i.MX23 Applications Processor Reference Manual

IMX23RM
Rev. 1
11/2009

freescale™
semiconductor

How to Reach Us:

Home Page:

www.freescale.com

Web Support:

http://www.freescale.com/support

USA/Europe or Locations Not Listed:

Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
+1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: IMX23RM
Rev. 1, 11/2009

freescale™
semiconductor

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 13
General-Purpose Media Interface (GPMI)

Contents

Contents

Contents

Chapter 18
LCD Interface (LCDIF)

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 33
Low-Resolution ADC and Touch-Screen Interface

Contents

Contents

Chapter 36
Pin Descriptions

Chapter 37
Pin Control and GPIO

Contents

Contents

Contents

Figures

Figures

Figures

Figures

Figures

Figures

Tables

Tables

Tables

Tables

Chapter 1
Product Overview

The i.MX23 is an applications processor targeted at devices that require low power, high performance, high integration and quality audio and video playback.

This chapter provides a general overview of the i.MX23 product and describes hardware features, application capability, design support, and additional documentation. See Table 1-1, and the pinout information in Chapter 36, "Pin Descriptions," for more detailed information about which functions described later in this document are supported in which package and part number.

Table 1-1. i.MX23 Functions by Package

Function	LQFP128	BGA169
External memory interface (2.5 V DDR, 1.8 V mDDR)	64MB Maximum, 16-bit data, 13-bit address, 1 chip enable	128MB Maximum 16-bit data, 13-bit address, 2 chip enable
General-Purpose Media Interface (GPMI): • NAND data width • Number of external NANDs supported	 8-bit data 4 (2 dedicated, 2 muxed)	 16-bit data 4 (3 dedicated, 1 muxed)
LCD Interface (LCDIF): • Up to 24-bit full-color parallel RGB mode • Up to 18-bit parallel RGB mode • 8-bit serial RGB mode • ITU-R BT.656 8-bit+clock mode • 8-bit system mode • Up to 18-bit parallel system mode • Up to 24 bit parallel system mode	 No No Yes Yes Yes No No	 Yes /w 8-bit NAND Yes Yes Yes Yes Yes Yes
Serial Audio Interface (SAIF or I^2S): • Interfaces supported (Note SAIF1/SAIF2 are muxed with LCD_DATA[8:16])	0	2
SPDIF Transmitter	No	Yes
Low-Resolution ADC (LRADC): • Number supported • Touch-screen supported	 2 (or 3 without 2.5 V DDR) No	 6 Yes
Application UART2: • Supported via dedicated pins	 No	 Yes

Table 1-1. i.MX23 Functions by Package (continued)

Function	LQFP128	BGA169
Synchronous Serial Port 1 (SSP1): • Data width	4-bit data	8-bit data
Synchronous Serial Port 2 (SSP2): • Data width (Note SSP2 is muxed with GPMI/NAND)	8-bit data	8-bit data
Embedded Trace Macrocell (ETM)	No	Yes
Pulse Width Modulation (PWM) Channels	3	5
Rotary Encoder	Yes (Muxed with PWM / DEBUG UART pins)	Yes (Dedicated pins)
Mono speaker amplifier	No	Yes
Real-Time Clock (RTC)	24 MHz	32 kHz and 24 MHz
Power Supply	Li-Ion	Li-Ion
4.2 V Regulated Output	Yes	Yes
Single Channel 10-bit Video DAC (Composite output)	Yes	Yes

1.1 Hardware Features

- ARM926 CPU Running at 454 MHz
  - — Integrated ARM926EJ-S CPU
  - — 16-Kbyte data cache and 16-Kbyte instruction cache
  - — ARM Embedded Trace Macrocell (ETM CoreSight 9) (169BGA only)
  - — One-wire JTAG interface
  - — Resistor-less boot mode selection using integrated OTP values
- 32 Kbytes of Integrated Low-Power On-Chip RAM
- 64 Kbytes of Integrated Mask-Programmable On-Chip ROM
- 1 Kbit of On-Chip One-Time-Programmable (OCOTP) ROM
- Universal Serial Bus (USB) High-Speed (Up to 480 Mb/s), Full-Speed (Up to 12 Mb/s)
  - — Full-speed/high-speed USB device and host functions
  - — Fully integrated full-speed/high-speed Physical Layer Protocol (PHY)
  - — Mass storage host-capable (uncertified by USB-IF)
- Power Management Unit
  - — Single inductor DC-DC switched converter with multi-channel output supporting Li-Ion batteries.
  - — Features multi-channel outputs for VDDIO (3.3 V), VDDD (1.2 V), VDDA (1.8 V), VDDM (2.5V) and regulated 4.2V source.
  - — Direct power from 5-V source (USB, wall power, or other source), with programmable current limits for load and battery charge circuits.

  - Silicon speed and temperature sensors enable adaptive power management over temperature and silicon process.
- Audio Codec
  - Stereo headphone DAC with 99 dB SNR
  - Stereo ADC with 85 dB SNR
  - Stereo headphone amplifier with short-circuit protection and direct drive to eliminate bulky capacitors
  - Mono speaker amplifier (169-Pin BGA only) providing up to 2W rms output, running directly from the battery.
  - Amplifiers are designed for click/pop free operation.
  - Two stereo line inputs
  - Microphone input
  - SPDIF digital out
- 16-Channel Low-Resolution ADC
  - 6 independent channels and 10 dedicated channels
  - Resistive touchscreen controller
  - Temperature sensor controller
  - Absolute accuracy of 1.3%
  - Up to 0.5% with bandgap calibration
- Security Features
  - Read-only unique ID for digital rights management algorithms
  - Secure boot using 128-bit AES hardware decryption
  - SHA-1 hashing hardware
  - Customer-programmed (OTP) 128 bit AES key is never visible to software.
- External Memory Interface (EMI)
  - Provides memory-mapped (load/store) access to external memories
  - Supports the following types DRAM:
    - 1.8-V Mobile DDR
    - Standard 2.5V DDR1
- Wide Assortment of External Media Interfaces
  - Up to four NAND flash memories with hardware management of device interleaving
  - High-speed MMC, secure digital (SD)
  - Hardware Reed-Solomon Error Correction Code (ECC) engine offers industry-leading protection and performance for NANDs.
  - Hardware BCH ECC engine allowing for up to 20-bit correction and programmable redundant area.
- Dual Peripheral Bus Bridges with 18 DMA Channels
  - Multiple peripheral clock domains save power while optimizing performance.

  - Direct Memory Access (DMA) with sophisticated linked DMA command architecture saves power and off-loads the CPU.
- Highly Flexible Display Controller
  - Up to 24-bit RGB (DOTCK) modes
  - Up to 24-bit system-mode including VSYNC and WSYNC modes.
  - Up to VGA (640x480) resolution at 60Hz LCD panel support
  - 8-bit data ITU-R BT.656 D1 digital video stream output mode (PAL/NTSC), with on-the-fly RGB to YCbCr color-space-conversion.
  - Flexible input formats
- Pixel Processing Pipeline (PXP)
  - Provides full path from color-space conversion, scaling, alpha-blending to