

63:32	31:0
(undef)	DAR

Channel destination address

Bits	Name	Memory Access	Description
63:32			Reserved for future use.
31:0	DAR	R/W	Channel destination address

#### 5.3.2.2.3 CTL1

**Size:** 64 bits

**Offset:** 0x70

**Memory Access:** R/W

**Value After Reset:** 0x200304825

63:45	44	43:42	41:32	31:23	22:20	19:17	16:14	13:11	10:9	8:7	6:1	0
-------	----	-------	-------	-------	-------	-------	-------	-------	------	-----	-----	---

(undef )	DONE	(undef )	BLOCK_TS	(undef )	TT_FC	(undef )	SRC_MSIZE	DEST_MSIZE	SINC	DINC	(undef )	INT_EN
----------	------	----------	----------	----------	-------	----------	-----------	------------	------	------	----------	--------

Channel control

Bits	Name	Memory Access	Description
63:45			Reserved for future use.
44	DONE	R/W	Done Bit
43:42			Reserved for future use.
41:32	BLOCK_TS	R/W	Block length
31:23			Reserved for future use.
22:20	TT_FC	R/W	Flow control
19:17			Reserved for future use.
16:14	SRC_MSIZE	R/W	Source transaction burst length
13:11	DEST_MSIZE	R/W	Destination transaction burst length
10:9	SINC	R/W	Source address direction control
8:7	DINC	R/W	Destination address direction control
6:1			Reserved for future use.
0	INT_EN	R/W	Interrupt enable

#### 5.3.2.2.4 CFG1

**Size:** 64 bits**Offset:** 0x98**Memory Access:** R/W**Value After Reset:** 0x400000e20

63:37	36:34	33	32	31:20	19	18	17:12	11	10	9	8	7:5	4:0
(und ef)	PROT CTL	FIFO_MODE	FCM ODE	(und ef)	SRC_HS _POL	DST_HS _POL	(und ef)	HS_SEL _SRC	HS_SEL _DST	FIFO_E MPTY	CH_S USP	CH_PR IOR	(und ef)

Channel configuration

Bits	Name	Memory Access	Description
63:37			Reserved for future use.
36:34	PROTCTL	R/W	AHB bus protctl bus control
33	FIFO_MODE	R/W	Channel FIFO mode control
32	FCMODE	R/W	Channel flow control mode
31:20			Reserved for future use.
19	SRC_HS_POL	R/W	Source handshake polarity
18	DST_HS_POL	R/W	Destination handshake polarity
17:12			Reserved for future use.

Bits	Name	Memory Access	Description
11	HS_SEL_SRC	R/W	Source handshake select
10	HS_SEL_DST	R/W	Destination handshake select
9	FIFO_EMPTY	R	Channel FIFO empty status
8	CH_SUSP	R/W	Channel Suspend control
7:5	CH_PRIOR	R/W	Channel Priority
4:0			Reserved for future use.

### 5.3.2.3 Interrupt registers

#### 5.3.2.3.1 RawTfr

**Size:** 64 bits

**Offset:** 0x2c0

**Memory Access:** R/W

**Value After Reset:** 0x0

63:2	1:0
(undef)	RAW

Raw Status for IntTfr Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	RAW	R/W	Raw Status for IntTfr Interrupt

#### 5.3.2.3.2 RawBlock

**Size:** 64 bits

**Offset:** 0x2c8

**Memory Access:** R/W

**Value After Reset:** 0x0

63:2	1:0
(undef)	RAW

Raw Status for IntBlock Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	RAW	R/W	Raw Status for IntBlock Interrupt

#### 5.3.2.3.3 RawSrcTran

**Size:** 64 bits

**Offset:** 0x2d0

**Memory Access:** R/W

**Value After Reset:** 0x0

63:2	1:0
(undef)	RAW

## Raw Status for IntSrcTran Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	RAW	R/W	Raw Status for IntSrcTran Interrupt

**5.3.2.3.4 RawDstTran****Size:** 64 bits**Offset:** 0x2d8**Memory Access:** R/W**Value After Reset:** 0x0

63:2	1:0
(undef)	RAW

## Raw Status for IntDstTran Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	RAW	R/W	Raw Status for IntDstTran Interrupt

**5.3.2.3.5 RawErr****Size:** 64 bits**Offset:** 0x2e0**Memory Access:** R/W**Value After Reset:** 0x0

63:2	1:0
(undef)	RAW

## Raw Status for IntErr Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	RAW	R/W	Raw Status for IntErr Interrupt

**5.3.2.3.6 StatusTfr****Size:** 64 bits**Offset:** 0x2e8**Memory Access:** R**Value After Reset:** 0x0

63:2	1:0
(undef)	STATUS

## Status for IntTfr Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	STATUS	R	Status for IntTfr Interrupt

**5.3.2.3.7 StatusBlock****Size:** 64 bits**Offset:** 0x2f0**Memory Access:** R**Value After Reset:** 0x0

63:2	1:0
(undef)	STATUS

Status for IntBlock Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	STATUS	R	Status for IntBlock Interrupt

**5.3.2.3.8 StatusSrcTran****Size:** 64 bits**Offset:** 0x2f8**Memory Access:** R**Value After Reset:** 0x0

63:2	1:0
(undef)	STATUS

Status for IntSrcTran Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	STATUS	R	Status for IntSrcTran Interrupt

**5.3.2.3.9 StatusDstTran****Size:** 64 bits**Offset:** 0x300**Memory Access:** R**Value After Reset:** 0x0

63:2	1:0
(undef)	STATUS

Status for IntDstTran Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	STATUS	R	Status for IntDstTran Interrupt

**5.3.2.3.10 StatusErr****Size:** 64 bits**Offset:** 0x308**Memory Access:** R**Value After Reset:** 0x0

63:2	1:0
(undef)	STATUS

(undef)	STATUS
---------	--------

Status for IntErr Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	STATUS	R	Status for IntErr Interrupt

**5.3.2.3.11 MaskTfr****Size:** 64 bits**Offset:** 0x310**Memory Access:** R/W**Value After Reset:** 0x0

63:10	9:8	7:2	1:0
(undef)	INT_MASK_WE	(undef)	INT_MASK

Mask for IntTfr Interrupt

Bits	Name	Memory Access	Description
63:10			Reserved for future use.
9:8	INT_MASK_WE	W	Interrupt Mask Write Enable
7:2			Reserved for future use.
1:0	INT_MASK	R/W	Mask for IntTfr Interrupt

**5.3.2.3.12 MaskBlock****Size:** 64 bits**Offset:** 0x318**Memory Access:** R/W**Value After Reset:** 0x0

63:10	9:8	7:2	1:0
(undef)	INT_MASK_WE	(undef)	INT_MASK

Mask for IntBlock Interrupt

Bits	Name	Memory Access	Description
63:10			Reserved for future use.
9:8	INT_MASK_WE	W	Interrupt Mask Write Enable
7:2			Reserved for future use.
1:0	INT_MASK	R/W	Mask for IntBlock Interrupt

**5.3.2.3.13 MaskSrcTran****Size:** 64 bits**Offset:** 0x320**Memory Access:** R/W**Value After Reset:** 0x0

63:10	9:8	7:2	1:0
-------	-----	-----	-----

(undef)	INT_MASK_WE	(undef)	INT_MASK
---------	-------------	---------	----------

Mask for IntSrcTran Interrupt

Bits	Name	Memory Access	Description
63:10			Reserved for future use.
9:8	INT_MASK_WE	W	Interrupt Mask Write Enable
7:2			Reserved for future use.
1:0	INT_MASK	R/W	Mask for IntSrcTran Interrupt

**5.3.2.3.14 MaskDstTran****Size:** 64 bits**Offset:** 0x328**Memory Access:** R/W**Value After Reset:** 0x0

63:10	9:8	7:2	1:0
(undef)	INT_MASK_WE	(undef)	INT_MASK

Mask for IntDstTran Interrupt

Bits	Name	Memory Access	Description
63:10			Reserved for future use.
9:8	INT_MASK_WE	W	Interrupt Mask Write Enable
7:2			Reserved for future use.
1:0	INT_MASK	R/W	Mask for IntDstTran Interrupt

**5.3.2.3.15 MaskErr****Size:** 64 bits**Offset:** 0x330**Memory Access:** R/W**Value After Reset:** 0x0

63:10	9:8	7:2	1:0
(undef)	INT_MASK_WE	(undef)	INT_MASK

Mask for IntErr Interrupt

Bits	Name	Memory Access	Description
63:10			Reserved for future use.
9:8	INT_MASK_WE	W	Interrupt Mask Write Enable
7:2			Reserved for future use.
1:0	INT_MASK	R/W	Mask for IntErr Interrupt

**5.3.2.3.16 ClearTfr****Size:** 64 bits**Offset:** 0x338**Memory Access:** W

63:2	1:0
(undef)	CLEAR

Clear for IntTfr Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	CLEAR	W	Clear for IntTfr Interrupt

### 5.3.2.3.17 ClearBlock

**Size:** 64 bits

**Offset:** 0x340

**Memory Access:** W

63:2	1:0
(undef)	CLEAR

Clear for IntBlock Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	CLEAR	W	Clear for IntBlock Interrupt

### 5.3.2.3.18 ClearSrcTran

**Size:** 64 bits

**Offset:** 0x348

**Memory Access:** W

63:2	1:0
(undef)	CLEAR

Clear for IntSrcTran Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	CLEAR	W	Clear for IntSrcTran Interrupt

### 5.3.2.3.19 ClearDstTran

**Size:** 64 bits

**Offset:** 0x350

**Memory Access:** W

63:2	1:0
(undef)	CLEAR

Clear for IntDstTran Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	CLEAR	W	Clear for IntDstTran Interrupt

**5.3.2.3.20 ClearErr****Size:** 64 bits**Offset:** 0x358**Memory Access:** W

63:2	1:0
(undef)	CLEAR

Clear for IntErr Interrupt

Bits	Name	Memory Access	Description
63:2			Reserved for future use.
1:0	CLEAR	W	Clear for IntErr Interrupt

**5.3.2.3.21 StatusInt****Size:** 64 bits**Offset:** 0x360**Memory Access:** R**Value After Reset:** 0x0

63:5	4	3	2	1	0
(undef)	ERR	DSTT	SRCT	BLOCK	TFR

Status for each Interrupt type

Bits	Name	Memory Access	Description
63:5			Reserved for future use.
4	ERR	R	Status for each Interrupt type
3	DSTT	R	Status for each Interrupt type
2	SRCT	R	Status for each Interrupt type
1	BLOCK	R	Status for each Interrupt type
0	TFR	R	Status for each Interrupt type

**5.3.2.4 Software Handshaking Registers****5.3.2.4.1 ReqSrcReg****Size:** 64 bits**Offset:** 0x368**Memory Access:** R/W**Value After Reset:** 0x0

63:10	9:8	7:2	1:0
(undef)	SRC_REQ_WE	(undef)	SRC_REQ

Source Software Transaction Request register

Bits	Name	Memory Access	Description
63:10			Reserved for future use.
9:8	SRC_REQ_WE	R/W	Source Software Transaction Request write enable
7:2			Reserved for future use.

Bits	Name	Memory Access	Description
1:0	SRC_REQ	R/W	Source Software Transaction Request register

**5.3.2.4.2 ReqDstReg****Size:** 64 bits**Offset:** 0x370**Memory Access:** R/W**Value After Reset:** 0x0

63:10	9:8	7:2	1:0
(undef)	DST_REQ_WE	(undef)	DST_REQ

Destination Software Transaction Request register

Bits	Name	Memory Access	Description
63:10			Reserved for future use.
9:8	DST_REQ_WE	R/W	Destination Software Transaction Request write enable
7:2			Reserved for future use.
1:0	DST_REQ	R/W	Destination Software Transaction Request register

**5.3.2.4.3 SglRqSrcReg****Size:** 64 bits**Offset:** 0x378**Memory Access:** R/W**Value After Reset:** 0x0

63:10	9:8	7:2	1:0
(undef)	SRC_SGLREQ_WE	(undef)	SRC_SGLREQ

Source Single Transaction Request register

Bits	Name	Memory Access	Description
63:10			Reserved for future use.
9:8	SRC_SGLREQ_WE	R/W	Source Single Transaction Request write enable
7:2			Reserved for future use.
1:0	SRC_SGLREQ	R/W	Source Single Transaction Request register

**5.3.2.4.4 SglRqDstReg****Size:** 64 bits**Offset:** 0x380**Memory Access:** R/W**Value After Reset:** 0x0

63:10	9:8	7:2	1:0
(undef)	DST_SGLREQ_WE	(undef)	DST_SGLREQ

Destination Single Transaction Request register

Bits	Name	Memory Access	Description

Bits	Name	Memory Access	Description
63:10			Reserved for future use.
9:8	DST_SGLREQ_WE	R/W	Destination Single Transaction Request write enable
7:2			Reserved for future use.
1:0	DST_SGLREQ	R/W	Destination Single Transaction Request register

#### 5.3.2.4.5 LstSrcReg

**Size:** 64 bits

**Offset:** 0x388

**Memory Access:** R/W

**Value After Reset:** 0x0

63:10	9:8	7:2	1:0
(undef)	LSTSRC_WE	(undef)	LSTSRC

Source Last Transaction Request register

Bits	Name	Memory Access	Description
63:10			Reserved for future use.
9:8	LSTSRC_WE	R/W	Source Last Transaction Request write enable
7:2			Reserved for future use.
1:0	LSTSRC	R/W	Source Last Transaction Request register

#### 5.3.2.4.6 LstDstReg

**Size:** 64 bits

**Offset:** 0x390

**Memory Access:** R/W

**Value After Reset:** 0x0

63:10	9:8	7:2	1:0
(undef)	LSTDST_WE	(undef)	LSTDST

Source Last Transaction Request register

Bits	Name	Memory Access	Description
63:10			Reserved for future use.
9:8	LSTDST_WE	R/W	Source Last Transaction Request write enable
7:2			Reserved for future use.
1:0	LSTDST	R/W	Destination Last Transaction Request register

### 5.3.2.5 Miscellaneous Registers

#### 5.3.2.5.1 DmaCfgReg

**Size:** 64 bits

**Offset:** 0x398

**Memory Access:** R/W

63:1	0
------	---

(undef)	DMA_EN
---------	--------

DMA Configuration Register

Bits	Name	Memory Access	Description
63:1			Reserved for future use.
0	DMA_EN	R/W	DMA global enable

**5.3.2.5.2 ChEnReg****Size:** 64 bits**Offset:** 0x3a0**Memory Access:** R/W

63:10	9:8	7:2	1:0
(undef)	CH_EN_WE	(undef)	CH_EN

Channel enable register

Bits	Name	Memory Access	Description
63:10			Reserved for future use.
9:8	CH_EN_WE	W	Channel enable register
7:2			Reserved for future use.
1:0	CH_EN	R/W	Channel enable register

**5.3.2.5.3 DmaldReg****Size:** 64 bits**Offset:** 0x3a8**Memory Access:** R**Value After Reset:** 0x0

63:32	31:0
(undef)	DMA_ID

DMA ID register

Bits	Name	Memory Access	Description
63:32			Reserved for future use.
31:0	DMA_ID	R	DMA ID register

**5.3.2.5.4 DmaTestReg****Size:** 64 bits**Offset:** 0x3b0**Memory Access:** R/W**Value After Reset:** 0x0

63:1	0
(undef)	TEST_SLV_IF

DMA Test register

Bits	Name	Memory Access	Description
------	------	---------------	-------------

Bits	Name	Memory Access	Description
63:1			Reserved for future use.
0	TEST_SLV_IF	R/W	DMA Test register

### 5.3.2.5.5 DMA\_COMP\_PARAMS\_6

**Size:** 64 bits

**Offset:** 0x3c8

**Memory Access:** R

**Value After Reset:** 0x1001201200000000

63	62:60	59: 57	56: 54	53: 51	50:48	47: 46	45	44	43	42	41	40	39	38	37: 35	34: 32	31: 0
(u nd ef )	CH7_ FIFO_ DEPT H	CH 7_ SM S	CH 7_ LM S	CH 7_ D MS	CH7_M AX_MU LT_SIZE	C H 7_ FC	CH7_ HC _LL P	CH7_ CTL _WB_E N	CH7_ MULTI _BLK_E N	CH7_ LOCK _LOC E N	CH7_S RC_G AT_EN	CH7_ DST_S CA_E N	CH7_ STA T_SR C	CH7_ STA T_DS T	CH 7_ ST W	CH 7_ DT W	(u nd ef )

Component Parameters registers

Bits	Name	Memory Access	Description
63			Reserved for future use.
62:60	CH7_FIFO_DEPTH	R	CH7_FIFO_DEPTH
59:57	CH7_SMS	R	CH7_SMS
56:54	CH7_LMS	R	CH7_LMS
53:51	CH7_DMS	R	CH7_DMS
50:48	CH7_MAX_MULT_SIZE	R	CH7_MAX_MULT_SIZE
47:46	CH7_FC	R	CH7_FC
45	CH7_HC_LL_P	R	CH7_HC_LL_P
44	CH7_CTL_WB_EN	R	CH7_CTL_WB_EN
43	CH7_MULTI_BLK_EN	R	CH7_MULTI_BLK_EN
42	CH7_LOCK_EN	R	CH7_LOCK_EN
41	CH7_SRC_GAT_EN	R	CH7_SRC_GAT_EN
40	CH7_DST_SCA_EN	R	CH7_DST_SCA_EN
39	CH7_STAT_SRC	R	CH7_STAT_SRC
38	CH7_STAT_DST	R	CH7_STAT_DST
37:35	CH7_STW	R	DMAH_CH7_STW
34:32	CH7_DTW	R	DMAH_CH7_DTW
31:0			Reserved for future use.

### **5.3.2.5.6 DMA\_COMP\_PARAMS\_5**

**Size:** 64 bits

**Offset:** 0x3d0

## Memory Access: R

**Value After Reset:** 0x1001201210012012

6 3	6 2: 6 0	5 9 : 5	5 6 : 5	5 3 : 5	50 : 48	4 7 : 46	4 4	4 5	4 4	4 3	4 2	4 1	4 0	3 9	3 8	3 7 : 35	3 4 : 32	3 1	3 0 : 28	2 7 : 25	2 4 : 22	2 1 : 19	18 : 16	1 5 : 14	1 1 : 13	1 1 : 12	1 1 : 11	1 0	9	8	7	6	5 : 3	2 : 0
( u n d e f )	C H 5 — F I O — D E P T H	C H 5 — C H 5 — S M — L M S	C H 5 — H C — F C	C H 5 — C T L — W B — E N	C H 5 — M U L T I — B L K — E N	C H 5 — S R C — G A T — E N	C H 5 — D S T — S C A — E N	C H 5 — S T A — S R C — D S T	C H 5 — D T W — S T W	C H 6 — F I O — D E P T H	C H 6 — F O — D E P T H	C H 6 — M A X — M U L T — S I Z E	C H 6 — H C — F C	C H 6 — C T L — W B — L L P	C H 6 — M U L T I — B L K — E N	C H 6 — S R C — G A T — S C A — E N	C H 6 — S T A — S C A — E N	C H 6 — S T A — S R C — D S T	C H 6 — S T A — D S T															

## Component Parameters registers

Bits	Name	Memory Access	Description
63			Reserved for future use.
62:60	CH5_FIFO_DEPTH	R	CH5_FIFO_DEPTH
59:57	CH5_SMS	R	CH5_SMS
56:54	CH5_LMS	R	CH5_LMS
53:51	CH5_DMS	R	CH5_DMS
50:48	CH5_MAX_MULT_SIZE	R	CH5_MAX_MULT_SIZE
47:46	CH5_FC	R	CH5_FC
45	CH5_HC LLP	R	CH5_HC LLP
44	CH5_CTL_WB_EN	R	CH5_CTL_WB_EN
43	CH5_MULTI_BLK_EN	R	CH5_MULTI_BLK_EN
42	CH5_LOCK_EN	R	CH5_LOCK_EN
41	CH5_SRC_GAT_EN	R	CH5_SRC_GAT_EN
40	CH5_DST_SCA_EN	R	CH5_DST_SCA_EN
39	CH5_STAT_SRC	R	CH5_STAT_SRC
38	CH5_STAT_DST	R	CH5_STAT_DST
37:35	CH5_STW	R	DMAH_CH5_STW

Bits	Name	Memory Access	Description
34:32	CH5_DTW	R	DMAH_CH5_DTW
31			Reserved for future use.
30:28	CH6_FIFO_DEPTH	R	CH6_FIFO_DEPTH
27:25	CH6_SMS	R	CH6_SMS
24:22	CH6_LMS	R	CH6_LMS
21:19	CH6_DMS	R	CH6_DMS
18:16	CH6_MAX_MULT_SIZE	R	CH6_MAX_MULT_SIZE
15:14	CH6_FC	R	CH6_FC
13	CH6_HC LLP	R	CH6_HC LLP
12	CH6_CTL_WB_EN	R	CH6_CTL_WB_EN
11	CH6_MULTI_BLK_EN	R	CH6_MULTI_BLK_EN
10	CH6_LOCK_EN	R	CH6_LOCK_EN
9	CH6_SRC_GAT_EN	R	CH6_SRC_GAT_EN
8	CH6_DST_SCA_EN	R	CH6_DST_SCA_EN
7	CH6_STAT_SRC	R	CH6_STAT_SRC
6	CH6_STAT_DST	R	CH6_STAT_DST
5:3	CH6_STW	R	DMAH_CH6_STW
2:0	CH6_DTW	R	DMAH_CH6_DTW

### 5.3.2.5.7 DMA\_COMP\_PARAMS\_4

**Size:** 64 bits

**Offset:** 0x3d8

**Memory Access:** R

**Value After Reset:** 0x1001201210012012

6	5	5	5	5	4	4	4	4	4	4	3	3	3	3	3	2	2	2	2	18	1	1	1	1	1	1	5	2					
2: 3	9: 6	5: 6	5: 3	5: 4	7: 4	4: 5	4: 4	4: 3	4: 2	4: 1	4: 0	3: 9	3: 8	3: 8	3: 7	3: 4	3: 3	3: 2	3: 1	3: 2	2: 5	2: 2	2: 1	2: 9	1: 6	1: 4	1: 3	1: 2	1: 0	9: 8	7: 6	5: 3	2: 0
(	C	H	C	H	C	H	C	H	C	H	C	H	C	H	C	H	C	H	C	H	C	H	C	H	C	H	C	H	C	H	C	C	
u	C	C	C	C	3	M	C	H	3	3	3	3	3	3	3	C	H	C	H	4	C	C	C	H	4	C	H	4	C	H	C	H	C
d	H	H	H	H	A	X	H	—	M	L	S	D	S	S	S	H	H	H	H	4	H	H	H	M	—	S	D	S	T	A	S	H	4
e	F	—	—	—	D	M	X	—	U	O	R	S	T	T	T	—	—	—	—	4	H	H	H	M	—	S	D	S	T	A	S	H	4
f	O	S	L	D	M	UL	M	—	L	T	C	C	T	A	T	S	D	S	D	4	H	H	H	M	—	S	D	S	T	A	S	H	4
)	D	M	M	M	UL	—	L	—	I	K	—	G	—	S	—	W	W	—	—	4	H	H	H	M	—	S	D	S	T	A	S	H	4
E	S	S	S	T	SI	—	L	—	B	—	G	—	S	—	D	W	W	—	—	4	H	H	H	M	—	S	D	S	T	A	S	H	4
P	Z	E	—	—	—	—	—	—	K	N	T	A	R	S	—	—	—	—	—	4	H	H	H	M	—	S	D	S	T	A	S	H	4

T	H						E	N	—	E	N	—	E	N	C	T			T	H						E	N	—	E	N	—	E	N	C	T		
---	---	--	--	--	--	--	---	---	---	---	---	---	---	---	---	---	--	--	---	---	--	--	--	--	--	---	---	---	---	---	---	---	---	---	---	--	--

## Component Parameters registers

Bits	Name	Memory Access	Description
63			Reserved for future use.
62:60	CH3_FIFO_DEPTH	R	CH3_FIFO_DEPTH
59:57	CH3_SMS	R	CH3_SMS
56:54	CH3_LMS	R	CH3_LMS
53:51	CH3_DMS	R	CH3_DMS
50:48	CH3_MAX_MULT_SIZE	R	CH3_MAX_MULT_SIZE
47:46	CH3_FC	R	CH3_FC
45	CH3_HC LLP	R	CH3_HC LLP
44	CH3_CTL_WB_EN	R	CH3_CTL_WB_EN
43	CH3_MULTI_BLK_EN	R	CH3_MULTI_BLK_EN
42	CH3_LOCK_EN	R	CH3_LOCK_EN
41	CH3_SRC_GAT_EN	R	CH3_SRC_GAT_EN
40	CH3_DST_SCA_EN	R	CH3_DST_SCA_EN
39	CH3_STAT_SRC	R	CH3_STAT_SRC
38	CH3_STAT_DST	R	CH3_STAT_DST
37:35	CH3_STW	R	DMAH_CH3_STW
34:32	CH3_DTW	R	DMAH_CH3_DTW
31			Reserved for future use.
30:28	CH4_FIFO_DEPTH	R	CH4_FIFO_DEPTH
27:25	CH4_SMS	R	CH4_SMS
24:22	CH4_LMS	R	CH4_LMS
21:19	CH4_DMS	R	CH4_DMS
18:16	CH4_MAX_MULT_SIZE	R	CH4_MAX_MULT_SIZE
15:14	CH4_FC	R	CH4_FC
13	CH4_HC LLP	R	CH4_HC LLP
12	CH4_CTL_WB_EN	R	CH4_CTL_WB_EN
11	CH4_MULTI_BLK_EN	R	CH4_MULTI_BLK_EN
10	CH4_LOCK_EN	R	CH4_LOCK_EN
9	CH4_SRC_GAT_EN	R	CH4_SRC_GAT_EN

Bits	Name	Memory Access	Description
8	CH4_DST_SCA_EN	R	CH4_DST_SCA_EN
7	CH4_STAT_SRC	R	CH4_STAT_SRC
6	CH4_STAT_DST	R	CH4_STAT_DST
5:3	CH4_STW	R	DMAH_CH4_STW
2:0	CH4_DTW	R	DMAH_CH4_DTW

#### **5.3.2.5.8 DMA\_COMP\_PARAMS\_3**

**Size:** 64 bits

**Offset:** 0x3e0

## Memory Access: R

**Value After Reset:** 0x5001201210012012

## Component Parameters registers

Bits	Name	Memory Access	Description
63			Reserved for future use.
62:60	CH1_FIFO_DEPTH	R	CH1_FIFO_DEPTH
59:57	CH1_SMS	R	CH1_SMS
56:54	CH1_LMS	R	CH1_LMS
53:51	CH1_DMS	R	CH1_DMS
50:48	CH1_MAX_MULT_SIZE	R	CH1_MAX_MULT_SIZE
47:46	CH1_FC	R	CH1_FC
45	CH1_HC_LL_P	R	CH1_HC_LL_P
44	CH1_CTL_WB_EN	R	CH1_CTL_WB_EN

Bits	Name	Memory Access	Description
43	CH1_MULTI_BLK_EN	R	CH1_MULTI_BLK_EN
42	CH1_LOCK_EN	R	CH1_LOCK_EN
41	CH1_SRC_GAT_EN	R	CH1_SRC_GAT_EN
40	CH1_DST_SCA_EN	R	CH1_DST_SCA_EN
39	CH1_STAT_SRC	R	CH1_STAT_SRC
38	CH1_STAT_DST	R	CH1_STAT_DST
37:35	CH1_STW	R	DMAH_CH1_STW
34:32	CH1_DTW	R	DMAH_CH1_DTW
31			Reserved for future use.
30:28	CH2_FIFO_DEPTH	R	CH2_FIFO_DEPTH
27:25	CH2_SMS	R	CH2_SMS
24:22	CH2_LMS	R	CH2_LMS
21:19	CH2_DMS	R	CH2_DMS
18:16	CH2_MAX_MULT_SIZE	R	CH2_MAX_MULT_SIZE
15:14	CH2_FC	R	CH2_FC
13	CH2_HC LLP	R	CH2_HC LLP
12	CH2_CTL_WB_EN	R	CH2_CTL_WB_EN
11	CH2_MULTI_BLK_EN	R	CH2_MULTI_BLK_EN
10	CH2_LOCK_EN	R	CH2_LOCK_EN
9	CH2_SRC_GAT_EN	R	CH2_SRC_GAT_EN
8	CH2_DST_SCA_EN	R	CH2_DST_SCA_EN
7	CH2_STAT_SRC	R	CH2_STAT_SRC
6	CH2_STAT_DST	R	CH2_STAT_DST
5:3	CH2_STW	R	DMAH_CH2_STW
2:0	CH2_DTW	R	DMAH_CH2_DTW

### 5.3.2.5.9 DMA\_COMP\_PARAMS\_2

**Size:** 64 bits

**Offset:** 0x3e8

**Memory Access:** R

**Value After Reset:** 0x50012012

63: 60	59: 56	55: 52	51: 48	47: 44	43: 40	39: 36	35: 32	3 1	30: 8	2: 2	2: 2	2: 1	18: 16	1: 1	5: 3	1: 1	12	11	10	9	8	7	6	5: 3	2: 0
-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	--------	----------	---------	---------	---------	-----------	---------	---------	---------	----	----	----	---	---	---	---	---------	---------

## Component Parameters registers

Bits	Name	Memory Access	Description
63:60	CH7_MULTI_BLK_TYPE	R	DMAH_CH7_MULTI_BLK_TYPE
59:56	CH6_MULTI_BLK_TYPE	R	DMAH_CH6_MULTI_BLK_TYPE
55:52	CH5_MULTI_BLK_TYPE	R	DMAH_CH5_MULTI_BLK_TYPE
51:48	CH4_MULTI_BLK_TYPE	R	DMAH_CH4_MULTI_BLK_TYPE
47:44	CH3_MULTI_BLK_TYPE	R	DMAH_CH3_MULTI_BLK_TYPE
43:40	CH2_MULTI_BLK_TYPE	R	DMAH_CH2_MULTI_BLK_TYPE
39:36	CH1_MULTI_BLK_TYPE	R	DMAH_CH1_MULTI_BLK_TYPE
35:32	CHO_MULTI_BLK_TYPE	R	DMAH_CHO_MULTI_BLK_TYPE
31			Reserved for future use.
30:28	CHO_FIFO_DEPTH	R	CHO_FIFO_DEPTH
27:25	CHO_SMS	R	CHO_SMS
24:22	CHO_LMS	R	CHO_LMS
21:19	CHO_DMS	R	CHO_DMS
18:16	CHO_MAX_MULT_SIZE	R	CHO_MAX_MULT_SIZE
15:14	CHO_FC	R	CHO_FC
13	CHO_HC LLP	R	CHO_HC LLP
12	CHO_CTL_WB_EN	R	CHO_CTL_WB_EN
11	CHO_MULTI_BLK_EN	R	CHO_MULTI_BLK_EN
10	CHO_LOCK_EN	R	CHO_LOCK_EN
9	CHO_SRC_GAT_EN	R	CHO_SRC_GAT_EN
8	CHO_DST_SCA_EN	R	CHO_DST_SCA_EN
7	CHO_STAT_SRC	R	CHO_STAT_SRC
6	CHO_STAT_DST	R	CHO_STAT_DST
5:3	CHO_STW	R	DMAH_CHO_STW

Bits	Name	Memory Access	Description
2:0	CHO_DTW	R	DMAH_CHO_DTW

**5.3.2.5.10 DMA\_COMP\_PARAMS\_1****Size:** 64 bits**Offset:** 0x3f0**Memory Access:** R**Value After Reset:** 0x3000010433333388

6 3 : 6 2	61	60	59 :5 5	54: 53	52: 51	50: 49	48: 47	46: 45	44: 43	42: 40	39 : 36	35	34 : 33	32	31: 28	27: 24	23: 20	19: 16	15: 12	11: 8	7:4	3:0
( u n d e f ) T	STA TIC_ END IAN _SE LEC	ADD _EN COD ED_ PAR AMS	N U M — H S_ IN T	M4 _H DA — W IDT H	M3 _H DA — W IDT H	M2 _H DA — W IDT H	M1 _H DA — W IDT H	S_HD AT A_WI DT	NU M_MA STE_R_I NT	NU M_MA HA_NN EL_S	( u n d e f ) R	M A_X — A_B R_ST	I N T R — B R	BI G — E N DI A N	CH 7_M X_BLK —_SI ZE	CH 6_M X_BLK —_SI ZE	CH 5_M X_BLK —_SI ZE	CH 4_M X_BLK —_SI ZE	CH 3_M X_BLK —_SI ZE	CH 2_M X_BLK —_SI ZE	CH 1_M X_LK_ SIZ_E	CH O_MA X_B LK_SIZ_E

Component Parameters registers

Bits	Name	Memory Access	Description
63:62			Reserved for future use.
61	STATIC_ENDIAN_SELECT	R	DMAH_STATIC_ENDIAN_SELECT
60	ADD_ENCODED_PARAMS	R	DMAH_ADD_ENCODED_PARAMS
59:55	NUM_HS_INT	R	DMAH_NUM_HS_INT
54:53	M4_HDATA_WIDTH	R	DMAH_M4_HDATA_WIDTH
52:51	M3_HDATA_WIDTH	R	DMAH_M3_HDATA_WIDTH
50:49	M2_HDATA_WIDTH	R	DMAH_M2_HDATA_WIDTH
48:47	M1_HDATA_WIDTH	R	DMAH_M1_HDATA_WIDTH
46:45	S_HDATA_WIDTH	R	DMAH_S_HDATA_WIDTH
44:43	NUM_MASTER_INT	R	DMAH_NUM_MASTER_INT
42:40	NUM_CHANNELS	R	DMAH_NUM_CHANNELS
39:36			Reserved for future use.
35	MAX_ABRST	R	DMAH_MAX_ABRST
34:33	INTR_IO	R	DMAH_INTR_IO
32	BIG_ENDIAN	R	DMAH_BIG_ENDIAN
31:28	CH7_MAX_BLK_SIZE	R	DMAH_CH7_MAX_BLK_SIZE
27:24	CH6_MAX_BLK_SIZE	R	DMAH_CH6_MAX_BLK_SIZE

Bits	Name	Memory Access	Description
23:20	CH5_MAX_BLK_SIZE	R	DMAH_CH5_MAX_BLK_SIZE
19:16	CH4_MAX_BLK_SIZE	R	DMAH_CH4_MAX_BLK_SIZE
15:12	CH3_MAX_BLK_SIZE	R	DMAH_CH3_MAX_BLK_SIZE
11:8	CH2_MAX_BLK_SIZE	R	DMAH_CH2_MAX_BLK_SIZE
7:4	CH1_MAX_BLK_SIZE	R	DMAH_CH1_MAX_BLK_SIZE
3:0	CHO_MAX_BLK_SIZE	R	DMAH_CHO_MAX_BLK_SIZE

### 5.3.2.5.11 DmaCompsID

**Size:** 64 bits

**Offset:** 0x3f8

**Memory Access:** R

**Value After Reset:** 0x3231372a44571110

63:32	31:0
DMA_COMP_VERSION	DMA_COMP_TYPE

DMA Component ID register

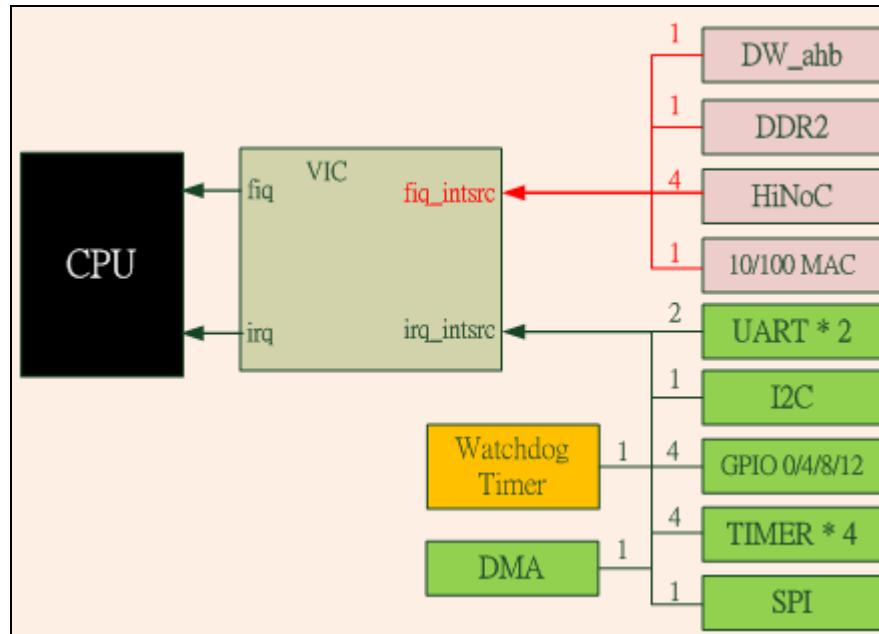
Bits	Name	Memory Access	Description
63:32	DMA_COMP_VERSION	R	DMA Component Version - See release notes.
31:0	DMA_COMP_TYPE	R	DMA Component identifier - Fixed at `h44571110

## 6 Interrupts (DW\_ICTL)

### 6.1 ICTL Function Description

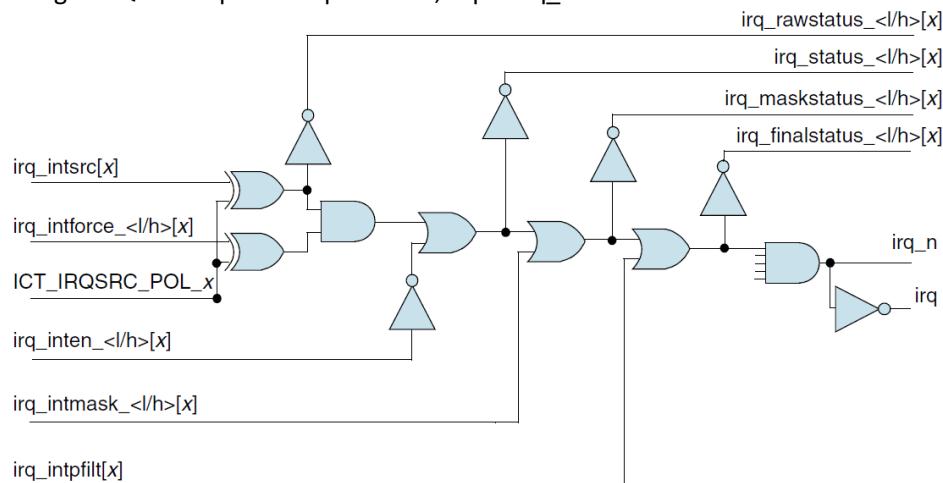
#### 6.1.1 Overview

It supports 14 normal interrupt (IRQ) sources that are processed to produce a single IRQ interrupt to the processor. It supports from 7 fast interrupt (FIQ) sources that are processed to produce a single FIQ interrupt to the processor. All interrupt processing is combinational so that interrupts are propagated if the bus interface of the ICTL is powered down. This means that reading any of the interrupt status registers (raw, status, or final\_status) is simply returning the status of the combinational logic, since there are no flip-flops associated with these registers. It is the user's responsibility to make sure that the interrupts stay asserted until they are serviced.



#### 6.1.2 IRQ Interrupt Processing

The ICTL can be [configured](#) to support from 14 IRQ interrupt sources (`irq_intsrcN`). The ICTL processes these interrupt sources to produce a single IRQ interrupt to the processor; `irq` or `irq_n`.



##### 6.1.2.1 IRQ Interrupt Polarity

The input polarity of each IRQ interrupt source is individually [configurable](#). This parameter also determines the polarity of the software-programmable interrupt force bits in the `irq_intforce_l` or `irq_intforce_h`.

All interrupt status registers are always active-high, regardless of the polarity [configured](#) for the interrupt sources and outputs.

### 6.1.2.2 IRQ Software-Programmable Interrupt

The ICTL supports forcing interrupts from software. To force an interrupt to be active, write to the corresponding bit in the irq\_intforce registers ([irq\\_intforce\\_l](#) or [irq\\_intforce\\_h](#)). The polarity of each bit in these registers is the same as the polarity of the corresponding interrupt source signal.

### 6.1.2.3 IRQ Enable and Masking

To enable each interrupt source independently, write a 1 to the corresponding bit of the irq\_inten registers.

To mask each interrupt source independently, write a 1 to the corresponding bit of the interrupt mask register. The reset value for each mask bit is 0 (unmasked).

### 6.1.2.4 IRQ Software-Programmable Priority Levels

The ICTL supports optional software programmable priority levels. To change the priority level of an interrupt, you write the priority value to the corresponding priority level register in the memory map. There is a priority register for each of the interrupt sources, which can be programmed to one of sixteen values from 0x0 to 0xf. Priority registers exist for only available interrupt sources.

### 6.1.2.5 IRQ Interrupt Status Registers

The ICTL includes up to four status registers used for querying the current status of any interrupt at various stages of the processing. All of the following status registers have the same polarity; a 1 indicates that an interrupt is active, a 0 indicates it is inactive.

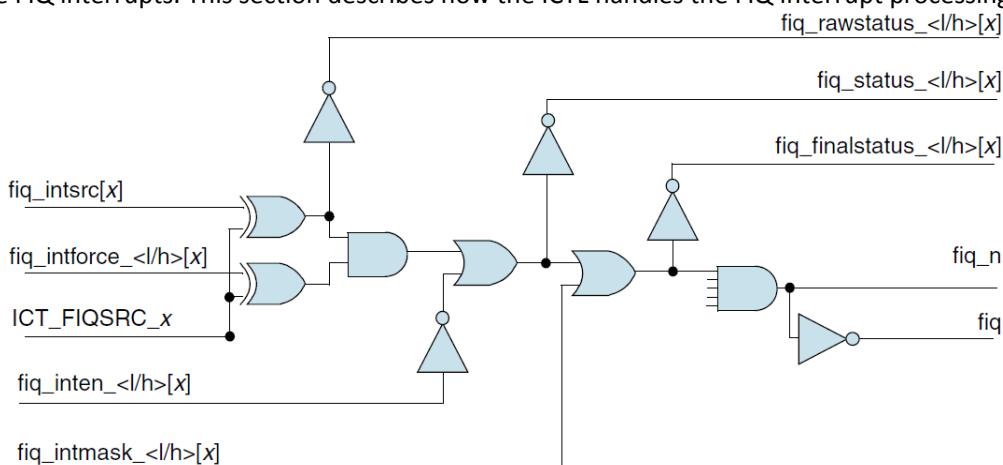
- [irq\\_rawstatus](#) ([irq\\_rawstatus\\_l](#)) – Contains the state of the interrupt sources after being adjusted for input polarity. Each bit of this register is set to 1 if the corresponding interrupt source bit is active, and is set to 0 if it is inactive.
- [irq\\_status](#) ([irq\\_status\\_l](#)) – Contains the state of all interrupts after the enabling stage; that is, an active-high bit indicates that a particular interrupt source is active and enabled.
- [irq\\_maskstatus](#) ([irq\\_maskstatus\\_l](#)) – Contains the state of all interrupts after the masking stage; that is, an active-high bit indicates that a particular interrupt source is active, enabled, and not masked.
- [irq\\_finalstatus](#) ([irq\\_finalstatus\\_l](#)) – Contains the state of all interrupts after the priority filtering stage; that is, an active-high bit indicates that particular interrupt source is active, enabled, not masked, and its [configured](#) priority level is greater or equal to the value programmed in the [irq\\_plevel](#) register. If priority filtering has not been selected, this register contains the same value as the [irq\\_maskstatus](#) register (the final stage of processing).

### 6.1.3 FIQ Interrupt Processing

FIQ interrupts are an optional feature of the ICTL interrupt controller.

You can [configure](#) the ICTL to support 7 ~~from 1 to 8~~ FIQ interrupt sources ([fiq\\_intsrcN](#)). The ICTL processes these interrupt sources to produce a single FIQ interrupt to the processor; [fiq](#) or [fiq\\_n](#).

FIQ interrupt processing is similar to IRQ interrupt processing, except that priority filtering and interrupt vectors are not supported for the FIQ interrupts. This section describes how the ICTL handles the FIQ interrupt processing.



### 6.1.3.1 FIQ Interrupt Polarity

The input polarity of each FIQ interrupt source-active-high;

All interrupt status registers are always active-high,

### 6.1.3.2 FIQ Software-Programmable Interrupts

The ICTL supports forcing interrupts from software. You force an interrupt to be active by writing to the corresponding bit in the [fiq\\_intforce](#) register. The polarity of each bit in this register is the same as the polarity of the corresponding interrupt source signal.

Regardless of the polarity you [configure](#), the reset state of each bit in the [fiq\\_intforce](#) register is always inactive.

### 6.1.3.3 FIQ Enable and Masking

You can enable each interrupt source independently by writing a 1 to the corresponding bit of the [fiq\\_inten](#) register. the corresponding bit of the [fiq\\_inten](#) register is 1 on reset, which enables the corresponding interrupt source.

You can mask each interrupt source independently by writing a 1 to the corresponding bit of the [fiq\\_intmask](#) register. The reset value for each mask bit is 0; that is, unmasked.

### 6.1.3.4 FIQ Interrupt Status Registers

The ICTL includes three status registers that you can use to query the current status of any FIQ interrupt at various stages of the processing. All of the following status registers have the same polarity; a 1 indicates that an interrupt is active, a 0 indicates inactive.

- [fiq\\_rawstatus](#) – Contains the state of the interrupt sources after being adjusted for input polarity. Each bit of this register is set to 1 if the corresponding interrupt source bit is active; it is set to 0 if it is inactive.
- [fiq\\_status](#) – Contains the state of all interrupts after the enabling stage; that is, an active-high bit indicates that a particular interrupt source is active and enabled.
- [fiq\\_finalstatus](#) – Contains the state of all interrupts after the masking; that is an active-high bit indicates that a particular interrupt source is active, enabled, and unmasked.

## 6.2 ICTL Programming

### 6.2.1 Initialization

A normal initialization sequence is as follows:

1. Disable all interrupts by writing to the [irq\\_inten](#) and [fiq\\_inten](#).
2. Initialize peripheral devices that could generate interrupts.
3. Program the, [irq\\_plevel](#), [irq\\_intmask](#), and [fiq\\_intmask](#) registers as appropriate.
4. Enable interrupts.

### 6.2.2 Interrupt Service

Without vectored interrupts, a normal interrupt servicing sequence is as follows:

1. Poll the interrupt status register ([irq\\_finalstatus](#), [irq\\_maskstatus](#), or [fiq\\_finalstatus](#), as appropriate) to determine which interrupt source caused the interrupt.
2. Service the interrupt.
3. Optionally read the [irq\\_status](#) or [fiq\\_status](#) register to see if other currently masked interrupts are pending; service these as required.

With vectored interrupt support, a normal interrupt servicing sequence is as follows:

1. Read the [irq\\_vector](#) register to get the address of the service routine.
2. Service routine reads [irq\\_finalstatus](#) to see which interrupt sources caused the interrupt.
3. Service the interrupt.
4. Optionally read the [irq\\_status](#) to see if other interrupts are pending; service these as required.

## 6.3 Interrupt Mapping

### 6.3.1 Register Memory Maps

Register	Offset	Size	Memory Access	Description
DW_apb_ictl address block				
<a href="#">IRQ_INTEN_L</a>	0x0	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Interrupt Source Enable (Low) Register Size: 2-32 bits Address Offset: 0x00 Read/Write Access: Read/Write
<a href="#">IRQ_INTMASK_L</a>	0x8	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Interrupt Source Mask (Low) Register Size: 2-32 bits Address Offset: 0x08 Read/Write Access: Read/Write
<a href="#">IRQ_INTFORCE_L</a>	0x10	32 bits	R/W	<b>Value After Reset:</b> 0x2000 Name: Interrupt Force (Low) Register Size: 2-32 bits Address Offset: 0x10 Read/Write Access: Read/Write
<a href="#">IRQ_RAWSTATUS_L</a>	0x18	32 bits	R	Name: Interrupt Raw Status (Low) Register Size: 2-32 bits Address Offset: 0x18 Read/Write Access: Read/Write
<a href="#">IRQ_STATUS_L</a>	0x20	32 bits	R	Name: Interrupt Status (Low) Register Size: 2-32 bits Address Offset: 0x20 Read/Write Access: Read
<a href="#">IRQ_MASKSTATUS_L</a>	0x28	32 bits	R	Name: Interrupt Mask Status (Low) Register Size: 2-32 bits Address Offset: 0x28 Read/Write Access: Read
<a href="#">IRQ_FINALSTATUS_L</a>	0x30	32 bits	R	Name: Interrupt Final Status (Low) Register Size: 2-32 bits Address Offset: 0x30 Read/Write Access: Read
<a href="#">IRQ_VECTOR</a>	0x38	32 bits	R	<b>Value After Reset:</b> 0x0 Name: IRQ Vector Register Size: 32 bits Address Offset: 0x38 Read/Write Access: Read
<a href="#">FIQ_INTEN</a>	0xc0	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Fast Interrupt Enable Register Size: 1-8 bits Address Offset: 0xc0 Read/Write Access: Read/Write
<a href="#">FIQ_INTMASK</a>	0xc4	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Fast Interrupt Mask Register Size: 1-8 bits Address Offset: 0xc4 Read/Write Access: Read/Write
<a href="#">FIQ_INTFORCE</a>	0xc8	32 bits	R/W	<b>Value After Reset:</b> 0x40 Name: Fast Interrupt Force Register Size: 1-8 bits Address Offset: 0xc8 Read/Write Access: Read/Write
<a href="#">FIQ_RAWSTATUS</a>	0xcc	32 bits	R	Name: Fast Interrupt Source Raw Status Register Size: 1-8 bits Address Offset: 0xcc Read/Write Access: Read
<a href="#">FIQ_STATUS</a>	0xd0	32 bits	R	Name: Fast Interrupt Status Register Size: 1-8 bits Address Offset: 0xd0 Read/Write Access: Read
<a href="#">FIQ_FINALSTATUS</a>	0xd4	32 bits	R	Name: Fast Interrupt Final Status Register Size: 1-8 bits Address Offset: 0xd4 Read/Write Access: Read
<a href="#">IRQ_PLEVEL</a>	0xd8	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: IRQ System Priority Level Register Size: 4 bits Address Offset: 0xd8 Read/Write Access: Read/Write

Register	Offset	Size	Memory Access	Description
<a href="#">ICTL_VERSION_ID</a>	0xe0	32 bits	R	<b>Value After Reset:</b> 0x3230352a Name: Component Version Register Size: 32 bits Address Offset: 0xe0 Read/Write Access: Read

### 6.3.2 Register and Field Descriptions

Following is a description of the individual registers of component DW\_apb\_ictl

#### 6.3.2.1 IRQ\_INTEN\_L

**Size:** 32 bits

**Offset:** 0x0

**Memory Access:** R/W

**Value After Reset:** 0x0

31:14	13:0
(undef)	IRQ_INTEN_L

Name: Interrupt Source Enable (Low) Register Size: 2-32 bits Address Offset: 0x00 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:14			Reserved for future use.
13:0	IRQ_INTEN_L	R/W	Interrupt enable bits for lower 32 interrupt sources. A 1 in any bit position enables the corresponding interrupt. 0 disable interrupt 1 enable interrupt

#### 6.3.2.2 IRQ\_INTMASK\_L

**Size:** 32 bits

**Offset:** 0x8

**Memory Access:** R/W

**Value After Reset:** 0x0

31:14	13:0
(undef)	IRQ_INTMASK_L

Name: Interrupt Source Mask (Low) Register Size: 2-32 bits Address Offset: 0x08 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:14			Reserved for future use.
13:0	IRQ_INTMASK_L	R/W	Interrupt mask bits for the lower 32 interrupt sources. A 1 in any bit position masks (disables) the corresponding interrupt. By default, all bits are unmasks. 0 unmask interrupt 1 mask interrupt

#### 6.3.2.3 IRQ\_INFORCE\_L

**Size:** 32 bits

**Offset:** 0x10

**Memory Access:** R/W

**Value After Reset:** 0x2000

31:14	13:0
(undef)	IRQ_INFORCE_L

Name: Interrupt Force (Low) Register Size: 2-32 bits Address Offset: 0x10 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:14			Reserved for future use.
13:0	IRQ_INTFORCE_L	R/W	Interrupt force bits for the lower 32 interrupt sources. Each bit corresponds to one bit of the irq_intsrc input. The polarity of the bits in the register correspond to the polarity of the associated irq_intsrc input. If the interrupt input is configured to be active-high, the corresponding bit in the register is also active-high. 0 active-low 1 active-high

#### 6.3.2.4 IRQ\_RAWSTATUS\_L

**Size:** 32 bits

**Offset:** 0x18

**Memory Access:** R

31:14	13:0
(undef)	IRQ_RAWSTATUS_L

Name: Interrupt Raw Status (Low) Register Size: 2-32 bits Address Offset: 0x18 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:14			Reserved for future use.
13:0	IRQ_RAWSTATUS_L	R	Actual interrupt source; these are the lower 32 interrupt sources.

#### 6.3.2.5 IRQ\_STATUS\_L

**Size:** 32 bits

**Offset:** 0x20

**Memory Access:** R

31:14	13:0
(undef)	IRQ_STATUS_L

Name: Interrupt Status (Low) Register Size: 2-32 bits Address Offset: 0x20 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:14			Reserved for future use.
13:0	IRQ_STATUS_L	R	Interrupt status after the forcing and interrupt enabling stage. These are the interrupt status signals for the lower 32 interrupt sources.

#### 6.3.2.6 IRQ\_MASKSTATUS\_L

**Size:** 32 bits

**Offset:** 0x28

**Memory Access:** R

31:14	13:0
(undef)	IRQ_MASKSTATUS_L

Name: Interrupt Mask Status (Low) Register Size: 2-32 bits Address Offset: 0x28 Read/Write Access: Read

Bits	Name	Memory Access	Description

Bits	Name	Memory Access	Description
31:14			Reserved for future use.
13:0	IRQ_MASKSTATUS_L	R	Interrupt status after the masking stage. These are the interrupt status signals for the lower 32 interrupt sources.

### 6.3.2.7 IRQ\_FINALSTATUS\_L

**Size:** 32 bits

**Offset:** 0x30

**Memory Access:** R

31:14	13:0
(undef)	IRQ_FINALSTATUS_L

Name: Interrupt Final Status (Low) Register Size: 2-32 bits Address Offset: 0x30 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:14			Reserved for future use.
13:0	IRQ_FINALSTATUS_L	R	Interrupt status after the priority level filtering stage. These are the interrupt status signals for the lower 32 interrupt sources. If there is no priority interrupt scheme configured, then this location contains the same value as irq_maskstatus_l.

### 6.3.2.8 IRQ\_VECTOR

**Size:** 32 bits

**Offset:** 0x38

**Memory Access:** R

**Value After Reset:** 0x0

31:0
IRQ_VECTOR

Name: IRQ Vector Register Size: 32 bits Address Offset: 0x38 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:0	IRQ_VECTOR	R	When an interrupt occurs, and provided vectored interrupts are supported, this location can be read; it returns one of the 16 vectors. The returned vector corresponds to the highest priority level interrupt. This register returns 0 when ICT_HAS_VECTOR = 0. This register is read coherent, allowing the register to be read, regardless of the data bus width.

### 6.3.2.9 FIQ\_INTEN

**Size:** 32 bits

**Offset:** 0xc0

**Memory Access:** R/W

**Value After Reset:** 0x0

31:7	6:0
(undef)	FIQ_INTEN

Name: Fast Interrupt Enable Register Size: 1-8 bits Address Offset: 0xc0 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:7			Reserved for future use.
6:0	FIQ_INTEN	R/W	Fast interrupt enable bits. A 1 in any bit position enables the corresponding interrupt. This register does not exist when ICT_HAS_FIQ = 0. 0 disable interrupt 1 enable interrupt

### 6.3.2.10 FIQ\_INTMASK

**Size:** 32 bits

**Offset:** 0xc4

**Memory Access:** R/W

**Value After Reset:** 0x0

31:7	6:0
(undef)	FIQ_INTMASK

Name: Fast Interrupt Mask Register Size: 1-8 bits Address Offset: 0xc4 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:7			Reserved for future use.
6:0	FIQ_INTMASK	R/W	Fast interrupt mask bits. A 1 in any bit position masks the corresponding interrupt. This register does not exist when ICT_HAS_FIQ = 0. 0 unmask interrupt 1 mask interrupt

### 6.3.2.11 FIQ\_INFORCE

**Size:** 32 bits

**Offset:** 0xc8

**Memory Access:** R/W

**Value After Reset:** 0x40

31:7	6:0
(undef)	FIQ_INFORCE

Name: Fast Interrupt Force Register Size: 1-8 bits Address Offset: 0xc8 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:7			Reserved for future use.
6:0	FIQ_INFORCE	R/W	Fast interrupt force bits. Each bit in this register corresponds to one bit of the irq_intsrc input. The polarity of the bits in the register corresponds to the polarity of the associated fiq_intsrc input. If the interrupt input is configured to be active-high, the corresponding bit in the register is also active-high. This register does not exist when ICT_HAS_FIQ = 0. 0 active-low 1 active-high

### 6.3.2.12 FIQ\_RAWSTATUS

**Size:** 32 bits

**Offset:** 0xcc

**Memory Access:** R

31:7	6:0
(undef)	FIQ_RAWSTATUS

Name: Fast Interrupt Source Raw Status Register Size: 1-8 bits Address Offset: 0xcc Read/Write Access: Read

Bits	Name	Memory Access	Description
31:7			Reserved for future use.
6:0	FIQ_RAWSTATUS	R	Fast interrupt source raw input status. This register does not exist when ICT_HAS_FIQ = 0.

### 6.3.2.13 FIQ\_STATUS

**Size:** 32 bits

**Offset:** 0xd0

**Memory Access:** R

31:7	6:0
(undef)	FIQ_STATUS

Name: Fast Interrupt Status Register Size: 1-8 bits Address Offset: 0xd0 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:7			Reserved for future use.
6:0	FIQ_STATUS	R	Fast interrupt status after the forcing and interrupt enabling stage. This register does not exist when ICT_HAS_FIQ = 0. 1 active 0 inactive

### 6.3.2.14 FIQ\_FINALSTATUS

**Size:** 32 bits

**Offset:** 0xd4

**Memory Access:** R

31:7	6:0
(undef)	FIQ_FINALSTATUS

Name: Fast Interrupt Final Status Register Size: 1-8 bits Address Offset: 0xd4 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:7			Reserved for future use.
6:0	FIQ_FINALSTATUS	R	Fast interrupt status after the masking stage. This register does not exist when ICT_HAS_FIQ = 0. 1 active 0 inactive

### 6.3.2.15 IRQ\_PLEVEL

**Size:** 32 bits

**Offset:** 0xd8

**Memory Access:** R/W

**Value After Reset:** 0x0

31:4	3:0
(undef)	IRQ_PLEVEL

Name: IRQ System Priority Level Register Size: 4 bits Address Offset: 0xd8 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description

Bits	Name	Memory Access	Description
31:4			Reserved for future use.
3:0	IRQ_PLEVEL	R/W	Interrupt controller system priority level for normal interrupt sources. The default state can be configured so that after reset, the interrupt controller accepts only interrupts that are enabled and have a priority the same or greater than the system level priority setting.

**6.3.2.16 ICTL\_VERSION\_ID****Size:** 32 bits**Offset:** 0xe0**Memory Access:** R**Value After Reset:** 0x3230352a

31:0
ICTL_VERSION_ID

Name: Component Version Register Size: 32 bits Address Offset: 0xe0 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:0	ICTL_VERSION_ID	R	Specific values of ICTL_VERSION_ID

## 7 DDR2 Memory

### 7.1 Overview

### 7.2 DDR2 Function Interface

### 7.3 Architecture and Function Description

### 7.4 DDR2 Programming

#### 7.4.1 DDR2 Register Manual

##### 7.4.1.1 DENALI\_CTL\_00 (Controller Address 0x00, AHB Address 0x00)

Bits	Name	Default	Range	Description
31:16	VERSION	0x2041	0x2041	Holds the controller version number. READ-ONLY
11:8	DRAM_CLASS	0x0	0x0-0xf	Defines the mode of operation of the controller.
0	START	0x0	0x0-0x1	Initiate command processing in the controller. Set to 1 to initiate.

##### 7.4.1.2 DENALI\_CTL\_01 (Controller Address 0x01, AHB Address 0x04)

Bits	Name	Default	Range	Description
31:24	READ_DATA_FIFO_DEPTH	0x00	0x0-0xff	Reports the depth of the controller core read data queue. READ-ONLY
16	MAX_CS_REG	0x1	0x0-0x1	Holds the maximum number of chip selects available. READ-ONLY
11:8	MAX_COL_REG	0xb	0x0-0xb	Holds the maximum width of column address in DRAMs. READ-ONLY
4:0	MAX_ROW_REG	0x10	0x0-0x10	Holds the maximum width of memory address bus. READ-ONLY

##### 7.4.1.3 DENALI\_CTL\_02 (Controller Address 0x02, AHB Address 0x08)

Bits	Name	Default	Range	Description
31:24	ASYNC_CDC_STAGES	0x00	0x0-0xff	Reports the number of synchronizer delays specified for the asynchronous boundary crossings. READ-ONLY
23:16	WRITE_DATA_FIFO_PTR_WIDTH	0x00	0x0-0xff	Reports the width of the controller core write data latency queue pointer. READ-ONLY
15:8	WRITE_DATA_FIFO_DEPTH	0x00	0x0-0xff	Reports the depth of the controller core write data latency queue. READ-ONLY
7:0	READ_DATA_FIFO_PTR_WIDTH	0x00	0x0-0xff	Reports the width of the controller core read data queue pointer. READ-ONLY

##### 7.4.1.4 DENALI\_CTL\_03 (Controller Address 0x03, AHB Address 0x0c)

Bits	Name	Default	Range	Description
27:24	INITAREF	0x0	0x0-0xf	Number of auto-refresh commands to execute during DRAM initialization.
23:0	TINIT	0x000000	0x0-0xffffffff	DRAM TINIT value in cycles.

##### 7.4.1.5 DENALI\_CTL\_04 (Controller Address 0x04, AHB Address 0x10)

Bits	Name	Default	Range	Description
31:16	TDLL	0x0000	0x0-0xffff	DRAM TDLL value in cycles.
15:0	TCPD	0x0000	0x0-0xffff	DRAM TCPD value in cycles.

**7.4.1.6 DENALI\_CTL\_05 (Controller Address 0x05, AHB Address 0x14)**

Bits	Name	Default	Range	Description
26:24	ADDITIVE_LAT	0x0	0x0-0x7	DRAM additive latency value in cycles.
19:16	WRLAT	0x0	0x0-0xf	DRAM WRLAT value in cycles.
11:8	CASLAT_LIN	0x0	0x0-0xf	Sets latency from read command send to data receive from/to controller. Bit (0) is half-cycle increment and the upper bits define memory CAS latency for the controller.
0	NO_CMD_INIT	0x0	0x0-0x1	Disable DRAM commands until the TDLL parameter has expired during initialization. Set to 1 to disable.

**7.4.1.7 DENALI\_CTL\_06 (Controller Address 0x06, AHB Address 0x18)**

Bits	Name	Default	Range	Description
31:24	TRC	0x00	0x0-0xff	DRAM TRC value in cycles.
23:16	TRRD	0x00	0x0-0xff	DRAM TRRD value in cycles.
12:8	TCCD	0x00	0x1-0x1f	DRAM CAS-to-CAS value in cycles.
2:0	TBST_INT_INTERVAL	0x00	0x1-0x7	DRAM burst interrupt interval value in cycles.

**7.4.1.8 DENALI\_CTL\_07 (Controller Address 0x07, AHB Address 0x1c)**

Bits	Name	Default	Range	Description
29:24	TFAW	0x00	0x0-0x3f	DRAM TFAW value in cycles.
20:16	TRP	0x00	0x0-0x1f	DRAM TRP value in cycles.
11:8	TWTR	0x0	0x0-0xf	DRAM TWTR value in cycles.
7:0	TRAS_MIN	0x00	0x0-0xff	DRAM TRAS_MIN value in cycles.

**7.4.1.9 DENALI\_CTL\_08 (Controller Address 0x08, AHB Address 0x20)**

Bits	Name	Default	Range	Description
23:16	TMOD	0x00	0x0-0xff	Number of cycles after MRS command and before any other command.
12:8	TMRD	0x00	0x0-0x1f	DRAM TMRD value in cycles.
2:0	TRTP	0x0	0x0-0x7	DRAM TRTP value in cycles.

**7.4.1.10 DENALI\_CTL\_09 (Controller Address 0x09, AHB Address 0x24)**

Bits	Name	Default	Range	Description
31:24	TCKESR	0x00	0x0-0xff	Minimum CKE low pulse width during a self-refresh.
18:16	TCKE	0x0	0x0-0x7	Minimum CKE pulse width.
15:0	TRAS_MAX	0x0000	0x0-0xffff	DRAM TRAS_MAX value in cycles.

**7.4.1.11 DENALI\_CTL\_10 (Controller Address 0x0a, AHB Address 0x28)**

Bits	Name	Default	Range	Description
24	AP	0x0	0x0-0x1	Enable auto pre-charge mode of controller. Set to 1 to enable.
21:16	TWR	0x00	0x0-0x3f	DRAM TWR value in cycles.
15:8	TRCD	0x00	0x0-0xff	DRAM TRCD value in cycles.

Bits	Name	Default	Range	Description
0	WRITEINTERP	0x0	0x0-0x1	Allow controller to interrupt a write burst to the DRAMs with a read command. Set to 1 to allow interruption.

**7.4.1.12 DENALI\_CTL\_11 (Controller Address 0x0b, AHB Address 0x2c)**

Bits	Name	Default	Range	Description
26:24	BSTLEN	0x2	0x0-0x7	Encoded burst length sent to DRAMs during initialization. Set to 1 for BL2, set to 2 for BL4, or set to 3 for BL8.
21:16	TDAL	0x00	0x0-0x3f	DRAM TDAL value in cycles.
8	TRAS_LOCKOUT	0x0	0x0-0x1	Allow the controller to execute auto pre-charge commands before the TRAS_MIN parameter expires. Set to 1 to enable.
0	CONCURRENTAP	0x0	0x0-0x1	Allow controller to issue commands to other banks while a bank is in auto pre-charge. Set to 1 to enable.

**7.4.1.13 DENALI\_CTL\_12 (Controller Address 0x0c, AHB Address 0x30)**

Bits	Name	Default	Range	Description
24	RESERVED	0x0	0x0-0x1	Reserved for future use. Must be cleared to 0x0.
16	AREFRESH	0x0	0x0-0x1	Initiate auto-refresh at the end of the current burst boundary. Set to 1 to trigger. WRITE-ONLY
8	REG_DIMM_ENABLE	0x0	0x0-0x1	Enable registered DIMM operation of the controller. Set to 1 to enable.
4:0	TRP_AB	0x0	0x0-0x1f	DRAM TRP all bank value in cycles.

**7.4.1.14 DENALI\_CTL\_13 (Controller Address 0x0d, AHB Address 0x34)**

Bits	Name	Default	Range	Description
17:8	TRFC	0x000	0x0-0x3ff	DRAM TRFC value in cycles.
0	TREF_ENABLE	0x0	0x0-0x1	Issue auto-refresh commands to the DRAMs at the interval defined in the TREF parameter. Set to 1 to enable.

**7.4.1.15 DENALI\_CTL\_14 (Controller Address 0x0e, AHB Address 0x38)**

Bits	Name	Default	Range	Description
24	POWER_DOWN	0x0	0x0-0x1	Disable clock enable and set DRAMs in power-down state. Set to 1 to disable.
13:0	TREF	0x0000	0x0-0x3fff	DRAM TREF value in cycles.

**7.4.1.16 DENALI\_CTL\_15 (Controller Address 0x0f, AHB Address 0x3c)**

Bits	Name	Default	Range	Description
31:16	TXSR	0x0000	0x0-0xffff	DRAM TXSR value in cycles.
15:0	TPDEX	0x0000	0x0-0xffff	DRAM TPDEX value in cycles.

**7.4.1.17 DENALI\_CTL\_16 (Controller Address 0x10, AHB Address 0x40)**

Bits	Name	Default	Range	Description
24	PWRUP_SREFRESH_EXIT	0x0	0x0-0x1	Allow powerup via self-refresh instead of full memory initialization. Set to 1 to enable.

Bits	Name	Default	Range	Description
16	SREFRESH	0x0	0x0-0x1	Place DRAMs in self-refresh mode. Set to 1 to trigger entry.
15:0	TXSNR	0x0000	0x0-0xffff	DRAM TXSNR value in cycles.

**7.4.1.18 DENALI\_CTL\_17 (Controller Address 0x11, AHB Address 0x44)**

Bits	Name	Default	Range	Description
28:24	RESERVED	0x00	0x0-0x1f	Reserved for future use. Must be cleared to 0x0.
18:16	CKE_DELAY	0x0	0x0-0x7	Additional cycles to delay CKE for status reporting.
8	ENABLE_QUICK_SREFRESH	0x0	0x0-0x1	Allow user to interrupt memory initialization to enter self-refresh mode. Set to 1 to allow interruption.
0	SREFRESH_EXIT_NO_REFRESH	0x0	0x0-0x1	Disables the automatic refresh request associated with self-refresh exit. Set to 1 to disable.

**7.4.1.19 DENALI\_CTL\_18 (Controller Address 0x12, AHB Address 0x48)**

Bits	Name	Default	Range	Description
25:0	WRITE_MODEREG	0x0000000	0x0-0x3fffff	Write memory mode register data to the DRAMs. Bits (7:0) define the memory mode register number if bit (23) is set, bits (15:8) define the chip select if bit (24) is clear, bits (23:16) define which memory mode register/s to write, bit (24) defines whether all chip selects will be written, and bit (25) triggers the write.

**7.4.1.20 DENALI\_CTL\_19 (Controller Address 0x13, AHB Address 0x4c)**

Bits	Name	Default	Range	Description
23:8	MRO_DATA_0	0x0000	0x0-0xffff	Data to program into memory mode register 0 for chip select 0.
7:0	MRW_STATUS	0x00	0x0-0xff	Write memory mode register status. Bit (0) set indicates a WRITE_MODEREG parameter programming error. READ-ONLY

**7.4.1.21 DENALI\_CTL\_20 (Controller Address 0x14, AHB Address 0x50)**

Bits	Name	Default	Range	Description
31:16	MR2_DATA_0	0x0000	0x0-0xffff	Data to program into memory mode register 2 for chip select 0.
15:0	MR1_DATA_0	0x0000	0x0-0xffff	Data to program into memory mode register 1 for chip select 0.

**7.4.1.22 DENALI\_CTL\_21 (Controller Address 0x15, AHB Address 0x54)**

Bits	Name	Default	Range	Description
31:16	MR3_DATA_0	0x0000	0x0-0xffff	Data to program into memory mode register 3 for chip select 0.
15:0	MRSINGLE_DATA_0	0x0000	0x0-0xffff	Data to program into memory mode register single write to chip select 0.

**7.4.1.23 DENALI\_CTL\_22 (Controller Address 0x16, AHB Address 0x58)**

Bits	Name	Default	Range	Description
24	BIST_DATA_CHECK	0x0	0x0-0x1	Enable data checking with BIST operation. Set to 1 to enable.
20:16	ADDR_SPACE	0x00	0x0-0x1f	Sets the number of address bits to check during BIST operation.
9:8	BIST_RESULT	0x0	0x0-0x3	BIST operation status (pass/fail). Bit (0) indicates data check status and bit (1) indicates address check status. Value of 1 is a passing result. READ-ONLY
0	BIST_GO	0x0	0x0-0x1	Initiate a BIST operation. Set to 1 to trigger. WRITE-ONLY

**7.4.1.24 DENALI\_CTL\_23 (Controller Address 0x17, AHB Address 0x5c)**

Bits	Name	Default	Range	Description
0	BIST_ADDR_CHECK	0x0	0x0-0x1	Enable address checking with BIST operation. Set to 1 to enable.

**7.4.1.25 DENALI\_CTL\_24 (Controller Address 0x18, AHB Address 0x60)**

Bits	Name	Default	Range	Description
30:0	BIST_START_ADDRESS	0x000000 00	0x0-0xffffffff	Start BIST checking at this address.

**7.4.1.26 DENALI\_CTL\_25 (Controller Address 0x19, AHB Address 0x64)**

Bits	Name	Default	Range	Description
31:0	BIST_DATA_MASK	0x0000000	0x0-0xffffffff	Mask applied to data for BIST error checking. Bit (0) controls memory data path bit (0), bit (1) controls memory data path bit (1), etc. Set each bit to 1 to mask.

**7.4.1.27 DENALI\_CTL\_26 (Controller Address 0x1a, AHB Address 0x68)**

Bits	Name	Default	Range	Description
27:24	APREBIT	0xa	0x0-0xf	Location of the auto pre-charge bit in the DRAM address.
18:16	COL_DIFF	0x0	0x0-0x7	Difference between number of column pins available and number being used.
10:8	ROW_DIFF	0x0	0x0-0x7	Difference between number of address pins available and number being used.
1:0	BANK_DIFF	0x0	0x0-0x3	Encoded number of banks on the DRAM(s).

**7.4.1.28 DENALI\_CTL\_27 (Controller Address 0x1b, AHB Address 0x6c)**

Bits	Name	Default	Range	Description
24	ADDR_COLLISION_MPM_DIS	0x0	0x0-0x1	Disable address collision detection extension using micro page mask for command queue placement and selection. Set to 1 to disable.
16	ADDR_CMP_EN	0x0	0x0-0x1	Enable address collision detection as a rule for command queue placement. Set to 1 to enable.
15:8	COMMAND_AGE_COUNT	0x00	0x0-0xff	Initial value of individual command aging counters for command aging.

Bits	Name	Default	Range	Description
7:0	AGE_COUNT	0x00	0x0-0xff	Initial value of master aging-rate counter for command aging.

**7.4.1.29 DENALI\_CTL\_28 (Controller Address 0x1c, AHB Address 0x70)**

Bits	Name	Default	Range	Description
24	RW_SAME_EN	0x0	0x0-0x1	Enable read/write grouping as a rule for command queue placement. Set to 1 to enable.
16	PRIORITY_EN	0x0	0x0-0x1	Enable priority as a rule for command queue placement. Set to 1 to enable.
8	PLACEMENT_EN	0x0	0x0-0x1	Enable placement logic for command queue. Set to 1 to enable.
0	BANK_SPLIT_EN	0x0	0x0-0x1	Enable bank splitting as a rule for command queue placement. Set to 1 to enable.

**7.4.1.30 DENALI\_CTL\_29 (Controller Address 0x1d, AHB Address 0x74)**

Bits	Name	Default	Range	Description
24	SWAP_EN	0x0	0x0-0x1	Enable command swapping logic in execution unit. Set to 1 to enable.
19:16	NUM_Q_ENTRIES_ACT_DISABLE	0x0	0x0-0xf	Number of queue entries in which ACT requests will be disabled. Setting to X will disable ACT requests from the X entries lowest in the command queue.
9:8	DISABLE_RW_GROUP_W_BNK_CONFLICT	0x0	0x0-0x3	Disables placement to read/write group when grouping creates a bank collision. Bit (0) controls placement next to bank conflict command and bit (1) controls placement 2 away from bank conflict command. Set each bit to 1 to disable.
0	RW_SAME_PAGE_EN	0x0	0x0-0x1	Enable page grouping when read/ write grouping as a rule for command queue placement. This is only valid when the RW_SAME_EN parameter is set. Set to 1 to enable.

**7.4.1.31 DENALI\_CTL\_30 (Controller Address 0x1e, AHB Address 0x78)**

Bits	Name	Default	Range	Description
27:24	Q_FULLNESS	0x0	0x0-0xf	Quantity that determines command queue full.
16	BIG_ENDIAN_EN	0x0	0x0-0x1	Set byte ordering as little endian or big endian. Set to 1 for big endian.
8	REDUC	0x0	0x0-0x1	Enable the half datapath feature of the controller. Set to 1 to enable.
0	INHIBIT_DRAM_CMD	0x0	0x0-0x1	Inhibit read/write command traffic and associated bank commands. Set to 1 to inhibit.

**7.4.1.32 DENALI\_CTL\_31 (Controller Address 0x1f, AHB Address 0x7c)**

Bits	Name	Default	Range	Description
24	CTRLUPD_REQ_PER_AREF_EN	0x0	0x0-0x1	Enable an automatic controller-initiated update (dfi_ctrlupd_req) after every refresh. Set to 1 to enable.

Bits	Name	Default	Range	Description
16	CTRLUPD_REQ	0x0	0x0-0x1	Assert the DFI controller-initiated update request signal dfi_ctrlupd_req. Set to 1 to trigger. WRITE-ONLY
8	CONTROLLER_BUSY	0x0	0x0-0x1	Indicator that the controller is processing a command. Evaluates all ports for outstanding transactions. Value of 1 indicates controller busy. READ-ONLY
0	IN_ORDER_ACCEPT	0x0	0x0-0x1	Forces the controller to accept commands in the order in which they are placed in the command queue.

#### 7.4.1.33 DENALI\_CTL\_32 (Controller Address 0x20, AHB Address 0x80)

Bits	Name	Default	Range	Description
24:16	INT_ACK	0x000	0x0-0x1ff	Clear mask of the INT_STATUS parameter. WRITE-ONLY
9:0	INT_STATUS	0x000	0x0-0x3ff	Status of interrupt features in the controller. READ-ONLY

#### 7.4.1.34 DENALI\_CTL\_33 (Controller Address 0x21, AHB Address 0x84)

Bits	Name	Default	Range	Description
9:0	INT_MASK	0x000	0x0-0x3ff	Mask for controller_int signals from the INT_STATUS parameter.

#### 7.4.1.35 DENALI\_CTL\_34 (Controller Address 0x22, AHB Address 0x88)

Bits	Name	Default	Range	Description
30:0	OUT_OF_RANGE_ADDR	0x0000000	0x0-0x7fffffff	Address of command that caused an out-of-range interrupt. READ-ONLY

#### 7.4.1.36 DENALI\_CTL\_35 (Controller Address 0x23, AHB Address 0x8c)

Bits	Name	Default	Range	Description
26:24	OUT_OF_RANGE_SOURCE_ID	0x0	0x0-0x7	Source ID of command that caused an out-of-range interrupt. READ-ONLY
21:16	OUT_OF_RANGE_TYPE	0x00	0x0-0x3f	Type of command that caused an out-of-range interrupt. READ-ONLY
9:0	OUT_OF_RANGE_LENGTH	0x000	0x0-0x3ff	Length of command that caused an out-of-range interrupt. READ-ONLY

#### 7.4.1.37 DENALI\_CTL\_36 (Controller Address 0x24, AHB Address 0x90)

Bits	Name	Default	Range	Description
31:0	BIST_EXP_DATA [31:0]	0x0000000	0x0-0xffffffff	Expected data on BIST error. READ-ONLY

#### 7.4.1.38 DENALI\_CTL\_37 (Controller Address 0x25, AHB Address 0x94)

Bits	Name	Default	Range	Description
31:0	BIST_EXP_DATA [63:32]	0x0000000	0x0-0xffffffff	Expected data on BIST error. READ-ONLY

**7.4.1.39 DENALI\_CTL\_38 (Controller Address 0x26, AHB Address 0x98)**

Bits	Name	Default	Range	Description
31:0	BIST_FAIL_DATA [31:0]	0x000000	0x0-0xffffffff	Actual data on BIST error. READ-ONLY

**7.4.1.40 DENALI\_CTL\_39 (Controller Address 0x27, AHB Address 0x9c)**

Bits	Name	Default	Range	Description
31:0	BIST_FAIL_DATA [63:32]	0x000000	0x0-0xffffffff	Actual data on BIST error. READ-ONLY

**7.4.1.41 DENALI\_CTL\_40 (Controller Address 0x28, AHB Address 0xa0)**

Bits	Name	Default	Range	Description
30:0	BIST_FAIL_ADDR	0x000000	0x0-0x7fffffff	Address of BIST error. READ-ONLY

**7.4.1.42 DENALI\_CTL\_41 (Controller Address 0x29, AHB Address 0xa4)**

Bits	Name	Default	Range	Description
27:24	WR_TO_ODTH	0x0	0x0-0xf	Defines the delay from a write command to ODT assertion.
16	ODT_EN	0x0	0x0-0x1	Enable support of DRAM ODT. When enabled, controller will assert and de-assert ODT output to DRAM as needed.
11:8	TODTH_WR	0x0	0x0-0xf	Defines the DRAM minimum ODT high time after an ODT assertion for a write command.
3:0	TODTL_2CMD	0x0	0x0-0xf	Defines the DRAM delay from an ODT de-assertion to the next non-write, non-read command.

**7.4.1.43 DENALI\_CTL\_42 (Controller Address 0x2a, AHB Address 0xa8)**

Bits	Name	Default	Range	Description
26:24	R2W_SAMECS_DLY	0x2	0x0-0x7	Additional delay to insert between reads and writes to the same chip select. Program to a non-zero value.
18:16	R2R_SAMECS_DLY	0x0	0x0-0x7	Additional delay to insert between two reads to the same chip select. Any value including 0x0 supported.
11:8	ADD_ODT_CLK_W2R_SAMECS	0x0	0x0-0xf	Additional delay to insert between write and read transaction types to the same chip select to meet ODT timing requirements. Any value including 0x0 supported.
3:0	ADD_ODT_CLK_R2W_SAMECS	0x0	0x0-0xf	Additional delay to insert between read and write transaction types to the same chip select to meet ODT timing requirements.

**7.4.1.44 DENALI\_CTL\_43 (Controller Address 0x2b, AHB Address 0xac)**

Bits	Name	Default	Range	Description
28:24	OCD_ADJUST_PUP_CS_0	0x00	0x0-0x1f	OCD pull-up adjust setting for DRAMs for chip select 0.
20:16	OCD_ADJUST_PDN_CS_0	0x00	0x0-0x1f	OCD pull-down adjust setting for DRAMs for chip select 0.

Bits	Name	Default	Range	Description
10:8	W2W_SAMECS_DLY	0x0	0x0-0x7	Additional delay to insert between two writes to the same chip select. Any value including 0x0 supported.
2:0	W2R_SAMECS_DLY	0x0	0x0-0x7	Additional delay to insert between writes and reads to the same chip select.

**7.4.1.45 DENALI\_CTL\_44 (Controller Address 0x2c, AHB Address 0xb0)**

Bits	Name	Default	Range	Description
26:16	AHBO_RDCNT	0x000	0x0-0x7ff	Number of bytes for an INCR read command on AHB port 0.
10:0	AHBO_WRCNT	0x000	0x0-0x7ff	Number of bytes for an INCR write command on AHB port 0.

**7.4.1.46 DENALI\_CTL\_45 (Controller Address 0x2d, AHB Address 0xb4)**

Bits	Name	Default	Range	Description
17:16	AHBO_FIFO_TYPE_REG	0x0	0x0-0x3	Clock domain relativity between AHB port 0 and the controller core. Set to 0 for asynchronous, set to 1 for 2:1 port:core pseudo-sync, set to 2 for 1:2 port:core pseudo-sync, or set to 3 for synchronous.
10:8	AHBO_W_PRIORITY	0x0	0x0-0x7	Priority of write commands from AHB port 0. 0 is the highest priority.
2:0	AHBO_R_PRIORITY	0x0	0x0-0x7	Priority of read commands from AHB port 0. 0 is the highest priority.

**7.4.1.47 DENALI\_CTL\_46 (Controller Address 0x2e, AHB Address 0xb8)**

Bits	Name	Default	Range	Description
26:16	AHB1_RDCNT	0x000	0x0-0x7ff	Number of bytes for an INCR read command on AHB port 1.
10:0	AHB1_WRCNT	0x000	0x0-0x7ff	Number of bytes for an INCR write command on AHB port 1.

**7.4.1.48 DENALI\_CTL\_47 (Controller Address 0x2f, AHB Address 0xbc)**

Bits	Name	Default	Range	Description
17:16	AHB1_FIFO_TYPE_REG	0x0	0x0-0x3	Clock domain relativity between AHB port 0 and the controller core. Set to 0 for asynchronous, set to 1 for 2:1 port:core pseudo-sync, set to 2 for 1:2 port:core pseudo-sync, or set to 3 for synchronous.
10:8	AHB1_W_PRIORITY	0x0	0x0-0x7	Priority of write commands from AHB port 0. 0 is the highest priority.
2:0	AHB1_R_PRIORITY	0x0	0x0-0x7	Priority of read commands from AHB port 0. 0 is the highest priority.

**7.4.1.49 DENALI\_CTL\_48 (Controller Address 0x30, AHB Address 0xc0)**

Bits	Name	Default	Range	Description
26:16	AHB2_RDCNT	0x000	0x0-0x7ff	Number of bytes for an INCR read command on AHB port 2.

Bits	Name	Default	Range	Description
10:0	AHB2_WRCNT	0x000	0x0-0x7ff	Number of bytes for an INCR write command on AHB port 2.

**7.4.1.50 DENALI\_CTL\_49 (Controller Address 0x31, AHB Address 0xc4)**

Bits	Name	Default	Range	Description
17:16	AHB2_FIFO_TYPE_REG	0x0	0x0-0x3	Clock domain relativity between AHB port 0 and the controller core. Set to 0 for asynchronous, set to 1 for 2:1 port:core pseudo-sync, set to 2 for 1:2 port:core pseudo-sync, or set to 3 for synchronous.
10:8	AHB2_W_PRIORITY	0x0	0x0-0x7	Priority of write commands from AHB port 0. 0 is the highest priority.
2:0	AHB2_R_PRIORITY	0x0	0x0-0x7	Priority of read commands from AHB port 0. 0 is the highest priority.

**7.4.1.51 DENALI\_CTL\_50 (Controller Address 0x32, AHB Address 0xc8)**

Bits	Name	Default	Range	Description
26:16	AHB3_RDCNT	0x000	0x0-0x7ff	Number of bytes for an INCR read command on AHB port 3.
10:0	AHB3_WRCNT	0x000	0x0-0x7ff	Number of bytes for an INCR write command on AHB port 3.

**7.4.1.52 DENALI\_CTL\_51 (Controller Address 0x33, AHB Address 0xcc)**

Bits	Name	Default	Range	Description
17:16	AHB3_FIFO_TYPE_REG	0x0	0x0-0x3	Clock domain relativity between AHB port 0 and the controller core. Set to 0 for asynchronous, set to 1 for 2:1 port:core pseudo-sync, set to 2 for 1:2 port:core pseudo-sync, or set to 3 for synchronous.
10:8	AHB3_W_PRIORITY	0x0	0x0-0x7	Priority of write commands from AHB port 0. 0 is the highest priority.
2:0	AHB3_R_PRIORITY	0x0	0x0-0x7	Priority of read commands from AHB port 0. 0 is the highest priority.

**7.4.1.53 DENALI\_CTL\_52 (Controller Address 0x34, AHB Address 0xd0)**

Bits	Name	Default	Range	Description
26:16	AHB4_RDCNT	0x000	0x0-0x7ff	Number of bytes for an INCR read command on AHB port 4.
10:0	AHB4_WRCNT	0x000	0x0-0x7ff	Number of bytes for an INCR write command on AHB port 4.

**7.4.1.54 DENALI\_CTL\_53 (Controller Address 0x35, AHB Address 0xd4)**

Bits	Name	Default	Range	Description
17:16	AHB4_FIFO_TYPE_REG	0x0	0x0-0x3	Clock domain relativity between AHB port 0 and the controller core. Set to 0 for asynchronous, set to 1 for 2:1 port:core pseudo-sync, set to 2 for 1:2 port:core pseudo-sync, or set to 3 for synchronous.

Bits	Name	Default	Range	Description
10:8	AHB4_W_PRIORITY	0x0	0x0-0x7	Priority of write commands from AHB port 0. 0 is the highest priority.
2:0	AHB4_R_PRIORITY	0x0	0x0-0x7	Priority of read commands from AHB port 0. 0 is the highest priority.

**7.4.1.55 DENALI\_CTL\_58 (Controller Address 0x3a, AHB Address 0xe8)**

Bits	Name	Default	Range	Description
30:24	AHB1_BDW	0x00	0x0-0x64	Maximum bandwidth percentage for port 1.
22:16	AHBO_CURRENT_BDW	0x00	0x0-0x64	Current bandwidth usage percentage for port 0. READ-ONLY
8	AHBO_BDW_OVERFLOW	0x0	0x0-0x1	Port 0 behavior when bandwidth maximized. Set to 1 to allow overflow.
6:0	AHBO_BDW	0x00	0x0-0x64	Maximum bandwidth percentage for port 0.

**7.4.1.56 DENALI\_CTL\_59 (Controller Address 0x3b, AHB Address 0xec)**

Bits	Name	Default	Range	Description
24	AHB2_BDW_OVERFLOW	0x0	0x0-0x1	Port 2 behavior when bandwidth maximized. Set to 1 to allow overflow.
22:16	AHB2_BDW	0x00	0x0-0x64	Maximum bandwidth percentage for port 2.
14:8	AHB1_CURRENT_BDW	0x00	0x0-0x64	Current bandwidth usage percentage for port 1. READ-ONLY
0	AHB1_BDW_OVERFLOW	0x0	0x0-0x1	Port 1 behavior when bandwidth maximized. Set to 1 to allow overflow.

**7.4.1.57 DENALI\_CTL\_60 (Controller Address 0x3c, AHB Address 0xf0)**

Bits	Name	Default	Range	Description
30:24	AHB3_CURRENT_BDW	0x00	0x0-0x64	Current bandwidth usage percentage for port 3. READ-ONLY
16	AHB3_BDW_OVERFLOW	0x00	0x0-0x1	Port 3 behavior when bandwidth maximized. Set to 1 to allow overflow.
14:8	AHB3_BDW	0x0	0x0-0x64	Maximum bandwidth percentage for port 3.
6:0	AHB2_CURRENT_BDW	0x00	0x0-0x64	Current bandwidth usage percentage for port 2. READ-ONLY

**7.4.1.58 DENALI\_CTL\_61 (Controller Address 0x3c, AHB Address 0xf0)**

Bits	Name	Default	Range	Description
22:16	AHB4_CURRENT_BDW	0x00	0x0-0x64	Current bandwidth usage percentage for port 4. READ-ONLY
8	AHB4_BDW_OVERFLOW	0x0	0x0-0x1	Port 4 behavior when bandwidth maximized. Set to 1 to allow overflow.
6:0	AHB4_BDW	0x00	0x0-0x64	Maximum bandwidth percentage for port 4.

**7.4.1.59 DENALI\_CTL\_63 (Controller Address 0x3f, AHB Address 0xfc)**

Bits	Name	Default	Range	Description
31:16	DLL_RST_DELAY	0x0000	0x0-0xffff	Minimum cycles required for DLL reset signal dll_rst_n to be held. If this signal is not being used by the PHY, this parameter may be ignored.

Bits	Name	Default	Range	Description
8	CKE_STATUS	0x0	0x0-0x1	Register access to cke_status signal. READ-ONLY

#### 7.4.1.60 DENALI\_CTL\_64 (Controller Address 0x40, AHB Address 0x100)

Bits	Name	Default	Range	Description
27:24	TDFI_PHY_RDLAT	0x6	0x0-0xf	Defines the DFI tPHY_RDLAT timing parameter (in DFI PHY clocks), the maximum cycles between a dfi_rddata_en assertion and a dfi_rddata_valid assertion.
22:16	UPDATE_ERROR_STATUS	0x00	0x0-0x7f	Identifies the source of any DFI MC-initiated or PHY-initiated update errors. Value of 1 indicates a timing violation of the associated timing parameter. READ-ONLY
11:8	TDFI_PHY_WRLAT	Calc Value	0x0-0xf	Holds the calculated DFI tPHY_WRLAT timing parameter (in DFI PHY clocks), the maximum cycles between a write command and a dfi_wrdata_en assertion. READ-ONLY
7:0	DLL_RST_ADJ_DLY	0x00	0x0-0xff	Minimum cycles after setting master delay in DLL until the DLL reset signal dll_rst_n may be asserted. If this signal is not being used by the PHY, this parameter may be ignored.

#### 7.4.1.61 DENALI\_CTL\_65 (Controller Address 0x41, AHB Address 0x104)

Bits	Name	Default	Range	Description
19:16	TDFI_CTRLUPD_MIN	0x4	0x0-0xf	Reports the DFI tCTRLUPD_MIN timing parameter (in DFI clocks), the minimum cycles that dfi_ctrlupd_req must be asserted. READ-ONLY
8	DRAM_CLK_DISABLE	0x0	0x0-0x1	Set value for the dfi_dram_clk_disable signal. Bit (0) controls cs0, bit (1) controls cs1, etc. Set each bit to 1 to disable.
3:0	TDFI_RDDATA_EN	Calc Value	0x0-0xf	Holds the calculated DFI tRDDATA_EN timing parameter (in DFI PHY clocks), the maximum cycles between a read command and a dfi_rddata_en assertion. READ-ONLY

#### 7.4.1.62 DENALI\_CTL\_66 (Controller Address 0x42, AHB Address 0x108)

Bits	Name	Default	Range	Description
31:16	TDFI_PHYUPD_TYPE0	0x0000	0x0-0xffff	Defines the DFI tPHYUPD_TYPE0 timing parameter (in DFI clocks), the maximum cycles that dfi_phyupd_req can assert after dfi_phyupd_ack for dfi_phyupd_type 0. If programmed to a non-zero, a timing violation will cause an interrupt and bit (2) set in the UPDATE_ERROR_STATUS parameter.

Bits	Name	Default	Range	Description
13:0	TDFI_CTRLUPD_MAX	0x0000	0x0-0x3fff	Defines the DFI tCTRLUPD_MAX timing parameter (in DFI clocks), the maximum cycles that dfi_ctrlupd_req can be asserted. If programmed to a non-zero, a timing violation will cause an interrupt and bit (1) set in the UPDATE_ERROR_STATUS parameter.

#### 7.4.1.63 DENALI\_CTL\_67 (Controller Address 0x43, AHB Address 0x10c)

Bits	Name	Default	Range	Description
31:16	TDFI_PHYUPD_TYPE2	0x0000	0x0-0xffff	Defines the DFI tPHYUPD_TYPE2 timing parameter (in DFI clocks), the maximum cycles that dfi_phyupd_req can assert after dfi_phyupd_ack for dfi_phyupd_type 2. If programmed to a non-zero, a timing violation will cause an interrupt and bit (4) set in the UPDATE_ERROR_STATUS parameter.
15:0	TDFI_PHYUPD_TYPE1	0x0000	0x0-0xffff	Defines the DFI tPHYUPD_TYPE1 timing parameter (in DFI clocks), the maximum cycles that dfi_phyupd_req can assert after dfi_phyupd_ack for dfi_phyupd_type 1. If programmed to a non-zero, a timing violation will cause an interrupt and bit (3) set in the UPDATE_ERROR_STATUS parameter.

#### 7.4.1.64 DENALI\_CTL\_68 (Controller Address 0x44, AHB Address 0x110)

Bits	Name	Default	Range	Description
29:16	TDFI_PHYUPD_RESP	0x0000	0x0-0x3fff	Defines the DFI tPHYUPD_RESP timing parameter (in DFI clocks), the maximum cycles between a dfi_phyupd_req assertion and a dfi_phyupd_ack assertion. If programmed to a non-zero, a timing violation will cause an interrupt and bit (6) set in the UPDATE_ERROR_STATUS parameter.
15:0	TDFI_PHYUPD_TYPE3	0x0000	0x0-0xffff	Defines the DFI tPHYUPD_TYPE3 timing parameter (in DFI clocks), the maximum cycles that dfi_phyupd_req can assert after dfi_phyupd_ack for dfi_phyupd_type 3. If programmed to a non-zero, a timing violation will cause an interrupt and bit (5) set in the UPDATE_ERROR_STATUS parameter.

#### 7.4.1.65 DENALI\_CTL\_69 (Controller Address 0x45, AHB Address 0x114)

Bits	Name	Default	Range	Description

<b>Bits</b>	<b>Name</b>	<b>Default</b>	<b>Range</b>	<b>Description</b>
31:0	TDFI_CTRLUPD_INTERVAL	0x000000	0x0-0xffffffff	Defines the DFI tCTRLUPD_INTERVAL timing parameter (in DFI clocks), the maximum cycles between dfi_ctrlupd_req assertions. If programmed to a non-zero, a timing violation will cause an interrupt and bit (0) set in the UPDATE_ERROR_STATUS parameter.

#### 7.4.1.66 DENALI\_CTL\_70 (Controller Address 0x46, AHB Address 0x118)

<b>Bits</b>	<b>Name</b>	<b>Default</b>	<b>Range</b>	<b>Description</b>
18:16	TDFI_PHY_WRDATA	0x0	0x0-0x7	Defines the DFI tPHY_WRDATA timing parameter (in DFI PHY clocks), the maximum cycles between a dfi_wrdata_en assertion and a dfi_wrdata signal.
11:8	WRLAT_ADJ	0x0	0x0-0xf	Adjustment value for PHY write timing.
3:0	RDLAT_ADJ	0x0	0x0-0xf	Adjustment value for PHY read timing.

## 8 HiNoC

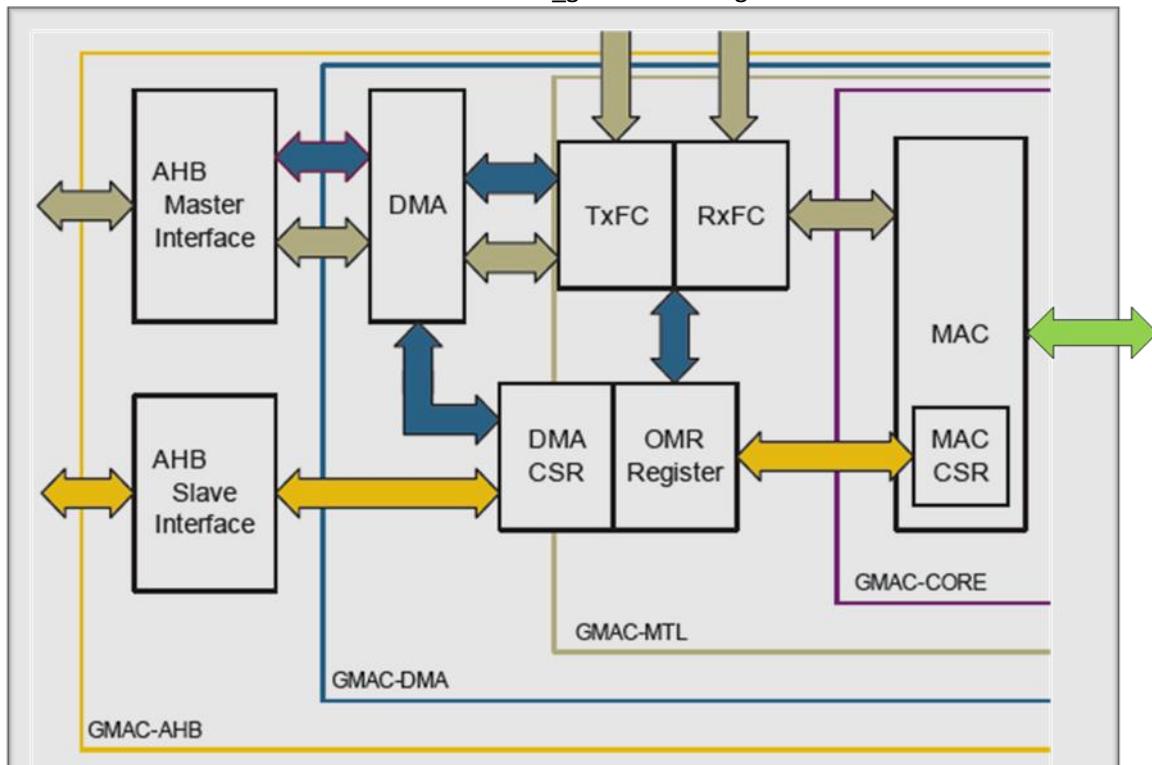
- 8.1 HiNoC Overview**
- 8.2 HiNoC Environment**
- 8.3 HiNoC Integration**
- 8.4 HiNoC Function Description and Data Transfer Bandwidth**
- 8.5 HiNoC Programming Model**
- 8.6 HiNoC Registers**

## 9 10/100 Ethernet Mac

### 9.1 10/100 Ethernet Mac Introduction

The DWC\_gmac enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3 standard and it supports the Media Independent Interface (MII). User can use the DWC\_gmac in number of applications such as switches and network interface cards.

Table 10-1 DWC\_gmac block diagram



#### Transmit and Receive FIFOs

The Transmit FIFO (Tx FIFO) buffers the data read, from the system memory, by the DMA before transmission by the MAC. Similarly, the Receive FIFO (Rx FIFO) stores the Ethernet frames received from the line until the DMA transfers them to system memory.

These are asynchronous FIFOs, as they also transfer the data between the application clock and the MAC line clocks.

#### MAC

The MAC supports the following features:

- ◆ 10 and 100Mbps data transfer rates with the PHY interfaces: IEEE 802.3-compliant GMII/MII interface to communicate with an external Gigabit/Fast Ethernet PHY
- ◆ Full-duplex operation:
  - IEEE 802.3x flow control automatic transmission of zero-quanta pause frame on flow control input de-assertion
  - Optional forwarding of received pause control frames to the user application
- ◆ Half-duplex operation:
  - CSMA/CD Protocol support
  - Flow control using back-pressure support

- Frame bursting and frame extension in 1000 Mbps half-duplex operation
- ◆ Preamble and start-of-frame data (SFD) insertion in Transmit path
- ◆ Preamble and SFD deletion in the Receive path
- ◆ Automatic CRC and pad generation controllable on a per-frame basis
- ◆ Automatic Pad and CRC Stripping options for receive frames
- ◆ Flexible address filtering modes, such as:
  - Up to 31 additional 48-bit perfect (DA) address filters with masks for each byte
  - Up to 96 additional 48-bit perfect (DA) address filters that can be selected in blocks of 32 and 64 registers
  - Up to 31 48-bit SA address comparison check with masks for each byte
  - Pass all incoming packets (as per filter) with a status report
- ◆ Programmable frame length to support Standard or Jumbo Ethernet frames with up to 16 KB of size
- ◆ Programmable Interframe Gap (IFG) (40-96 bit times in steps of 8)
- ◆ Option to transmit frames with reduced preamble size
- ◆ Separate 32-bit status for transmit and receive packets
- ◆ IEEE 802.1Q VLAN tag detection for reception frames
- ◆ Additional frame filtering:
  - VLAN tag-based: Perfect match and Hash-based (optional) filtering
  - Layer 3 and Layer 4-based: TCP or UDP over IPv4 or IPv6
- ◆ Separate transmission, reception, and control interfaces to the application
- ◆ **Configurable** big-endian and little-endian for Transmit and Receive paths
- ◆ 32-bit, 64-bit, or 128-bit data transfer interface on system-side
- ◆ Network statistics (optional) with RMON/MIB Counters (RFC2819/RFC2665)
- ◆ Optional module to detect LAN wake-up frames and AMD Magic Packet frames
- ◆ Optional Receive module for checksum off-load for received IPv4 and TCP packets encapsulated by the Ethernet frame (Type 1)
- ◆ Optional Enhanced Receive module for checking IPv4 header checksum and TCP, UDP, or ICMP checksum encapsulated in IPv4 or IPv6 datagrams (Type 2)
- ◆ ~~Optional module to support Ethernet frame timestamping as described in IEEE 1588-2002 and IEEE 1588-2008. The 64 bit timestamps are given in the transmit or receive status of each frame.~~
- ◆ MDIO Master interface (optional) for PHY device **configuration** and management
- ◆ ~~Standard IEEE P802.3az, version D3.2 for Energy Efficient Ethernet~~
- ◆ ~~Flexibility to control the Pulse Per Second (PPS) output signal (ptp\_pps\_o)~~
- ◆ CRC replacement, Source Address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted frames with per-frame control

### Transaction Layer (MTL)

The MTL block consists of two sets of FIFOs: a Transmit FIFO with programmable threshold capability and a Receive FIFO with a programmable threshold (64 bytes).

The MTL block supports the following features:

- ◆ 32-bit, 64-bit, or 128-bit Transaction Layer block (bridges the application and the GMAC-CORE)
- ◆ Single-channel Transmit and Receive engines
- ◆ Data transfers executed using simple FIFO-protocol
- ◆ Synchronization for all clocks in the design (Transmit, Receive, and system clocks)
- ◆ Optimization for packet-oriented transfers with frame delimiters
- ◆ Four separate ports for system-side and GMAC-CORE-side transmission and reception
- ◆ Two 2-port RAM-based asynchronous FIFOs with synchronous/asynchronous Read and Write operation with respect to the Read and Write clocks (one for transmission and one for reception)
- ◆ FIFO instantiation outside the top-level module to facilitate memory testing/instantiation
- ◆ 128-byte, 256-byte, or 512-byte, or 1-KB, 2-KB, 4-KB, 8-KB, 16-KB, or 32-KB receive FIFO sizes on reception
- ◆ Optional interface to indicate the length of a received frame at the top of the MTL Rx FIFO in the GMAC-MTL **configuration**
- ◆ Programmable burst-length for starting a burst up to half the size of the MTL Rx and Tx FIFO in the GMAC-MTL **configuration**
- ◆ Insertion of Receive Status vectors into the Receive FIFO after the EOF transfer This enables multiple-frame storage in the Receive FIFO without requiring another FIFO to store those frames' Receive Status.

- ◆ Configurable Receive FIFO threshold (default fixed at 64 bytes) in Cut-Through or Threshold mode
- ◆ Option to filter all error frames on reception and not forward them to the application in Store-and-Forward mode
- ◆ Option to forward under-sized good frames
- ◆ Supports statistics by generating pulses for frames dropped or corrupted (due to overflow) in the Receive FIFO
- ◆ 256-byte or 512-byte, or 1-KB, 2-KB, 4-KB, 8-KB, or 16-KB FIFO sizes on transmission
- ◆ Store and Forward mechanism for transmission to the MAC
- ◆ Threshold control for transmit buffer management
- ◆ Configurable number of frames to be stored in FIFO at any time. The default is two frames (fixed) with internal DMA, and up to eight frames in GMAC-MTL configuration
- ◆ Automatic generation of PAUSE frame control or backpressure signal to the MAC based on Receive FIFO-fill (threshold configurable) level
- ◆ Automatic retransmission of Collision frames for transmission
- ◆ Discard frames on late collision, excessive collisions, excessive deferral, and under-run conditions
- ◆ Software control to flush Tx FIFO
- ◆ Disabling of Data FIFO RAM chip-select when inactive to reduce power consumption
- ◆ Optional module to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum in frames transmitted in the store-and-forward mode

### DMA Block

The DMA block exchanges data between the MTL block and host memory. The host can use a set of registers (DMA CSR) to control the DMA operations.

The DMA block supports the following features:

- ◆ 32-bit, 64-bit, and 128-bit data transfers
- ◆ Single-channel Transmit and Receive engines
- ◆ Fully synchronous design operating on a single system clock (except for CSR module, when a separate CSR clock is configured)
- ◆ Optimization for packet-oriented DMA transfers with frame delimiters
- ◆ Byte-aligned addressing for data buffer support
- ◆ Dual-buffer (ring) or linked-list (chained) descriptor chaining
- ◆ Descriptor architecture to allow large blocks of data transfer with minimum CPU intervention (each descriptor can transfer up to 8 KB of data)
- ◆ Comprehensive status reporting for normal operation and transfers with errors
- ◆ Individual programmable burst size for Transmit and Receive DMA Engines for optimal host bus utilization
- ◆ Programmable interrupt options for different operational conditions
- ◆ Per-frame Transmit or Receive complete interrupt control
- ◆ Round-robin or fixed-priority arbitration between Receive and Transmit engines
- ◆ Start and Stop modes
- ◆ Separate ports for host CSR access and host data interface

### 9.2 DMA controller:

The DMA has independent Transmit and Receive engines, and a CSR space. The Transmit engine transfers data from system memory to the device port (MTL), while the Receive engine transfers data from the device port to the system memory. The controller uses descriptors to efficiently move data from source to destination with minimal Host CPU intervention. The DMA is designed for packet-oriented data transfers such as frames in Ethernet. The controller can be programmed to interrupt the Host CPU for situations such as Frame Transmit and Receive transfer completion, and other normal/error conditions.

The DMA and the Host driver communicate through two data structures:

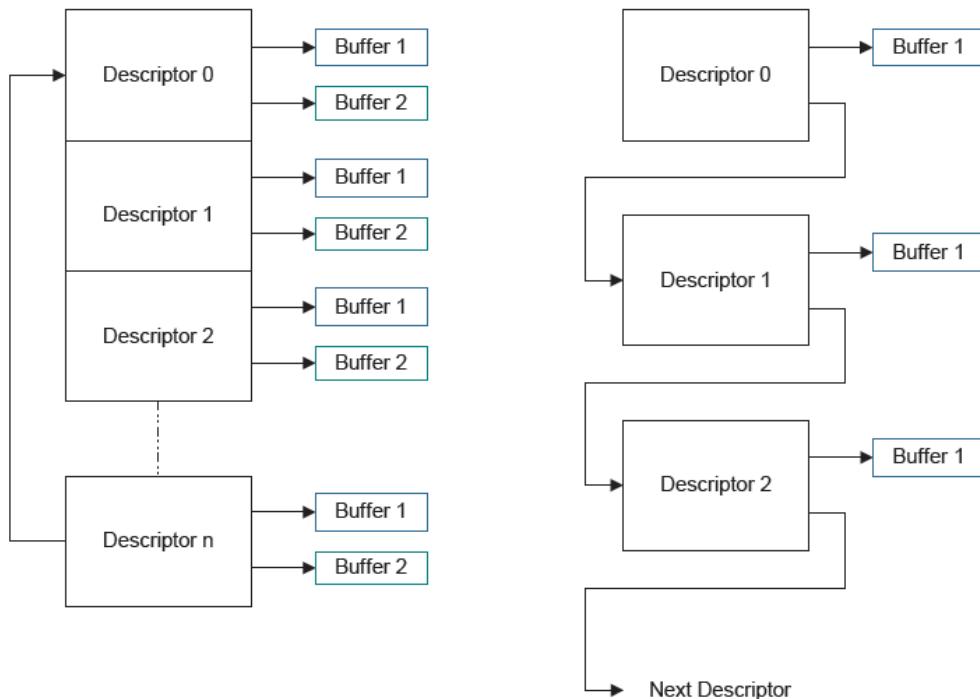
- ❖ Control and Status registers (CSR)
- ❖ Descriptor lists and data buffers

The DMA transfers data frames received by the MAC to the Receive Buffer in the Host memory, and Transmit data frames from the Transmit Buffer in the Host memory. Descriptors that reside in the Host memory act as pointers to these buffers.

There are two descriptor lists; one for reception, and one for transmission. The base address of each list is written into Register 3 (Receive Descriptor List Address Register) and Register 4 (Transmit Descriptor List Address Register), respectively. A descriptor list is forward linked (either implicitly or explicitly). The last descriptor may point back to the first entry to create a ring structure. Explicit chaining of descriptors is accomplished by setting the second address chained in both Receive and Transmit descriptors (RDES1[24] and TDES1[24]). The descriptor lists resides in the Host physical memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the Host physical memory space, and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers contain only data, buffer status is maintained in the descriptor. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA skips to the next frame buffer when end-of-frame is detected. Data chaining can be enabled or disabled.

Ring Structure	Chain Structure
----------------	-----------------



### 9.2.1 Initialization

Initialization for the DWC\_gmac is as follows.

1. Write to Register 0 (Bus Mode Register) to set Host bus access parameters.
  2. Write to Register 7 (Interrupt Enable Register) to mask unnecessary interrupt causes.
  3. Create the Transmit and Receive descriptor lists, and then write to DMA Register 3 (Receive Descriptor List Address Register) and Register 4 (Transmit Descriptor List Address Register), providing the DMA with the starting address of each list.
  4. Write to Register 1 (MAC Frame Filter), Register 2 (Hash Table High Register), and Register 3 (Hash Table Low Register) for desired filtering options.
  5. Write to Register 1 (MAC Frame Filter) to **configure** the operating mode and enable the transmit operation (Bit 3: Transmitter Enable). The PS and DM bits are set based on the auto-negotiation result (read from the PHY).
  6. Write to Register 6 (Operation Mode Register) to set Bits 13 and 1 to start transmission and reception.

- 
7. Write to Register 0 (MAC Configuration Register) to enable the Receive operation (Bit 2: Receiver Enable). The Transmit and Receive engines enter the Running state and attempt to acquire descriptors from the respective descriptor lists. The Receive and Transmit engines then begin processing Receive and Transmit operations. The Transmit and Receive processes are independent of each other and can be started or stopped separately.

### 9.2.2 Transmission

The Transmit DMA engine in default mode proceeds as follows:

1. The Host sets up the transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Frame data.
2. When Bit 13 (ST) of Register 6 (Operation Mode Register) is set, the DMA enters the Run state.
3. While in the Run state, the DMA polls the Transmit Descriptor list for frames requiring transmission. After polling starts, it continues in either sequential descriptor ring order or chained order. If the DMA detects a descriptor flagged as owned by the Host (TDES0[31] = 1'b0), or if an error condition occurs, transmission is suspended and both the Bit 2 (Transmit Buffer Unavailable) and Bit 16 (Normal Interrupt Summary) of the Register 5 (Status Register) are set. The Transmit Engine proceeds to Step 9.
4. If the acquired descriptor is flagged as owned by DMA (TDES0[31] = 1'b1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
5. The DMA fetches the Transmit data from the Host memory and transfers the data to the MTL for transmission.
6. If an Ethernet frame is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3, 4, and 5 are repeated until the end-of-Ethernet-frame data is transferred to the MTL.
7. When frame transmission is complete, if IEEE 1588 timestamping was enabled for the frame (as indicated in the transmit status) the timestamp value obtained from MTL is written to the transmit descriptor (TDES2 and TDES3) that contains the end-of-frame buffer. The status information is then written to this transmit descriptor (TDES0). Because the Own bit is cleared during this step, the Host now owns this descriptor. If timestamping was not enabled for this frame, the DMA does not alter the contents of TDES2 and TDES3.
8. Bit 0 (Transmit Interrupt) of Register 5 (Status Register) is set after completing transmission of a frame that has Interrupt on Completion (TDES1[31]) set in its Last Descriptor. The DMA engine then returns to Step 3.
9. In the Suspend state, the DMA tries to re-acquire the descriptor (and thereby return to Step 3) when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared.

### 9.2.3 Reception

The Receive DMA engine's reception sequence is depicted in Figure 3-23 and proceeds as follows:

1. The host sets up Receive descriptors (RDES0-RDES3) and sets the Own bit (RDES0[31]).
2. When Bit 1 (SR) of Register 6 (Operation Mode Register) is set, the DMA enters the Run state. While in the Run state, the DMA polls the Receive Descriptor list, attempting to acquire free descriptors. If the fetched descriptor is not free (is owned by the host), the DMA enters the Suspend state and jumps to Step 9.
3. The DMA decodes the receive data buffer address from the acquired descriptors.
4. Incoming frames are processed and placed in the acquired descriptor's data buffers.
5. When the buffer is full or the frame transfer is complete, the Receive engine fetches the next descriptor.
6. If the current frame transfer is complete, the DMA proceeds to Step 7. If the DMA does not own the next fetched descriptor and the frame transfer is not complete (EOF is not yet transferred), the DMA sets the Descriptor Error bit in the RDES0 (unless flushing is disabled in Bit 24 of Register 6 (Operation Mode Register)). The DMA closes the current descriptor (clears the Own bit) and marks it as intermediate by clearing the Last Segment (LS) bit in the RDES0 value (marks it as Last Descriptor if flushing is not disabled), then proceeds to Step 8. If the DMA does own the next descriptor but the current frame transfer is not complete, the DMA closes the current descriptor as intermediate and reverts to Step 4.
7. If IEEE 1588 timestamping is enabled, the DMA writes the timestamp (if available) to the current descriptor's RDES2 and RDES3. It then takes the receive frame's status from the MTL and writes the status word to the current descriptor's RDES0, with the Own bit cleared and the Last Segment bit set.
8. The Receive engine checks the latest descriptor's Own bit. If the host owns the descriptor (Own bit is 1'b0), the Bit 7 (Receive Buffer Unavailable) of Register 5 (Status Register) is set and the DMA Receive engine enters the

- Suspended state (Step 9). If the DMA owns the descriptor, the engine returns to Step 4 and awaits the next frame.
9. Before the Receive engine enters the Suspend state, partial frames are flushed from the Receive FIFO. You can control flushing using Bit 24 of Register 6 (Operation Mode Register).
  10. The Receive DMA exits the Suspend state when a Receive Poll demand is given or the start of next frame is available from the MTL's Receive FIFO. The engine proceeds to Step 2 and refetches the next descriptor.

#### 9.2.4 Interrupts

Interrupts can be generated as a result of various events. The DMA Register 5 (Status Register) contains all the bits that might cause an interrupt. Register 7 (Interrupt Enable Register) contains an enable bit for each of the events that can cause an interrupt.

There are two groups of interrupts, Normal and Abnormal, as described in Register 5 (Status Register). Interrupts are cleared by writing a 1'b1 to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both the summary bits are cleared, the interrupt signal `sbd_intr_o` is de-asserted. If the MAC is the cause for assertion of the interrupt, then any of the GLI, GMI, GPI, TTI, or GLPII bits of Register 5 (Status Register) are set high.

Interrupts are not queued and if the interrupt event occurs before the driver has responded to it, no additional interrupts are generated. For example, Receive Interrupt [Bit 6 of Register 5 (Status Register)] indicates that one or more frames were transferred to the Host buffer. The driver must scan all descriptors, from the last recorded position to the first one owned by the DMA.

An interrupt is generated only once for simultaneous, multiple events. The driver must scan the Register 5 (Status Register) for the cause of the interrupt. The interrupt is not generated again unless a new interrupting event occurs, after the driver has cleared the appropriate bit in Register 5 (Status Register). For example, the controller generates a Receive interrupt [Bit 6 of Register 5 (Status Register)] and the driver begins reading Register 5 (Status Register). Next, Receive Buffer Unavailable [Bit 7 of Register 5 (Status Register)] occurs. The driver clears the Receive interrupt. Even then, the `sbd_intr_o` signal is not de-asserted, because of the active or pending Receive Buffer Unavailable interrupt.

An interrupt timer RIWT (Bits 7:0 in Register 9 (Receive Interrupt Watchdog Timer Register)) is given for flexible control of Receive Interrupt. When this Interrupt timer is programmed with a non-zero value, it gets activated as soon as the Rx DMA completes a transfer of a received frame to system memory without asserting the Receive Interrupt because it is not enabled in the corresponding Receive Descriptor (RDES1[31] in Table 8-8). When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RI is enabled in Register 7 (Interrupt Enable Register). This timer gets disabled before it runs out, when a frame is transferred to memory and the RI is set because it is enabled for that descriptor.

#### 9.2.5 Err Response

For any data transfer initiated by a DMA channel, if the slave replies with an error response, that DMA stops all operations and updates the error bits and the Fatal Bus Error bit in the Register 5 (Status Register). The DMA controller can resume operation only after soft resetting or hard resetting the DWC\_gmac and re-initializing the DMA. This is applicable for all **configurations** with DMAs (GMAC-DMA, GMAC-AXI, and GMAC-AHB). In the GMAC-DMA **configuration**, the DMA receives the error response through `mdc_error_i` signal (shown in Figure 3-30). In the GMAC-AHB **configuration**, the error response is received through `hresp_i` (shown in Figure 3-13). In the GMAC-AXI **configuration**, the error response is received through `rresp_m_i` (shown in Figure 3-16) or `bresp_m_i` signal.

### 9.3 MAC Transaction Layer (MTL):

The MAC Transaction Layer provides FIFO memory to buffer and regulate the frames between the application system memory and the MAC. It also enables the data to be transferred between the application clock domain and the MAC clock domains. The MTL layer has 2 data paths, namely the Transmit path and the Receive Path. The data path for both directions is 32-bit, 64-bit, or 128-bit wide and operates with a simple FIFO protocol. The GMAC-MTL communicates with

---

the application side with the Application Transmit Interface (ATI), Application Receive Interface (ARI), and the MAC Control Interface (MCI).

### 9.3.1 Transmit Path

the DMA controls all transactions for the transmit path through the ATI. The Ethernet frames read from the system memory are pushed into the FIFO by the DMA. The frame is then popped out and transferred to the MAC when triggered. When the end-of-frame is transferred, the status of the transmission is taken from the MAC and transferred back to the DMA.

The Transmit FIFO has a default depth of 2K bytes. FIFO-fill level is indicated to the DMA so that it can initiate a data fetch in required bursts from the system memory, using the AHB or AXI interface. The data from the AHB or AXI Master interface is pushed into the FIFO with the appropriate byte lanes qualified by the DMA. The DMA also indicates the start-of-frame (SOF) and end-of-frame (EOF) transfers along with a few sideband signals controlling the pad-insertion or CRC generation for that frame in the MAC.

The per-frame control bits, such as Automatic Pad or CRC Stripping disable, timestamp capture, and so forth are taken as sideband control inputs on the ATI, stored in a separate register FIFO, and passed on to the MAC transmitter when the corresponding frame data is read from the Transmit FIFO.

There are two modes of operation for popping data towards the MAC. In Threshold mode, as soon as the number of bytes in the FIFO crosses the **configured** threshold level (or when the end-of-frame is written before the threshold is crossed), the data is ready to be popped out and forwarded to the MAC. The threshold level is **configured** using the TTC bits of Register 0 (Bus Mode Register). In store-and-forward mode, the MTL pops the frame towards the MAC only when one or more of the following conditions are true:

- ◆ When a complete frame is stored in the FIFO
- ◆ When the TX FIFO becomes almost full
- ◆ When the ATI watermark becomes low. The watermark becomes low when the requested FIFO does not have space to accommodate the requested burst-length on the ATI.

Therefore, the MTL never stops in the store-and-forward mode even if the Ethernet frame length is bigger than the Tx FIFO size.

The application can flush the Transmit FIFO of all contents by setting the Bit 20 (FTF) of Register 6 (Operation Mode Register). This bit is self-clearing and initializes the FIFO pointers to the default state. If the FTF bit is set during a frame transfer from the MTL to the MAC, then the MTL stops further transfer as the FIFO is considered to be empty. Hence an underflow event occurs at the MAC transmitter and the corresponding Status word is forwarded to the DMA.

### 9.3.2 Receive Path

This module receives the frames given out by the MAC and pushes them into the Rx FIFO. The status (fill level) of this FIFO is indicated to the DMA once it crosses the **configured** Receive threshold (Bits [4:3] of Register 6 (Operation Mode Register)). The MTL also indicates the FIFO fill level so that the DMA can initiate pre-**configured** burst transfers towards the AHB interface.

During an Rx operation, the MTL is slaved to the MAC. The general sequence of Receive operation events is as follows:

1. When the MAC receives a frame, it pushes in data along with byte enables. The MAC also indicates the SOF and EOF. The MTL accepts the data and pushes it into the Rx FIFO. After the EOF is transferred, the MAC drives the status word, which is also pushed into the same Rx FIFO by the MTL.
2. When IEEE 1588 timestamping is enabled and the 64-bit timestamp is available along with the receive status, it is appended to the frame received from the MAC and is pushed into the Rx FIFO before the corresponding receive status word is written. Thus, in 32-bit data bus mode, two additional locations per frame are taken for storing the timestamp in the Rx FIFO, while in 64-/128- bit mode, one additional location is taken.
3. The MTL\_RX engine takes the data out of the FIFO and sends it to the DMA. In the default Cut- Through mode, when 64 bytes (**configured** with Bits [4:3] (RTC) of Register 6 (Operation Mode Register)) or a full packet of data are received into the FIFO, the MTL\_RX engine pops out the data and indicates its availability to the DMA. Once

the DMA initiates the transfer to the AHB or AXI interface, the MTL\_RX engine continues to transfer data from the FIFO until a complete packet has been transferred. Upon completion of the EOF frame transfer, the MTL pops out the status word and sends it to the DMA controller. If the 64-bit timestamp is read out before the receive status, it is marked by the assertion of the `ari_timestamp_val_o` signal on the ARI. Otherwise, the status is given after the EOF data.

4. In the Rx FIFO store-and-forward mode (**configured** by the RSF bit of Register 6 (Operation Mode Register)), a frame is read out only after being written completely into the Receive FIFO. In this mode, all error frames are dropped (if the MAC is **configured** to do so) such that only valid frames are read out and forwarded to the application. In the Cut-Through mode, some error frames are not dropped, because the error status is received at the end-of-frame, by which time the start of that frame has already been read out of the FIFO.

## 9.4 MAC:

The MAC supports many interfaces towards the PHY chip. The PHY interface can be selected only once after reset. The MAC communicates with the application side with the MAC Transmit Interface (MTI), MAC Receive Interface (MRI) and the MAC Control Interface (MCI).

### 9.4.1 Transmission Path

Transmission is initiated when the MTL Application pushes in data with the SOF (`mti_sof_i`) signal asserted. When the SOF signal is detected, the MAC accepts the data and begins transmitting to the GMII or MII. The time required to transmit the frame data to the GMII or MII after the Application initiates transmission varies, depending on delay factors like IFG delay, time to transmit preamble or SFD, and any back-off delays for half-duplex mode. Until then, the MAC does not accept the data received from MTL by de-asserting the `mti_rdy_o` signal.

After the EOF (`mti_eof_i`) is transferred to the MAC, the MAC completes the normal transmission and gives the Status of Transmission to the MTL. If a normal collision (in half-duplex mode) occurs during transmission, the MAC makes valid the Transmit Status to the MTL. It then accepts and drops all further data until the next SOF is received. The MTL block should retransmit the same frame from SOF on observing a Retry request (in the Status) from the MAC.

The MAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. During the normal transfer of a frame from MTL, if the MAC receives a SOF without getting an EOF for the previous frame, then it ignores the SOF and considers the new frame as continuation of the previous frame.

### 9.4.2 Reception

A receive operation is initiated when the MAC detects an SFD on the GMII or MII. The MAC strips the preamble and SFD before proceeding to process the frame. The header fields are checked for the filtering and the FCS field used to verify the CRC for the frame. The received frame is stored in a shallow buffer until the address filtering is performed. The frame is dropped in the MAC if it fails the address filter.

## 9.5 MAC Management Counter

The counters in the MAC Management Counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted frames. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the MAC Control Interface (MCI). Each register is 32-bits wide. The write data is qualified with the corresponding `mci_be_i` signals. Therefore, non-32-bit accesses are allowed as long as the address is word-aligned. The MMCs are accessed using transactions, in the same way the CSR address space is accessed.

### 9.5.1 PMT Register Map

GMAC CSR Register No.	Address Offset	Register Name	Register Description
64	0x0100	mmc_cntrl	MMC Control establishes the operating mode of MMC.
65	0x0104	mmc_intr_rx	MMC Receive Interrupt maintains the interrupt generated from all of the receive statistic counters.
66	0x0108	mmc_intr_tx	MMC Transmit Interrupt maintains the interrupt generated from all of the transmit statistic counters.
67	0x010C	mmc_intr_mask_rx	MMC Receive Interrupt mask maintains the mask for the interrupt generated from all of the receive statistic counters.
68	0x0110	mmc_intr_mask_tx	MMC Transmit Interrupt Mask maintains the mask for the interrupt generated from all of the transmit statistic counters.
69	0x0114	txoctetcount_gb	Number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad frames.
70	0x0118	txframecount_gb	Number of good and bad frames transmitted, exclusive of retried frames.
71	0x011C	txbroadcastframes_g	Number of good broadcast frames transmitted.
72	0x0120	txmulticastframes_g	Number of good multicast frames transmitted.
73	0x0124	tx64octets_gb	Number of good and bad frames transmitted with length 64 bytes, exclusive of preamble and retried frames.
74	0x0128	tx65to127octets_gb	Number of good and bad frames transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.
75	0x012C	tx128to255octets_gb	Number of good and bad frames transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames.
76	0x0130	tx256to511octets_gb	Number of good and bad frames transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.
77	0x0134	tx512to1023octets_gb	Number of good and bad frames transmitted with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble and retried frames.
78	0x0138	tx1024tomaxoctets_gb	Number of good and bad frames transmitted with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames
79	0x013C	txunicastframes_gb	Number of good and bad unicast frames transmitted.
80	0x0140	txmulticastframes_gb	Number of good and bad multicast frames transmitted.
81	0x0144	txbroadcastframes_gb	Number of good and bad broadcast frames transmitted.
82	0x0148	txunderflowerror	Number of frames aborted because of frame underflow error.
83	0x014C	txsinglecol_g	Number of successfully transmitted frames after a single collision in the half-duplex mode.
84	0x0150	txmulticol_g	Number of successfully transmitted frames after multiple collisions in the half-duplex mode.
85	0x0154	txdeferred	Number of successfully transmitted frames after a deferral in the half-duplex mode.
86	0x0158	txlatecol	Number of frames aborted because of late collision error.
87	0x015C	txexesscol	Number of frames aborted because of excessive (16) collision errors.
88	0x0160	txcarriererror	Number of frames aborted because of carrier sense error (no carrier or loss of carrier).
89	0x0164	txoctetcount_g	Number of bytes transmitted, exclusive of preamble, in good frames only.
90	0x0168	txframecount_g	Number of good frames transmitted.
91	0x016C	txexcessdef	Number of frames aborted because of excessive deferral error (deferred for more than two max-sized frame times).
92	0x0170	txpauseframes	Number of good PAUSE frames transmitted.
93	0x0174	txvlanframes_g	Number of good VLAN frames transmitted, exclusive of retried frames.
94	0x0178	txoversize_g	Number of frames transmitted without errors and with length greater

GMAC CSR Register No.	Address Offset	Register Name	Register Description
			than the maxsize (1,518 or 1,522 bytes for VLAN tagged frames; 2000 bytes if enabled in Bit 27 of Register 0 (MAC Configuration Register)).
95	0x017C	Reserved	
96	0x0180	rxframecount_gb	Number of good and bad frames received.
97	0x0184	rxoctetcount_gb	Number of bytes received, exclusive of preamble, in good and bad frames.
98	0x0188	rxoctetcount_g	Number of bytes received, exclusive of preamble, only in good frames.
99	0x018C	rbroadcastframes_g	Number of good broadcast frames received.
100	0x0190	rxmulticastframes_g	Number of good multicast frames received.
101	0x0194	rxcrcerror	Number of frames received with CRC error.
102	0x0198	rxalignmenterror	Number of frames received with alignment (dribble) error. Valid only in 10/100 mode.
103	0x019C	rxrunterror	Number of frames received with runt (<64 bytes and CRC error) error.
104	0x01A0	rxjabbererror	Number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames.
105	0x01A4	rxundersize_g	Number of frames received with length less than 64 bytes, without any errors.
106	0x01A8	rxoversize_g	Number of frames received without errors, with length greater than the maxsize (1,518 or 1,522 for VLAN tagged frames; 2,000 bytes if enabled in Bit 27 of Register 0 (MAC Configuration Register)).
107	0x01AC	rx64octets_gb	Number of good and bad frames received with length 64 bytes, exclusive of preamble.
108	0x01B0	rx65to127octets_gb	Number of good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble.
109	0x01B4	rx128to255octets_gb	Number of good and bad frames received with length between 128 and 255 (inclusive) bytes, exclusive of preamble.
110	0x01B8	rx256to511octets_gb	Number of good and bad frames received with length between 256 and 511 (inclusive) bytes, exclusive of preamble.
111	0x01BC	rx512to1023octets_gb	Number of good and bad frames received with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble.
112	0x01C0	rx1024tomaxoctets_gb	Number of good and bad frames received with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.
113	0x01C4	rxunicastframes_g	Number of received good unicast frames.
114	0x01C8	rxlengtherror	Number of frames received with length error (Length type field ≠ frame size), for all frames with valid length field.
115	0x01CC	rxoutofrangetype	Number of frames received with length field not equal to the valid frame size (greater than 1,500 but less than 1,536).
116	0x01D0	rxpauseframes	Number of good and valid PAUSE frames received.
117	0x01D4	rxfifooverflow	Number of missed received frames because of FIFO overflow. This counter is not present in the GMAC-CORE configuration.
118	0x01D8	rxvlanframes_gb	Number of good and bad VLAN frames received.
119	0x01DC	rxwatchdogerror	Number of frames received with error because of watchdog timeout error (frames with a data load larger than 2,048 bytes).
120	0x01E0	rxrcverror	Number of frames received with Receive error or Frame Extension error on the GMII or MII interface.
121	0x01E4	rxctrlframes_g	Number of received good control frames.

## 9.6 Power Management

The power management (PMT) block supports the reception of network (remote) wake-up frames and magic packet frames. The PMT block does not perform the clock gate function, but generates interrupts for wake-up frames and magic packets that the MAC receives.

When you enable the power-down mode in PMT block, then the MAC drops all received frames and does not forward these frames to the application. The MAC comes out of the power-down mode only when a magic packet or a remote wake-up frame is received and the corresponding detection is enabled.

The PMT block is available in the receive path of MAC. You can enable the PMT block by selecting the Enable Power Management option in coreConsultant. You can select both types of power management frames (remote wake-up and magic packet). You can use Bit 2 and Bit 1 of the PMT Control and Status Register to generate power management events. The application should program these bits. You can access the PMT registers in the similar manner as you access the MAC CSR registers.

### 9.6.1 PMT Register Map

Register No.	Offset Address	Register Name and Description
0	0x0028	<p>Remote Wake-Up Frame Filter Register</p> <p>This is the address through which the application writes or reads the remote wake-up frame filter registers (wkupfmfilter_reg). The wkupfmfilter_reg register is a pointer to eight wkupfmfilter_reg registers. The wkupfmfilter_reg register is loaded by sequentially loading the eight register values. Eight sequential writes to this address (0x0028) write all wkupfmfilter_reg registers. Similarly, eight sequential reads from this address (0x0028) read all wkupfmfilter_reg registers. This register contains the higher 16 bits of the seventh MAC address.</p> <p>This register is present only when you select the PMT module Remote Wake-up feature in coreConsultant.</p>
1	0x002C	<p>PMT Control and Status Register</p> <p>This register is present only when you select the PMT module in coreConsultant.</p>

### 9.6.2 Remote Wake-Up Frame Filter Register

The register wkupfmfilter\_reg, at address (028H), loads the Wake-up Frame Filter register. To load values in a Wake-up Frame Filter register, the entire register (wkupfmfilter\_reg) must be written. The wkupfmfilter\_reg register is loaded by sequentially loading the eight register values in address (028) for wkupfmfilter\_reg0, wkupfmfilter\_reg1, ..., wkupfmfilter\_reg7, respectively. The wkupfmfilter\_reg register is read in a similar way.

If you are accessing these registers in byte or half-word mode, then the internal counter, to access the appropriate wkupfmfilter\_reg, is incremented when CPU accesses the lane3 (or lane 0 in big-endian mode).

Wake-up frame filter register:

wkupfmfilter_reg0	Filter 0 Byte Mask											
wkupfmfilter_reg1	Filter 1 Byte Mask											
wkupfmfilter_reg2	Filter 2 Byte Mask											
wkupfmfilter_reg3	Filter 3 Byte Mask											
wkupfmfilter_reg4	RSVD	Filter3 Command	RSVD	Filter2 Command	RSVD	Filter1 Command	RSVD	Filter0 Command				
wkupfmfilter_reg5	Filter3 Offset		Filter 2 Offset		Filter 1 Offset		Filter 0 Offset					
wkupfmfilter_reg6	Filter 1 CRC - 16				Filter 0 CRC - 16							
wkupfmfilter_reg7	Filter 3 CRC - 16				Filter 2 CRC - 16							

#### Filter *i* Byte Mask

The Filter *i* Byte Mask register defines the bytes of the frame that are examined by filter *i* (0, 1, 2, and 3) in order to determine whether or not a frame is a wake-up frame. The MSB (31st bit) must be zero. The bit *j* [30:0] is the Byte Mask. If bit *j* (byte number) of the Byte Mask is set, then the CRC block processes the Filter *i* Offset + *j* of the incoming frame; otherwise Filter *i* Offset + *j* is ignored.

### Filter *i* Command

The 4-bit Filter *i* Command controls the filter *i* operation. The Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast frames; when the bit is reset, the pattern applies only to unicast frame. Bit 2 and Bit 1 are reserved. Bit 0 is the enable for filter *i*. If Bit 0 is not set, filter *i* is disabled.

### Filter *i* Offset

This Filter *i* Offset register defines the offset (within the frame) from which the filter *i* examines the frames. This 8-bit pattern-offset is the offset for the filter *i* first byte to be examined. The minimum allowed offset is 12, which refers to the 13th byte of the frame. The offset value 0 refers to the first byte of the frame.

### Filter *i* CRC-16

This Filter *i* CRC-16 register contains the CRC\_16 value calculated from the pattern, as well as the byte mask programmed to the wake-up filter register block.

## 9.7 Register

The Access column of each register description that follows specifies how the application and the core can access the register fields of the CSRs.

The Access column uses the following conventions:

Read Only (RO)	Register field can only be read by the application. Writes to read-only fields have no effect.
Write Only (WO)	Register field can only be written by the application.
Read, Write, and Self Clear (R_W_SC)	Register field can be read and written by the application. The application can set this field by writing 1'b1 and can clear it by writing 1'b0.
Read, Write, and Self Clear (R_W_SC)	Register field can be read and written by the application. The bit can be cleared to 1'b0 either by the core itself (Self Clear) or by the application with a register write of 1'b0 (Write Clear).
Read, Self Set, and Write Clear (R_SS_WC)	Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), and can be cleared to 1'b0 by the application with a register write of 1'b1 (Write Clear). A register write of 1'b0 has no effect on this field. The conditions under which the core sets this field are explained in detail in the description of the field (for example, interrupt bits).
Read, Write Set, and Self Clear (R_WS_SC)	Register field can be read by the application (Read), can be set to 1'b1 by the application with a register write of 1'b1 (Write Set), and is cleared to 1'b0 by the core (Self Clear). The application cannot clear this type of field, and a register write of 1'b0 to this bit has no effect on this field. The conditions under which the core clears this field are explained in detail in the description of the field, for example, soft-reset signals.
Read, Self Set, and Self Clear or Write Clear (R_SS_SC_WC)	Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), and can be cleared to 1'b0 either by the core itself (Self Clear) or by the application with a register write of 1'b0 (Write Clear). A register write of 1'b1 to this bit has no effect on this field. The conditions under which the core sets or clears this field are explained in detail in the description of the field.
Read Only and Write Trigger (RO_WT)	Register field can be read by the application, and when a write operation is performed with any data value, an event is triggered, as explained in the description of the field, for example, Tx Poll Demand register.
Read, Self Set, and Read Clear (R_SS_RC)	Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), and is automatically cleared to 1'b0 on a register read. A register write has no effect on this field. The conditions under which the core sets this field are explained in detail in the description of the field, for example, Overflow counter.
Read, Write, and Self Update (R_W_SU)	Register field can be read and written by the application. The register field updates itself based on the event. For example, system time in PTP configuration.

Read, Self Set, Self Clear, and Latch Low (R_SS_SC_LLO)	Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), can be cleared to 1'b0 by the core (Self Clear), and on reset, it is latched to low value (Latch Low). For example, link up and down status (LS: Link Status) in AN Status Register.
---	---

### 9.7.1 GMA Register Map

Registers for the DWC\_gmac\_top\_map Memory Map

The address block GMAC contains up to 736 Registers : MAC_Configuration to MAC_Address127_Low				
Register	Offset	Size	Memory Access	Description
<a href="#">MAC_Configuration</a>	0x0	32 bits	R/W	<p><b>Value After Reset:</b> 0x8800</p> <p><b>Register 0 (MAC Configuration Register)</b></p> <p>The MAC Configuration register establishes receive and transmit operating modes.</p>
<a href="#">MAC_Frame_Filter</a>	0x4	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 1 (MAC Frame Filter)</b></p> <p>The MAC Frame Filter register contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering. The second level of filtering is performed on the incoming frame, based on other controls such as Pass Bad Frames and Pass Control Frames.</p>
<a href="#">GMII_Address</a>	0x10	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 4 (GMII Address Register)</b></p> <p>The GMII Address register controls the management cycles to the external PHY through the management interface.</p> <p>Note: This register is present for all PHY interface when you select the Station Management (MDIO) feature in coreConsultant.</p>
<a href="#">GMII_Data</a>	0x14	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 5 (GMII Data Register)</b></p> <p>The GMII Data register stores Write data to be written to the PHY register located at the address specified in Register 4 (GMII Address Register). This register also stores the Read data from the PHY register located at the address specified by Register 4.</p> <p>Note: This register is present for all PHY interface when you select the Station Management (MDIO) feature in coreConsultant.</p>
<a href="#">Flow_Control</a>	0x18	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 6 (Flow Control Register)</b></p> <p>The Flow Control register controls the generation and reception of the Control (Pause Command) frames by the MAC's Flow control module. A Write to a register with the Busy bit set to '1' triggers the Flow Control block to generate a Pause Control frame. The fields of the control frame are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control frame. The Busy bit remains set until the control frame is transferred onto the cable. The Host must make sure that the Busy bit is cleared before writing to the register.</p>

Register	Offset	Size	Memory Access	Description
<a href="#">VLAN Tag</a>	0x1c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 7 (VLAN Tag Register)</b></p> <p>The VLAN Tag register contains the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the receiving frame (Length/Type) with 16'h8100, and the following two bytes are compared with the VLAN tag. If a match occurs, the MAC sets the received VLAN bit in the receive frame status. The legal length of the frame is increased from 1,518 bytes to 1,522 bytes. If the VLAN Tag register is configured to be double-synchronized to the (G)MII clock domain, then consecutive writes to these register should be performed only after at least four clock cycles in the destination clock domain.</p>
<a href="#">Version</a>	0x20	32 bits	R	<p><b>Value After Reset:</b> 0x1037</p> <p><b>Register 8 (Version Register)</b></p> <p>The Version registers identifies the version of the DWC_gmac. This register contains two bytes: one that Synopsys uses to identify the core release number, and the other that you set during core configuration.</p>
<a href="#">Debug</a>	0x24	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 9 (Debug Register)</b></p> <p>The Debug register gives the status of all main modules of the transmit and receive data-paths and the FIFOs. An all-zero status indicates that the MAC is in idle state (and FIFOs are empty) and no activity is going on in the data-paths.</p> <p><b>Note:</b> The reset values, given for the Debug register, are valid only if the following clocks are present during the reset operation: clk_csr_i, clk_app_i, hclk_i, or aclk_i clk_tx_i clk_rx_i</p>
<a href="#">Remote Wake Up Frame Filter</a>	0x28	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 10 (Remote Wake-Up Frame Filter Register)</b></p> <p>This is the address through which the application writes or reads the remote wake-up frame filter registers (wkupfmfilter_reg). The wkupfmfilter_reg register is a pointer to eight wkupfmfilter_reg registers. The wkupfmfilter_reg register is loaded by sequentially loading the eight register values. Eight sequential writes to this address (0x0028) writes all wkupfmfilter_reg registers. Similarly, eight sequential reads from this address (0x0028) read all wkupfmfilter_reg registers. This register contains the higher 16 bits of the seventh MAC address. This register is present only when you select the PMT module Remote Wake-up feature in coreConsultant.</p>
<a href="#">PMT Control Status</a>	0x2c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 11 (PMT Control and Status Register)</b></p> <p>This register is present only when you select the PMT module in the coreConsultant.</p>
<a href="#">Interrupt Status</a>	0x38	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 14 (Interrupt Register)</b></p>

Register	Offset	Size	Memory Access	Description
				The Interrupt Status register identifies the events in the MAC that can generate interrupt. All interrupt events are generated only when the corresponding optional feature is selected during core configuration and enabled during operation. Therefore, these bits are reserved when the corresponding features are not present in the core.
<a href="#">Interrupt_Mask</a>	0x3c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 15 (Interrupt Mask Register)</b></p> <p>The Interrupt Mask Register bits enable you to mask the interrupt signal because of the corresponding event in the Interrupt Status Register. The interrupt signal is sbd_intr_o in the GMAC-AHB, GMAC-AXI, and GMAC-DMA configuration and mci_intr_o in the GMAC-MTL and GMAC-CORE configuration.</p>
<a href="#">MAC_Address0_High</a>	0x40	32 bits	R/W	<p><b>Value After Reset:</b> 0x8000ffff</p> <p><b>Register 16 (MAC Address0 High Register)</b></p> <p>The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the (G)MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the (G)MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x665544332211.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC_Address0_Low</a>	0x44	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 17 (MAC Address0 Low Register)</b></p> <p>The MAC Address0 Low register holds the lower 32 bits of the first 6-byte MAC address of the station.</p>
<a href="#">MAC_Address1_High</a>	0x48	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 18 (MAC Address1 High Register)</b></p> <p>The MAC Address1 High register holds the upper 16 bits of the second 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address1 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC_Address1_Low</a>	0x4c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 19 (MAC Address1 Low Register)</b></p>

Register	Offset	Size	Memory Access	Description
				The MAC Address1 Low register holds the lower 32 bits of the second 6-byte MAC address of the station.
<a href="#"><u>MAC Address2 High</u></a>	0x50	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 20 (MAC Address2 High Register)</b></p> <p>The MAC Address2 High register holds the upper 16 bits of the third 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address2 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address2 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address2 Low</u></a>	0x54	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 21 (MAC Address2 Low Register)</b></p> <p>The MAC Address2 Low register holds the lower 32 bits of the third 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address3 High</u></a>	0x58	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 22 (MAC Address3 High Register)</b></p> <p>The MAC Address3 High register holds the upper 16 bits of the fourth 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address3 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address3 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address3 Low</u></a>	0x5c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 23 (MAC Address3 Low Register)</b></p> <p>The MAC Address3 Low register holds the lower 32 bits of the fourth 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address4 High</u></a>	0x60	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 24 (MAC Address4 High Register)</b></p> <p>The MAC Address4 High register holds the upper 16 bits of the fifth 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address4 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address4 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address4 Low</u></a>	0x64	32 bits	R/W	<b>Value After Reset:</b> 0xffffffff

Register	Offset	Size	Memory Access	Description
				<b>Register 25 (MAC Address4 Low Register)</b> The MAC Address4 Low register holds the lower 32 bits of the fifth 6-byte MAC address of the station.
<a href="#">MAC Address5 High</a>	0x68	32 bits	R/W	<b>Value After Reset:</b> 0xffff <b>Register 26 (MAC Address5 High Register)</b> The MAC Address5 High register holds the upper 16 bits of the sixth 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address5 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address5 Low Register must be performed after at least four clock cycles in the destination clock domain.
<a href="#">MAC Address5 Low</a>	0x6c	32 bits	R/W	<b>Value After Reset:</b> 0xffffffff <b>Register 27 (MAC Address5 Low Register)</b> The MAC Address5 Low register holds the lower 32 bits of the sixth 6-byte MAC address of the station.
<a href="#">MAC Address6 High</a>	0x70	32 bits	R/W	<b>Value After Reset:</b> 0xffff <b>Register 28 (MAC Address6 High Register)</b> The MAC Address6 High register holds the upper 16 bits of the seventh 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address6 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address6 Low Register should be performed after at least four clock cycles in the destination clock domain.
<a href="#">MAC Address6 Low</a>	0x74	32 bits	R/W	<b>Value After Reset:</b> 0xffffffff <b>Register 29 (MAC Address6 Low Register)</b> The MAC Address6 Low register holds the lower 32 bits of the seventh 6-byte MAC address of the station.
<a href="#">MAC Address7 High</a>	0x78	32 bits	R/W	<b>Value After Reset:</b> 0xffff <b>Register 30 (MAC Address7 High Register)</b> The MAC Address7 High register holds the upper 16 bits of the eighth 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address7 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address7 Low Register should be performed after at least four clock cycles in the destination clock domain.
<a href="#">MAC Address7 Low</a>	0x7c	32 bits	R/W	<b>Value After Reset:</b> 0xffffffff <b>Register 31 (MAC Address7 Low Register)</b>

Register	Offset	Size	Memory Access	Description
				The MAC Address7 Low register holds the lower 32 bits of the eighth 6-byte MAC address of the station.
<a href="#">MAC Address8 High</a>	0x80	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 32 (MAC Address8 High Register)</b></p> <p>The MAC Address8 High register holds the upper 16 bits of the ninth 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address8 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address8 Low Register should be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address8 Low</a>	0x84	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 33 (MAC Address8 Low Register)</b></p> <p>The MAC Address8 Low register holds the lower 32 bits of the ninth 6-byte MAC address of the station.</p>
<a href="#">MAC Address9 High</a>	0x88	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 34 (MAC Address9 High Register)</b></p> <p>The MAC Address9 High register holds the upper 16 bits of the tenth 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address9 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address9 Low Register should be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address9 Low</a>	0x8c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 35 (MAC Address9 Low Register)</b></p> <p>The MAC Address9 Low register holds the lower 32 bits of the tenth 6-byte MAC address of the station.</p>
<a href="#">MAC Address10 High</a>	0x90	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 36 (MAC Address10 High Register)</b></p> <p>The MAC Address10 High register holds the upper 16 bits of the 11th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address10 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address10 Low Register should be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address10 Low</a>	0x94	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 37 (MAC Address10 Low Register)</b></p> <p>The MAC Address10 Low register holds the lower 32 bits of the 11th 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#"><u>MAC Address11 High</u></a>	0x98	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 38 (MAC Address11 High Register)</b></p> <p>The MAC Address11 High register holds the upper 16 bits of the 12th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address11 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address11 Low Register should be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address11 Low</u></a>	0x9c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 39 (MAC Address11 Low Register)</b></p> <p>The MAC Address11 Low register holds the lower 32 bits of the 12th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address12 High</u></a>	0xa0	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 40 (MAC Address12 High Register)</b></p> <p>The MAC Address12 High register holds the upper 16 bits of the 13th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address12 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address12 Low Register should be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address12 Low</u></a>	0xa4	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 41 (MAC Address12 Low Register)</b></p> <p>The MAC Address12 Low register holds the lower 32 bits of the 13th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address13 High</u></a>	0xa8	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 42 (MAC Address13 High Register)</b></p> <p>The MAC Address13 High register holds the upper 16 bits of the 14th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address13 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address13 Low Register should be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address13 Low</u></a>	0xac	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 43 (MAC Address13 Low Register)</b></p> <p>The MAC Address13 Low register holds the lower 32 bits of the 14th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address14 High</u></a>	0xb0	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 44 (MAC Address14 High Register)</b></p>

Register	Offset	Size	Memory Access	Description
				The MAC Address14 High register holds the upper 16 bits of the 15th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address15 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address14 Low Register should be performed after at least four clock cycles in the destination clock domain.
<a href="#">MAC Address14 Low</a>	0xb4	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 45 (MAC Address14 Low Register)</b></p> <p>The MAC Address14 Low register holds the lower 32 bits of the 15th 6-byte MAC address of the station.</p>
<a href="#">MAC Address15 High</a>	0xb8	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 46 (MAC Address15 High Register)</b></p> <p>The MAC Address15 High register holds the upper 16 bits of the 16th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address15 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address15 Low Register should be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address15 Low</a>	0xbc	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 47 (MAC Address15 Low Register)</b></p> <p>The MAC Address15 Low register holds the lower 32 bits of the 16th 6-byte MAC address of the station.</p>
<a href="#">WDog Timeout</a>	0xdc	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 55 (Watchdog Timeout Register)</b> This register controls the watchdog timeout for received frames.</p>
<a href="#">MMC Control</a>	0x100	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 64 (MMC Control Register)</b></p> <p>The MMC Control register establishes the operating mode of the management counters.</p> <p>Note: The bit 0 (Counters Reset) has higher priority than bit 4 (Counter Preset). Therefore, when the Software tries to set both bits in the same write cycle, all counters are cleared and the bit 4 is not set.</p>
<a href="#">MMC Receive Interrupt</a>	0x104	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 65 (MMC Receive Interrupt Register)</b></p> <p>The MMC Receive Interrupt register maintains the interrupts that are generated when the following happens:</p> <p>Receive statistic counters reach half of their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter).</p> <p>Receive statistic counters cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter).</p>

Register	Offset	Size	Memory Access	Description
				When the Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.
<a href="#"><u>MMC Transmit Interrupt</u></a>	0x108	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 66 (MMC Transmit Interrupt Register)</b></p> <p>The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half of their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter), and the maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Transmit Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.</p>
<a href="#"><u>MMC Receive Interrupt Mask</u></a>	0x10c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Registers 67 (MMC Receive Interrupt Mask Register)</b></p> <p>The MMC Receive Interrupt Mask register maintains the masks for the interrupts generated when the receive statistic counters reach half of their maximum value, or maximum value. This register is 32-bits wide.</p>
<a href="#"><u>MMC Transmit Interrupt Mask</u></a>	0x110	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 68 (MMC Transmit Interrupt Mask Register)</b></p> <p>The MMC Transmit Interrupt Mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or maximum value. This register is 32-bits wide.</p>
<a href="#"><u>Tx Octet Count Good Bad</u></a>	0x114	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 69 (Transmit Octet Count for Good and Bad Frames)</b></p> <p>This register maintains the number of bytes transmitted in good and bad frames exclusive of preamble and retried bytes.</p>
<a href="#"><u>Tx Frame Count Good Bad</u></a>	0x118	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 70 (Transmit Frame Count for Good and Bad Frames)</b></p> <p>This register maintains the number of good and bad frames transmitted, exclusive of retried frames.</p>
<a href="#"><u>Tx Broadcast Frames Good</u></a>	0x11c	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 71 (Transmit Frame Count for Good Broadcast Frames)</b></p> <p>This register maintains the number of transmitted good broadcast frames.</p>
<a href="#"><u>Tx Multicast Frames Good</u></a>	0x120	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 72 (Transmit Frame Count for Good Multicast Frames)</b></p> <p>This register maintains the number of transmitted good multicast</p>

Register	Offset	Size	Memory Access	Description
				frames.
<a href="#"><u>Tx_64Octets_Frames_Good_Bad</u></a>	0x124	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 73 (Transmit Octet Count for Good and Bad 64 Byte Frames)</b></p> <p>This register maintains the number of transmitted good and bad frames with length of 64 bytes, exclusive of preamble and retried frames.</p>
<a href="#"><u>Tx_65To127Octets_Frames_Good_Bad</u></a>	0x128	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 74 (Transmit Octet Count for Good and Bad 65 to 127 Bytes Frames)</b></p> <p>This register maintains the number of transmitted good and bad frames with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.</p>
<a href="#"><u>Tx_128To255Octets_Frames_Good_Bad</u></a>	0x12c	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 75 (Transmit Octet Count for Good and Bad 128 to 255 Bytes Frames)</b></p> <p>This register maintains the number of transmitted good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames.</p>
<a href="#"><u>Tx_256To511Octets_Frames_Good_Bad</u></a>	0x130	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 76 (Transmit Octet Count for Good and Bad 256 to 511 Bytes Frames)</b></p> <p>This register maintains the number of transmitted good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.</p>
<a href="#"><u>Tx_512To1023Octets_Frames_Good_Bad</u></a>	0x134	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 77 (Transmit Octet Count for Good and Bad 512 to 1023 Bytes Frames)</b></p> <p>This register maintains the number of transmitted good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble and retried frames.</p>
<a href="#"><u>Tx_1024ToMaxOctets_Frames_Good_Bad</u></a>	0x138	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 78 (Transmit Octet Count for Good and Bad 1024 to Maxsize Bytes Frames)</b></p> <p>This register maintains the number of transmitted good and bad frames with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.</p>
<a href="#"><u>Tx_Uncast_Frames_Good_Bad</u></a>	0x13c	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 79 (Transmit Frame Count for Good and Bad Unicast Frames)</b></p> <p>This register maintains the number of transmitted good and bad unicast frames.</p>
<a href="#"><u>Tx_Multicast_Frames_Good_Bad</u></a>	0x140	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 80 (Transmit Frame Count for Good and Bad Multicast Frames)</b></p> <p>This register maintains the number of transmitted good and bad</p>

Register	Offset	Size	Memory Access	Description
				multicast frames.
<a href="#"><u>Tx_Broadcast_Frame_S_Good_Bad</u></a>	0x144	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 81 (Transmit Frame Count for Good and Bad Broadcast Frames)</b> This register maintains the number of transmitted good and bad broadcast frames.
<a href="#"><u>Tx_Underflow_Error_Frames</u></a>	0x148	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 82 (Transmit Frame Count for Underflow Error Frames)</b> This register maintains the number of frames aborted because of frame underflow error.
<a href="#"><u>Tx_Octet_Count_Good</u></a>	0x164	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 89 (Transmit Octet Count for Good Frames)</b> This register maintains the number of bytes transmitted, exclusive of preamble, in good frames.
<a href="#"><u>Tx_Frame_Count_Good</u></a>	0x168	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 90 (Transmit Frame Count for Good Frames)</b> This register maintains the number of transmitted good frames, exclusive of preamble.
<a href="#"><u>Tx_Pause_Frames</u></a>	0x170	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 92 (Transmit Frame Count for Good PAUSE Frames)</b> This register maintains the number of transmitted good PAUSE frames.
<a href="#"><u>Tx_VLAN_Frames_Good</u></a>	0x174	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 93 (Transmit Frame Count for Good VLAN Frames)</b> This register maintains the number of transmitted good VLAN frames, exclusive of retried frames.
<a href="#"><u>Tx_OSize_Frames_Good</u></a>	0x178	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 94 (Transmit Frame Count for Good Oversize Frames)</b> This register maintains the number of transmitted good Oversize frames, exclusive of retried frames.
<a href="#"><u>Rx_Frames_Count_Good_Bad</u></a>	0x180	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 96 (Receive Frame Count for Good and Bad Frames)</b> This register maintains the number of received good and bad frames.
<a href="#"><u>Rx_Octet_Count_Good_Bad</u></a>	0x184	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 97 (Receive Octet Count for Good and Bad Frames)</b> This register maintains the number of bytes received, exclusive of preamble, in good and bad frames.
<a href="#"><u>Rx_Octet_Count_Good</u></a>	0x188	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 98 (Receive Octet Count for Good Frames)</b> This register maintains the number of bytes received, exclusive of preamble, only in good frames.
<a href="#"><u>Rx_Broadcast_Frame</u></a>	0x18c	32 bits	R	<b>Value After Reset:</b> 0x0

Register	Offset	Size	Memory Access	Description
<a href="#"><u>s_Good</u></a>				<b>Register 99 (Receive Frame Count for Good Broadcast Frames)</b> This register maintains the number of received good broadcast frames.
<a href="#"><u>Rx_Multicast_Frames_Good</u></a>	0x190	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 100 (Receive Frame Count for Good Multicast Frames)</b> This register maintains the number of received good multicast frames.
<a href="#"><u>Rx_CRC_Error_Frames</u></a>	0x194	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 101 (Receive Frame Count for CRC Error Frames)</b> This register maintains the number of frames received with CRC error.
<a href="#"><u>Rx_Alignment_Error_Frames</u></a>	0x198	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 102 (Receive Frame Count for Alignment Error Frames)</b> This register maintains the number of frames received with alignment (dribble) error. This field is valid only in the 10 or 100 Mbps mode.
<a href="#"><u>Rx_Runt_Error_Frames</u></a>	0x19c	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 103 (Receive Frame Count for Runt Error Frames)</b> This register maintains the number of frames received with runt error(<64 bytes and CRC error).
<a href="#"><u>Rx_Jabber_Error_Frames</u></a>	0x1a0	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 104 (Receive Frame Count for Jabber Error Frames)</b> This register maintains the number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames.
<a href="#"><u>Rx_Undersize_Frames_Good</u></a>	0x1a4	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 105 (Receive Frame Count for Undersize Frames)</b> This register maintains the number of frames received with length less than 64 bytes and without errors.
<a href="#"><u>Rx_Oversize_Frames_Good</u></a>	0x1a8	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 106 (Receive Frame Count for Oversize Frames)</b> This register maintains the number of frames received with length greater than the maxsize (1,518 or 1,522 for VLAN tagged frames) and without errors.
<a href="#"><u>Rx_64Octets_Frames_Good_Bad</u></a>	0x1ac	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 107 (Receive Frame Count for Good and Bad 64 Byte Frames)</b> This register maintains the number of received good and bad frames with length 64 bytes, exclusive of preamble.
<a href="#"><u>Rx_65To127Octets_Frames_Good_Bad</u></a>	0x1b0	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 108 (Receive Frame Count for Good and Bad 65 to 127 Bytes Frames)</b> This register maintains the number of received good and bad

Register	Offset	Size	Memory Access	Description
				frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble.
<a href="#"><u>Rx_128To255Octets_Frames_Good_Bad</u></a>	0x1b4	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 109 (Receive Frame Count for Good and Bad 128 to 255 Bytes Frames)</b> This register maintains the number of received good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble.
<a href="#"><u>Rx_256To511Octets_Frames_Good_Bad</u></a>	0x1b8	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 110 (Receive Frame Count for Good and Bad 256 to 511 Bytes Frames)</b> This register maintains the number of received good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble.
<a href="#"><u>Rx_512To1023Octets_Frames_Good_Bad</u></a>	0x1bc	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 111 (Receive Frame Count for Good and Bad 512 to 1,023 Bytes Frames)</b> This register maintains the number of received good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble.
<a href="#"><u>Rx_1024ToMaxOctets_Frames_Good_Bad</u></a>	0x1c0	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 112 (Receive Frame Count for Good and Bad 1,024 to Maxsize Bytes Frames)</b> This register maintains the number of received good and bad frames with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble.
<a href="#"><u>Rx_Uicast_Frames_Good</u></a>	0x1c4	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 113 (Receive Frame Count for Good Unicast Frames)</b> This register maintains the number of received good unicast frames.
<a href="#"><u>Rx_Length_Error_Frames</u></a>	0x1c8	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 114 (Receive Frame Count for Length Error Frames)</b> This register maintains the number of frames received with length error (Length type field not equal to frame size) for all frames with valid length field.
<a href="#"><u>Rx_Out_Of_Range_Type_Frames</u></a>	0x1cc	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 115 (Receive Frame Count for Out of Range Frames)</b> This register maintains the number of received frames with length field not equal to the valid frame size (greater than 1,500 but less than 1,536).
<a href="#"><u>Rx_Pause_Frames</u></a>	0x1d0	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 116 (Receive Frame Count for PAUSE Frames)</b> This register maintains the number of received good and valid PAUSE frames.
<a href="#"><u>Rx_FIFO_Overflow_Frames</u></a>	0x1d4	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 117 (Receive Frame Count for FIFO Overflow Frames)</b>

Register	Offset	Size	Memory Access	Description
				This register maintains the number of received frames missed because of FIFO overflow.
<a href="#">Rx VLAN Frames Good Bad</a>	0x1d8	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 118 (Receive Frame Count for Good and Bad VLAN Frames)</b> This register maintains the number of received good and bad VLAN frames.
<a href="#">Rx Watchdog Error Frames</a>	0x1dc	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 119 (Receive Frame Count for Watchdog Error Frames)</b> This register maintains the number of frames received with error because of the watchdog timeout error (frames with more than 2,048 bytes or value programmed in Register 55 (Watchdog Timeout Register)).
<a href="#">Rx Receive Error Frames</a>	0x1e0	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 120 (Receive Frame Count for Receive Error Frames)</b> This register maintains the number of frames received with error because of the GMII/MII RXER error.
<a href="#">Rx Control Frames Good</a>	0x1e4	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 121 (Receive Frame Count for Good Control Frames)</b> This register maintains the number of good control frames received.
<a href="#">L3_L4_Control0</a>	0x400	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>Register 256 (Layer 3 and Layer 4 Control Register 0)</b> This register controls the operations of the filter 0 of Layer 3 and Layer 4. This register is reserved if the Layer 3 and Layer 4 Filtering feature is not selected during core configuration.
<a href="#">Layer4_Address0</a>	0x404	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>Register 257 (Layer 4 Address Register 0)</b> You can configure the Layer 3 and Layer 4 Address Registers to be double-synchronized by selecting the Synchronize Layer 3 and Layer 4 Address Registers to Rx Clock Domain option in coreConsultant. If the Layer 3 and Layer 4 Address Registers are configured to be double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock. If the Layer 3 and Layer 4 Filtering feature is not selected during core configuration, this register and registers 260 through 299 are reserved (RO with default value).
<a href="#">Layer3_Addr0_Reg0</a>	0x410	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>Register 260 (Layer 3 Address 0 Register 0)</b> For IPv4 frames, the Layer 3 Address 0 Register 0 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits[31:0] of the

Register	Offset	Size	Memory Access	Description
				128-bit IP Source Address or Destination Address field.
<a href="#">Layer3 Addr1 Reg0</a>	0x414	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 261 (Layer 3 Address 1 Register 0)</b></p> <p>For IPv4 frames, the Layer 3 Address 1 Register 0 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.</p>
<a href="#">Layer3 Addr2 Reg0</a>	0x418	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 262 (Layer 3 Address 2 Register 0)</b></p> <p>For IPv4 frames, the Layer 3 Address 2 Register 0 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.</p>
<a href="#">Layer3 Addr3 Reg0</a>	0x41c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 263 (Layer 3 Address 3 Register 0)</b></p> <p>For IPv4 frames, the Layer 3 Address 3 Register 0 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.</p>
<a href="#">L3_L4_Control1</a>	0x430	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 268 (Layer 3 and Layer 4 Control Register 1)</b></p> <p>This register controls the operations of the filter 0 of Layer 3 and Layer 4. This register is reserved if the Layer 3 and Layer 4 Filtering feature is not selected during core configuration.</p>
<a href="#">Layer4_Address1</a>	0x434	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 269 (Layer 4 Address Register 1)</b></p> <p>You can configure the Layer 3 and Layer 4 Address Registers to be double-synchronized by selecting the Synchronize Layer 3 and Layer 4 Address Registers to Rx Clock Domain option in coreConsultant. If the Layer 3 and Layer 4 Address Registers are configured to be double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock.</p> <p>If the Layer 3 and Layer 4 Filtering feature is not selected during core configuration, this register is reserved (RO with default value).</p>
<a href="#">Layer3 Addr0_Reg1</a>	0x440	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 272 (Layer 3 Address 0 Register 1)</b></p> <p>For IPv4 frames, the Layer 3 Address 0 Register 1 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.</p>
<a href="#">Layer3 Addr1_Reg1</a>	0x444	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 273 (Layer 3 Address 1 Register 1)</b></p> <p>For IPv4 frames, the Layer 3 Address 1 Register 1 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field</p>

Register	Offset	Size	Memory Access	Description
<a href="#">Layer3 Addr2 Reg1</a>	0x448	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 274 (Layer 3 Address 2 Register 1)</b></p> <p>For IPv4 frames, the Layer 3 Address 2 Register 1 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.</p>
<a href="#">Layer3 Addr3 Reg1</a>	0x44c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 275 (Layer 3 Address 3 Register 1)</b></p> <p>For IPv4 frames, the Layer 3 Address 3 Register 1 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.</p>
<a href="#">L3_L4_Control2</a>	0x460	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 280 (Layer 3 and Layer 4 Control Register 2)</b></p> <p>This register controls the operations of the filter 2 of Layer 3 and Layer 4. This register is reserved if the Layer 3 and Layer 4 Filtering feature is not selected during core configuration.</p>
<a href="#">Layer4_Address2</a>	0x464	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 281 (Layer 4 Address Register 2)</b></p> <p>You can configure the Layer 3 and Layer 4 Address Registers to be double-synchronized by selecting the Synchronize Layer 3 and Layer 4 Address Registers to Rx Clock Domain option in coreConsultant. If the Layer 3 and Layer 4 Address Registers are configured to be double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock.</p> <p>If the Layer 3 and Layer 4 Filtering feature is not selected during core configuration, this register is reserved (RO with default value).</p>
<a href="#">Layer3_Addr0_Reg2</a>	0x470	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 284 (Layer 3 Address 0 Register 2)</b></p> <p>For IPv4 frames, the Layer 3 Address 0 Register 2 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits [31:0] of the 128-bit IP Source Address or Destination Address field.</p>
<a href="#">Layer3_Addr1_Reg2</a>	0x474	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 285 (Layer 3 Address 1 Register 2)</b></p> <p>For IPv4 frames, the Layer 3 Address 1 Register 2 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits [63:32] of the 128-bit IP Source Address or Destination Address field.</p>
<a href="#">Layer3_Addr2_Reg2</a>	0x478	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 286 (Layer 3 Address 2 Register 2)</b></p> <p>For IPv4 frames, the Layer 3 Address 2 Register 2 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.</p>
<a href="#">Layer3_Addr3_Reg2</a>	0x47c	32 bits	R/W	<b>Value After Reset:</b> 0x0

Register	Offset	Size	Memory Access	Description
				<b>Register 287 (Layer 3 Address 3 Register 2)</b> For IPv4 frames, the Layer 3 Address 3 Register 2 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.
<a href="#">L3_L4_Control3</a>	0x490	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>Register 292 (Layer 3 and Layer 4 Control Register 3)</b> This register controls the operations of the filter 0 of Layer 3 and Layer 4. This register is reserved if the Layer 3 and Layer 4 Filtering feature is not selected during core configuration.
<a href="#">Layer4_Address3</a>	0x494	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>Register 293 (Layer 4 Address Register 3)</b> You can configure the Layer 3 and Layer 4 Address Registers to be double-synchronized by selecting the Synchronize Layer 3 and Layer 4 Address Registers to Rx Clock Domain option in coreConsultant. If the Layer 3 and Layer 4 Address Registers are configured to be double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock. If the Layer 3 and Layer 4 Filtering feature is not selected during core configuration, this register is reserved (RO with default value).
<a href="#">Layer3_Addr0_Reg3</a>	0x4a0	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>Register 296 (Layer 3 Address 0 Register 3)</b> For IPv4 frames, the Layer 3 Address 0 Register 3 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits [31:0] of the 128-bit IP Source Address or Destination Address field.
<a href="#">Layer3_Addr1_Reg3</a>	0x4a4	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>Register 297 (Layer 3 Address 1 Register 3)</b> For IPv4 frames, the Layer 3 Address 1 Register 3 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits [63:32] of the 128-bit IP Source Address or Destination Address field.
<a href="#">Layer3_Addr2_Reg3</a>	0x4a8	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>Register 298 (Layer 3 Address 2 Register 3)</b> For IPv4 frames, the Layer 3 Address 2 Register 3 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.
<a href="#">Layer3_Addr3_Reg3</a>	0x4ac	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>Register 299 (Layer 3 Address 3 Register 3)</b> For IPv4 frames, the Layer 3 Address 3 Register 3 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.
<a href="#">Hash_Table_Reg0</a>	0x500	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>Register 320 (Hash Table Register 0)</b>

Register	Offset	Size	Memory Access	Description
				<p>This register contains the first 32 bits of the hash table when the width of the Hash table is 128 bits or 256 bits. You can specify the width of the hash table by using the Hash Table Size option in coreConsultant.</p> <p>The 128-bit or 256-bit Hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming frame is passed through the CRC logic and the upper seven (eight in 256-bit Hash) bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register X), and the least significant five bits determine the bit within the register. For example, a hash value of 7b'1100000 (in 128-bit Hash) selects Bit 0 of the Hash Table Register 3 and a value of 8b'10111111 (in 256-bit Hash) selects Bit 31 of the Hash Table Register 5.</p> <p>The hash value of the destination address is calculated in the following way:</p> <ol style="list-style-type: none"> <li>1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).</li> <li>2. Perform bitwise reversal for the value obtained in Step 1.</li> <li>3. Take the upper 7 (or 8) bits from the value obtained in Step 2.</li> </ol> <p>If the corresponding bit value of the register is 1'b1, the frame is accepted. Otherwise, it is rejected. If the Bit 1 (Pass All Multicast) is set in Register 1 (MAC Frame Filter), then all multicast frames are accepted regardless of the multicast hash values. If the Hash Table register is configured to be double-synchronized to the (G)MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Hash Table Register X registers are written.</p> <p>Note: If double-synchronization is enabled, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">Hash Table Reg1</a>	0x504	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 321 (Hash Table Register 1)</b></p> <p>This register contains the second 32 bits of the hash table when the width of the Hash table is 128 bits or 256 bits. You can specify the width of the hash table by using the Hash Table Size option in coreConsultant.</p>
<a href="#">Hash Table Reg2</a>	0x508	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 322 (Hash Table Register 2)</b></p> <p>This register contains the third 32 bits of the hash table when the width of the Hash table is 128 bits or 256 bits. You can specify the width of the hash table by using the Hash Table Size option in coreConsultant.</p>
<a href="#">Hash Table Reg3</a>	0x50c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 323 (Hash Table Register 3)</b></p> <p>This register contains the fourth 32 bits of the hash table when the width of the Hash table is 128 bits or 256 bits. You can specify the width of the hash table by using the Hash Table Size option in coreConsultant.</p>

Register	Offset	Size	Memory Access	Description
<a href="#">VLAN_Incl_Reg</a>	0x584	32 bits	R/W	<p><b>Register 353 (VLAN Tag Inclusion or Replacement Register)</b>  The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the transmit frames. This register is present only when the Enable SA, VLAN, and CRC Insertion on TX option is selected during core configuration.</p>
<a href="#">VLAN_Hash_Table_Reg</a>	0x588	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 354 (VLAN Hash Table Register)</b>  The 16-bit Hash table is used for group address filtering based on VLAN tag when Bit 18 (VTHM) of Register 7 (VLAN Tag Register) is set. For hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on Bit 16 (ETV) of VLAN Tag Register) in the incoming frame is passed through the CRC logic and the upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a hash value of 4b'1000 selects Bit 8 of the VLAN Hash table. The hash value of the destination address is calculated in the following way:  1. Calculate the 32-bit CRC for the VLAN tag or ID (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).  2. Perform bitwise reversal for the value obtained in Step 1.  3. Take the upper four bits from the value obtained in Step 2.  If the corresponding bit value of the register is 1'b1, the frame is accepted. Otherwise, it is rejected. If the Hash Table register is configured to be double-synchronized to the (G)MII clock domain, the synchronization is triggered only when Bits[15:8] (in little-endian mode) or Bits[7:0] (in big-endian mode) of this register are written.  <b>Notes:</b>  If double-synchronization is enabled, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.  To help you program the hash table, a sample C routine that generates a VLAN tag's 4-bit hash is included in /sample_codes/ directory of your workspace.</p>
<a href="#">MAC_Address16_High</a>	0x800	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 512 (MAC Address16 High Register)</b>  The MAC Address16 High register holds the upper 16 bits of the 17th 6-byte MAC address of the station.  If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address16 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address16 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC_Address16_Low</a>	0x804	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 513 (MAC Address16 Low Register)</b>  The MAC Address16 Low register holds the lower 32 bits of the 17th 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#"><u>MAC Address17 High</u></a>	0x808	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 514 (MAC Address17 High Register)</b></p> <p>The MAC Address17 High register holds the upper 16 bits of the 18th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address17 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address17 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address17 Low</u></a>	0x80c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 515 (MAC Address17 Low Register)</b></p> <p>The MAC Address17 Low register holds the lower 32 bits of the 18th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address18 High</u></a>	0x810	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 516 (MAC Address18 High Register)</b></p> <p>The MAC Address18 High register holds the upper 16 bits of the 19th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address18 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address18 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address18 Low</u></a>	0x814	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 517 (MAC Address18 Low Register)</b></p> <p>The MAC Address18 Low register holds the lower 32 bits of the 19th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address19 High</u></a>	0x818	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 518 (MAC Address19 High Register)</b></p> <p>The MAC Address19 High register holds the upper 16 bits of the 20th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address19 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address19 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address19 Low</u></a>	0x81c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 519 (MAC Address19 Low Register)</b></p> <p>The MAC Address19 Low register holds the lower 32 bits of the 20th 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#"><u>MAC Address20_High</u></a>	0x820	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 520 (MAC Address20 High Register)</b></p> <p>The MAC Address20 High register holds the upper 16 bits of the 21st 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address20 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address20 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address20_Low</u></a>	0x824	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 521 (MAC Address20 Low Register)</b></p> <p>The MAC Address20 Low register holds the lower 32 bits of the 21st 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address21_High</u></a>	0x828	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 522 (MAC Address21 High Register)</b></p> <p>The MAC Address21 High register holds the upper 16 bits of the 22nd 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address21 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address21 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address21_Low</u></a>	0x82c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 523 (MAC Address21 Low Register)</b></p> <p>The MAC Address21 Low register holds the lower 32 bits of the 22nd 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address22_High</u></a>	0x830	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 524 (MAC Address22 High Register)</b></p> <p>The MAC Address22 High register holds the upper 16 bits of the 23rd 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address22 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address22 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address22_Low</u></a>	0x834	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 525 (MAC Address22 Low Register)</b></p> <p>The MAC Address22 Low register holds the lower 32 bits of the 23rd 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#"><u>MAC Address23_High</u></a>	0x838	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 526 (MAC Address23 High Register)</b></p> <p>The MAC Address23 High register holds the upper 16 bits of the 24th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address23 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address23 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address23_Low</u></a>	0x83c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 527 (MAC Address23 Low Register)</b></p> <p>The MAC Address23 Low register holds the lower 32 bits of the 24th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address24_High</u></a>	0x840	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 528 (MAC Address24 High Register)</b></p> <p>The MAC Address24 High register holds the upper 16 bits of the 25th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address24 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address24 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address24_Low</u></a>	0x844	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 529 (MAC Address24 Low Register)</b></p> <p>The MAC Address24 Low register holds the lower 32 bits of the 25th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address25_High</u></a>	0x848	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 530 (MAC Address25 High Register)</b></p> <p>The MAC Address25 High register holds the upper 16 bits of the 6-byte 26th MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address25 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address25 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address25_Low</u></a>	0x84c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 531 (MAC Address25 Low Register)</b></p> <p>The MAC Address25 Low register holds the lower 32 bits of the 26th 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#"><u>MAC Address26 High</u></a>	0x850	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 532 (MAC Address26 High Register)</b></p> <p>The MAC Address26 High register holds the upper 16 bits of the 27th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address26 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address26 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address26 Low</u></a>	0x854	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 533 (MAC Address26 Low Register)</b></p> <p>The MAC Address26 Low register holds the lower 32 bits of the 27th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address27 High</u></a>	0x858	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 534 (MAC Address27 High Register)</b></p> <p>The MAC Address27 High register holds the upper 16 bits of the 28th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address27 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address27 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address27 Low</u></a>	0x85c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 535 (MAC Address27 Low Register)</b></p> <p>The MAC Address27 Low register holds the lower 32 bits of the 28th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address28 High</u></a>	0x860	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 536 (MAC Address28 High Register)</b></p> <p>The MAC Address28 High register holds the upper 16 bits of the 29th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address28 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address28 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address28 Low</u></a>	0x864	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 537 (MAC Address28 Low Register)</b></p> <p>The MAC Address28 Low register holds the lower 32 bits of the 29th 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#">MAC Address29 High</a>	0x868	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 538 (MAC Address29 High Register)</b></p> <p>The MAC Address29 High register holds the upper 16 bits of the 6-byte 30th MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address29 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address29 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address29 Low</a>	0x86c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 539 (MAC Address29 Low Register)</b></p> <p>The MAC Address29 Low register holds the lower 32 bits of the 30th 6-byte MAC address of the station.</p>
<a href="#">MAC Address30 High</a>	0x870	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 540 (MAC Address30 High Register)</b></p> <p>The MAC Address30 High register holds the upper 16 bits of the 31st 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address30 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address30 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address30 Low</a>	0x874	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 541 (MAC Address30 Low Register)</b></p> <p>The MAC Address30 Low register holds the lower 32 bits of the 31st 6-byte MAC address of the station.</p>
<a href="#">MAC Address31 High</a>	0x878	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 542 (MAC Address31 High Register)</b></p> <p>The MAC Address31 High register holds the upper 16 bits of the 32nd 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address31 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address31 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address31 Low</a>	0x87c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 543 (MAC Address31 Low Register)</b></p> <p>The MAC Address31 Low register holds the lower 32 bits of the 32nd 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#">MAC Address32 High</a>	0x880	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 544 (MAC Address32 High Register)</b></p> <p>The MAC Address32 High register holds the upper 16 bits of the 33rd 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address32 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address32 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address32 Low</a>	0x884	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 545 (MAC Address32 Low Register)</b></p> <p>The MAC Address32 Low register holds the lower 32 bits of the 33rd 6-byte MAC address of the station.</p>
<a href="#">MAC Address33 High</a>	0x888	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 546 (MAC Address33 High Register)</b></p> <p>The MAC Address33 High register holds the upper 16 bits of the 34th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address33 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address33 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address33 Low</a>	0x88c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 547 (MAC Address33 Low Register)</b></p> <p>The MAC Address33 Low register holds the lower 32 bits of the 34th 6-byte MAC address of the station.</p>
<a href="#">MAC Address34 High</a>	0x890	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 548 (MAC Address34 High Register)</b></p> <p>The MAC Address34 High register holds the upper 16 bits of the 35th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address34 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address34 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address34 Low</a>	0x894	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 549 (MAC Address34 Low Register)</b></p> <p>The MAC Address34 Low register holds the lower 32 bits of the 35th 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#">MAC Address35_High</a>	0x898	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 550 (MAC Address35 High Register)</b></p> <p>The MAC Address35 High register holds the upper 16 bits of the 36th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address35 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address35 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address35_Low</a>	0x89c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 551 (MAC Address35 Low Register)</b></p> <p>The MAC Address35 Low register holds the lower 32 bits of the 36th 6-byte MAC address of the station.</p>
<a href="#">MAC Address36_High</a>	0x8a0	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 552 (MAC Address36 High Register)</b></p> <p>The MAC Address36 High register holds the upper 16 bits of the 37th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address36 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address36 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address36_Low</a>	0x8a4	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 553 (MAC Address36 Low Register)</b></p> <p>The MAC Address36 Low register holds the lower 32 bits of the 34th 6-byte MAC address of the station.</p>
<a href="#">MAC Address37_High</a>	0x8a8	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 554 (MAC Address37 High Register)</b></p> <p>The MAC Address37 High register holds the upper 16 bits of the 38th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address37 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address37 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address37_Low</a>	0x8ac	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 555 (MAC Address37 Low Register)</b></p> <p>The MAC Address37 Low register holds the lower 32 bits of the 37th 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#">MAC Address38_High</a>	0x8b0	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 556 (MAC Address38 High Register)</b></p> <p>The MAC Address38 High register holds the upper 16 bits of the 39th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address38 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address38 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address38_Low</a>	0x8b4	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 557 (MAC Address38 Low Register)</b></p> <p>The MAC Address38 Low register holds the lower 32 bits of the 39th 6-byte MAC address of the station.</p>
<a href="#">MAC Address39_High</a>	0x8b8	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 558 (MAC Address39 High Register)</b></p> <p>The MAC Address39 High register holds the upper 16 bits of the 40th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address40 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address40 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address39_Low</a>	0x8bc	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 559 (MAC Address39 Low Register)</b></p> <p>The MAC Address39 Low register holds the lower 32 bits of the 40th 6-byte MAC address of the station.</p>
<a href="#">MAC Address40_High</a>	0x8c0	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 560 (MAC Address40 High Register)</b></p> <p>The MAC Address40 High register holds the upper 16 bits of the 41st 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address40 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address40 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address40_Low</a>	0x8c4	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 561 (MAC Address40 Low Register)</b></p> <p>The MAC Address40 Low register holds the lower 32 bits of the 41st 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#"><u>MAC Address41_High</u></a>	0x8c8	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 562 (MAC Address41 High Register)</b></p> <p>The MAC Address41 High register holds the upper 16 bits of the 42nd 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address41 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address41 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address41_Low</u></a>	0x8cc	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 563 (MAC Address41 Low Register)</b></p> <p>The MAC Address41 Low register holds the lower 32 bits of the 42nd 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address42_High</u></a>	0x8d0	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 564 (MAC Address42 High Register)</b></p> <p>The MAC Address42 High register holds the upper 16 bits of the 43rd 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address42 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address42 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address42_Low</u></a>	0x8d4	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 565 (MAC Address42 Low Register)</b></p> <p>The MAC Address42 Low register holds the lower 32 bits of the 43rd 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address43_High</u></a>	0x8d8	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 566 (MAC Address43 High Register)</b></p> <p>The MAC Address43 High register holds the upper 16 bits of the 44th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address43 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address43 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address43_Low</u></a>	0x8dc	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 567 (MAC Address43 Low Register)</b></p> <p>The MAC Address43 Low register holds the lower 32 bits of the 44th 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#"><u>MAC Address44 High</u></a>	0x8e0	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 568 (MAC Address44 High Register)</b></p> <p>The MAC Address44 High register holds the upper 16 bits of the 45th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address44 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address44 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address44 Low</u></a>	0x8e4	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 569 (MAC Address44 Low Register)</b></p> <p>The MAC Address44 Low register holds the lower 32 bits of the 45th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address45 High</u></a>	0x8e8	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 570 (MAC Address45 High Register)</b></p> <p>The MAC Address45 High register holds the upper 16 bits of the 46th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address45 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address45 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address45 Low</u></a>	0x8ec	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 571 (MAC Address45 Low Register)</b></p> <p>The MAC Address45 Low register holds the lower 32 bits of the 46th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address46 High</u></a>	0x8f0	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 572 (MAC Address46 High Register)</b></p> <p>The MAC Address46 High register holds the upper 16 bits of the 47th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address46 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address46 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address46 Low</u></a>	0x8f4	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 573 (MAC Address46 Low Register)</b></p> <p>The MAC Address46 Low register holds the lower 32 bits of the 47th 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#"><u>MAC Address47_High</u></a>	0x8f8	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 574 (MAC Address47 High Register)</b></p> <p>The MAC Address47 High register holds the upper 16 bits of the 48th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address47 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address47 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address47_Low</u></a>	0x8fc	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 575 (MAC Address47 Low Register)</b></p> <p>The MAC Address47 Low register holds the lower 32 bits of the 48th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address48_High</u></a>	0x900	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 576 (MAC Address48 High Register)</b></p> <p>The MAC Address48 High register holds the upper 16 bits of the 49th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address48 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address48 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address48_Low</u></a>	0x904	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 577 (MAC Address48 Low Register)</b></p> <p>The MAC Address48 Low register holds the lower 32 bits of the 49th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address49_High</u></a>	0x908	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 578 (MAC Address49 High Register)</b></p> <p>The MAC Address49 High register holds the upper 16 bits of the 50th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address49 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address49 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address49_Low</u></a>	0x90c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 579 (MAC Address49 Low Register)</b></p> <p>The MAC Address49 Low register holds the lower 32 bits of the 50th 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#">MAC Address50_High</a>	0x910	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 580 (MAC Address50 High Register)</b></p> <p>The MAC Address50 High register holds the upper 16 bits of the 51st 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address50 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address50 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address50_Low</a>	0x914	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 581 (MAC Address50 Low Register)</b></p> <p>The MAC Address50 Low register holds the lower 32 bits of the 51st 6-byte MAC address of the station.</p>
<a href="#">MAC Address51_High</a>	0x918	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 582 (MAC Address51 High Register)</b></p> <p>The MAC Address51 High register holds the upper 16 bits of the 52nd 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address51 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address51 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address51_Low</a>	0x91c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 583 (MAC Address51 Low Register)</b></p> <p>The MAC Address51 Low register holds the lower 32 bits of the 52nd 6-byte MAC address of the station.</p>
<a href="#">MAC Address52_High</a>	0x920	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 584 (MAC Address52 High Register)</b></p> <p>The MAC Address52 High register holds the upper 16 bits of the 53rd 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address52 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address52 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address52_Low</a>	0x924	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 585 (MAC Address52 Low Register)</b></p> <p>The MAC Address52 Low register holds the lower 32 bits of the 53rd 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#"><u>MAC Address53_High</u></a>	0x928	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 586 (MAC Address53 High Register)</b></p> <p>The MAC Address53 High register holds the upper 16 bits of the 54th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address53 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address53 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address53_Low</u></a>	0x92c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 587 (MAC Address53 Low Register)</b></p> <p>The MAC Address53 Low register holds the lower 32 bits of the 54th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address54_High</u></a>	0x930	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 588 (MAC Address54 High Register)</b></p> <p>The MAC Address54 High register holds the upper 16 bits of the 55th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address54 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address54 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address54_Low</u></a>	0x934	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 589 (MAC Address54 Low Register)</b></p> <p>The MAC Address54 Low register holds the lower 32 bits of the 55th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address55_High</u></a>	0x938	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 590 (MAC Address55 High Register)</b></p> <p>The MAC Address55 High register holds the upper 16 bits of the 56th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address55 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address55 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address55_Low</u></a>	0x93c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 591 (MAC Address55 Low Register)</b></p> <p>The MAC Address55 Low register holds the lower 32 bits of the 56th 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#"><u>MAC Address56 High</u></a>	0x940	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 592 (MAC Address56 High Register)</b></p> <p>The MAC Address56 High register holds the upper 16 bits of the 57th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address56 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address56 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address56 Low</u></a>	0x944	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 593 (MAC Address56 Low Register)</b></p> <p>The MAC Address56 Low register holds the lower 32 bits of the 57th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address57 High</u></a>	0x948	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 594 (MAC Address57 High Register)</b></p> <p>The MAC Address57 High register holds the upper 16 bits of the 58th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address57 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address57 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address57 Low</u></a>	0x94c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 595 (MAC Address57 Low Register)</b></p> <p>The MAC Address57 Low register holds the lower 32 bits of the 58th 6-byte MAC address of the station.</p>
<a href="#"><u>MAC Address58 High</u></a>	0x950	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 596 (MAC Address58 High Register)</b></p> <p>The MAC Address58 High register holds the upper 16 bits of the 59th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address58 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address58 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#"><u>MAC Address58 Low</u></a>	0x954	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 597 (MAC Address58 Low Register)</b></p> <p>The MAC Address58 Low register holds the lower 32 bits of the 59th 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#">MAC Address59_High</a>	0x958	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 598 (MAC Address59 High Register)</b></p> <p>The MAC Address59 High register holds the upper 16 bits of the 60th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address59 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address59 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address59_Low</a>	0x95c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 599 (MAC Address59 Low Register)</b></p> <p>The MAC Address59 Low register holds the lower 32 bits of the 60th 6-byte MAC address of the station.</p>
<a href="#">MAC Address60_High</a>	0x960	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 600 (MAC Address60 High Register)</b></p> <p>The MAC Address60 High register holds the upper 16 bits of the 61st 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address60 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address60 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address60_Low</a>	0x964	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 601 (MAC Address60 Low Register)</b></p> <p>The MAC Address60 Low register holds the lower 32 bits of the 61st 6-byte MAC address of the station.</p>
<a href="#">MAC Address61_High</a>	0x968	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 602 (MAC Address61 High Register)</b></p> <p>The MAC Address61 High register holds the upper 16 bits of the 62nd 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address61 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address61 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address61_Low</a>	0x96c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 603 (MAC Address61 Low Register)</b></p> <p>The MAC Address61 Low register holds the lower 32 bits of the 62nd 6-byte MAC address of the station.</p>

Register	Offset	Size	Memory Access	Description
<a href="#">MAC Address62 High</a>	0x970	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 604 (MAC Address62 High Register)</b></p> <p>The MAC Address62 High register holds the upper 16 bits of the 63rd 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address62 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address62 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address62 Low</a>	0x974	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 605 (MAC Address62 Low Register)</b></p> <p>The MAC Address62 Low register holds the lower 32 bits of the 63rd 6-byte MAC address of the station.</p>
<a href="#">MAC Address63 High</a>	0x978	32 bits	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p><b>Register 606 (MAC Address63 High Register)</b></p> <p>The MAC Address63 High register holds the upper 16 bits of the 64th 6-byte MAC address of the station.</p> <p>If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address63 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address63 Low Register must be performed after at least four clock cycles in the destination clock domain.</p>
<a href="#">MAC Address63 Low</a>	0x97c	32 bits	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p><b>Register 607 (MAC Address63 Low Register)</b></p> <p>The MAC Address63 Low register holds the lower 32 bits of the 64th 6-byte MAC address of the station.</p>

The address block DMA contains 23 Registers : Bus\_Mode to HW\_Feature

<a href="#">Bus Mode</a>	0x1000	32 bits	R/W	<p><b>Value After Reset:</b> 0x20101</p> <p><b>Register 0 (Bus Mode Register)</b></p> <p>The Bus Mode register establishes the bus operating modes for the DMA.</p>
<a href="#">Transmit Poll Demand</a>	0x1004	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 1 (Transmit Poll Demand Register)</b></p> <p>The Transmit Poll Demand register enables the Tx DMA to check whether or not the DMA owns the current descriptor. The Transmit Poll Demand command is given to wake up the Tx DMA if it is in the Suspend mode. The Tx DMA can go into the Suspend mode because of an Underflow error in a transmitted frame or the unavailability of descriptors owned by it. You can give this command anytime and the Tx DMA resets this command when it again starts fetching the current descriptor from host memory. When this register is read, it</p>

Register	Offset	Size	Memory Access	Description
				always returns zero.
<a href="#"><u>Receive_Poll_Demand</u></a>	0x1008	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 2 (Receive Poll Demand Register)</b></p> <p>The Receive Poll Demand register enables the receive DMA to check for new descriptors. This command is used to wake up the Rx DMA from the SUSPEND state. The RxDMA can go into the SUSPEND state only because of the unavailability of descriptors it owns. When this register is read, it always returns zero.</p>
<a href="#"><u>Receive_Descriptor_List_Address</u></a>	0x100c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 3 (Receive Descriptor List Address Register)</b></p> <p>The Receive Descriptor List Address register points to the start of the Receive Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given.</p> <p>You can write to this register only when Rx DMA has stopped, that is, Bit 1 (SR) is set to zero in Register 6 (Operation Mode Register). When stopped, this register can be written with a new descriptor list address. When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.</p> <p>If this register is not changed when the SR bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.</p>
<a href="#"><u>Transmit_Descriptor_List_Address</u></a>	0x1010	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 4 (Transmit Descriptor List Address Register)</b></p> <p>The Transmit Descriptor List Address register points to the start of the Transmit Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.</p> <p>You can write to this register only when the Tx DMA has stopped, that is, Bit 13 (ST) is set to zero in Register 6 (Operation Mode Register). When stopped, this register can be written with a new descriptor list address. When you set the ST bit to 1, the DMA takes the newly programmed descriptor base address.</p> <p>If this register is not changed when the ST bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.</p>
<a href="#"><u>Status</u></a>	0x1014	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 5 (Status Register)</b></p> <p>The Status register contains all status bits that the DMA reports to the host. The Software driver reads this register during an interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted. The bits of this register are not cleared when read. Writing 1'b1 to (unreserved) Bits[16:0] of this register clears these bits and writing 1'b0 has no effect. Each field</p>

Register	Offset	Size	Memory Access	Description
				(Bits[16:0]) can be masked by masking the appropriate bit in Register 7 (Interrupt Enable Register).
<a href="#">Operation Mode</a>	0x1018	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 6 (Operation Mode Register)</b></p> <p>The Operation Mode register establishes the Transmit and Receive operating modes and commands. This register should be the last CSR to be written as part of the DMA initialization. This register is also present in the GMAC-MTL configuration with unused and reserved bits 24, 13, 2, and 1.</p>
<a href="#">Interrupt Enable</a>	0x101c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 7 (Interrupt Enable Register)</b></p> <p>The Interrupt Enable register enables the interrupts reported by Register 5 (Status Register). Setting a bit to 1'b1 enables a corresponding interrupt. After a hardware or software reset, all interrupts are disabled.</p>
<a href="#">Missed Frame And Buffer Overflow Counter</a>	0x1020	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 8 (Missed Frame and Buffer Overflow Counter Register)</b></p> <p>The DMA maintains two counters to track the number of frames missed during reception. This register reports the current value of the counter. The counter is used for diagnostic purposes. Bits[15:0] indicate missed frames because of the host buffer being unavailable. Bits[27:17] indicate missed frames because of buffer overflow conditions (MTL and MAC) and runt frames (good frames of less than 64 bytes) dropped by the MTL.</p>
<a href="#">Receive Interrupt Watchdog Timer</a>	0x1024	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 9 (Receive Interrupt Watchdog Timer Register)</b></p> <p>This register, when written with non-zero value, enables the watchdog timer for the Receive Interrupt (Bit 6) of Register 5 (Status Register)</p>
<a href="#">AHB or AXI Status</a>	0x102c	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 11 (AHB or AXI Status Register)</b></p> <p>This register provides the active status of the AHB master interface or AXI interface's read and write channels. This register is present and valid only in the GMAC-AHB and GMAC-AXI configurations. This register is useful for debugging purposes. In addition, this register is valid only in the Channel 0 DMA when multiple channels are present in the AV mode.</p>
<a href="#">Current Host Transmit Descriptor</a>	0x1048	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 18 (Current Host Transmit Descriptor Register)</b></p> <p>The Current Host Transmit Descriptor register points to the start address of the current Transmit Descriptor read by the DMA.</p>
<a href="#">Current Host Receive Descriptor</a>	0x104c	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p><b>Register 19 (Current Host Receive Descriptor Register)</b></p> <p>The Current Host Receive Descriptor register points to the start address of the current Receive Descriptor read by the DMA.</p>
<a href="#">Current Host Trans</a>	0x1050	32 bits	R	<b>Value After Reset:</b> 0x0

Register	Offset	Size	Memory Access	Description
<a href="#">mit_Buffer_Address</a>				<b>Register 20 (Current Host Transmit Buffer Address Register)</b> The Current Host Transmit Buffer Address register points to the current Transmit Buffer Address being read by the DMA.
<a href="#">Current_Host_Receive_Buffer_Address</a>	0x1054	32 bits	R	<b>Value After Reset:</b> 0x0 <b>Register 21 (Current Host Receive Buffer Address Register)</b> The Current Host Receive Buffer Address register points to the current Receive Buffer address being read by the DMA.
<a href="#">HW_Feature</a>	0x1058	32 bits	R	<b>Value After Reset:</b> 0x90c0fb9 <b>Register 22 (HW Feature Register)</b> This register indicates the presence of the optional features or functions of the DWC_gmac. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks. Note: All bits are set or reset as per the selection of features during the DWC_gmac configuration.

### 9.7.2 Register Descriptions

Following is a description of the individual registers of design DWC\_gmac

#### 9.7.2.1 GMA Register Description

##### 9.7.2.1.1 Register 0 (MAC Configuration Register)

**Size:** 32 bits

**Offset:** 0x0

**Memory Access:** R/W

**Value After Reset:** 0x8800

31	30: 28	27	26	25	24	23	22	21	20	19: 17	16	15	14	13	12	11	10	9	8	7	6: 5	4	3	2	1:0
Reserve_31	SA RC	TWO KPE	SFT ERR	C S T	T C	W D	J D	B E	J E	IFG	DC RS	P S	F E S	D O	L M	D M	I P C	D R	L U D	A C S	B L	D C	T E	R E	PRE LEN

The MAC Configuration register establishes receive and transmit operating modes.

Bits	Name	Memory Access	Description
31	Reserved_31	R	<b>Value After Reset:</b> 0x0 Reserved
30:28	SARC	R/W	<b>Value After Reset:</b> 0x0 Source Address Insertion or Replacement Control This field controls the source address insertion or replacement for all transmitted frames. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits [29:28]: * 2'b0x: The input signals mti_sa_ctrl_i and ati_sa_ctrl_i control the SA field generation. * 2'b10: If Bit 30 is set to 0, the MAC inserts the content of the MAC Address 0 registers (registers 16 and 17) in the SA field of all transmitted frames.

Bits	Name	Memory Access	Description
			<p>If Bit 30 is set to 1 and the Enable MAC Address Register 1 option is selected during core configuration, the MAC inserts the content of the MAC Address 1 registers (registers 18 and 19) in the SA field of all transmitted frames.</p> <p>* 2'b11:</p> <p>If Bit 30 is set to 0, the MAC replaces the content of the MAC Address 0 registers (registers 16 and 17) in the SA field of all transmitted frames.</p> <p>If Bit 30 is set to 1 and the Enable MAC Address Register 1 option is selected during core configuration, the MAC replaces the content of the MAC Address 1 registers (registers 18 and 19) in the SA field of all transmitted frames.</p> <p>Note:</p> <p>Changes to this field take effect only on the start of a frame. If you write this register field when a frame is being transmitted, only the subsequent frame can use the updated value, that is, the current frame does not use the updated value.</p> <p>These bits are reserved and RO when the Enable SA, VLAN, and CRC Insertion on TX feature is not selected during core configuration.</p>
27	TWOKPE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>IEEE 802.3as Support for 2K Packets</p> <p>When set, the MAC considers all frames, with up to 2,000 bytes length, as normal packets. When Bit 20 (JE) is not set, the MAC considers all received frames of size more than 2K bytes as Giant frames. When this bit is reset and Bit 20 (JE) is not set, the MAC considers all received frames of size more than 1,518 bytes (1,522 bytes for tagged) as Giant frames. When Bit 20 is set, setting this bit has no effect on Giant Frame status.</p>
26	SFTRR	R	<p><b>Value After Reset:</b> 0x0</p> <p>SMII Force Transmit Error</p> <p>When set, this bit indicates to the PHY to force a transmit error in the SMII frame being transmitted. This bit is reserved if the SMII PHY port is not selected during core configuration.</p>
25	CST	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>CRC Stripping for Type Frames</p> <p>When this bit is set, the last 4 bytes (FCS) of all frames of Ether type (Length/Type field greater than or equal to 1,536) are stripped and dropped before forwarding the frame to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver. This function is valid when Type 2 Checksum Offload Engine is enabled.</p>
24	TC	R	<p><b>Value After Reset:</b> 0x0</p> <p>Transmit Configuration in RGMII, SGMII, or SMII</p> <p>When set, this bit enables the transmission of duplex mode, link speed, and link up or down information to the PHY in the RGMII, SMII, or SGMII port. When this bit is reset, no such information is driven to the PHY. This bit is reserved (and RO) if the RGMII, SMII, or SGMII PHY port is not selected during core configuration.</p>
23	WD	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Watchdog Disable</p> <p>When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive frames of up to 16,384 bytes. When this bit is reset, the MAC does not allow a receive frame which more than 2,048 bytes (10,240 if JE is set high) or the value programmed in Register 55 (Watchdog Timeout Register).</p> <p>The MAC cuts off any bytes received after the watchdog limit number of bytes.</p>
22	JD	R/W	<b>Value After Reset:</b> 0x0

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
			<p><b>Jabber Disable</b>  When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer frames of up to 16,384 bytes. When this bit is reset, the MAC cuts off the transmitter if the application sends out more than 2,048 bytes of data (10,240 if JE is set high) during transmission.</p>
21	BE	R	<p><b>Value After Reset:</b> 0x0  <b>Frame Burst Enable</b>  When this bit is set, the MAC allows frame bursting during transmission in the GMII half-duplex mode. This bit is reserved (and RO) in the 10/100 Mbps only or full-duplex-only configurations.</p>
20	JE	R/W	<p><b>Value After Reset:</b> 0x0  <b>Jumbo Frame Enable</b>  When this bit is set, the MAC allows Jumbo frames of 9,018 bytes (9,022 bytes for VLAN tagged frames) without reporting a giant frame error in the receive frame status.</p>
19:17	IFG	R/W	<p><b>Value After Reset:</b> 0x0  <b>Inter-Frame Gap</b>  These bits control the minimum IFG between frames during transmission.  000: 96 bit times  001: 88 bit times  010: 80 bit times  ...  111: 40 bit times  In the half-duplex mode, the minimum IFG can be configured only for 64 bit times (IFG = 100). Lower values are not considered. In the 1000-Mbps mode, the minimum IFG supported is 64 bit times (and above) in the GMAC-CORE configuration and 80 bit times (and above) in other configurations.</p>
16	DCRS	R	<p><b>Value After Reset:</b> 0x0  <b>Disable Carrier Sense During Transmission</b>  When set high, this bit makes the MAC transmitter ignore the (G)MII CRS signal during frame transmission in the half-duplex mode. This request results in no errors generated because of Loss of Carrier or No Carrier during such transmission. When this bit is low, the MAC transmitter generates such errors because of Carrier Sense and can even abort the transmissions.  This bit is reserved (and RO) in the full-duplex-only configurations.</p>
15	PS	R	<p><b>Value After Reset:</b> 0x1  <b>Port Select</b>  This bit selects the Ethernet line speed:  0: For 1000 Mbps operations  1: For 10 or 100 Mbps operations  In 10 or 100 Mbps operations, this bit, along with FES bit, selects the exact line speed. In the 10 or 100 Mbps-only (always 1) or 1000 Mbps-only (always 0) configurations, this bit is read-only with the appropriate value. In default 10, 100, or 1000 Mbps configuration, this bit is R_W. The mac_portselect_o signal reflects the value of this bit.</p>
14	FES	R	<p><b>Value After Reset:</b> 0x0  <b>Speed</b>  This bit selects the speed in the MII, RMII, SMII, RGMII, SGMII, or RevMII interface  0: 10 Mbps  1: 100 Mbps</p>

Bits	Name	Memory Access	Description
			This bit is reserved (RO) by default and is enabled only when the parameter SPEED_SELECT = Enabled. This bit generates link speed encoding when Bit 24 (TC) is set in the RGMII, SMI, or SGMI mode. This bit is always enabled for RGMII, SGMI, SMI, or RevMII interface. In configurations with RGMII, SGMI, SMI, or RevMII interface, this bit is driven as an output signal (mac_speed_o[0]) to reflect the value of this bit in the mac_speed_o signal. In configurations with RMII, MII, or GMII interface, you can optionally drive this bit as an output signal (mac_speed_o[0]) to reflect its value in the mac_speed_o signal.
13	DO	R	<p><b>Value After Reset:</b> 0x0</p> <p>Disable Receive Own</p> <p>When this bit is set, the MAC disables the reception of frames when the phy_txen_o is asserted in the half-duplex mode. When this bit is reset, the MAC receives all packets that are given by the PHY while transmitting. This bit is not applicable if the MAC is operating in the full-duplex mode. This bit is reserved (RO with default value) if the MAC is configured for the full-duplex-only operation.</p>
12	LM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Loopback Mode</p> <p>When this bit is set, the MAC operates in the loopback mode at GMII or MII. The (G)MII Receive clock input (clk_rx_i) is required for the loopback to work properly, because the Transmit clock is not looped-back internally.</p>
11	DM	R	<p><b>Value After Reset:</b> 0x1</p> <p>Duplex Mode</p> <p>When this bit is set, the MAC operates in the full-duplex mode where it can transmit and receive simultaneously. This bit is RO with default value of 1'b1 in the full-duplex-only configuration.</p>
10	IPC	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Checksum Offload</p> <p>When this bit is set, the MAC calculates the 16-bit one's complement of the one's complement sum of all received Ethernet frame payloads. It also checks whether the IPv4 Header checksum (assumed to be bytes 2526 or 2930 (VLAN-tagged) of the received Ethernet frame) is correct for the received frame and gives the status in the receive status word. The MAC also appends the 16-bit checksum calculated for the IP header datagram payload (bytes after the IPv4 header) and appends it to the Ethernet frame transferred to the application (when Type 2 COE is deselected).</p> <p>When this bit is reset, this function is disabled.</p> <p>When Type 2 COE is selected, this bit, when set, enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled and the corresponding PCE and IP HCE status bits are always cleared.</p> <p>If the IP Checksum Offload feature is not enabled during core configuration, this bit is reserved (RO with default value).</p>
9	DR	R	<p><b>Value After Reset:</b> 0x0</p> <p>Disable Retry</p> <p>When this bit is set, the MAC attempts only one transmission. When a collision occurs on the GMII or MII interface, the MAC ignores the current frame transmission and reports a Frame Abort with excessive collision error in the transmit frame status.</p> <p>When this bit is reset, the MAC attempts retries based on the settings of the BL field (Bits [6:5]). This bit is applicable only in the half-duplex mode and is reserved (RO with default value) in the full-duplex-only configuration.</p>

Bits	Name	Memory Access	Description
8	LUD	R	<p><b>Value After Reset:</b> 0x0</p> <p>Link Up or Down</p> <p>This bit indicates whether the link is up or down during the transmission of configuration in the RGMII, SGMII, or SMII interface:</p> <p>0: Link Down 1: Link Up</p> <p>This bit is reserved (RO with default value) and is enabled when the RGMII, SGMII, or SMII interface is enabled during core configuration.</p>
7	ACS	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Automatic Pad or CRC Stripping</p> <p>When this bit is set, the MAC strips the Pad or FCS field on the incoming frames only if the value of the length field is less than 1,536 bytes. All received frames with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field.</p> <p>When this bit is reset, the MAC passes all incoming frames, without modifying them, to the Host.</p>
6:5	BL	R	<p><b>Value After Reset:</b> 0x0</p> <p>Back-Off Limit</p> <p>The Back-Off limit determines the random integer number (<math>r</math>) of slot time delays (4,096 bit times for 1000 Mbps and 512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. This bit is applicable only in the half-duplex mode and is reserved (RO) in the full-duplex-only configuration.</p> <p>00: <math>k = \min(n, 10)</math> 01: <math>k = \min(n, 8)</math> 10: <math>k = \min(n, 4)</math> 11: <math>k = \min(n, 1)</math></p> <p>where <math>n</math> = retransmission attempt. The random integer <math>r</math> takes the value in the range <math>0 \leq r &lt; k</math>th power of 2</p>
4	DC	R	<p><b>Value After Reset:</b> 0x0</p> <p>Deferral Check</p> <p>When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Frame Abort status, along with the excessive deferral error bit set in the transmit frame status, when the transmit state machine is deferred for more than 24,288 bit times in the 10 or 100 Mbps mode.</p> <p>If the MAC is configured for 1000 Mbps operation or if the Jumbo frame mode is enabled in the 10 or 100 Mbps mode, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on GMII or MII.</p> <p>The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and then the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0 and it is restarted. When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. This bit is applicable only in the half-duplex mode and is reserved (RO) in the full-duplex-only configuration.</p>
3	TE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Transmitter Enable</p>

Bits	Name	Memory Access	Description
			When this bit is set, the transmit state machine of the MAC is enabled for transmission on the GMII or MII. When this bit is reset, the MAC transmit state machine is disabled after the completion of the transmission of the current frame, and does not transmit any further frames.
2	RE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Receiver Enable</p> <p>When this bit is set, the receiver state machine of the MAC is enabled for receiving frames from the GMII or MII. When this bit is reset, the MAC receive state machine is disabled after the completion of the reception of the current frame, and does not receive any further frames from the GMII or MII.</p>
1:0	PRELEN	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Preamble Length for Transmit Frames</p> <p>These bits control the number of preamble bytes that are added to the beginning of every Transmit frame. The preamble reduction occurs only when the MAC is operating in the full-duplex mode.</p> <p>2'b00: 7 bytes of preamble      2'b01: 5 byte of preamble      2'b10: 3 bytes of preamble      2'b11: reserved</p>

### 9.7.2.1.2 Register 1 (MAC Frame Filter)

**Size:** 32 bits

**Offset:** 0x4

**Memory Access:** R/W

**Value After Reset:** 0x0

31	30:22	21	20	19:17	16	15:11	10	9	8	7: 6	5	4	3	2	1	0
R A	Reserved_30_22	DNT U	IPF E	Reserved_19_17	VTF E	Reserved_15_11	HP F	SA F	SAI F	PC F	DB F	P M	DAI F	HM C	HU C	P R

The MAC Frame Filter register contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering. The second level of filtering is performed on the incoming frame, based on other controls such as Pass Bad Frames and Pass Control Frames.

Bits	Name	Memory Access	Description
31	RA	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Receive All</p> <p>When this bit is set, the MAC Receiver module passes all received frames, irrespective of whether they pass the address filter or not, to the Application. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bits in the Receive Status Word.</p> <p>When this bit is reset, the Receiver module passes only those frames to the Application that pass the SA or DA address filter.</p>
30:22	Reserved_30_22	R	<p><b>Value After Reset:</b> 0x0</p> <p>Reserved</p>
21	DNTU	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Drop non-TCP/UDP over IP Frames</p>

Bits	Name	Memory Access	Description
			<p>When set, this bit enables the MAC to drop the non-TCP or UDP over IP frames. The MAC forward only those frames that are processed by the Layer 4 filter.</p> <p>When reset, this bit enables the MAC to forward all non-TCP or UDP over IP frames.</p> <p>If the Layer 3 and Layer 4 Filtering feature is not selected during core configuration, this bit is reserved (RO with default value).</p>
20	IPFE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Layer 3 and Layer 4 Filter Enable</p> <p>When set, this bit enables the MAC to drop frames that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect.</p> <p>When reset, the MAC forwards all frames irrespective of the match status of the Layer 3 and Layer 4 fields.</p> <p>If the Layer 3 and Layer 4 Filtering feature is not selected during core configuration, this bit is reserved (RO with default value).</p>
19:17	Reserved_19_17	R	<p><b>Value After Reset:</b> 0x0</p> <p>Reserved</p>
16	VTFE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>VLAN Tag Filter Enable</p> <p>When set, this bit enables the MAC to drop VLAN tagged frames that do not match the VLAN Tag comparison.</p> <p>When reset, the MAC forwards all frames irrespective of the match status of the VLAN Tag.</p>
15:11	Reserved_15_11	R	<p><b>Value After Reset:</b> 0x0</p> <p>Reserved</p>
10	HPF	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Hash or Perfect Filter</p> <p>When this bit is set, it configures the address filter to pass a frame if it matches either the perfect filtering or the hash filtering as set by the HMC or HUC bits.</p> <p>When this bit is low and the HUC or HMC bit is set, the frame is passed only if it matches the Hash filter. This bit is reserved (and RO) if the Hash filter is not selected during core configuration.</p>
9	SAF	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Source Address Filter Enable</p> <p>When this bit is set, the MAC compares the SA field of the received frames with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the frame. When this bit is reset, the MAC forwards the received frame to the application with updated SAF bit of the Rx Status depending on the SA address comparison.</p> <p>Note: According to the IEEE specification, Bit 47 of the SA is reserved and set to 0. However, in DWC_gmac, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.</p>
8	SAIF	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>SA Inverse Filtering</p> <p>When this bit is set, the Address Check block operates in inverse filtering mode for the SA address comparison. The frames whose SA matches the SA registers are marked as failing the SA Address filter.</p> <p>When this bit is reset, frames whose SA does not match the SA registers are</p>

Bits	Name	Memory Access	Description
			marked as failing the SA Address filter.
7:6	PCF	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Pass Control Frames</b></p> <p>These bits control the forwarding of all control frames (including unicast and multicast PAUSE frames).</p> <p>00: MAC filters all control frames from reaching the application.</p> <p>01: MAC forwards all control frames except PAUSE control frames to application even if they fail the Address filter.</p> <p>10: MAC forwards all control frames to application even if they fail the Address Filter.</p> <p>11: MAC forwards control frames that pass the Address Filter.</p> <p>The following conditions should be true for the PAUSE control frames processing:</p> <p>Condition 1: The MAC is in the full-duplex mode and flow control is enabled by setting Bit 2 (RFE) of Register 6 (Flow Control Register) to 1.</p> <p>Condition 2: The destination address (DA) of the received frame matches the special multicast address or the MAC Address 0 when Bit 3 (UP) of the Register 6 (Flow Control Register) is set.</p> <p>Condition 3: The Type field of the received frame is 0x8808 and the OPCODE field is 0x0001.</p> <p><b>Note:</b></p> <p>This field should be set to 01 only when the Condition 1 is true, that is, the MAC is programmed to operate in the full-duplex mode and the RFE bit is enabled.</p> <p>Otherwise, the PAUSE frame filtering may be inconsistent. When Condition 1 is false, the PAUSE frames are considered as generic control frames. Therefore, to pass all control frames (including PAUSE control frames) when the full-duplex mode and flow control is not enabled, you should set the PCF field to 10 or 11 (as required by the application).</p>
5	DBF	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Disable Broadcast Frames</b></p> <p>When this bit is set, the AFM module filters all incoming broadcast frames. In addition, it overrides all other filter settings.</p> <p>When this bit is reset, the AFM module passes all received broadcast frames.</p>
4	PM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Pass All Multicast</b></p> <p>When set, this bit indicates that all received frames with a multicast destination address (first bit in the destination address field is '1') are passed.</p> <p>When reset, filtering of multicast frame depends on HMC bit.</p>
3	DAIF	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>DA Inverse Filtering</b></p> <p>When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast frames.</p> <p>When reset, normal filtering of frames is performed.</p>
2	HMC	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>Hash Multicast</b></p> <p>When set, MAC performs destination address filtering of received multicast frames according to the hash table.</p> <p>When reset, the MAC performs a perfect destination address filtering for multicast frames, that is, it compares the DA field with the values programmed in DA registers. If Hash Filter is not selected during core configuration, this bit is reserved</p>

Bits	Name	Memory Access	Description
			(and RO).
1	HUC	R/W	<p><b>Value After Reset:</b> 0x0 Hash Unicast When set, MAC performs destination address filtering of unicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for unicast frames, that is, it compares the DA field with the values programmed in DA registers. If Hash Filter is not selected during core configuration, this bit is reserved (and RO).</p>
0	PR	R/W	<p><b>Value After Reset:</b> 0x0 Promiscuous Mode When this bit is set, the Address Filter module passes all incoming frames regardless of its destination or source address. The SA or DA Filter Fails status bits of the Receive Status Word are always cleared when PR is set.</p>

#### 9.7.2.1.3 Register 4 (GMII Address Register)

**Size:** 32 bits

**Offset:** 0x10

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:11	10:6	5:2	1	0
Reserved_31_16	PA	GR	CR	GW	GB

The GMII Address register controls the management cycles to the external PHY through the management interface.

Note: This register is present for all PHY interface when you select the Station Management (MDIO) feature in coreConsultant.

Bits	Name	Memory Access	Description
31:16	Reserved_31_16	R	<p><b>Value After Reset:</b> 0x0 Reserved</p>
15:11	PA	R/W	<p><b>Value After Reset:</b> 0x0 Physical Layer Address This field indicates which of the 32 possible PHY devices are being accessed. For RevMII, this field gives the PHY Address of the RevMII module.</p>
10:6	GR	R/W	<p><b>Value After Reset:</b> 0x0 GMII Register These bits select the desired GMII register in the selected PHY device. For RevMII, these bits select the desired CSR register in the RevMII Registers set.</p>
5:2	CR	R/W	<p><b>Value After Reset:</b> 0x0 CSR Clock Range The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency used in your design. The suggested range of CSR clock frequency applicable for each value (when Bit[5] = 0) ensures that the MDC clock is approximately between the frequency range 1.0 MHz - 2.5 MHz. 0000: The frequency of the CSR clock is 60-100 MHz and the MDC clock is CSR clock/42.</p>

Bits	Name	Memory Access	Description
			<p>0001: The frequency of the CSR clock is 100-150 MHz and the MDC clock is CSR clock/62.</p> <p>0010: The frequency of the CSR clock is 20-35 MHz and the MDC clock is CSR clock/16.</p> <p>0011: The frequency of the CSR clock is 35-60 MHz and the MDC clock is CSR clock/26.</p> <p>0100: The frequency of the CSR clock is 150-250 MHz and the MDC clock is CSR clock/102.</p> <p>0100: The frequency of the CSR clock is 250-300 MHz and the MDC clock is CSR clock/124.</p> <p>0110 and 0111: Reserved</p> <p>When Bit 5 is set, you can achieve MDC clock of frequency higher than the IEEE 802.3 specified frequency limit of 2.5 MHz and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits as 1010, then the resultant MDC clock is of 12.5 MHz which is outside the limit of IEEE 802.3 specified range.</p> <p>Program the following values only if the interfacing chips support faster MDC clocks:</p> <ul style="list-style-type: none"> <li>1000: CSR clock/4</li> <li>1001: CSR clock/6</li> <li>1010: CSR clock/8</li> <li>1011: CSR clock/10</li> <li>1100: CSR clock/12</li> <li>1101: CSR clock/14</li> <li>1110: CSR clock/16</li> <li>1111: CSR clock/18</li> </ul> <p>These bits are not used for accessing RevMII. These bits are read-only if the RevMII interface is selected as single PHY interface.</p>
1	GW	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>GMII Write</b></p> <p>When set, this bit indicates to the PHY or RevMII that this is a Write operation using the GMII Data register. If this bit is not set, it indicates that this is a Read operation, that is, placing the data in the GMII Data register.</p>
0	GB	R/W	<p><b>Value After Reset:</b> 0x0</p> <p><b>GMII Busy</b></p> <p>This bit should read logic 0 before writing to Register 4 and Register 5. During a PHY or RevMII register access, the software sets this bit to 1'b1 to indicate that a Read or Write access is in progress.</p> <p>Register 5 is invalid until this bit is cleared by the MAC. Therefore, Register 5 (GMII Data) should be kept valid until the MAC clears this bit during a PHY Write operation. Similarly for a read operation, the contents of Register 5 are not valid until this bit is cleared.</p> <p>The subsequent read or write operation should happen only after the previous operation is complete. Because there is no acknowledgment from the PHY to MAC after a read or write operation is completed, there is no change in the functionality of this bit even when the PHY is not present.</p>

#### 9.7.2.1.4 Register 5 (GMII Data Register)

**Size:** 32 bits

**Offset:** 0x14

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:0
Reserved_31_16	GD

The GMII Data register stores Write data to be written to the PHY register located at the address specified in Register 4 (GMII Address Register). This register also stores the Read data from the PHY register located at the address specified by Register 4.

Note: This register is present for all PHY interface when you select the Station Management (MDIO) feature in coreConsultant.

Bits	Name	Memory Access	Description
31:16	Reserved_31_16	R	<b>Value After Reset:</b> 0x0 Reserved
15:0	GD	R/W	<b>Value After Reset:</b> 0x0 GMII Data This field contains the 16-bit data value read from the PHY or RevMII after a Management Read operation or the 16-bit data value to be written to the PHY or RevMII before a Management Write operation.

#### 9.7.2.1.5 Register 6 (Flow Control Register)

**Size:** 32 bits

**Offset:** 0x18

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:8	7	6	5:4	3	2	1	0
PT	Reserved_15_8	DZPQ	Reserved_6	PLT	UP	RFE	TFE	FCA_BPA

The Flow Control register controls the generation and reception of the Control (Pause Command) frames by the MAC's Flow control module. A Write to a register with the Busy bit set to '1' triggers the Flow Control block to generate a Pause Control frame. The fields of the control frame are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control frame. The Busy bit remains set until the control frame is transferred onto the cable. The Host must make sure that the Busy bit is cleared before writing to the register.

Bits	Name	Memory Access	Description
31:16	PT	R/W	<b>Value After Reset:</b> 0x0 Pause Time This field holds the value to be used in the Pause Time field in the transmit control frame. If the Pause Time bits is configured to be double-synchronized to the (G)MII clock domain, then consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.
15:8	Reserved_15_8	R	<b>Value After Reset:</b> 0x0 Reserved
7	DZPQ	R/W	<b>Value After Reset:</b> 0x0 Disable Zero-Quanta Pause When this bit is set, it disables the automatic generation of the Zero-Quanta Pause Control frames on the de-assertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i/mti_flowctrl_i). When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled.

Bits	Name	Memory Access	Description
6	Reserved_6	R	<p><b>Value After Reset:</b> 0x0</p> <p>Reserved</p>
5:4	PLT	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Pause Low Threshold</p> <p>This field configures the threshold of the PAUSE timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of PAUSE Frame.</p> <p>The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot-times), and PLT = 01, then a second PAUSE frame is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256 - 28) slot times after the first PAUSE frame is transmitted.</p> <p>The following list provides the threshold values for different values:</p> <ul style="list-style-type: none"> <li>00: The threshold is Pause time minus 4 slot times (PT - 4 slot times).</li> <li>01: The threshold is Pause time minus 28 slot times (PT - 28 slot times).</li> <li>10: The threshold is Pause time minus 144 slot times (PT - 144 slot times).</li> <li>11: The threshold is Pause time minus 256 slot times (PT - 256 slot times).</li> </ul> <p>The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII or MII interface.</p>
3	UP	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Unicast Pause Frame Detect</p> <p>A pause frame is processed when it has the unique multicast address specified in the IEEE Std 802.3. When this bit is set, the MAC can also detect Pause frames with unicast address of the station. This unicast address should be as specified in the MAC Address0 High Register and MAC Address0 Low Register.</p> <p>When this bit is reset, the MAC only detects Pause frames with unique multicast address.</p> <p>Note: The MAC does not process a Pause frame if the multicast address of received frame is different from the unique multicast address.</p>
2	RFE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Receive Flow Control Enable</p> <p>When this bit is set, the MAC decodes the received Pause frame and disables its transmitter for a specified (Pause) time. When this bit is reset, the decode function of the Pause frame is disabled.</p>
1	TFE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Transmit Flow Control Enable</p> <p>In the full-duplex mode, when this bit is set, the MAC enables the flow control operation to transmit Pause frames. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause frames.</p> <p>In half-duplex mode, when this bit is set, the MAC enables the back-pressure operation. When this bit is reset, the back-pressure feature is disabled.</p>
0	FCA_BPA	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Flow Control Busy or Backpressure Activate</p> <p>This bit initiates a Pause Control frame in the full-duplex mode and activates the backpressure function in the half-duplex mode if the TFE bit is set.</p> <p>In the full-duplex mode, this bit should be read as 1'b0 before writing to the Flow Control register. To initiate a Pause control frame, the Application must set this bit to 1'b1. During a transfer of the Control Frame, this bit continues to be set to signify that a frame transmission is in progress. After the completion of Pause control frame transmission, the MAC resets this bit to 1'b0. The Flow Control register should not be</p>

Bits	Name	Memory Access	Description
			written to until this bit is cleared. In the half-duplex mode, when this bit is set (and TFE is set), then backpressure is asserted by the MAC. During backpressure, when the MAC receives a new frame, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically ORed with the mti_flowctrl_i input signal for the backpressure function. When the MAC is configured for the full-duplex mode, the BPA is automatically disabled.

### 9.7.2.1.6 Register 7 (VLAN Tag Register)

**Size:** 32 bits

**Offset:** 0x1c

**Memory Access:** R/W

**Value After Reset:** 0x0

31:20	19	18	17	16	15:0
Reserved_31_20	VTHM	ESVL	VTIM	ETV	VL

The VLAN Tag register contains the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the receiving frame (Length/Type) with 16'h8100, and the following two bytes are compared with the VLAN tag. If a match occurs, the MAC sets the received VLAN bit in the receive frame status. The legal length of the frame is increased from 1,518 bytes to 1,522 bytes. If the VLAN Tag register is configured to be double-synchronized to the (G)MII clock domain, then consecutive writes to these register should be performed only after at least four clock cycles in the destination clock domain.

Bits	Name	Memory Access	Description
31:20	Reserved_31_20	R	<b>Value After Reset:</b> 0x0 Reserved
19	VTHM	R/W	<b>Value After Reset:</b> 0x0 VLAN Tag Hash Table Match Enable When set, the most significant four bits of the VLAN tag's CRC are used to index the content of Register 354 (VLAN Hash Table Register). A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the frame matched the VLAN hash table. When Bit 16 (ETV) is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison whereas when ETV is reset, the CRC of the 16-bit VLAN tag is used for comparison. When reset, the VLAN Hash Match operation is not performed. If the VLAN Hash feature is not enabled during core configuration, this bit is reserved (RO with default value).
18	ESVL	R/W	<b>Value After Reset:</b> 0x0 Enable S-VLAN When this bit is set, the MAC transmitter and receiver also consider the S-VLAN (Type = 0x88A8) frames as valid VLAN tagged frames.
17	VTIM	R/W	<b>Value After Reset:</b> 0x0 VLAN Tag Inverse Match Enable When set, this bit enables the VLAN Tag inverse matching. The frames that do not have matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The frames with matched VLAN Tag are marked as matched.

Bits	Name	Memory Access	Description
16	ETV	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Enable 12-Bit VLAN Tag Comparison</p> <p>When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits [11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged frame. Similarly, when enabled, only 12 bits of the VLAN tag in the received frame are used for hash-based VLAN filtering.</p> <p>When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN frame are used for comparison and VLAN hash filtering.</p>
15:0	VL	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>VLAN Tag Identifier for Receive Frames</p> <p>This field contains the 802.1Q VLAN tag to identify the VLAN frames and is compared to the 15th and 16th bytes of the frames being received for VLAN frames. The following list describes the bits of this field:</p> <ul style="list-style-type: none"> <li>Bits [15:13]: User Priority</li> <li>Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)</li> <li>Bits[11:0]: VLAN tag's VLAN Identifier (VID) field</li> </ul> <p>When the ETV bit is set, only the VID (Bits[11:0]) is used for comparison. If VL (VL[11:0] if ETV is set) is all zeros, the MAC does not check the fifteenth and 16th bytes for VLAN tag comparison, and declares all frames with a Type field value of 0x8100 or 0x88a8 as VLAN frames.</p>

#### 9.7.2.1.7 Register 8 (Version Register)

**Size:** 32 bits

**Offset:** 0x20

**Memory Access:** R

**Value After Reset:** 0x1037

31:16	15:8	7:0
Reserved_31_16	USERVER	SNPSVER

The Version registers identifies the version of the DWC\_gmac. This register contains two bytes: one that Synopsys uses to identify the core release number, and the other that you set during core configuration.

Bits	Name	Memory Access	Description
31:16	Reserved_31_16	R	<p><b>Value After Reset:</b> 0x0</p> <p>Reserved</p>
15:8	USERVER	R	<p><b>Value After Reset:</b> 0x10</p> <p>User-defined Version (Configured with the coreConsultant)</p>
7:0	SNPSVER	R	<p><b>Value After Reset:</b> 0x37</p> <p>Synopsys-defined Version (3.7)</p>

#### 9.7.2.1.8 Register 9 (Debug Register)

**Size:** 32 bits

**Offset:** 0x24

**Memory Access:** R

**Value After Reset:** 0x0

31:26	25	24	23	22	21: 20	19	18: 17	16	15:10	9:8	7	6:5	4	3	2:1	0
-------	----	----	----	----	-----------	----	-----------	----	-------	-----	---	-----	---	---	-----	---

Reserve_d_31_26	TXST SFSTS	TXF STS	Reserv ed_23	TW CST S	TRC STS	TXPA USED	TFC STS	TPE STS	Reserve d_15_10	RXF STS	Reser ved_7	RR CST S	RW CST S	Reser ved_3	RFCF CSTS	RPE STS
-----------------	------------	---------	--------------	----------	---------	-----------	---------	---------	-----------------	---------	-------------	----------	----------	-------------	-----------	---------

The Debug register gives the status of all main modules of the transmit and receive data-paths and the FIFOs. An all-zero status indicates that the MAC is in idle state (and FIFOs are empty) and no activity is going on in the data-paths.

Note:

The reset values, given for the Debug register, are valid only if the following clocks are present during the reset operation:

clk\_csr\_i, clk\_app\_i, hclk\_i, or aclk\_i  
clk\_tx\_i  
clk\_rx\_i

Bits	Name	Memory Access	Description
31:26	Reserved_31_26	R	<b>Value After Reset:</b> 0x0 Reserved
25	TXSTSFSTS	R	<b>Value After Reset:</b> 0x0 MTL TxStatus FIFO Full Status When high, this bit indicates that the MTL TxStatus FIFO is full. Therefore, the MTL cannot accept any more frames for transmission. This bit is reserved in the GMAC-AHB and GMAC-DMA configurations.
24	TXFSTS	R	<b>Value After Reset:</b> 0x0 MTL Tx FIFO Not Empty Status When high, this bit indicates that the MTL Tx FIFO is not empty and some data is left for transmission.
23	Reserved_23	R	<b>Value After Reset:</b> 0x0 Reserved
22	TWCSTS	R	<b>Value After Reset:</b> 0x0 MTL Tx FIFO Write Controller Active Status When high, this bit indicates that the MTL Tx FIFO Write Controller is active and transferring data to the Tx FIFO.
21:20	TRCSTS	R	<b>Value After Reset:</b> 0x0 MTL Tx FIFO Read Controller Status This field indicates the state of the Tx FIFO Read Controller: 00: IDLE state 01: READ state (transferring data to MAC transmitter) 10: Waiting for TxStatus from MAC transmitter 11: Writing the received TxStatus or flushing the Tx FIFO
19	TXPAUSED	R	<b>Value After Reset:</b> 0x0 MAC transmitter in PAUSE When high, this bit indicates that the MAC transmitter is in the PAUSE condition (in the full-duplex only mode) and hence does not schedule any frame for transmission.
18:17	TFCSTS	R	<b>Value After Reset:</b> 0x0 MAC Transmit Frame Controller Status This field indicates the state of the MAC Transmit Frame Controller module: 00: IDLE state 01: Waiting for Status of previous frame or IFG or backoff period to be over 10: Generating and transmitting a PAUSE control frame (in the full-duplex mode) 11: Transferring input frame for transmission

Bits	Name	Memory Access	Description
16	TPESTS	R	<b>Value After Reset:</b> 0x0 MAC GMII or MII Transmit Protocol Engine Status When high, this bit indicates that the MAC GMII or MII transmit protocol engine is actively transmitting data and is not in the IDLE state.
15:10	Reserved_15_10	R	<b>Value After Reset:</b> 0x0 Reserved
9:8	RXFSTS	R	<b>Value After Reset:</b> 0x0 MTL Rx FIFO Fill-level Status This field gives the status of the fill-level of the Rx FIFO: 00: Rx FIFO Empty 01: Rx FIFO fill level is below the flow-control deactivate threshold 10: Rx FIFO fill level is above the flow-control activate threshold 11: Rx FIFO Full
7	Reserved_7	R	<b>Value After Reset:</b> 0x0 Reserved
6:5	RRCTS	R	<b>Value After Reset:</b> 0x0 MTL Rx FIFO Read Controller State This field gives the state of the Rx FIFO read Controller: 00: IDLE state 01: Reading frame data 10: Reading frame status (or timestamp) 11: Flushing the frame data and status
4	RWCSTS	R	<b>Value After Reset:</b> 0x0 MTL Rx FIFO Write Controller Active Status When high, this bit indicates that the MTL Rx FIFO Write Controller is active and is transferring a received frame to the FIFO.
3	Reserved_3	R	<b>Value After Reset:</b> 0x0 Reserved
2:1	RFCFCSTS	R	<b>Value After Reset:</b> 0x0 MAC Receive Frame Controller FIFO Status When high, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Frame Controller Module.
0	RPESTS	R	<b>Value After Reset:</b> 0x0 MAC GMII or MII Receive Protocol Engine Status When high, this bit indicates that the MAC GMII or MII receive protocol engine is actively receiving data and not in IDLE state.

#### 9.7.2.1.9 Register 10 (Remote Wake-Up Frame Filter Register)

**Size:** 32 bits

**Offset:** 0x28

**Memory Access:** R/W

**Value After Reset:** 0x0

31:0
WKUPFRMFTR

This is the address through which the application writes or reads the remote wake-up frame filter registers (wkupfmfilter\_reg). The wkupfmfilter\_reg register is a pointer to eight wkupfmfilter\_reg registers. The wkupfmfilter\_reg register is loaded by sequentially loading the eight register values. Eight sequential writes to this address (0x0028) writes all wkupfmfilter\_reg registers. Similarly, eight sequential reads from this address (0x0028) read all wkupfmfilter\_reg registers. This register contains the higher 16 bits of the seventh MAC address. This register is present only when you select the PMT module Remote Wake-up feature in coreConsultant.

Bits	Name	Memory Access	Description
31:0	WKUPFRMFTR	R/W	<b>Value After Reset:</b> 0x0 Remote Wake-Up Frame Filter

#### 9.7.2.1.10 Register 11 (PMT Control and Status Register)

**Size:** 32 bits

**Offset:** 0x2c

**Memory Access:** R/W

**Value After Reset:** 0x0

31	30:27	26:24	23:10	9	8:7	6	5	4:3	2	1	0
RWKFILTRST	Reserved_30_27	RWK PTR	Reserved_23_10	GLBLU CAST	Reserved_8_7	RWKPR CVD	MGKPR CVD	Reserved_4_3	RWKPK TEN	MGKPK TEN	PWRD WN

This register is present only when you select the PMT module in the coreConsultant.

Bits	Name	Memory Access	Description
31	RWKFILTRST	R/W	<b>Value After Reset:</b> 0x0 Wake-Up Frame Filter Register Pointer Reset When this bit is set, it resets the remote wake-up frame filter register pointer to 3'b000. It is automatically cleared after 1 clock cycle.
30:27	Reserved_30_27	R	<b>Value After Reset:</b> 0x0 Reserved
26:24	RWK PTR	R	<b>Value After Reset:</b> 0x0 Remote Wake-up FIFO Pointer This gives the current value (0 to 7) of the Remote Wake-up Frame filter register pointer. The contents of the Remote Wake-up Frame Filter Register are transferred to the clk_rx_i domain when a write occurs to that register when this pointer value equals 7.
23:10	Reserved_23_10	R	<b>Value After Reset:</b> 0x0 Reserved
9	GLBLUCAST	R/W	<b>Value After Reset:</b> 0x0 Global Unicast When set, enables any unicast packet filtered by the MAC (DAF) address recognition to be a wake-up frame.
8:7	Reserved_8_7	R	<b>Value After Reset:</b> 0x0 Reserved
6	RWKPR CVD	R	<b>Value After Reset:</b> 0x0 Wake-Up Frame Received When set, this bit indicates the power management event is generated because of the reception of a wake-up frame. This bit is cleared by a Read into this register.
5	MGKPR CVD	R	<b>Value After Reset:</b> 0x0

Bits	Name	Memory Access	Description
			Magic Packet Received When set, this bit indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared by a Read into this register.
4:3	Reserved_4_3	R	<b>Value After Reset:</b> 0x0 Reserved
2	RWKPKTEN	R/W	<b>Value After Reset:</b> 0x0 Wake-Up Frame Enable When set, enables generation of a power management event because of wake-up frame reception.
1	MGKPKTEN	R/W	<b>Value After Reset:</b> 0x0 Magic Packet Enable When set, enables generation of a power management event because of magic packet reception.
0	PWRDWN	R/W	<b>Value After Reset:</b> 0x0 Power Down When set, the MAC receiver drops all received frames until it receives the expected magic packet or wake-up frame. This bit is then self-cleared and the power-down mode is disabled. The Software can also clear this bit before the expected magic packet or wake-up frame is received. The frames, received by the MAC after this bit is cleared, are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Wake-Up Frame Enable bit is set high. Note: You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit.

#### 9.7.2.1.11 Register 14 (Interrupt Register)

**Size:** 32 bits

**Offset:** 0x38

**Memory Access:** R

**Value After Reset:** 0x0

31:12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved_31_12	GPII S	LPII S	TSI S	Reserved _8	MMCRXI PIS	MMCTX IS	MMCRX IS	MMC IS	PMTI S	PCSANC IS	PCSLCH GIS	RGSMII IS

The Interrupt Status register identifies the events in the MAC that can generate interrupt. All interrupt events are generated only when the corresponding optional feature is selected during core configuration and enabled during operation. Therefore, these bits are reserved when the corresponding features are not present in the core.

Bits	Name	Memory Access	Description
31:12	Reserved_31_12	R	<b>Value After Reset:</b> 0x0 Reserved
11	GPIIS	R	<b>Value After Reset:</b> 0x0 GPI Interrupt Status When the GPIO feature is enabled, this bit is set when any active event (LL or LH) occurs on GPIS field (Bits [3:0]) of Register 56 (General Purpose IO Register) and the

Bits	Name	Memory Access	Description
			corresponding GPIO bit is enabled. This bit is cleared on reading the lane 0 (GPIS) of Register 56 (General Purpose IO Register). When the GPIO feature is not enabled, this bit is reserved.
10	LPIIS	R	<p><b>Value After Reset:</b> 0x0  <b>LPI Interrupt Status</b>  When the Energy Efficient Ethernet feature is enabled, this bit is set for any LPI state entry or exit in the MAC Transmitter or Receiver. This bit is cleared on reading Bit 0 of Register 12 (LPI Control and Status Register). In all other modes, this bit is reserved.</p>
9	TSIS	R	<p><b>Value After Reset:</b> 0x0  <b>Timestamp Interrupt Status</b>  When the Advanced Timestamp feature is enabled, this bit is set when any of the following conditions is true:  The system time value equals or exceeds the value specified in the Target Time High and Low registers.  There is an overflow in the seconds register.  The Auxiliary snapshot trigger is asserted.  This bit is cleared on reading Bit 0 of the Register 458 (Timestamp Status Register). If default Timestamping is enabled, when set, this bit indicates that the system time value is equal to or exceeds the value specified in the Target Time registers. In this mode, this bit is cleared after the completion of the read of this bit. In all other modes, this bit is reserved.</p>
8	Reserved_8	R	<p><b>Value After Reset:</b> 0x0  <b>Reserved</b></p>
7	MMCRXIPIS	R	<p><b>Value After Reset:</b> 0x0  <b>MMC Receive Checksum Offload Interrupt Status</b>  This bit is set high when an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared.  This bit is valid only when you select the optional MMC module and Checksum Offload Engine (Type 2) during core configuration.</p>
6	MMCTXIS	R	<p><b>Value After Reset:</b> 0x0  <b>MMC Transmit Interrupt Status</b>  This bit is set high when an interrupt is generated in the MMC Transmit Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared.  This bit is valid only when you select the optional MMC module during core configuration.</p>
5	MMCRXIS	R	<p><b>Value After Reset:</b> 0x0  <b>MMC Receive Interrupt Status</b>  This bit is set high when an interrupt is generated in the MMC Receive Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared.  This bit is valid only when you select the optional MMC module during core configuration.</p>
4	MMCIS	R	<p><b>Value After Reset:</b> 0x0  <b>MMC Interrupt Status</b>  This bit is set high when any of the Bits [7:5] is set high and cleared only when all of these bits are low.</p>

Bits	Name	Memory Access	Description
			This bit is valid only when you select the optional MMC module during core configuration.
3	PMTIS	R	<p><b>Value After Reset:</b> 0x0  <b>PMT Interrupt Status</b>  This bit is set when a Magic packet or Wake-on-LAN frame is received in the power-down mode (see Bits 5 and 6 in the PMT Control and Status Register). This bit is cleared when both Bits[6:5] are cleared because of a read operation to the PMT Control and Status register.  This bit is valid only when you select the optional PMT module during core configuration.</p>
2	PCSANCIS	R	<p><b>Value After Reset:</b> 0x0  <b>PCS Auto-Negotiation Complete</b>  This bit is set when the Auto-negotiation is completed in the TBI, RTBI, or SGMII PHY interface (Bit 5 in Register 49 (AN Status Register)). This bit is cleared when you perform a read operation to the AN Status register.  This bit is valid only when you select the optional TBI, RTBI, or SGMII PHY interface during core configuration and operation.</p>
1	PCSLCHGIS	R	<p><b>Value After Reset:</b> 0x0  <b>PCS Link Status Changed</b>  This bit is set because of any change in Link Status in the TBI, RTBI, or SGMII PHY interface (Bit 2 in Register 49 (AN Status Register)). This bit is cleared when you perform a read operation on the AN Status register. This bit is valid only when you select the optional TBI, RTBI, or SGMII PHY interface during core configuration and operation.</p>
0	RGSMIIIS	R	<p><b>Value After Reset:</b> 0x0  <b>RGMII or SMII Interrupt Status</b>  This bit is set because of any change in value of the Link Status of RGMII or SMII interface (Bit 3 in Register 54 (SGMII/RGMII/SMII Status Register)). This bit is cleared when you perform a read operation on the SGMII/RGMII/SMII Status Register.  This bit is valid only when you select the optional RGMII or SMII PHY interface during core configuration and operation.</p>

#### 9.7.2.1.12 Register 15 (Interrupt Mask Register)

**Size:** 32 bits

**Offset:** 0x3c

**Memory Access:** R/W

**Value After Reset:** 0x0

31:11	10	9	8:4	3	2	1	0
Reserved_31_11	LPIIM	TSIM	Reserved_8_4	PMTIM	PCSANCIM	PCSLCHGIM	RGSMIIIM

The Interrupt Mask Register bits enable you to mask the interrupt signal because of the corresponding event in the Interrupt Status Register. The interrupt signal is sbd\_intr\_o in the GMAC-AHB, GMAC-AXI, and GMAC-DMA configuration and mci\_intr\_o in the GMAC-MTL and GMAC-CORE configuration.

Bits	Name	Memory Access	Description
31:11	Reserved_31_11	R	<b>Value After Reset:</b> 0x0

Bits	Name	Memory Access	Description
			Reserved
10	LPIIM	R	<p><b>Value After Reset:</b> 0x0 LPI Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of the LPI Interrupt Status bit in Register 14 (Interrupt Status Register). This bit is valid only when you select the Energy Efficient Ethernet feature during core configuration. In all other modes, this bit is reserved.</p>
9	TSIM	R	<p><b>Value After Reset:</b> 0x0 Timestamp Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of Timestamp Interrupt Status bit in Register 14 (Interrupt Status Register). This bit is valid only when IEEE1588 timestamping is enabled. In all other modes, this bit is reserved.</p>
8:4	Reserved_8_4	R	<p><b>Value After Reset:</b> 0x0 Reserved</p>
3	PMTIM	R/W	<p><b>Value After Reset:</b> 0x0 PMT Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of PMT Interrupt Status bit in Register 14 (Interrupt Status Register).</p>
2	PCSANCIM	R	<p><b>Value After Reset:</b> 0x0 PCS AN Completion Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of PCS Auto-negotiation complete bit in Register 14 (Interrupt Status Register).</p>
1	PCSLCHGIM	R	<p><b>Value After Reset:</b> 0x0 PCS Link Status Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of the PCS Link-status changed bit in Register 14 (Interrupt Status Register).</p>
0	RGSMIIIM	R	<p><b>Value After Reset:</b> 0x0 RGMII or SMII Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of the RGMII or SMII Interrupt Status bit in Register 14 (Interrupt Status Register).</p>

#### 9.7.2.1.13 Register 16 (MAC Address0 High Register)

**Size:** 32 bits

**Offset:** 0x40

**Memory Access:** R/W

**Value After Reset:** 0x8000ffff

31	30:16	15:0
AE	Reserved_30_16	ADDRHI

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the (G)MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the (G)MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x665544332211.

If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

Bits	Name	Memory Access	Description
31	AE	R	<b>Value After Reset:</b> 0x1 Address Enable This bit is always set to 1.
30:16	Reserved_30_16	R	<b>Value After Reset:</b> 0x0 Reserved
15:0	ADDRHI	R/W	<b>Value After Reset:</b> 0xffff MAC Address0 [47:32] This field contains the upper 16 bits (47:32) of the first 6-byte MAC address. The MAC uses this field for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.

#### 9.7.2.1.14 Register 17 (MAC Address0 Low Register)

**Size:** 32 bits

**Offset:** 0x44

**Memory Access:** R/W

**Value After Reset:** 0xffffffff



The MAC Address0 Low register holds the lower 32 bits of the first 6-byte MAC address of the station.

Bits	Name	Memory Access	Description
31:0	ADDRLO	R/W	<b>Value After Reset:</b> 0xffffffff MAC Address0 [31:0] This field contains the lower 32 bits of the first 6-byte MAC address. This is used by the MAC for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.

#### 9.7.2.1.15 Register 18 (MAC Address1 High Register)

**Size:** 32 bits

**Offset:** 0x48

**Memory Access:** R/W

**Value After Reset:** 0xffff



The MAC Address1 High register holds the upper 16 bits of the second 6-byte MAC address of the station.

If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address1 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

Bits	Name	Memory Access	Description
31	AE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Address Enable</p> <p>When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.</p>
30	SA	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Source Address</p> <p>When this bit is set, the MAC Address1[47:0] is used to compare with the SA fields of the received frame.</p> <p>When this bit is reset, the MAC Address1[47:0] is used to compare with the DA fields of the received frame.</p>
29:24	MBC	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Mask Byte Control</p> <p>These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:</p> <ul style="list-style-type: none"> <li>Bit 29: Register 18[15:8]</li> <li>Bit 28: Register 18[7:0]</li> <li>Bit 27: Register 19[31:24]</li> <li>...</li> <li>Bit 24: Register 19[7:0]</li> </ul> <p>You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.</p>
23:16	Reserved_23_16	R	<p><b>Value After Reset:</b> 0x0</p> <p>Reserved</p>
15:0	ADDRHI	R/W	<p><b>Value After Reset:</b> 0xffff</p> <p>MAC Address1 [47:32]</p> <p>This field contains the upper 16 bits (47:32) of the second 6-byte MAC address.</p>

#### 9.7.2.1.16 Register 19 (MAC Address1 Low Register)

**MAC\_Address1\_Low**

**Size:** 32 bits

**Offset:** 0x4c

**Memory Access:** R/W

**Value After Reset:** 0xffffffff



The MAC Address1 Low register holds the lower 32 bits of the second 6-byte MAC address of the station.

Bits	Name	Memory Access	Description
31:0	ADDRLO	R/W	<p><b>Value After Reset:</b> 0xffffffff</p> <p>MAC Address1 [31:0]</p> <p>This field contains the lower 32 bits of the second 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.</p>

**NOTE:**

- The descriptions for registers 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, and 46 (MAC Address2 High Register through MAC Address15 High Register) are the same as for the Register 18 (MAC Address1 High Register).
- The descriptions for registers 21, 23, 25, 27, 29, 31, 33, 35, 37, 38, 41, 43, 45, and 47 (MAC Address2 Low Register through MAC Address15 Low Register) are the same as for the Register 19 (MAC Address1 Low Register).
- The descriptions for registers 512, 514, 516, 518, 520, 522, 524, 526, 528, 530, 532, 534, 536, 538, 540, and 542 (MAC Address16 High Register through MAC Address31 High Register) are the same as for the Register 18 (MAC Address1 High Register).
- The descriptions for registers 513, 515, 517, 519, 521, 523, 525, 527, 529, 531, 533, 535, 537, 539, 541, and 543 (MAC Address16 Low Register through MAC Address31 Low Register) are the same as for the Register 19 (MAC Address1 Low Register).
- The descriptions for registers 546, 548, 550, 552, 554, 556, 558, 560, 562, 564, 566, 568, 570, 572, 574, 576, 578, 580, 582, 584, 586, 588, 590, 592, 594, 596, 598, 600, 602, 604, 606, 608, 610, 612, 614, 616, 618, 620, 622, 624, 626, 628, 630, 632, 634, 636, 638, 640, 642, 644, 646, 648, 650, 652, 654, 656, 658, 660, 662, 664, 666, 668, 670, 672, 674, 676, 678, 680, 682, 684, 686, 688, 690, 692, 694, 696, 698, 700, 702, 704, 706, 708, 710, 712, 714, 716, 718, 720, 722, 724, 726, 728, 730, 732, and 734 (MAC Address33 High Register through MAC Address127 High Register) are the same as for the Register 544 (MAC Address32 High Register).
- The descriptions for registers 545, 547, 549, 551, 553, 555, 557, 559, 561, 563, 565, 567, 569, 571, 573, 575, 577, 579, 581, 583, 585, 587, 589, 591, 593, 595, 597, 599, 601, 603, 605, 607, 609, 611, 613, 615, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 663, 665, 667, 669, 671, 673, 675, 677, 679, 681, 683, 685, 687, 689, 691, 693, 695, 697, 699, 701, 703, 705, 707, 709, 711, 713, 715, 717, 719, 721, 723, 725, 727, 729, 731, 733, and 735 (MAC Address32 Low Register through MAC Address127 Low Register) are the same as for the Register 19 (MAC Address1 Low Register).

**9.7.2.1.17 Register 55 (Watchdog Timeout Register)****WDog\_Timeout****Size:** 32 bits**Offset:** 0xdc**Memory Access:** R/W**Value After Reset:** 0x0

31:17	16	15:14	13:0
Reserved_31_17	PWE	Reserved_15_14	WTO

This register controls the watchdog timeout for received frames.

Bits	Name	Memory Access	Description
31:17	Reserved_31_17	R	<b>Value After Reset:</b> 0x0 Reserved
16	PWE	R/W	<b>Value After Reset:</b> 0x0 Programmable Watchdog Enable When this bit is set and Bit 23 (WD) of Register 0 (MAC Configuration Register) is reset, the WTO field (Bits[13:0]) is used as watchdog timeout for a received frame. When this bit is cleared, the watchdog timeout for a received frame is controlled by the setting of Bit 23 (WD) and Bit 20 (JE) in Register 0 (MAC Configuration Register).
15:14	Reserved_15_14	R	<b>Value After Reset:</b> 0x0 Reserved
13:0	WTO	R/W	<b>Value After Reset:</b> 0x0

Bits	Name	Memory Access	Description
			<p><b>Watchdog Timeout</b>  When Bit 16 (PWE) is set and Bit 23 (WD) of Register 0 (MAC Configuration Register) is reset, this field is used as watchdog timeout for a received frame. If the length of a received frame exceeds the value of this field, such frame is terminated and declared as an error frame.</p> <p>Note: When Bit 16 (PWE) is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE Std 802.3-specified valid tagged frames are declared as error frames and are dropped.</p>

#### 9.7.2.1.18 Register 64 (MMC Control Register)

##### MMC\_Control

**Size:** 32 bits

**Offset:** 0x100

**Memory Access:** R/W

**Value After Reset:** 0x0

31:9	8	7:6	5	4	3	2	1	0
Reserved_31_9	UCDBC	Reserved_7_6	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST

The MMC Control register establishes the operating mode of the management counters.

Note: The bit 0 (Counters Reset) has higher priority than bit 4 (Counter Preset). Therefore, when the Software tries to set both bits in the same write cycle, all counters are cleared and the bit 4 is not set.

Bits	Name	Memory Access	Description
31:9	Reserved_31_9	R	<p><b>Value After Reset:</b> 0x0</p> <p>Reserved</p>
8	UCDBC	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Update MMC Counters for Dropped Broadcast Frames</p> <p>When set, this bit enables MAC to update all the related MMC Counters for Broadcast frames dropped due to setting of DBF bit (Disable Broadcast Frames) of MAC Filter Register at offset 0x0004. When reset, MMC Counters are not updated for dropped Broadcast frames.</p>
7:6	Reserved_7_6	R	<p><b>Value After Reset:</b> 0x0</p> <p>Reserved</p>
5	CNTPRSTLVL	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Full-Half Preset</p> <p>When low and bit 4 is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF_F800 (half - 2KBytes) and all frame-counters gets preset to 0x7FFF_FFF0 (half - 16). When this bit is high and bit 4 is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (full - 2KBytes) and all frame-counters gets preset to 0xFFFF_FFF0 (full - 16). For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and frame counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFF00.</p>
4	CNTPRST	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Counters Preset</p> <p>When this bit is set, all counters are initialized or preset to almost full or almost half according to bit 5. This bit is cleared automatically after 1 clock cycle. This bit, along</p>

Bits	Name	Memory Access	Description
			with bit 5, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.
3	CNTFREEZ	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Counter Freeze</p> <p>When this bit is set, it freezes all MMC counters to their current value. Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received frame. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.</p>
2	RSTONRD	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Reset on Read</p> <p>When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (bits[7:0]) is read.</p>
1	CNTSTOPRO	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Counters Stop Rollover</p> <p>When this bit is set, after reaching maximum value, the counter does not roll over to zero.</p>
0	CNTRST	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Counters Reset</p> <p>When this bit is set, all counters are reset. This bit is cleared automatically after one clock cycle.</p>

### **9.7.2.1.19 Register 65 (MMC Receive Interrupt Register)**

## **MMC\_Receive\_Interrupt**

**Size:** 32 bits

**Offset:** 0x104

## Memory Access: R

**Value After Reset: 0x0**

31 :2 6	2 5	2 4	2 3	22	2 1	2 0	19	1 8	1 7	16	15	14	13	12	11	1 0	9	8	7	6	5	4	3	2	1	0
Re se rv ed _3 1 _26	R X C T R L F I S	R X R V E R R F I S	R X W D O N G F B F I S	R X F O A V U S I S	R X P A N G F I S	R X O R A N G E R F I S	R X L U N E R G F I S	R X 024 TM AX OCT CTG BFI S	R X 12T 102 30 CT GB FIS	R X RX5 102 30 CT GB FIS	R X 256 T51 10 CT GB FIS	R X 128 T25 50 CT GB FIS	R X 65 T1 27 OC TG BFI S	R X 64 O CT G BF IS	R X U SI Z E G FI S	R X O SI Z E G FI S	R X A B E R FI S	R X U SI Z E G FI S	R X R U N T FI S	R X AL G N E RF IS	R X C R C E R FI S	R X M C G FI S	R X B C G FI S	R X G O C TI S	R X G B O C TI S	R X G B F R M IS

The MMC Receive Interrupt register maintains the interrupts that are generated when the following happens:

Receive statistic counters reach half of their maximum values (0x8000\_0000 for 32-bit counter and 0x8000 for 16-bit counter).

Receive statistic counters cross their maximum values (0xFFFF\_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When the Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

Bits	Name	Memory Access	Description
31:26	Reserved_31_26	R	<b>Value After Reset:</b> 0x0 Reserved
25	RXCTRLFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive Control Frame Counter Interrupt Status This bit is set when the rxctrlframes_g counter reaches half of the maximum value or the maximum value.
24	RXRCVERRFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive Error Frame Counter Interrupt Status This bit is set when the rxrcverror counter reaches half of the maximum value or the maximum value.
23	RXWDOGFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive Watchdog Error Frame Counter Interrupt Status This bit is set when the rxwatchdogerror counter reaches half of the maximum value or the maximum value.
22	RXVLANGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive VLAN Good Bad Frame Counter Interrupt Status This bit is set when the rxvlanframes_gb counter reaches half of the maximum value or the maximum value.
21	RXFOVFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive FIFO Overflow Frame Counter Interrupt Status This bit is set when the rxfifooverflow counter reaches half of the maximum value or the maximum value.
20	RXPAUSFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive Pause Frame Counter Interrupt Status This bit is set when the rxpauseframe counter reaches half of the maximum value or the maximum value.
19	RXORANGEFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive Out Of Range Error Frame Counter Interrupt Status This bit is set when the rxoutofrangecounter reaches half of the maximum value or the maximum value.
18	RXLENERFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive Length Error Frame Counter Interrupt Status This bit is set when the rxlengtherror counter reaches half of the maximum value or the maximum value.
17	RXUCGFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive Unicast Good Frame Counter Interrupt Status This bit is set when the rxunicastframes_gb counter reaches half of the maximum value or the maximum value.
16	RX1024TMAXOCTGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive 1024 to Maximum Octet Good Bad Frame Counter Interrupt Status This bit is set when the rx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value.
15	RX512T1023OCTGBFIS	R	<b>Value After Reset:</b> 0x0

Bits	Name	Memory Access	Description
			MMC Receive 512 to 1023 Octet Good Bad Frame Counter Interrupt Status This bit is set when the rx512to1023octets_gb counter reaches half of the maximum value or the maximum value.
14	RX256T511OCTGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive 256 to 511 Octet Good Bad Frame Counter Interrupt Status This bit is set when the rx256to511octets_gb counter reaches half of the maximum value or the maximum value.
13	RX128T255OCTGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive 128 to 255 Octet Good Bad Frame Counter Interrupt Status This bit is set when the rx128to255octets_gb counter reaches half of the maximum value or the maximum value.
12	RX65T127OCTGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive 65 to 127 Octet Good Bad Frame Counter Interrupt Status This is set when the rx65to127octets_gb counter reaches half of the maximum value or the maximum value.
11	RX64OCTGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive 64 Octet Good Bad Frame Counter Interrupt Status This bit is set when the rx64octets_gb counter reaches half of the maximum value or the maximum value.
10	RXOSIZEGFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive Oversize Good Frame Counter Interrupt Status This bit is set when the rxoversize_g counter reaches half of the maximum value or the maximum value.
9	RXUSIZEGFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive Undersize Good Frame Counter Interrupt Status This bit is set when the rxundersize_g counter reaches half of the maximum value or the maximum value.
8	RXJABERFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive Jabber Error Frame Counter Interrupt Status This bit is set when the rxjabbererror counter reaches half of the maximum value or the maximum value.
7	RXRUNTFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive Runt Frame Counter Interrupt Status This bit is set when the rxrunterror counter reaches half of the maximum value or the maximum value.
6	RXALGNERFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive Alignment Error Frame Counter Interrupt Status This bit is set when the rxalignmenterror counter reaches half of the maximum value or the maximum value.
5	RXCRCERFIS	R	<b>Value After Reset:</b> 0x0 MMC Receive CRC Error Frame Counter Interrupt Status This bit is set when the rxcrcerror counter reaches half of the maximum value or the maximum value.

Bits	Name	Memory Access	Description
4	RXMGCFIS	R	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Receive Multicast Good Frame Counter Interrupt Status</p> <p>This bit is set when the rxmulticastframes_g counter reaches half of the maximum value or the maximum value.</p>
3	RXBBCGFIS	R	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Receive Broadcast Good Frame Counter Interrupt Status.</p> <p>This bit is set when the rxbroadcastframes_g counter reaches half of the maximum value or the maximum value.</p>
2	RXGOCTIS	R	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Receive Good Octet Counter Interrupt Status.</p> <p>This bit is set when the rxoctetcount_g counter reaches half of the maximum value or the maximum value.</p>
1	RXGBOCTIS	R	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Receive Good Bad Octet Counter Interrupt Status</p> <p>This bit is set when the rxoctetcount_bg counter reaches half of the maximum value or the maximum value.</p>
0	RXGBFRMIS	R	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Receive Good Bad Frame Counter Interrupt Status</p> <p>This bit is set when the rxframecount_bg counter reaches half of the maximum value or the maximum value.</p>

#### 9.7.2.1.20 Register 66 (MMC Transmit Interrupt Register)

## **MMC\_Transmit\_Interrupt**

**Size:** 32 bits

**Offset:** 0x108

## Memory Access: R

**Value After Reset: 0x0**

31 :2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	13	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
Re ser ve d_ 31 _2 6	T X S I Z E G G F I S	T X V L A N G G F I S	T X P A U S E R M T I S	T X E G O C R C O T F I S	T X C A R E C O O F I S	T X E X C D E C O F F I S	T X XL A T C O O L G F I S	T X M D E C O O L G F I S	T X S C C O L G F I S	T X UF LO W ER FI S	T X B C G B B FI S	T X M C G B B FI S	T X U C G B B FI S	TX1 024 TM AX OCT GBF IS	TX5 12T 102 30 CTG BFI S	TX2 56T 511 OC TG BFI S	TX1 28T 255 OC TG BFI S	TX 65 T1 27 OC TG BFI S	TX 64 O CT G BF IS	T X M C G F I S	T X B C G F R M IS	T X G B O C TI S				

The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half of their maximum values (0x8000\_0000 for 32-bit counter and 0x8000 for 16-bit counter), and the maximum values (0xFFFF\_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Transmit Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

Bits	Name	Memory Access	Description
------	------	---------------	-------------

Bits	Name	Memory Access	Description
31:26	Reserved_31_26	R	<b>Value After Reset:</b> 0x0 Reserved
25	TXOSIZEGFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Oversize Good Frame Counter Interrupt Status This bit is set when the txoversize_g counter reaches half of the maximum value or the maximum value.
24	TXVLANGFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit VLAN Good Frame Counter Interrupt Status This bit is set when the txvlanframes_g counter reaches half of the maximum value or the maximum value.
23	TXPAUSFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Pause Frame Counter Interrupt Status This bit is set when the txpauseframeserror counter reaches half of the maximum value or the maximum value.
22	TXEXDEFFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Excessive Deferral Frame Counter Interrupt Status This bit is set when the txexcessdef counter reaches half of the maximum value or the maximum value.
21	TXGFRMIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Good Frame Counter Interrupt Status This bit is set when the txframecount_g counter reaches half of the maximum value or the maximum value.
20	TXGOCTIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Good Octet Counter Interrupt Status This bit is set when the txoctetcountr_g counter reaches half of the maximum value or the maximum value.
19	TXCARERFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Carrier Error Frame Counter Interrupt Status This bit is set when the txcarriererror counter reaches half of the maximum value or the maximum value.
18	TXEXCOLFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Excessive Collision Frame Counter Interrupt Status This bit is set when the txexcesscol counter reaches half of the maximum value or the maximum value.
17	TXLATCOLFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Late Collision Frame Counter Interrupt Status This bit is set when the txlatecol counter reaches half of the maximum value or the maximum value.
16	TXDEFFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Deferred Frame Counter Interrupt Status This bit is set when the txdeferred counter reaches half of the maximum value or the maximum value.
15	TXMCOLGFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Multiple Collision Good Frame Counter Interrupt Status

Bits	Name	Memory Access	Description
			This bit is set when the txmulticol_g counter reaches half of the maximum value or the maximum value.
14	TXSCOLGFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Single Collision Good Frame Counter Interrupt Status This bit is set when the txsinglecol_g counter reaches half of the maximum value or the maximum value.
13	TXUFLOWERFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Underflow Error Frame Counter Interrupt Status This bit is set when the txunderflowerror counter reaches half of the maximum value or the maximum value.
12	TXBCGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Broadcast Good Bad Frame Counter Interrupt Status This bit is set when the txbroadcastframes_gb counter reaches half of the maximum value or the maximum value.
11	TXMCGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Multicast Good Bad Frame Counter Interrupt Status This bit is set when the txmulticastframes_gb counter reaches half of the maximum value or the maximum value.
10	TXUCGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Unicast Good Bad Frame Counter Interrupt Status This bit is set when the txunicastframes_gb counter reaches half of the maximum value or the maximum value.
9	TX1024TMAXOCTGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit 1024 to Maximum Octet Good Bad Frame Counter Interrupt Status This bit is set when the tx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value.
8	TX512T1023OCTGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit 512 to 1023 Octet Good Bad Frame Counter Interrupt Status This bit is set when the tx512to1023octets_gb counter reaches half of the maximum value or the maximum value.
7	TX256T511OCTGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit 256 to 511 Octet Good Bad Frame Counter Interrupt Status This bit is set when the tx256to511octets_gb counter reaches half of the maximum value or the maximum value.
6	TX128T255OCTGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit 128 to 255 Octet Good Bad Frame Counter Interrupt Status This bit is set when the tx128to255octets_gb counter reaches half of the maximum value or the maximum value.
5	TX65T127OCTGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit 65 to 127 Octet Good Bad Frame Counter Interrupt Status This bit is set when the tx65to127octets_gb counter reaches half of the maximum value or the maximum value.

Bits	Name	Memory Access	Description
			maximum value or the maximum value.
4	TX64OCTGBFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit 64 Octet Good Bad Frame Counter Interrupt Status. This bit is set when the tx64octets_gb counter reaches half of the maximum value or the maximum value.
3	TXMCGFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Multicast Good Frame Counter Interrupt Status This bit is set when the txmulticastframes_g counter reaches half of the maximum value or the maximum value.
2	TXBCGFIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Broadcast Good Frame Counter Interrupt Status This bit is set when the txbroadcastframes_g counter reaches half of the maximum value or the maximum value.
1	TXGBFRMIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Good Bad Frame Counter Interrupt Status This bit is set when the txframecount_gb counter reaches half of the maximum value or the maximum value.
0	TXGBOCTIS	R	<b>Value After Reset:</b> 0x0 MMC Transmit Good Bad Octet Counter Interrupt Status This bit is set when the txoctetcountr_gb counter reaches half of the maximum value or the maximum value.

#### 9.7.2.1.21 Register 67 (MMC Receive Interrupt Mask Register)

##### MMC\_Receive\_Interrupt\_Mask

**Size:** 32 bits

**Offset:** 0x10c

**Memory Access:** R/W

**Value After Reset:** 0x0

31 :26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Re se rv ed _3 1 26	R X C T R L F I M	R X W D O G BF FI M	R X VL A N O V S FI I	R X P F U A N V M	R X O R E U A N G	R X L X U C N E G	R X O R E U C N G	R X 024 TM 102 30 OCT CT GB GBF IM	R X 12T 10 50 CT CT GB GB FIM	R X 256 T51 125 T25 50 CT CT GB TG FI FI	R X 128 T25 T1 27 OC G TG BFI IM	R X 65 T1 27 OC G TG BFI IM	R X 64 O CT G TG B E G FI IM	R X O SI Z OC G TG B E G FI IM	R X U SI Z CT G TG B E G FI IM	R X A SI Z E G FI IM	R X U SI Z E G FI IM	R X A B E R FI IM	R X R U N E RF FI IM	R X C R U N E RF FI IM	R X C R U N E RF FI IM	R X G B C G O C TI M	R X G B C G O C TI M			

The MMC Receive Interrupt Mask register maintains the masks for the interrupts generated when the receive statistic counters reach half of their maximum value, or maximum value. This register is 32-bits wide.

Bits	Name	Memory Access	Description
------	------	---------------	-------------

Bits	Name	Memory Access	Description
31:26	Reserved_31_26	R	<b>Value After Reset:</b> 0x0 Reserved
25	RXCTRLFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Control Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxctrlframes counter reaches half the maximum value, and also when it reaches the maximum value.
24	RXRCVERRFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxrcerror error counter reaches half the maximum value, and also when it reaches the maximum value.
23	RXWDOGFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Watchdog Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxwatchdog counter reaches half of the maximum value or the maximum value.
22	RXVLANGBFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive VLAN Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxvlanframes_gb counter reaches half of the maximum value or the maximum value.
21	RXFOVFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive FIFO Overflow Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxfifooverflow counter reaches half of the maximum value or the maximum value.
20	RXPAUSFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Pause Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxpauseframes counter reaches half of the maximum value or the maximum value.
19	RXORANGEFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Out Of Range Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxoutofrangetype counter reaches half of the maximum value or the maximum value.
18	RXLENERFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Length Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxlengtherror counter reaches half of the maximum value or the maximum value.
17	RXUCGFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Unicast Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxunicastframes_g counter reaches half of the maximum value or the maximum value.
16	RX1024TMAXOCTGBFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive 1024 to Maximum Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value.

Bits	Name	Memory Access	Description
15	RX512T1023OCTGBFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive 512 to 1023 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rx512to1023octets_gb counter reaches half of the maximum value or the maximum value.
14	RX256T511OCTGBFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive 256 to 511 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rx256to511octets_gb counter reaches half of the maximum value or the maximum value.
13	RX128T255OCTGBFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive 128 to 255 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rx128to255octets_gb counter reaches half of the maximum value or the maximum value.
12	RX65T127OCTGBFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive 65 to 127 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rx65to127octets_gb counter reaches half of the maximum value or the maximum value.
11	RX64OCTGBFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive 64 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rx64octets_gb counter reaches half of the maximum value or the maximum value.
10	RXOSIZEGFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Oversize Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxoversize_g counter reaches half of the maximum value or the maximum value.
9	RXUSIZEGFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Undersize Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxundersize_g counter reaches half of the maximum value or the maximum value.
8	RXJABERFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Jabber Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxjabbererror counter reaches half of the maximum value or the maximum value.
7	RXRUNTFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Runt Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxrunterror counter reaches half of the maximum value or the maximum value.
6	RXALGNERFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Alignment Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxalignmenrror counter reaches half of the maximum value or the maximum value.
5	RXCRCERFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive CRC Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxcrcerror counter reaches half of the maximum value or the maximum value.

Bits	Name	Memory Access	Description
4	RXMGCFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Multicast Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxmulticastframes_g counter reaches half of the maximum value or the maximum value.
3	RXBGCFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Broadcast Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxbroadcastframes_g counter reaches half of the maximum value or the maximum value.
2	RXGOCTIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxoctetcount_g counter reaches half of the maximum value or the maximum value.
1	RXGBOCTIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Good Bad Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxoctetcount_gb counter reaches half of the maximum value or the maximum value.
0	RXGBFRMIM	R/W	<b>Value After Reset:</b> 0x0 MMC Receive Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxframecount_gb counter reaches half of the maximum value or the maximum value.

### 9.7.2.1.22 Register 68 (MMC Transmit Interrupt Mask Register)

#### MMC\_Transmit\_Interrupt\_Mask

**Size:** 32 bits

**Offset:** 0x110

**Memory Access:** R/W

**Value After Reset:** 0x0

31 :26	25	24	23	22	21	20	9	8	7	6	5	4	13	2	1	1	0	9	8	7	6	5	4	3	2	1	0	
Re se rv ed _3 _1 _26	T X V L A U N S E M F I M	T X E P X D R M F I M	T X G G O C E O F I M	T X A G R C O O F I M	T X T D C C F L I M	T X M C O L G I M	T X S C W G B I M	T X U F O R E R F I M	T X B C G G B F I M	T X M C C G B F I M	T X U C G B F I M	T X C G B F I M	TX1 024 12T 102 30 CT GB IM	TX5 56T 511 255 OC TG BFI M	TX2 28T 511 255 TG BFI M	TX1 27 OC TG BFI M	TX1 65 T1 27 OC TG BFI M	TX 64 O CT G BF I M	T X M C G F I M	T X G B O C T I M	T X G B O C T I M	T X G B O C T I M	T X G B O C T I M	T X G B O C T I M	T X G B O C T I M	T X G B O C T I M	T X G B O C T I M	T X G B O C T I M

The MMC Transmit Interrupt Mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or maximum value. This register is 32-bits wide.

Bits	Name	Memory Access	Description
31:26	Reserved_31_26	R	<b>Value After Reset:</b> 0x0 Reserved

Bits	Name	Memory Access	Description
25	TXOSIZEGFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Oversize Good Frame Counter Interrupt Mask</p> <p>Setting this bit masks the interrupt when the txoversize_g counter reaches half of the maximum value or the maximum value.</p>
24	TXVLANGFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit VLAN Good Frame Counter Interrupt Mask</p> <p>Setting this bit masks the interrupt when the txvlanframes_g counter reaches half of the maximum value or the maximum value.</p>
23	TXPAUSFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Pause Frame Counter Interrupt Mask</p> <p>Setting this bit masks the interrupt when the txpauseframes counter reaches half of the maximum value or the maximum value.</p>
22	TEXEXDEFFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Excessive Deferral Frame Counter Interrupt Mask</p> <p>Setting this bit masks the interrupt when the txexcessdef counter reaches half of the maximum value or the maximum value.</p>
21	TXGFRMIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Good Frame Counter Interrupt Mask</p> <p>Setting this bit masks the interrupt when the txframecount_g counter reaches half of the maximum value or the maximum value.</p>
20	TXGOCTIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Good Octet Counter Interrupt Mask</p> <p>Setting this bit masks the interrupt when the txoctetcount_g counter reaches half of the maximum value or the maximum value.</p>
19	TXCARERFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Carrier Error Frame Counter Interrupt Mask</p> <p>Setting this bit masks the interrupt when the txcarriererror counter reaches half of the maximum value or the maximum value.</p>
18	TEXEXCOLFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Excessive Collision Frame Counter Interrupt Mask</p> <p>Setting this bit masks the interrupt when the txexcesscol counter reaches half of the maximum value or the maximum value.</p>
17	TXLATCOLFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Late Collision Frame Counter Interrupt Mask</p> <p>Setting this bit masks the interrupt when the txlatecol counter reaches half of the maximum value or the maximum value.</p>
16	TXDEFFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Deferred Frame Counter Interrupt Mask</p> <p>Setting this bit masks the interrupt when the txdeferred counter reaches half of the maximum value or the maximum value.</p>
15	TXMCOLGFI	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Multiple Collision Good Frame Counter Interrupt Mask</p> <p>Setting this bit masks the interrupt when the txmulticol_g counter reaches half of the maximum value or the maximum value.</p>

Bits	Name	Memory Access	Description
14	TXSCOLGFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Single Collision Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the txsinglecol_g counter reaches half of the maximum value or the maximum value.</p>
13	TXUFLOWERFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Underflow Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the txunderflowerror counter reaches half of the maximum value or the maximum value.</p>
12	TXBCGBFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Broadcast Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the txbroadcastframes_gb counter reaches half of the maximum value or the maximum value.</p>
11	TXMCGBFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Multicast Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the txmulticastframes_gb counter reaches half of the maximum value or the maximum value.</p>
10	TXUCGBFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit Unicast Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the txunicastframes_gb counter reaches half of the maximum value or the maximum value.</p>
9	TX1024TMAXOCTGBFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit 1024 to Maximum Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the tx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value.</p>
8	TX512T1023OCTGBFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit 512 to 1023 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the tx512to1023octets_gb counter reaches half of the maximum value or the maximum value.</p>
7	TX256T511OCTGBFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit 256 to 511 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the tx256to511octets_gb counter reaches half of the maximum value or the maximum value.</p>
6	TX128T255OCTGBFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit 128 to 255 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the tx128to255octets_gb counter reaches half of the maximum value or the maximum value.</p>
5	TX65T127OCTGBFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit 65 to 127 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the tx65to127octets_gb counter reaches half of the maximum value or the maximum value.</p>
4	TX64OCTGBFIM	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>MMC Transmit 64 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the tx64octets_gb counter</p>

Bits	Name	Memory Access	Description
			reaches half of the maximum value or the maximum value.
3	TXMCGFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Transmit Multicast Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the txmulticastframes_g counter reaches half of the maximum value or the maximum value.
2	TXBCGFIM	R/W	<b>Value After Reset:</b> 0x0 MMC Transmit Broadcast Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the txbroadcastframes_g counter reaches half of the maximum value or the maximum value.
1	TXGBFRMIM	R/W	<b>Value After Reset:</b> 0x0 MMC Transmit Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the txframecount_gb counter reaches half of the maximum value or the maximum value.
0	TXGBOCTIM	R/W	<b>Value After Reset:</b> 0x0 MMC Transmit Good Bad Octet Counter Interrupt Mask Setting this bit masks the interrupt when the txoctetcount_gb counter reaches half of the maximum value or the maximum value.

#### 9.7.2.1.23 Register 69 (Transmit Octet Count for Good and Bad Frames)

**Tx\_Octet\_Count\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x114

**Memory Access:** R

**Value After Reset:** 0x0



This register maintains the number of bytes transmitted in good and bad frames exclusive of preamble and retried bytes.

Bits	Name	Memory Access	Description
31:0	TXOCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of bytes transmitted in good and bad frames exclusive of preamble and retried bytes.

#### 9.7.2.1.24 Register 70 (Transmit Frame Count for Good and Bad Frames)

**Tx\_Frame\_Count\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x118

**Memory Access:** R

**Value After Reset:** 0x0



This register maintains the number of good and bad frames transmitted, exclusive of retried frames.

Bits	Name	Memory Access	Description
31:0	TXFRMGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of good and bad frames transmitted, exclusive of retried frames

#### 9.7.2.1.25 Register 71 (Transmit Frame Count for Good Broadcast Frames)

**Tx\_Broadcast\_Frames\_Good**

**Size:** 32 bits

**Offset:** 0x11c

**Memory Access:** R

**Value After Reset:** 0x0



This register maintains the number of transmitted good broadcast frames.

Bits	Name	Memory Access	Description
31:0	TXBCASTG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of transmitted good broadcast frames.

#### 9.7.2.1.26 Register 72 (Transmit Frame Count for Good Multicast Frames)

**Tx\_Multicast\_Frames\_Good**

**Size:** 32 bits

**Offset:** 0x120

**Memory Access:** R

**Value After Reset:** 0x0



This register maintains the number of transmitted good multicast frames.

Bits	Name	Memory Access	Description
31:0	TXMCASTG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of transmitted good multicast frames.

#### 9.7.2.1.27 Register 73 (Transmit Octet Count for Good and Bad 64 Byte Frames)

**Tx\_64Octets\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x124

**Memory Access:** R

**Value After Reset:** 0x0



This register maintains the number of transmitted good and bad frames with length of 64 bytes, exclusive of preamble and retried frames.

Bits	Name	Memory Access	Description

Bits	Name	Memory Access	Description
31:0	TX64OCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of transmitted good and bad frames with length of 64 bytes, exclusive of preamble and retried frames.

#### 9.7.2.1.28 Register 74 (Transmit Octet Count for Good and Bad 65 to 127 Bytes Frames)

**Tx\_65To127Octets\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x128

**Memory Access:** R

**Value After Reset:** 0x0

31:0	TX65_127OCTGB
------	---------------

This register maintains the number of transmitted good and bad frames with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.

Bits	Name	Memory Access	Description
31:0	TX65_127OCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of transmitted good and bad frames with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.

#### 9.7.2.1.29 Register 75 (Transmit Octet Count for Good and Bad 128 to 255 Bytes Frames)

**Tx\_128To255Octets\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x12c

**Memory Access:** R

**Value After Reset:** 0x0

31:0	TX128_255OCTGB
------	----------------

This register maintains the number of transmitted good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames.

Bits	Name	Memory Access	Description
31:0	TX128_255OCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of transmitted good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames.

#### 9.7.2.1.30 Register 76 (Transmit Octet Count for Good and Bad 256 to 511 Bytes Frames)

**Tx\_256To511Octets\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x130

**Memory Access:** R

**Value After Reset:** 0x0

31:0	TX256_511OCTGB
------	----------------

This register maintains the number of transmitted good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.

Bits	Name	Memory Access	Description
31:0	TX256_511OCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of transmitted good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.

#### 9.7.2.1.31 Register 77 (Transmit Octet Count for Good and Bad 512 to 1023 Bytes Frames)

**Tx\_512To1023Octets\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x134

**Memory Access:** R

**Value After Reset:** 0x0

31:0	TX512_1023OCTGB
------	-----------------

This register maintains the number of transmitted good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble and retried frames.

Bits	Name	Memory Access	Description
31:0	TX512_1023OCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of transmitted good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble and retried frames.

#### 9.7.2.1.32 Register 78 (Transmit Octet Count for Good and Bad 1024 to Maxsize Bytes Frames)

**Tx\_1024ToMaxOctets\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x138

**Memory Access:** R

**Value After Reset:** 0x0

31:0	TX1024_MAXOCTGB
------	-----------------

This register maintains the number of transmitted good and bad frames with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.

Bits	Name	Memory Access	Description
31:0	TX1024_MAXOCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of good and bad frames transmitted with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.

#### 9.7.2.1.33 Register 79 (Transmit Frame Count for Good and Bad Unicast Frames)

**Tx\_Uncast\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x13c

**Memory Access:** R

**Value After Reset:** 0x0

31:0
TXUCASTGB

This register maintains the number of transmitted good and bad unicast frames.

Bits	Name	Memory Access	Description
31:0	TXUCASTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of transmitted good and bad unicast frames.

#### 9.7.2.1.34 Register 80 (Transmit Frame Count for Good and Bad Multicast Frames)

**Tx\_Multicast\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x140

**Memory Access:** R

**Value After Reset:** 0x0

31:0
TXMCASTGB

This register maintains the number of transmitted good and bad multicast frames.

Bits	Name	Memory Access	Description
31:0	TXMCASTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of transmitted good and bad multicast frames.

#### 9.7.2.1.35 Register 81 (Transmit Frame Count for Good and Bad Broadcast Frames)

**Tx\_Broadcast\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x144

**Memory Access:** R

**Value After Reset:** 0x0

31:0
TXBCASTGB

This register maintains the number of transmitted good and bad broadcast frames.

Bits	Name	Memory Access	Description
31:0	TXBCASTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of transmitted good and bad broadcast frames.

#### 9.7.2.1.36 Register 82 (Transmit Frame Count for Underflow Error Frames)

**Tx\_Underflow\_Error\_Frames**

**Size:** 32 bits

**Offset:** 0x148

**Memory Access:** R

**Value After Reset:** 0x0

31:0
TXUNDRFLW

This register maintains the number of frames aborted because of frame underflow error.

Bits	Name	Memory Access	Description
31:0	TXUNDRFLW	R	<b>Value After Reset:</b> 0x0 This field indicates the number of frames aborted because of frame underflow error.

#### 9.7.2.1.37 Register 89 (Transmit Octet Count for Good Frames)

**Tx\_Octet\_Count\_Good**

**Size:** 32 bits

**Offset:** 0x164

**Memory Access:** R

**Value After Reset:** 0x0



This register maintains the number of bytes transmitted, exclusive of preamble, in good frames.

Bits	Name	Memory Access	Description
31:0	TXOCTG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of bytes transmitted, exclusive of preamble, in good frames.

#### 9.7.2.1.38 Register 90 (Transmit Frame Count for Good Frames)

**Tx\_Frame\_Count\_Good**

**Size:** 32 bits

**Offset:** 0x168

**Memory Access:** R

**Value After Reset:** 0x0



This register maintains the number of transmitted good frames, exclusive of preamble.

Bits	Name	Memory Access	Description
31:0	TXFRMG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of transmitted good frames, exclusive of preamble.

#### 9.7.2.1.39 Register 92 (Transmit Frame Count for Good PAUSE Frames)

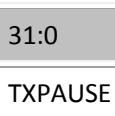
**Tx\_Pause\_Frames**

**Size:** 32 bits

**Offset:** 0x170

**Memory Access:** R

**Value After Reset:** 0x0



This register maintains the number of transmitted good PAUSE frames.

Bits	Name	Memory Access	Description
31:0	TXPAUSE	R	

Bits	Name	Memory Access	Description
31:0	TXPAUSE	R	<b>Value After Reset:</b> 0x0 This field indicates the number of transmitted good PAUSE frames.

#### 9.7.2.1.40 Register 93 (Transmit Frame Count for Good VLAN Frames)

**Tx\_VLAN\_Frames\_Good**

**Size:** 32 bits

**Offset:** 0x174

**Memory Access:** R

**Value After Reset:** 0x0

31:0
TXVLANG

This register maintains the number of transmitted good VLAN frames, exclusive of retried frames.

Bits	Name	Memory Access	Description
31:0	TXVLANG	R	<b>Value After Reset:</b> 0x0 This register maintains the number of transmitted good VLAN frames, exclusive of retried frames.

#### 9.7.2.1.41 Register 94 (Transmit Frame Count for Good Oversize Frames)

**Tx\_OSize\_Frames\_Good**

**Size:** 32 bits

**Offset:** 0x178

**Memory Access:** R

**Value After Reset:** 0x0

31:0
TXOSIZG

This register maintains the number of transmitted good Oversize frames, exclusive of retried frames.

Bits	Name	Memory Access	Description
31:0	TXOSIZG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of frames transmitted without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged frames; 2000 bytes if enabled in bit 27 of Register 0 (MAC Configuration Register)).

#### 9.7.2.1.42 Register 96 (Receive Frame Count for Good and Bad Frames)

**Rx\_Frames\_Count\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x180

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXFRMGB

This register maintains the number of received good and bad frames.

Bits	Name	Memory Access	Description

Bits	Name	Memory Access	Description
31:0	RXFRMGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received good and bad frames.

#### 9.7.2.1.43 Register 97 (Receive Octet Count for Good and Bad Frames)

**Rx\_Octet\_Count\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x184

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXOCTGB

This register maintains the number of bytes received, exclusive of preamble, in good and bad frames.

Bits	Name	Memory Access	Description
31:0	RXOCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of bytes received, exclusive of preamble, in good and bad frames.

#### 9.7.2.1.44 Register 98 (Receive Octet Count for Good Frames)

**Rx\_Octet\_Count\_Good**

**Size:** 32 bits

**Offset:** 0x188

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXOCTG

This register maintains the number of bytes received, exclusive of preamble, only in good frames.

Bits	Name	Memory Access	Description
31:0	RXOCTG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of bytes received, exclusive of preamble, only in good frames.

#### 9.7.2.1.45 Register 99 (Receive Frame Count for Good Broadcast Frames)

**Rx\_Broadcast\_Frames\_Good**

**Size:** 32 bits

**Offset:** 0x18c

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXBCASTG

This register maintains the number of received good broadcast frames.

Bits	Name	Memory Access	Description
31:0	RXBCASTG	R	

Bits	Name	Memory Access	Description
31:0	RXBCASTG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received good broadcast frames.

#### 9.7.2.1.46 Register 100 (Receive Frame Count for Good Multicast Frames)

**Rx\_Multicast\_Frames\_Good**

**Size:** 32 bits

**Offset:** 0x190

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXMCSTG

This register maintains the number of received good multicast frames.

Bits	Name	Memory Access	Description
31:0	RXMCSTG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received good multicast frames.

#### 9.7.2.1.47 Register 101 (Receive Frame Count for CRC Error Frames)

**Rx\_CRC\_Error\_Frames**

**Size:** 32 bits

**Offset:** 0x194

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXCRCERR

This register maintains the number of frames received with CRC error.

Bits	Name	Memory Access	Description
31:0	RXCRCERR	R	<b>Value After Reset:</b> 0x0 This field indicates the number of frames received with CRC error.

#### 9.7.2.1.48 Register 102 (Receive Frame Count for Alignment Error Frames)

**Rx\_Alignment\_Error\_Frames**

**Size:** 32 bits

**Offset:** 0x198

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXALGNERR

This register maintains the number of frames received with alignment (dribble) error. This field is valid only in the 10 or 100 Mbps mode.

Bits	Name	Memory Access	Description
31:0	RXALGNERR	R	<b>Value After Reset:</b> 0x0 This field indicates the number of frames received with alignment (dribble) error. This

Bits	Name	Memory Access	Description
			field is valid only in the 10 or 100 Mbps mode.

#### 9.7.2.1.49 Register 103 (Receive Frame Count for Runt Error Frames)

**Rx\_Runt\_Error\_Frames**

**Size:** 32 bits

**Offset:** 0x19c

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXRUNTERR

This register maintains the number of frames received with runt error(<64 bytes and CRC error).

Bits	Name	Memory Access	Description
31:0	RXRUNTERR	R	<b>Value After Reset:</b> 0x0 This field indicates the number of frames received with runt error(<64 bytes and CRC error).

#### 9.7.2.1.50 Register 104 (Receive Frame Count for Jabber Error Frames)

**Rx\_Jabber\_Error\_Frames**

**Size:** 32 bits

**Offset:** 0x1a0

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXJABERR

This register maintains the number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames.

Bits	Name	Memory Access	Description
31:0	RXJABERR	R	<b>Value After Reset:</b> 0x0 This field indicates the number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames.

#### 9.7.2.1.51 Register 105 (Receive Frame Count for Undersize Frames)

**Rx\_Undersize\_Frames\_Good**

**Size:** 32 bits

**Offset:** 0x1a4

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXUNDERSZG

This register maintains the number of frames received with length less than 64 bytes and without errors.

Bits	Name	Memory Access	Description
31:0	RXUNDERSZG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of frames received with length less than 64 bytes and without errors.

#### 9.7.2.1.52 Register 106 (Receive Frame Count for Oversize Frames)

**Rx\_Oversize\_Frames\_Good**

**Size:** 32 bits

**Offset:** 0x1a8

**Memory Access:** R

**Value After Reset:** 0x0



This register maintains the number of frames received with length greater than the maxsize (1,518 or 1,522 for VLAN tagged frames) and without errors.

Bits	Name	Memory Access	Description
31:0	RXOVERSZG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of frames received without errors, with length greater than the maxsize (1,518 or 1,522 for VLAN tagged frames; 2,000 bytes if enabled in bit 27 of Register 0 (MAC Configuration Register)).

#### 9.7.2.1.53 Register 107 (Receive Frame Count for Good and Bad 64 Byte Frames)

**Rx\_64Octets\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x1ac

**Memory Access:** R

**Value After Reset:** 0x0



This register maintains the number of received good and bad frames with length 64 bytes, exclusive of preamble.

Bits	Name	Memory Access	Description
31:0	RX64OCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received good and bad frames with length 64 bytes, exclusive of preamble.

#### 9.7.2.1.54 Register 108 (Receive Frame Count for Good and Bad 65 to 127 Bytes Frames)

**Rx\_65To127Octets\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x1b0

**Memory Access:** R

**Value After Reset:** 0x0



**RX65\_127OCTGB**

This register maintains the number of received good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble.

Bits	Name	Memory Access	Description
31:0	RX65_127OCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble.

**9.7.2.1.55 Register 109 (Receive Frame Count for Good and Bad 128 to 255 Bytes Frames)****Rx\_128To255Octets\_Frames\_Good\_Bad****Size:** 32 bits**Offset:** 0xb4**Memory Access:** R**Value After Reset:** 0x0

31:0	RX128_255OCTGB
------	----------------

This register maintains the number of received good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble.

Bits	Name	Memory Access	Description
31:0	RX128_255OCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble.

**9.7.2.1.56 Register 110 (Receive Frame Count for Good and Bad 256 to 511 Bytes Frames)****Rx\_256To511Octets\_Frames\_Good\_Bad****Size:** 32 bits**Offset:** 0xb8**Memory Access:** R**Value After Reset:** 0x0

31:0	RX256_511OCTGB
------	----------------

This register maintains the number of received good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble.

Bits	Name	Memory Access	Description
31:0	RX256_511OCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble.

**9.7.2.1.57 Register 111 (Receive Frame Count for Good and Bad 512 to 1,023 Bytes Frames)****Rx\_512To1023Octets\_Frames\_Good\_Bad****Size:** 32 bits**Offset:** 0xbc**Memory Access:** R**Value After Reset:** 0x0

31:0
RX512_1023OCTGB

This register maintains the number of received good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble.

Bits	Name	Memory Access	Description
31:0	RX512_1023OCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble.

#### 9.7.2.1.58 Register 112 (Receive Frame Count for Good and Bad 1,024 to Maxsize Bytes Frames)

**Rx\_1024ToMaxOctets\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x1c0

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RX1024_MAXOCTGB

This register maintains the number of received good and bad frames with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble.

Bits	Name	Memory Access	Description
31:0	RX1024_MAXOCTGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received good and bad frames with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.

#### 9.7.2.1.59 Register 113 (Receive Frame Count for Good Unicast Frames)

**Rx\_Uncast\_Frames\_Good**

**Size:** 32 bits

**Offset:** 0x1c4

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXUCASTG

This register maintains the number of received good unicast frames.

Bits	Name	Memory Access	Description
31:0	RXUCASTG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received good unicast frames.

#### 9.7.2.1.60 Register 114 (Receive Frame Count for Length Error Frames)

**Rx\_Length\_Error\_Frames**

**Size:** 32 bits

**Offset:** 0x1c8

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXLENERR

This register maintains the number of frames received with length error (Length type field not equal to frame size) for all frames with valid length field.

Bits	Name	Memory Access	Description
31:0	RXLENERR	R	<b>Value After Reset:</b> 0x0 This field indicates the number of frames received with length error (Length type field not equal to frame size) for all frames with valid length field.

#### 9.7.2.1.61 Register 115 (Receive Frame Count for Out of Range Frames)

**Rx\_Out\_Of\_Range\_Type\_Frames**

**Size:** 32 bits

**Offset:** 0x1cc

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXOUTOFRNG

This register maintains the number of received frames with length field not equal to the valid frame size (greater than 1,500 but less than 1,536).

Bits	Name	Memory Access	Description
31:0	RXOUTOFRNG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received frames with length field not equal to the valid frame size (greater than 1,500 but less than 1,536).

#### 9.7.2.1.62 Register 116 (Receive Frame Count for PAUSE Frames)

**Rx\_Pause\_Frames**

**Size:** 32 bits

**Offset:** 0x1d0

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXPAUSEFRM

This register maintains the number of received good and valid PAUSE frames.

Bits	Name	Memory Access	Description
31:0	RXPAUSEFRM	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received good and valid PAUSE frames.

#### 9.7.2.1.63 Register 117 (Receive Frame Count for FIFO Overflow Frames)

**Rx\_FIFO\_Overflow\_Frames**

**Size:** 32 bits

**Offset:** 0x1d4

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXFIFOVFL

This register maintains the number of received frames missed because of FIFO overflow.

Bits	Name	Memory Access	Description
31:0	RXFIFOVFL	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received frames missed because of FIFO overflow.

#### 9.7.2.1.64 Register 118 (Receive Frame Count for Good and Bad VLAN Frames)

**Rx\_VLAN\_Frames\_Good\_Bad**

**Size:** 32 bits

**Offset:** 0x1d8

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXVLANFRGB

This register maintains the number of received good and bad VLAN frames.

Bits	Name	Memory Access	Description
31:0	RXVLANFRGB	R	<b>Value After Reset:</b> 0x0 This field indicates the number of received good and bad VLAN frames.

#### 9.7.2.1.65 Register 119 (Receive Frame Count for Watchdog Error Frames)

**Rx\_Watchdog\_Error\_Frames**

**Size:** 32 bits

**Offset:** 0x1dc

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXWDGERR

This register maintains the number of frames received with error because of the watchdog timeout error (frames with more than 2,048 bytes or value programmed in Register 55 (Watchdog Timeout Register)).

Bits	Name	Memory Access	Description
31:0	RXWDGERR	R	<b>Value After Reset:</b> 0x0 This field indicates the number of frames received with error because of the watchdog timeout error (frames with more than 2,048 bytes or value programmed in Register 55 (Watchdog Timeout Register)).

#### 9.7.2.1.66 Register 120 (Receive Frame Count for Receive Error Frames)

**Rx\_Receive\_Error\_Frames**

**Size:** 32 bits

**Offset:** 0x1e0

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXRCVERR

This register maintains the number of frames received with error because of the GMII/MII RXER error.

Bits	Name	Memory Access	Description
31:0	RXRCVERR	R	<b>Value After Reset:</b> 0x0 This field indicates the number of frames received with error because of the GMII/MII RXER error or Frame Extension error on GMII.

#### 9.7.2.1.67 Register 121 (Receive Frame Count for Good Control Frames Frames)

##### Rx\_Control\_Frames\_Good

**Size:** 32 bits

**Offset:** 0x1e4

**Memory Access:** R

**Value After Reset:** 0x0

31:0
RXCTRLG

This register maintains the number of good control frames received.

Bits	Name	Memory Access	Description
31:0	RXCTRLG	R	<b>Value After Reset:</b> 0x0 This field indicates the number of good control frames received.

#### 9.7.2.1.68 Register 256 (Layer 3 and Layer 4 Control Register 0)

##### L3\_L4\_Control0

**Size:** 32 bits

**Offset:** 0x400

**Memory Access:** R/W

**Value After Reset:** 0x0

31:22	21	20	19	18	17	16	15:11	10:6	5	4	3	2	1	0
Reserved_31_22	L4DPI_M0	L4DP_M0	L4SPI_M0	L4SP_M0	Reserve_d_17	L4PE_N0	L3HD_BM0	L3HS_BM0	L3DAI_M0	L3DA_M0	L3SAI_M0	L3SA_M0	Reserv_ed_1	L3PE_N0

This register controls the operations of the filter 0 of Layer 3 and Layer 4. This register is reserved if the Layer 3 and Layer 4 Filtering feature is not selected during core configuration.

Bits	Name	Memory Access	Description
31:22	Reserved_31_22	R	<b>Value After Reset:</b> 0x0 Reserved
21	L4DPIM0	R/W	<b>Value After Reset:</b> 0x0 Layer 4 Destination Port Inverse Match Enable When set, this bit indicates that the Layer 4 Destination Port number field is enabled for inverse matching. When reset, this bit indicates that the Layer 4 Destination Port number field is enabled for perfect matching. This bit is valid and applicable only when Bit 20 (L4DPM0) is set high.
20	L4DPM0	R/W	<b>Value After Reset:</b> 0x0 Layer 4 Destination Port Match Enable

Bits	Name	Memory Access	Description
			When set, this bit indicates that the Layer 4 Destination Port number field is enabled for matching. When reset, the MAC ignores the Layer 4 Destination Port number field for matching.
19	L4SPIMO	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Layer 4 Source Port Inverse Match Enable</p> <p>When set, this bit indicates that the Layer 4 Source Port number field is enabled for inverse matching. When reset, this bit indicates that the Layer 4 Source Port number field is enabled for perfect matching.</p> <p>This bit is valid and applicable only when Bit 18 (L4SPM0) is set high.</p>
18	L4SPM0	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Layer 4 Source Port Match Enable</p> <p>When set, this bit indicates that the Layer 4 Source Port number field is enabled for matching. When reset, the MAC ignores the Layer 4 Source Port number field for matching.</p>
17	Reserved_17	R	<p><b>Value After Reset:</b> 0x0</p> <p>Reserved</p>
16	L4PENO	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Layer 4 Protocol Enable</p> <p>When set, this bit indicates that the Source and Destination Port number fields for UDP frames are used for matching. When reset, this bit indicates that the Source and Destination Port number fields for TCP frames are used for matching.</p> <p>The Layer 4 matching is done only when either L4SPM0 or L4DPM0 bit is set high.</p>
15:11	L3HDBM0	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Layer 3 IP DA Higher Bits Match</p> <p>IPv4 Frames:</p> <p>This field contains the number of higher bits of IP Destination Address that are matched in the IPv4 frames. The following list describes the values of this field:</p> <ul style="list-style-type: none"> <li>0: No bits are masked.</li> <li>1: LSb[0] is masked.</li> <li>2: Two LSbs [1:0] are masked.</li> <li>...</li> <li>31: All bits except MSb are masked.</li> </ul> <p>IPv6 Frames:</p> <p>Bits [12:11] of this field correspond to Bits [6:5] of L3HSBM0, which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 frames. The following list describes the concatenated values of the L3HDBM0[1:0] and L3HSBM0 bits:</p> <ul style="list-style-type: none"> <li>0: No bits are masked.</li> <li>1: LSb[0] is masked.</li> <li>2: Two LSbs [1:0] are masked.</li> <li>...</li> <li>127: All bits except MSb are masked.</li> </ul> <p>This field is valid and applicable only if L3DAM0 or L3SAM0 is set high.</p>
10:6	L3HSBM0	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Layer 3 IP SA Higher Bits Match</p> <p>IPv4 Frames:</p> <p>This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 frames. The following list describes the values of this field:</p>

Bits	Name	Memory Access	Description
			<p>0: No bits are masked.      1: LSb[0] is masked.      2: Two LSbs [1:0] are masked.      ...      31: All bits except MSb are masked.</p> <p><b>IPv6 Frames:</b>      This field contains Bits [4:0] of the field that indicates the number of higher bits of IP Source or Destination Address matched in the IPv6 frames. This field is valid and applicable only if L3DAM0 or L3SAM0 is set high.</p>
5	L3DAIM0	R/W	<p><b>Value After Reset:</b> 0x0  <b>Layer 3 IP DA Inverse Match Enable</b>      When set, this bit indicates that the Layer 3 IP Destination Address field is enabled for inverse matching. When reset, this bit indicates that the Layer 3 IP Destination Address field is enabled for perfect matching.      This bit is valid and applicable only when Bit 4 (L3DAM0) is set high.</p>
4	L3DAM0	R/W	<p><b>Value After Reset:</b> 0x0  <b>Layer 3 IP DA Match Enable</b>      When set, this bit indicates that Layer 3 IP Destination Address field is enabled for matching. When reset, the MAC ignores the Layer 3 IP Destination Address field for matching.      Note: When Bit 0 (L3PENO) is set, you should set either this bit or Bit 2 (L3SAM0) because either IPv6 DA or SA can be checked for filtering.</p>
3	L3SAIM0	R/W	<p><b>Value After Reset:</b> 0x0  <b>Layer 3 IP SA Inverse Match Enable</b>      When set, this bit indicates that the Layer 3 IP Source Address field is enabled for inverse matching. When reset, this bit indicates that the Layer 3 IP Source Address field is enabled for perfect matching.      This bit is valid and applicable only when Bit 2 (L3SAM0) is set high.</p>
2	L3SAM0	R/W	<p><b>Value After Reset:</b> 0x0  <b>Layer 3 IP SA Match Enable</b>      When set, this bit indicates that the Layer 3 IP Source Address field is enabled for matching. When reset, the MAC ignores the Layer 3 IP Source Address field for matching.      Note: When Bit 0 (L3PENO) is set, you should set either this bit or Bit 4 (L3DAM0) because either IPv6 SA or DA can be checked for filtering.</p>
1	Reserved_1	R	<p><b>Value After Reset:</b> 0x0      Reserved</p>
0	L3PENO	R/W	<p><b>Value After Reset:</b> 0x0  <b>Layer 3 Protocol Enable</b>      When set, this bit indicates that the Layer 3 IP Source or Destination Address matching is enabled for the IPv6 frames. When reset, this bit indicates that the Layer 3 IP Source or Destination Address matching is enabled for the IPv4 frames. The Layer 3 matching is done only when either L3SAM0 or L3DAM0 bit is set high.</p>

#### 9.7.2.1.69 Register 257 (Layer 4 Address Register 0)

**Layer4\_Address0**

**Size:** 32 bits

**Offset:** 0x404

**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
L4DPO	L4SPO

You can configure the Layer 3 and Layer 4 Address Registers to be double-synchronized by selecting the Synchronize Layer 3 and Layer 4 Address Registers to Rx Clock Domain option in coreConsultant. If the Layer 3 and Layer 4 Address Registers are configured to be double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock.

If the Layer 3 and Layer 4 Filtering feature is not selected during core configuration, this register and registers 260 through 299 are reserved (RO with default value).

Bits	Name	Memory Access	Description
31:16	L4DPO	R/W	<b>Value After Reset:</b> 0x0 Layer 4 Destination Port Number Field When Bit 16 (L4PENO) is reset and Bit 20 (L4DPM0) is set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 frames. When Bit 16 (L4PENO) and Bit 20 (L4DPM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 frames.
15:0	L4SPO	R/W	<b>Value After Reset:</b> 0x0 Layer 4 Source Port Number Field Layer 4 Source Port Number Field When Bit 16 (L4PENO) is reset and Bit 20 (L4DPM0) is set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 frames. When Bit 16 (L4PENO) and Bit 20 (L4DPM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 frames.

#### 9.7.2.1.70 Register 260 (Layer 3 Address 0 Register 0)

**Layer3\_Adr0\_Reg0****Size:** 32 bits**Offset:** 0x410**Memory Access:** R/W**Value After Reset:** 0x0

31:0
L3A00

For IPv4 frames, the Layer 3 Address 0 Register 0 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

Bits	Name	Memory Access	Description
31:0	L3A00	R/W	<b>Value After Reset:</b> 0x0 Layer 3 Address 0 Field When Bit 0 (L3PENO) and Bit 2 (L3SAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 frames.

Bits	Name	Memory Access	Description
			<p>When Bit 0 (L3PENO) and Bit 4 (L3DAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits [31:0] of the IP Destination Address field in the IPv6 frames.</p> <p>When Bit 0 (L3PENO) is reset and Bit 2 (L3SAM0) is set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the IP Source Address field in the IPv4 frames.</p>

#### 9.7.2.1.71 Register 261 (Layer 3 Address 1 Register 0)

**Layer3\_Adr1\_Reg0**

**Size:** 32 bits

**Offset:** 0x414

**Memory Access:** R/W

**Value After Reset:** 0x0

31:0

L3A10

For IPv4 frames, the Layer 3 Address 1 Register 0 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

Bits	Name	Memory Access	Description
31:0	L3A10	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Layer 3 Address 1 Field</p> <p>When Bit 0 (L3PENO) and Bit 2 (L3SAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits [63:32] of the IP Source Address field in the IPv6 frames.</p> <p>When Bit 0 (L3PENO) and Bit 4 (L3DAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits [31:0] of the IP Destination Address field in the IPv6 frames.</p> <p>When Bit 0 (L3PENO) is reset and Bit 2 (L3SAM0) is set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the IP Destination Address field in the IPv4 frames.</p>

#### 9.7.2.1.72 Register 262 (Layer 3 Address 2 Register 0)

**Layer3\_Adr2\_Reg0**

**Size:** 32 bits

**Offset:** 0x418

**Memory Access:** R/W

**Value After Reset:** 0x0

31:0

L3A20

For IPv4 frames, the Layer 3 Address 2 Register 0 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.

Bits	Name	Memory Access	Description
31:0	L3A20	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Layer 3 Address 2 Field</p> <p>When Bit 0 (L3PENO) and Bit 2 (L3SAM0) are set in Register 256 (Layer 3 and Layer 4 Control</p>

Bits	Name	Memory Access	Description
			<p>Register 0), this field contains the value to be matched with Bits [95:64] of the IP Source Address field in the IPv6 frames.</p> <p>When Bit 0 (L3PENO) and Bit 4 (L3DAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains value to be matched with Bits [95:64] of the IP Destination Address field in the IPv6 frames.</p> <p>When Bit 0 (L3PENO) is reset in Register 256 (Layer 3 and Layer 4 Control Register 0), this register is not used.</p>

#### 9.7.2.1.73 Register 263 (Layer 3 Address 3 Register 0)

**Layer3\_Adr3\_Reg0**

**Size:** 32 bits

**Offset:** 0x41c

**Memory Access:** R/W

**Value After Reset:** 0x0

31:0

L3A30

For IPv4 frames, the Layer 3 Address 3 Register 0 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.

Bits	Name	Memory Access	Description
31:0	L3A30	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Layer 3 Address 3 Field</p> <p>When Bit 0 (L3PENO) and Bit 2 (L3SAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits [127:96] of the IP Source Address field in the IPv6 frames.</p> <p>When Bit 0 (L3PENO) and Bit 4 (L3DAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits [127:96] of the IP Destination Address field in the IPv6 frames.</p> <p>When Bit 0 (L3PENO) is reset in Register 256 (Layer 3 and Layer 4 Control Register 0), this register is not used.</p>

- Registers 268, 280, and 292 are similar to Register 256 (Layer 3 and Layer 4 Control Register 0).
- Registers 269, 281, and 293 are similar to Register 257 (Layer 4 Address Register 0).
- Registers 272, 284, and 296 are similar to Register 260 (Layer 3 Address 0 Register 0).
- Registers 273, 285, and 297 are similar to Register 261 (Layer 3 Address 1 Register 0).
- Registers 274, 286, and 298 are similar to Register 262 (Layer 3 Address 2 Register 0).
- Registers 275, 287, and 299 are similar to Register 263 (Layer 3 Address 3 Register 0).
- Registers 268 through 275 are present—
- Registers 280 through 287 are present
- Registers 292 through 299 are present

#### 9.7.2.1.74 Register 320 (Hash Table Register 0)

**Hash\_Table\_Reg0**

**Size:** 32 bits

**Offset:** 0x500

**Memory Access:** R/W

**Value After Reset:** 0x0

31:0
HT31T0

This register contains the first 32 bits of the hash table when the width of the Hash table is 128 bits or 256 bits. You can specify the width of the hash table by using the Hash Table Size option in coreConsultant.

The 128-bit or 256-bit Hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming frame is passed through the CRC logic and the upper seven (eight in 256-bit Hash) bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register X), and the least significant five bits determine the bit within the register. For example, a hash value of 7b'1100000 (in 128-bit Hash) selects Bit 0 of the Hash Table Register 3 and a value of 8b'10111111 (in 256-bit Hash) selects Bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 (or 8) bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1'b1, the frame is accepted. Otherwise, it is rejected. If the Bit 1 (Pass All Multicast) is set in Register 1 (MAC Frame Filter), then all multicast frames are accepted regardless of the multicast hash values. If the Hash Table register is configured to be double-synchronized to the (G)MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Hash Table Register X registers are written.

Note: If double-synchronization is enabled, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

Bits	Name	Memory Access	Description
31:0	HT31T0	R/W	<b>Value After Reset:</b> 0x0 First 32 bits of Hash Table This field contains the first 32 Bits (31:0) of the Hash table.

#### 9.7.2.1.75 Register 321 (Hash Table Register 1)

**Hash\_Table\_Reg1**

**Size:** 32 bits

**Offset:** 0x504

**Memory Access:** R/W

**Value After Reset:** 0x0

31:0
HT63T32

This register contains the second 32 bits of the hash table when the width of the Hash table is 128 bits or 256 bits. You can specify the width of the hash table by using the Hash Table Size option in coreConsultant.

Bits	Name	Memory Access	Description
31:0	HT63T32	R/W	<b>Value After Reset:</b> 0x0 Second 32 bits of Hash Table This field contains the second 32 Bits (63:32) of the Hash table.

#### 9.7.2.1.76 Register 322 (Hash Table Register 2)

**Hash\_Table\_Reg2**

**Size:** 32 bits

**Offset:** 0x508

**Memory Access:** R/W

**Value After Reset:** 0x0

31:0
HT95T64

This register contains the third 32 bits of the hash table when the width of the Hash table is 128 bits or 256 bits. You can specify the width of the hash table by using the Hash Table Size option in coreConsultant.

Bits	Name	Memory Access	Description
31:0	HT95T64	R/W	<b>Value After Reset:</b> 0x0 Third 32 bits of Hash Table This field contains the third 32 Bits (95:64) of the Hash table.

#### 9.7.2.1.77 Register 323 (Hash Table Register 3)

**Hash\_Table\_Reg3**

**Size:** 32 bits

**Offset:** 0x50c

**Memory Access:** R/W

**Value After Reset:** 0x0

31:0
HT127T96

This register contains the fourth 32 bits of the hash table when the width of the Hash table is 128 bits or 256 bits. You can specify the width of the hash table by using the Hash Table Size option in coreConsultant.

Bits	Name	Memory Access	Description
31:0	HT127T96	R/W	<b>Value After Reset:</b> 0x0 Fourth 32 bits of Hash Table This field contains the third 32 Bits (127:96) of the Hash table.

#### 9.7.2.1.78 Register 353 (VLAN Tag Inclusion or Replacement Register)

**VLAN\_Incl\_Reg**

**Size:** 32 bits

**Offset:** 0x584

**Memory Access:** R/W

31:20	19	18	17:16	15:0
Reserved_31_20	CSVL	VLP	VLC	VLT

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the transmit frames. This register is present only when the Enable SA, VLAN, and CRC Insertion on TX option is selected during core configuration.

Bits	Name	Memory Access	Description
31:20	Reserved_31_20	R	<b>Value After Reset:</b> 0x0 Reserved
19	CSVL	R/W	<b>Value After Reset:</b> 0x0 C-VLAN or S-VLAN When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted frames. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the transmitted frames.
18	VLP	R/W	<b>Value After Reset:</b> 0x0 VLAN Priority Control

Bits	Name	Memory Access	Description
			When this bit is set, the control Bits [17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, the mti_vlan_ctrl_i control input is used, and Bits [17:16] are ignored.
17:16	VLC	R/W	<p>VLAN Tag Control in Transmit Frames</p> <p>2'b00: No VLAN tag deletion, insertion, or replacement</p> <p>2'b01: VLAN tag deletion</p> <p>The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted frames with VLAN tags.</p> <p>2'b10: VLAN tag insertion</p> <p>The MAC inserts VLT in bytes 15 and 16 of the frame after inserting the Type value (0x8100/0x88a8) in bytes 13 and 14. This operation is performed on all transmitted frames, irrespective of whether they already have a VLAN tag.</p> <p>2'b11: VLAN tag replacement</p> <p>The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted frames (Bytes 13 and 14 are 0x8100/0x88a8).</p> <p>Note: Changes to this field take effect only on the start of a frame. If you write this register field when a frame is being transmitted, only the subsequent frame can use the updated value, that is, the current frame does not use the updated value.</p>
15:0	VLT	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>VLAN Tag for Transmit Frames</p> <p>This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority, Bit 12 is the CFI/DEI, and Bits[11:0] are the VLAN tag's VID field.</p>

#### 9.7.2.1.79 Register 354 (VLAN Hash Table Register)

##### VLAN\_Hash\_Table\_Reg

**Size:** 32 bits

**Offset:** 0x588

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:0
Reserved_31_16	VLHT

The 16-bit Hash table is used for group address filtering based on VLAN tag when Bit 18 (VTHM) of Register 7 (VLAN Tag Register) is set. For hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on Bit 16 (ETV) of VLAN Tag Register) in the incoming frame is passed through the CRC logic and the upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a hash value of 4b'1000 selects Bit 8 of the VLAN Hash table. The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper four bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1'b1, the frame is accepted. Otherwise, it is rejected. If the Hash Table register is configured to be double-synchronized to the (G)MII clock domain, the synchronization is triggered only when Bits[15:8] (in little-endian mode) or Bits[7:0] (in big-endian mode) of this register are written.

**Notes:**

If double-synchronization is enabled, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

To help you program the hash table, a sample C routine that generates a VLAN tag's 4-bit hash is included in /sample\_codes/ directory of your workspace.

Bits	Name	Memory Access	Description
31:16	Reserved_31_16	R	<b>Value After Reset:</b> 0x0 Reserved
15:0	VLHT	R/W	<b>Value After Reset:</b> 0x0 VLAN Hash Table This field contains the 16-bit VLAN Hash Table.

### 9.7.2.2 DMA Register Description

#### 9.7.2.2.1 Register 0 (Bus Mode Register)

**Bus\_Mode**

**Size:** 32 bits

**Offset:** 0x1000

**Memory Access:** R/W

**Value After Reset:** 0x20101

31	30	29:28	27	26	25	24	23	22:17	16	15:14	13:8	7	6:2	1	0
RIB	Reserved_30	PRWG	TXPR	MB	AAL	PBLx8	USP	RPBL	FB	PR	PBL	ATDS	DSL	DA	SWR

The Bus Mode register establishes the bus operating modes for the DMA.

Bits	Name	Memory Access	Description
31	RIB	R/W	<b>Value After Reset:</b> 0x0 Rebuild INCRx Burst When this bit is set high and the AHB master gets an EBT (Retry, Split, or Losing bus grant), the AHB master interface rebuilds the pending beats of any burst transfer initiated with INCRx. The AHB master interface rebuilds the beats with a combination of specified bursts with INCRx and SINGLE. By default, the AHB master interface rebuilds pending beats of an EBT with an unspecified (INCR) burst. This bit is valid only in the GMAC-AHB configuration. It is reserved in all other configuration.
30	Reserved_30	R	<b>Value After Reset:</b> 0x0 Reserved
29:28	PRWG	R	<b>Value After Reset:</b> 0x0 Channel Priority Weights This field sets the priority weights for Channel 0 during the round-robin arbitration between the DMA channels for the system bus. 00: The priority weight is 1. 01: The priority weight is 2. 10: The priority weight is 3. 11: The priority weight is 4. This field is present in all DWC_gmac configurations except GMAC-AXI when you select the AV feature. Otherwise, this field is reserved and read-only (RO).
27	TXPR	R/W	<b>Value After Reset:</b> 0x0 Transmit Priority When set, this bit indicates that the transmit DMA has higher priority than the receive DMA during arbitration for the system-side bus. In the GMAC-AXI configuration, this bit is reserved and read-only (RO).
26	MB	R/W	<b>Value After Reset:</b> 0x0

Bits	Name	Memory Access	Description
			<p>Mixed Burst When this bit is set high and the FB bit is low, the AHB Master interface starts all bursts of length more than 16 with INCR (undefined burst) whereas it reverts to fixed burst transfers (INCRx and SINGLE) for burst length of 16 and less. This bit is valid only in the GMAC-AHB configuration and reserved in all other configuration.</p>
25	AAL	R/W	<p><b>Value After Reset:</b> 0x0 Address Aligned Beats When this bit is set high and the FB bit is equal to 1, the AHB or AXI interface generates all bursts aligned to the start address LS bits. If the FB bit is equal to 0, the first burst (accessing the data buffer's start address) is not aligned, but subsequent bursts are aligned to the address. This bit is valid only in the GMAC-AHB and GMAC-AXI configuration and is reserved (RO with default value 0) in all other configurations.</p>
24	PBLx8	R/W	<p><b>Value After Reset:</b> 0x0 PBLx8 Mode When set high, this bit multiplies the programmed PBL value (Bits[22:17] and Bits[13:8]) eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value. Note: This bit function is not backward compatible. Before release 3.50a, this bit was 4xPBL.</p>
23	USP	R/W	<p><b>Value After Reset:</b> 0x0 Use Separate PBL When set high, this bit configures the Rx DMA to use the value configured in Bits[22:17] as PBL. The PBL value in Bits[13:8] is applicable only to the Tx DMA operations. When reset to low, the PBL value in Bits[13:8] is applicable for both DMA engines.</p>
22:17	RPBL	R/W	<p><b>Value After Reset:</b> 0x1 Rx DMA PBL This field indicates the maximum number of beats to be transferred in one Rx DMA transaction. This is the maximum value that is used in a single block Read or Write. The Rx DMA always attempts to burst as specified in the RPBL bit each time it starts a Burst transfer on the host bus. You can program RPBL with values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. This field is valid and applicable only when USP is set high.</p>
16	FB	R/W	<p><b>Value After Reset:</b> 0x0 Fixed Burst This bit controls whether the AHB or AXI Master interface performs fixed burst transfers or not. When set, the AHB interface uses only SINGLE, INCR4, INCR8, or INCR16 during start of the normal burst transfers. When reset, the AHB or AXI interface uses SINGLE and INCR burst transfer operations. For more information, see Bit 0 (UNDEF) of the AXI Bus Mode register in the GMAC-AXI configuration.</p>
15:14	PR	R/W	<p><b>Value After Reset:</b> 0x0 Priority Ratio These bits control the priority ratio in the weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when Bit 1 (DA) is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether Bit 27 (TXPR) is reset or set.</p>

Bits	Name	Memory Access	Description
			<p>00: The Priority Ratio is 1:1.      01: The Priority Ratio is 2:1.      10: The Priority Ratio is 3:1.      11: The Priority Ratio is 4:1.      In the GMAC-AXI configuration, these bits are reserved and read-only (RO).</p>
13:8	PBL	R/W	<p><b>Value After Reset:</b> 0x1  <b>Programmable Burst Length</b>      These bits indicate the maximum number of beats to be transferred in one DMA transaction. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a Burst transfer on the host bus. PBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. When USP is set high, this PBL value is applicable only for Tx DMA transactions. If the number of beats to be transferred is more than 32, then perform the following steps:      1. Set the PBLx8 mode.      2. Set the PBL.      For example, if the maximum number of beats to be transferred is 64, then first set PBLx8 to 1 and then set PBL to 8. The PBL values have the following limitation: The maximum number of possible beats (PBL) is limited by the size of the Tx FIFO and Rx FIFO in the MTL layer and the data bus width on the DMA. The FIFO has a constraint that the maximum beat supported is half the depth of the FIFO, except when specified. For different data bus widths and FIFO sizes, the valid PBL range (including x8 mode) is provided in the following list. If the PBL is common for both transmit and receive DMA, the minimum Rx FIFO and Tx FIFO depths must be considered.      Note: In the half-duplex mode, the valid PBL range specified in the following list is applicable only for Tx FIFO.      * 32-Bit Data Bus Width      128 Bytes FIFO Depth: In the full-duplex mode, the valid PBL range is 16 or less. In the half-duplex mode, the valid PBL range is 8 or less for the 10 or 100 Mbps mode.      256 Bytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 32 or less.      512 Bytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 64 or less.      1 KB FIFO Depth: In the full-duplex mode, the valid PBL range is 128 or less. In the half-duplex mode, the valid PBL range is 128 or less in the 10 or 100 Mbps mode and 64 or less in the 1000 Mbps mode.      2 KB and Higher FIFO Depth: All PBL values are supported in the full-duplex mode and half-duplex modes.      * 64-Bit Data Bus Width      128 Bytes FIFO Depth: In the full-duplex mode, the valid PBL range is 8 or less. In the half-duplex mode, the valid PBL range is 4 or less for the 10 or 100 Mbps mode.      256 Bytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 16 or less.      512 Bytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 32 or less.      1 KB FIFO Depth: In the full-duplex mode, the valid PBL range is 64 or less. In the half-duplex mode, the valid PBL range is 64 or less in the 10 or 100 Mbps mode and 32 or less in the 1000-Mbps mode.      2 KB FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 128 or less.      4 KB and Higher FIFO Depth: All PBL values are supported in the full-duplex and   </p>

Bits	Name	Memory Access	Description
			<p>half-duplex modes.</p> <p>* 128-Bit Data Bus Width</p> <p>128 Bytes FIFO Depth: In the full-duplex mode, the valid PBL range is 4 or less. In the half-duplex mode, the valid PBL range is 2 or less for the 10 or 100 Mbps mode.</p> <p>256 Bytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 8 or less.</p> <p>512 Bytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 16 or less.</p> <p>1 KB FIFO Depth: In the full-duplex mode, the valid PBL range is 32 or less. In the half-duplex mode, the valid PBL range is 32 or less in the 10 or 100 Mbps mode and 16 or less in the 1000-Mbps mode.</p> <p>2 KB FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 64 or less.</p> <p>4 KB FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 128 or less.</p> <p>8 KB and Higher FIFO Depth: All PBL values are supported in the full-duplex and half-duplex modes.</p>
7	ATDS	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Alternate Descriptor Size</p> <p>When set, the size of the alternate descriptor increases to 32 bytes (8 DWORDS). This is required when the Advanced Timestamp feature or the IPC Full Offload Engine (Type 2) is enabled in the receiver. The enhanced descriptor is not required if the Advanced Timestamp and IPC Full Checksum Offload (Type 2) features are not enabled. In such cases, you can use the 16 bytes descriptor to save 4 bytes of memory.</p> <p>This bit is present only when you select the Alternate Descriptor feature and any one of the following features during core configuration:</p> <ul style="list-style-type: none"> <li>Advanced Timestamp feature</li> <li>IPC Full Checksum Offload Engine (Type 2) feature</li> </ul> <p>Otherwise, this bit is reserved and read-only.</p> <p>When reset, the descriptor size reverts back to 4 DWORDs (16 bytes). This bit preserves the backward compatibility for the descriptor size. In versions prior to 3.50a, the descriptor size is 16 bytes for both normal and enhanced descriptor. In version 3.50a, descriptor size is increased to 32 bytes because of the Advanced Timestamp and IPC Full Checksum Offload Engine (Type 2) features.</p>
6:2	DSL	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Descriptor Skip Length</p> <p>This bit specifies the number of Word, Dword, or Lword (depending on the 32-bit, 64-bit, or 128-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of current descriptor to the start of next descriptor. When the DSL value is equal to zero, the descriptor table is taken as contiguous by the DMA in Ring mode.</p>
1	DA	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>DMA Arbitration Scheme</p> <p>This bit specifies the arbitration scheme between the transmit and receive paths of Channel 0.</p> <p>0: Weighted round-robin with Rx:Tx or Tx:Rx</p> <p>The priority between the paths is according to the priority specified in bits 15:14 (PR) and priority weights specified in Bit 27 (TXPR).</p> <p>1: Fixed priority</p> <p>The transmit path has priority over receive path when Bit 27 (TXPR) is set. Otherwise,</p>

Bits	Name	Memory Access	Description
			receive path has priority over the transmit path. In the GMAC-AXI configuration, these bits are reserved and read-only (RO).
0	SWR	R/W	<p><b>Value After Reset:</b> 0x1  <b>Software Reset</b>  When this bit is set, the MAC DMA Controller resets the logic and all internal registers of the MAC. It is cleared automatically after the reset operation has completed in all of the DWC_gmac clock domains. Before reprogramming any register of the DWC_gmac, you should read a zero (0) value in this bit .</p> <p><b>Note:</b>  The Software reset function is driven only by this bit. Bit 0 of Register 64 (Channel 1 Bus Mode Register) or Register 128 (Channel 2 Bus Mode Register) has no impact on the Software reset function.</p> <p>The reset operation is completed only when all resets in all active clock domains are de-asserted. Therefore, it is essential that all the PHY inputs clocks (applicable for the selected PHY interface) are present for the software reset completion.</p>

#### 9.7.2.2.2 Register 1 (Transmit Poll Demand Register)

##### Transmit\_Poll\_Demand

**Size:** 32 bits

**Offset:** 0x1004

**Memory Access:** R/W

**Value After Reset:** 0x0

31:0
TPD

The Transmit Poll Demand register enables the Tx DMA to check whether or not the DMA owns the current descriptor. The Transmit Poll Demand command is given to wake up the Tx DMA if it is in the Suspend mode. The Tx DMA can go into the Suspend mode because of an Underflow error in a transmitted frame or the unavailability of descriptors owned by it. You can give this command anytime and the Tx DMA resets this command when it again starts fetching the current descriptor from host memory. When this register is read, it always returns zero.

Bits	Name	Memory Access	Description
31:0	TPD	R/W	<p><b>Value After Reset:</b> 0x0  <b>Transmit Poll Demand</b>  When these bits are written with any value, the DMA reads the current descriptor pointed to by Register 18 (Current Host Transmit Descriptor Register). If that descriptor is not available (owned by the Host), the transmission returns to the Suspend state and the Bit 2 (TU) of Register 5 (Status Register) is asserted. If the descriptor is available, the transmission resumes.</p>

#### 9.7.2.2.3 Register 2 (Receive Poll Demand Register)

##### Receive\_Poll\_Demand

**Size:** 32 bits

**Offset:** 0x1008

**Memory Access:** R/W

**Value After Reset:** 0x0

31:0
RPD

The Receive Poll Demand register enables the receive DMA to check for new descriptors. This command is used to wake up the Rx DMA from the SUSPEND state. The RxDMA can go into the SUSPEND state only because of the unavailability of descriptors it owns. When this register is read, it always returns zero.

Bits	Name	Memory Access	Description
31:0	RPD	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Receive Poll Demand</p> <p>When these bits are written with any value, the DMA reads the current descriptor pointed to by Register 19 (Current Host Receive Descriptor Register). If that descriptor is not available (owned by the Host), the reception returns to the Suspended state and the Bit 7 (RU) of Register 5 (Status Register) is not asserted. If the descriptor is available, the Rx DMA returns to the active state.</p>

#### 9.7.2.2.4 Register 3 (Receive Descriptor List Address Register)

##### Receive\_Descriptor\_List\_Address

**Size:** 32 bits

**Offset:** 0x100c

**Memory Access:** R/W

**Value After Reset:** 0x0

31:2	1:0
RDESLA_32bit	Reserved_1_0

The Receive Descriptor List Address register points to the start of the Receive Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given.

You can write to this register only when Rx DMA has stopped, that is, Bit 1 (SR) is set to zero in Register 6 (Operation Mode Register). When stopped, this register can be written with a new descriptor list address. When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

If this register is not changed when the SR bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

Bits	Name	Memory Access	Description
31:2	RDESLA_32bit	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Start of Receive List</p> <p>This field contains the base address of the first descriptor in the Receive Descriptor list. The LSB bits (1:0) for 32-bit bus width are ignored and internally taken as all-zero by the DMA. Therefore, these LSB bits are read-only (RO).</p>
1:0	Reserved_1_0	R	<p><b>Value After Reset:</b> 0x0</p> <p>Reserved</p>

#### 9.7.2.2.5 Register 4 (Transmit Descriptor List Address Register)

##### Transmit\_Descriptor\_List\_Address

**Size:** 32 bits

**Offset:** 0x1010

**Memory Access:** R/W

**Value After Reset:** 0x0

31:2	1:0
------	-----

TDESLA_32bit	Reserved_1_0
--------------	--------------

The Transmit Descriptor List Address register points to the start of the Transmit Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

You can write to this register only when the Tx DMA has stopped, that is, Bit 13 (ST) is set to zero in Register 6 (Operation Mode Register). When stopped, this register can be written with a new descriptor list address. When you set the ST bit to 1, the DMA takes the newly programmed descriptor base address.

If this register is not changed when the ST bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

Bits	Name	Memory Access	Description
31:2	TDESLA_32bit	R/W	<b>Value After Reset:</b> 0x0 Start of Transmit List This field contains the base address of the first descriptor in the Transmit Descriptor list. The LSB bits (1:0) for 32-bit bus width are ignored and are internally taken as all-zero by the DMA. Therefore, these LSB bits are read-only (RO).
1:0	Reserved_1_0	R	<b>Value After Reset:</b> 0x0 Reserved

#### 9.7.2.2.6 Register 5 (Status Register)

##### Status

**Size:** 32 bits

**Offset:** 0x1014

**Memory Access:** R/W

**Value After Reset:** 0x0

31	30	29	28	27	26	25:23	22:20	19:17	16	15	14	13	12:11	10	9	8	7	6	5	4	3	2	1	0
Reserve d_31	GL PII	T TI	G PI	G M I	G L I	EB	TS	RS	N IS	A IS	E RI	F B I	Reserved_12_11	E TI	R W T	R P S	R U I	R I F	U N F	O VF	T J T	T U	T P S	T I

The Status register contains all status bits that the DMA reports to the host. The Software driver reads this register during an interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted. The bits of this register are not cleared when read. Writing 1'b1 to (unreserved) Bits[16:0] of this register clears these bits and writing 1'b0 has no effect. Each field (Bits[16:0]) can be masked by masking the appropriate bit in Register 7 (Interrupt Enable Register).

Bits	Name	Memory Access	Description
31	Reserved_31	R	<b>Value After Reset:</b> 0x0 Reserved
30	GLPII	R	<b>Value After Reset:</b> 0x0 GMAC LPI Interrupt (for Channel 0) This bit indicates an interrupt event in the LPI logic of the DWC_gmac. To reset this bit to 1'b0, the software must read the corresponding registers in the DWC_gmac to get the exact cause of the interrupt and clear its source. Note: GLPII status is given only in Channel 0 DMA register and is applicable only when the Energy Efficient Ethernet feature is enabled. Otherwise, this bit is reserved. When this bit is high, the interrupt signal from the MAC (sbd_intr_o) is high.

Bits	Name	Memory Access	Description
29	TTI	R	<p><b>Value After Reset:</b> 0x0</p> <p>Timestamp Trigger Interrupt</p> <p>This bit indicates an interrupt event in the Timestamp Generator block of DWC_gmac. The software must read the corresponding registers in the DWC_gmac to get the exact cause of interrupt and clear its source to reset this bit to 1'b0. When this bit is high, the interrupt signal from the DWC_gmac subsystem (sbd_intr_o) is high.</p> <p>This bit is applicable only when the IEEE 1588 Timestamp feature is enabled. Otherwise, this bit is reserved.</p>
28	GPI	R	<p><b>Value After Reset:</b> 0x0</p> <p>GMAC PMT Interrupt</p> <p>This bit indicates an interrupt event in the PMT module of the DWC_gmac. The software must read the PMT Control and Status Register in the MAC to get the exact cause of interrupt and clear its source to reset this bit to 1'b0. The interrupt signal from the DWC_gmac subsystem (sbd_intr_o) is high when this bit is high.</p> <p>This bit is applicable only when the Power Management feature is enabled. Otherwise, this bit is reserved.</p> <p>Note: The GPI and pmt_intr_o interrupts are generated in different clock domains.</p>
27	GMI	R	<p><b>Value After Reset:</b> 0x0</p> <p>GMAC MMC Interrupt</p> <p>This bit reflects an interrupt event in the MMC module of the DWC_gmac. The software must read the corresponding registers in the DWC_gmac to get the exact cause of interrupt and clear the source of interrupt to make this bit as 1'b0. The interrupt signal from the DWC_gmac subsystem (sbd_intr_o) is high when this bit is high.</p> <p>This bit is applicable only when the MAC Management Counters (MMC) are enabled. Otherwise, this bit is reserved.</p>
26	GLI	R	<p><b>Value After Reset:</b> 0x0</p> <p>GMAC Line interface Interrupt When set, this bit reflects any of the following interrupt events in the DWC_gmac interfaces (if present and enabled in your configuration):</p> <ul style="list-style-type: none"> <li>PCS (TBI, RTBI, or SGMII): Link change or auto-negotiation complete event</li> <li>SMII or RGMII: Link change event</li> <li>General Purpose Input Status (GPIS): Any LL or LH event on the gpi_i input ports</li> </ul> <p>To identify the exact cause of the interrupt, the software must first read Bit 11 and Bits[2:0] of Register 14 (Interrupt Status Register) and then to clear the source of interrupt (which also clears the GLI interrupt), read any of the following corresponding registers:</p> <ul style="list-style-type: none"> <li>PCS (TBI, RTBI, or SGMII): Register 49 (AN Status Register)</li> <li>SMII or RGMII: Register 54 (SGMII/RGMII/SMII Status Register)</li> <li>General Purpose Input (GPI): Register 56 (General Purpose IO Register)</li> </ul> <p>The interrupt signal from the DWC_gmac subsystem (sbd_intr_o) is high when this bit is high.</p>
25:23	EB	R	<p><b>Value After Reset:</b> 0x0</p> <p>Error Bits This field indicates the type of error that caused a Bus Error, for example, error response on the AHB or AXI interface. This field is valid only when Bit 13 (FBI) is set. This field does not generate an interrupt.</p> <p>0 0 0: Error during Rx DMA Write Data Transfer</p> <p>0 1 1: Error during Tx DMA Read Data Transfer</p>

Bits	Name	Memory Access	Description
			<p>1 0 0: Error during Rx DMA Descriptor Write Access      1 0 1: Error during Tx DMA Descriptor Write Access      1 1 0: Error during Rx DMA Descriptor Read Access      1 1 1: Error during Tx DMA Descriptor Read Access      Note: 001 and 010 are reserved.</p>
22:20	TS	R	<p><b>Value After Reset:</b> 0x0  <b>Transmit Process State</b>      This field indicates the Transmit DMA FSM state. This field does not generate an interrupt.      3'b000: Stopped; Reset or Stop Transmit Command issued      3'b001: Running; Fetching Transmit Transfer Descriptor      3'b010: Running; Waiting for status      3'b011: Running; Reading Data from host memory buffer and queuing it to transmit buffer (Tx FIFO)      3'b100: TIME_STAMP write state      3'b101: Reserved for future use      3'b110: Suspended; Transmit Descriptor Unavailable or Transmit Buffer Underflow      3'b111: Running; Closing Transmit Descriptor</p>
19:17	RS	R	<p><b>Value After Reset:</b> 0x0  <b>Received Process State</b>      This field indicates the Receive DMA FSM state. This field does not generate an interrupt.      3'b000: Stopped: Reset or Stop Receive Command issued      3'b001: Running: Fetching Receive Transfer Descriptor      3'b010: Reserved for future use      3'b011: Running: Waiting for receive packet      3'b100: Suspended: Receive Descriptor Unavailable      3'b101: Running: Closing Receive Descriptor      3'b110: TIME_STAMP write state      3'b111: Running: Transferring the receive packet data from receive buffer to host memory</p>
16	NIS	R/W	<p><b>Value After Reset:</b> 0x0  <b>Normal Interrupt Summary</b>      Normal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in Register 7 (Interrupt Enable Register):      Register 5[0]: Transmit Interrupt      Register 5[2]: Transmit Buffer Unavailable      Register 5[6]: Receive Interrupt      Register 5[14]: Early Receive Interrupt      Only unmasked bits (interrupts for which interrupt enable is set in Register 7) affect the Normal Interrupt Summary bit.      This is a sticky bit and must be cleared (by writing 1 to this bit) each time a corresponding bit, which causes NIS to be set, is cleared.</p>
15	AIS	R/W	<p><b>Value After Reset:</b> 0x0  <b>Abnormal Interrupt Summary</b>      Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in Register 7 (Interrupt Enable Register):      Register 5[1]: Transmit Process Stopped      Register 5[3]: Transmit Jabber Timeout</p>

Bits	Name	Memory Access	Description
			<p>Register 5[4]: Receive FIFO Overflow      Register 5[5]: Transmit Underflow      Register 5[7]: Receive Buffer Unavailable      Register 5[8]: Receive Process Stopped      Register 5[9]: Receive Watchdog Timeout      Register 5[10]: Early Transmit Interrupt      Register 5[13]: Fatal Bus Error</p> <p>Only unmasked bits affect the Abnormal Interrupt Summary bit.      This is a sticky bit and must be cleared each time a corresponding bit, which causes AIS to be set, is cleared.</p>
14	ERI	R/W	<p><b>Value After Reset:</b> 0x0  <b>Early Receive Interrupt</b>      This bit indicates that the DMA filled the first data buffer of the packet. This bit is cleared when the software writes 1 to this bit or Bit 6 (RI) of this register is set (whichever occurs earlier).</p>
13	FBI	R/W	<p><b>Value After Reset:</b> 0x0  <b>Fatal Bus Error Interrupt</b>      This bit indicates that a bus error occurred, as described in Bits[25:23]. When this bit is set, the corresponding DMA engine disables all of its bus accesses.</p>
12:11	Reserved_12_11	R	<p><b>Value After Reset:</b> 0x0      Reserved</p>
10	ETI	R/W	<p><b>Value After Reset:</b> 0x0  <b>Early Transmit Interrupt</b>      This bit indicates that the frame to be transmitted is fully transferred to the MTL Transmit FIFO.</p>
9	RWT	R/W	<p><b>Value After Reset:</b> 0x0  <b>Receive Watchdog Timeout</b>      When set, this bit indicates that the Receive Watchdog Timer expired while receiving the current frame and the current frame is truncated after the watchdog timeout.</p>
8	RPS	R/W	<p><b>Value After Reset:</b> 0x0  <b>Receive Process Stopped</b>      This bit is asserted when the Receive Process enters the Stopped state.</p>
7	RU	R/W	<p><b>Value After Reset:</b> 0x0  <b>Receive Buffer Unavailable</b>      This bit indicates that the host owns the Next Descriptor in the Receive List and the DMA cannot acquire it. The Receive Process is suspended. To resume processing Receive descriptors, the host should change the ownership of the descriptor and issue a Receive Poll Demand command. If no Receive Poll Demand is issued, the Receive Process resumes when the next recognized incoming frame is received. This bit is set only when the previous Receive Descriptor is owned by the DMA.</p>
6	RI	R/W	<p><b>Value After Reset:</b> 0x0  <b>Receive Interrupt</b>      This bit indicates that the frame reception is complete. When reception is complete, the Bit 31 of RDES1 (Disable Interrupt on Completion) is reset in the last Descriptor, and the specific frame status information is updated in the descriptor. The reception remains in the Running state.</p>

Bits	Name	Memory Access	Description
5	UNF	R/W	<b>Value After Reset:</b> 0x0 Transmit Underflow This bit indicates that the Transmit Buffer had an Underflow during frame transmission. Transmission is suspended and an Underflow Error TDES0[1] is set.
4	OVF	R/W	<b>Value After Reset:</b> 0x0 Receive Overflow This bit indicates that the Receive Buffer had an Overflow during frame reception. If the partial frame is transferred to the application, the overflow status is set in RDES0[11].
3	TJT	R/W	<b>Value After Reset:</b> 0x0 Transmit Jabber Timeout This bit indicates that the Transmit Jabber Timer expired, which happens when the frame size exceeds 2,048 (10,240 bytes when the Jumbo frame is enabled). When the Jabber Timeout occurs, the transmission process is aborted and placed in the Stopped state. This causes the Transmit Jabber Timeout TDES0[14] flag to assert.
2	TU	R/W	<b>Value After Reset:</b> 0x0 Transmit Buffer Unavailable This bit indicates that the host owns the Next Descriptor in the Transmit List and the DMA cannot acquire it. Transmission is suspended. Bits[22:20] explain the Transmit Process state transitions. To resume processing Transmit descriptors, the host should change the ownership of the descriptor by setting TDES0[31] and then issue a Transmit Poll Demand command.
1	TPS	R/W	<b>Value After Reset:</b> 0x0 Transmit Process Stopped This bit is set when the transmission is stopped.
0	TI	R/W	<b>Value After Reset:</b> 0x0 Transmit Interrupt This bit indicates that the frame transmission is complete. When transmission is complete, Bit 31 (OWN) of TDES0 is reset, and the specific frame status information is updated in the descriptor.

#### 9.7.2.2.7 Register 6 (Operation Mode Register)

##### Operation\_Mode

**Size:** 32 bits

**Offset:** 0x1018

**Memory Access:** R/W

**Value After Reset:** 0x0

31:27	2 6	2 5	24	23	22	2 1	2 0	19:17	16: 14	1 3	12: 11	10: 9	8	7	6	5	4: 3	2	1	0
Reserved_31_27	D T	R SF	D FF	RFA _2	RFD _2	TS F	FT F	Reserved_19_17	TTC	S T	RF D	RF A	EF C	FE F	F U F	D GF	RT C	O SF	S R	Reserve d_0

The Operation Mode register establishes the Transmit and Receive operating modes and commands. This register should be the last CSR to be written as part of the DMA initialization. This register is also present in the GMAC-MTL configuration with unused and reserved bits 24, 13, 2, and 1.

Bits	Name	Memory Access	Description
31:27	Reserved_31_27	R	<b>Value After Reset:</b> 0x0 Reserved
26	DT	R/W	<b>Value After Reset:</b> 0x0 Disable Dropping of TCP/IP Checksum Error Frames When this bit is set, the MAC does not drop the frames which only have errors detected by the Receive Checksum Offload engine. Such frames do not have any errors (including FCS error) in the Ethernet frame received by the MAC but have errors only in the encapsulated payload. When this bit is reset, all error frames are dropped if the FEF bit is reset. If the IPC Full Checksum Offload Engine (Type 2) is disabled, this bit is reserved (RO with value 1'b0).
25	RSF	R/W	<b>Value After Reset:</b> 0x0 Receive Store and Forward When this bit is set, the MTL reads a frame from the Rx FIFO only after the complete frame has been written to it, ignoring the RTC bits. When this bit is reset, the Rx FIFO operates in the cut-through mode, subject to the threshold specified by the RTC bits.
24	DFF	R/W	<b>Value After Reset:</b> 0x0 Disable Flushing of Received Frames When this bit is set, the Rx DMA does not flush any frames because of the unavailability of receive descriptors or buffers as it does normally when this bit is reset. This bit is reserved (and RO) in the GMAC-MTL configuration.
23	RFA_2	R	<b>Value After Reset:</b> 0x0 MSB of Threshold for Activating Flow Control If the DWC_gmac is configured for an Rx FIFO depth of 8 KB or more, this bit (when set) provides additional threshold levels for activating the flow control in both half-duplex and full-duplex modes. This bit (as Most Significant Bit) along with the RFA (Bits[10:9]) gives the following thresholds for activating flow control: * 100: Full minus 5 KB, that is, FULL - 5KB * 101: Full minus 6 KB, that is, FULL - 6KB * 110: Full minus 7 KB, that is, FULL - 7KB * 111: Reserved This bit is reserved (and RO) if the Rx FIFO is 4 KB or less deep.
22	RFD_2	R	<b>Value After Reset:</b> 0x0 MSB of Threshold for Deactivating Flow Control If the DWC_gmac is configured for Rx FIFO size of 8 KB or more, this bit (when set) provides additional threshold levels for deactivating the flow control in both half-duplex and full-duplex modes. This bit (as Most Significant Bit) along with the RFD (Bits[12:11]) gives the following thresholds for deactivating flow control: * 100: Full minus 5 KB, that is, FULL - 5KB * 101: Full minus 6 KB, that is, FULL - 6KB * 110: Full minus 7 KB, that is, FULL - 7KB * 111: Reserved This bit is reserved (and RO) if the Rx FIFO is 4 KB or less deep.
21	TSF	R/W	<b>Value After Reset:</b> 0x0 Transmit Store and Forward When this bit is set, transmission starts when a full frame resides in the MTL Transmit FIFO. When this bit is set, the TTC values specified in Bits[16:14] are ignored. This bit should be changed only when the transmission is stopped.
20	FTF	R/W	<b>Value After Reset:</b> 0x0

Bits	Name	Memory Access	Description
			<p><b>Flush Transmit FIFO</b>  When this bit is set, the transmit FIFO controller logic is reset to its default values and thus all data in the Tx FIFO is lost or flushed. This bit is cleared internally when the flushing operation is completed. The Operation Mode register should not be written to until this bit is cleared. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt frame transmission.</p> <p>Note: The flush operation is complete only when the Tx FIFO is emptied of its contents and all the pending Transmit Status of the transmitted frames are accepted by the host. To complete this flush operation, the PHY transmit clock (clk_tx_i) is required to be active.</p>
19:17	Reserved_19_17	R	<p><b>Value After Reset:</b> 0x0  Reserved</p>
16:14	TTC	R/W	<p><b>Value After Reset:</b> 0x0  <b>Transmit Threshold Control</b>  These bits control the threshold level of the MTL Transmit FIFO. Transmission starts when the frame size within the MTL Transmit FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are used only when Bit 21 (TSF) is reset.</p> <p>000: 64  001: 128  010: 192  011: 256  100: 40  101: 32  110: 24  111: 16</p>
13	ST	R/W	<p><b>Value After Reset:</b> 0x0  <b>Start or Stop Transmission Command</b>  When this bit is set, transmission is placed in the Running state, and the DMA checks the Transmit List at the current position for a frame to be transmitted. Descriptor acquisition is attempted either from the current position in the list, which is the Transmit List Base Address set by Register 4 (Transmit Descriptor List Address Register), or from the position retained when transmission was stopped previously. If the DMA does not own the current descriptor, transmission enters the Suspended state and Bit 2 (Transmit Buffer Unavailable) of Register 5 (Status Register) is set. The Start Transmission command is effective only when transmission is stopped. If the command is issued before setting Register 4 (Transmit Descriptor List Address Register), then the DMA behavior is unpredictable.</p> <p>When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and it becomes the current position when transmission is restarted. To change the list address, you need to program Register 4 (Transmit Descriptor List Address Register) with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current frame is complete or the transmission is in the Suspended state.</p>
12:11	RFD	R/W	<b>Value After Reset:</b> 0x0

Bits	Name	Memory Access	Description
			<p>Threshold for Deactivating Flow Control (in half-duplex and full-duplex)  These bits control the threshold (Fill-level of Rx FIFO) at which the flow control is de-asserted after activation.</p> <p>00: Full minus 1 KB, that is, FULL - 1KB  01: Full minus 2 KB, that is, FULL - 2KB  10: Full minus 3 KB, that is, FULL - 3KB  11: Full minus 4 KB, that is, FULL - 4KB</p> <p>The de-assertion is effective only after flow control is asserted. If the Rx FIFO is 8 KB or more, an additional bit (RFD_2) is used for more threshold levels as described in Bit 22. These bits are reserved and read-only when the Rx FIFO depth is less than 4 KB. Note: For proper flow control, the value programmed in the "RFD_2, RFD" fields should be equal to or more than the value programmed in the "RFA_2, RFA" fields.</p>
10:9	RFA	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Threshold for Activating Flow Control (in half-duplex and full-duplex)  These bits control the threshold (Fill level of Rx FIFO) at which the flow control is activated.</p> <p>00: Full minus 1 KB, that is, FULL - 1KB  01: Full minus 2 KB, that is, FULL - 2KB  10: Full minus 3 KB, that is, FULL - 3KB  11: Full minus 4 KB, that is, FULL - 4KB</p> <p>These values are applicable only to Rx FIFOs of 4 KB or more and when Bit 8 (EFC) is set high. If the Rx FIFO is 8 KB or more, an additional Bit (RFA_2) is used for more threshold levels as described in Bit 23. These bits are reserved and read-only when the depth of Rx FIFO is less than 4 KB.</p> <p>Note: When FIFO size is exactly 4 KB, although the DWC_gmac allows you to program the value of these bits to 11, the software should not program these bits to 2'b11. The value 2'b11 means flow control on FIFO empty condition.</p>
8	EFC	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Enable HW Flow Control</p> <p>When this bit is set, the flow control signal operation based on the fill-level of Rx FIFO is enabled. When reset, the flow control operation is disabled. This bit is not used (reserved and always reset) when the Rx FIFO is less than 4 KB.</p>
7	FEF	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Forward Error Frames</p> <p>When this bit is reset, the Rx FIFO drops frames with error status (CRC error, collision error, GMII_ER, giant frame, watchdog timeout, or overflow). However, if the start byte (write) pointer of a frame is already transferred to the read controller side (in Threshold mode), then the frame is not dropped. In the GMAC-MTL configuration in which the Frame Length FIFO is also enabled during core configuration, the Rx FIFO drops the error frames if that frame's start byte is not transferred (output) on the ARI bus. When the FEF bit is set, all frames except runt error frames are forwarded to the DMA. If the Bit 25 (RSF) is set and the Rx FIFO overflows when a partial frame is written, then the frame is dropped irrespective of the FEF bit setting. However, if the Bit 25 (RSF) is reset and the Rx FIFO overflows when a partial frame is written, then a partial frame may be forwarded to the DMA.</p> <p>Note: When FEF bit is reset, the giant frames are dropped if the giant frame status is given in Rx Status in the following configurations:  The IP checksum engine (Type 1) and full checksum offload engine (Type 2) are not selected.  The advanced timestamp feature is not selected but the extended status is</p>

Bits	Name	Memory Access	Description
			<p>selected. The extended status is available with the following features:</p> <ul style="list-style-type: none"> <li>- L3-L4 filter in GMAC-CORE or GMAC-MTL configurations</li> <li>- Full checksum offload engine (Type 2) with enhanced descriptor format in the GMAC-DMA, GMAC-AHB, or GMAC-AXI configurations.</li> </ul>
6	FUF	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Forward Undersized Good Frames</p> <p>When set, the Rx FIFO forwards Undersized frames (frames with no Error and length less than 64 bytes) including pad-bytes and CRC.</p> <p>When reset, the Rx FIFO drops all frames of less than 64 bytes, unless a frame is already transferred because of the lower value of Receive Threshold, for example, RTC = 01.</p>
5	DGF	R	<p><b>Value After Reset:</b> 0x0</p> <p>Drop Giant Frames When set, the MAC drops the received giant frames in the Rx FIFO, that is, frames that are larger than the computed giant frame limit. When reset, the MAC does not drop the giant frames in the Rx FIFO.</p> <p>Note: This bit is available in the following configurations in which the giant frame status is not provided in Rx status and giant frames are not dropped by default:</p> <ul style="list-style-type: none"> <li>Configurations in which IP Checksum Offload (Type 1) is selected in Rx</li> <li>Configurations in which the IPC Full Checksum Offload Engine (Type 2) is selected in Rx with normal descriptor format</li> <li>Configurations in which the Advanced Timestamp feature is selected</li> <li>In all other configurations, this bit is not used (reserved and always reset).</li> </ul>
4:3	RTC	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Receive Threshold Control</p> <p>These two bits control the threshold level of the MTL Receive FIFO. Transfer (request) to DMA starts when the frame size within the MTL Receive FIFO is larger than the threshold. In addition, full frames with length less than the threshold are transferred automatically.</p> <p>The value of 11 is not applicable if the configured Receive FIFO size is 128 bytes.</p> <p>These bits are valid only when the RSF bit is zero, and are ignored when the RSF bit is set to 1.</p> <ul style="list-style-type: none"> <li>00: 64</li> <li>01: 32</li> <li>10: 96</li> <li>11: 128</li> </ul>
2	OSF	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Operate on Second Frame</p> <p>When this bit is set, it instructs the DMA to process the second frame of the Transmit data even before the status for the first frame is obtained.</p>
1	SR	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Start or Stop Receive</p> <p>When this bit is set, the Receive process is placed in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes the incoming frames. The descriptor acquisition is attempted from the current position in the list, which is the address set by Register 3 (Receive Descriptor List Address Register) or the position retained when the Receive process was previously stopped. If the DMA does not own the descriptor, reception is suspended and Bit 7 (Receive Buffer Unavailable) of Register 5 (Status Register) is set. The Start Receive command is effective only when the reception has stopped. If the command is</p>

Bits	Name	Memory Access	Description
			issued before setting Register 3 (Receive Descriptor List Address Register), the DMA behavior is unpredictable. When this bit is cleared, the Rx DMA operation is stopped after the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process is restarted. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state.
0	Reserved_0	R	<b>Value After Reset:</b> 0x0 Reserved

#### 9.7.2.2.8 Register 7 (Interrupt Enable Register)

##### Interrupt\_Enable

**Size:** 32 bits

**Offset:** 0x101c

**Memory Access:** R/W

**Value After Reset:** 0x0

31:17	16	15	14	13	12:11	10	9	8	7	6	5	4	3	2	1	0
Reserved_31_17	NIE	AIE	ERE	FB E	Reserved_12_1 1	ET E	RW E	RS E	RU E	RI E	UN E	OV E	TJ E	TU E	TS E	TI E

The Interrupt Enable register enables the interrupts reported by Register 5 (Status Register). Setting a bit to 1'b1 enables a corresponding interrupt. After a hardware or software reset, all interrupts are disabled.

Bits	Name	Memory Access	Description
31:17	Reserved_31_17	R	<b>Value After Reset:</b> 0x0 Reserved
16	NIE	R/W	<b>Value After Reset:</b> 0x0 Normal Interrupt Summary Enable When this bit is set, normal interrupt summary is enabled. When this bit is reset, normal interrupt summary is disabled. This bit enables the following interrupts in Register 5 (Status Register): Register 5[0]: Transmit Interrupt Register 5[2]: Transmit Buffer Unavailable Register 5[6]: Receive Interrupt Register 5[14]: Early Receive Interrupt
15	AIE	R/W	<b>Value After Reset:</b> 0x0 Abnormal Interrupt Summary Enable When this bit is set, abnormal interrupt summary is enabled. When this bit is reset, the abnormal interrupt summary is disabled. This bit enables the following interrupts in Register 5 (Status Register): Register 5[1]: Transmit Process Stopped Register 5[3]: Transmit Jabber Timeout Register 5[4]: Receive Overflow Register 5[5]: Transmit Underflow Register 5[7]: Receive Buffer Unavailable Register 5[8]: Receive Process Stopped Register 5[9]: Receive Watchdog Timeout Register 5[10]: Early Transmit Interrupt

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
			Register 5[13]: Fatal Bus Error
14	ERE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Early Receive Interrupt Enable</p> <p>When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Early Receive Interrupt is enabled. When this bit is reset, the Early Receive Interrupt is disabled.</p>
13	FBE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Fatal Bus Error Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Fatal Bus Error Interrupt is enabled. When this bit is reset, the Fatal Bus Error Enable Interrupt is disabled.</p>
12:11	Reserved_12_11	R	<p><b>Value After Reset:</b> 0x0</p> <p>Reserved</p>
10	ETE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Early Transmit Interrupt Enable</p> <p>When this bit is set with an Abnormal Interrupt Summary Enable (Bit 15), the Early Transmit Interrupt is enabled. When this bit is reset, the Early Transmit Interrupt is disabled.</p>
9	RWE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Receive Watchdog Timeout Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Watchdog Timeout Interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout Interrupt is disabled.</p>
8	RSE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Receive Stopped Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped Interrupt is disabled.</p>
7	RUE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Receive Buffer Unavailable Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Buffer Unavailable Interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable Interrupt is disabled.</p>
6	RIE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Receive Interrupt Enable</p> <p>When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled.</p>
5	UNE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Underflow Interrupt Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmit Underflow Interrupt is enabled. When this bit is reset, the Underflow Interrupt is disabled.</p>
4	OVE	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Overflow Interrupt Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive</p>

Bits	Name	Memory Access	Description
			Overflow Interrupt is enabled. When this bit is reset, the Overflow Interrupt is disabled.
3	TJE	R/W	<b>Value After Reset:</b> 0x0 Transmit Jabber Timeout Enable When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmit Jabber Timeout Interrupt is enabled. When this bit is reset, the Transmit Jabber Timeout Interrupt is disabled.
2	TUE	R/W	<b>Value After Reset:</b> 0x0 Transmit Buffer Unavailable Enable When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Transmit Buffer Unavailable Interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable Interrupt is disabled.
1	TSE	R/W	<b>Value After Reset:</b> 0x0 Transmit Stopped Enable When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmission Stopped Interrupt is enabled. When this bit is reset, the Transmission Stopped Interrupt is disabled.
0	TIE	R/W	<b>Value After Reset:</b> 0x0 Transmit Interrupt Enable When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled.

#### 9.7.2.2.9 Register 8 (Missed Frame and Buffer Overflow Counter Register)

##### Missed\_Frame\_And\_Buffer\_Overflow\_Counter

**Size:** 32 bits

**Offset:** 0x1020

**Memory Access:** R

**Value After Reset:** 0x0

31:29	28	27:17	16	15:0
Reserved_31_29	OVFCNTOVF	OVFFRMCNT	MISCNTOVF	MISFRMCNT

The DMA maintains two counters to track the number of frames missed during reception. This register reports the current value of the counter. The counter is used for diagnostic purposes. Bits[15:0] indicate missed frames because of the host buffer being unavailable. Bits[27:17] indicate missed frames because of buffer overflow conditions (MTL and MAC) and runt frames (good frames of less than 64 bytes) dropped by the MTL.

Bits	Name	Memory Access	Description
31:29	Reserved_31_29	R	<b>Value After Reset:</b> 0x0 Reserved
28	OVFCNTOVF	R	<b>Value After Reset:</b> 0x0 Overflow Bit for FIFO Overflow Counter This bit is set every time the Overflow Frame Counter (Bits[27:17]) overflows, that is, the Rx FIFO overflows with the overflow frame counter at maximum value. In such a scenario, the overflow frame counter is reset to all-zeros and this bit indicates that the rollover happened.
27:17	OVFFRMCNT	R	<b>Value After Reset:</b> 0x0

Bits	Name	Memory Access	Description
			<p><b>Overflow Frame Counter</b>  This field indicates the number of frames missed by the application. This counter is incremented each time the MTL FIFO overflows. The counter is cleared when this register is read with mci_be_i[2] at 1'b1.</p>
16	MISCNTOVF	R	<p><b>Value After Reset:</b> 0x0  <b>Overflow Bit for Missed Frame Counter</b>  This bit is set every time Missed Frame Counter (Bits[15:0]) overflows, that is, the DMA discards an incoming frame because of the Host Receive Buffer being unavailable with the missed frame counter at maximum value. In such a scenario, the Missed frame counter is reset to all-zeros and this bit indicates that the rollover happened.</p>
15:0	MISFRMCNT	R	<p><b>Value After Reset:</b> 0x0  <b>Missed Frame Counter</b>  This field indicates the number of frames missed by the controller because of the Host Receive Buffer being unavailable. This counter is incremented each time the DMA discards an incoming frame. The counter is cleared when this register is read with mci_be_i[0] at 1'b1.</p>

#### 9.7.2.2.10 Register 9 (Receive Interrupt Watchdog Timer Register)

**Receive\_Interrupt\_Watchdog\_Timer**

**Size:** 32 bits

**Offset:** 0x1024

**Memory Access:** R/W

**Value After Reset:** 0x0

31:8	7:0
Reserved_31_8	RIWT

This register, when written with non-zero value, enables the watchdog timer for the Receive Interrupt (Bit 6) of Register 5 (Status Register)

Bits	Name	Memory Access	Description
31:8	Reserved_31_8	R	<p><b>Value After Reset:</b> 0x0  Reserved</p>
7:0	RIWT	R/W	<p><b>Value After Reset:</b> 0x0  <b>RI Watchdog Timer Count</b>  This bit indicates the number of system clock cycles multiplied by 256 for which the watchdog timer is set. The watchdog timer gets triggered with the programmed value after the Rx DMA completes the transfer of a frame for which the RI status bit is not set because of the setting in the corresponding descriptor RDES1[31]. When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per RDES1[31] of any received frame.</p>

#### 9.7.2.2.11 Register 11 (AHB or AXI Status Register)

**AHB\_or\_AXI\_Status**

**Size:** 32 bits

**Offset:** 0x102c

**Memory Access:** R

**Value After Reset:** 0x0

31:2	1	0
Reserved_31_2	AXIRDSTS	AXWHSTS

This register provides the active status of the AHB master interface or AXI interface's read and write channels. This register is present and valid only in the GMAC-AHB and GMAC-AXI configurations. This register is useful for debugging purposes. In addition, this register is valid only in the Channel 0 DMA when multiple channels are present in the AV mode.

Bits	Name	Memory Access	Description
31:2	Reserved_31_2	R	<b>Value After Reset:</b> 0x0 Reserved
1	AXIRDSTS	R	<b>Value After Reset:</b> 0x0 AXI Master Read Channel Status When high, it indicates that AXI Master's read channel is active and transferring data.
0	AXWHSTS	R	<b>Value After Reset:</b> 0x0 AXI Master Write Channel or AHB Master Status When high, it indicates that AXI Master's write channel is active and transferring data in the GMAC-AXI configuration. In the GMAC-AHB configuration, it indicates that the AHB master interface FSMs are in the non-idle state.

#### 9.7.2.2.12 Register 18 (Current Host Transmit Descriptor Register)

**Current\_Host\_Transmit\_Descriptor****Size:** 32 bits**Offset:** 0x1048**Memory Access:** R**Value After Reset:** 0x0

31:0
CURTDESAPTR

The Current Host Transmit Descriptor register points to the start address of the current Transmit Descriptor read by the DMA.

Bits	Name	Memory Access	Description
31:0	CURTDESAPTR	R	<b>Value After Reset:</b> 0x0 Host Transmit Descriptor Address Pointer Cleared on Reset. Pointer updated by the DMA during operation.

#### 9.7.2.2.13 Register 19 (Current Host Receive Descriptor Register)

**Current\_Host\_Receive\_Descriptor****Size:** 32 bits**Offset:** 0x104c**Memory Access:** R**Value After Reset:** 0x0

31:0
CURRDESAPTR

The Current Host Receive Descriptor register points to the start address of the current Receive Descriptor read by the DMA.

Bits	Name	Memory Access	Description
31:0	CURRDESAPTR	R	

Bits	Name	Memory Access	Description
31:0	CURRDESAPTR	R	<b>Value After Reset:</b> 0x0 Host Receive Descriptor Address Pointer Cleared on Reset. Pointer updated by the DMA during operation.

#### 9.7.2.2.14 Register 20 (Current Host Transmit Buffer Address Register)

**Current\_Host\_Transmit\_Buffer\_Address**

**Size:** 32 bits

**Offset:** 0x1050

**Memory Access:** R

**Value After Reset:** 0x0

31:0	CURTBUFAPTR
------	-------------

The Current Host Transmit Buffer Address register points to the current Transmit Buffer Address being read by the DMA.

Bits	Name	Memory Access	Description
31:0	CURTBUFAPTR	R	<b>Value After Reset:</b> 0x0 Host Transmit Buffer Address Pointer Cleared on Reset. Pointer updated by the DMA during operation.

#### 9.7.2.2.15 Register 21 (Current Host Receive Buffer Address Register)

**Current\_Host\_Receive\_Buffer\_Address**

**Size:** 32 bits

**Offset:** 0x1054

**Memory Access:** R

**Value After Reset:** 0x0

31:0	CURRBUFAPTR
------	-------------

The Current Host Receive Buffer Address register points to the current Receive Buffer address being read by the DMA.

Bits	Name	Memory Access	Description
31:0	CURRBUFAPTR	R	<b>Value After Reset:</b> 0x0 Host Receive Buffer Address Pointer Cleared on Reset. Pointer updated by the DMA during operation.

#### 9.7.2.2.16 Register 22 (HW Feature Register)

**HW\_Feature**

**Size:** 32 bits

**Offset:** 0x1058

**Memory Access:** R

**Value After Reset:** 0x90c0fb9

31	3 0: 2 8	27	26	2 5	24	2 3: 2 2	2 1: 2 0	19	18	17	1 6	1 5	1 4	13	12	1 1	1 0	9	8	7	6	5	4	3	2	1	0
Re ser ve	A C T	SA VL A	FL EX IP	I N T	E N H	T X C	R X C	RX FI C	RX TY P2	RX TY P1	T X C	A V S	E E E	TS V ER	TS V ER	M M C	M G K	R W K	S M A	L3 L4 FL	P C S	AD DM AC	H A S	EX TH AS	H D S	G M II	M I S

d_31	P H YI F	N I S N	P S E N	T S E N	D E S S	H C N T	H C N T	O SI ZE	C O E	C O E	O ES EL	E L	S E L	2S EL	1S EL	S E L	S E L	S E L	TR EN	S E L	AD RSE L	H S E L	HE N E L	E L S E L	E L
------	-------------------	------------------	------------------	------------------	------------------	------------------	------------------	---------------	-------------	-------------	---------------	--------	-------------	----------	----------	-------------	-------------	-------------	----------	-------------	----------------	------------------	-------------------	-----------------------	--------

This register indicates the presence of the optional features or functions of the DWC\_gmac. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Note: All bits are set or reset as per the selection of features during the DWC\_gmac configuration.

Bits	Name	Memory Access	Description
31	Reserved_31	R	<b>Value After Reset:</b> 0x0 Reserved
30:28	ACTPHYIF	R	<b>Value After Reset:</b> 0x0 Active or Selected PHY interface When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion 0000: GMII or MII 0001: RGMII 0010: SGMII 0011: TBI 0100: RMII 0101: RTBI 0110: SMII 0111: RevMII All Others: Reserved
27	SAVLANINS	R	<b>Value After Reset:</b> 0x1 Source Address or VLAN Insertion
26	FLEXIPPSEN	R	<b>Value After Reset:</b> 0x0 Flexible Pulse-Per-Second Output
25	INTTSEN	R	<b>Value After Reset:</b> 0x0 Timestamping with Internal System Time
24	ENHDESEL	R	<b>Value After Reset:</b> 0x1 Alternate (Enhanced Descriptor)
23:22	TXCHCNT	R	<b>Value After Reset:</b> 0x0 Number of additional Tx channels
21:20	RXCHCNT	R	<b>Value After Reset:</b> 0x0 Number of additional Rx channels
19	RXFIFOSIZE	R	<b>Value After Reset:</b> 0x1 Rx FIFO > 2,048 Bytes
18	RXTYP2COE	R	<b>Value After Reset:</b> 0x1 IP Checksum Offload (Type 2) in Rx
17	RXTYP1COE	R	<b>Value After Reset:</b> 0x0 IP Checksum Offload (Type 1) in Rx Note: If IPCKSUM_EN = Enabled and IPC_FULL_OFFLOAD = Enabled, then RXTYP1COE = 0 and RXTYP2COE =1.
16	TXCOESEL	R	<b>Value After Reset:</b> 0x0 Checksum Offload in Tx

Bits	Name	Memory Access	Description
15	AVSEL	R	<b>Value After Reset:</b> 0x0 AV Feature
14	EEESEL	R	<b>Value After Reset:</b> 0x0 Energy Efficient Ethernet
13	TSVER2SEL	R	<b>Value After Reset:</b> 0x0 IEEE 1588-2008 Advanced Timestamp
12	TSVER1SEL	R	<b>Value After Reset:</b> 0x0 Only IEEE 1588-2002 Timestamp
11	MMCSEL	R	<b>Value After Reset:</b> 0x1 RMON Module
10	MGKSEL	R	<b>Value After Reset:</b> 0x1 PMT Magic Packet
9	RWKSEL	R	<b>Value After Reset:</b> 0x1 PMT Remote Wakeup
8	SMASEL	R	<b>Value After Reset:</b> 0x1 SMA (MDIO) Interface
7	L3L4FLTREN	R	<b>Value After Reset:</b> 0x1 Layer 3 and Layer 4 Filter Feature
6	PCSSEL	R	<b>Value After Reset:</b> 0x0 PCS registers (TBI, SGMII, or RTBI PHY interface)
5	ADDMACADDRSEL	R	<b>Value After Reset:</b> 0x1 Multiple MAC Address Registers
4	HASHSEL	R	<b>Value After Reset:</b> 0x1 HASH Filter
3	EXTHASHEN	R	<b>Value After Reset:</b> 0x1 Expanded DA Hash Filter
2	HDSEL	R	<b>Value After Reset:</b> 0x0 Half-Duplex support
1	GMIISEL	R	<b>Value After Reset:</b> 0x0 1000 Mbps support
0	MIISEL	R	<b>Value After Reset:</b> 0x1 10 or 100 Mbps support

## 10 Timers

### 10.1 General-Purpose Timer

#### 10.1.1 Function Description

##### 10.1.1.1 Timer Operation

The Timers component implements up to four identical but separately-programmable timers.

Timers count down from a programmed value and generate an interrupt when the count reaches zero.

The initial value for each timer – that is, the value from which it counts down – is loaded into the timer using the appropriate load count register (TimerNLoadCount). Two events can cause a timer to load the initial count from its TimerNLoadCount register:

- Timer is enabled after being reset or disabled
- Timer counts down to 0

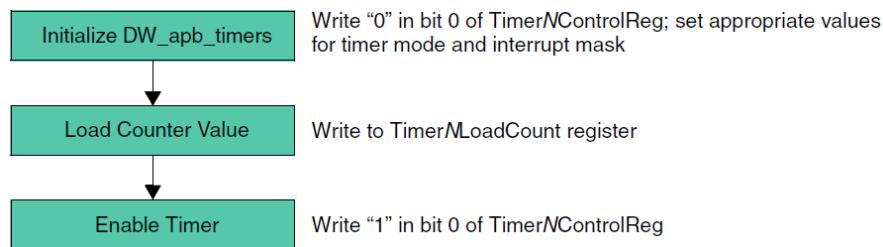
All interrupt status registers and end-of-interrupt registers can be accessed at any time.

##### 10.1.1.2 Timers Usage Flow

The procedure is a basic flow to follow when programming the Timers. More advanced functions are discussed later in this chapter.

1. Initialize the timer through the TimerNControlReg register (where  $N$  is in the range 1 to 4):
  - a. Disable the timer by writing a “0” to the timer enable bit (bit 0); accordingly, the timer\_en output signal is de-asserted.
  - b. Program the timer mode—user-defined or free-running—by writing a “0” or “1,” respectively, to the timer mode bit (bit 1).
  - c. Set the interrupt mask as either masked or not masked by writing a “1” or “0,” respectively, to the timer interrupt mask bit (bit 2).
2. Load the timer counter value into the TimerNLoadCount register (where  $N$  is in the range 1 to 4).
3. Enable the timer by writing a “1” to bit 0 of TimerNControlReg.

Timer Usage flow:



##### 10.1.1.3 Choosing the Number of Timers

There are several registers with names specific to the number of timers that you choose (where  $N$  is from 1 to 4):

- TimerNLoadCount – TimerN load count register
- TimerNCurrentValue – TimerN current value register
- TimerNControlReg – TimerN control register
- TimerNEOI – TimerN end-of-interrupt register
- TimerNIntStatus – TimerN interrupt status register

##### 10.1.1.3.1 Enabling and Disabling a Timer

You use bit 0 of the TimerNControlReg, where  $N$  is in the range 1 to 4, to either enable or disable a timer.

##### 10.1.1.3.2 Loading a Timer Countdown Value

When a timer counter is enabled after being reset or disabled, the count value is loaded from the TimerNLoadCount register; this occurs in both free-running and user-defined count modes. When a timer counts down to 0, it loads one of two values, depending on the timer operating mode:

- User-defined count mode – Timer loads the current value of the TimerNLoadCount register. Use this mode if you want a fixed, timed interrupt. Designate this mode by writing a “1” to bit 1 of TimerNControlReg.
- Free-running mode – Timer loads the maximum value, which is dependent on the timer width; that is, the TimerNLoadCount register is comprised of  $2^{32} - 1$  bits, all of which are loaded with 1s. The timer counter wrapping to its maximum value allows time to reprogram or disable the timer before another interrupt occurs. Use this mode if you want a single timed interrupt. Designate this mode by writing a “0” to bit 1 of TimerNControlReg.

### 10.1.1.3.3 Working with Interrupts

The TimerNIntStatus and TimerNEOI registers handle interrupts in order to ensure safe operation of the interrupt clearing. Because of the hclk/pclk ratio, if pclk can perform a write to clear an interrupt, it could continue with another transfer on the bus without knowing whether the write has occurred. Therefore, it is much safer to clear the interrupt by a read operation.

#### Clearing Interrupts

Provided the timer is enabled, its interrupt remains asserted until it is cleared by reading one of two end-of-interrupt registers (TimerNEOI or TimersEOI, the individual and global end-of-interrupt registers, respectively). When the timer is disabled, the timer interrupt is cleared. You can clear an individual timer interrupt by reading its TimerNEOI register. You can clear all active timer interrupts at once by reading the global TimersEOI register or by disabling the timer.

#### Checking Interrupt Status

You can query the interrupt status of an individual timer without clearing its interrupt by reading the TimerNIntStatus register. You can query the interrupt status of all timers without clearing the interrupts by reading the global TimersIntStatus register.

#### Masking Interrupts

Each individual timer interrupt can be masked using its TimerNControlReg register. To mask an interrupt, you write a “1” to bit 2 of TimerNControlReg.

#### Setting Interrupt Polarity

The polarity of the generated timer interrupts can be [configured](#) to be either active-high or active-low using the TIM\_INTRPT\_PLRITY parameter (Interrupt Polarity).

### 10.1.2 Timers Registers

#### 10.1.2.1 Register Memory Maps

Register	Offset	Size	Memory Access	Description
DW_apb_timers address block				
<a href="#">TIMER1LOADCOUNT</a>	0x0	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Timer1 Load Count Register Size: 8-32 bits Address Offset: 0x00 Read/Write Access: Read/Write
<a href="#">TIMER1CURRENTVAL</a>	0x4	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timer1 Current Value Size: 8-32 bits Address Offset: 4 Read/Write Access: Read
<a href="#">TIMER1CONTROLREG</a>	0x8	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Timer1 Control Register Size: 3 bits Address Offset: 8 Read/Write Access: Read/Write This register controls enabling, operating mode (free-running or defined-count), and interrupt mask of Timer1. You can program each Timer1ControlReg to enable or disable a specific timer and to control its mode of operation.

Register	Offset	Size	Memory Access	Description
<a href="#">TIMER1EOI</a>	0xc	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timer1 End-of-Interrupt Register Size: 1 bit Address Offset: 12 Read/Write Access: Read
<a href="#">TIMER1INTSTAT</a>	0x10	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timer1 Interrupt Status Register Size: 1 bit Address Offset: 16 Read/Write Access: Read
<a href="#">TIMER2LOADCOUNT</a>	0x14	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Timer2 Load Count Register Size: 8-32 bits Address Offset: 20 Read/Write Access: Read/Write
<a href="#">TIMER2CURRENTVAL</a>	0x18	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timer2 Current Value Size: 8-32 bits Address Offset: 24 Read/Write Access: Read
<a href="#">TIMER2CONTROLREG</a>	0x1c	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Timer2 Control Register Size: 3 bits Address Offset: 28 Read/Write Access: Read/Write This register controls enabling, operating mode (free-running or defined-count), and interrupt mask of Timer2. You can program each Timer1ControlReg to enable or disable a specific timer and to control its mode of operation.
<a href="#">TIMER2EOI</a>	0x20	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timer2 End-of-Interrupt Register Size: 1 bit Address Offset: 32 Read/Write Access: Read
<a href="#">TIMER2INTSTAT</a>	0x24	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timer2 Interrupt Status Register Size: 1 bit Address Offset: 36 Read/Write Access: Read
<a href="#">TIMER3LOADCOUNT</a>	0x28	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Timer1 Load Count Register Size: 8-32 bits Address Offset: 40 Read/Write Access: Read/Write
<a href="#">TIMER3CURRENTVAL</a>	0x2c	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timer3 Current Value Size: 8-32 bits Address Offset: 44 Read/Write Access: Read
<a href="#">TIMER3CONTROLREG</a>	0x30	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Timer3 Control Register Size: 3 bits Address Offset: 48 Read/Write Access: Read/Write This register controls enabling, operating mode (free-running or defined-count), and interrupt mask of Timer3. You can program each Timer1ControlReg to enable or disable a specific timer and to control its mode of operation.
<a href="#">TIMER3EOI</a>	0x34	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timer1 End-of-Interrupt Register Size: 1 bit Address Offset: 52 Read/Write Access: Read
<a href="#">TIMER3INTSTAT</a>	0x38	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timer3 Interrupt Status Register Size: 1 bit Address Offset: 56 Read/Write Access: Read
<a href="#">TIMER4LOADCOUNT</a>	0x3c	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Timer4 Load Count Register Size: 8-32 bits Address Offset: 60

Register	Offset	Size	Memory Access	Description
				Read/Write Access: Read/Write
<a href="#">TIMER4CURRENTVAL</a>	0x40	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timer4 Current Value Size: 8-32 bits Address Offset: 64 Read/Write Access: Read
<a href="#">TIMER4CONTROLREG</a>	0x44	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Timer4 Control Register Size: 3 bits Address Offset: 68 Read/Write Access: Read/Write This register controls enabling, operating mode (free-running or defined-count), and interrupt mask of Timer4. You can program each Timer4ControlReg to enable or disable a specific timer and to control its mode of operation.
<a href="#">TIMER4EOI</a>	0x48	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timer4 End-of-Interrupt Register Size: 1 bit Address Offset: 72 Read/Write Access: Read
<a href="#">TIMER4INTSTAT</a>	0x4c	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timer4 Interrupt Status Register Size: 1 bit Address Offset: 76 Read/Write Access: Read
<a href="#">TIMERSINTSTAT</a>	0xa0	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timers Interrupt Status Register Size: 1-9 bits Address Offset: 0xa0 Read/Write Access: Read
<a href="#">TIMERSEOI</a>	0xa4	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timers End-of-Interrupt Register Size: 1-9 bits Address Offset: 0xa4 Read/Write Access: Read
<a href="#">TIMERSRAWINTSTAT</a>	0xa8	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Timers Raw Interrupt Status Register Size: 1-9 bits Address Offset: 0xa8 Read/Write Access: Read
<a href="#">TIMERSCOMPVERSION</a>	0xac	32 bits	R	<b>Value After Reset:</b> 0x3230382a Name: Timers Component Version Size: 32 bits Address Offset: 0xac Read/Write Access: Read

### 10.1.2.2 Register and Field Descriptions

Following is a description of the individual registers of component DW\_apb\_timers

#### 10.1.2.2.1 TIMER1LOADCOUNT

**Size:** 32 bits

**Offset:** 0x0

**Memory Access:** R/W

**Value After Reset:** 0x0



Name: Timer1 Load Count Register Size: 8-32 bits Address Offset: 0x00 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:0	TIMER1LOADCOUNT	R/W	Value to be loaded into Timer1. This is the value from which counting

Bits	Name	Memory Access	Description
			commences. Any value written to this register is loaded into the associated timer.

#### 10.1.2.2.2 TIMER1CURRENTVAL

**Size:** 32 bits

**Offset:** 0x4

**Memory Access:** R

**Value After Reset:** 0x0

31:0
TIMER1CURRENTVAL

Name: Timer1 Current Value Size: 8-32 bits Address Offset: 4 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:0	TIMER1CURRENTVAL	R	Current Value of Timer1. This register is supported only when timer_1_clk is synchronous to pclk. Reading this register when using independent clocks results in an undefined value.

#### 10.1.2.2.3 TIMER1CONTROLREG

**Size:** 32 bits

**Offset:** 0x8

**Memory Access:** R/W

**Value After Reset:** 0x0

31:3	2	1	0
(undef)	TIMER_INTERRUPT_MASK	TIMER_MODE	TIMER_ENABLE

Name: Timer1 Control Register Size: 3 bits Address Offset: 8 Read/Write Access: Read/Write This register controls enabling, operating mode (free-running or defined-count), and interrupt mask of Timer1. You can program each Timer1ControlReg to enable or disable a specific timer and to control its mode of operation.

Bits	Name	Memory Access	Description
31:3			Reserved for future use.
2	TIMER_INTERRUPT_MASK	R/W	Timer interrupt mask for Timer1. 0: not masked 1: masked
1	TIMER_MODE	R/W	Timer mode for Timer1. 0: free-running mode 1: user-defined count mode NOTE: You must set the Timer1LoadCount register to all 1s before enabling the timer in free-running mode.
0	TIMER_ENABLE	R/W	Timer enable bit for Timer1. 0: disable 1: enable

#### 10.1.2.2.4 TIMER1EOI

**Size:** 32 bits

**Offset:** 0xc

**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
------	---

(undef)	TIMER1EOI
---------	-----------

Name: Timer1 End-of-Interrupt Register Size: 1 bit Address Offset: 12 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	TIMER1EOI	R	Reading from this register returns all zeroes (0) and clears the interrupt from Timer1.

#### 10.1.2.2.5 TIMER1INTSTAT

**Size:** 32 bits

**Offset:** 0x10

**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
(undef)	TIMER1INTSTAT

Name: Timer1 Interrupt Status Register Size: 1 bit Address Offset: 16 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	TIMER1INTSTAT	R	Contains the interrupt status for Timer1.

#### 10.1.2.2.6 TIMER2LOADCOUNT

**Size:** 32 bits

**Offset:** 0x14

**Memory Access:** R/W

**Value After Reset:** 0x0

31:0	TIMER2LOADCOUNT
------	-----------------

Name: Timer2 Load Count Register Size: 8-32 bits Address Offset: 20 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:0	TIMER2LOADCOUNT	R/W	Value to be loaded into Timer2. This is the value from which counting commences. Any value written to this register is loaded into the associated timer.

#### 10.1.2.2.7 TIMER2CURRENTVAL

**Size:** 32 bits

**Offset:** 0x18

**Memory Access:** R

**Value After Reset:** 0x0

31:0	TIMER2CURRENTVAL
------	------------------

Name: Timer2 Current Value Size: 8-32 bits Address Offset: 24 Read/Write Access: Read

Bits	Name	Memory Access	Description
------	------	---------------	-------------

Bits	Name	Memory Access	Description
31:0	TIMER2CURRENTVAL	R	Current Value of Timer2. This register is supported only when timer_2_clk is synchronous to pclk. Reading this register when using independent clocks results in an undefined value.

#### 10.1.2.2.8 TIMER2CONTROLREG

**Size:** 32 bits

**Offset:** 0x1c

**Memory Access:** R/W

**Value After Reset:** 0x0

31:3	2	1	0
(undef)	TIMER_INTERRUPT_MASK	TIMER_MODE	TIMER_ENABLE

Name: Timer2 Control Register Size: 32 bits Address Offset: 28 Read/Write Access: Read/Write This register controls enabling, operating mode (free-running or defined-count), and interrupt mask of Timer2. You can program each Timer1ControlReg to enable or disable a specific timer and to control its mode of operation.

Bits	Name	Memory Access	Description
31:3			Reserved for future use.
2	TIMER_INTERRUPT_MASK	R/W	Timer interrupt mask for Timer2. 0: not masked 1: masked
1	TIMER_MODE	R/W	Timer mode for Timer2. 0: free-running mode 1: user-defined count mode NOTE: You must set the Timer1LoadCount register to all 1s before enabling the timer in free-running mode.
0	TIMER_ENABLE	R/W	Timer enable bit for Timer2. 0: disable 1: enable

#### 10.1.2.2.9 TIMER2EOI

**Size:** 32 bits

**Offset:** 0x20

**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
(undef)	TIMER2EOI

Name: Timer2 End-of-Interrupt Register Size: 1 bit Address Offset: 32 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	TIMER2EOI	R	Reading from this register returns all zeroes (0) and clears the interrupt from Timer2.

#### 10.1.2.2.10 TIMER2INTSTAT

**Size:** 32 bits

**Offset:** 0x24

**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
------	---

(undef)	TIMER2INTSTAT
---------	---------------

Name: Timer2 Interrupt Status Register Size: 1 bit Address Offset: 36 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	TIMER2INTSTAT	R	Contains the interrupt status for Timer2.

#### 10.1.2.2.11 TIMER3LOADCOUNT

**Size:** 32 bits

**Offset:** 0x28

**Memory Access:** R/W

**Value After Reset:** 0x0

31:0	TIMER3LOADCOUNT
------	-----------------

Name: Timer1 Load Count Register Size: 8-32 bits Address Offset: 40 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:0	TIMER3LOADCOUNT	R/W	Value to be loaded into Timer3. This is the value from which counting commences. Any value written to this register is loaded into the associated timer.

#### 10.1.2.2.12 TIMER3CURRENTVAL

**Size:** 32 bits

**Offset:** 0x2c

**Memory Access:** R

**Value After Reset:** 0x0

31:0	TIMER3CURRENTVAL
------	------------------

Name: Timer3 Current Value Size: 8-32 bits Address Offset: 44 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:0	TIMER3CURRENTVAL	R	Current Value of Timer3. This register is supported only when timer_3_clk is synchronous to pclk. Reading this register when using independent clocks results in an undefined value.

#### 10.1.2.2.13 TIMER3CONTROLREG

**Size:** 32 bits

**Offset:** 0x30

**Memory Access:** R/W

**Value After Reset:** 0x0

31:3	2	1	0
(undef)	TIMER_INTERRUPT_MASK	TIMER_MODE	TIMER_ENABLE

Name: Timer3 Control Register Size: 3 bits Address Offset: 48 Read/Write Access: Read/Write This register controls enabling, operating mode (free-running or defined-count), and interrupt mask of Timer3. You can program each Timer1ControlReg to enable or disable a specific timer and to control its mode of operation.

Bits	Name	Memory Access	Description
31:3			Reserved for future use.
2	TIMER_INTERRUPT_MASK	R/W	Timer interrupt mask for Timer3. 0: not masked 1: masked
1	TIMER_MODE	R/W	Timer mode for Timer3. 0: free-running mode 1: user-defined count mode NOTE: You must set the Timer1LoadCount register to all 1s before enabling the timer in free-running mode.
0	TIMER_ENABLE	R/W	Timer enable bit for Timer3. 0: disable 1: enable

**10.1.2.2.14 TIMER3EOI****Size:** 32 bits**Offset:** 0x34**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	TIMER3EOI

Name: Timer1 End-of-Interrupt Register Size: 1 bit Address Offset: 52 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	TIMER3EOI	R	Reading from this register returns all zeroes (0) and clears the interrupt from Timer1.

**10.1.2.2.15 TIMER3INTSTAT****Size:** 32 bits**Offset:** 0x38**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	TIMER3INTSTAT

Name: Timer3 Interrupt Status Register Size: 1 bit Address Offset: 56 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	TIMER3INTSTAT	R	Contains the interrupt status for Timer3.

**10.1.2.2.16 TIMER4LOADCOUNT****Size:** 32 bits**Offset:** 0x3c**Memory Access:** R/W**Value After Reset:** 0x0

31:0	
(undef)	TIMER4LOADCOUNT

Name: Timer4 Load Count Register Register Size: 8-32 bits Address Offset: 60 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:0	TIMER4LOADCOUNT	R/W	Value to be loaded into Timer4. This is the value from which counting commences. Any value written to this register is loaded into the associated timer.

**10.1.2.2.17 TIMER4CURRENTVAL****Size:** 32 bits**Offset:** 0x40**Memory Access:** R**Value After Reset:** 0x0

31:0	TIMER4CURRENTVAL
------	------------------

Name: Timer4 Current Value Size: 8-32 bits Address Offset: 64 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:0	TIMER4CURRENTVAL	R	Current Value of Timer4. This register is supported only when timer_3_clk is synchronous to pclk. Reading this register when using independent clocks results in an undefined value.

**10.1.2.2.18 TIMER4CONTROLREG****Size:** 32 bits**Offset:** 0x44**Memory Access:** R/W**Value After Reset:** 0x0

31:3	2	1	0
(undef)	TIMER_INTERRUPT_MASK	TIMER_MODE	TIMER_ENABLE

Name: Timer4 Control Register Size: 3 bits Address Offset: 68 Read/Write Access: Read/Write This register controls enabling, operating mode (free-running or defined-count), and interrupt mask of Timer4. You can program each Timer4ControlReg to enable or disable a specific timer and to control its mode of operation.

Bits	Name	Memory Access	Description
31:3			Reserved for future use.
2	TIMER_INTERRUPT_MASK	R/W	Timer interrupt mask for Timer4. 0: not masked 1: masked
1	TIMER_MODE	R/W	Timer mode for Timer4. 0: free-running mode 1: user-defined count mode NOTE: You must set the Timer4LoadCount register to all 1s before enabling the timer in free-running mode.
0	TIMER_ENABLE	R/W	Timer enable bit for Timer4. 0: disable 1: enable

**10.1.2.2.19 TIMER4EOI****Size:** 32 bits**Offset:** 0x48**Memory Access:** R**Value After Reset:** 0x0

31:1	0
------	---

(undef)	TIMER4EOI
---------	-----------

Name: Timer4 End-of-Interrupt Register Size: 1 bit Address Offset: 72 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	TIMER4EOI	R	Reading from this register returns all zeroes (0) and clears the interrupt from Timer4.

#### 10.1.2.2.20 TIMER4INTSTAT

**Size:** 32 bits

**Offset:** 0x4c

**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
(undef)	TIMER4INTSTAT

Name: Timer4 Interrupt Status Register Size: 1 bit Address Offset: 76 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	TIMER4INTSTAT	R	Contains the interrupt status for Timer4.

#### 10.1.2.2.21 TIMERSINTSTAT

**Size:** 32 bits

**Offset:** 0xa0

**Memory Access:** R

**Value After Reset:** 0x0

31:5	4:0
(undef)	TIMERSINTSTAT

Name: Timers Interrupt Status Register Size: 1-9 bits Address Offset: 0xa0 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:5			Reserved for future use.
4:0	TIMERSINTSTAT	R	Contains the interrupt status of all timers in the component. If a bit of this register is 0, then the corresponding timer interrupt is not active and the corresponding interrupt could be on either the timer_intr bus or the timer_intr_n bus, depending on the interrupt polarity you have chosen. Similarly, if a bit of this register is 1, then the corresponding interrupt bit has been set in the relevant interrupt bus. In both cases, the status reported is the status after the interrupt mask has been applied. Reading from this register does not clear any active interrupts: 0 = either timer_intr or timer_intr_n is not active after masking 1 = either timer_intr or timer_intr_n is active after masking.

#### 10.1.2.2.22 TIMERSEOI

**Size:** 32 bits

**Offset:** 0xa4

**Memory Access:** R

**Value After Reset:** 0x0

31:5	4:0
(undef)	TIMERSEOI

Name: Timers End-of-Interrupt Register Size: 1-9 bits Address Offset: 0xa4 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:5			Reserved for future use.
4:0	TIMERSEOI	R	Reading this register returns all zeroes (0) and clears all active interrupts.

### 10.1.2.2.23 TIMERSRAWINTSTAT

**Size:** 32 bits

**Offset:** 0xa8

**Memory Access:** R

**Value After Reset:** 0x0

31:5	4:0
(undef)	TIMERSRAWINTSTAT

Name: Timers Raw Interrupt Status Register Size: 1-9 bits Address Offset: 0xa8 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:5			Reserved for future use.
4:0	TIMERSRAWINTSTAT	R	The register contains the unmasked interrupt status of all timers in the component. 0 = either timer_intr or timer_intr_n is not active prior to masking 1 = either timer_intr or timer_intr_n is active prior to masking.

### 10.1.2.2.24 TIMERSCOMPVERSION

**Size:** 32 bits

**Offset:** 0xac

**Memory Access:** R

**Value After Reset:** 0x3230382a

31:0
TIMERSCOMPVERSION

Name: Timers Component Version Size: 32 bits Address Offset: 0xac Read/Write Access: Read

Bits	Name	Memory Access	Description
31:0	TIMERSCOMPVERSION	R	Current revision number of the DW_apb_timers component.

## 10.2 Watchdog Timer

### 10.2.1 Features

The WDT supports the following features:

- Configurable APB data bus widths of 32 bits-
- Configurable watchdog counter width of 32 bits-

- Counter counts down from a preset value to 0 to indicate the occurrence of a timeout.
- If a timeout occurs the WDT can perform one of the following operations:
  - Generate a system reset
  - First generate an interrupt and if this is not cleared by the service routine by the time a second timeout occurs then generate a system reset
- Programmable timeout range (period). The option of hard coding this value during **configuration** is available to reduce the register requirements.
- Programmable and hard coded reset pulse length.
- Prevention of accidental restart of the WDT counter.
- Prevention of accidental disabling of the WDT.

## 10.2.2 Function Description

### 10.2.2.1 Counter

The WDT counts from a preset (timeout) value in descending order to zero. When the counter reaches zero, depending on the output response mode selected, either a system reset or an interrupt occurs. When the counter reaches zero, it wraps to the selected timeout value and continues decrementing. The user can restart the counter to its initial value. This is programmed by writing to the restart register at any time. The process of restarting the watchdog counter is sometimes referred to as *kicking the dog*. As a safety feature to prevent accidental restarts, the value 0x76 must be written to the Current Counter Value Register (WDT\_CRR).

### 10.2.2.2 Interrupts

The WDT can be programmed to generate an interrupt (and then a system reset) when a timeout occurs. When a 1 is written to the response mode field (RMOD, bit 1) of the Watchdog Timer Control Register (WDT\_CR), the WDT generates an interrupt. If it is not cleared by the time a second timeout occurs, then it generates a system reset. If a restart occurs at the same time the watchdog counter reaches zero, an interrupt is not generated.

### 10.2.2.3 System Resets

When a 0 is written to the output response mode field (RMOD, bit 1) of the Watchdog Timer Control Register (WDT\_CR), the WDT generates a system reset when a timeout occurs. The WDT can be **configured** so that it is always enabled upon reset of the WDT. If this is the case, it overrides whatever has been written in bit 0 of the WDT\_CR register (the WDT enable field).

If a restart occurs at the same time the watchdog counter reaches zero, a system reset is not generated.

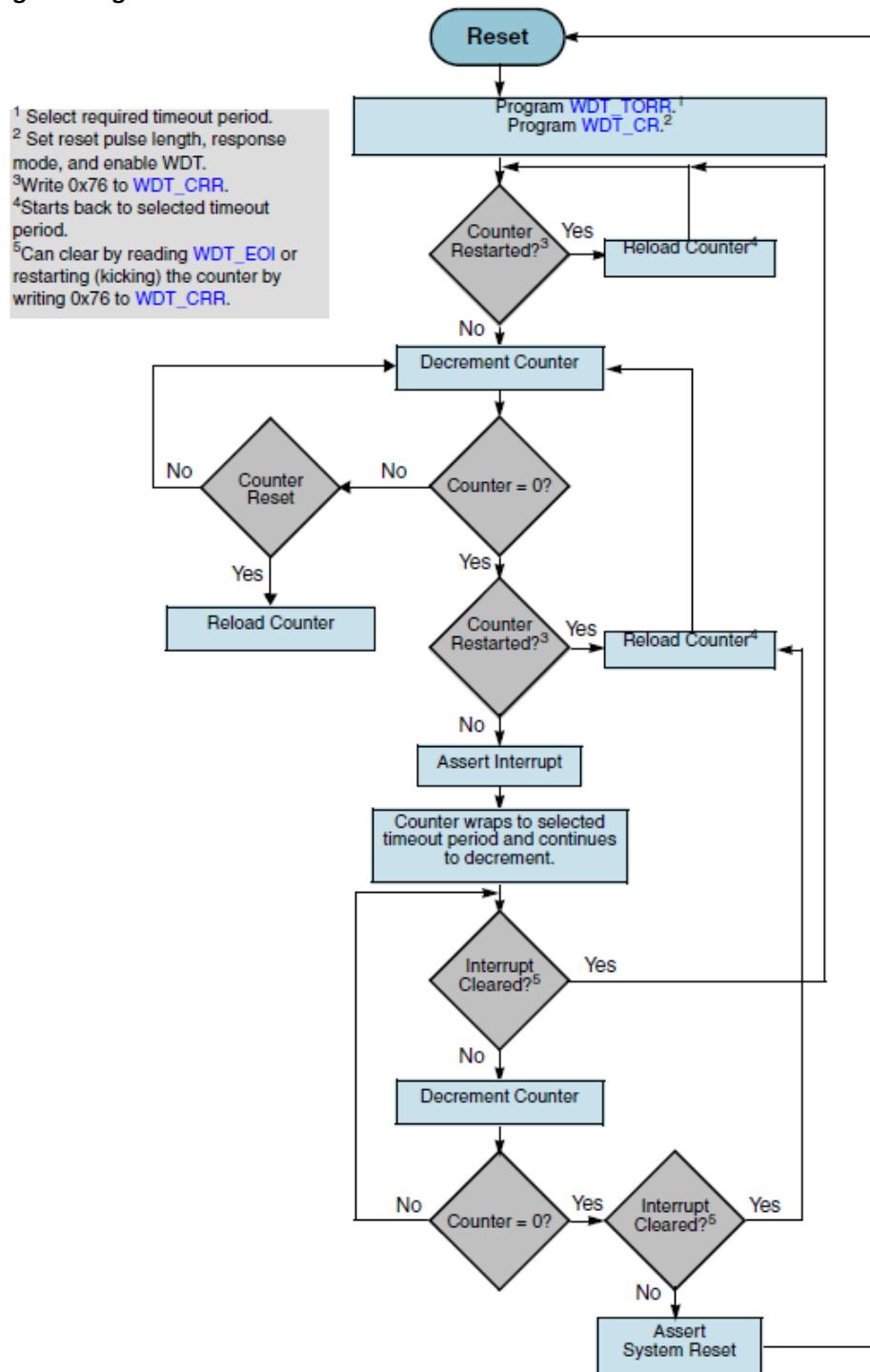
### 10.2.2.4 Reset Pulse Length

The reset pulse length is the number of pclk cycles for which a system reset is asserted. When a system reset is generated, it remains asserted for the number of cycles specified by the reset pulse length plus two cycles of synchronization delay, or until the system is reset. A counter restart has no effect on the system reset once it has been asserted.

### 10.2.2.5 Timeout Period Values

The WDT can be **configured** to have a fixed or user-defined timeout period range. In both cases, the values that may be selected are limited by the WDT counter width. If the timeout period range that is selected is greater than the counter width, the timeout period is truncated to fit to the counter width. In the case of user-defined timeout period ranges, the value is limited at the time of **configuration**, and values greater than the count are not accepted.

### 10.2.3 Function Programming



### 10.2.4 WDT Registers

#### 10.2.4.1 Register Memory Maps

Register	Offset	Size	Memory Access	Description
wdt_address_block				
<a href="#">WDT_CR</a>	0x0	32 bits	R/W	Value After Reset: 0x0

Register	Offset	Size	Memory Access	Description
				Control Register
<a href="#">WDT_TORR</a>	0x4	32 bits	R/W	<b>Value After Reset:</b> 0x0 Timeout Range Register
<a href="#">WDT_CCVR</a>	0x8	32 bits	R	<b>Value After Reset:</b> 0xffff Current Counter Value Register.
<a href="#">WDT_CRR</a>	0xc	32 bits	W	<b>Value After Reset:</b> 0x0 Counter Restart Register.
<a href="#">WDT_STAT</a>	0x10	32 bits	R	<b>Value After Reset:</b> 0x0 Interrupt Status Register.
<a href="#">WDT_EOI</a>	0x14	32 bits	R	<b>Value After Reset:</b> 0x0 Interrupt Clear Register.
WDT_COMP_PARAM_5	0xe4	32 bits	R	<b>Value After Reset:</b> 0x0 Component Parameters Register 5
WDT_COMP_PARAM_4	0xe8	32 bits	R	<b>Value After Reset:</b> 0x0 Component Parameters Register 4
WDT_COMP_PARAM_3	0xec	32 bits	R	<b>Value After Reset:</b> 0x0 Component Parameters Register 3
WDT_COMP_PARAM_2	0xf0	32 bits	R	<b>Value After Reset:</b> 0xffff Component Parameters Register 2
<a href="#">WDT_COMP_PARAM_1</a>	0xf4	32 bits	R	<b>Value After Reset:</b> 0x10000240 Component Parameters Register 1
<a href="#">WDT_COMP_VERSION</a>	0xf8	32 bits	R	<b>Value After Reset:</b> 0x3130372a Component Version Register
<a href="#">WDT_COMP_TYPE</a>	0xfc	32 bits	R	<b>Value After Reset:</b> 0x44570120 Component Type Register

#### 10.2.4.2 Register and Field Descriptions

Following is a description of the individual registers of component DW\_apb\_wdt

##### 10.2.4.2.1 WDT\_CR

**Size:** 32 bits

**Offset:** 0x0

**Memory Access:** R/W

**Value After Reset:** 0x0

31:5	4:2	1	0
(undef)	RPL	RMOD	WDT_EN

Control Register

Bits	Name	Memory Access	Description
31:5			Reserved for future use.
4:2	RPL	R/W	Reset pulse length. Writes have no effect when the configuration parameter WDT_HC_RPL is

Bits	Name	Memory Access	Description
			1, making the register bits read-only. This is used to select the number of pclk cycles for which the system reset stays asserted. The range of values available is 2 to 256 pclk cycles. 000 - 2 pclk cycles 001 - 4 pclk cycles 010 - 8 pclk cycles 011 - 16 pclk cycles 100 - 32 pclk cycles 101 - 64 pclk cycles 110 - 128 pclk cycles 111 - 256 pclk cycles
1	RMOD	R/W	Response mode. Writes have no effect when the parameter WDT_HC_RMOD = 1, thus this register becomes read-only. Selects the output response generated to a timeout. 0 = Generate a system reset. 1 = First generate an interrupt and if it is not cleared by the time a second timeout occurs then generate a system reset.
0	WDT_EN	R/W	WDT enable. Writable when the configuration parameter WDT_ALWAYS_EN = 1, otherwise, it is readable. This bit is used to enable and disable the DW_apb_wdt. When disabled, the counter does not decrement. Thus, no interrupts or system resets are generated. Once this bit has been enabled, it can be cleared only by a system reset. 0 = WDT disabled. 1 = WDT enabled.

#### 10.2.4.2.2 WDT\_TORR

**Size:** 32 bits

**Offset:** 0x4

**Memory Access:** R/W

**Value After Reset:** 0x0

31:8	7:4	3:0
Reserved	(undef)	TOP

Timeout Range Register

Bits	Name	Memory Access	Description
31:8	Reserved	R	Reserved and read as 0.
7:4			Reserved for future use.
3:0	TOP	R/W	Timeout period. Writes have no effect when the configuration parameter WDT_HC_TOP = 1, thus making this register read-only. This field is used to select the timeout period from which the watchdog counter restarts. A change of the timeout period takes effect only after the next counter restart (kick). The range of values is limited by the WDT_CNT_WIDTH. If TOP is programmed to select a range that is greater than the counter width, the timeout period is truncated to fit to the counter width. This affects only the non-user specified values as users are limited to these boundaries during configuration. The range of values available for a 32-bit watchdog counter are: Where i = TOP and t = timeout period For i = 0 to 15 if WDT_USE_FIX_TOP==1 t = 2(16 + i) else t = WDT_USER_TOP_(i)

#### 10.2.4.2.3 WDT\_CCVR

**Size:** 32 bits

**Offset:** 0x8

**Memory Access:** R

**Value After Reset:** 0xfffff

31:0
wdt_ccvr

Current Counter Value Register.

Bits	Name	Memory Access	Description
31:0	wdt_ccvr	R	This register, when read, is the current value of the internal counter. This value is read coherently when ever it is read, which is relevant when the APB_DATA_WIDTH is less than the counter width.

**10.2.4.2.4 WDT\_CRR****Size:** 32 bits**Offset:** 0xc**Memory Access:** W**Value After Reset:** 0x0

31:8	7:0
(undef)	wdt_crr

Counter Restart Register.

Bits	Name	Memory Access	Description
31:8			Reserved for future use.
7:0	wdt_crr	W	This register is used to restart the WDT counter. As a safety feature to prevent accidental restarts, the value 0x76 must be written. A restart also clears the WDT interrupt. Reading this register returns zero.

**10.2.4.2.5 WDT\_STAT****Size:** 32 bits**Offset:** 0x10**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	wdt_stat

Interrupt Status Register.

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	wdt_stat	R	This register shows the interrupt status of the WDT. 1 = Interrupt is active regardless of polarity. 0 = Interrupt is inactive.

**10.2.4.2.6 WDT\_EOI****Size:** 32 bits**Offset:** 0x14**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	wdt_eoi

Interrupt Clear Register.

Bits	Name	Memory Access	Description														
31:1			Reserved for future use.														
0	wdt_eoi	R	Clears the watchdog interrupt. This can be used to clear the interrupt without restarting the watchdog counter.														

#### 10.2.4.2.7 WDT\_COMP\_PARAM\_1

**Size:** 32 bits

**Offset:** 0xf4

**Memory Access:** R

**Value After Reset:** 0x10000240

31:2 9	28:24	23:20	19:16	15:1 3	12:10	9:8	7	6	5	4	3	2	1	0				
RSV_D_3_1_2_9	WDT_CNT_WIDTH_H	WDT_DFLT_TO_P_INIT	WDT_DFLT_TOP	RSV_D_1_5_1_3	WDT_DFLT_RPL	APB_DATA_WIDTH	WDT_PAUSE	WDT_USE_FIX_TOP	WDT_HC_TOP	WDT_HC_RPL	WDT_HC_RMOD	WDT_DUAL_TOP	WDT_DFLT_RMOD	WDT_ALWAYS_EN				

Component Parameters Register 1

Bits	Name	Memory Access	Description
31:29	RSVD_31_29	R	<b>Value After Reset:</b> 0x0
28:24	WDT_CNT_WIDTH	R	<b>Value After Reset:</b> 0x10
23:20	WDT_DFLT_TOP_INIT	R	<b>Value After Reset:</b> 0x0
19:16	WDT_DFLT_TOP	R	<b>Value After Reset:</b> 0x0
15:13	RSVD_15_13	R	<b>Value After Reset:</b> 0x0
12:10	WDT_DFLT_RPL	R	<b>Value After Reset:</b> 0x0
9:8	APB_DATA_WIDTH	R	<b>Value After Reset:</b> 0x2
7	WDT_PAUSE	R	<b>Value After Reset:</b> 0x0
6	WDT_USE_FIX_TOP	R	<b>Value After Reset:</b> 0x1
5	WDT_HC_TOP	R	<b>Value After Reset:</b> 0x0
4	WDT_HC_RPL	R	<b>Value After Reset:</b> 0x0
3	WDT_HC_RMOD	R	<b>Value After Reset:</b> 0x0
2	WDT_DUAL_TOP	R	<b>Value After Reset:</b> 0x0
1	WDT_DFLT_RMOD	R	<b>Value After Reset:</b> 0x0
0	WDT_ALWAYS_EN	R	<b>Value After Reset:</b> 0x0

#### 10.2.4.2.8 WDT\_COMP\_VERSION

**Size:** 32 bits

**Offset:** 0xf8

**Memory Access:** R

**Value After Reset:** 0x3130372a

31:0
wdt_comp_version

Component Version Register

Bits	Name	Memory Access	Description
31:0	wdt_comp_version	R	ASCII value for each number in the version, followed by *. For example, 32_30_31_2A represents the version 2.01*. Reset Value: See the Releases table in the DW_apb_rtc Release Notes.

**10.2.4.2.9 WDT\_COMP\_TYPE****Size:** 32 bits**Offset:** 0xfc**Memory Access:** R**Value After Reset:** 0x44570120

31:0
wdt_comp_type

Component Type Register

Bits	Name	Memory Access	Description
31:0	wdt_comp_type	R	Component Type Register

## 11 UART

The UART is a programmable Universal Asynchronous Receiver/Transmitter (UART).

### 11.1 UART Features

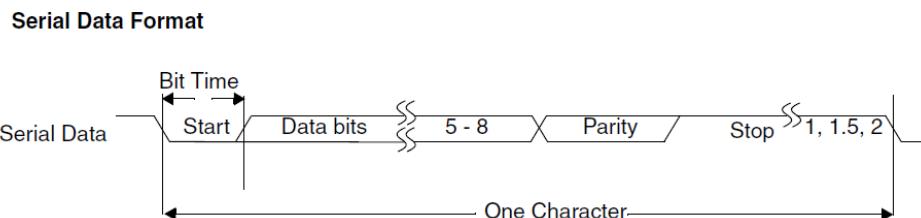
- Configurable parameters for the following:
  - APB data bus widths of 32
  - Transmit and receive FIFO depths of 16

### 11.2 UART Function Description

This chapter describes the functional operation of UART.

#### 11.2.1 UART (RS232) Serial Protocol

Because the serial communication between the UART and a selected device is asynchronous, additional bits (start and stop) are added to the serial data to indicate the beginning and end. Utilizing these bits allows two devices to be synchronized. This structure of serial data—accompanied by start and stop bits—is referred to as a character, as shown in following Figure.



An additional parity bit can be added to the serial character. This bit appears after the last data bit and before the stop bit(s) in the character structure in order to provide the UART with the ability to perform simple error checking on the received data. The UART Line Control Register is used to control the serial character characteristics. The individual bits of the data word are sent after the start bit, starting with the least significant bit (LSB). These are followed by the optional parity bit, followed by the stop bit(s), which can be 1, 1.5, or 2.

All the bits in the transmission are transmitted for exactly the same time duration; the exception to this is the half-stop bit when 1.5 stop bits are used. This duration is referred to as a Bit Period or Bit Time; one Bit Time equals sixteen baud clocks.

To ensure stability on the line, the receiver samples the serial input data at approximately the midpoint of the Bit Time once the start bit has been detected. Because the exact number of baud clocks is known for which each bit was transmitted, calculating the midpoint for sampling is not difficult; that is, every sixteen baud clocks after the midpoint sample of the start bit.

Together with serial input debouncing, this sampling helps to avoid the detection of false start bits. Short glitches are filtered out by debouncing, and no transition is detected on the line. If a glitch is wide enough to avoid filtering by debouncing, a falling edge is detected. However, a start bit is detected only if the line is again sampled low after half a bit time has elapsed.

#### 11.2.2 FIFO Support

UART implement FIFOs that buffer transmit and receive data.

Data can be written to the transmit FIFO as normal; however no serial transmission occurs in this mode—normal operation halted—and thus no data leave the FIFO. The data that has been written to the transmit FIFO can be read back with the Transmit FIFO Read (TFR) register, which when read gives the current data at the top of the transmit FIFO.

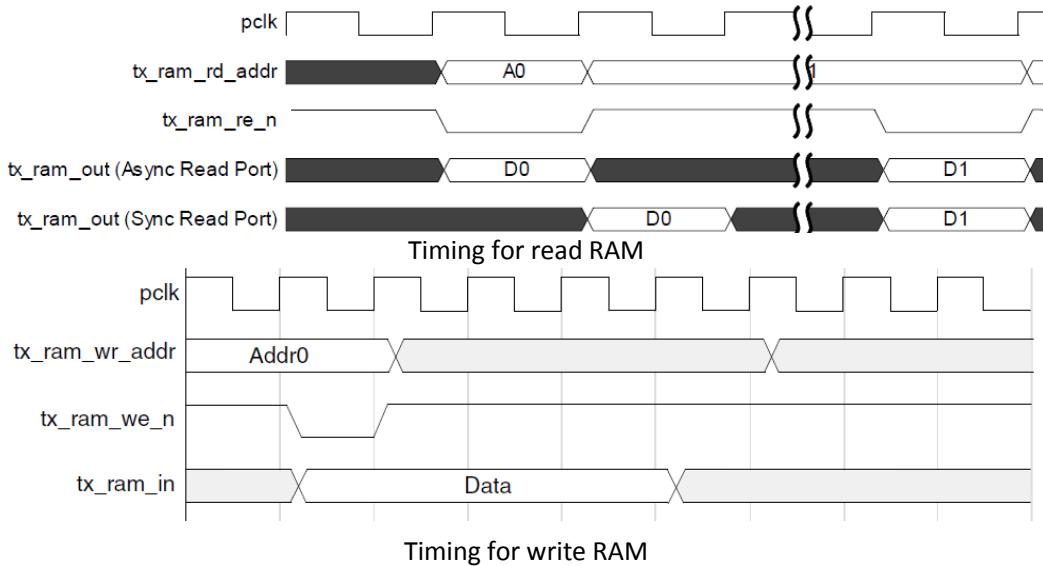
Similarly, data can be read from the receive FIFO as normal. Since the normal operation of the UART is halted in this mode, data must be written to the receive FIFO so the data can be read back.

Data is written to the receive FIFO using the Receive FIFO Write (RFW) register. The upper two bits of the 10-bit register are used to write framing error and parity error detection information to the receive FIFO, as follows:

- RFW[9] indicates framing error
- RFW[8] indicates parity error

Although these bits cannot be read back through the Receive Buffer Register, they can be checked by reading the Line Status Register and checking the corresponding bits when the data in question is at the top of the receive FIFO.

The following figures show the timing diagram for read and write for asynchronous and synchronous RAM.



### 11.2.3 Back-to-Back Character Stream Transmission

This section describes:

- Scenarios under which the UART is capable of transmitting back-to-back characters on the serial interface, with no idle time between them
- Worst-case idle time that exists between back-to-back characters

When the Transmit FIFO contains multiple data entries, the UART transmits the characters in the FIFO back-to-back on the serial bus.

#### 11.2.3.1 Single Clock Mode

There is no idle time between back-to-back characters if data is ready in the transmit FIFO. In this case, because *sync\_delay* equals one *pclk*, the requirement to avoid idle time between consecutive characters is met for all {DLH,DLL} values.

$$\text{sync\_delay} \leq \{\text{DLH}, \text{DLL}\} * \text{sclk}$$

For example, when {DLH, DLL} equals 1, then

$$1 \text{ pclk} \leq 1 * \text{pclk}$$

### 11.2.4 Interrupts

Assertion of the UART interrupt output signal (intr)—a positive-level interrupt—occurs whenever one of the several prioritized interrupt types are enabled and active.

When an interrupt occurs, the master accesses the IIR register.

The following interrupt types can be enabled with the IER register:

- Receiver Error
- Receiver Data Available
- Character Timeout (in FIFO mode only)
- Transmitter Holding Register Empty at/below threshold (in Programmable THRE interrupt mode)
- Modem Status
- Busy Detect Indication

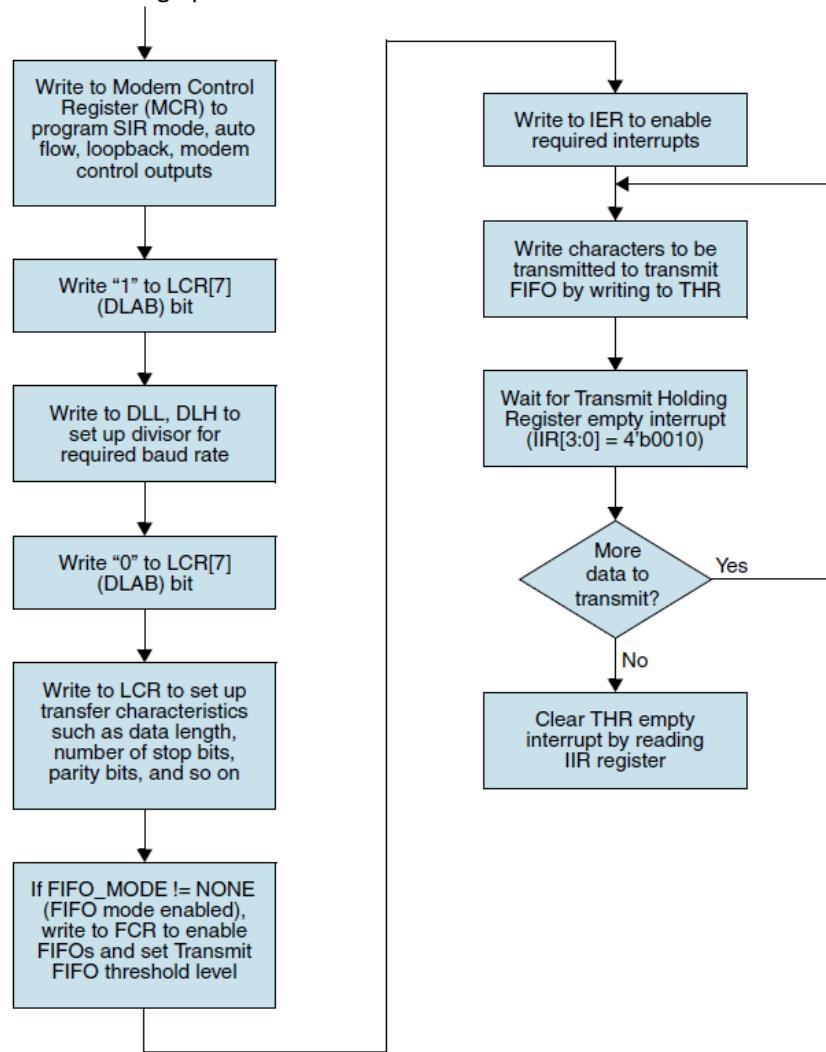
### 11.2.5 Auto Flow Control

The UART can be [configured](#) to have a 16750-compatible Auto RTS and Auto CTS serial data flow control mode available; if FIFOs are not implemented, this mode cannot be selected. When Auto Flow Control is not selected, none of the corresponding logic is implemented and the mode cannot be enabled, reducing overall gate counts. When Auto Flow Control mode is selected, it can be enabled with the Modem Control Register (MCR[5]).

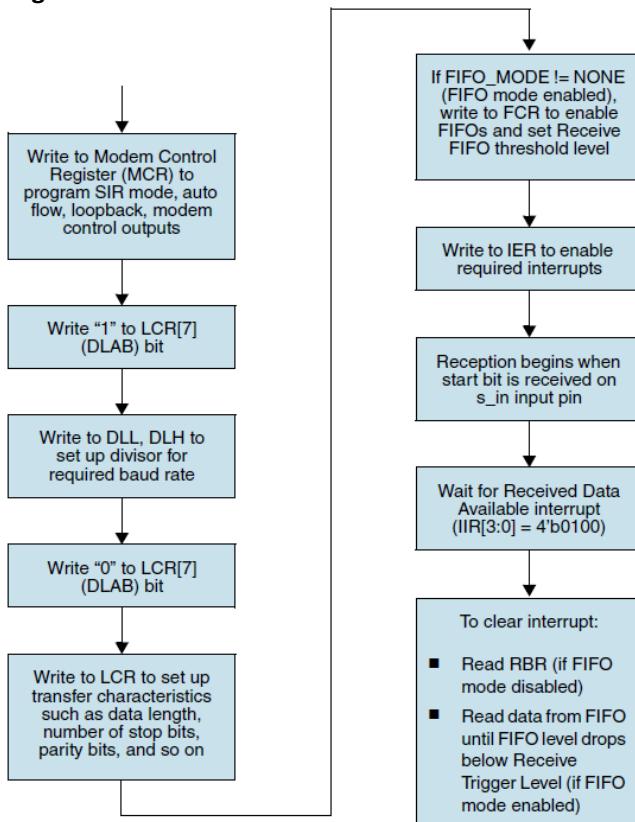
## 11.3 UART Programming

### 11.3.1 UART Transmit Programming Flowchart

The programming sequence for setting up the UART for transmission:



### 11.3.2 UART Receive Programming Flowchart



## 11.4 UART Registers

### 11.4.1 Register Memory Maps

Register	Offset	Size	Memory Access	Description
uart_address_block				
<a href="#">THR</a>	0x0	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>AlternateRegister:</b> RBR Receive Buffer Register, reading this register when the DLAB bit is zero; Transmit Holding Register, writing to this register when the DLAB is zero; Divisor Latch (Low), when DLAB bit is one
<a href="#">DLL</a>	0x0	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>AlternateRegister:</b> RBR Receive Buffer Register, reading this register when the DLAB bit is zero; Transmit Holding Register, writing to this register when the DLAB is zero; Divisor Latch (Low), when DLAB bit is one
<a href="#">RBR</a>	0x0	32 bits	R	<b>Value After Reset:</b> 0x0 Receive Buffer Register, reading this register when the DLAB bit is zero; Transmit Holding Register, writing to this register when the DLAB is zero; Divisor Latch (Low), when DLAB bit is one
<a href="#">DLH</a>	0x4	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>AlternateRegister:</b> IER

Register	Offset	Size	Memory Access	Description
				Divisor Latch High (DLH) Register. Interrupt Enable Register, when the DLAB bit is zero; Divisor Latch (High), when the DLAB bit is one. This register makes up the upper 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. If UART_16550_COMPATIBLE == NO, then this register may only be accessed when the DLAB bit (LCR[7]) is set and the UART is not busy (USR[0] is zero), otherwise this register may only be accessed when the DLAB bit (LCR[7]) is set. The output baud rate is equal to the serial clock (pclk if one clock design, sclk if two clock design (CLOCK_MODE == Enabled)) frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock freq) / (16 * divisor) Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications will occur. Also, once the DLH is set, at least 8 clock cycles of the slowest DW_apb_uart clock should be allowed to pass before transmitting or receiving data.
<a href="#">IER</a>	0x4	32 bits	R/W	<b>Value After Reset:</b> 0x0 Interrupt Enable Register: Interrupt Enable Register, when the DLAB bit is zero; Divisor Latch (High), when the DLAB bit is one. Each of the bits used has a different function and will be detailed in the bit field descriptions.
<a href="#">FCR</a>	0x8	32 bits	R/W	<b>Value After Reset:</b> 0x0 <b>AlternateRegister:</b> IIR FIFO Control Register. This register is only valid when the DW_apb_uart is configured to have FIFO's implemented (FIFO_MODE != NONE). If FIFO's are not implemented, this register does not exist and writing to this register address will have no effect.
<a href="#">IIR</a>	0x8	32 bits	R	<b>Value After Reset:</b> 0x1 Interrupt Identification Register
<a href="#">LCR</a>	0xc	32 bits	R/W	<b>Value After Reset:</b> 0x0 Line Control Register
<a href="#">MCR</a>	0x10	32 bits	R/W	<b>Value After Reset:</b> 0x0 Modem Control Register
<a href="#">LSR</a>	0x14	32 bits	R	<b>Value After Reset:</b> 0x60 Line Status Register
<a href="#">MSR</a>	0x18	32 bits	R	<b>Value After Reset:</b> 0x0 Modem Status Register It should be noted that whenever bits 0, 1, 2 or 3 is set to logic one, to indicate a change on the modem control inputs, a modem status interrupt will be generated if enabled via the IER regardless of when the change occurred. Since the delta bits (bits 0, 1, 3) can get set after a reset if their respective modem signals are active (see individual bits for details), a read of the MSR after reset can be performed to prevent unwanted interrupts.
<a href="#">SCR</a>	0x1c	32 bits	R/W	<b>Value After Reset:</b> 0x0 Scratchpad Register
<a href="#">FAR</a>	0x70	32 bits	R	<b>Value After Reset:</b> 0x0 FIFO Access Register
<a href="#">USR</a>	0x7c	32 bits	R	<b>Value After Reset:</b> 0x0 UART Status register.
<a href="#">HTX</a>	0xa4	32	R/W	<b>Value After Reset:</b> 0x0

Register	Offset	Size	Memory Access	Description
		bits		Halt TX
<a href="#">DMASA</a>	0xa8	32 bits	R	<b>Value After Reset:</b> 0x0 DMA Software Acknowledge

### 11.4.2 Register and Field Descriptions

Following is a description of the individual registers of UART\_0. UART\_1 has the same register description.

#### 11.4.2.1 THR

**Size:** 32 bits

**Offset:** 0x0

**Memory Access:** R/W

**Value After Reset:** 0x0

31:8	7:0
RSVD_THR_31to8	thr

Receive Buffer Register, reading this register when the DLAB bit is zero; Transmit Holding Register, writing to this register when the DLAB is zero; Divisor Latch (Low), when DLAB bit is one

Bits	Name	Memory Access	Description
31:8	RSVD_THR_31to8	R	Reserved bits [31:8] - Read Only
7:0	thr	W	Transmit Holding Register: This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If in non-FIFO mode or FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If in FIFO mode and FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.

#### 11.4.2.2 DLL

**Size:** 32 bits

**Offset:** 0x0

**Memory Access:** R/W

**Value After Reset:** 0x0

31:8	7:0
RSVD_DLL_31to8	dll

Receive Buffer Register, reading this register when the DLAB bit is zero; Transmit Holding Register, writing to this register when the DLAB is zero; Divisor Latch (Low), when DLAB bit is one

Bits	Name	Memory Access	Description
31:8	RSVD_DLL_31to8	R	Reserved bits [31:8] - Read Only

Bits	Name	Memory Access	Description
7:0	dll	R/W	Divisor Latch (Low): This register makes up the lower 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. If <b>UART_16550_COMPATIBLE == No</b> , then this register may only be accessed when the DLAB bit (LCR[7]) is set and the UART is not busy (USR[0] is zero), otherwise this register may only be accessed when the DLAB bit (LCR[7]) is set. The output baud rate is equal to the serial clock (pclk if one clock design, sclk if two clock design ( <b>CLOCK_MODE == Enabled</b> )) frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock freq) / (16 * divisor) Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications will occur. Also, once the DLL is set, at least 8 clock cycles of the slowest DW_apb_uart clock should be allowed to pass before transmitting or receiving data.

#### 11.4.2.3 RBR

**Size:** 32 bits

**Offset:** 0x0

**Memory Access:** R

**Value After Reset:** 0x0

31:8	7:0
RSVD_RBR_31to8	rbr

Receive Buffer Register, reading this register when the DLAB bit is zero; Transmit Holding Register, writing to this register when the DLAB is zero; Divisor Latch (Low), when DLAB bit is one

Bits	Name	Memory Access	Description
31:8	RSVD_RBR_31to8	R	Reserved bits [31:8] - Read Only
7:0	rbr	R	Receive Buffer Register: This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If in non-FIFO mode ( <b>FIFO_MODE == NONE</b> ) or FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If in FIFO mode ( <b>FIFO_MODE != NONE</b> ) and FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.

#### 11.4.2.4 DLH

**Size:** 32 bits

**Offset:** 0x4

**Memory Access:** R/W

**Value After Reset:** 0x0

31:8	7:0
RSVD_DLH_31to8	dlh

Divisor Latch High (DLH) Register. Interrupt Enable Register, when the DLAB bit is zero; Divisor Latch (High), when the DLAB bit is one.

This register makes up the upper 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. If `UART_16550_COMPATIBLE == NO`, then this register may only be accessed when the DLAB bit (LCR[7]) is set and the UART is not busy (`USR[0]` is zero), otherwise this register may only be accessed when the DLAB bit (LCR[7]) is set. The output baud rate is equal to the serial clock (pclk if one clock design, sclk if two clock design (`CLOCK_MODE == Enabled`)) frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock freq) / (16 \* divisor). Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications will occur. Also, once the DLH is set, at least 8 clock cycles of the slowest DW\_apb\_uart clock should be allowed to pass before transmitting or receiving data.

Bits	Name	Memory Access	Description
31:8	RSVD_DLH_31to8	R	Reserved bits [31:8] - Read Only
7:0	dlh	R/W	Divisor Latch High 8-bit register field - used to set the UART baud-rate

#### 11.4.2.5 IER

**Size:** 32 bits

**Offset:** 0x4

**Memory Access:** R/W

**Value After Reset:** 0x0

31:8	7	6:4	3	2	1	0
RSVD_IER_31to8	PTIME	RSVD_IER_6to4	EDSSI	ELSI	ETBEI	ERBFI

Interrupt Enable Register: Interrupt Enable Register, when the DLAB bit is zero; Divisor Latch (High), when the DLAB bit is one. Each of the bits used has a different function and will be detailed in the bit field descriptions.

Bits	Name	Memory Access	Description
31:8	RSVD_IER_31to8	R	Reserved bits [31:8] - Read Only
7	PTIME	R	Interrupt Enable Register: PTIME, Programmable THRE Interrupt Mode Enable. Writeable only when <code>THRE_MODE_USER == Enabled</code> , always readable. This is used to enable/disable the generation of THRE Interrupt. 0 = disabled 1 = enabled
6:4	RSVD_IER_6to4	R	Reserved bits [6:4] - Read Only
3	EDSSI	R/W	Interrupt Enable Register: EDSSI, Enable Modem Status Interrupt. This is used to enable/disable the generation of Modem Status Interrupt. This is the fourth highest priority interrupt. 0 = disabled 1 = enabled
2	ELSI	R/W	Interrupt Enable Register: ELSI, Enable Receiver Line Status Interrupt. This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt. 0 = disabled 1 = enabled
1	ETBEI	R/W	Interrupt Enable Register: ETBEI, Enable Transmit Holding Register Empty Interrupt. This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt. This is the third highest priority interrupt. 0 = disabled 1 = enabled
0	ERBFI	R/W	Interrupt Enable Register: ERBFI, Enable Received Data Available Interrupt. This is used to enable/disable the generation of Received Data Available Interrupt and the Character Timeout Interrupt (if in FIFO mode and FIFO's enabled). These are the second highest priority interrupts. 0 = disabled 1 = enabled

#### 11.4.2.6 FCR

**Size:** 32 bits

**Offset:** 0x8

**Memory Access:** R/W

**Value After Reset:** 0x0

31:8	7:6	5:4	3	2	1	0
RSVD_FCR_31to8	RT	TET	DMAM	XFIFOR	RFIFOR	FIFOE

FIFO Control Register. This register is only valid when the DW\_apb\_uart is configured to have FIFO's implemented (FIFO\_MODE != NONE). If FIFO's are not implemented, this register does not exist and writing to this register address will have no effect.

Bits	Name	Memory Access	Description
31:8	RSVD_FCR_31to8	R	Reserved bits [31:8] - Read Only
7:6	RT	W	Bits[7:6], RCVR Trigger (or RT): This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt will be generated. In auto flow control mode it is used to determine when the rts_n signal will be de-asserted. It also determines when the dma_rx_req_n signal will be asserted when in certain modes of operation. See section 5.9 on page 56 for details on DMA support. The following trigger levels are supported: 00 = 1 character in the FIFO 01 = FIFO 1/4 full 10 = FIFO 1/2 full 11 = FIFO 2 less than full
5:4	TET	R	Bits[5:4], TX Empty Trigger (or TET): Writes will have no effect when THRE_MODE_USER == Disabled. This is used to select the empty threshold level at which the THRE Interrupts will be generated when the mode is active. It also determines when the dma_tx_req_n signal will be asserted when in certain modes of operation. See section 5.9 on page 56 for details on DMA support. The following trigger levels are supported: 00 = FIFO empty 01 = 2 characters in the FIFO 10 = FIFO 1/4 full 11 = FIFO full
3	DMAM	W	Bit[3], DMA Mode (or DMAM): This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals when additional DMA handshaking signals are not selected (DMA_EXTRA == NO). See section 5.9 on page 56 for details on DMA support. 0 = mode 0 1 = mode 1
2	XFIFOR	W	Bit[2], XMIT FIFO Reset (or XFIFOR): This resets the control portion of the transmit FIFO and treats the FIFO as empty. This will also de-assert the DMA TX request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA == YES). Note that this bit is 'self-clearing' and it is not necessary to clear this bit.
1	RFIFOR	W	Bit[1], RCVR FIFO Reset (or RFIFOR): This resets the control portion of the receive FIFO and treats the FIFO as empty. This will also de-assert the DMA RX request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA == YES). Note that this bit is 'self-clearing' and it is not necessary to clear this bit.
0	FIFOE	W	Bit[0], FIFO Enable (or FIFOE): This enables/disables the transmit (XMIT) and receive (RCVR) FIFO's. Whenever the value of this bit is changed both the XMIT and RCVR controller portion of FIFO's will be reset.

**11.4.2.7 IIR****Size:** 32 bits**Offset:** 0x8**Memory Access:** R**Value After Reset:** 0x1

31:8	7:6	5:4	3:0
RSVD_IIR_31to8	FIFOE	RSVD_IIR_5to4	IID

Interrupt Identification Register

Bits	Name	Memory Access	Description
31:8	RSVD_IIR_31to8	R	Reserved bits [31:8] - Read Only
7:6	FIFOSE	R	Bits[7:6], FIFO's Enabled (or FIFOSE): This is used to indicate whether the FIFO's are enabled or disabled. 00 = disabled. 11 = enabled
5:4	RSVD_IIR_5to4	R	Reserved bits [5:4] - Read Only
3:0	IID	R	Bits[3:0], Interrupt ID (or IID): This indicates the highest priority pending interrupt which can be one of the following types: 0000 = modem status. 0001 = no interrupt pending. 0010 = THR empty. 0100 = received data available. 0110 = receiver line status. 0111 = busy detect. 1100 = character timeout. Note, an interrupt of type 0111 (busy detect) will never get indicated if UART_16550_COMPATIBLE == YES in coreConsultant.

#### 11.4.2.8 LCR

**Size:** 32 bits

**Offset:** 0xc

**Memory Access:** R/W

**Value After Reset:** 0x0

31:8	7	6	5	4	3	2	1:0
RSVD_LCR_31to8	DLAB	Break	SP	EPS	PEN	STOP	DLS

Line Control Register

Bits	Name	Memory Access	Description
31:8	RSVD_LCR_31to8	R	Reserved bits [31:8] - Read Only
7	DLAB	R/W	Divisor Latch Access Bit. If UART_16550_COMPATIBLE == NO then, writeable only when UART is not busy (USR[0] is zero), otherwise always writable, always readable. This bit is used to enable reading and writing of the Divisor Latch register (DLL and DLH) to set the baud rate of the UART. This bit must be cleared after initial baud rate setup in order to access other registers.
6	Break	R/W	Break Control Bit. This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared. If SIR_MODE == Enabled and active (MCR[6] set to one) the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver and the sir_out_n line is forced low.
5	SP	R/W	From DW_apb_uart_regfile.sv: // aaraujo @ 17/05/2011 : CRM_9000431453 // Stick parity lcr_ir[5] is now programmable lcr_ir[5:0] <= ipwdata[5:0];
4	EPS	R/W	Even Parity Select. If UART_16550_COMPATIBLE == NO then, writeable only when UART is not busy (USR[0] is zero), otherwise always writable, always readable. This is used to select between even and odd parity, when parity is enabled (PEN set to one). If set to one, an even number of logic '1's is transmitted or checked. If set to zero, an odd number of logic '1's is transmitted or checked.
3	PEN	R/W	Parity Enable. If UART_16550_COMPATIBLE == NO then, writeable only when UART is not busy (USR[0] is zero), otherwise always writable, always readable. This bit is

Bits	Name	Memory Access	Description
			used to enable and disable parity generation and detection in transmitted and received serial character respectively. 0 = parity disabled 1 = parity enabled
2	STOP	R/W	Number of stop bits. If UART_16550_COMPATIBLE == NO then, writeable only when UART is not busy (USR[0] is zero), otherwise always writable, always readable. This is used to select the number of stop bits per character that the peripheral will transmit and receive. If set to zero, one stop bit is transmitted in the serial data. If set to one and the data bits are set to 5 (LCR[1:0] set to zero) one and a half stop bits is transmitted. Otherwise, two stop bits are transmitted. Note that regardless of the number of stop bits selected the receiver will only check the first stop bit. 0 = 1 stop bit 1 = 1.5 stop bits when DLS (LCR[1:0]) is zero, else 2 stop bit
1:0	DLS	R/W	Data Length Select. If UART_16550_COMPATIBLE == NO then, writeable only when UART is not busy (USR[0] is zero), otherwise always writable, always readable. This is used to select the number of data bits per character that the peripheral will transmit and receive. The number of bit that may be selected areas follows: 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits

**11.4.2.9 MCR****Size:** 32 bits**Offset:** 0x10**Memory Access:** R/W**Value After Reset:** 0x0

31:7	6	5	4	3	2	1	0
RSVD_MCR_31to7	SIRE	AFCE	LoopBack	OUT2	OUT1	RTS	DTR

Modem Control Register

Bits	Name	Memory Access	Description
31:7	RSVD_MCR_31to7	R	Reserved bits [31:7] - Read Only
6	SIRE	R	SIR Mode Enable. Writeable only when SIR_MODE == Enabled, always readable. This is used to enable/ disable the IrDA SIR Mode features as described in section 5.2 on page 47. 0 = IrDA SIR Mode disabled 1 = IrDA SIR Mode enabled
5	AFCE	R	Auto Flow Control Enable. Writeable only when AFCE_MODE == Enabled, always readable. When FIFOs are enabled and the Auto Flow Control Enable (AFCE) bit is set, Auto Flow Control features are enabled as described in section 5.6 on page 51. 0 = Auto Flow Control Mode disabled 1 = Auto Flow Control Mode enabled
4	LoopBack	R/W	LoopBack Bit. This is used to put the UART into a dDW_agnostic mode for test purposes. If operating in UART mode (SIR_MODE != Enabled OR NOT active, MCR[6] set to zero), data on the sout line is held high, while serial data output is looped back to the sin line, internally. In this mode all the interrupts are fully functional. Also, in loopback mode, the modem control inputs (dsr_n, cts_n, ri_n, dcd_n) are disconnected and the modem control outputs (dtr_n, rts_n, out1_n, out2_n) are looped back to the inputs, internally. If operating in infrared mode (SIR_MODE == Enabled AND active, MCR[6] set to one), data on the sir_out_n line is held low, while serial data output is inverted and looped back to the sir_in line.
3	OUT2	R/W	OUT2. This is used to directly control the user-designated Output2 (out2_n) output. The value written to this location is inverted and driven out on out2_n, that is: 0 =

Bits	Name	Memory Access	Description
			out2_n de-asserted (logic 1) 1 = out2_n asserted (logic 0) Note that in Loopback mode (MCR[4] set to one), the out2_n output is held inactive high while the value of this location is internally looped back to an input.
2	OUT1	R/W	OUT1. This is used to directly control the user-designated Output1 (out1_n) output. The value written to this location is inverted and driven out on out1_n, that is: 0 = out1_n de-asserted (logic 1) 1 = out1_n asserted (logic 0) Note that in Loopback mode (MCR[4] set to one), the out1_n output is held inactive high while the value of this location is internally looped back to an input.
1	RTS	R/W	Request to Send. This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the UART is ready to exchange data. When Auto RTS Flow Control is not enabled (MCR[5] set to zero), the rts_n signal is set low by programming MCR[1] (RTS) to a high. In Auto Flow Control, AFCE_MODE == Enabled and active (MCR[5] set to one) and FIFO's enable (FCR[0] set to one), the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold). The rts_n signal will be de-asserted when MCR[1] is set low. Note that in Loopback mode (MCR[4] set to one), the rts_n output is held inactive high while the value of this location is internally looped back to an input.
0	DTR	R/W	Data Terminal Ready. This is used to directly control the Data Terminal Ready (dtr_n) output. The value written to this location is inverted and driven out on dtr_n, that is: 0 = dtr_n de-asserted (logic 1) 1 = dtr_n asserted (logic 0) The Data Terminal Ready output is used to inform the modem or data set that the UART is ready to establish communications. Note that in Loopback mode (MCR[4] set to one), the dtr_n output is held inactive high while the value of this location is internally looped back to an input.

**11.4.2.10 LSR****Size:** 32 bits**Offset:** 0x14**Memory Access:** R**Value After Reset:** 0x60

31:8	7	6	5	4	3	2	1	0
RSVD_LSR_31to8	RFE	TEMPT	THRE	BI	FE	PE	OE	DR

Line Status Register

Bits	Name	Memory Access	Description
31:8	RSVD_LSR_31to8	R	Reserved bits [31:8] - Read Only
7	RFE	R	Receiver FIFO Error bit. This bit is only relevant when FIFO_MODE != NONE AND FIFO's are enabled (FCR[0] set to one). This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO. That is: 0 = no error in RX FIFO 1 = error in RX FIFO This bit is cleared when the LSR is read and the character with the error is at the top of the receiver FIFO and there are no subsequent errors in the FIFO.
6	TEMPT	R	Transmitter Empty bit. If in FIFO mode (FIFO_MODE != NONE) and FIFO's enabled (FCR[0] set to one), this bit is set whenever the Transmitter Shift Register and the

Bits	Name	Memory Access	Description
			FIFO are both empty. If in the non-FIFO mode or FIFO's are disabled, this bit is set whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty.
5	THRE	R	Transmit Holding Register Empty bit. If THRE_MODE_USER == Disabled or THRE mode is disabled (IER[7] set to zero) and regardless of FIFO's being implemented/enabled or not, this bit indicates that the THR or TX FIFO is empty. This bit is set whenever data is transferred from the THR or TX FIFO to the transmitter shift register and no new data has been written to the THR or TX FIFO. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. If THRE_MODE_USER == Enabled AND FIFO_MODE != NONE and both modes are active (IER[7] set to one and FCR[0] set to one respectively), the functionality is switched to indicate the transmitter FIFO is full, and no longer controls THRE interrupts, which are then controlled by the FCR[5:4] threshold setting. Programmable THRE interrupt mode operation is described in detail in section 5.7 on page 52.
4	BI	R	Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data. If in UART mode it is set whenever the serial input, sin, is held in a logic '0' state for longer than the sum of start time + data bits + parity + stop bits. If in infrared mode it is set whenever the serial input, sir_in, is continuously pulsed to logic '0' for longer than the sum of start time + data bits + parity + stop bits. A break condition on serial input causes one and only one character, consisting of all zeros, to be received by the UART. In the FIFO mode, the character associated with the break condition is carried through the FIFO and is revealed when the character is at the top of the FIFO. Reading the LSR clears the BI bit. In the non-FIFO mode, the BI indication occurs immediately and persists until the LSR is read.
3	FE	R	Framing Error bit. This is used to indicate the occurrence of a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data. In the FIFO mode, since the framing error is associated with a character received, it is revealed when the character with the framing error is at the top of the FIFO. When a framing error occurs the UART will try resynchronize. It does this by assuming that the error was due to the start bit of the next character and then continues receiving the other bit i.e. data, and/or parity and stop. It should be noted that the Framing Error (FE) bit (LSR[3]) will be set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]). 0 = no framing error 1 = framing error Reading the LSR clears the FE bit.
2	PE	R	Parity Error bit. This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PEN) bit (LCR[3]) is set. In the FIFO mode, since the parity error is associated with a character received, it is revealed when the character with the parity error arrives at the top of the FIFO. It should be noted that the Parity Error (PE) bit (LSR[2]) will be set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]). 0 = no parity error 1 = parity error Reading the LSR clears the PE bit.
1	OE	R	Overrun error bit. This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read. In the non-FIFO mode, the OE bit is set when a new character arrives in the receiver before the previous character was read from the RBR. When this happens, the data in the RBR is overwritten. In the FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver. The data in the FIFO is retained and the data in the receive shift register is lost. 0 = no overrun error 1 = overrun error

Bits	Name	Memory Access	Description
			Reading the LSR clears the OE bit.
0	DR	R	Data Ready bit. This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO. 0 = no data ready 1 = data ready This bit is cleared when the RBR is read in the non-FIFO mode, or when the receiver FIFO is empty, in the FIFO mode.

**11.4.2.11 MSR****Size:** 32 bits**Offset:** 0x18**Memory Access:** R**Value After Reset:** 0x0

31:8	7	6	5	4	3	2	1	0
RSVD_MSR_31to8	DCD	RI	DSR	CTS	DDCD	TERI	DDSR	DCTS

Modem Status Register It should be noted that whenever bits 0, 1, 2 or 3 is set to logic one, to indicate a change on the modem control inputs, a modem status interrupt will be generated if enabled via the IER regardless of when the change occurred. Since the delta bits (bits 0, 1, 3) can get set after a reset if their respective modem signals are active (see individual bits for details), a read of the MSR after reset can be performed to prevent unwanted interrupts.

Bits	Name	Memory Access	Description
31:8	RSVD_MSR_31to8	R	Reserved bits [31:8] - Read Only
7	DCD	R	Data Carrier Detect. This is used to indicate the current state of the modem control line dcd_n. That is this bit is the complement dcd_n. When the Data Carrier Detect input (dcd_n) is asserted it is an indication that the carrier has been detected by the modem or data set. 0 = dcd_n input is de-asserted (logic 1) 1 = dcd_n input is asserted (logic 0) In Loopback Mode (MCR[4] set to one), DCD is the same as MCR[3] (Out2).
6	RI	R	Ring Indicator. This is used to indicate the current state of the modem control line ri_n. That is this bit is the complement ri_n. When the Ring Indicator input (ri_n) is asserted it is an indication that a telephone ringing signal has been received by the modem or data set. 0 = ri_n input is de-asserted (logic 1) 1 = ri_n input is asserted (logic 0) In Loopback Mode (MCR[4] set to one), RI is the same as MCR[2] (Out1).
5	DSR	R	Data Set Ready. This is used to indicate the current state of the modem control line dsr_n. That is this bit is the complement dsr_n. When the Data Set Ready input (dsr_n) is asserted it is an indication that the modem or data set is ready to establish communications with the DW_apb_uart. 0 = dsr_n input is de-asserted (logic 1) 1 = dsr_n input is asserted (logic 0) In Loopback Mode (MCR[4] set to one), DSR is the same as MCR[0] (DTR).
4	CTS	R	Clear to Send. This is used to indicate the current state of the modem control line cts_n. That is, this bit is the complement cts_n. When the Clear to Send input (cts_n) is asserted it is an indication that the modem or data set is ready to exchange data with the DW_apb_uart. 0 = cts_n input is de-asserted (logic 1) 1 = cts_n input is asserted (logic 0) In Loopback Mode (MCR[4] set to one), CTS is the same as MCR[1] (RTS).
3	DDCD	R	Delta Data Carrier Detect. This is used to indicate that the modem control line dcd_n has changed since the last time the MSR was read. That is: 0 = no change on

Bits	Name	Memory Access	Description
			dcd_n since last read of MSR 1 = change on dcd_n since last read of MSR Reading the MSR clears the DDCD bit. In Loopback Mode (MCR[4] set to one), DDCD reflects changes on MCR[3] (Out2). Note, if the DDCD bit is not set and the dcd_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDCD bit will get set when the reset is removed if the dcd_n signal remains asserted.
2	TERI	R	Trailing Edge of Ring Indicator. This is used to indicate that a change on the input ri_n (from an active low, to an inactive high state) has occurred since the last time the MSR was read. That is: 0 = no change on ri_n since last read of MSR 1 = change on ri_n since last read of MSR Reading the MSR clears the TERI bit. In Loopback Mode (MCR[4] set to one), TERI reflects when MCR[2] (Out1) has changed state from a high to a low.
1	DDSR	R	Delta Data Set Ready. This is used to indicate that the modem control line dsr_n has changed since the last time the MSR was read. That is: 0 = no change on dsr_n since last read of MSR 1 = change on dsr_n since last read of MSR Reading the MSR clears the DDSR bit. In Loopback Mode (MCR[4] set to one), DDSR reflects changes on MCR[0] (DTR). Note, if the DDSR bit is not set and the dsr_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDSR bit will get set when the reset is removed if the dsr_n signal remains asserted.
0	DCTS	R	Delta Clear to Send. This is used to indicate that the modem control line cts_n has changed since the last time the MSR was read. That is: 0 = no change on cts_n since last read of MSR 1 = change on cts_n since last read of MSR Reading the MSR clears the DCTS bit. In Loopback Mode (MCR[4] set to one), DCTS reflects changes on MCR[1] (RTS). Note, if the DCTS bit is not set and the cts_n signal is asserted (low) and a reset occurs (software or otherwise), then the DCTS bit will get set when the reset is removed if the cts_n signal remains asserted.

**11.4.2.12 SCR****Size:** 32 bits**Offset:** 0x1c**Memory Access:** R/W**Value After Reset:** 0x0

31:8	7:0
RSVD_SCR_31to8	scr

Scratchpad Register

Bits	Name	Memory Access	Description
31:8	RSVD_SCR_31to8	R	Reserved bits [31:8] - Read Only
7:0	scr	R/W	This register is for programmers to use as a temporary storage space. It has no defined purpose in the DW_apb_uart.

**FAR****Size:** 32 bits**Offset:** 0x70**Memory Access:** R**Value After Reset:** 0x0

31:1	0
------	---

RSVD_FAR_31to1	far
----------------	-----

## FIFO Access Register

Bits	Name	Memory Access	Description
31:1	RSVD_FAR_31to1	R	Reserved bits [31:1] - Read Only
0	far	R	Writes will have no effect when FIFO_ACCESS == No, always readable. This register is used to enable a FIFO access mode for testing, so that the receive FIFO can be written by the master and the transmit FIFO can be read by the master when FIFO's are implemented and enabled. When FIFO's are not implemented or not enabled it allows the RBR to be written by the master and the THR to be read by the master. 0 = FIFO access mode disabled 1 = FIFO access mode enabled Note, that when the FIFO access mode is enabled/disabled, the control portion of the receive FIFO and transmit FIFO is reset and the FIFO's are treated as empty.

## 11.4.2.13 USR

**Size:** 32 bits**Offset:** 0x7c**Memory Access:** R**Value After Reset:** 0x0

31:5	4	3	2	1	0
RSVD_USR_31to5	RSVD_RFF	RSVD_RFNE	RSVD_TFE	RSVD_TFNF	BUSY

UART Status register.

Bits	Name	Memory Access	Description
31:5	RSVD_USR_31to5	R	Reserved bits [31:5] - Read Only
4	RSVD_RFF	R	Receive FIFO Full. This bit is only valid when FIFO_STAT == YES. This is used to indicate that the receive FIFO is completely full. That is: 0 = Receive FIFO not full 1 = Receive FIFO Full This bit is cleared when the RX FIFO is no longer full.
3	RSVD_RFNE	R	Receive FIFO Not Empty. This bit is only valid when FIFO_STAT == YES. This is used to indicate that the receive FIFO contains one or more entries. 0 = Receive FIFO is empty 1 = Receive FIFO is not empty This bit is cleared when the RX FIFO is empty.
2	RSVD_TFE	R	Transmit FIFO Empty. This bit is only valid when FIFO_STAT == YES. This is used to indicate that the transmit FIFO is completely empty. 0 = Transmit FIFO is not empty 1 = Transmit FIFO is empty This bit is cleared when the TX FIFO is no longer empty.
1	RSVD_TFNF	R	Transmit FIFO Not Full. This bit is only valid when FIFO_STAT == YES. This is used to indicate that the transmit FIFO is not full. 0 = Transmit FIFO is full 1 = Transmit FIFO is not full This bit is cleared when the TX FIFO is full.
0	BUSY	R	UART Busy. This bit is only valid when UART_16550_COMPATIBLE == NO. This indicates that a serial transfer is in progress, when cleared indicates that the DW_apb_uart is idle or inactive. 0 = DW_apb_uart is idle or inactive 1 - DW_apb_uart is busy (actively transferring data) Note that it is possible for the UART Busy bit to be cleared even though a new character may have been sent from another device. That is, if the DW_apb_uart has no data in the THR and RBR and there is no transmission in progress and a start bit of a new character has just reached the DW_apb_uart. This is due to the fact that a valid start is not seen until the middle of the bit period and this duration is dependent on the baud divisor that

Bits	Name	Memory Access	Description
			has been programmed. If a second system clock has been implemented (CLOCK_MODE == Enabled) the assertion of this bit will also be delayed by several cycles of the slower clock.

**11.4.2.14 HTX****Size:** 32 bits**Offset:** 0xa4**Memory Access:** R/W**Value After Reset:** 0x0

31:1	0
RSVDHTX_31to1	htx

Halt TX

Bits	Name	Memory Access	Description
31:1	RSVDHTX_31to1	R	Reserved bits [31:1] - Read Only
0	htx	R/W	Halt TX. Writes will have no effect when FIFO_MODE == NONE, always readable. This register is used to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFO's are implemented and enabled. Note, if FIFO's are implemented and not enabled the setting of the halt TX register will have no effect on operation. 0 = Halt TX disabled 1 = Halt TX enabled

**11.4.2.15 DMASA****Size:** 32 bits**Offset:** 0xa8**Memory Access:** R**Value After Reset:** 0x0

31:1	0
RSVD_DMASA_31to1	dmasa

DMA Software Acknowledge

Bits	Name	Memory Access	Description
31:1	RSVD_DMASA_31to1	R	Reserved bits [31:1] - Read Only
0	dmasa	R	DMA Software Acknowledge. Writes will have no effect when DMA_EXTRA == No. This register is used to perform DMA software acknowledge if a transfer needs to be terminated due to an error condition. For example, if the DMA disables the channel, then the DW_apb_uart should clear its request. This will cause the TX request, TX single, RX request and RX single signals to de-assert. Note that this bit is 'self-clearing' and it is not necessary to clear this bit.

## 12 I2C

### 12.1 I2C Overview

The I2C is a programmable control bus that provides support for the communications link between integrated circuits in a system. It is a simple two-wire bus with a software-defined protocol for system control, which is used in temperature sensors and voltage level translators to EEPROMs, general-purpose I/O, A/D and D/A converters, CODECs, and many types of microprocessors.

The following is not supported:

- Fast Mode Plus speed is untested and therefore Fast Mode Plus is unsupported at this time. However, if the dividers are set to the proper values, the Fast Mode Plus speed should work properly.

### 12.2 I2C Function Description

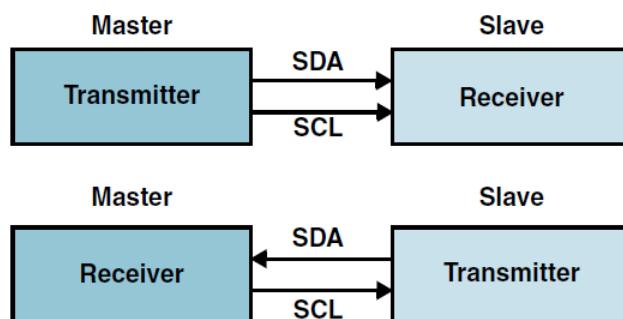
The I2C bus is a two-wire serial interface, consisting of a serial data line (SDA) and a serial clock (SCL). These wires carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a “transmitter” or “receiver,” depending on the function of the device. Devices can also be considered as masters or slaves when performing data transfers. A master is a device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

#### 12.2.1 I2C Bus Terms

The following terms relate to how the role of the I2C device and how it interacts with other I2C devices on the bus.

- **Transmitter** – the device that sends data to the bus. A transmitter can either be a device that initiates the data transmission to the bus (*a master-transmitter*) or responds to a request from the master to send data to the bus (*a slave-transmitter*).
- **Receiver** – the device that receives data from the bus. A receiver can either be a device that receives data on its own request (*a master-receiver*) or in response to a request from the master (*a slave-receiver*).
- **Master** — the component that initializes a transfer (START command), generates the clock (SCL) signal and terminates the transfer (STOP command). A master can be either a transmitter or a receiver.
- **Slave** – the device addressed by the master. A slave can be either receiver or transmitter.
- **Multi-master** – the ability for more than one master to co-exist on the bus at the same time without collision or data loss.
- **Arbitration** – the predefined procedure that authorizes only one master at a time to take control of the bus.
- **Synchronization** – the predefined procedure that synchronizes the clock signals provided by two or more masters.
- **SDA** – data signal line (Serial DAta)
- **SCL** – clock signal line (Serial Clock)

#### Master/Slave and Transmitter/Receiver Relationships



- **START (RESTART)** – data transfer begins with a START or RESTART condition. The level of the SDA data line changes from high to low, while the SCL clock line remains high. When this occurs, the bus becomes busy.
- **STOP** – data transfer is terminated by a STOP condition. This occurs when the level on the SDA data line passes from the low state to the high state, while the SCL clock line remains high. When the data transfer has been terminated, the bus is free.

#### 12.2.2 I2C Behavior

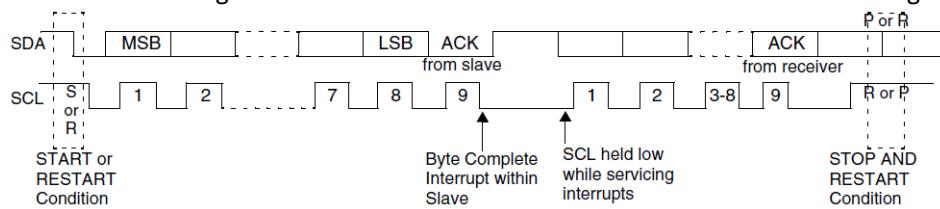
The I2C can be controlled via software to be either:

- An I2C master only, communicating with other I2C slaves; OR

The master is responsible for generating the clock and controlling the transfer of data. The slave is responsible for either transmitting or receiving data to/from the master. The acknowledgement of data is sent by the device that is receiving data, which can be either a master or a slave. As mentioned previously, the I2C protocol also allows multiple masters to reside on the I2C bus and uses an arbitration procedure to determine bus ownership.

Each slave has a unique address that is determined by the system designer. When a master wants to communicate with a slave, the master transmits a START/RESTART condition that is then followed by the slave's address and a control bit (R/W) to determine if the master wants to transmit data or receive data from the slave. The slave then sends an acknowledge (ACK) pulse after the address.

If the master (master-transmitter) is writing to the slave (slave-receiver), the receiver gets one byte of data. This transaction continues until the master terminates the transmission with a STOP condition. If the master is reading from a slave (master-receiver), the slave transmits (slave-transmitter) a byte of data to the master, and the master then acknowledges the transaction with the ACK pulse. This transaction continues until the master terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a RESTART condition. This behavior is illustrated in following figure.



The I2C is a synchronous serial interface. The SDA line is a bidirectional signal and changes only while the SCL line is low, except for STOP, START, and RESTART conditions. The output drivers are open-drain or open-collector to perform wire-AND functions on the bus. The maximum number of devices on the bus is limited by only the maximum capacitance specification of 400 pF. Data is transmitted in byte packages.

### 12.2.2.1 START and STOP Generation

When operating as an I2C master, putting data into the transmit FIFO causes the I2C to generate a START condition on the I2C bus. If the IC\_EMPTYFIFO\_HOLD\_MASTER\_EN parameter is set to 0, allowing the transmit FIFO to empty causes the I2C to generate a STOP condition on the I2C bus. If IC\_EMPTYFIFO\_HOLD\_MASTER\_EN is set to 1, then writing a 1 to IC\_DATA\_CMD[9] causes the I2C to generate a STOP condition on the I2C bus; a STOP condition is not issued if this bit is not set, even if the transmit FIFO is empty.

When operating as a slave, the I2C does not generate START and STOP conditions, as per the protocol. However, if a read request is made to the I2C, it holds the SCL line low until read data has been supplied to it. This stalls the I2C bus until read data is provided to the slave I2C, or the I2C slave is disabled by writing a 0 to bit 0 of the IC\_ENABLE register.

### 12.2.2.2 Combined Formats

The I2C supports mixed read and write combined format transactions in both 7-bit and 10-bit addressing modes.

The I2C does not support mixed address and mixed address format—that is, a 7-bit address transaction followed by a 10-bit address transaction or vice versa—combined format transactions.

To initiate combined format transfers, IC\_CON.IC\_RESTART\_EN should be set to 1. With this value set and operating as a master, when the I2C completes an I2C transfer, it checks the transmit FIFO and executes the next transfer. If the direction of this transfer differs from the previous transfer, the combined format is used to issue the transfer. If the transmit FIFO is empty when the current I2C transfer completes—depending on the value of IC\_EMPTYFIFO\_HOLD\_MASTER\_EN:

- Either a STOP is issued or,
- IC\_DATA\_CMD[9] is checked and:
  - If set to 1, a STOP bit is issued.
  - If set to 0, the SCL is held low until the next command is written to the transmit FIFO.

### 12.2.3 Tx FIFO Management and START, STOP and RESTART Generation

When operating as a master, the I2C component supports two modes of Tx FIFO management. You use the IC\_EMPTYFIFO\_HOLD\_MASTER\_EN parameter to select between these two modes:

- IC\_EMPTYFIFO\_HOLD\_MASTER\_EN equals 0
- IC\_EMPTYFIFO\_HOLD\_MASTER\_EN equals 1

#### 12.2.3.1 Tx FIFO Management

When the value of IC\_EMPTYFIFO\_HOLD\_MASTER\_EN is 0, the component generates a STOP on the bus whenever the Tx FIFO becomes empty. If RESTART generation capability is enabled, the component generates a RESTART when the direction of the transfer in the Tx FIFO commands changes from Read to Write or vice-versa; if RESTART is not enabled, a STOP followed by a START is generated in this situation.

The following figure shows the bits in the IC\_DATA\_CMD register.

IC_DATA_CMD	CMD	DATA	
	8      7		0

DATA –Read/Write field; data retrieved from slave is read from this field; data to be sent to slave is written to this field.

CMD –Write-only field; this bit determines whether transfer to be carried out is Read (CMD=1) or Write (CMD=0)

### 12.2.4 Operation Modes

This section provides information on operation modes.

#### 12.2.4.1 Master Mode Operation

This section discusses master mode procedures.

##### 12.2.4.1.1 Initial Configuration

The initial configuration procedure for Master Mode Operation depends on the configuration parameter I2C\_DYNAMIC\_TAR\_UPDATE. When set to “Yes” (1), the target address and address format can be changed dynamically without having to disable I2C. This parameter only applies to when I2C is acting as a master because the slave requires the component to be disabled before any changes can be made to the address.

The procedures are very similar and are only different with regard to where the IC\_10BITADDR\_MASTER bit is set (either bit 4 of IC\_CON register or bit 12 of IC\_TAR register).

##### 12.2.4.1.2 Master Transmit and Master Receive

The I2C supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the I2C Rx/Tx Data Buffer and Command Register (IC\_DATA\_CMD). The *CMD* bit [8] should be written to 0 for I2C write operations. Subsequently, a read command may be issued by writing “don’t cares” to the lower byte of the IC\_DATA\_CMD register, and a 1 should be written to the *CMD* bit. The I2C master continues to initiate transfers as long as there are commands present in the transmit FIFO. If the transmit FIFO becomes empty—depending on the value of IC\_EMPTYFIFO\_HOLD\_MASTER\_EN, the master either inserts a STOP condition after completing the current transfers, or it checks to see if IC\_DATA\_CMD[9] is set to 1.

- If set to 1, it issues a STOP condition after completing the current transfer.
- If set to 0, it holds SCL low until next command is written to the transmit FIFO.

#### 12.2.4.2 Disabling I2C

The register IC\_ENABLE\_STATUS is added to allow software to unambiguously determine when the hardware has completely shutdown in response to bit 0 of the IC\_ENABLE register being set from 1 to 0. Only one register is required to be monitored, as opposed to monitoring two registers (IC\_STATUS and IC\_RAW\_INTR\_STAT) which is a requirement for I2C versions 1.05a or earlier.

##### 12.2.4.2.1 Procedure

1. Define a timer interval (ti2c\_poll) equal to the 10 times the signaling period for the highest I2C transfer speed used in the system and supported by I2C. For example, if the highest I2C transfer mode is 400 kb/s, then this ti2c\_poll is 25us.

2. Define a maximum time-out parameter, MAX\_T\_POLL\_COUNT, such that if any repeated polling operation exceeds this maximum value, an error is reported.
3. Execute a blocking thread/process/function that prevents any further I2C master transactions to be started by software, but allows any pending transfers to be completed.
4. The variable POLL\_COUNT is initialized to zero.
5. Set bit 0 of the IC\_ENABLE register to 0.
6. Read the IC\_ENABLE\_STATUS register and test the IC\_EN bit (bit 0). Increment POLL\_COUNT by one. If POLL\_COUNT >= MAX\_T\_POLL\_COUNT, exit with the relevant error code.
7. If IC\_ENABLE\_STATUS[0] is 1, then sleep for ti2c\_poll and proceed to the previous step. Otherwise, exit with a relevant success code.

#### 12.2.4.3 Aborting I2C Transfers

The ABORT control bit of the IC\_ENABLE register allows the software to relinquish the I2C bus before completing the issued transfer commands from the Tx FIFO. In response to an ABORT request, the controller issues the STOP condition over the I2C bus, followed by Tx FIFO flush. Aborting the transfer is allowed only in master mode of operation.

##### 12.2.4.3.1 Procedure

1. Stop filling the Tx FIFO (IC\_DATA\_CMD) with new commands.
2. When operating in DMA mode, disable the transmit DMA by setting TDMAE to 0.
3. Set bit 1 of the IC\_ENABLE register (ABORT) to 1.
4. Wait for the M\_TX\_ABRT interrupt.
5. Read the IC\_TX\_ABRT\_SOURCE register to identify the source as ABRT\_USER\_ABRT.

#### 12.2.5 Spike Suppression

The I2C contains programmable spike suppression logic that match requirements imposed by the *I2C Bus Specification* for SS/FS (tSP, Table 4) and HS (tSP, Table 6) modes.

This logic is based on counters that monitor the input signals (SCL and SDA), checking if they remain stable for a predetermined amount of ic\_clk cycles before they are sampled internally. There is one separate counter for each signal (SCL and SDA). The number of ic\_clk cycles can be programmed by the user and should be calculated taking into account the frequency of ic\_clk and the relevant spike length specification.

The *I2C Bus Specification* calls for different maximum spike lengths according to the operating mode—50 ns for SS and FS; 10 ns for HS—so two registers are required to store the values needed for each case:

- Register IC\_FS\_SPKLEN holds the maximum spike length for SS and FS modes
- Register IC\_HS\_SPKLEN holds the maximum spike value for HS mode.

#### 12.2.6 IC\_CLK Frequency Configuration

When the I2C is **configured** as a master, the \*CNT registers must be set before any I2C bus transaction can take place in order to ensure proper I/O timing. The \*CNT registers are:

- IC\_SS\_SCL\_HCNT
- IC\_SS\_SCL\_LCNT
- IC\_FS\_SCL\_HCNT
- IC\_FS\_SCL\_LCNT
- IC\_HS\_SCL\_HCNT
- IC\_HS\_SCL\_LCNT

#### 12.2.7 SDA Hold Time

The I2C protocol specification requires 300ns of hold time on the SDA signal (tHD;DAT) in standard and fast speed modes, and a hold time long enough to bridge the undefined part between logic 1 and logic 0 of the falling edge of SCL in high speed mode.

Board delays on the SCL and SDA signals can mean that the hold-time requirement is met at the I2C master, but not at the I2C slave (or vice-versa). As each application will encounter differing board delays, the I2C contains a software programmable register (IC\_SDA\_HOLD) to enable dynamic adjustment of the SDA hold-time.

---

The IC\_SDA\_HOLD register can be used to alter the timing of the generated SDA (ic\_data\_oe) signal by the I2C. Each value in the IC\_SDA\_HOLD register represents a unit of one ic\_clk period.

When the I2C is operating in Master Mode, the minimum tHD:DAT timing is one ic\_clk period. Therefore even when IC\_SDA\_HOLD has a value of zero, the I2C will drive SDA (ic\_data\_oe) one ic\_clk cycle after driving SCL (ic\_clk\_oe) to logic 0. For all other values of IC\_SDA\_HOLD, the following is true:

- Drive on SDA (ic\_data\_oe) occurs  $IC\_SDA\_HOLD$  ic\_clk cycles after driving SCL (ic\_clk\_oe) to logic 0

When the I2C is operating in Slave Mode, the minimum tHD:DAT timing is  $SPKLEN + 7$  ic\_clk periods, where SPKLEN is:

- IC\_FS\_SPKLEN if the component is operating in SS or FS
- IC\_HS\_SPKLEN if the component is operating in HS

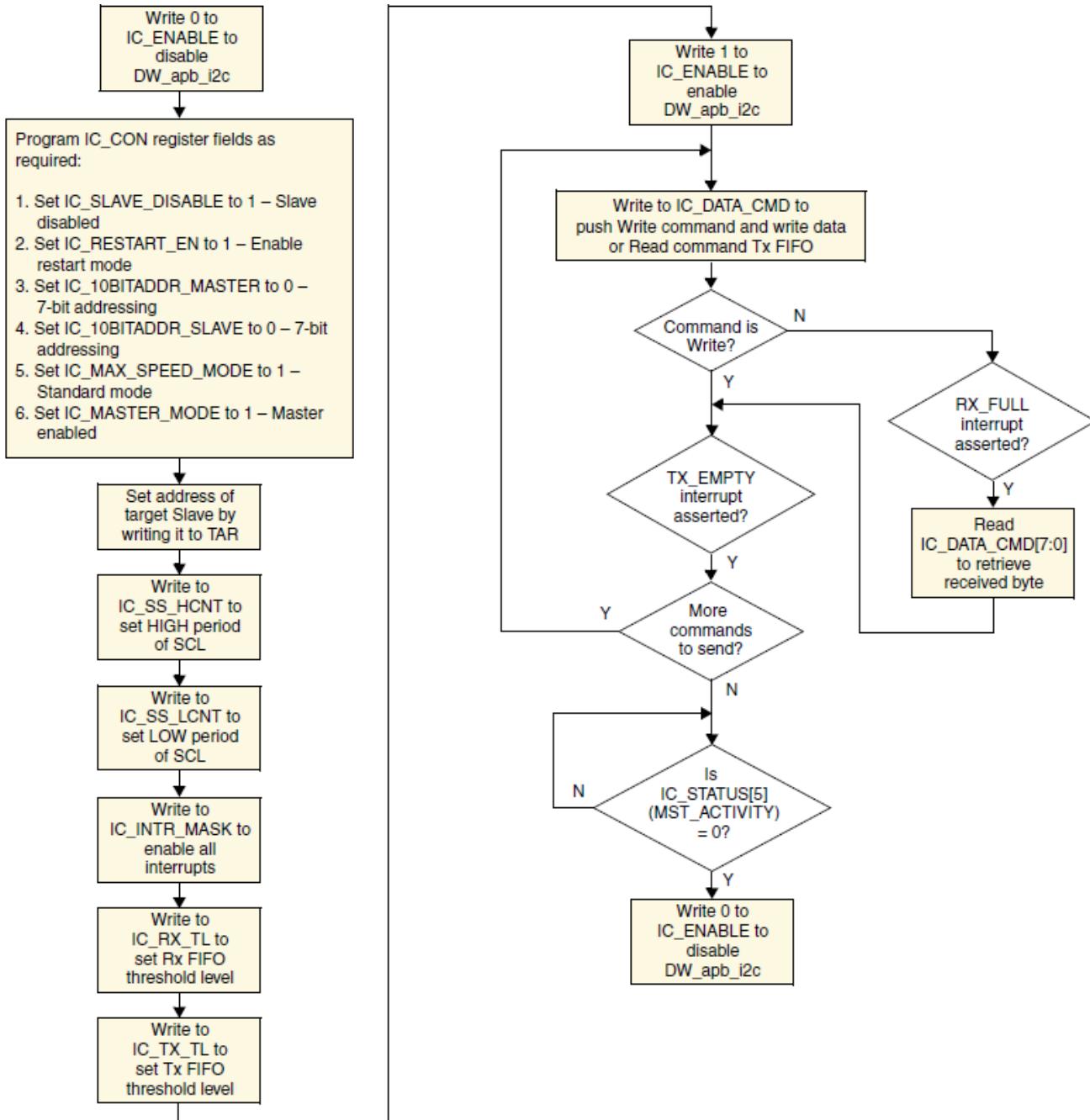
This delay allows for synchronization and spike suppression on the SCL (ic\_clk\_in\_a) sample. Therefore, even when IC\_SDA\_HOLD has a value less than  $SPKLEN + 7$ , the I2C drives SDA (ic\_data\_oe)  $SPKLEN + 7$  ic\_clk cycles after SCL (ic\_clk\_in) has transitioned to logic 0. For all other values of IC\_SDA\_HOLD, the following is true:

- Drive on SDA (ic\_data\_oe) occurs  $IC\_SDA\_HOLD$  ic\_clk cycles after SCL (ic\_clk\_in\_a) has transitioned to logic 0

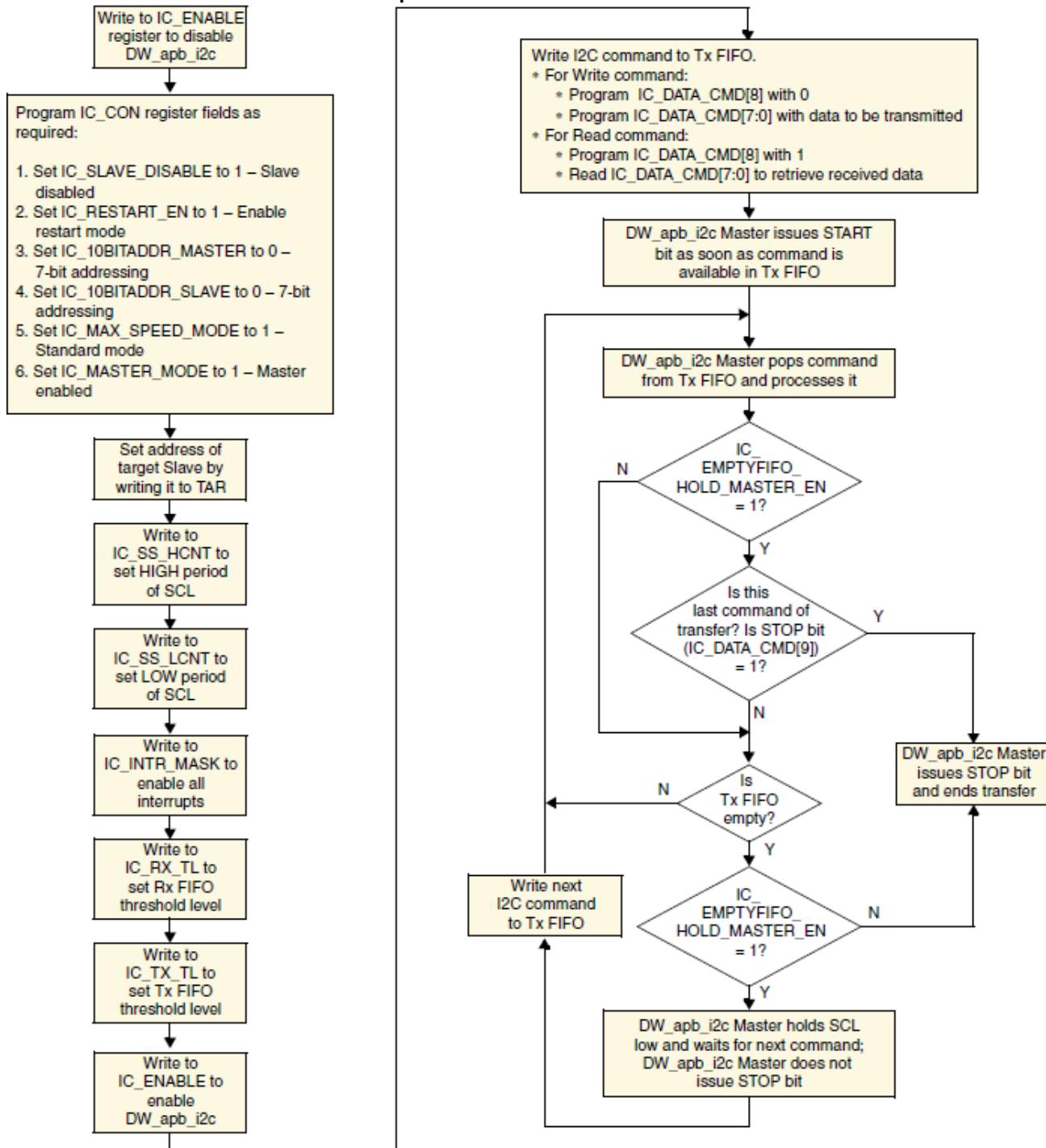
If different SDA hold times are required for different speed modes, the IC\_SDA\_HOLD register must be reprogrammed when the speed mode is being changed. The IC\_SDA\_HOLD register can be programmed only when the I2C is disabled (IC\_ENABLE[0] = 0).

### 12.3 I2C Programming

### 12.3.1 I2C Master Flowchart



### 12.3.2 I2C Master in Standard or Fast Speed Mode Flowchart



### 12.4 I2C Register

This section describes the programmable registers of the I2C.

#### 12.4.1 Register Memory Maps

Register	Offset	Size	Memory Access	Description
DW_apb_i2c address block				
<a href="#">IC_CON</a>	0x0	32	R/W	Value After Reset: 0x7f

Register	Offset	Size	Memory Access	Description
		bits		Name: I2C Control Register Size: 7 bits Address Offset: 0x00 Read/Write Access: If configuration parameter I2C_DYNAMIC_TAR_UPDATE = 0, all bits are Read/Write. If I2C_DYNAMIC_TAR_UPDATE = 1, bit 4 is Read-only. This register can be written only when the DW_apb_i2c is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect.
<a href="#">IC_TAR</a>	0x4	32 bits	R/W	<b>Value After Reset:</b> 0x55 Name: I2C Target Address Register Size: 12 bits or 13 bits; 13 bits only when I2C_DYNAMIC_TAR_UPDATE = 1 Address Offset: 0x04 Read/Write Access: Read/Write If the configuration parameter I2C_DYNAMIC_TAR_UPDATE is set to 'No' (0), this register is 12 bits wide, and bits 15:12 are reserved. This register can be written to only when IC_ENABLE is set to 0. However, if I2C_DYNAMIC_TAR_UPDATE = 1, then the register becomes 13 bits wide. All bits can be dynamically updated as long as any set of the following conditions are true: - DW_apb_i2c is NOT enabled (IC_ENABLE is set to 0); or - DW_apb_i2c is enabled (IC_ENABLE=1); AND DW_apb_i2c is NOT engaged in any Master (tx, rx) operation (IC_STATUS[5]=0); AND DW_apb_i2c is enabled to operate in Master mode (IC_CON[0]=1); AND there are NO entries in the TX FIFO (IC_STATUS[2]=1)
<a href="#">IC_SAR</a>	0x8	32 bits	R/W	<b>Value After Reset:</b> 0x55 Name: I2C Slave Address Register Size: 10 bits Address Offset: 0x08 Read/Write Access: Read/Write
<a href="#">IC_HS_MADDR</a>	0xc	32 bits	R/W	<b>Value After Reset:</b> 0x1 Name: I2C High Speed Master Mode Code Address Register Size: 3 bits Address Offset: 0x0c Read/Write Access: Read/Write
<a href="#">IC_DATA_CMD</a>	0x10	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: I2C Rx/Tx Data Buffer and Command Register; this is the register the CPU writes to when filling the TX FIFO and the CPU reads from when retrieving bytes from RX FIFO Size: 9 bits (writes) 8 bits (reads) Address Offset: 0x10 Read/Write Access: Read/Write NOTE: With nine bits required for writes, the DW_apb_i2c requires 16-bit data on the APB bus transfers when writing into the transmit FIFO. Eight-bit transfers remain for reads from the receive FIFO.
<a href="#">IC_SS_SCL_HCNT</a>	0x14	32 bits	R/W	<b>Value After Reset:</b> 0x190 Name: Standard Speed I2C Clock SCL High Count Register Size: 16 bits Address Offset: 0x14 Read/Write Access: Read/Write
<a href="#">IC_SS_SCL_LCNT</a>	0x18	32 bits	R/W	<b>Value After Reset:</b> 0x1d6 Name: Standard Speed I2C Clock SCL Low Count Register Size: 16 bits Address Offset: 0x18 Read/Write Access: Read/Write
<a href="#">IC_FS_SCL_HCNT</a>	0x1c	32 bits	R/W	<b>Value After Reset:</b> 0x3c Name: Fast Speed I2C Clock SCL High Count Register Size: 16 bits Address Offset: 0x1c Read/Write Access: Read/Write
<a href="#">IC_FS_SCL_LCNT</a>	0x20	32 bits	R/W	<b>Value After Reset:</b> 0x82 Name: Fast Speed I2C Clock SCL Low Count Register Size: 16 bits Address Offset: 0x20 Read/Write Access: Read/Write

Register	Offset	Size	Memory Access	Description
<a href="#">IC_HS_SCL_HCNT</a>	0x24	32 bits	R/W	<b>Value After Reset:</b> 0x6 Name: High Speed I2C Clock SCL High Count Register Size: 16 bits Address Offset: 0x24 Read/Write Access: Read/Write
<a href="#">IC_HS_SCL_LCNT</a>	0x28	32 bits	R/W	<b>Value After Reset:</b> 0x10 Name: High Speed I2C Clock SCL Low Count Register Size: 16 bits Address Offset: 0x28 Read/Write Access: Read/Write
<a href="#">IC_INTR_STAT</a>	0x2c	32 bits	R	<b>Value After Reset:</b> 0x0 Name: I2C Interrupt Status Register Size: 12 bits Address Offset: 0x2c Read/Write Access: Read Each bit in this register has a corresponding mask bit in the IC_INTR_MASK register. These bits are cleared by reading the matching interrupt clear register. The unmasked raw versions of these bits are available in the IC_RAW_INTR_STAT register.
<a href="#">IC_INTR_MASK</a>	0x30	32 bits	R/W	<b>Value After Reset:</b> 0x8ff Name: I2C Interrupt Mask Register Size: 12 bits Address Offset: 0x30 Read/Write Access: Read/Write These bits mask their corresponding interrupt status bits. They are active high; a value of 0 prevents a bit from generating an interrupt.
<a href="#">IC_RAW_INTR_STAT</a>	0x34	32 bits	R	<b>Value After Reset:</b> 0x0 Name: I2C Raw Interrupt Status Register Size: 12 bits Address Offset: 0x34 Read/Write Access: Read Unlike the IC_INTR_STAT register, these bits are not masked so they always show the true status of the DW_apb_i2c.
<a href="#">IC_RX_TL</a>	0x38	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: I2C Receive FIFO Threshold Register Size: 8bits Address Offset: 0x38 Read/Write Access: Read/Write
<a href="#">IC_TX_TL</a>	0x3c	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: I2C Transmit FIFO Threshold Register Size: 8 bits Address Offset: 0x3c Read/Write Access: Read/Write
<a href="#">IC_CLR_INTR</a>	0x40	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Clear Combined and Individual Interrupt Register Size: 1 bit Address Offset: 0x40 Read/Write Access: Read
<a href="#">IC_CLR_RX_UNDER</a>	0x44	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Clear RX_UNDER Interrupt Register Size: 1 bit Address Offset: 0x44 Read/Write Access: Read
<a href="#">IC_CLR_RX_OVER</a>	0x48	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Clear RX_OVER Interrupt Register Size: 1 bit Address Offset: 0x48 Read/Write Access: Read
<a href="#">IC_CLR_TX_OVER</a>	0x4c	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Clear TX_OVER Interrupt Register Size: 1 bit Address Offset: 0x4c Read/Write Access: Read
<a href="#">IC_CLR_RD_REQ</a>	0x50	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Clear RD_REQ Interrupt Register Size: 1 bit Address Offset: 0x50 Read/Write Access: Read
<a href="#">IC_CLR_TX_ABRT</a>	0x54	32	R	<b>Value After Reset:</b> 0x0

Register	Offset	Size	Memory Access	Description
		bits		Name: Clear TX_ABRT Interrupt Register Size: 1 bit Address Offset: 0x54 Read/Write Access: Read
<a href="#"><u>IC_CLR_RX_DONE</u></a>	0x58	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Clear RX_DONE Interrupt Register Size: 1 bit Address Offset: 0x58 Read/Write Access: Read
<a href="#"><u>IC_CLR_ACTIVITY</u></a>	0x5c	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Clear ACTIVITY Interrupt Register Size: 1 bit Address Offset: 0x5c Read/Write Access: Read
<a href="#"><u>IC_CLR_STOP_DET</u></a>	0x60	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Clear STOP_DET Interrupt Register Size: 1 bit Address Offset: 0x60 Read/Write Access: Read
<a href="#"><u>IC_CLR_START_DET</u></a>	0x64	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Clear START_DET Interrupt Register Size: 1 bit Address Offset: 0x64 Read/Write Access: Read
<a href="#"><u>IC_CLR_GEN_CALL</u></a>	0x68	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Clear GEN_CALL Interrupt Register Size: 1 bit Address Offset: 0x68 Read/Write Access: Read
<a href="#"><u>IC_ENABLE</u></a>	0x6c	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: I2C Enable Register Size: 1 bit Address Offset: 0x6c Read/Write Access: Read/Write
<a href="#"><u>IC_STATUS</u></a>	0x70	32 bits	R	<b>Value After Reset:</b> 0x6 Name: I2C Status Register Size: 7 bits Address Offset: 0x70 Read/Write Access: Read This is a read-only register used to indicate the current transfer status and FIFO status. The status register may be read at any time. None of the bits in this register request an interrupt. When the I2C is disabled by writing 0 in bit 0 of the IC_ENABLE register: - Bits 1 and 2 are set to 1 - Bits 3 and 4 are set to 0 When the master or slave state machines goes to idle and ic_en=0: - Bits 5 and 6 are set to 0
<a href="#"><u>IC_TXFLR</u></a>	0x74	32 bits	R	<b>Value After Reset:</b> 0x0 Name: I2C Transmit FIFO Level Register Size: TX_ABW + 1 Address Offset: 0x74 Read/Write Access: Read This register contains the number of valid data entries in the transmit FIFO buffer. It is cleared whenever: - The I2C is disabled - There is a transmit abort that is, TX_ABRT bit is set in the IC_RAW_INTR_STAT register - The slave bulk transmit mode is aborted The register increments whenever data is placed into the transmit FIFO and decrements when data is taken from the transmit FIFO.
<a href="#"><u>IC_RXFLR</u></a>	0x78	32 bits	R	<b>Value After Reset:</b> 0x0 Name: I2C Receive FIFO Level Register Size: RX_ABW + 1 Address Offset: 0x78 Read/Write Access: Read This register contains the number of valid data entries in the receive FIFO buffer. It is cleared whenever: - The I2C is disabled - Whenever there is a transmit abort caused by any of the events tracked in IC_TX_ABRT_SOURCE The register increments whenever data is placed into the receive FIFO and decrements when data is taken from the receive FIFO.

Register	Offset	Size	Memory Access	Description
<a href="#">IC_SDA_HOLD</a>	0x7c	32 bits	R/W	<p><b>Value After Reset:</b> 0x1</p> <p>Name: I2C SDA Hold Register Size: 16 bits Address Offset: 0x7c Read/Write Access: Read/Write This register controls the amount of time delay (in terms of number of ic_clk clock periods) introduced in the falling edge of SCL, relative to SDA changing, when DW_apb_i2c services a read request in a slave-transmitter operation. The relevant I2C requirement is thd:DAT as detailed in the I2C Bus Specification.</p>
<a href="#">IC_TX_ABRT_SOURCE</a>	0x80	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p>Name: I2C Transmit Abort Source Register Size: 16 bits Address Offset: 0x80 Read/Write Access: Read/Write This register has 16 bits that indicate the source of the TX_ABRT bit. Except for Bit 9, this register is cleared whenever the IC_CLR_TX_ABRT register or the IC_CLR_INTR register is read. To clear Bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; RESTART must be enabled (IC_CON[5]=1), the SPECIAL bit must be cleared (IC_TAR[11]), or the GC_OR_START bit must be cleared (IC_TAR[10]). Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, Bit 9 clears for one cycle and is then re-asserted.</p>
<a href="#">IC_SDA_SETUP</a>	0x94	32 bits	R/W	<p><b>Value After Reset:</b> 0x64</p> <p>Name: I2C SDA Setup Register Size: 8 bits Address Offset: 0x94 Read/Write Access: Read/Write This register controls the amount of time delay (in terms of number of ic_clk clock periods) introduced in the rising edge of SCL, relative to SDA changing, when DW_apb_i2c services a read request in a slave-transmitter operation. The relevant I2C requirement is tSU:DAT (note 4) as detailed in the I2C Bus Specification.</p>
<a href="#">IC_ACK_GENERAL_CALL</a>	0x98	32 bits	R/W	<p><b>Value After Reset:</b> 0x1</p> <p>Name: I2C ACK General Call Register Size: 1 bit Address Offset: 0x98 Read/Write Access: Read/Write The register controls whether DW_apb_i2c responds with a ACK or NACK when it receives an I2C General Call address.</p>
<a href="#">IC_ENABLE_STATUS</a>	0x9c	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p>Name: I2C Enable Status Register Size: 3 bits Address Offset: 0x9C Read/Write Access: Read The register is used to report the DW_apb_i2c hardware status when the IC_ENABLE register is set from 1 to 0; that is, when DW_apb_i2c is disabled. If IC_ENABLE has been set to 1, bits 2:1 are forced to 0, and bit 0 is forced to 1. If IC_ENABLE has been set to 0, bits 2:1 is only be valid as soon as bit 0 is read as '0'. Note When IC_ENABLE has been written with '0'a delay occurs for bit 0 to be read as '0' because disabling the DW_apb_i2c depends on I2C bus activities.</p>
<a href="#">IC_FS_SPKLEN</a>	0xa0	32 bits	R/W	<p><b>Value After Reset:</b> 0x5</p> <p>Name: I2C SS and FS spike suppression limit Size: 8 bits Address: 0xA0 Read/Write Access: Read/Write This register is used to store the duration, measured in ic_clk cycles, of the longest spike that is filtered out by the spike suppression logic when the component is operating in SS or FS modes. The relevant I2C requirement is tSP (table 4) as</p>

Register	Offset	Size	Memory Access	Description
				detailed in the I2C Bus Specification. This register must be programmed with a minimum value of 2.
<a href="#">IC_HS_SPKLEN</a>	0xa4	32 bits	R/W	<p><b>Value After Reset:</b> 0x1</p> <p>Name: I2C HS spike suppression limit Size: 8 bits Address: 0xA4 Read/Write Access: Read/Write This register is used to store the duration, measured in ic_clk cycles, of the longest spike that is filtered out by the spike suppression logic when the component is operating in HS modes. The relevant I2C requirement is tSP (table 6) as detailed in the I2C Bus Specification. This register must be programmed with a minimum value of 2.</p>
<a href="#">IC_COMP_PARAM_1</a>	0xf4	32 bits	R	<p><b>Value After Reset:</b> 0x707ae</p> <p>Name: Component Parameter Register 1 Size: 32 bits Address Offset: 0xf4 Read/Write Access: Read Note This is a constant read-only register that contains encoded information about the component's parameter settings. The reset value depends on coreConsultant parameter(s).</p>
<a href="#">IC_COMP_VERSION</a>	0xf8	32 bits	R	<p><b>Value After Reset:</b> 0x3132302a</p> <p>Name: I2C Component Version Register Size: 32 bits Address Offset: 0xf8 Read/Write Access: Read</p>
<a href="#">IC_COMP_TYPE</a>	0xfc	32 bits	R	<p><b>Value After Reset:</b> 0x44570140</p> <p>Name: I2C Component Type Register Size: 32 bits Address Offset: 0xfc Read/Write Access: Read</p>

#### 12.4.2 Registers and Field Descriptions

Following is a description of the individual registers of component DW\_apb\_i2c

##### 12.4.2.1 IC\_CON

**Size:** 32 bits

**Offset:** 0x0

**Memory Access:** R/W

**Value After Reset:** 0x7f

31:7	6	5	4	3	2:1	0
RSVD_IC_CON_31t07	IC_SLAVE_DISABLE	IC_RESTART_EN	IC_10BITADDR_MASTER	IC_10BITADDR_SLAVE	SPEED	MASTER_MODE

Name: I2C Control Register Size: 7 bits Address Offset: 0x00 Read/Write Access: If configuration parameter I2C\_DYNAMIC\_TAR\_UPDATE = 0, all bits are Read/Write. If I2C\_DYNAMIC\_TAR\_UPDATE = 1, bit 4 is Read-only. This register can be written only when the DW\_apb\_i2c is disabled, which corresponds to the IC\_ENABLE register being set to 0. Writes at other times have no effect.

Bits	Name	Memory Access	Description
31:7	RSVD_IC_CON_31to7	R	Reserved bits [31:1] - Read Only
6	IC_SLAVE_DISABLE	R/W	This bit controls whether I2C has its slave disabled, which means once the presetn signal is applied, then this bit takes on the value of the configuration parameter IC_SLAVE_DISABLE. You have the choice of having the slave

Bits	Name	Memory Access	Description
			enabled or disabled after reset is applied, which means software does not have to configure the slave. By default, the slave is always enabled (in reset state as well). If you need to disable it after reset, set this bit to 1. If this bit is set (slave is disabled), DW_apb_i2c functions only as a master and does not perform any action that requires a slave. 0: slave is enabled 1: slave is disabled Reset value: IC_SLAVE_DISABLE configuration parameter NOTE: Software should ensure that if this bit is written with 0, then bit 0 should also be written with a 0.
5	IC_RESTART_EN	R/W	Determines whether RESTART conditions may be sent when acting as a master. Some older slaves do not support handling RESTART conditions; however, RESTART conditions are used in several DW_apb_i2c operations. 0: disable 1: enable When RESTART is disabled, the master is prohibited from performing the following functions: - Change direction within a transfer (split) - Send a START BYTE - High-speed mode operation - Combined format transfers in 7-bit addressing modes - Read operation with a 10-bit address - Send multiple bytes per transfer By replacing RESTART condition followed by a STOP and a subsequent START condition, split operations are broken down into multiple DW_apb_i2c transfers. If the above operations are performed, it will result in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register. Reset value: IC_RESTART_EN configuration parameter
4	IC_10BITADDR_MASTER	R/W	If the I2C_DYNAMIC_TAR_UPDATE configuration parameter is set to 'No' (0), this bit is named IC_10BITADDR_MASTER and controls whether the DW_apb_i2c starts its transfers in 7- or 10-bit addressing mode when acting as a master. If I2C_DYNAMIC_TAR_UPDATE is set to 'Yes' (1), the function of this bit is handled by bit 12 of IC_TAR register, and becomes a read-only copy called IC_10BITADDR_MASTER_rd_only. 0: 7-bit addressing 1: 10-bit addressing Dependencies: If I2C_DYNAMIC_TAR_UPDATE = 1, then this bit is read-only. If I2C_DYNAMIC_TAR_UPDATE = 0, then this bit can be read or write. Reset value: IC_10BITADDR_MASTER configuration parameter
3	IC_10BITADDR_SLAVE	R/W	When acting as a slave, this bit controls whether the DW_apb_i2c responds to 7- or 10-bit addresses. 0: 7-bit addressing. The DW_apb_i2c ignores transactions that involve 10-bit addressing; for 7-bit addressing, only the lower 7 bits of the IC_SAR register are compared. 1: 10-bit addressing. The DW_apb_i2c responds to only 10-bit addressing transfers that match the full 10 bits of the IC_SAR register. Reset value: IC_10BITADDR_SLAVE configuration parameter
2:1	SPEED	R/W	These bits control at which speed the DW_apb_i2c operates; its setting is relevant only if one is operating the DW_apb_i2c in master mode. Hardware protects against illegal values being programmed by software. This register should be programmed only with a value in the range of 1 to IC_MAX_SPEED_MODE; otherwise, hardware updates this register with the value of IC_MAX_SPEED_MODE. 1: standard mode (100 kbit/s) 2: fast mode (400 kbit/s) 3: high speed mode (3.4 Mbit/s) Reset value: IC_MAX_SPEED_MODE configuration
0	MASTER_MODE	R/W	This bit controls whether the DW_apb_i2c master is enabled. 0: master disabled 1: master enabled Reset value: IC_MASTER_MODE configuration parameter NOTE: Software should ensure that if this bit is written with '1' then bit 6 should also be written with a '1'.

**12.4.2.2 IC\_TAR****Size:** 32 bits**Offset:** 0x4**Memory Access:** R/W**Value After Reset:** 0x55

31:12	11	10	9:0
(undef)	SPECIAL	GC_OR_START	IC_TAR

Name: I2C Target Address Register Size: 12 bits or 13 bits; 13 bits only when I2C\_DYNAMIC\_TAR\_UPDATE = 1 Address Offset: 0x04 Read/Write Access: Read/Write If the configuration parameter I2C\_DYNAMIC\_TAR\_UPDATE is set to 'No' (0), this register is 12 bits wide, and bits 15:12 are reserved. This register can be written to only when IC\_ENABLE is set to 0. However, if I2C\_DYNAMIC\_TAR\_UPDATE = 1, then the register becomes 13 bits wide. All bits can be dynamically updated as long as any set of the following conditions are true: - DW\_apb\_i2c is NOT enabled (IC\_ENABLE is set to 0); or - DW\_apb\_i2c is enabled (IC\_ENABLE=1); AND DW\_apb\_i2c is NOT engaged in any Master (tx, rx) operation (IC\_STATUS[5]=0); AND DW\_apb\_i2c is enabled to operate in Master mode (IC\_CON[0]=1); AND there are NO entries in the TX FIFO (IC\_STATUS[2]=1)

Bits	Name	Memory Access	Description
31:12			Reserved for future use.
11	SPECIAL	R/W	This bit indicates whether software performs a General Call or START BYTE command. 0: ignore bit 10 GC_OR_START and use IC_TAR normally 1: perform special I2C command as specified in GC_OR_START bit Reset value: 0x0
10	GC_OR_START	R/W	If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a General Call or START byte command is to be performed by the DW_apb_i2c. 0: General Call Address after issuing a General Call, only writes may be performed. Attempting to issue a read command results in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register. The DW_apb_i2c remains in General Call mode until the SPECIAL bit value (bit 11) is cleared. 1: START BYTE Reset value: 0x0
9:0	IC_TAR	R/W	This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE, the CPU needs to write only once into these bits. Reset value: IC_DEFAULT_TAR_SLAVE_ADDR configuration parameter If the IC_TAR and IC_SAR are the same, loopback exists but the FIFOs are shared between master and slave, so full loopback is not feasible. Only one direction loopback mode is supported (simplex), not duplex. A master cannot transmit to itself; it can transmit to only a slave.

**12.4.2.3 IC\_SAR****Size:** 32 bits**Offset:** 0x8**Memory Access:** R/W**Value After Reset:** 0x55

31:10	9:0
(undef)	IC_SAR

Name: I2C Slave Address Register Size: 10 bits Address Offset: 0x08 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:10			Reserved for future use.

Bits	Name	Memory Access	Description
9:0	IC_SAR	R/W	The IC_SAR holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only IC_SAR[6:0] is used. This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. Note The default values cannot be any of the reserved address locations: that is, 0x00 to 0x07, or 0x78 to 0x7f. The correct operation of the device is not guaranteed if you program the IC_SAR or IC_TAR to a reserved value. Reset value: IC_DEFAULT_SLAVE_ADDR configuration parameter

**12.4.2.4 IC\_HS\_MADDR****Size:** 32 bits**Offset:** 0xc**Memory Access:** R/W**Value After Reset:** 0x1

31:3	2:0
(undef)	IC_HS_MAR

Name: I2C High Speed Master Mode Code Address Register  
**Size:** 3 bits  
**Address Offset:** 0x0c  
**Read/Write Access:** Read/Write

Bits	Name	Memory Access	Description
31:3			Reserved for future use.
2:0	IC_HS_MAR	R/W	This bit field holds the value of the I2C HS mode master code. HS-mode master codes are reserved 8-bit codes (00001xxx) that are not used for slave addressing or other purposes. Each master has its unique master code; up to eight high-speed mode masters can be present on the same I2C bus system. Valid values are from 0 to 7. This register goes away and becomes read-only returning 0s if the IC_MAX_SPEED_MODE configuration parameter is set to either Standard (1) or Fast (2). This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. Reset value: IC_HS_MASTER_CODE configuration parameter

**12.4.2.5 IC\_DATA\_CMD****Size:** 32 bits**Offset:** 0x10**Memory Access:** R/W**Value After Reset:** 0x0

31:9	8	7:0
(undef)	CMD	DAT

Name: I2C Rx/Tx Data Buffer and Command Register; this is the register the CPU writes to when filling the TX FIFO and the CPU reads from when retrieving bytes from RX FIFO  
**Size:** 9 bits (writes) 8 bits (reads)  
**Address Offset:** 0x10  
**Read/Write Access:** Read/Write  
**NOTE:** With nine bits required for writes, the DW\_apb\_i2c requires 16-bit data on the APB bus transfers when writing into the transmit FIFO. Eight-bit transfers remain for reads from the receive FIFO.

Bits	Name	Memory Access	Description
31:9			Reserved for future use.

Bits	Name	Memory Access	Description
8	CMD	W	This bit controls whether a read or a write is performed. This bit does not control the direction when the DW_apb_i2c acts as a slave. It controls only the direction when it acts as a master. 1 = Read 0 = Write When a command is entered in the TX FIFO, this bit distinguishes the write and read commands. In slave-receiver mode, this bit is a 'don't care' because writes to this register are not required. In slave-transmitter mode, a '0' indicates that CPU data is to be transmitted and as DAT or IC_DATA_CMD[7:0]. When programming this bit, you should remember the following: attempting to perform a read operation after a General Call command has been sent results in a TX_ABRT interrupt (bit 6 of the IC_RAW_INTR_STAT register), unless bit 11 (SPECIAL) in the IC_TAR register has been cleared. If a '1' is written to this bit after receiving a RD_REQ interrupt, then a TX_ABRT interrupt occurs. NOTE: It is possible that while attempting a master I2C read transfer on DW_apb_i2c, a RD_REQ interrupt may have occurred simultaneously due to a remote I2C master addressing DW_apb_i2c. In this type of scenario, DW_apb_i2c ignores the IC_DATA_CMD write, generates a TX_ABRT interrupt, and waits to service the RD_REQ interrupt. Reset value: 0x0
7:0	DAT	R/W	This register contains the data to be transmitted or received on the I2C bus. If you are writing to this register and want to perform a read, bits 7:0 (DAT) are ignored by the DW_apb_i2c. However, when you read this register, these bits return the value of data received on the DW_apb_i2c interface. Reset value: 0x0

#### 12.4.2.6 IC\_SS\_SCL\_HCNT

**Size:** 32 bits

**Offset:** 0x14

**Memory Access:** R/W

**Value After Reset:** 0x190

31:16	15:0
(undef)	IC_SS_SCL_HCNT

Name: Standard Speed I2C Clock SCL High Count Register Size: 16 bits Address Offset: 0x14 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	IC_SS_SCL_HCNT	R/W	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for standard speed. The table below shows some sample IC_SS_SCL_HCNT calculations. These values apply only if the ic_clk is set to the given frequency in the table. This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. For designs with APB_DATA_WIDTH = 8, the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed. When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only. NOTE: This register must not be programmed to a value higher than 65525, because DW_apb_i2c uses a 16-bit counter to flag an I2C bus idle condition when this counter reaches a value of IC_SS_SCL_HCNT + 10. Reset value: IC_SS_SCL_HIGH_COUNT configuration parameter

**12.4.2.7 IC\_SS\_SCL\_LCNT****Size:** 32 bits**Offset:** 0x18**Memory Access:** R/W**Value After Reset:** 0x1d6

31:16	15:0
(undef)	IC_SS_SCL_LCNT

Name: Standard Speed I2C Clock SCL Low Count Register Size: 16 bits Address Offset: 0x18 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	IC_SS_SCL_LCNT	R/W	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for standard speed. The table below shows some sample IC_SS_SCL_LCNT calculations. These values apply only if the ic_clk is set to the given frequency in the table. This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted, results in 8 being set. For designs with APB_DATA_WIDTH = 8, the order of programming is important to ensure the correct operation of DW_apb_i2c. The lower byte must be programmed first, and then the upper byte is programmed. When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only. Reset value: IC_SS_SCL_LOW_COUNT configuration parameter

**12.4.2.8 IC\_FS\_SCL\_HCNT****Size:** 32 bits**Offset:** 0x1c**Memory Access:** R/W**Value After Reset:** 0x3c

31:16	15:0
(undef)	IC_FS_SCL_HCNT

Name: Fast Speed I2C Clock SCL High Count Register Size: 16 bits Address Offset: 0x1c Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	IC_FS_SCL_HCNT	R/W	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast speed. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. The table below shows some sample IC_FS_SCL_HCNT calculations. These values apply only if the ic_clk is set to the given frequency in the table. This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE = standard. This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. For designs with APB_DATA_WIDTH == 8 the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed. When the configuration parameter IC_HC_COUNT_VALUES is set to 1,

Bits	Name	Memory Access	Description
			this register is read only. Reset value: IC_FS_SCL_HIGH_COUNT configuration parameter

**12.4.2.9 IC\_FS\_SCL\_LCNT****Size:** 32 bits**Offset:** 0x20**Memory Access:** R/W**Value After Reset:** 0x82

31:16	15:0
(undef)	IC_FS_SCL_LCNT

Name: Fast Speed I2C Clock SCL Low Count Register Size: 16 bits Address Offset: 0x20 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	IC_FS_SCL_LCNT	R/W	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for fast speed. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. The table below shows some sample IC_FS_SCL_LCNT calculations. These values apply only if the ic_clk is set to the given frequency in the table. This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE = standard. This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set. For designs with APB_DATA_WIDTH = 8 the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed. If the value is less than 8 then the count value gets changed to 8. When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only. Reset value: IC_FS_SCL_LOW_COUNT configuration parameter

**12.4.2.10 IC\_HS\_SCL\_HCNT****Size:** 32 bits**Offset:** 0x24**Memory Access:** R/W**Value After Reset:** 0x6

31:16	15:0
(undef)	IC_HS_SCL_HCNT

Name: High Speed I2C Clock SCL High Count Register Size: 16 bits Address Offset: 0x24 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	IC_HS_SCL_HCNT	R/W	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high period count for high speed. The table below shows some sample IC_HS_SCL_HCNT calculations. These values

Bits	Name	Memory Access	Description
			apply only if the ic_clk is set to the given frequency in the table. The SCL High time depends on the loading of the bus. For 100pF loading, the SCL High time is 60ns; for 400pF loading, the SCL High time is 120ns. This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE != high. This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. For designs with APB_DATA_WIDTH = 8 the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed. When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only. Reset value: IC_HS_SCL_HIGH_COUNT configuration parameter

**12.4.2.11 IC\_HS\_SCL\_LCNT****Size:** 32 bits**Offset:** 0x28**Memory Access:** R/W**Value After Reset:** 0x10

31:16	15:0
(undef)	IC_HS_SCL_LCNT

Name: High Speed I2C Clock SCL Low Count Register Size: 16 bits Address Offset: 0x28 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	IC_HS_SCL_LCNT	R/W	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for high speed. The table below shows some sample IC_HS_SCL_LCNT calculations. These values apply only if the ic_clk is set to the given frequency in the table. The SCL low time depends on the loading of the bus. For 100pF loading, the SCL low time is 160ns; for 400pF loading, the SCL low time is 320ns. This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE != high. This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set. For designs with APB_DATA_WIDTH == 8 the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed. If the value is less than 8 then the count value gets changed to 8. When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only. Reset value: IC_HS_SCL_LOW_COUNT configuration parameter

**12.4.2.12 IC\_INTR\_STAT****Size:** 32 bits**Offset:** 0x2c**Memory Access:** R**Value After Reset:** 0x0

31:1 2	11	10	9	8	7	6	5	4	3	2	1	0
(und ef)	R_GEN_ CALL	R_START_ DET	R_STOP_ DET	R_ACTI VITY	R_RX_ DONE	R_TX_ ABRT	R_RD_ REQ	R_TX_E MPTY	R_TX_ OVER	R_RX_ FULL	R_RX_ OVER	R_RX_U NDER

Name: I2C Interrupt Status Register  
Size: 12 bits  
Address Offset: 0x2C  
Read/Write Access: Read  
Each bit in this register has a corresponding mask bit in the IC\_INTR\_MASK register. These bits are cleared by reading the matching interrupt clear register. The unmasked raw versions of these bits are available in the IC\_RAW\_INTR\_STAT register.

Bits	Name	Memory Access	Description
31:12			Reserved for future use.
11	R_GEN_CALL	R	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling DW_apb_i2c or when the CPU reads bit 0 of the IC_CLR_GEN_CALL register. DW_apb_i2c stores the received data in the Rx buffer. Reset value: 0x0
10	R_START_DET	R	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether DW_apb_i2c is operating in slave or master mode. Reset value: 0x0
9	R_STOP_DET	R	Indicates whether a STOP condition has occurred on the I2C interface regardless of whether DW_apb_i2c is operating in slave or master mode. Reset value: 0x0
8	R_ACTIVITY	R	This bit captures DW_apb_i2c activity and stays set until it is cleared. There are four ways to clear it: - Disabling the DW_apb_i2c - Reading the IC_CLR_ACTIVITY register - Reading the IC_CLR_INTR register - System reset Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the DW_apb_i2c module is idle, this bit remains set until cleared, indicating that there was activity on the bus. Reset value: 0x0
7	R_RX_DONE	R	When the DW_apb_i2c is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done. Reset value: 0x0
6	R_TX_ABRT	R	This bit indicates if DW_apb_i2c, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a 'transmit abort'. When this bit is set to 1, the IC_TX_ABRT_SOURCE register indicates the reason why the transmit abort takes place. NOTE: The DW_apb_i2c flushes/resets/empties the TX FIFO whenever this bit is set. The TX FIFO remains in this flushed state until the register IC_CLR_TX_ABRT is read. Once this read is performed, the TX FIFO is then ready to accept more data bytes from the APB interface. Reset value: 0x0
5	R_RD_REQ	R	This bit is set to 1 when DW_apb_i2c is acting as a slave and another I2C master is attempting to read data from DW_apb_i2c. The DW_apb_i2c holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the IC_DATA_CMD register. This bit is set to 0 just after the processor reads the IC_CLR_RD_REQ register. Reset value: 0x0
4	R_TX_EMPTY	R	This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the IC_ENABLE bit 0 is 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1,

Bits	Name	Memory Access	Description
			provided there is activity in the master or slave state machines. When there is no longer activity, then with ic_en=0, this bit is set to 0. Reset value: 0x0
3	R_TX_OVER	R	Set during transmit if the transmit buffer is filled to IC_TX_BUFFER_DEPTH and the processor attempts to issue another I2C command by writing to the IC_DATA_CMD register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. Reset value: 0x0
2	R_RX_FULL	R	Set when the receive buffer reaches or goes above the RX_TL threshold in the IC_RX_TL register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (IC_ENABLE[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once the IC_ENABLE bit 0 is programmed with a 0, regardless of the activity that continues. Reset value: 0x0
1	R_RX_OVER	R	Set if the receive buffer is completely filled to IC_RX_BUFFER_DEPTH and an additional byte is received from an external I2C device. The DW_apb_i2c acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. Reset value: 0x0
0	R_RX_UNDER	R	Set if the processor attempts to read the receive buffer when it is empty by reading from the IC_DATA_CMD register. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. Reset value: 0x0

**12.4.2.13 IC\_INTR\_MASK****Size:** 32 bits**Offset:** 0x30**Memory Access:** R/W**Value After Reset:** 0x8ff

31:12	11	10	9	8	7	6	5	4	3	2	1	0
(un def)	M_GEN_CALL	M_START_DET	M_STOP_DET	M_ACTIVITY	M_RX_DONE	M_TX_ABRT	M_RD_REQ	M_TX_EMPTY	M_TX_OVER	M_RX_FULL	M_RX_OVER	M_RX_UNDER

Name: I2C Interrupt Mask Register Size: 12 bits Address Offset: 0x30 Read/Write Access: Read/Write These bits mask their corresponding interrupt status bits. They are active high; a value of 0 prevents a bit from generating an interrupt.

Bits	Name	Memory Access	Description
31:12			Reserved for future use.
11	M_GEN_CALL	R/W	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling DW_apb_i2c or when the CPU reads bit 0 of the IC_CLR_GEN_CALL register. DW_apb_i2c stores the received data in the Rx buffer. Reset value: 0x1
10	M_START_DET	R/W	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether DW_apb_i2c is operating in slave or master mode. Reset value: 0x0
9	M_STOP_DET	R/W	Indicates whether a STOP condition has occurred on the I2C interface regardless of

Bits	Name	Memory Access	Description
			whether DW_apb_i2c is operating in slave or master mode. Reset value: 0x0
8	M_ACTIVITY	R/W	This bit captures DW_apb_i2c activity and stays set until it is cleared. There are four ways to clear it: - Disabling the DW_apb_i2c - Reading the IC_CLR_ACTIVITY register - Reading the IC_CLR_INTR register - System reset Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the DW_apb_i2c module is idle, this bit remains set until cleared, indicating that there was activity on the bus. Reset value: 0x0
7	M_RX_DONE	R/W	When the DW_apb_i2c is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done. Reset value: 0x1
6	M_TX_ABRT	R/W	This bit indicates if DW_apb_i2c, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a 'transmit abort'. When this bit is set to 1, the IC_TX_ABRT_SOURCE register indicates the reason why the transmit abort takes places. NOTE: The DW_apb_i2c flushes/resets/empties the TX FIFO whenever this bit is set. The TX FIFO remains in this flushed state until the register IC_CLR_TX_ABRT is read. Once this read is performed, the TX FIFO is then ready to accept more data bytes from the APB interface. Reset value: 0x1
5	M_RD_REQ	R/W	This bit is set to 1 when DW_apb_i2c is acting as a slave and another I2C master is attempting to read data from DW_apb_i2c. The DW_apb_i2c holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the IC_DATA_CMD register. This bit is set to 0 just after the processor reads the IC_CLR_RD_REQ register. Reset value: 0x1
4	M_TX_EMPTY	R/W	This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the IC_ENABLE bit 0 is 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer activity, then with ic_en=0, this bit is set to 0. Reset value: 0x1
3	M_TX_OVER	R/W	Set during transmit if the transmit buffer is filled to IC_TX_BUFFER_DEPTH and the processor attempts to issue another I2C command by writing to the IC_DATA_CMD register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. Reset value: 0x1
2	M_RX_FULL	R/W	Set when the receive buffer reaches or goes above the RX_TL threshold in the IC_RX_TL register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (IC_ENABLE[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once the IC_ENABLE bit 0 is programmed with a 0, regardless of the activity that continues. Reset value: 0x1
1	M_RX_OVER	R/W	Set if the receive buffer is completely filled to IC_RX_BUFFER_DEPTH and an additional byte is received from an external I2C device. The DW_apb_i2c acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. Reset

Bits	Name	Memory Access	Description
			value: 0x1
0	M_RX_UNDER	R/W	Set if the processor attempts to read the receive buffer when it is empty by reading from the IC_DATA_CMD register. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. Reset value: 0x1

**12.4.2.14 IC\_RAW\_INTR\_STAT****Size:** 32 bits**Offset:** 0x34**Memory Access:** R**Value After Reset:** 0x0

31:12	11	10	9	8	7	6	5	4	3	2	1	0
(und ef)	GEN_C ALL	START_DET	STOP_D ET	ACTIVI TY	RX_DO NE	TX_AB RT	RD_R EQ	TX_EMP TY	TX_OV ER	RX_FU LL	RX_OV ER	RX_UND ER

Name: I2C Raw Interrupt Status Register Size: 12 bits Address Offset: 0x34 Read/Write Access: Read Unlike the IC\_INTR\_STAT register, these bits are not masked so they always show the true status of the DW\_apb\_i2c.

Bits	Name	Memory Access	Description
31:12			Reserved for future use.
11	GEN_CALL	R	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling DW_apb_i2c or when the CPU reads bit 0 of the IC_CLR_GEN_CALL register. DW_apb_i2c stores the received data in the Rx buffer. Reset value: 0x0
10	START_DET	R	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether DW_apb_i2c is operating in slave or master mode. Reset value: 0x0
9	STOP_DET	R	Indicates whether a STOP condition has occurred on the I2C interface regardless of whether DW_apb_i2c is operating in slave or master mode. Reset value: 0x0
8	ACTIVITY	R	This bit captures DW_apb_i2c activity and stays set until it is cleared. There are four ways to clear it: - Disabling the DW_apb_i2c - Reading the IC_CLR_ACTIVITY register - Reading the IC_CLR_INTR register - System reset Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the DW_apb_i2c module is idle, this bit remains set until cleared, indicating that there was activity on the bus. Reset value: 0x0
7	RX_DONE	R	When the DW_apb_i2c is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done. Reset value: 0x0
6	TX_ABRT	R	This bit indicates if DW_apb_i2c, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a 'transmit abort'. When this bit is set to 1, the IC_TX_ABRT_SOURCE register indicates the reason why the transmit abort takes places. NOTE: The DW_apb_i2c flushes/resets/empties the TX FIFO whenever this bit is set. The TX FIFO remains in this flushed state until the register IC_CLR_TX_ABRT is read. Once this read is performed, the TX FIFO is then ready to accept more data bytes from the APB interface. Reset value: 0x0

Bits	Name	Memory Access	Description
5	RD_REQ	R	This bit is set to 1 when DW_apb_i2c is acting as a slave and another I2C master is attempting to read data from DW_apb_i2c. The DW_apb_i2c holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the IC_DATA_CMD register. This bit is set to 0 just after the processor reads the IC_CLR_RD_REQ register. Reset value: 0x0
4	TX_EMPTY	R	This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_RX_TL register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the IC_ENABLE bit 0 is 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer activity, then with ic_en=0, this bit is set to 0. Reset value: 0x0
3	TX_OVER	R	Set during transmit if the transmit buffer is filled to IC_RX_BUFFER_DEPTH and the processor attempts to issue another I2C command by writing to the IC_DATA_CMD register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. Reset value: 0x0
2	RX_FULL	R	Set when the receive buffer reaches or goes above the RX_TL threshold in the IC_RX_TL register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (IC_ENABLE[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once the IC_ENABLE bit 0 is programmed with a 0, regardless of the activity that continues. Reset value: 0x0
1	RX_OVER	R	Set if the receive buffer is completely filled to IC_RX_BUFFER_DEPTH and an additional byte is received from an external I2C device. The DW_apb_i2c acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. Reset value: 0x0
0	RX_UNDER	R	Set if the processor attempts to read the receive buffer when it is empty by reading from the IC_DATA_CMD register. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. Reset value: 0x0

**12.4.2.15 IC\_RX\_TL****Size:** 32 bits**Offset:** 0x38**Memory Access:** R/W**Value After Reset:** 0x0

31:8	7:0
(undef)	RX_TL

Name: I2C Receive FIFO Threshold Register Size: 8bits Address Offset: 0x38 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:8			Reserved for future use.
7:0	RX_TL	R/W	Receive FIFO Threshold Level Controls the level of entries (or above) that triggers the RX_FULL

Bits	Name	Memory Access	Description
			interrupt (bit 2 in IC_RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 1 entry, and a value of 255 sets the threshold for 256 entries. Reset value: IC_RX_TL configuration parameter

**12.4.2.16 IC\_TX\_TL****Size:** 32 bits**Offset:** 0x3c**Memory Access:** R/W**Value After Reset:** 0x0

31:8	7:0
(undef)	TX_TL

Name: I2C Transmit FIFO Threshold Register Size: 8 bits Address Offset: 0x3c Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:8			Reserved for future use.
7:0	TX_TL	R/W	Transmit FIFO Threshold Level Controls the level of entries (or below) that trigger the TX_EMPTY interrupt (bit 4 in IC_RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that it may not be set to value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 0 entries, and a value of 255 sets the threshold for 255 entries. Reset value: IC_TX_TL configuration parameter

**12.4.2.17 IC\_CLR\_INTR****Size:** 32 bits**Offset:** 0x40**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	CLR_INTR

Name: Clear Combined and Individual Interrupt Register Size: 1 bit Address Offset: 0x40 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	CLR_INTR	R	Read this register to clear the combined interrupt, all individual interrupts, and the IC_TX_ABRT_SOURCE register. This bit does not clear hardware clearable interrupts but software clearable interrupts. Refer to Bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE. Reset value: 0x0

**12.4.2.18 IC\_CLR\_RX\_UNDER****Size:** 32 bits**Offset:** 0x44**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
(undef)	CLR_RX_UNDER

Name: Clear RX\_UNDER Interrupt Register Size: 1 bit Address Offset: 0x44 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	CLR_RX_UNDER	R	Read this register to clear the RX_UNDER interrupt (bit 0) of the IC_RAW_INTR_STAT register. Reset value: 0x0

**12.4.2.19 IC\_CLR\_RX\_OVER****Size:** 32 bits**Offset:** 0x48**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	CLR_RX_OVER

Name: Clear RX\_OVER Interrupt Register Size: 1 bit Address Offset: 0x48 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	CLR_RX_OVER	R	Read this register to clear the RX_OVER interrupt (bit 1) of the IC_RAW_INTR_STAT register. Reset value: 0x0

**12.4.2.20 IC\_CLR\_TX\_OVER****Size:** 32 bits**Offset:** 0x4c**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	CLR_TX_OVER

Name: Clear TX\_OVER Interrupt Register Size: 1 bit Address Offset: 0x4c Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	CLR_TX_OVER	R	Read this register to clear the TX_OVER interrupt (bit 3) of the IC_RAW_INTR_STAT register. Reset value: 0x0

**12.4.2.21 IC\_CLR\_RD\_REQ****Size:** 32 bits**Offset:** 0x50**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	CLR_RD_REQ

Name: Clear RD\_REQ Interrupt Register Size: 1 bit Address Offset: 0x50 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	CLR_RD_REQ	R	Read this register to clear the RD_REQ interrupt (bit 5) of the IC_RAW_INTR_STAT register. Reset value: 0x0

#### 12.4.2.22 IC\_CLR\_TX\_ABRT

**Size:** 32 bits

**Offset:** 0x54

**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
(undef)	CLR_TX_ABRT

Name: Clear TX\_ABRT Interrupt Register Size: 1 bit Address Offset: 0x54 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	CLR_TX_ABRT	R	Read this register to clear the TX_ABRT interrupt (bit 6) of the IC_RAW_INTR_STAT register, and the IC_TX_ABRT_SOURCE register. This also releases the TX FIFO from the flushed/reset state, allowing more writes to the TX FIFO. Refer to Bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE. Reset value: 0x0

#### 12.4.2.23 IC\_CLR\_RX\_DONE

**Size:** 32 bits

**Offset:** 0x58

**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
(undef)	CLR_RX_DONE

Name: Clear RX\_DONE Interrupt Register Size: 1 bit Address Offset: 0x58 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	CLR_RX_DONE	R	Read this register to clear the RX_DONE interrupt (bit 7) of the IC_RAW_INTR_STAT register. Reset value: 0x0

#### 12.4.2.24 IC\_CLR\_ACTIVITY

**Size:** 32 bits

**Offset:** 0x5c

**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
(undef)	CLR_ACTIVITY

Name: Clear ACTIVITY Interrupt Register Size: 1 bit Address Offset: 0x5c Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	CLR_ACTIVITY	R	Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY interrupt bit continues to be set. It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus. The value read from this register to get status of the ACTIVITY interrupt (bit 8) of the IC_RAW_INTR_STAT register. Reset value: 0x0

**12.4.2.25 IC\_CLR\_STOP\_DET****Size:** 32 bits**Offset:** 0x60**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	CLR_STOP_DET

Name: Clear STOP\_DET Interrupt Register Size: 1 bit Address Offset: 0x60 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	CLR_STOP_DET	R	Read this register to clear the STOP_DET interrupt (bit 9) of the IC_RAW_INTR_STAT register. Reset value: 0x0

**12.4.2.26 IC\_CLR\_START\_DET****Size:** 32 bits**Offset:** 0x64**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	CLR_START_DET

Name: Clear START\_DET Interrupt Register Size: 1 bit Address Offset: 0x64 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	CLR_START_DET	R	Read this register to clear the START_DET interrupt (bit 10) of the IC_RAW_INTR_STAT register. Reset value: 0x0

**12.4.2.27 IC\_CLR\_GEN\_CALL****Size:** 32 bits**Offset:** 0x68

**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	CLR_GEN_CALL

Name: Clear GEN\_CALL Interrupt Register Size: 1 bit Address Offset: 0x68 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	CLR_GEN_CALL	R	Read this register to clear the GEN_CALL interrupt (bit 11) of IC_RAW_INTR_STAT register. Reset value: 0x0

**12.4.2.28 IC\_ENABLE****Size:** 32 bits**Offset:** 0x6c**Memory Access:** R/W**Value After Reset:** 0x0

31:1	0
(undef)	ENABLE

Name: I2C Enable Register Size: 1 bit Address Offset: 0x6c Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	ENABLE	R/W	Controls whether the DW_apb_i2c is enabled. 0: Disables DW_apb_i2c (TX and RX FIFOs are held in an erased state) 1: Enables DW_apb_i2c Software can disable DW_apb_i2c while it is active. However, it is important that care be taken to ensure that DW_apb_i2c is disabled properly. When DW_apb_i2c is disabled, the following occurs: - The TX FIFO and RX FIFO get flushed. - Status bits in the IC_INTR_STAT register are still active until DW_apb_i2c goes into IDLE state. If the module is transmitting, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. If the module is receiving, the DW_apb_i2c stops the current transfer at the end of the current byte and does not acknowledge the transfer. In systems with asynchronous pclk and ic_clk when IC_CLK_TYPE parameter set to asynchronous (1), there is a two ic_clk delay when enabling or disabling the DW_apb_i2c. Reset value: 0x0

**12.4.2.29 IC\_STATUS****Size:** 32 bits**Offset:** 0x70**Memory Access:** R**Value After Reset:** 0x6

31:7	6	5	4	3	2	1	0
(undef)	SLV_ACTIVITY	MST_ACTIVITY	RFF	RFNE	TFE	TFNF	ACTIVITY

Name: I2C Status Register Size: 7 bits Address Offset: 0x70 Read/Write Access: Read This is a read-only register used to indicate the current transfer status and FIFO status. The status register may be read at any time. None of the bits in this register request an interrupt. When the I2C is disabled by writing 0 in bit 0 of the IC\_ENABLE register: - Bits 1 and 2 are set to 1 - Bits 3 and 4 are set to 0 When the master or slave state machines goes to idle and ic\_en=0: - Bits 5 and 6 are set to 0

Bits	Name	Memory Access	Description
31:7			Reserved for future use.
6	SLV_ACTIVITY	R	Slave FSM Activity Status. When the Slave Finite State Machine (FSM) is not in the IDLE state, this bit is set. 0: Slave FSM is in IDLE state so the Slave part of DW_apb_i2c is not Active 1: Slave FSM is not in IDLE state so the Slave part of DW_apb_i2c is Active Reset value: 0x0
5	MST_ACTIVITY	R	Master FSM Activity Status. When the Master Finite State Machine (FSM) is not in the IDLE state, this bit is set. 0: Master FSM is in IDLE state so the Master part of DW_apb_i2c is not Active 1: Master FSM is not in IDLE state so the Master part of DW_apb_i2c is Active Note IC_STATUS[0]-that is, ACTIVITY bit-is the OR of SLV_ACTIVITY and MST_ACTIVITY bits. Reset value: 0x0
4	RFF	R	Receive FIFO Completely Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. 0: Receive FIFO is not full 1: Receive FIFO is full Reset value: 0x0
3	RFNE	R	Receive FIFO Not Empty. This bit is set when the receive FIFO contains one or more entries; it is cleared when the receive FIFO is empty. 0: Receive FIFO is empty 1: Receive FIFO is not empty Reset value: 0x0
2	TFE	R	Transmit FIFO Completely Empty. When the transmit FIFO is completely empty, this bit is set. When it contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. 0: Transmit FIFO is not empty 1: Transmit FIFO is empty Reset value: 0x1
1	TFNF	R	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. 0: Transmit FIFO is full 1: Transmit FIFO is not full Reset value: 0x1
0	ACTIVITY	R	I2C Activity Status. Reset value: 0x0

**12.4.2.30 IC\_TXFLR****Size:** 32 bits**Offset:** 0x74**Memory Access:** R**Value After Reset:** 0x0

31:4	3:0
(undef)	TXFLR

Name: I2C Transmit FIFO Level Register Size: TX\_ABW + 1 Address Offset: 0x74 Read/Write Access: Read This register contains the number of valid data entries in the transmit FIFO buffer. It is cleared whenever: - The I2C is disabled - There is a transmit abort that is, TX\_ABRT bit is set in the IC\_RAW\_INTR\_STAT register - The slave bulk transmit mode is aborted The register increments whenever data is placed into the transmit FIFO and decrements when data is taken from the transmit FIFO.

Bits	Name	Memory Access	Description
31:4			Reserved for future use.
3:0	TXFLR	R	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO. Reset value: 0x0

**12.4.2.31 IC\_RXFLR****Size:** 32 bits**Offset:** 0x78**Memory Access:** R**Value After Reset:** 0x0

31:4	3:0
(undef)	RXFLR

Name: I2C Receive FIFO Level Register Size: RX\_ABW + 1 Address Offset: 0x78 Read/Write Access: Read This register contains the number of valid data entries in the receive FIFO buffer. It is cleared whenever: - The I2C is disabled - Whenever there is a transmit abort caused by any of the events tracked in IC\_TX\_ABRT\_SOURCE The register increments whenever data is placed into the receive FIFO and decrements when data is taken from the receive FIFO.

Bits	Name	Memory Access	Description
31:4			Reserved for future use.
3:0	RXFLR	R	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO. Reset value: 0x0

**12.4.2.32 IC\_SDA\_HOLD****Size:** 32 bits**Offset:** 0x7c**Memory Access:** R/W**Value After Reset:** 0x1

31:16	15:0
(undef)	IC_SDA_HOLD

Name: I2C SDA Hold Register Size: 16 bits Address Offset: 0x7c Read/Write Access: Read/Write This register controls the amount of time delay (in terms of number of ic\_clk clock periods) introduced in the falling edge of SCL, relative to SDA changing, when DW\_apb\_i2c services a read request in a slave-transmitter operation. The relevant I2C requirement is thd:DAT as detailed in the I2C Bus Specification.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	IC_SDA_HOLD	R/W	SDA Hold. Default Reset value: 0x1, but can be hardcoded by setting the IC_DEFAULT_SDA_HOLD configuration parameter.

**12.4.2.33 IC\_TX\_ABRT\_SOURCE****Size:** 32 bits**Offset:** 0x80**Memory Access:** R**Value After Reset:** 0x0

31:16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
(u	ABR_T_SL	ABR_T_SL	ABRT_SLF	A_R	ABR_T_M	ABRT_10B	ABRT_SBY	ABR_T_HS	ABRT_SBY	ABR_T_H	ABR_T_G	ABRT_GC	ABRT_TXD	ABRT_10A	ABRT_10A	ABRT_7B

n d e f)	VRD _INT X	V_A RBL OST	LUSH _TXFI FO	B_ LO ST	ASTE R_DI S	RD_N ORST RT	TE_N ORST RT	_NO RSTR T	TE_A CKDE T	S_A CKD ET	CALL _RE AD	ALL_ NOA CK	ATA_ NOA CK	DDR2 _NOA CK	DDR1 _NOA CK	ADDR _NOA CK
-------------------	------------------	-------------------	---------------------	----------------	-------------------	--------------------	--------------------	------------------	-------------------	------------------	-------------------	-------------------	-------------------	--------------------	--------------------	--------------------

Name: I2C Transmit Abort Source Register Size: 16 bits Address Offset: 0x80 Read/Write Access: Read/Write This register has 16 bits that indicate the source of the TX\_ABRT bit. Except for Bit 9, this register is cleared whenever the IC\_CLR\_TX\_ABRT register or the IC\_CLR\_INTR register is read. To clear Bit 9, the source of the ABRT\_SBYTE\_NORSTRT must be fixed first; RESTART must be enabled (IC\_CON[5]=1), the SPECIAL bit must be cleared (IC\_TAR[11]), or the GC\_OR\_START bit must be cleared (IC\_TAR[10]). Once the source of the ABRT\_SBYTE\_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT\_SBYTE\_NORSTRT is not fixed before attempting to clear this bit, Bit 9 clears for one cycle and is then re-asserted.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15	ABRT_SLVRD_INTX	R	1: When the processor side responds to a slave mode request for data to be transmitted to a remote master and user writes a 1 in CMD (bit 8) of IC_DATA_CMD register. Reset value: 0x0 Role of DW_apb_i2c: Slave-Transmitter
14	ABRT_SLV_ARBLOST	R	1: Slave lost the bus while transmitting data to a remote master. IC_TX_ABRT_SOURCE[12] is set at the same time. Note: Even though the slave never 'owns' the bus, something could go wrong on the bus. This is a fail safe check. For instance, during a data transmission at the low-to-high transition of SCL, if what is on the data bus is not what is supposed to be transmitted, then DW_apb_i2c no longer own the bus. Reset value: 0x0 Role of DW_apb_i2c: Slave-Transmitter
13	ABRT_SLVFLUSH_TXFIFO	R	1: Slave has received a read command and some data exists in the TX FIFO so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO. Reset value: 0x0 Role of DW_apb_i2c: Slave-Transmitter
12	ARB_LOST	R	1: Master has lost arbitration, or if IC_TX_ABRT_SOURCE[14] is also set, then the slave transmitter has lost arbitration. Note: I2C can be both master and slave at the same time. Reset value: 0x0 Role of DW_apb_i2c: Master-Transmitter or Slave-Transmitter
11	ABRT_MASTER_DIS	R	1: User tries to initiate a Master operation with the Master mode disabled. Reset value: 0x0 Role of DW_apb_i2c: Master-Transmitter or Master-Receiver
10	ABRT_10B_RD_NORSTRT	R	1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) =0) and the master sends a read command in 10-bit addressing mode. Reset value: 0x0 Role of DW_apb_i2c: Master-Receiver
9	ABRT_SBYTE_NORSTRT	R	To clear Bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; restart must be enabled (IC_CON[5]=1), the SPECIAL bit must be cleared (IC_TAR[11]), or the GC_OR_START bit must be cleared (IC_TAR[10]). Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, bit 9 clears for one cycle and then gets reasserted. 1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) =0) and the user is trying to send a START Byte. Reset value: 0x0 Role of DW_apb_i2c: Master
8	ABRT_HS_NORSTRT	R	1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) =0) and the user

Bits	Name	Memory Access	Description
			is trying to use the master to transfer data in High Speed mode. Reset value: 0x0 Role of DW_apb_i2c: Master-Transmitter or Master-Receiver
7	ABRT_SBYTE_ACKDET	R	1: Master has sent a START Byte and the START Byte was acknowledged (wrong behavior). Reset value: 0x0 Role of DW_apb_i2c: Master
6	ABRT_HS_ACKDET	R	1: Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior). Reset value: 0x0 Role of DW_apb_i2c: Master
5	ABRT_GCALL_READ	R	1: DW_apb_i2c in master mode sent a General Call but the user programmed the byte following the General Call to be a read from the bus (IC_DATA_CMD[9] is set to 1). Reset value: 0x0 Role of DW_apb_i2c: Master-Transmitter
4	ABRT_GCALL_NOACK	R	1: DW_apb_i2c in master mode sent a General Call and no slave on the bus acknowledged the General Call. Reset value: 0x0 Role of DW_apb_i2c: Master-Transmitter
3	ABRT_TXDATA_NOACK	R	1: This is a master-mode only bit. Master has received an acknowledgement for the address, but when it sent data byte(s) following the address, it did not receive an acknowledge from the remote slave(s). Reset value: 0x0 Role of DW_apb_i2c: Master-Transmitter
2	ABRT_10ADDR2_NOACK	R	1: Master is in 10-bit address mode and the second address byte of the 10-bit address was not acknowledged by any slave. Reset value: 0x0 Role of DW_apb_i2c: Master-Transmitter or Master-Receiver
1	ABRT_10ADDR1_NOACK	R	1: Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave. Reset value: 0x0 Role of DW_apb_i2c: Master-Transmitter or Master-Receiver
0	ABRT_7B_ADDR_NOACK	R	1: Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave. Reset value: 0x0 Role of DW_apb_i2c: Master-Transmitter or Master-Receiver

#### 12.4.2.34 IC\_SDA\_SETUP

**Size:** 32 bits

**Offset:** 0x94

**Memory Access:** R/W

**Value After Reset:** 0x64

31:8	7:0
(undef)	SDA_SETUP

Name: I2C SDA Setup Register Size: 8 bits Address Offset: 0x94 Read/Write Access: Read/Write This register controls the amount of time delay (in terms of number of ic\_clk clock periods) introduced in the rising edge of SCL, relative to SDA changing, when DW\_apb\_i2c services a read request in a slave-transmitter operation. The relevant I2C requirement is tSU:DAT (note 4) as detailed in the I2C Bus Specification.

Bits	Name	Memory Access	Description
31:8			Reserved for future use.
7:0	SDA_SETUP	R/W	SDA Setup. It is recommended that if the required delay is 1000ns, then for an ic_clk

Bits	Name	Memory Access	Description
			frequency of 10 MHz, IC_SDA_SETUP should be programmed to a value of 11. Default Reset value: 0x64, but can be hardcoded by setting the IC_DEFAULT_SDA_SETUP configuration parameter.

#### 12.4.2.35 IC\_ACK\_GENERAL\_CALL

**Size:** 32 bits

**Offset:** 0x98

**Memory Access:** R/W

**Value After Reset:** 0x1

31:1	0
RSVD_IC_ACK_GEN_31to1	ACK_GEN_CALL

Name: I2C ACK General Call Register Size: 1 bit Address Offset: 0x98 Read/Write Access: Read/Write The register controls whether DW\_apb\_i2c responds with a ACK or NACK when it receives an I2C General Call address.

Bits	Name	Memory Access	Description
31:1	RSVD_IC_ACK_GEN_31to1	R	Reserved bits [31:1] - Read Only
0	ACK_GEN_CALL	R/W	ACK General Call. When set to 1, DW_apb_i2c responds with a ACK (by asserting ic_data_oe) when it receives a General Call. Otherwise, DW_apb_i2c responds with a NACK (by negating ic_data_oe). Default Reset value: 0x1, but can be hardcoded by setting the IC_DEFAULT_ACK_GENERAL_CALL configuration parameter.

#### 12.4.2.36 IC\_ENABLE\_STATUS

**Size:** 32 bits

**Offset:** 0x9c

**Memory Access:** R

**Value After Reset:** 0x0

31:3	2	1	0
(undef)	SLV_RX_DATA_LOST	SLV_DISABLED_WHILE_BUSY	IC_EN

Name: I2C Enable Status Register Size: 3 bits Address Offset: 0x9C Read/Write Access: Read The register is used to report the DW\_apb\_i2c hardware status when the IC\_ENABLE register is set from 1 to 0; that is, when DW\_apb\_i2c is disabled. If IC\_ENABLE has been set to 1, bits 2:1 are forced to 0, and bit 0 is forced to 1. If IC\_ENABLE has been set to 0, bits 2:1 is only valid as soon as bit 0 is read as '0'. Note When IC\_ENABLE has been written with '0'a delay occurs for bit 0 to be read as '0' because disabling the DW\_apb\_i2c depends on I2C bus activities.

Bits	Name	Memory Access	Description
31:3			Reserved for future use.
2	SLV_RX_DATA_LOST	R	Slave Received Data Lost. This bit indicates if a Slave-Receiver operation has been aborted with at least one data byte received from an I2C transfer due to the setting of IC_ENABLE from 1 to 0. When read as 1, DW_apb_i2c is deemed to have been actively engaged in an aborted I2C transfer (with matching address) and the data phase of the I2C transfer has been entered, even though a data byte has been responded with a NACK. NOTE: If the remote I2C master terminates the transfer with a

Bits	Name	Memory Access	Description
			STOP condition before the DW_apb_i2c has a chance to NACK a transfer, and IC_ENABLE has been set to 0, then this bit is also set to 1. When read as 0, DW_apb_i2c is deemed to have been disabled without being actively involved in the data phase of a Slave-Receiver transfer. NOTE: The CPU can safely read this bit when IC_EN (bit 0) is read as 0. Reset value: 0x0
1	SLV_DISABLED_WHILE_BUSY	R	Slave Disabled While Busy (Transmit, Receive). This bit indicates if a potential or active Slave operation has been aborted due to the setting of the IC_ENABLE register from 1 to 0. This bit is set when the CPU writes a 0 to the IC_ENABLE register while: (a) DW_apb_i2c is receiving the address byte of the Slave-Transmitter operation from a remote master; OR, (b) address and data bytes of the Slave-Receiver operation from a remote master. When read as 1, DW_apb_i2c is deemed to have forced a NACK during any part of an I2C transfer, irrespective of whether the I2C address matches the slave address set in DW_apb_i2c (IC_SAR register) OR if the transfer is completed before IC_ENABLE is set to 0 but has not taken effect. NOTE: If the remote I2C master terminates the transfer with a STOP condition before the DW_apb_i2c has a chance to NACK a transfer, and IC_ENABLE has been set to 0, then this bit will also be set to 1. When read as 0, DW_apb_i2c is deemed to have been disabled when there is master activity, or when the I2C bus is idle. NOTE: The CPU can safely read this bit when IC_EN (bit 0) is read as 0. Reset value: 0x0
0	IC_EN	R	ic_en Status. This bit always reflects the value driven on the output port ic_en. When read as 1, DW_apb_i2c is deemed to be in an enabled state. When read as 0, DW_apb_i2c is deemed completely inactive. NOTE: The CPU can safely read this bit anytime. When this bit is read as 0, the CPU can safely read SLV_RX_DATA_LOST (bit 2) and SLV_DISABLED_WHILE_BUSY (bit 1). Reset value: 0x0

**12.4.2.37 IC\_FS\_SPKLEN****Size:** 32 bits**Offset:** 0xa0**Memory Access:** R/W**Value After Reset:** 0x5

31:8	7:0
(undef)	IC_FS_SPKLEN

Name: I2C SS and FS spike suppression limit  
 Size: 8 bits  
 Address: 0xA0  
 Read/Write Access: Read/Write  
 This register is used to store the duration, measured in ic\_clk cycles, of the longest spike that is filtered out by the spike suppression logic when the component is operating in SS or FS modes. The relevant I2C requirement is tSP (table 4) as detailed in the I2C Bus Specification. This register must be programmed with a minimum value of 2.

Bits	Name	Memory Access	Description
31:8			Reserved for future use.
7:0	IC_FS_SPKLEN	R/W	This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that will be filtered out by the spike suppression logic. This register can be written only when the I2C interface is disabled which corresponds to the

Bits	Name	Memory Access	Description
			IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 2; hardware prevents values less than this being written, and if attempted results in 2 being set. Default Reset value: IC_DEFAULT_FS_SPKLEN configuration parameter.

**12.4.2.38 IC\_HS\_SPKLEN****Size:** 32 bits**Offset:** 0xa4**Memory Access:** R/W**Value After Reset:** 0x1

31:8	7:0
(undef)	IC_HS_SPKLEN

Name: I2C HS spike suppression limit Size: 8 bits Address: 0xA4 Read/Write Access: Read/Write This register is used to store the duration, measured in ic\_clk cycles, of the longest spike that is filtered out by the spike suppression logic when the component is operating in HS modes. The relevant I2C requirement is tSP (table 6) as detailed in the I2C Bus Specification. This register must be programmed with a minimum value of 2.

Bits	Name	Memory Access	Description
31:8			Reserved for future use.
7:0	IC_HS_SPKLEN	R/W	This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that will be filtered out by the spike suppression logic. This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 2; hardware prevents values less than this being written, and if attempted results in 2 being set. Default Reset value: IC_DEFAULT_HS_SPKLEN configuration parameter.

**12.4.2.39 IC\_COMP\_PARAM\_1****Size:** 32 bits**Offset:** 0xf4**Memory Access:** R**Value After Reset:** 0x707ae

31:2 4	23:16	15:8	7	6	5	4	3:2	1:0
(und ef)	TX_BUFFER_ DEPTH	RX_BUFFER_ DEPTH	ADD_ENCODED_P ARAMS	HAS_D MA	INTR _IO	HC_COUNT_V ALUES	MAX_SPEED_ MODE	APB_DATA_ WIDTH

Name: Component Parameter Register 1 Size: 32 bits Address Offset: 0xf4 Read/Write Access: Read Note This is a constant read-only register that contains encoded information about the component's parameter settings. The reset value depends on coreConsultant parameter(s).

Bits	Name	Memory Access	Description
31:24			Reserved for future use.
23:16	TX_BUFFER_DEPTH	R	The value of this register is derived from the IC_TX_BUFFER_DEPTH

Bits	Name	Memory Access	Description
			coreConsultant parameter. 0x00 = Reserved 0x01 = 2 0x02 = 3 to 0xFF = 256
15:8	RX_BUFFER_DEPTH	R	The value of this register is derived from the IC_RX_BUFFER_DEPTH coreConsultant parameter. 0x00: Reserved 0x01: 2 0x02: 3 to 0xFF: 256
7	ADD_ENCODED_PARAMS	R	The value of this register is derived from the IC_ADD_ENCODED_PARAMS coreConsultant parameter. Reading 1 in this bit means that the capability of reading these encoded parameters via software has been included. Otherwise, the entire register is 0 regardless of the setting of any other parameters that are encoded in the bits. 0: False 1: True
6	HAS_DMA	R	The value of this register is derived from the IC_HAS_DMA coreConsultant parameter 0: False 1: True
5	INTR_IO	R	The value of this register is derived from the IC_INTR_IO coreConsultant parameter 0: Individual 1: Combined
4	HC_COUNT_VALUES	R	The value of this register is derived from the IC_HC_COUNT_VALUES coreConsultant parameter 0: False 1: True
3:2	MAX_SPEED_MODE	R	The value of this register is derived from the IC_MAX_SPEED_MODE coreConsultant parameter. 0x0: Reserved 0x1: Standard 0x2: Fast 0x3: High
1:0	APB_DATA_WIDTH	R	The value of this register is derived from the APB_DATA_WIDTH coreConsultant parameter. 0x0: 8 bits 0x1: 16 bits 0x2: 32 bits 0x3: Reserved

#### 12.4.2.40 IC\_COMP\_VERSION

**Size:** 32 bits

**Offset:** 0xf8

**Memory Access:** R

**Value After Reset:** 0x3132302a



Name: I2C Component Version Register Size: 32 bits Address Offset: 0xf8 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:0	IC_COMP_VERSION	R	Specific values for this register are described in the Releases Table in the DW_apb_i2c Release Notes

#### 12.4.2.41 IC\_COMP\_TYPE

**Size:** 32 bits

**Offset:** 0xfc

**Memory Access:** R

**Value After Reset:** 0x44570140



Name: I2C Component Type Register Size: 32 bits Address Offset: 0xfc Read/Write Access: Read

Bits	Name	Memory Access	Description
31:0	IC_COMP_TYPE	R	Designware Component Type number = 0x44_57_01_40. This assigned unique hex value is constant and is derived from the two ASCII letters 'DW' followed by a 16-bit unsigned number.

## 13 Boot SPI

### 13.1 Features

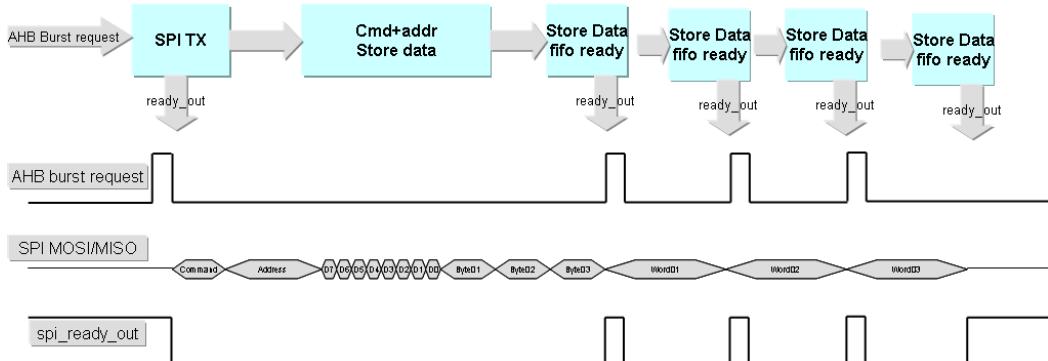
- Compatible with Motorola's SPI specifications
- Supports four signal interface (MOSI, MISO, SCK and SS)
- Supports 8 slave select (SS) bits for each slave on the SPI bus
- Programmable output data rate
- Supports programmable clock latching polarity
- Supports programmable data output polarity
- Supports auto slave select (SS) signal
- Supports MSB/LSB first transactions
- Supports transfer length of 1-bit to 32-bits (programmable data length)
- Supports a programmable write protection bit to do hardware protection
- Direct access can support different size and different burst. (8bits, 16bits, 32bits, 4-beat, 8-beat, 16-beat)

### 13.2 Direct Access Mode

#### 13.2.1 Description

This IP and Serial Flash will work like a memory. It will automatic translate the AHB read command to serial peripheral signal. Therefore, it can be accessed by AHB master directly. And it support burst mode and support different size access.

##### • Ex. 32bits data, 4-beat incrementing burst



#### 13.2.2 Setup Direct Access Mode

1. The input pin "mem\_mode" should be 1.
  2. Setting the baudrate of serial flash clock. It's defined by parameter "CLOCK\_DIV". (This parameter is defined inside spi\_mem.v)
- For example: If the hclk = 500Mhz, and the datasheet of serial flash described the flash clock should below 10Mhz. The parameter should be large than 25.

Parameter CLOCK_DIV	SCLK(serial flash speed) = hclk/??
1	Freq(SLCK)= Freq(hclk)/2
2	Freq(SLCK)= Freq(hclk)/4
..	..
15	Freq(SLCK)= Freq(hclk)/30
16	Freq(SLCK)= Freq(hclk)/32

After these two setting, SPI can be accessed like an AHB slave module. It will automatic translate the AHB read command to serial peripheral signal.

### 13.3 Registers

#### 13.3.1 Registers Memory Mapping

Offset	Type	Parameter	Description	Default value
0x00	R/W	spi_go	Start spi transfer and transfer finish	0x0000_0000
0x04	R/W	ctrl0	Define the transfer data length	0x0000_0000
0x08	R/W	ctrl1	Spi control register1	0x0000_6600
0x0C	R/W	ctrl2	Spi control register2	0x0000_0000
0x10	R	Rx0	Data receive register 0	0x0000_0000
0x14	R	Rx1	Data receive register 1	0x0000_0000
0x18	R	Rx2	Data receive register 2	0x0000_0000
0x1C	R	Rx3	Data receive register 3	0x0000_0000
0x20	R/W	Tx0	Data transmit register 0	0x0000_0000
0x24	R/W	Tx1	Data transmit register 1	0x0000_0000
0x28	R/W	Tx2	Data transmit register 2	0x0000_0000
0x2C	R/W	Tx3	Data transmit register 3	0x0000_0000

#### 13.3.2 Register Description

##### 13.3.2.1 SPI\_GO (0x00)

Bit	Type	Parameter	Description
0	R/W	spi_go	Writing 1 to this bit starts the transfer. This bit remains set during the transfer and is automatically cleared after the transfer finished. Writing 0 to this bit has no effect.

##### 13.3.2.2 Ctrl0 (0x04)

Bit	Type	Parameter	Description
6-0	R/W	char_len	Char_len = 7'b0 => 128bits Char_len = 7'd127 => 127bits ..... Char_len = 7'd2 => 2bits Char_len = 7'd1 => 1bits

##### 13.3.2.3 Ctrl1 (0x08)

Bit	Type	Parameter	Description
14	R/W	wp_pad_o	wp_pad_o = 1'b1 => wp pin is equal to 1. wp_pad_o = 1'b0 => wp pin is equal to 0.
13	R/W	ass	Ass = 1'b1 => ss_pad_o signals are generated automatically. It means the slave select signal is enable by spi_go, and disable automatically after the transition is finished.
11	R/W	lsb	If this bit is set, the LSB is sent first on the line (bit TxL[0]), and the first bit received from the line will be put in the LSB position in the Rx register (bit RxL[0]). If this bit is cleared, the MSB is transmitted/received first (which bit in TxX/RxX register that is depends on the CHAR_LEN field in the CTRL register).
10	R/W	mosi_neg	Send the data when sclk is negative edge.
9	R/W	miso_neg	Latch the data when sclk is negative edge.

##### 13.3.2.4 Ctrl2 (0x0C)

Bit	Type	Parameter	Description
23-16	R/W	spi_ss	Setting the spi_ss register to zero will choose the connection slave devices.

15-0	R/W	sclk_div	16'd0 => sclk = 2 x system clock 16'd1 => sclk = 4 x system clock 16'd2 => sclk = 6 x system clock ..... 16'd65535 => sclk = 131072 x system clock
------	-----	----------	--

**13.3.2.5 RX0~RX3 (0x10~0x1C)**

Bit	Type	Parameter	Description
31-0	R	Rx0~Rx3	The Data which is received from SPI slave. After a read command, the received data will be stored here.

**13.3.2.6 TX0~TX3 (0x20~0x2C)**

Bit	Type	Parameter	Description
31-0	R/W	Tx0~Tx3	The Data which will be sent by SPI master. Before send a command, the command and sending data should be store here first.

An example for read data:

Set 0x2C=0x030000// Command=0x03,addr=0x000000

Set 0x04=0x00000000//128bits

Set 0x00=0x01

----

Wait 0x00=0x00

----

Read 0x18//{Byte0,Byte1,Byte2,Byte3}

Read 0x14//{Byte4,Byte5,Byte6,Byte7}

Read 0x10//{Byte8,Byte9,Byte10,Byte11}

## 14 SSI

### 14.1 Function Description

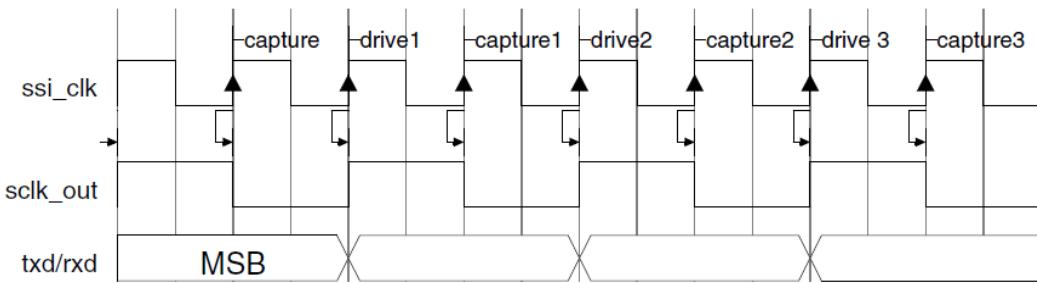
#### 14.1.1 Clock Ratio

The frequency of the DW\_apb\_ssi serial input clock (`ssi_clk`) must be less than or equal to the frequency of `pclk`, which guarantees that control signals from the `ssi_clk` domain are synchronized to the `pclk` domain. When `pclk` and `ssi_clk` are asynchronous, synchronization logic transfers control signals from one clock domain to the other. If both clocks are synchronous, the synchronization logic is unneeded, thus reducing latency in the peripheral.

When DW\_apb\_ssi is a master, the maximum frequency of the bit-rate clock (`sclk_out`) is one-half the frequency of `ssi_clk`. This allows the shift control logic to capture data on one clock edge of `sclk_out` and propagate data on the opposite edge.

Figure 3-2 on page 32 illustrates the maximum ratio between `sclk_out` and `ssi_clk`.

**Maximum `sclk_out/ssi_clk` Ratio**



The `sclk_out` line toggles only when an active transfer is in progress. At all other times it is held in an inactive state, as defined by the serial protocol under which it operates.

The frequency of `sclk_out` can be derived from the following equation:

$$F_{\text{sclk\_out}} = \frac{F_{\text{ssi\_clk}}}{\text{SCKDV}}$$

`SCKDV` is a bit field in the programmable register `BAUDR`, holding any even value in the range 0 to 65,534. If `SCKDV` is 0, then `sclk_out` is disabled.

The DW\_apb\_gpio is a programmable General Purpose Programming I/O (GPIO) peripheral. The component is an AMBA 2.0-compliant Advanced Peripheral Bus (APB) slave device.

Following functional groupings of the main interfaces to the DW\_apb\_gpio block:

#### 14.1.2 Transmit and Receive FIFO Buffers

The FIFO buffers used by the DW\_apb\_ssi are internal D-type flip-flops that can be configured in depth between 2 to 256. The width of both transmit and receive FIFO buffers is fixed at 16 bits due to the serial specifications, which state that a serial transfer (data frame) can be 4 to 16 bits in length. Data frames that are less than 16 bits in size must be right-justified when written into the transmit FIFO buffer. The shift control logic automatically right-justifies receive data in the receive FIFO buffer.

Each data entry in the FIFO buffers contains a single data frame. It is impossible to store multiple data frames in a single FIFO location; for example, you may not store two 8-bit data frames in a single FIFO location. If an 8-bit data frame is required, the upper 8-bits of the FIFO entry are ignored or unused when the serial shifter transmits the data.

The transmit FIFO is loaded by APB write commands to the DW\_apb\_ssi data register (`DR`). Data are popped (removed) from the transmit FIFO by the shift control logic into the transmit shift register. The

transmit FIFO generates a FIFO empty interrupt request (`ssi_txe_intr`) when the number of entries in the FIFO is less than or equal to the FIFO threshold value. The threshold value, set through the programmable register `TXFTLR`, determines the level of FIFO entries at which an interrupt is generated. The threshold value allows you to provide early indication to the processor that the transmit FIFO is nearly empty. A transmit FIFO overflow interrupt (`ssi_txo_intr`) is generated if you attempt to write data into an already full transmit FIFO.

Data are popped from the receive FIFO by APB read commands to the `DW_apb_ssi` data register (`DR`). The receive FIFO is loaded from the receive shift register by the shift control logic. The receive FIFO generates a FIFO-full interrupt request (`ssi_rxf_intr`) when the number of entries in the FIFO is greater than or equal to the FIFO threshold value plus 1. The threshold value, set through programmable register `RXFTLR`, determines the level of FIFO entries at which an interrupt is generated.

The threshold value allows you to provide early indication to the processor that the receive FIFO is nearly full. A receive FIFO overrun interrupt (`ssi_rxo_intr`) is generated when the receive shift logic attempts to load data into a completely full receive FIFO. However, this newly received data are lost. A receive FIFO underflow interrupt (`ssi_rxu_intr`) is generated if you attempt to read from an empty receive FIFO. This alerts the processor that the read data are invalid.

#### 14.1.3 Interrupts

The `DW_apb_ssi` supports combined and individual interrupt requests, each of which can be masked. The combined interrupt request is the ORed result of all other `DW_apb_ssi` interrupts after masking. The system designer has the choice of routing individual interrupt requests or only the combined interrupt request to the Interrupt Controller. All `DW_apb_ssi` interrupts are level interrupts and have the same active polarity level; you can [configure](#) this polarity level as active-high or active-low.

The `DW_apb_ssi` interrupts are described as follows:

- Transmit FIFO Empty Interrupt (`ssi_txe_intr`) – Set when the transmit FIFO is equal to or below its threshold value and requires service to prevent an under-run. The threshold value, set through a software-programmable register, determines the level of transmit FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data are written into the transmit FIFO buffer, bringing it over the threshold level.
- Transmit FIFO Overflow Interrupt (`ssi_txo_intr`) – Set when an APB access attempts to write into the transmit FIFO after it has been completely filled. When set, data written from the APB is discarded. This interrupt remains set until you read the transmit FIFO overflow interrupt clear register (`TXOICR`).
- Receive FIFO Full Interrupt (`ssi_rxf_intr`) – Set when the receive FIFO is equal to or above its threshold value plus 1 and requires service to prevent an overflow. The threshold value, set through a software-programmable register, determines the level of receive FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data are read from the receive FIFO buffer, bringing it below the threshold level.
- Receive FIFO Overflow Interrupt (`ssi_rxo_intr`) – Set when the receive logic attempts to place data into the receive FIFO after it has been completely filled. When set, newly received data are discarded. This interrupt remains set until you read the receive FIFO overflow interrupt clear register (`RXOICR`).
- Receive FIFO Underflow Interrupt (`ssi_rxu_intr`) – Set when an APB access attempts to read from the receive FIFO when it is empty. When set, zeros are read back from the receive FIFO. This interrupt remains set until you read the receive FIFO underflow interrupt clear register (`RXUICR`).
- Multi-Master Contention Interrupt (`ssi_mst_intr`) – Present only when the `DW_apb_ssi` component is [configured](#) as a serial-master device. The interrupt is set when another serial master on the serial bus selects the `DW_apb_ssi` master as a serial-slave device and is actively transferring data. This informs the processor of possible contention on the serial bus. This interrupt remains set until you read the multi-master interrupt clear register (`MSTICR`).
- Combined Interrupt Request (`ssi_intr`) – OR'ed result of all the above interrupt requests after masking. To mask this interrupt signal, you must mask all other `DW_apb_ssi` interrupt requests.

#### 14.1.4 Transfer Modes

When transferring data on the serial bus, the DW\_apb\_ssi operates in the modes discussed in this section.

##### 14.1.4.1 Transmit and Receive

When TMOD = 2'b00, both transmit and receive logic are valid. The data transfer occurs as normal according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the txd line to the target device, which replies with data on the rxd line. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame.

##### 14.1.4.2 Transmit Only

When TMOD = 2'b01, the receive data are invalid and should not be stored in the receive FIFO. The data transfer occurs as normal, according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the txd line to the target device, which replies with data on the rxd line. At the end of the data frame, the receive shift register does not load its newly received data into the receive FIFO. The data in the receive shift register is overwritten by the next transfer. You should mask interrupts originating from the receive logic when this mode is entered.

##### 14.1.4.3 Receive Only

When TMOD = 2'b10, the transmit data are invalid. When **configured** as a slave, the transmit FIFO is never popped in Receive Only mode. The txd output remains at a constant logic level during the transmission. The data transfer occurs as normal according to the selected frame format (serial protocol). The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame. You should mask interrupts originating from the transmit logic when this mode is entered.

##### 14.1.4.4 EEPROM Read

When TMOD = 2'b11, the transmit data is used to transmit an opcode and/or an address to the EEPROM device. Typically this takes three data frames (8-bit opcode followed by 8-bit upper address and 8-bit lower address). During the transmission of the opcode and address, no data is captured by the receive logic (as long as the DW\_apb\_ssi master is transmitting data on its txd line, data on the rxd line is ignored). The DW\_apb\_ssi master continues to transmit data until the transmit FIFO is empty. Therefore, you should ONLY have enough data frames in the transmit FIFO to supply the opcode and address to the EEPROM. If more data frames are in the transmit FIFO than are needed, then read data is lost.

When the transmit FIFO becomes empty (all control information has been sent), data on the receive line (rxd) is valid and is stored in the receive FIFO; the txd output is held at a constant logic level. The serial transfer continues until the number of data frames received by the DW\_apb\_ssi master matches the value of the NDF field in the **CTRLR1** register + 1.

#### 14.1.5 Master SPI Serial Transfers

When the transfer mode is “transmit and receive” or “transmit only” (TMOD = 2'b00 or TMOD = 2'b01, respectively), transfers are terminated by the shift control logic when the transmit FIFO is empty. For continuous data transfers, you must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level (TXFTLR) can be used to early interrupt (**ssi\_txe\_intr**) the processor indicating that the transmit FIFO buffer is nearly empty.

When the transfer mode is “receive only” (TMOD = 2'b10), a serial transfer is started by writing one “dummy” data word into the transmit FIFO when a serial slave is selected. The txd output from the DW\_apb\_ssi is held at a constant logic level for the duration of the serial transfer. The transmit FIFO is popped only once at the beginning and may remain empty for the duration of the serial transfer. The end of the serial transfer is controlled by the “number of data frames” (NDF) field in control register 1 (**CTRLR1**).

If, for example, you want to receive 24 data frames from a serial-slave peripheral, you should program the NDF field with the value 23; the receive logic terminates the serial transfer when the number of frames received is equal to the NDF value + 1. This transfer mode increases the bandwidth of the APB bus as the transmit FIFO never needs to be serviced during the transfer. The receive FIFO buffer should be read each

---

time the receive FIFO generates a FIFO full interrupt request to prevent an overflow.

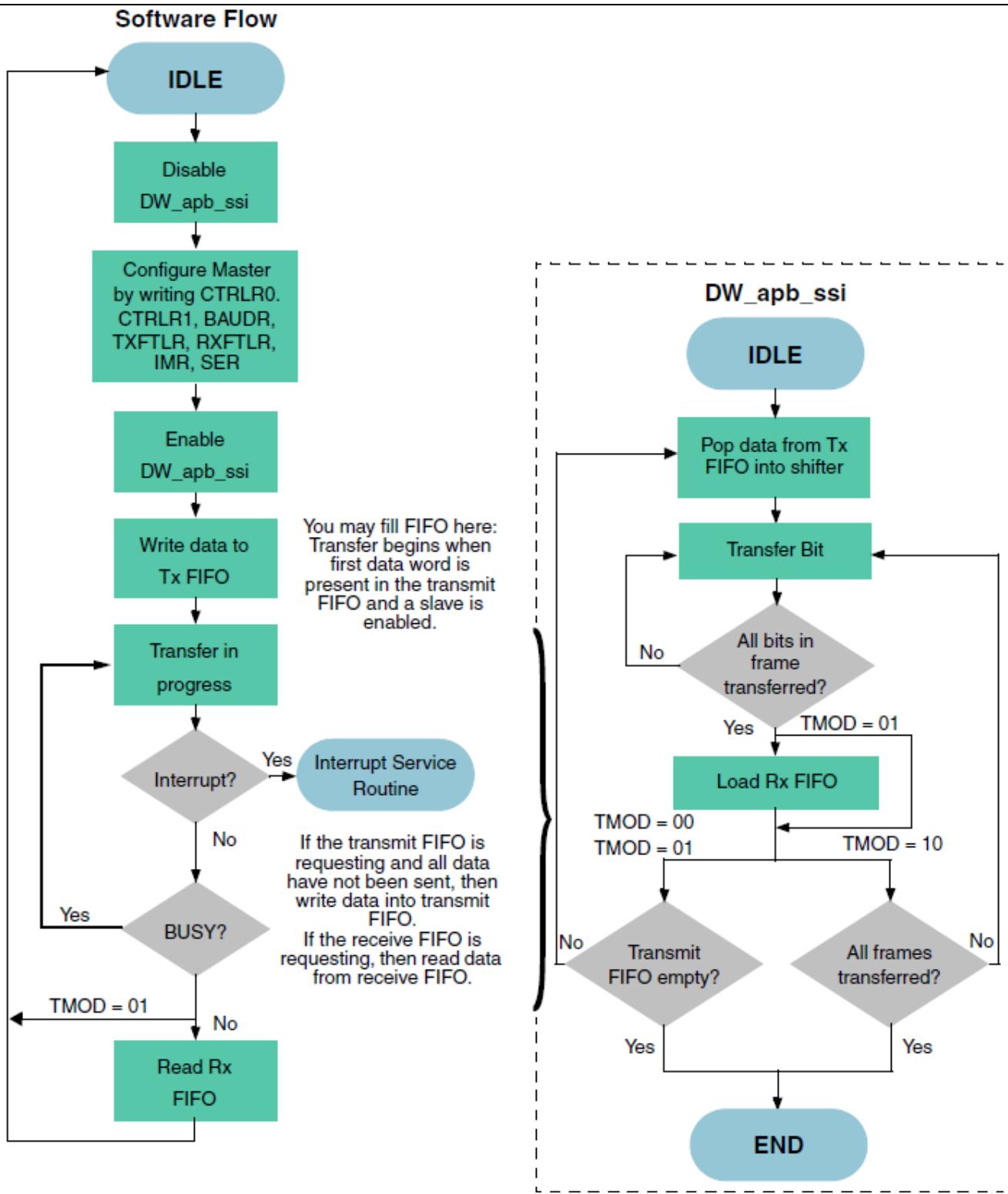
When the transfer mode is “eeprom\_read” ( $\text{TMOD} = 2'b11$ ), a serial transfer is started by writing the opcode and/or address into the transmit FIFO when a serial slave (EEPROM) is selected. The opcode and address are transmitted to the EEPROM device, after which read data is received from the EEPROM device and stored in the receive FIFO. The end of the serial transfer is controlled by the NDF field in the control register 1 ([CTRLR1](#)).

The receive FIFO threshold level (RXFTLR) can be used to give early indication that the receive FIFO is nearly full.

A typical software flow for completing an SPI serial transfer from the DW\_apb\_ssi serial master is outlined as follows:

1. If the DW\_apb\_ssi is enabled, disable it by writing 0 to the SSI Enable register ([SSIENR](#)).
2. Set up the DW\_apb\_ssi control registers for the transfer; these registers can be set in any order.
  - Write Control Register 0 ([CTRLR0](#)). For SPI transfers, the serial clock polarity and serial clock phase parameters must be set identical to target slave device.
  - If the transfer mode is *receive only*, write [CTRLR1](#) (Control Register 1) with the number of frames in the transfer minus 1; for example, if you want to receive four data frames, write this register with 3.
  - Write the Baud Rate Select Register ([BAUDR](#)) to set the baud rate for the transfer.
  - Write the Transmit and Receive FIFO Threshold Level registers ([TXFTLR](#) and [RXFTLR](#), respectively) to set FIFO threshold levels.
  - Write the [IMR](#) register to set up interrupt masks.
  - The Slave Enable Register ([SER](#)) register can be written here to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to writing to the Data Register ([DR](#)), the transfer does not begin until a slave is enabled.
3. Enable the DW\_apb\_ssi by writing 1 to the [SSIENR](#) register.
4. Write data for transmission to the target slave into the transmit FIFO (write [DR](#)).  
If no slaves were enabled in the [SER](#) register at this point, enable it now to begin the transfer.
5. Poll the BUSY status to wait for completion of the transfer. The BUSY status cannot be polled immediately; for more information, see the note on [page 40](#).
- If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write [DR](#)). If a receive FIFO full interrupt request is made, read the receive FIFO (read [DR](#)).
6. The transfer is stopped by the shift control logic when the transmit FIFO is empty. If the transfer mode is *receive only* ( $\text{TMOD} = 2'b10$ ), the transfer is stopped by the shift control logic when the specified number of frames have been received. When the transfer is done, the BUSY status is reset to 0.
7. If the transfer mode is not *transmit only* ( $\text{TMOD} != 01$ ), read the receive FIFO until it is empty.
8. Disable the DW\_apb\_ssi by writing 0 to [SSIENR](#).

[Figure 3-8](#) shows a typical software flow for starting a DW\_apb\_ssi master SPI serial transfer. The diagram also shows the hardware flow inside the serial-master component.



## 14.2 SSI Registers

### 14.2.1 Register Memory Maps

Register	Offset	Size	Memory Access	Description
ssi_address_block				

Register	Offset	Size	Memory Access	Description
<a href="#">CTRLR0</a>	0x0	32 bits	R/W	<p><b>Value After Reset:</b> 0x7</p> <p>Control Register 0: This register controls the serial data transfer. It is impossible to write to this register when the DW_apb_ssi is enabled. The DW_apb_ssi is enabled and disabled by writing to the SSIENR register.</p>
<a href="#">CTRLR1</a>	0x4	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Control Register 1 This register exists only when the DW_apb_ssi is configured as a master device. When the DW_apb_ssi is configured as a serial slave, writing to this location has no effect; reading from this location returns 0. Control register 1 controls the end of serial transfers when in receive-only mode. It is impossible to write to this register when the DW_apb_ssi is enabled. The DW_apb_ssi is enabled and disabled by writing to the SSIENR register.</p>
<a href="#">SSIENR</a>	0x8	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>SSI Enable Register</p>
<a href="#">MWCR</a>	0xc	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Microwire Control Register. This register controls the direction of the data word for the half-duplex Microwire serial protocol. It is impossible to write to this register when the DW_apb_ssi is enabled. The DW_apb_ssi is enabled and disabled by writing to the SSIENR register.</p>
<a href="#">SER</a>	0x10	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Slave Enable Register. This register is valid only when the DW_apb_ssi is configured as a master device. When the DW_apb_ssi is configured as a serial slave, writing to this location has no effect; reading from this location returns 0. The register enables the individual slave select output lines from the DW_apb_ssi master. Up to 16 slave-select output pins are available on the DW_apb_ssi master. You cannot write to this register when DW_apb_ssi is busy and when SSI_EN = 1.</p>
<a href="#">BAUDR</a>	0x14	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Baud Rate Select. This register is valid only when the DW_apb_ssi is configured as a master device. When the DW_apb_ssi is configured as a serial slave, writing to this location has no effect; reading from this location returns 0. The register derives the frequency of the serial clock that regulates the data transfer. The 16-bit field in this register defines the ssi_clk divider value. It is impossible to write to this register when the DW_apb_ssi is enabled. The DW_apb_ssi is enabled and disabled by writing to the SSIENR register.</p>
<a href="#">TXFTLR</a>	0x18	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Transmit FIFO Threshold Level. This register controls the threshold value for the transmit FIFO memory. The DW_apb_ssi is enabled and disabled by writing to the SSIENR register.</p>
<a href="#">RXFTLR</a>	0x1c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>Receive FIFO Threshold level. This register controls the threshold value for the receive FIFO memory. The DW_apb_ssi is enabled and disabled by writing to the SSIENR register.</p>
<a href="#">TXFLR</a>	0x20	32 bits	R	<p><b>Value After Reset:</b> 0x0</p> <p>Transmit FIFO Level Register</p>

Register	Offset	Size	Memory Access	Description
<a href="#">RXFLR</a>	0x24	32 bits	R	<b>Value After Reset:</b> 0x0 Receive FIFO Level Register
<a href="#">IMR</a>	0x2c	32 bits	R/W	<b>Value After Reset:</b> 0x3f Interrupt Mask Register
<a href="#">ISR</a>	0x30	32 bits	R	<b>Value After Reset:</b> 0x0 Interrupt Status Register
<a href="#">RISR</a>	0x34	32 bits	R	<b>Value After Reset:</b> 0x0 Raw Interrupt Status Register
<a href="#">TXOICR</a>	0x38	32 bits	R	<b>Value After Reset:</b> 0x0 Transmit FIFO Overflow Interrupt Clear Register
<a href="#">Rxoicr</a>	0x3c	32 bits	R	<b>Value After Reset:</b> 0x0 Receive FIFO Overflow Interrupt Clear Register
<a href="#">RXUICR</a>	0x40	32 bits	R	<b>Value After Reset:</b> 0x0 Receive FIFO Underflow Interrupt Clear Register
<a href="#">MSTICR</a>	0x44	32 bits	R	<b>Value After Reset:</b> 0x0 Multi-Master Interrupt Clear Register
<a href="#">ICR</a>	0x48	32 bits	R	<b>Value After Reset:</b> 0x0 Interrupt Clear Register
IDR	0x58	32 bits	R	<b>Value After Reset:</b> 0xffffffff Identification Register. This register contains the peripherals identification code, which is written into the register at configuration time using coreConsultant.
SSI_VERSION_ID	0x5c	32 bits	R/W	<b>Value After Reset:</b> 0x3332322a coreKit Version ID Register
<a href="#">DRO</a>	0x60	32 bits	R/W	<b>Value After Reset:</b> 0x0 The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
<a href="#">DR1</a>	0x64	32 bits	R/W	<b>Value After Reset:</b> 0x0 The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
<a href="#">DR2</a>	0x68	32 bits	R/W	<b>Value After Reset:</b> 0x0 The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.

Register	Offset	Size	Memory Access	Description
<a href="#">DR3</a>	0x6c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR4</a>	0x70	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR5</a>	0x74	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR6</a>	0x78	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR7</a>	0x7c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR8</a>	0x80	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR9</a>	0x84	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR10</a>	0x88	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>

Register	Offset	Size	Memory Access	Description
				SSI_EN = 0.
<a href="#">DR11</a>	0x8c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR12</a>	0x90	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR13</a>	0x94	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR14</a>	0x98	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR15</a>	0x9c	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR16</a>	0xa0	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR17</a>	0xa4	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR18</a>	0xa8	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit</p>

Register	Offset	Size	Memory Access	Description
				FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
<a href="#">DR19</a>	0xac	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR20</a>	0xb0	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR21</a>	0xb4	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR22</a>	0xb8	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR23</a>	0xbc	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR24</a>	0xc0	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR25</a>	0xc4	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.</p>
<a href="#">DR26</a>	0xc8	32 bits	R/W	<p><b>Value After Reset:</b> 0x0</p> <p>The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO</p>

Register	Offset	Size	Memory Access	Description
				buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
<a href="#">DR27</a>	0xcc	32 bits	R/W	<b>Value After Reset:</b> 0x0 The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
<a href="#">DR28</a>	0xd0	32 bits	R/W	<b>Value After Reset:</b> 0x0 The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
<a href="#">DR29</a>	0xd4	32 bits	R/W	<b>Value After Reset:</b> 0x0 The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
<a href="#">DR30</a>	0xd8	32 bits	R/W	<b>Value After Reset:</b> 0x0 The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
<a href="#">DR31</a>	0xdc	32 bits	R/W	<b>Value After Reset:</b> 0x0 The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
<a href="#">DR32</a>	0xe0	32 bits	R/W	<b>Value After Reset:</b> 0x0 The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
<a href="#">DR33</a>	0xe4	32 bits	R/W	<b>Value After Reset:</b> 0x0 The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
<a href="#">DR34</a>	0xe8	32 bits	R/W	<b>Value After Reset:</b> 0x0 The DW_apb_ssi data register is a 16-bit read/write buffer for the

Register	Offset	Size	Memory Access	Description
				transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
<a href="#">DR35</a>	0xec	32 bits	R/W	<b>Value After Reset:</b> 0x0 The DW_apb_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.
RSVD_0	0xf4	32 bits	R	<b>Value After Reset:</b> 0x0 RSVD_0 - Reserved address location
RSVD_1	0xf8	32 bits	R	<b>Value After Reset:</b> 0x0 RSVD_1 - Reserved address location
RSVD_2	0xfc	32 bits	R	<b>Value After Reset:</b> 0x0 RSVD_2 - Reserved address location

#### 14.2.2 Registers and Field Descriptions

Following is a description of the individual registers of DW\_apb\_ssi

##### 14.2.2.1 CTRLR0

**Size:** 32 bits

**Offset:** 0x0

**Memory Access:** R/W

**Value After Reset:** 0x7

31:16	15:12	11	10	9:8	7	6	5:4	3:0
(undef)	CFS	SRL	(undef)	TMOD	SCPOL	SCPH	FRF	DFS

Control Register 0: This register controls the serial data transfer. It is impossible to write to this register when the DW\_apb\_ssi is enabled. The DW\_apb\_ssi is enabled and disabled by writing to the SSIENR register.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:12	CFS	R/W	Control Frame Size. Selects the length of the control word for the Microwire frame format
11	SRL	R/W	Shift Register Loop. Used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input. 0 - Normal Mode Operation 1 - Test Mode Operation
10			Reserved for future use.
9:8	TMOD	R/W	Transfer Mode. Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid. In transmit-only mode, data received from the external device is not valid and is not stored in the receive FIFO memory; it is overwritten on the next transfer. In receive-only mode,

Bits	Name	Memory Access	Description
			transmitted data are not valid. After the first write to the transmit FIFO, the same word is retransmitted for the duration of the transfer. In transmit-and-receive mode, both transmit and receive data are valid. The transfer continues until the transmit FIFO is empty. Data received from the external device are stored into the receive FIFO memory, where it can be accessed by the host processor. 00 - Transmit & Receive 01 - Transmit Only 10 - Receive Only 11 - Reserved
7	SCPOL	R/W	Serial Clock Polarity. Valid when the frame format (FRF) is set to Motorola SPI. Used to select the polarity of the inactive serial clock, which is held inactive when the DW_apb_ssi master is not actively transferring data on the serial bus. 0 - Inactive state of serial clock is low 1 - Inactive state of serial clock is high
6	SCPH	R/W	Serial Clock Phase. Valid when the frame format (FRF) is set to Motorola SPI. The serial clock phase selects the relationship of the serial clock with the slave select signal. When SCPH = 0, data are captured on the first edge of the serial clock. When SCPH = 1, the serial clock starts toggling one cycle after the slave select line is activated, and data are captured on the second edge of the serial clock. 0: Serial clock toggles in middle of first data bit 1: Serial clock toggles at start of first data bit
5:4	FRF	R/W	Frame Format. Selects which serial protocol transfers the data. 00 - Motorola SPI 01 - Texas Instruments SSP 10 - National Semiconductors Microwire 11 - Reserved
3:0	DFS	R/W	Data Frame Size. Selects the data frame length. When the data frame size is programmed to be less than 16 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded. You must right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data

#### 14.2.2.2 CTRLR1

**Size:** 32 bits

**Offset:** 0x4

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:0
(undef)	NDF

Control Register 1 This register exists only when the DW\_apb\_ssi is configured as a master device. When the DW\_apb\_ssi is configured as a serial slave, writing to this location has no effect; reading from this location returns 0. Control register 1 controls the end of serial transfers when in receive-only mode. It is impossible to write to this register when the DW\_apb\_ssi is enabled. The DW\_apb\_ssi is enabled and disabled by writing to the SSIENR register.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	NDF	R/W	Number of Data Frames. When TMOD = 10, this register field sets the number of data frames to be continuously received by the DW_apb_ssi. The DW_apb_ssi continues to receive serial data until the number of data frames received is equal to this register value plus 1, which enables you to receive up to 64 KB of data in a continuous transfer. When the DW_apb_ssi is configured as a serial slave, the transfer continues for as long as the slave is selected. Therefore, this register serves no purpose and is not present when the DW_apb_ssi is configured as a serial slave.

**14.2.2.3 SSIENR****Size:** 32 bits**Offset:** 0x8**Memory Access:** R/W**Value After Reset:** 0x0

31:1	0
(undef)	SSI_EN

SSI Enable Register

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	SSI_EN	R/W	SSI Enable. Enables and disables all DW_apb_ssi operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the DW_apb_ssi control registers when enabled. When disabled, the ssi_sleep output is set (after delay) to inform the system that it is safe to remove the ssi_clk, thus saving power consumption in the system.

**14.2.2.4 MWCR****Size:** 32 bits**Offset:** 0xc**Memory Access:** R/W**Value After Reset:** 0x0

31:3	2	1	0
(undef)	MHS	MDD	MWMOD

Microwire Control Register. This register controls the direction of the data word for the half-duplex Microwire serial protocol. It is impossible to write to this register when the DW\_apb\_ssi is enabled. The DW\_apb\_ssi is enabled and disabled by writing to the SSIENR register.

Bits	Name	Memory Access	Description
31:3			Reserved for future use.
2	MHS	R/W	Microwire Handshaking. Relevant only when the DW_apb_ssi is configured as a serial-master device. When configured as a serial slave, this bit field has no functionality. Used to enable and disable the busy/ready handshaking interface for the Microwire protocol. When enabled, the DW_apb_ssi checks for a ready status from the target slave, after the transfer of the last data/control bit, before clearing the BUSY status in the SR register. 0: handshaking interface is disabled 1: handshaking interface is enabled
1	MDD	R/W	Microwire Control. Defines the direction of the data word when the Microwire serial protocol is used. When this bit is set to 0, the data word is received by the DW_apb_ssi MacroCell from the external serial device. When this bit is set to 1, the data word is transmitted from the DW_apb_ssi MacroCell to the external serial device.
0	MWMOD	R/W	Microwire Transfer Mode. Defines whether the Microwire transfer is sequential or non-sequential. When sequential mode is used, only one control word is needed to transmit or receive a block of data words. When non-sequential mode is used, there must be a control word for each data word that is transmitted or received. 0: non-sequential transfer 1: sequential transfer

**14.2.2.5 SER****Size:** 32 bits**Offset:** 0x10**Memory Access:** R/W**Value After Reset:** 0x0

31:1	0
(undef)	SER

Slave Enable Register. This register is valid only when the DW\_apb\_ssi is configured as a master device. When the DW\_apb\_ssi is configured as a serial slave, writing to this location has no effect; reading from this location returns 0. The register enables the individual slave select output lines from the DW\_apb\_ssi master. Up to 16 slave-select output pins are available on the DW\_apb\_ssi master. You cannot write to this register when DW\_apb\_ssi is busy and when SSI\_EN = 1.

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	SER	R/W	Slave Select Enable Flag. Each bit in this register corresponds to a slave select line (ss_x_n] from the DW_apb_ssi master. When a bit in this register is set (1), the corresponding slave select line from the master is activated when a serial transfer begins. It should be noted that setting or clearing bits in this register have no effect on the corresponding slave select outputs until a transfer is started. Before beginning a transfer, you should enable the bit in this register that corresponds to the slave device with which the master wants to communicate. When not operating in broadcast mode, only one bit in this field should be set. 1: Selected 0: Not Selected

**14.2.2.6 BAUDR****Size:** 32 bits**Offset:** 0x14**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	SCKDV

Baud Rate Select. This register is valid only when the DW\_apb\_ssi is configured as a master device. When the DW\_apb\_ssi is configured as a serial slave, writing to this location has no effect; reading from this location returns 0. The register derives the frequency of the serial clock that regulates the data transfer. The 16-bit field in this register defines the ssi\_clk divider value. It is impossible to write to this register when the DW\_apb\_ssi is enabled. The DW\_apb\_ssi is enabled and disabled by writing to the SSIENR register.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	SCKDV	R/W	SSI Clock Divider. The LSB for this field is always set to 0 and is unaffected by a write operation, which ensures an even value is held in this register. If the value is 0, the serial output clock (sclk_out) is disabled. The frequency of the sclk_out is derived from the following equation: $F_{sclk\_out} = F_{ssi\_clk}/SCKDV$ where SCKDV is any even value between 2 and 65534. For example: for Fssi_clk = 3.6864MHz and SCKDV = 2 $F_{sclk\_out} = 3.6864/2 = 1.8432\text{MHz}$

**14.2.2.7 TXFTLR****Size:** 32 bits

**Offset:** 0x18**Memory Access:** R/W**Value After Reset:** 0x0

31:3	2:0
(undef)	TFT

Transmit FIFO Threshold Level. This register controls the threshold value for the transmit FIFO memory. The DW\_apb\_ssi is enabled and disabled by writing to the SSIENR register.

Bits	Name	Memory Access	Description
31:3			Reserved for future use.
2:0	TFT	R/W	Transmit FIFO Threshold. Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt. The FIFO depth is configurable in the range 2-256; this register is sized to the number of address bits needed to access the FIFO. If you attempt to set this value greater than or equal to the depth of the FIFO, this field is not written and retains its current value. When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty interrupt is triggered.

#### 14.2.2.8 RXFTLR

**Size:** 32 bits**Offset:** 0x1c**Memory Access:** R/W**Value After Reset:** 0x0

31:3	2:0
(undef)	RFT

Receive FIFO Threshold level. This register controls the threshold value for the receive FIFO memory. The DW\_apb\_ssi is enabled and disabled by writing to the SSIENR register.

Bits	Name	Memory Access	Description
31:3			Reserved for future use.
2:0	RFT	R/W	Receive FIFO Threshold. Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt. The FIFO depth is configurable in the range 2-256. This register is sized to the number of address bits needed to access the FIFO. If you attempt to set this value greater than the depth of the FIFO, this field is not written and retains its current value. When the number of receive FIFO entries is greater than or equal to this value + 1, the receive FIFO full interrupt is triggered.

#### 14.2.2.9 TXFLR

**Size:** 32 bits**Offset:** 0x20**Memory Access:** R**Value After Reset:** 0x0

31:3	2:0
(undef)	TXFL

Transmit FIFO Level Register

Bits	Name	Memory Access	Description

Bits	Name	Memory Access	Description
31:3			Reserved for future use.
2:0	TXTFL	R	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

**14.2.2.10 RXFLR****Size:** 32 bits**Offset:** 0x24**Memory Access:** R**Value After Reset:** 0x0

31:3	2:0
(undef)	RXTFL

Receive FIFO Level Register

Bits	Name	Memory Access	Description
31:3			Reserved for future use.
2:0	RXTFL	R	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO.

**14.2.2.11 IMR****Size:** 32 bits**Offset:** 0x2c**Memory Access:** R/W**Value After Reset:** 0x3f

31:6	5	4	3	2	1	0
(undef)	MSTIM	RXFIM	RXOIM	RXUIM	TXOIM	TXEIM

Interrupt Mask Register

Bits	Name	Memory Access	Description
31:6			Reserved for future use.
5	MSTIM	R/W	Multi-Master Contention Interrupt Mask. This bit field is not present if the DW_apb_ssi is configured as a serial-slave device. 0 - ssi_mst_intr interrupt is masked 1 - ssi_mst_intr interrupt is not masked
4	RXFIM	R/W	Receive FIFO Full Interrupt Mask 0 - ssi_rxf_intr interrupt is masked 1 - ssi_rxf_intr interrupt is not masked
3	RXOIM	R/W	Receive FIFO Overflow Interrupt Mask 0 - ssi_rxo_intr interrupt is masked 1 - ssi_rxo_intr interrupt is not masked
2	RXUIM	R/W	Receive FIFO Underflow Interrupt Mask 0 - ssi_rxu_intr interrupt is masked 1 - ssi_rxu_intr interrupt is not masked
1	TXOIM	R/W	Transmit FIFO Overflow Interrupt Mask 0 - ssi_txo_intr interrupt is masked 1 - ssi_txo_intr interrupt is not masked
0	TXEIM	R/W	Transmit FIFO Empty Interrupt Mask 0 - ssi_txe_intr interrupt is masked 1 - ssi_txe_intr interrupt is not masked

**14.2.2.12 ISR****Size:** 32 bits**Offset:** 0x30**Memory Access:** R**Value After Reset:** 0x0

31:6	5	4	3	2	1	0
(undef)	MSTIS	RXFIS	RXOIS	RXUIS	TXOIS	TXEIS

Interrupt Status Register

Bits	Name	Memory Access	Description
31:6			Reserved for future use.
5	MSTIS	R	Multi-Master Contention Interrupt Status. This bit field is not present if the DW_apb_ssi is configured as a serial-slave device. 0 = ssi_mst_intr interrupt not active after masking 1 = ssi_mst_intr interrupt is active after masking
4	RXFIS	R	Receive FIFO Full Interrupt Status 0 = ssi_rxf_intr interrupt is not active after masking 1 = ssi_rxf_intr interrupt is full after masking
3	RXOIS	R	Receive FIFO Overflow Interrupt Status 0 = ssi_rxo_intr interrupt is not active after masking 1 = ssi_rxo_intr interrupt is active after masking
2	RXUIS	R	Receive FIFO Underflow Interrupt Status 0 = ssi_rxu_intr interrupt is not active after masking 1 = ssi_rxu_intr interrupt is active after masking
1	TXOIS	R	Transmit FIFO Overflow Interrupt Status 0 = ssi_txo_intr interrupt is not active after masking 1 = ssi_txo_intr interrupt is active after masking
0	TXEIS	R	Transmit FIFO Empty Interrupt Status 0 = ssi_txe_intr interrupt is not active after masking 1 = ssi_txe_intr interrupt is active after masking

**14.2.2.13 RISR****Size:** 32 bits**Offset:** 0x34**Memory Access:** R**Value After Reset:** 0x0

31:6	5	4	3	2	1	0
(undef)	MSTIR	RXFIR	RXOIR	RXUIR	TXOIR	TXEIR

Raw Interrupt StatusRegister

Bits	Name	Memory Access	Description
31:6			Reserved for future use.
5	MSTIR	R	Multi-Master Contention Raw Interrupt Status. This bit field is not present if the DW_apb_ssi is configured as a serial-slave device. 0 = ssi_mst_intr interrupt is not active prior to masking 1 = ssi_mst_intr interrupt is active prior to masking
4	RXFIR	R	Receive FIFO Full Raw Interrupt Status 0 = ssi_rxf_intr interrupt is not active prior to masking 1 = ssi_rxf_intr interrupt is active prior to masking
3	RXOIR	R	Receive FIFO Overflow Raw Interrupt Status 0 = ssi_rxo_intr interrupt is not active prior to masking 1 = ssi_rxo_intr interrupt is active prior to masking

Bits	Name	Memory Access	Description
2	RXUIR	R	Receive FIFO Underflow Raw Interrupt Status 0 = ssi_rxu_intr interrupt is not active prior to masking 1 = ssi_rxu_intr interrupt is active prior to masking
1	TXOIR	R	Transmit FIFO Overflow Raw Interrupt Status 0 = ssi_txo_intr interrupt is not active prior to masking 1 = ssi_txo_intr interrupt is active prior masking
0	TXEIR	R	Transmit FIFO Empty Raw Interrupt Status 0 = ssi_txe_intr interrupt is not active prior to masking 1 = ssi_txe_intr interrupt is active prior masking

**14.2.2.14 TXOICR****Size:** 32 bits**Offset:** 0x38**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	TXOICR

Transmit FIFO Overflow Interrupt Clear Register

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	TXOICR	R	Clear Transmit FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_txo_intr interrupt; writing has no effect.

**14.2.2.15 RXOICR****Size:** 32 bits**Offset:** 0x3c**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	RXOICR

Receive FIFO Overflow Interrupt Clear Register

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	RXOICR	R	Clear Receive FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxo_intr interrupt; writing has no effect.

**14.2.2.16 RXUICR****Size:** 32 bits**Offset:** 0x40**Memory Access:** R**Value After Reset:** 0x0

31:1	0
------	---

(undef)	RXUICR
---------	--------

Receive FIFO Underflow Interrupt Clear Register

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	RXUICR	R	Clear Receive FIFO Underflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxu_intr interrupt; writing has no effect.

**14.2.2.17 MSTICR****Size:** 32 bits**Offset:** 0x44**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	MSTICR

Multi-Master Interrupt Clear Register

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	MSTICR	R	Clear Multi-Master Contention Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_mst_intr interrupt; writing has no effect.

**14.2.2.18 ICR****Size:** 32 bits**Offset:** 0x48**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	ICR

Interrupt Clear Register

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	ICR	R	Clear Interrupts. This register is set if any of the interrupts below are active. A read clears the ssi_txo_intr, ssi_rxu_intr, ssi_rxo_intr, and the ssi_mst_intr interrupts. Writing to this register has no effect.

**14.2.2.19 DRO****Size:** 32 bits**Offset:** 0x60**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
-------	------

(undef)	dr0
---------	-----

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr0	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.20 DR1

**Size:** 32 bits

**Offset:** 0x64

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:0
(undef)	dr1

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr1	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.21 DR2

**Size:** 32 bits

**Offset:** 0x68

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:0
(undef)	dr2

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr2	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.22 DR3

**Size:** 32 bits

**Offset:** 0x6c

**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr3

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr3	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.23 DR4****Size:** 32 bits**Offset:** 0x70**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr4

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr4	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.24 DR5****Size:** 32 bits**Offset:** 0x74**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr5

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr5	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.25 DR6****Size:** 32 bits**Offset:** 0x78**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr6

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr6	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.26 DR7****Size:** 32 bits**Offset:** 0x7c**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr7

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr7	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.27 DR8****Size:** 32 bits**Offset:** 0x80**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr8

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr8	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.28 DR9****Size:** 32 bits**Offset:** 0x84**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr9

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr9	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.29 DR10****Size:** 32 bits**Offset:** 0x88**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr10

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr10	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.30 DR11****Size:** 32 bits**Offset:** 0x8c**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
-------	------

(undef)	dr11
---------	------

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr11	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.31 DR12

**Size:** 32 bits

**Offset:** 0x90

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:0
(undef)	dr12

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr12	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.32 DR13

**Size:** 32 bits

**Offset:** 0x94

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:0
(undef)	dr13

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr13	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.33 DR14

**Size:** 32 bits

**Offset:** 0x98

**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr14

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr14	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.34 DR15****Size:** 32 bits**Offset:** 0x9c**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr15

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr15	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.35 DR16****Size:** 32 bits**Offset:** 0xa0**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr16

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr16	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.36 DR17****Size:** 32 bits**Offset:** 0xa4**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr17

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr17	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.37 DR18****Size:** 32 bits**Offset:** 0xa8**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr18

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr18	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.38 DR19****Size:** 32 bits**Offset:** 0xac**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr19

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr19	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.39 DR20****Size:** 32 bits**Offset:** 0xb0**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr20

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr20	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.40 DR21****Size:** 32 bits**Offset:** 0xb4**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr21

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr21	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.41 DR22****Size:** 32 bits**Offset:** 0xb8**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
-------	------

(undef)	dr22
---------	------

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr22	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.42 DR23

**Size:** 32 bits

**Offset:** 0xbc

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:0
(undef)	dr23

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr23	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.43 DR24

**Size:** 32 bits

**Offset:** 0xc0

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:0
(undef)	dr24

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr24	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.44 DR25

**Size:** 32 bits

**Offset:** 0xc4

**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr25

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr25	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.45 DR26

**Size:** 32 bits**Offset:** 0xc8**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr26

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr26	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.46 DR27

**Size:** 32 bits**Offset:** 0xcc**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr27

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr27	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.47 DR28****Size:** 32 bits**Offset:** 0xd0**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr28

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr28	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.48 DR29****Size:** 32 bits**Offset:** 0xd4**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr29

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr29	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.49 DR30****Size:** 32 bits**Offset:** 0xd8**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr30

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr30	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.50 DR31****Size:** 32 bits**Offset:** 0xdc**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr31

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr31	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.51 DR32****Size:** 32 bits**Offset:** 0xe0**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
(undef)	dr32

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr32	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

**14.2.2.52 DR33****Size:** 32 bits**Offset:** 0xe4**Memory Access:** R/W**Value After Reset:** 0x0

31:16	15:0
-------	------

(undef)	dr33
---------	------

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr33	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.53 DR34

**Size:** 32 bits

**Offset:** 0xe8

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:0
(undef)	dr34

The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr34	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

#### 14.2.2.54 DR35

**Size:** 32 bits

**Offset:** 0xec

**Memory Access:** R/W

**Value After Reset:** 0x0

31:16	15:0
(undef)	dr35

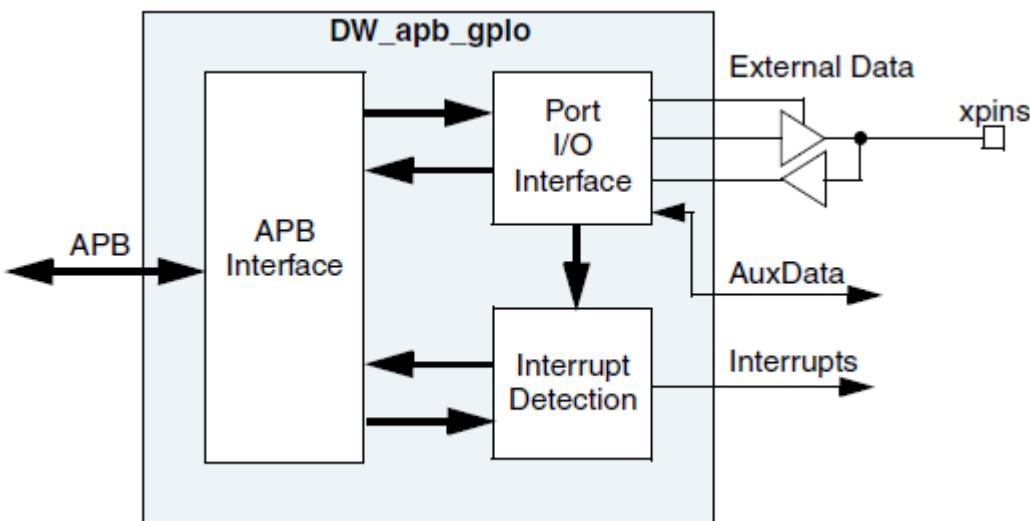
The DW\_apb\_ssi data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI\_EN = 1. FIFOs are reset when SSI\_EN = 0.

Bits	Name	Memory Access	Description
31:16			Reserved for future use.
15:0	dr35	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

## 15 GPIO

### 15.1 GPIO Modules

The DW\_apb\_gpio is a programmable General Purpose Programming I/O (GPIO) peripheral.



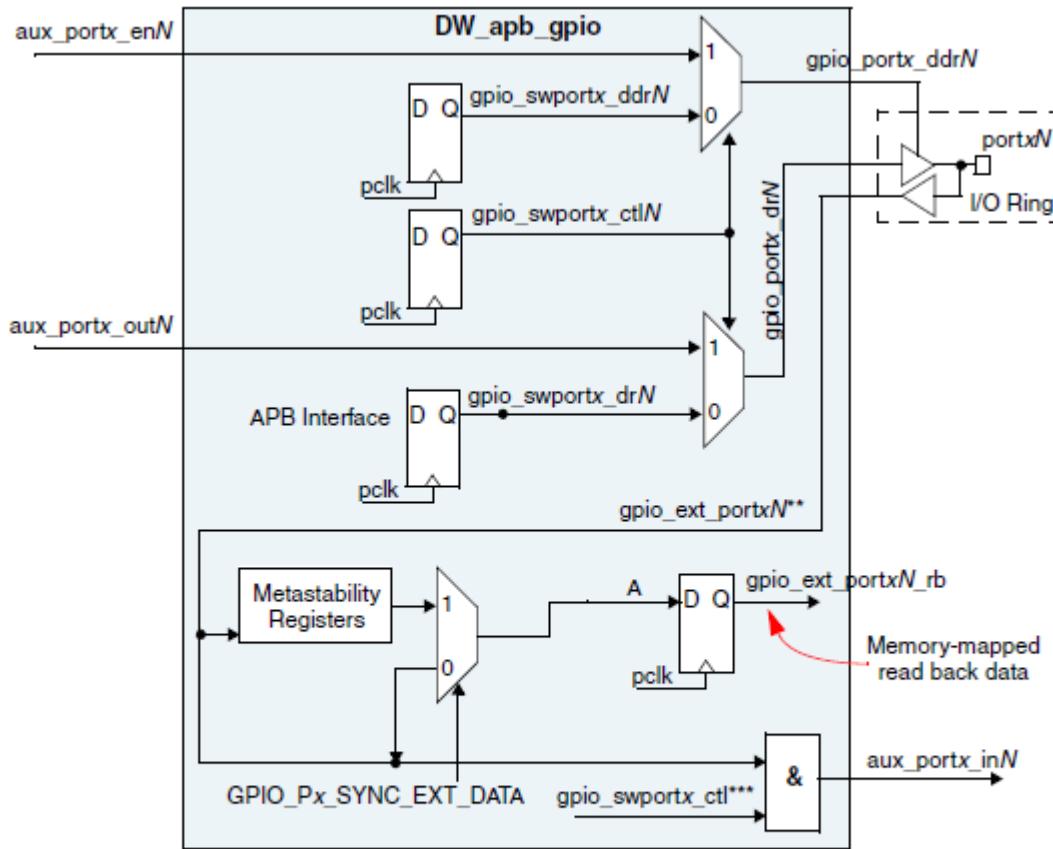
### 15.2 Function Description

#### 15.2.1 Data and Control Flow

The DW\_apb\_gpio controls the output data and direction of external I/O pads. It also can read back the data on external pads using memory-mapped registers.

There are two methods for generating the default source of the input data, the output data, and the control of each signal—through software or hardware control. Software control occurs over the APB bus interface; hardware control is through the auxiliary hardware control interface.

The software option always exists and is described in more detail in “[Software Control Mode](#)” on page 22. The hardware option for each signal exists only if you choose it during [configuration](#) time. Provided the hardware option is built, you can switch between software and hardware modes by writing to a control register for the corresponding signal. Also, the device can be [configured](#) so that you can individually switch between hardware and software modes for each bit of each signal, provided that hardware mode exists; for more detail, refer to “[Hardware Control Mode](#)” on page 14.2.1.2. The data and control flow for a signal are shown in [Figure 15-2](#).



N = 0 through GPIO\_PWIDHT\_X where "X" is A, B, C or D.

\* These data are multiplexed onto prdata when a read from the gpio\_ext\_portx register occurs.

\*\* This is a port signal. See [Table 5-1](#) on page 50 for more information.

\*\*\* If configuration parameter GPIO\_PORTN\_SINGLE\_CTL = 0, this remains a single bit as shown.

If GPIO\_PORTN\_SINGLE\_CTL = 1, this becomes a bus, **gpio\_swportx\_ctlN**.

### 15.2.1.1 Software Control Mode

When a signal is [configured](#) for software control, the data and direction control for the signal are sourced from the data register (**gpio\_swportx\_dr**) and direction control register (**gpio\_swportx\_ddr**), where x is either a, b, c, or d.

Under software control, the direction of the external I/O pad is controlled by a write to the Portx data direction register (**gpio\_swportx\_ddr**). The data written to this memory-mapped register gets mapped onto an output signal, **gpio\_portx\_ddr**, of the DW\_apb\_gpio peripheral. This output signal controls the direction of an external I/O pad.

The data written to the Portx data register (**gpio\_swportx\_dr**) drives the output buffer of the I/O pad.

External data are input on the external data signal, **gpio\_ext\_portx**. Depending on whether **gpio\_ext\_portx** is [configured](#) as an input or output, the following occurs:

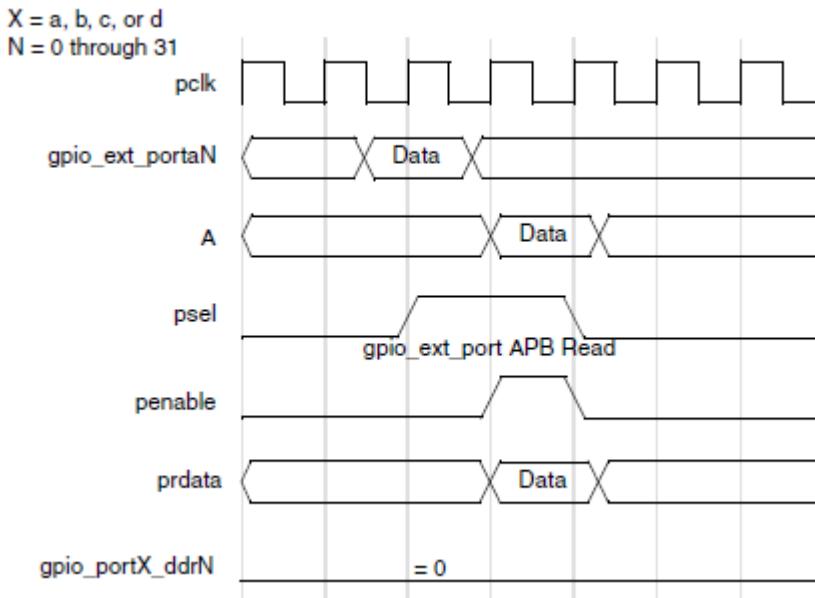
- Input – Reads the values on the signal
- Output – Reads the data register for Portx

The **gpio\_ext\_portx** register is read-only, meaning that it cannot be written from the APB software interface.

### 15.2.1.2 Reading External Signals

The data on the external **gpio** signal is read by an APB read of the memory-mapped register, **gpio\_ext\_portx**. An APB read to the **gpio\_ext\_portx** register provides either the data on the **gpio\_ext\_portx** control lines or the contents of the **gpio\_swportx\_dr**, depending on the value of **gpio\_swportx\_ddr**.

**Figure 15-3** shows a timing diagram of a read to the **gpio\_ext\_portx** memory map registers when the direction is set to *Input* and the metastability registers are included.



### 15.2.2 Interrupts

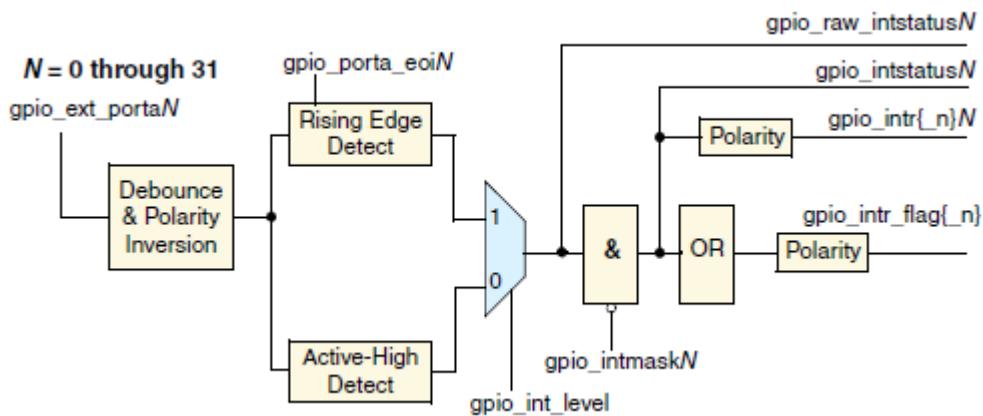
Port A can be programmed to accept external signals as interrupt sources on any of the bits of the signal. The type of interrupt is programmable with one of the following settings:

- Active-high and level
- Active-low and level
- Rising edge
- Falling edge

The interrupts can be masked by programming the `gpio_intmask` register. The interrupt status can be read before masking (called raw status) and after masking.

Whenever Port A is **configured** for interrupts, the data direction must be set to Input and the mode must be set to Software for interrupts to be latched. If the data direction register is reprogrammed to Output, then any pending interrupts are not lost. However, no new interrupts are generated.

**Figure 15-4** illustrates how the interrupts are generated and how the data flows. The signal names in the diagram correspond to either I/O signals or memory-mapped registers.



Two interrupt request connection schemes are supported, and one scheme is chosen during **configuration**. The simplest connection scheme is where the combined interrupt `gpio_intr_flag` is generated by ORing together the bits of the `gpio_intr` bus. When only the combined interrupt request is used, then the `gpio_status` register must be read in the interrupt service routine (ISR) to find the source of the interrupt. When the individual interrupts lines are connected directly to the interrupt controller, then the `gpio_status` register does not have to be read by the ISR.

For edge-detected interrupts, the ISR can clear the interrupt by writing a 1 to the `gpio_porta_eoi` register for the corresponding bit to disable the interrupt. This write also clears the interrupt status and raw status registers. It is

recommended that the interrupt source be cleared prior to writing to the `gpio_porta_eoi` register. Writing to the `gpio_porta_eoi` register has no effect on level-sensitive interrupts.

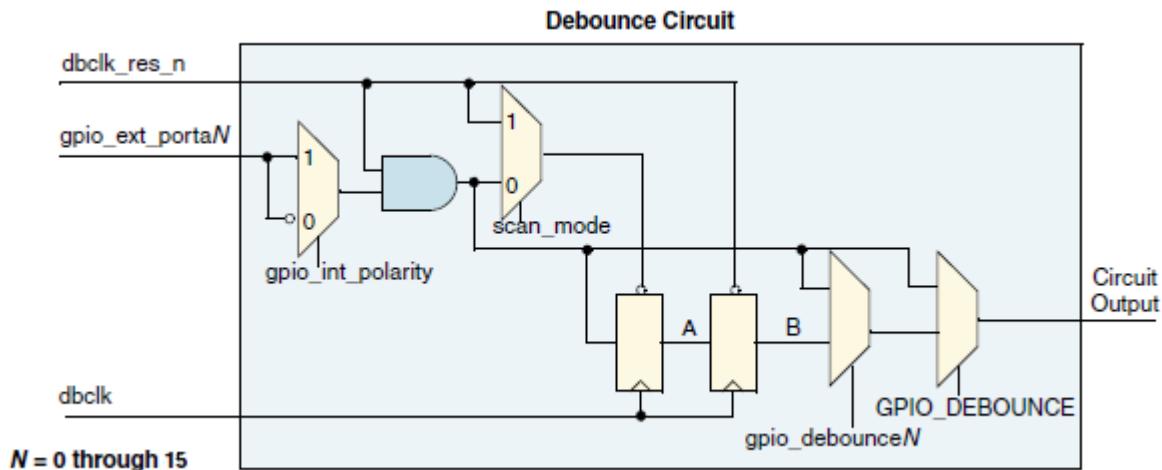
If level-sensitive interrupts cause the processor to interrupt, then the ISR can poll the `gpio_rawint` status register until the interrupt source disappears, or it can write to the `gpio_intmask` register to mask the interrupt before exiting the ISR. If the ISR exits without masking or disabling the interrupt prior to exiting, then the level-sensitive interrupt repeatedly requests an interrupt until the interrupt is cleared at the source.

If the `gpio_intr_flag` connection scheme is used and the interrupt service routine reads the `gpio_intr_status` register to find multiple pending interrupt requests, then it is up to the processor to prioritize these pending interrupt requests. There are no restrictions on the number of edge-detected interrupts that can be cleared simultaneously by writing multiple 1's to the `gpio_porta_eoi` register.

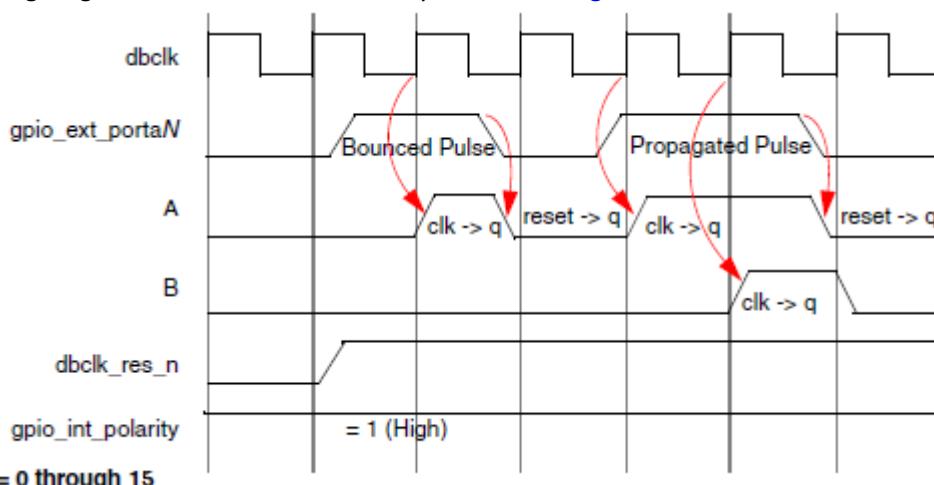
### 15.2.2.1 Debounce Operation

The external signal can be debounced to remove any spurious glitches that are less than one period of the external debouncing clock.

**Figure 15-5** shows an RTL diagram of the debounce circuitry. The timing diagram shows an active-high input signal on `gpio_ext_portaN`. The polarity of the input signal detection is controlled by the memory-mapped signal, `gpio_int_polarity`. For a falling-edge- or active-low-sensitive input, the input is then inverted and the same debounce logic is used as for rising-edge or active-high level-sensitive interrupts.



A timing diagram of the debounce circuitry is shown in [Figure 15-6](#).



When input interrupt signals are debounced using a debounce clock, the signals must be active for a minimum of two cycles of the debounce clock to guarantee that they are registered. Any input pulse widths less than a debounce clock period are bounced. A pulse width between one and two debounce clock widths may or may not propagate, depending on its phase relationship to the debounce clock. If the input pulse spans two rising edges of the debounce clock, it is registered. If it spans only one rising edge, it is not registered.

The timing diagram in [Figure 15-6](#) shows both cases: the input signal being bounced, and later, a propagated input signal. If the DW\_apb\_gpio supports debounce, then debouncing input signals on Port A can be enabled or disabled under software control.

The dbclk\_res\_n signal is asynchronously asserted and synchronously de-asserted to the debounce clock, dbclk. The system reset signal, presetn, is asynchronously asserted and synchronously de-asserted to pclk; synchronization must be external to the component. The pclk and dbclk signals are assumed to be asynchronous to each other.

The debounce circuitry works with only asynchronous reset flip-flops.

### 15.2.2.2 Synchronization of Interrupt Signals to the System Clock

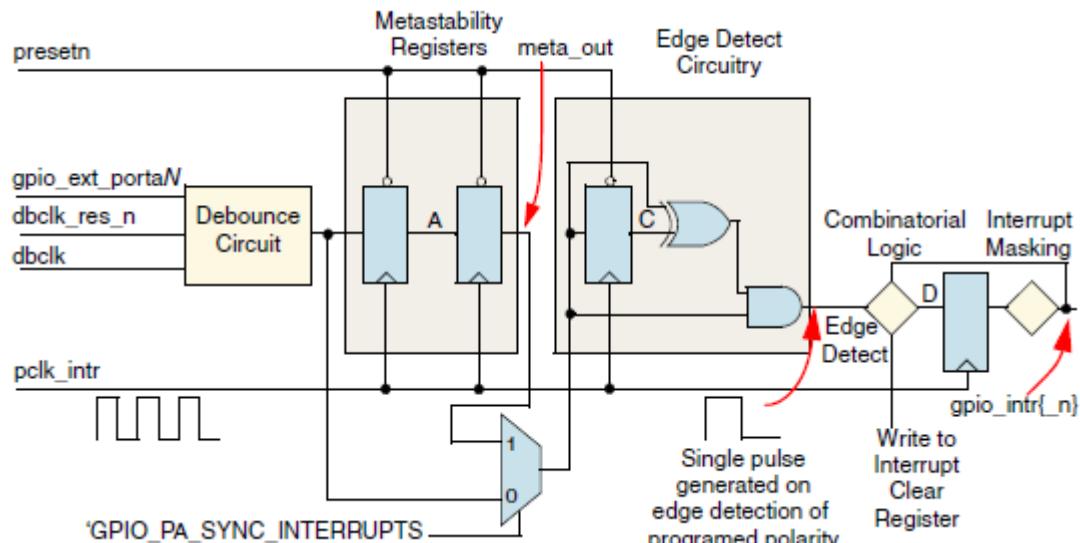
Interrupt signals can be internally synchronized to a system clock, pclk\_intr. Synchronization to pclk\_intr must occur for edge-detect signals. Edge-detected interrupts to the processor are always synchronous to the system bus clock. With level-sensitive interrupts, synchronization is optional and under software control.

The pclk\_intr signal is needed for systems that may have the DW\_apb\_gpio pclk bus clock gated off, but the system still wants to detect interrupts. It is assumed that this clock is synchronous to pclk. If interrupt detection is required only when pclk is running, then pclk\_intr and pclk can be connected to the same clock source. If the system employs a gated pclk to the DW\_apb\_gpio, pclk\_intr needs to be running for interrupt detection to occur.

The gpio\_intrclk\_en output signal is asserted when either edge-sensitive interrupts or level-sensitive interrupts requiring synchronization are enabled in the DW\_apb\_gpio block. Both cases require a clock for detection. Therefore, this signal can cause the external clock generator block to generate pclk\_intr.

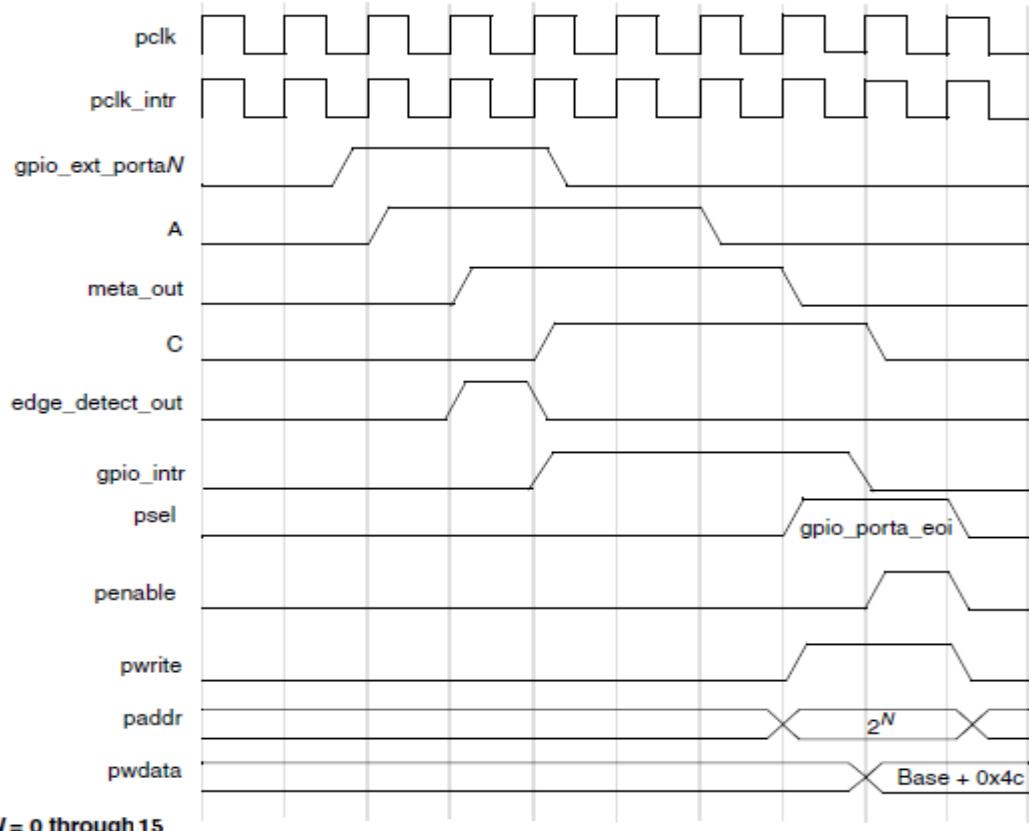
### 15.2.2.3 Interrupt Edge Detection

[Figure 15-7](#) shows an RTL diagram of the synchronization and edge detection of interrupt sources on gpio\_ext\_portaN signals.



The MUX allows inclusion or exclusion of the metastability registers at [configuration](#) depending on the value of GPIO\_PA\_SYNC\_INTERRUPTS. If this parameter is a 1, the registers are included.

[Figure 15-8](#) shows a timing diagram in which an interrupt is generated on the rising edge of an input on Port A; this is where the debounce logic is disabled and Metastability registers are included. It also shows how an interrupt is cleared by a write to the interrupt clear register.



A case may arise where the Interrupt Service Routine (ISR) writes to the interrupt clear register in order to clear an existing interrupt during the same clock cycle in which a new interrupt is detected. In such a case, writing to the interrupt clear register clears only the first interrupt. The second interrupt is not lost, since setting an interrupt has a higher priority than clearing it.

Figure 15-9 shows a timing diagram similar to Figure 15-8, except that Metastability registers are removed. It also shows how an interrupt is cleared by a write to the interrupt clear register.

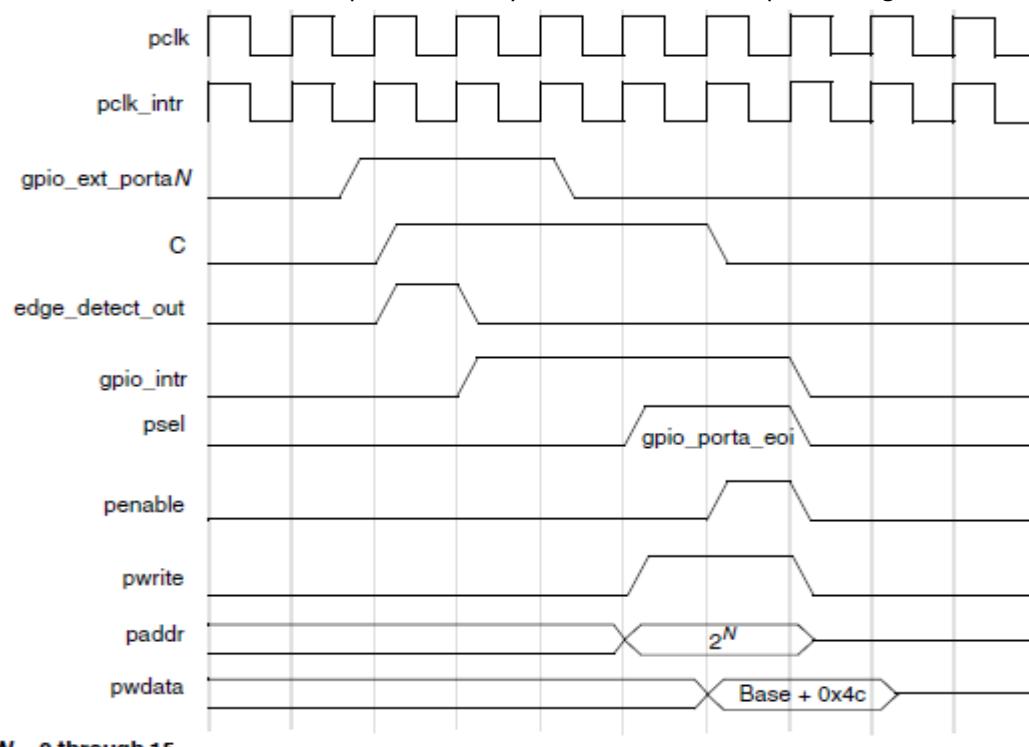
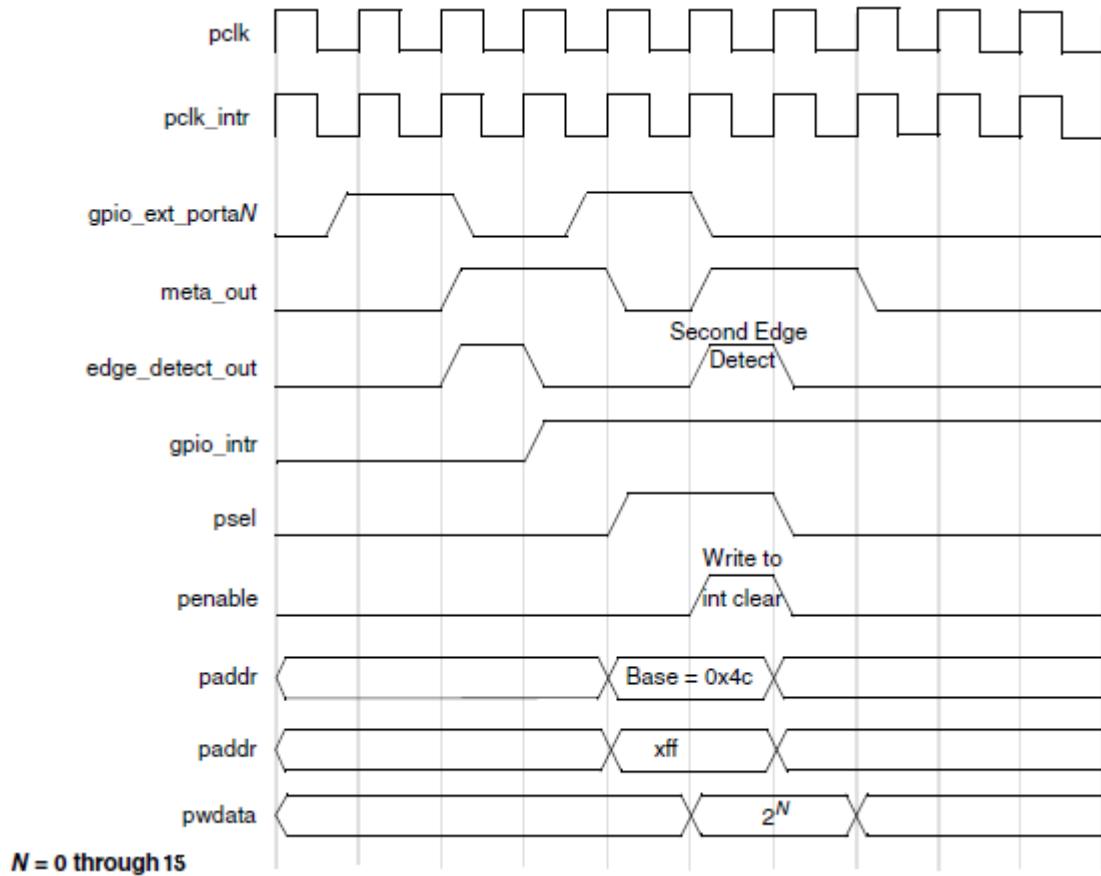
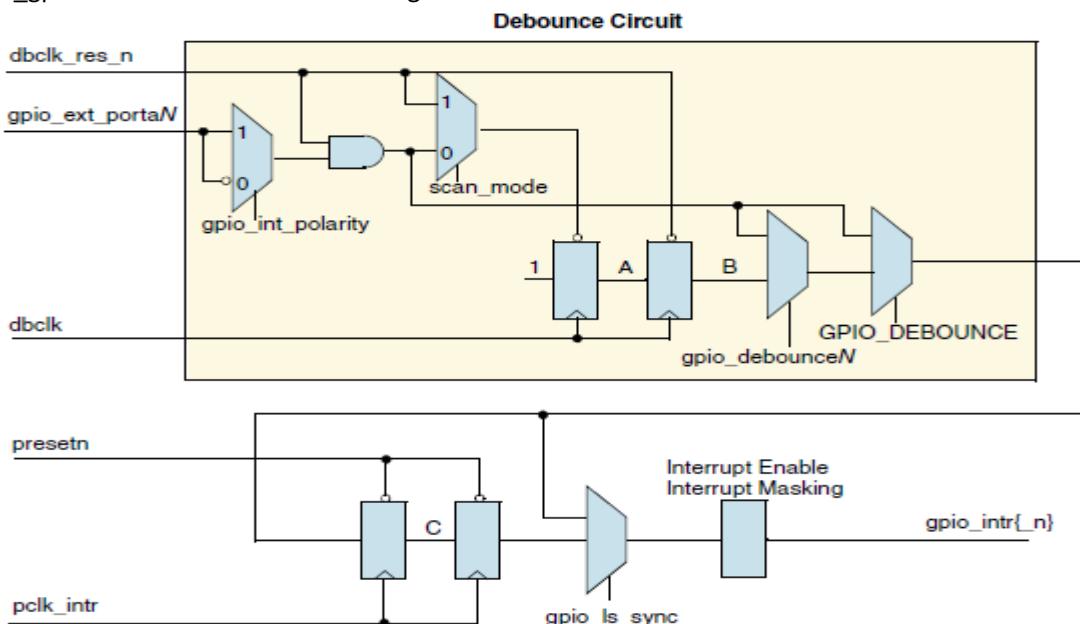


Figure 15-10 shows such a case where the debounce logic is unused. In this timing diagram, meta\_out and edge\_detect\_out are the outputs of the second metastability register and the edge detect logic, respectively. The second edge detection occurs on the same cycle as the write to the interrupt clear register. In this example, the write to the interrupt clear register does not clear the second interrupt, and the gpio\_intr{ $_n$ } signal is not de-asserted.



#### 15.2.2.4 Level-Sensitive Interrupts

Figure 15-11 shows the generation of level-sensitive interrupts. As for edge-detect interrupts, the user can [configure](#) DW\_apb\_gpio with or without debounce logic.

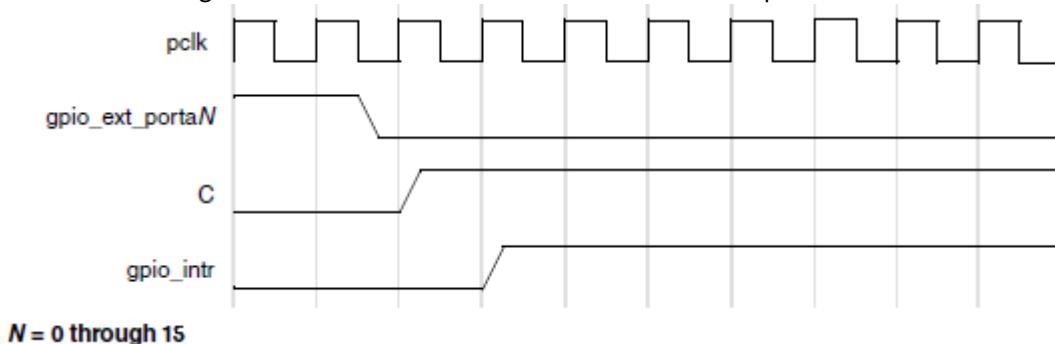


With level-sensitive interrupts, there is a choice of whether they are synchronized to the interrupt clock **pclk\_intr** or are entirely combinational (aside from the debounce circuitry). The selection is done by programming the **gpio\_ls\_sync** (GPIO Level Sensitive Synchronous) register.

This is a memory-mapped bit that inserts two metastability registers clocked off of pclk\_intr to synchronize the level-sensitive interrupts to pclk\_intr. When gpio\_ls\_sync is not asserted, then there is no guarantee that the interrupt lines are synchronous to pclk\_intr. A processor status register may need to be set to indicate asynchronous interrupts. When gpio\_ls\_sync is asserted, then the pclk\_intr clock must be present to pass the interrupt to the interrupt controller block. The gpio\_intrclk\_en output signal is asserted when level-sensitive interrupts that are to be synchronized to pclk\_intr are selected. The gpio\_intrclk\_en signal can be used in the clock generation block to turn on pclk\_intr.

The input signal is inverted for active-low level-sensitive interrupts. The same detection logic is used here as is used for active-high level-sensitive interrupts.

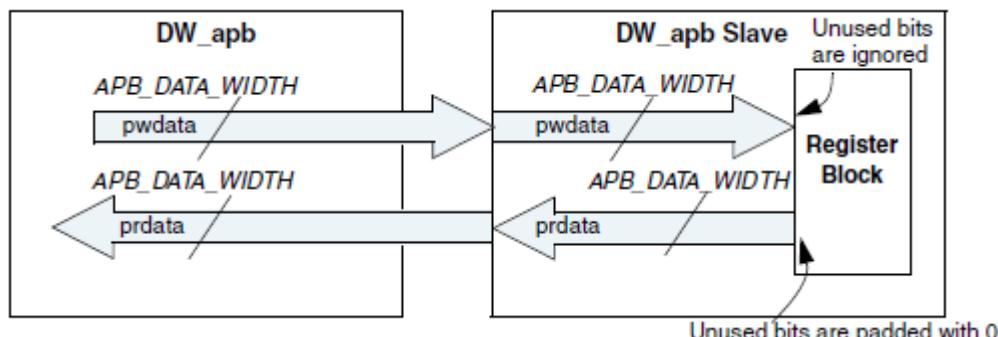
Figure 15-12 shows the generation of an active-low level-sensitive interrupt where the debounce circuitry is disabled.



### 15.3 GPIO Register Manual

#### 15.3.1 Bus Interface

The DW\_apb\_gpio peripheral has a standard AMBA 2.0 APB interface for reading and writing the internal registers. This peripheral supports APB data bus widths of 8, 16, or 32 bits., which is set with the.



#### 15.3.2 GPIO Register Mapping

Following is a register mapping of the GPIO\_0. GPIO\_1/2/3 have the same register mapping.

Register definitions for memory map.

Register	Offset	Size	Memory Access	Description
DW_apb_gpio address block				
<a href="#">GPIO_SWPORTA_DR</a>	0x0	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Port A data register set_attribute \$reg -attr RALAttribute -sub NO_BIT_BASH_TEST -value 1 Size: 1-32 bits Address Offset: 0x00 Read/Write Access: Read/Write
<a href="#">GPIO_SWPORTA_DDR</a>	0x4	32 bits	R/W	Name: Port A Data Direction Register Size: 1-32 bits Address Offset: 0x04 Read/Write Access: Read/Write

Register	Offset	Size	Memory Access	Description
<a href="#">GPIO_SWPORTB_DR</a>	0xc	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Port B data register Size: 1-32 bits Address Offset: 0x0c Read/Write Access: Read/Write
<a href="#">GPIO_SWPORTB_DDR</a>	0x10	32 bits	R/W	Name: Port B Data Direction Register Size: 1-32 bits Address Offset: 0x10 Read/Write Access: Read/Write
<a href="#">GPIO_SWPORTC_DR</a>	0x18	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Port C data register Size: 1-32 bits Address Offset: 0x18 Read/Write Access: Read/Write
<a href="#">GPIO_SWPORTC_DDR</a>	0x1c	32 bits	R/W	Name: Port C Data Direction Register Size: 1-32 bits Address Offset: 0x1c Read/Write Access: Read/Write
<a href="#">GPIO_SWPORTD_DR</a>	0x24	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Port D data register Size: 1-32 bits Address Offset: 0x24 Read/Write Access: Read/Write
<a href="#">GPIO_SWPORTD_DDR</a>	0x28	32 bits	R/W	Name: Port D Data Direction Register Size: 1-32 bits Address Offset: 0x28 Read/Write Access: Read/Write
<a href="#">GPIO_INTEN</a>	0x30	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Interrupt enable register Size: 1-32 bits Address Offset: 0x30 Read/Write Access: Read/Write
<a href="#">GPIO_INTMASK</a>	0x34	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Interrupt mask register Size: 1-32 bits Address Offset: 0x34 Read/Write Access: Read/Write
<a href="#">GPIO_INTTYPE_LEVEL</a>	0x38	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Interrupt level Size: 1-32 bits Address Offset: 0x38 Read/Write Access: Read/Write
<a href="#">GPIO_INT_POLARITY</a>	0x3c	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Interrupt polarity Size: 1-32 bits Address Offset: 0x3c Read/Write Access: Read/Write
<a href="#">GPIO_INTSTATUS</a>	0x40	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Interrupt status Size: 1-32 bits Address Offset: 0x40 Read/Write Access: Read/Write
<a href="#">GPIO_RAW_INTSTATUS</a>	0x44	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Raw interrupt status Size: 1-32 bits Address Offset: 0x44 Read/Write Access: Read/Write
<a href="#">GPIO_DEBOUNCE</a>	0x48	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Debounce enable Size: 1-32 bits Address Offset: 0x48 Read/Write Access: Read/Write
<a href="#">GPIO_PORTA_EOI</a>	0x4c	32 bits	W	<b>Value After Reset:</b> 0x0 Name: Port A clear interrupt register Size: 1-32 bits Address Offset: 0x4c Read/Write Access: Write
<a href="#">GPIO_EXT_PORTA</a>	0x50	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Port A external port register Size: 1-32 bits Address Offset: 0x50 Read/Write Access: Read

Register	Offset	Size	Memory Access	Description
<a href="#">GPIO_EXT_PORTB</a>	0x54	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Port B external port register Size: 1-32 bits Address Offset: 0x54 Read/Write Access: Read
<a href="#">GPIO_EXT_PORTC</a>	0x58	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Port C external port register Size: 1-32 bits Address Offset: 0x58 Read/Write Access: Read
<a href="#">GPIO_EXT_PORTD</a>	0x5c	32 bits	R	<b>Value After Reset:</b> 0x0 Name: Port D external port register Size: 1-32 bits Address Offset: 0x5c Read/Write Access: Read
<a href="#">GPIO_LS_SYNC</a>	0x60	32 bits	R/W	<b>Value After Reset:</b> 0x0 Name: Synchronization level Size: 1 bit Address Offset: 0x60 Read/Write Access: Read/Write
<a href="#">GPIO_ID_CODE</a>	0x64	32 bits	R	<b>Value After Reset:</b> 0x0 Name: GPIO ID code Size: 1-32 bits Address Offset: 0x64 Read/Write Access: Read
<a href="#">GPIO_VER_ID_CODE</a>	0x6c	32 bits	R	<b>Value After Reset:</b> 0x3230392a Name: GPIO Component Version Size: 32 bits Address Offset: 0x6c Read/Write Access: Read
<a href="#">GPIO_CONFIG_REG2</a>	0x70	32 bits	R	<b>Value After Reset:</b> 0x0 Name: GPIO Configuration Register 2 Size: 32 bits Address Offset: 0x70 Read/Write Access: Read
<a href="#">GPIO_CONFIG_REG1</a>	0x74	32 bits	R	<b>Value After Reset:</b> 0x1f70fe Name: GPIO Configuration Register 1 Size: 32 bits Address Offset: 0x74 Read/Write Access: Read

### 15.3.3 Register and Field Descriptions

Following is a description of the individual registers of component GPIO\_0. GPIO\_1/2/3 have the same register description.

#### 15.3.3.1 GPIO\_SWPORTA\_DR

**Size:** 32 bits

**Offset:** 0x0

Memory Access: R/W

Value After Reset: 0x0

31:1	0
(undef)	GPIO_SWPORTA_DR

Name: Port A data register set\_attribute \$reg -attr RALAttribute -sub NO\_BIT\_BASH\_TEST -value 1 Size: 1-32 bits Address Offset: 0x00 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_SWPORTA_DR	R/W	Values written to this register are output on the I/O signals for Port A if the

Bits	Name	Memory Access	Description
			corresponding data direction bits for Port A are set to Output mode and the corresponding control bit for Port A is set to Software mode. The value read back is equal to the last value written to this register.

### 15.3.3.2 GPIO\_SWPORTA\_DDR

**Size:** 32 bits

**Offset:** 0x4

**Memory Access:** R/W

31:1	0
(undef)	GPIO_SWPORTA_DDR

Name: Port A Data Direction Register Size: 1-32 bits Address Offset: 0x04 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_SWPORTA_DDR	R/W	<b>Value After Reset:</b> 0x0 Values written to this register independently control the direction of the corresponding data bit in Port A. The default direction can be configured as input or output after system reset through the GPIO_DFLT_SRC_A parameter. 0 Input (default) 1 Output

### 15.3.3.3 GPIO\_SWPORTB\_DR

**Size:** 32 bits

**Offset:** 0xc

**Memory Access:** R/W

**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_SWPORTB_DR

Name: Port B data register Size: 1-32 bits Address Offset: 0x0c Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_SWPORTB_DR	R/W	Values written to this register are output on the I/O signals for Port B if the corresponding data direction bits for Port B are set to Output mode and the corresponding control bit for Port B is set to Software mode. The value read back is equal to the last value written to this register.

### 15.3.3.4 GPIO\_SWPORTB\_DDR

**Size:** 32 bits

**Offset:** 0x10

**Memory Access:** R/W

31:1	0
(undef)	GPIO_SWPORTB_DDR

Name: Port B Data Direction Register Size: 1-32 bits Address Offset: 0x10 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_SWPORTB_DDR	R/W	<b>Value After Reset:</b> 0x0 Values written to this register independently control the direction of the corresponding data bit in Port B. The default direction can be configured as input or output after system reset through the GPIO_DFLT_SRC_B parameter. 0 Input (default) 1 Output

### 15.3.3.5 GPIO\_SWPORTC\_DR

**Size:** 32 bits

**Offset:** 0x18

**Memory Access:** R/W

**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_SWPORTC_DR

Name: Port C data register Size: 1-32 bits Address Offset: 0x18 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_SWPORTC_DR	R/W	Values written to this register are output on the I/O signals for Port C if the corresponding data direction bits for Port C are set to Output mode and the corresponding control bit for Port C is set to Software mode. The value read back is equal to the last value written to this register.

### 15.3.3.6 GPIO\_SWPORTC\_DDR

**Size:** 32 bits

**Offset:** 0x1c

**Memory Access:** R/W

31:1	0
(undef)	GPIO_SWPORTC_DDR

Name: Port C Data Direction Register Size: 1-32 bits Address Offset: 0x1c Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_SWPORTC_DDR	R/W	<b>Value After Reset:</b> 0x0 Values written to this register independently control the direction of the corresponding data bit in Port C. The default direction can be configured as input or output after system reset through the GPIO_DFLT_SRC_C parameter. 0 Input (default) 1 Output

### 15.3.3.7 GPIO\_SWPORTD\_DR

**Size:** 32 bits

**Offset:** 0x24

**Memory Access:** R/W

**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_SWPORTD_DR

Name: Port D data register Size: 1-32 bits Address Offset: 0x24 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_SWPORTD_DR	R/W	Values written to this register are output on the I/O signals for Port D if the corresponding data direction bits for Port D are set to Output mode and the corresponding control bit for Port D is set to Software mode. The value read back is equal to the last value written to this register.

**GPIO\_SWPORTD\_DDR****Size:** 32 bits**Offset:** 0x28**Memory Access:** R/W

31:1	0
(undef)	GPIO_SWPORTD_DDR

Name: Port D Data Direction Register Size: 1-32 bits Address Offset: 0x28 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_SWPORTD_DDR	R/W	<b>Value After Reset:</b> 0x0 Values written to this register independently control the direction of the corresponding data bit in Port D. The default direction can be configured as input or output after system reset through the GPIO_DFLT_SRC_D parameter. 0 Input (default) 1 Output

**15.3.3.8 GPIO\_INTEN****Size:** 32 bits**Offset:** 0x30**Memory Access:** R/W**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_INTEN

Name: Interrupt enable register Size: 1-32 bits Address Offset: 0x30 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_INTEN	R/W	Allows each bit of Port A to be configured for interrupts. By default the generation of interrupts is disabled. Whenever a 1 is written to a bit of this register, it configures the corresponding bit on Port A to become an interrupt; otherwise, Port A operates as a normal GPIO signal. Interrupts are disabled on the corresponding bits of Port A if the corresponding data direction register is set to Output or if Port A mode is set to Hardware. 0 Configure Port A bit as normal GPIO signal (default) 1 Configure Port A bit as

Bits	Name	Memory Access	Description
			interrupt

**15.3.3.9 GPIO\_INTMASK****Size:** 32 bits**Offset:** 0x34**Memory Access:** R/W**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_INTMASK

Name: Interrupt mask register Size: 1-32 bits Address Offset: 0x34 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_INTMASK	R/W	Controls whether an interrupt on Port A can create an interrupt for the interrupt controller by not masking it. By default, all interrupts bits are unmasked. Whenever a 1 is written to a bit in this register, it masks the interrupt generation capability for this signal; otherwise interrupts are allowed through. The unmasked status can be read as well as the resultant status after masking. 0 Interrupt bits are unmasked (default) 1 Mask interrupt

**15.3.3.10 GPIO\_INTPTYPE\_LEVEL****Size:** 32 bits**Offset:** 0x38**Memory Access:** R/W**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_INTPTYPE_LEVEL

Name: Interrupt level Size: 1-32 bits Address Offset: 0x38 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_INTPTYPE_LEVEL	R/W	Controls the type of interrupt that can occur on Port A. Whenever a 0 is written to a bit of this register, it configures the interrupt type to be level-sensitive; otherwise, it is edge-sensitive. 0 Level-sensitive (default) 1 Edge-sensitive

**15.3.3.11 GPIO\_INT\_POLARITY****Size:** 32 bits**Offset:** 0x3c**Memory Access:** R/W**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_INT_POLARITY

Name: Interrupt polarity Size: 1-32 bits Address Offset: 0x3c Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_INT_POLARITY	R/W	Controls the polarity of edge or level sensitivity that can occur on input of Port A. Whenever a 0 is written to a bit of this register, it configures the interrupt type to falling-edge or active-low sensitive; otherwise, it is rising-edge or active-high sensitive. 0 Active-low (default) 1 Active-high

### 15.3.3.12 GPIO\_INTSTATUS

**Size:** 32 bits

**Offset:** 0x40

**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_INTSTATUS

Name: Interrupt status Size: 1-32 bits Address Offset: 0x40 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_INTSTATUS	R	Interrupt status of Port A.

### 15.3.3.13 GPIO\_RAW\_INTSTATUS

**Size:** 32 bits

**Offset:** 0x44

**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_RAW_INTSTATUS

Name: Raw interrupt status Size: 1-32 bits Address Offset: 0x44 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_RAW_INTSTATUS	R	Raw interrupt of status of Port A (premasking bits)

### 15.3.3.14 GPIO\_DEBOUNCE

**Size:** 32 bits

**Offset:** 0x48

**Memory Access:** R/W

**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_DEBOUNCE

Name: Debounce enable Size: 1-32 bits Address Offset: 0x48 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_DEBOUNCE	R/W	Controls whether an external signal that is the source of an interrupt needs to be debounced to remove any spurious glitches. Writing a 1 to a bit in this register enables the debouncing circuitry. A signal must be valid for two periods of an external clock before it is internally processed. 0 No debounce (default) 1 Enable debounce

**15.3.3.15 GPIO\_PORTA\_EOI****Size:** 32 bits**Offset:** 0x4c**Memory Access:** W**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_PORTA_EOI

Name: Port A clear interrupt register Size: 1-32 bits Address Offset: 0x4c Read/Write Access: Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_PORTA_EOI	W	Controls the clearing of edge type interrupts from Port A. When a 1 is written into a corresponding bit of this register, the interrupt is cleared. All interrupts are cleared when Port A is not configured for interrupts. 0 No interrupt clear (default) 1 Clear interrupt

**15.3.3.16 GPIO\_EXT\_PORTA****Size:** 32 bits**Offset:** 0x50**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_EXT_PORTA

Name: Port A external port register Size: 1-32 bits Address Offset: 0x50 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_EXT_PORTA	R	When Port A is configured as Input, then reading this location reads the values on the signal. When the data direction of Port A is set as Output, reading this location reads the data register for Port A.

**15.3.3.17 GPIO\_EXT\_PORTB****Size:** 32 bits**Offset:** 0x54**Memory Access:** R**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_EXT_PORTB

Name: Port B external port register Size: 1-32 bits Address Offset: 0x54 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_EXT_PORTB	R	When Port B is configured as Input, then reading this location reads the values on the signal. When the data direction of Port B is set as Output, reading this location reads the data register for Port B.

### 15.3.3.18 GPIO\_EXT\_PORTC

**Size:** 32 bits

**Offset:** 0x58

**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_EXT_PORTC

Name: Port C external port register Size: 1-32 bits Address Offset: 0x58 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_EXT_PORTC	R	When Port C is configured as Input, then reading this location reads the values on the signal. When the data direction of Port C is set as Output, reading this location reads the data register for Port C.

### 15.3.3.19 GPIO\_EXT\_PORTD

**Size:** 32 bits

**Offset:** 0x5c

**Memory Access:** R

**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_EXT_PORTD

Name: Port D external port register Size: 1-32 bits Address Offset: 0x5c Read/Write Access: Read

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_EXT_PORTD	R	When Port D is configured as Input, then reading this location reads the values on the signal. When the data direction of Port D is set as Output, reading this location reads the data register for Port D.

### 15.3.3.20 GPIO\_LS\_SYNC

**Size:** 32 bits

**Offset:** 0x60

**Memory Access:** R/W

**Value After Reset:** 0x0

31:1	0
(undef)	GPIO_LS_SYNC

Name: Synchronization level Size: 1 bit Address Offset: 0x60 Read/Write Access: Read/Write

Bits	Name	Memory Access	Description
31:1			Reserved for future use.
0	GPIO_LS_SYNC	R/W	Writing a 1 to this register results in all level-sensitive interrupts being synchronized to pclk_intr. 0 No synchronization to pclk_intr (default) 1 Synchronize to pclk_intr

**15.3.3.21 GPIO\_ID\_CODE****Size:** 32 bits**Offset:** 0x64**Memory Access:** R**Value After Reset:** 0x0

31:0
GPIO_ID_CODE

Name: GPIO ID code Size: 1-32 bits Address Offset: 0x64 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:0	GPIO_ID_CODE	R	This is a user-specified code that a system can read. It can be used for chip identification, and so on.

**15.3.3.22 GPIO\_VER\_ID\_CODE****Size:** 32 bits**Offset:** 0x6c**Memory Access:** R**Value After Reset:** 0x3230392a

31:0
GPIO_VER_ID_CODE

Name: GPIO Component Version Size: 32 bits Address Offset: 0x6c Read/Write Access: Read

Bits	Name	Memory Access	Description
31:0	GPIO_VER_ID_CODE	R	ASCII value for each number in the version.

**15.3.3.23 GPIO\_CONFIG\_REG2****Size:** 32 bits**Offset:** 0x70**Memory Access:** R**Value After Reset:** 0x0

31:20	19:15	14:10	9:5	4:0
(undef)	ENCODED_ID_PWIDTH_D	ENCODED_ID_PWIDTH_C	ENCODED_ID_PWIDTH_B	ENCODED_ID_PWIDTH_A

Name: GPIO Configuration Register 2 Size: 32 bits Address Offset: 0x70 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:20			Reserved for future use.
19:15	ENCODED_ID_PWIDTH_D	R	The value of this register is derived from the GPIO_PWIDTH_D configuration parameter. 0x0 = 8 bits 0x1 = 16 bits 0x2 = 32 bits 0x3 = Reserved
14:10	ENCODED_ID_PWIDTH_C	R	The value of this register is derived from the GPIO_PWIDTH_C configuration parameter. 0x0 = 8 bits 0x1 = 16 bits 0x2 = 32 bits 0x3 = Reserved
9:5	ENCODED_ID_PWIDTH_B	R	The value of this register is derived from the GPIO_PWIDTH_B configuration parameter. 0x0 = 8 bits 0x1 = 16 bits 0x2 = 32 bits 0x3 = Reserved
4:0	ENCODED_ID_PWIDTH_A	R	The value of this register is derived from the GPIO_PWIDTH_A configuration parameter. 0x0 = 8 bits 0x1 = 16 bits 0x2 = 32 bits 0x3 = Reserved

### 15.3.3.24 GPIO\_CONFIG\_REG1

**Size:** 32 bits

**Offset:** 0x74

**Memory Access:** R

**Value After Reset:** 0x1f70fe

31:2	20:16	15	14	13	12	11	10	9	8	7	6	5	4	3:2	1:0
(undef)	ENCODED_ID_WIDTH	GPIO_ID	ADD_ENCODED_PARAMS	DEBOUNCE	PORATA_INTR	HW_PORTRTD	HW_PORTRTC	HW_PORTRTB	HW_PORTRTA	PORTD_SINGLE_CTL	PORTC_SINGLE_CTL	PORTB_SINGLE_CTL	PORTA_SINGLE_CTL	NUM_PORTS	APB_DATA_WIDTH

Name: GPIO Configuration Register 1 Size: 32 bits Address Offset: 0x74 Read/Write Access: Read

Bits	Name	Memory Access	Description
31:21			Reserved for future use.
20:16	ENCODED_ID_WIDTH	R	The value of this register is derived from the GPIO_ID_WIDTH configuration parameter.
15	GPIO_ID	R	The value of this register is derived from the GPIO_ID configuration parameter. 0 = Exclude 1 = Include
14	ADD_ENCODED_PARAMS	R	The value of this register is derived from the GPIO_ADD_ENCODED_PARAMS configuration parameter. 0 = False 1 = True
13	DEBOUNCE	R	The value of this register is derived from the GPIO_DEBOUNCE configuration parameter. 0 = Exclude 1 = Include
12	PORTA_INTR	R	The value of this register is derived from the GPIO_PORTA_INTR configuration parameter. 0 = Exclude 1 = Include
11	HW_PORTD	R	The value of this register is derived from the GPIO_HW_PORTD

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
			configuration parameter. 0 = Exclude 1 = Include
10	HW_PORTC	R	The value of this register is derived from the GPIO_HW_PORTC configuration parameter. 0 = Exclude 1 = Include
9	HW_PORTB	R	The value of this register is derived from the GPIO_HW_PORTB configuration parameter. 0 = Exclude 1 = Include
8	HW_PORTA	R	The value of this register is derived from the GPIO_HW_PORTA configuration parameter. 0 = Exclude 1 = Include
7	PORTD_SINGLE_CTL	R	The value of this register is derived from the GPIO_PORTD_SINGLE_CTL configuration parameter. 0 = False 1 = True
6	PORTC_SINGLE_CTL	R	The value of this register is derived from the GPIO_PORTC_SINGLE_CTL configuration parameter. 0 = False 1 = True
5	PORTB_SINGLE_CTL	R	The value of this register is derived from the GPIO_PORTB_SINGLE_CTL configuration parameter. 0 = False 1 = True
4	PORTA_SINGLE_CTL	R	The value of this register is derived from the GPIO_PORTA_SINGLE_CTL configuration parameter. 0 = False 1 = True
3:2	NUM_PORTS	R	The value of this register is derived from the GPIO_NUM_PORT configuration parameter. 0x0 = 1 0x1 = 2 0x2 = 3 0x3 = 4
1:0	APB_DATA_WIDTH	R	The value of this register is derived from the GPIO_APB_DATA_WIDTH configuration parameter. 0x0 = 8 bits 0x1 = 16 bits 0x2 = 32 bits 0x3 = Reserved

## 16 Initialization

### 16.1 Initialization Overview

This chapter provides an overview of the requirements for initializing an device from power-on through OS load and applications running. An overview of the overall initialization process is presented, including both hardware and software-related steps, a general overview of the boot ROM operational requirements, and the behavioral expectations.

A brief overview of the whole initialization process and its steps is illustrated.

Initialization of device consists of several steps:

- Preinitialization
- Ramp sequence for power, clocks, and resets
- Boot ROM
- Boot loader (Initial software launched by the ROM code during flash booting phase)
- OS start

Each of these steps, up to OS/applications running, is explained in detail in the following sections.



The first two steps in the initialization process are more hardware oriented, but do require a good understanding of the process of **configuring** those system interface pads (pins or balls on the device) that have software **configurable** functionality. Pad **configuration** is an essential part of chip **configuration** and is application-dependent. This chapter refers to those pads and the associated **configuration** registers that are vital for proper device initialization.

It is highly recommended that users refer to the *Pinout* and *Configuration* chapters for further details regarding the use of the pad **configuration** registers.

### 16.2 Pre-initialization

Certain hardware **configuration** settings must be made in order to accomplish a successful boot-up operation with an device. The clock, reset, and power connections, as well as pads involved in setting the boot memory space for the MPU, must be connected and driven properly for successful device initialization. This section details the specific requirements for the pre-initialization stage.

- Power Connections
  - The device can be supplied by an external companion chip.
- Clock **configuration**
- Reset **configuration**
- Boot **configuration**
- Pin Multiplexing and Pull-up/Pull-down

### 16.3 Power, Clock, and Reset Sequence on Power-up

Power-up Sequence for the device

Device power supplies can be provided by external companion chips.

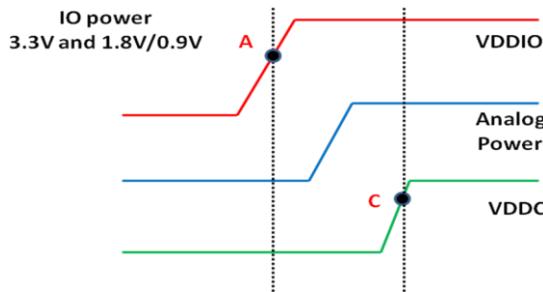
Below describes the power-on sequence for the device, including clocks, signals, and their relationship to the core and PLL voltages with the following steps:

1. Keep RESET\_N in low state.
2. Turn on I/O power. This includes 3.3V I/O power supply, DDR1.8V power supply and 0.9V reference voltage for DDR2 IO
3. Turn on 2.5V power to ADC/DAC/SNPS\_PLL
4. Turn on 1.2V core power.
5. When the power supplies are stabilized, raise RESET\_N from low to high and keep in high state during normal operation. And the CLK\_25MHz and CLK\_32MHz clock input is activated by external clock source.

- 
6. Once the voltages and clock are ramped and stable, the sysbot pins are sampled and the device is ready to enable the boot ROM to run.

Refer to figure below. Point 'A' is VDDIO crossing point when I/O power on control (POC) circuit starts to take control. When POC control is active, all outputs are in high impedance state. Point 'C' is the 70% crossing point of core power. POC control turns off when point 'C' is reached.

As long as IO power is greater than core power by one diode voltage the SoC will power-up normally without inducing crowbar current. There is 10us minimum time requirement between point 'A' and point 'C'. For best insurance, it is recommended to fully turn on VDDIO before turning on VDDC.

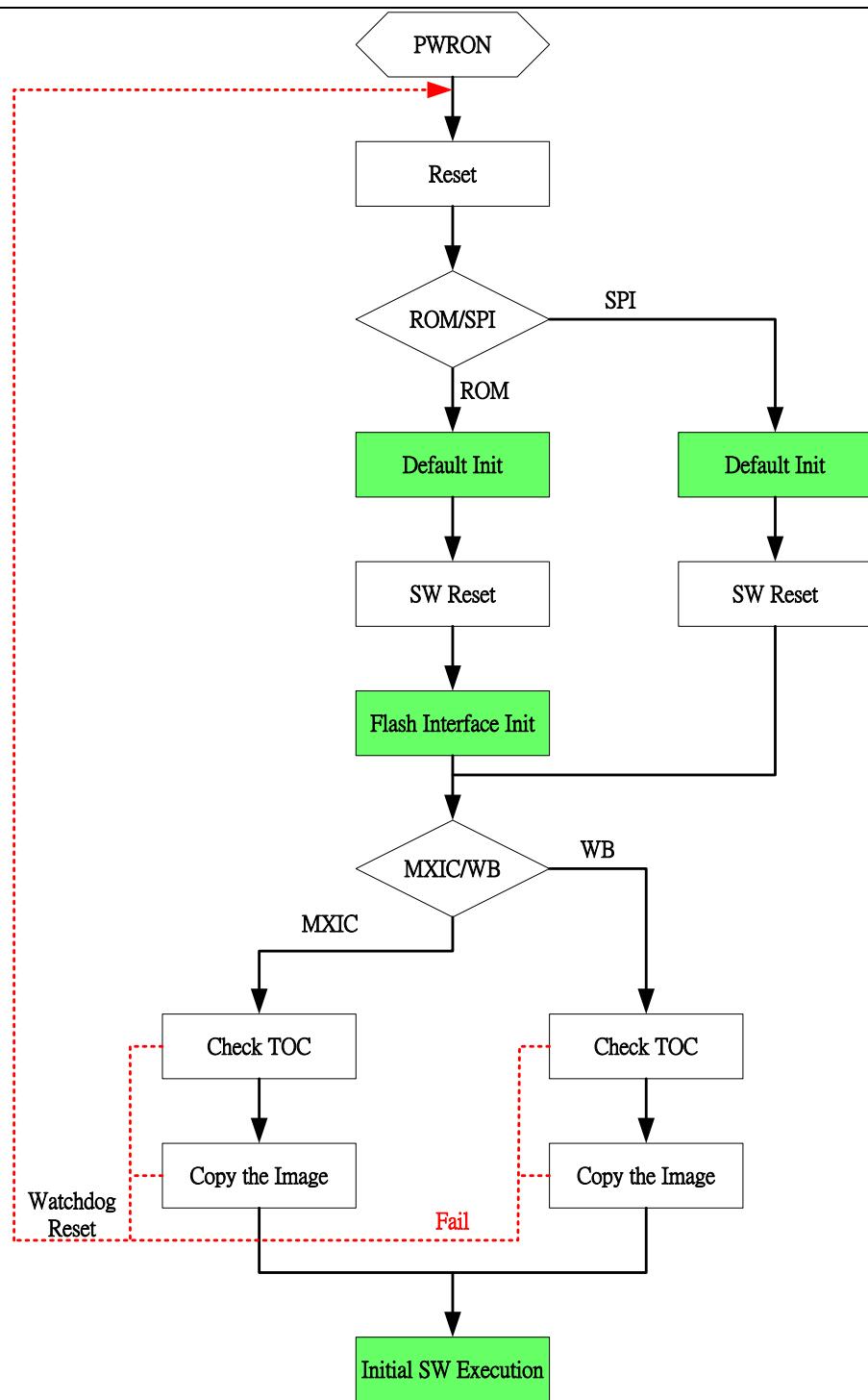


## 16.4 Device Booting by ROM code

This section describes the high-level booting concepts on the device and provides basic knowledge of booting on the device.

Bootstrapping is referred herein as the process of starting a bootstrap from the booting flash. The code executed during memory booting is usually used for OS start-up.

The ROM code finds the boot loader in external flash memory, copies it to external DDR, and executes it. The flow chart represents the ROM code structure.



## 17 Configuration

### 17.1 Function Configuration Overview

There are 4 pins (TESTM3, TESTM2, TESTM1, TESTM0) to define different function/test mode.

Test Mode Config.	TESTM3_2_1_0	Remark
Function 0	4'b0000	From SPI - HM ARMJTAG
Function 1	4'b0001	From SPI - HM Non ARMJTAG
Function 2	4'b0010	From SPI - HB ARMJTAG
Function 3	4'b0011	From SPI - HB Non ARMJTAG
Function 4	4'b0100	From InternalROM - HM ARMJTAG
Function 5	4'b0101	From InternalROM - HM Non ARMJTAG
Function 6	4'b0110	From InternalROM - HB ARMJTAG
Function 7	4'b0111	From InternalROM - HB Non ARMJTAG
ADC testing	4'b1000	ADC diagnosis
DAC testing	4'b1001	DAC diagnosis
ADC Scan testing	4'b1010	ADC Scan testing
Scan	4'b1011	DFT Scan testing
BIST	4'b1100	DFT BIST testing
BSD	4'b1101	DFT BSD testing
----	4'b1110	Reserved
----	4'b1111	Reserved

Scan/BIST/BSB are belongs to DFT testing. Wont' describe in here.

PIN_NAME	Function 0 ~ 7	ADC testing		DAC testing		ADC Scan testing	
		I/O	Signal	I/O	Signal	I/O	Signal
TESTM3		I	TESTM3	I	TESTM3	I	TESTM3
TESTM2		I	TESTM2	I	TESTM2	I	TESTM2
TESTM1		I	TESTM1	I	TESTM1	I	TESTM1
TESTM0		I	TESTM0	I	TESTM0	I	TESTM0
TCK							
TDI							
RTCK							
TDO							
TMS							
TRST_N							
RESET_N							
PHY_RESET							
OSCIN_32M							
OSCIN_25M							
CPU_TX_CLK		I	adc_clk_ext				
CPU_TXD3		I	adc_opm1 (connect to opmi1 and opmq1 both)	I	dac_endaci	I	adc_scan_clki
CPU_TXD2		I	adc_opm0 (connect to opmi0 and opmq0 both)	I	dac_endacq	I	adc_scan_clkq
CPU_TXD1		I	adc_use_prev_f	I	dac_stdby	I	adc_scanini_13

CPU_RXD0		I	adc_endcr	I	dac_enctr4	I	adc_scanini_12
CPU_TX_EN		I	adc_selof	I	dac_enctr3	I	adc_scanini_11
CPU_TX_ER		I	adc_startcal	I	dac_enctr2	I	adc_scanini_10
CPU_RXD3		I	adc_bcal1	I	dac_enctr1	I	adc_scanini_9
CPU_RXD2		I	adc_bcal0	I	dac_enctr0	I	adc_scanini_8
CPU_RXD1		I	adc_fsctrli7	I	dac_dacib11	I	adc_scanini_7
CPU_RXD0		I	adc_fsctrli6	I	dac_dacib10	I	adc_scanini_6
CPU_RX_CLK		I	adc_fsctrli5	I	dac_dacib9	I	adc_scanini_5
CPU_RX_ER		I	adc_fsctrli4	I	dac_dacib8	I	adc_scanini_4
CPU_RX_DV		I	adc_fsctrli3	I	dac_dacib7	I	adc_scanini_3
CPU_CRS		I	adc_fsctrli2	I	dac_dacib6	I	adc_scanini_2
CPU_COL		I	adc_fsctrli1	I	dac_dacib5	I	adc_scanini_1
CPU_MDC		I	adc_fsctrli0	I	dac_dacib4	I	adc_scanini_0
CPU_MDIO		I	adc_fsctrlq7	I	dac_dacib3	I	adc_scaninq_13
GMTX_GCLK		I	adc_fsctrlq6	I	dac_dacib2	I	adc_scaninq_12
GMTX_CLK		I	adc_fsctrlq5	I	dac_dacib1	I	adc_scaninq_11
GMTX_D7		I	adc_fsctrlq4	I	dac_dacib0	I	adc_scaninq_10
GMTX_D6		I	adc_fsctrlq3	I	dac_dacqb11	I	adc_scaninq_9
GMTX_D5		I	adc_fsctrlq2	I	dac_dacqb10	I	adc_scaninq_8
GMTX_D4		I	adc_fsctrlq1	I	dac_dacqb9	I	adc_scaninq_7
GMTX_D3		I	adc_fsctrlq0	I	dac_dacqb8	I	adc_scaninq_6
GMTX_D2		I	adc_bi11	I	dac_dacqb7	I	adc_scaninq_5
GMTX_D1		I	adc_bi10	I	dac_dacqb6	I	adc_scaninq_4
GMTX_D0		I	adc_bi9	I	dac_dacqb5	I	adc_scaninq_3
GMTX_ER		I	adc_bi8	I	dac_dacqb4	I	adc_scaninq_2
GMTX_EN		I	adc_bi7	I	dac_dacqb3	I	adc_scaninq_1
GMRX_CLK		I	adc_bi6	I	dac_dacqb2	I	adc_scaninq_0
GMRX_D7		I	adc_bi5	I	dac_dacqb1	O	adc_scanouti_13
GMRX_D6		I	adc_bi4	I	dac_dacqb0	O	adc_scanouti_12
GMRX_D5		I	adc_bi3	I	dac_clk	O	adc_scanouti_11
GMRX_D4		I	adc_bi2	I	dac_gain3	O	adc_scanouti_10
GMRX_D3		I	adc_bi1	I	dac_gain2	O	adc_scanouti_9
GMRX_D2		I	adc_bi0	I	dac_gain1	O	adc_scanouti_8
GMRX_D1		I	adc_bq11	I	dac_gain0	O	adc_scanouti_7
GMRX_D0		I	adc_bq10			O	adc_scanouti_6
GMRX_EN		O	adc_bq9			O	adc_scanouti_5
GMRX_ER		O	adc_bq8			O	adc_scanouti_4
GMRX_COL		O	adc_bq7			O	adc_scanouti_3
GMRX_CRS		O	adc_bq6			O	adc_scanouti_2
SPI_MEM_CS		O	adc_bq5			O	adc_scanouti_1
SPI_MEM_CLK		O	adc_bq4			O	adc_scanouti_0
SPI_MEM_DI		O	adc_bq3			O	adc_scanoutq_13
SPI_MEM_DO		O	adc_bq2			O	adc_scanoutq_12
SPI_MEM_WP		O	adc_bq1			O	adc_scanoutq_11
SPI_CLK		O	adc_bq0			O	adc_scanoutq_10
SPI_CS		O	adc_rflagi2			O	adc_scanoutq_9

SPI_DI		O	adc_rflagi1			O	adc_scanoutq_8
SPI_DO		O	adc_rflagi0			O	adc_scanoutq_7
GPIO15		O	adc_rflagq2			O	adc_scanoutq_6
GPIO14		O	adc_rflagq1			O	adc_scanoutq_5
GPIO13		O	adc_rflagq0			O	adc_scanoutq_4
GPIO12		O	adc_adcrdyi			O	adc_scanoutq_3
GPIO11		O	adc_adcrdyq			O	adc_scanoutq_2
GPIO10		O	adc_clkouti			O	adc_scanoutq_1
GPIO9		O	adc_clkoutq			O	adc_scanoutq_0
GPIO8		O	adc_i2c_scli			I	adc_scan_eni
GPIO7		O	adc_i2c_sclq			I	adc_scan_enq
GPIO6		O	adc_i2c_sdai			I	adc_scan_modei
GPIO5		O	adc_i2c_sdaq			I	adc_scan_modeq
GPIO4		O	adc_i2c_sdouti			I	adc_scan_resetzi
GPIO3		O	adc_i2c_sdoutq			I	adc_scan_resetzq
GPIO2		O	adc_i2c_addressi6			I	adc_i2c_resetz
GPIO1		O	adc_i2c_addressi5				
GPIO0		O	adc_i2c_addressi4				
UART_TX1		O	adc_i2c_addressi3				
UART_RX1		O	adc_i2c_addressi2				
UART_TX2		O	adc_i2c_addressi1				
UART_RX2		O	adc_i2c_addressi0				
I2C_SCLK		O	adc_i2c_addressq6				
I2C_SDA		I	adc_i2c_addressq5				
AFE_GAIN0		I	adc_i2c_addressq4				
AFE_GAIN1		I	adc_i2c_addressq3				
AFE_GAIN2		I	adc_i2c_addressq2				
AFE_GAIN3		O	adc_i2c_addressq1				
AFE_GAIN4		O	adc_i2c_addressq0				
AFE_POWERDN		I	adc_i2c_resetz				
AFE_TXEN							
AFE_RXEN							
PD_OUT							
PD_IN							
TXIN							
TXIP							
iref							
vdref							
TXQP							
TXQN							

## 17.2 I/O Configuration

There are 98 digital signals (excluding DDRIO) could be controlled their IO driver strength. Below table describes the signals used with dedicated IO cell type. DDRIO configuration is defined as chapter 6.

Cell type naming style: Ex: PDUW**0204**CDG

**02:** 2 mA driving strength

**04:** 4 mA driving strength

Name	Cell Type	PE	IE	DS
TCK	PDUW0204CDG	1	1	0
TDI	PDUW0204CDG	1	1	0
RTCK	PDUW0204CDG	1	1	0
TDO	PDUW0204CDG	1	1	0
TMS	PDUW0204CDG	1	1	0
TRST_N	PDUW0204CDG	1	1	0
RESET_N	PDUW0204CDG	1	1	0
PHY_RESET	PDUW0204CDG	1	1	0
OSCIN_32M	PDUW0204CDG	1	1	0
OSCIN_25M	PDUW0204CDG	1	1	0
CPU_TX_CLK	PDDW0408CDG	1	1	Reg_0 (Default 0)
CPU_RXD3	PDDW0408CDG	1	1	
CPU_RXD2	PDDW0408CDG	1	1	
CPU_RXD1	PDDW0408CDG	1	1	
CPU_RXD0	PDDW0408CDG	1	1	
CPU_RX_EN	PDDW0408CDG	1	1	
CPU_RX_ER	PDDW0408CDG	1	1	
CPU_RXD3	PDDW0408CDG	1	1	
CPU_RXD2	PDDW0408CDG	1	1	
CPU_RXD1	PDDW0408CDG	1	1	
CPU_RXD0	PDDW0408CDG	1	1	
CPU_RX_CLK	PDDW0408CDG	1	1	
CPU_RX_ER	PDDW0408CDG	1	1	
CPU_RX_DV	PDDW0408CDG	1	1	
CPU_CRS	PDDW0408CDG	1	1	
CPU_COL	PDDW0408CDG	1	1	
CPU_MDC	PDDW0408CDG	1	1	
CPU_MDIO	PDDW0408CDG	1	1	
GMTX_GCLK	PDDW0812CDG	1	1	Reg_1 (Default 0)
GMTX_CLK	PDDW0812CDG	1	1	
GMTX_D7	PDDW0812CDG	1	1	
GMTX_D6	PDDW0812CDG	1	1	
GMTX_D5	PDDW0812CDG	1	1	
GMTX_D4	PDDW0812CDG	1	1	
GMTX_D3	PDDW0812CDG	1	1	
GMTX_D2	PDDW0812CDG	1	1	
GMTX_D1	PDDW0812CDG	1	1	

GMTX_D0	PDDW0812CDG	1	1	
GMTX_ER	PDDW0812CDG	1	1	
GMTX_EN	PDDW0812CDG	1	1	
GMRX_CLK	PDDW0812CDG	1	1	
GMRX_D7	PDDW0812CDG	1	1	
GMRX_D6	PDDW0812CDG	1	1	
GMRX_D5	PDDW0812CDG	1	1	
GMRX_D4	PDDW0812CDG	1	1	
GMRX_D3	PDDW0812CDG	1	1	
GMRX_D2	PDDW0812CDG	1	1	
GMRX_D1	PDDW0812CDG	1	1	
GMRX_D0	PDDW0812CDG	1	1	
GMRX_EN	PDDW0812CDG	1	1	
GMRX_ER	PDDW0812CDG	1	1	
GMRX_COL	PDDW0812CDG	1	1	
GMRX_CRS	PDDW0812CDG	1	1	
SPI_MEM_CS	PRDW0408CDG	1	1	Reg_2 (Default 0)
SPI_MEM_CLK	PRDW0408CDG	1	1	
SPI_MEM_DI	PRDW0408CDG	1	1	
SPI_MEM_DO	PRDW0408CDG	1	1	
SPI_MEM_WP	PRDW0408CDG	1	1	
SPI_CLK	PRDW0408CDG	1	1	Reg_3 (Default 0)
SPI_CS	PRDW0408CDG	1	1	
SPI_DI	PRDW0408CDG	1	1	
SPI_DO	PRDW0408CDG	1	1	
GPIO15	PDUW0408CDG	1	1	Reg_4 (Default 0)
GPIO14	PDUW0408CDG	1	1	
GPIO13	PDUW0408CDG	1	1	
GPIO12	PDUW0408CDG	1	1	
GPIO11	PDUW0408CDG	1	1	
GPIO10	PDUW0408CDG	1	1	
GPIO9	PDUW0408CDG	1	1	
GPIO8	PDUW0408CDG	1	1	
GPIO7	PDUW0408CDG	1	1	
GPIO6	PDUW0408CDG	1	1	
GPIO5	PDUW0408CDG	1	1	Reg_5 (Default 0)
GPIO4	PDUW0408CDG	1	1	
GPIO3	PDUW0408CDG	1	1	
GPIO2	PDUW0408CDG	1	1	
GPIO1	PDUW0408CDG	1	1	
GPIO0	PDUW0408CDG	1	1	
UART_TX1	PDDW0204CDG	1	1	
UART_RX1	PDDW0204CDG	1	1	
UART_TX2	PDDW0204CDG	1	1	

UART_RX2	PDDW0204CDG	1	1	
I2C_SCLK	PRDW0408CDG	1	1	Reg_6 (Default 0)
I2C_SDA	PRDW0408CDG	1	1	
TESTM3	PDDW0204CDG	1	1	0
TESTM2	PDDW0204CDG	1	1	0
TESTM1	PDDW0204CDG	1	1	0
TESTM0	PDDW0204CDG	1	1	0
AFE_GAIN0	PDUW0408CDG	1	1	Reg_7 (Default 0)
AFE_GAIN1	PDUW0408CDG	1	1	
AFE_GAIN2	PDUW0408CDG	1	1	
AFE_GAIN3	PDUW0408CDG	1	1	
AFE_GAIN4	PDUW0408CDG	1	1	
AFE_POWERDN	PDUW0408CDG	1	1	
AFE_TXEN	PDUW0408CDG	1	1	
AFE_RXEN	PDUW0408CDG	1	1	
PD_OUT	PDUW0408CDG	1	1	
PD_IN	PDUW0408CDG	1	1	

### 17.3 I/O Multiplexing

### 17.4 Debug Module

## 18 Electrical Characteristics

### 18.1 DC Characteristics

#### 18.1.1 GPIO groups DC characteristics

The GPIO groups (Excluding ADC/DAC/DDR interface)

DC Characteristics					
Parameter		Min.	Nom.	Max.	Units
$V_{IL}$	Input Low Voltage	-0.3		0.8	V
$V_{IH}$	Input High Voltage	2		3.6	V
$V_T$	Threshold Point	1.42	1.56	1.73	V
$V_{T+}$	Schmitt Trigger Low to High Threshold Point	1.6	1.73	1.88	V
$V_{T-}$	Schmitt Trigger High to Low Threshold Point	1.25	1.39	1.55	V
$V_{T_{PU}}$	Threshold Point with Pull-up Resistor Enabled	1.4	1.54	1.71	V
$V_{T_{PD}}$	Threshold Point with Pull-down Resistor Enabled	1.44	1.57	1.74	V
$V_{T_{PU}^+}$	Schmitt Trigger Low to High Threshold Point with Pull-up Resistor Enabled	1.58	1.71	1.87	V
$V_{T_{PU}^-}$	Schmitt Trigger High to Low Threshold Point with Pull-up Resistor Enabled	1.23	1.37	1.54	V
$V_{T_{PD}^+}$	Schmitt Trigger Low to High Threshold Point with Pull-down Resistor Enabled	1.61	1.74	1.89	V
$V_{T_{PD}^-}$	Schmitt Trigger High to Low Threshold Point with Pull-down Resistor Enabled	1.25	1.4	1.56	V
$I_I$	Input Leakage Current @ $V_I=3.3V$ or 0V			$\pm 10\mu$	A
$I_{OZ}$	Tri-state Output Leakage Current @ $V_O=3.3V$ or 0V			$\pm 10\mu$	A
$R_{PU}$	Pull-up Resistor	35K	50K	81K	$\Omega$
$R_{PD}$	Pull-down Resistor	37K	52K	89K	$\Omega$
$V_{OL}$	Output Low Voltage			0.4	V
$V_{OH}$	Output High Voltage	2.4			V
$I_{OL}$	Low Level Output Current @ $V_{OL}(\max)$				

#### 18.1.2 DDR IO DC Electrical characteristics

The following table indicates the absolute maximum rating of DDRIO. Note that DDR IO are not guaranteed to function correctly when supply voltages are outside of the ranges listed in the table below, and may result in device damage.

Operating Voltage Range for 1.8V SSTL18 (DDR2) application.

Symbol	Parameters	Conditions	Min	Nom	Max
VDDPSSTL	SSTL18 1.8V post-driver power		1.7V	1.8V	1.9V
VD25SSTL	SSTL18 1.8V level shifter power		1.7V	1.8V	1.9V
VDDSSTL	SSTL18 core logic power in the I/O domain	$\pm 10\%$	0.9*VDD_CORE	VDD_CORE	1.1*VDD_CORE
VDDSSTD	Power cell core logic power in the digital block	$\pm 10\%$	0.9*VDD_CORE	VDD_CORE	1.1*VDD_CORE
VDDSSTLE	Power cell for dedicated core logic power in the I/O domain	$\pm 10\%$	0.9*VDD_CORE	VDD_CORE	1.1*VDD_CORE
VREFSSTL	SSTL18 reference voltage supply	VDDPSSTL /2	0.85V	0.9V	0.95V
$V_I$	Pad input voltage (PAD)		0V	VDDPSSTL	1.9V
$V_O$	Pad output voltage (PAD)		0V	VDDPSSTL	1.9V
$V_{IC}$	Core input voltage (I, OEN, PWD)		0V	VDDSSTL	1.1*VDD_CORE
$V_{OC}$	Core output voltage (C)		0V	VDDSSTL	1.1*VDD_CORE
$T_{OPT}$	Operating temperature		-40°C	25°C	125°C

The following table summarizes the DC parameters of 1.8V SSTL18 I/O cell. About the specifications, users could refer to the EIA/JEDEC standard document of JESD8-15A.

**1.8V SSTL18 DC Electrical Specification (Full driving strength,  
PMEMIO and PMEMIOP)**

Symbol	Parameters	Conditions	Min	Nom	Max
$V_{DDQ}$	SSTL18 1.8V I/O power for VDDPSST and VD25SSTL		1.7V	1.8V	1.9V
$V_{REF}$	SSTL18 reference voltage for VREFSSTL		0.833V	0.900V	0.969V
$V_{TT}$	Termination voltage		$V_{REF} - 0.04V$	$V_{REF}$	$V_{REF} + 0.04V$
$V_{IH(dc)}$	DC input logic high		$V_{REF} + 0.125V$		$V_{DDQ} + 0.3V$
$V_{IL(dc)}$	DC input logic low		-0.3V		$V_{REF} - 0.125V$
$I_{OH(dc)}$	Minimum DC source current of output driver	$V_{OH(dc)} = V_{DDQmin} - 0.28V$	$  -13.4mA  $		
$I_{OL(dc)}$	Minimum DC sink current of output driver	$V_{OL(dc)} = 0.28V$ and $V_{DDQmin} = 1.7V$	13.4mA		
$R_T$	Termination resistor		25 Ohm		
$R_S$	Series resistor		20 Ohm		

**1.8V SSTL18 DC Electrical Specifications (Half driving strength,  
PMEMIO and PMEMIOP)**

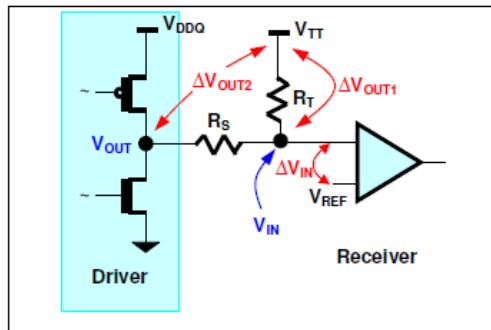
Symbol	Parameters	Conditions	Min	Nom	Max
$V_{DDQ}$	SSTL18 1.8V I/O power for VDDPSST and VD25SSTL		1.7V	1.8V	1.9V
$V_{REF}$	SSTL18 reference voltage for VREFSSTL		0.833V	0.900V	0.969V
$V_{TT}$	Termination voltage		$V_{REF} - 0.04V$	$V_{REF}$	$V_{REF} + 0.04V$
Symbol	Parameters	Conditions	Min	Nom	Max
$V_{IH(dc)}$	DC input logic high		$V_{REF} + 0.125V$		$V_{DDQ} + 0.3V$
$V_{IL(dc)}$	DC input logic low		-0.3V		$V_{REF} - 0.125V$
$I_{OH(dc)}$	Minimum DC source current of output driver	$@V_{OH(dc)} = V_{DDQmin} - 0.28V$	$  -6.7mA  $		
$I_{OL(dc)}$	Minimum DC sink current of output driver	$@V_{OL(dc)} = 0.28V$ and $V_{DDQmin} = 1.7V$	6.7mA		
$R_T$	Termination resistor		50 Ohm		
$R_S$	Series resistor		20 Ohm		

Table 4-2-3-3 1.8V SSTL18 Differential DC Electrical Specifications

Symbol	Parameters	Conditions	Min	Nom	Max
$V_{IN(dc)}$	DC input signal voltage		-0.3V		$V_{DDQ}+0.3V$
$V_{ID(dc)}$	DC differential input voltage	$ V_{TR}-V_{CP} ^*$	0.25V**		$V_{DDQ}+0.6V$

\*  $V_{TR}$  is the "true" input level and  $V_{CP}$  is the "complementary" input level.

\*\* This minimum value is equal to  $V_{IH(dc)}-V_{IL(dc)}$  from Table 4-2-3-2.

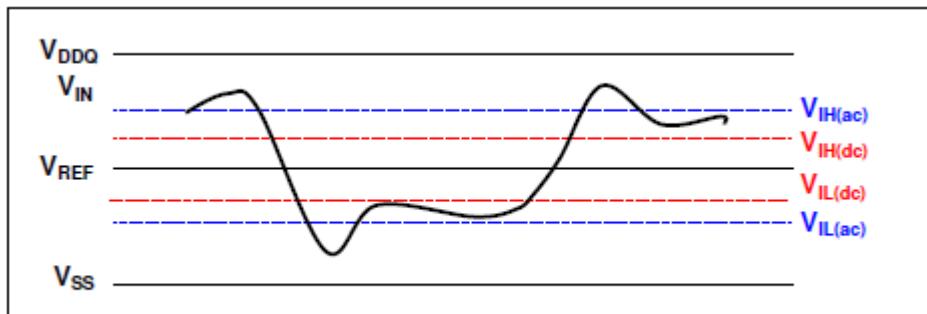


Typical SSTL18 systems and their symbols

## 18.2 AC Electrical Characteristics

### 18.2.1 DDR IO groups AC characteristics

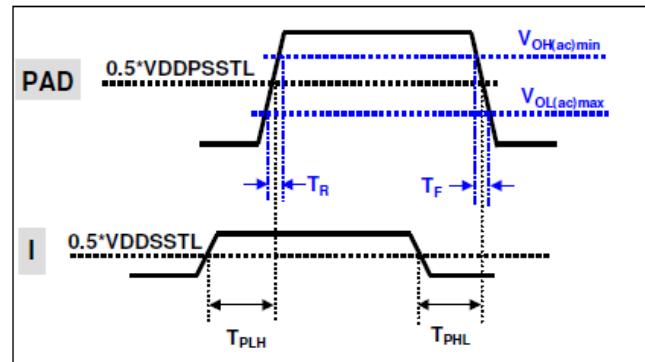
The AC values indicate the voltage levels at which the receiver must meet its timing specification. The DC values indicate the voltage level at which the final logic state of the receiver. Once the receiver input has crossed the ac value, the receiver will change to the new logic state. The new logic state will then be maintained as long as the input stays inside the DC threshold.



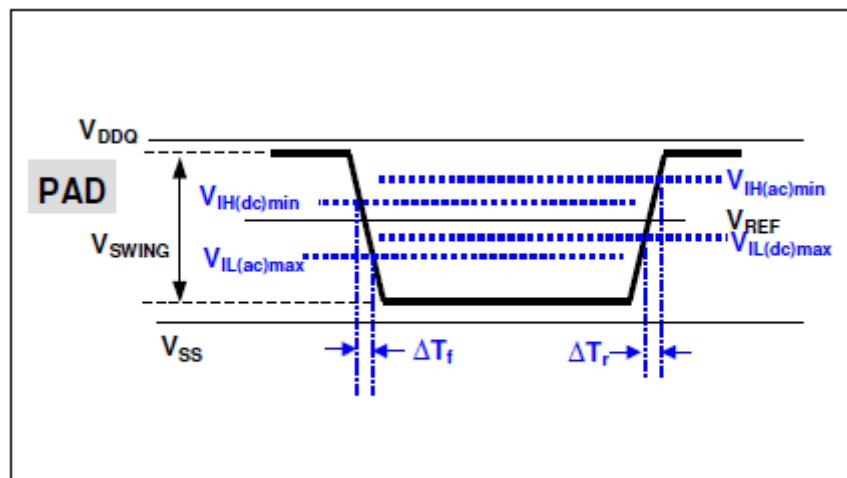
4 1.8V SSTL18 DC/AC input voltage

1.8V SSTL18 AC Electrical Specifications

Symbol	Parameter	Conditions	Min	Nom	Max
$V_{ID(ac)}$	AC differential input voltage		0.5V		$V_{DDQ}+0.6V$
$V_{IX(ac)}$	AC differential input cross point voltage		$0.5*V_{DDQ}-0.175V$		$0.5*V_{DDQ}+0.175V$
$V_{OX(ac)}$	AC differential output cross point voltage		$0.5*V_{DDQ}-0.125V$		$0.5*V_{DDQ}+0.125V$
$\Delta V_{OUT1}$	AC voltage across $R_S+R_T$			0.603V	
$V_{IH(ac)}$	AC input logic high	$(V_{REF}+\Delta V_{IN})$	$V_{REF}+0.25V$		
$V_{IL(ac)}$	AC input logic low	$(V_{REF}-\Delta V_{IN})$			$V_{REF}-0.25V$
$V_{OH(ac)}$	AC output logic high	$I_{OH}=13.4mA$	$V_{TTmax}+0.603V$		
$V_{OL(ac)}$	AC output logic low	$I_{OL}=13.4mA$			$V_{TTmin}-0.603V$
$V_{SWING}$	Input maximum swing	Peak-to-peak			1.0V
SLEW	Input minimum slew rate*		1.0V/ns		

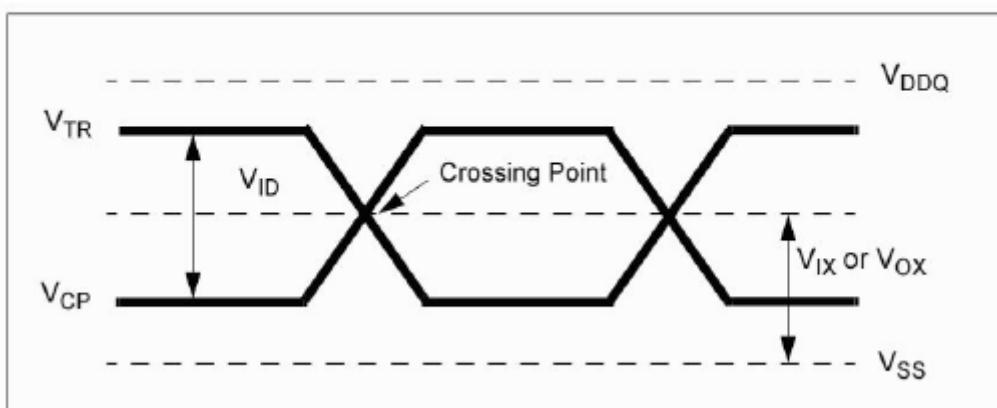


The definition of SSTL18 output rise time, fall time, and propagation delay.



The definition of SSTL18 single-ended input AC parameters

\* Input signal minimum slew rate needs to be maintained within the range from  $V_{IL(dc)max}$  to  $V_{IH(ac)min}$  for rising edges and the range from  $V_{IH(dc)min}$  to  $V_{IL(ac)max}$  for falling edges as show Fig. 6(a). Falling Slew is defined as  $(V_{IH(dc)min} - V_{IL(ac)max}) / \Delta T_f$ . Rising Slew is defined as  $(V_{IH(ac)min} - V_{IL(dc)max}) / \Delta T_r$ .



The definition of SSTL18 differential input AC parameters.

## 19 Package Specifications

- Package Size (D)(E): 13\*13 mm
- Package Type: LFBGA
- Ball count: 225
- Package height (A): 1.2 mm max.
- Substrate layer #: 4 layers
- Ball pitch (e): 0.8 mm
- Signal pin count: 140
- Pwr/Gnd pin count: 85
- Thermal: 1.5 W

## 20 Memory Usage

Memory	Size (Byte)	Usage
DCache	16K	Used for ARM data cache
ICache	16K	Used for ARM instruction cache
internal SRAM	64K	Used for internal buffer
Boot ROM	256x32bit	Used for ARM boot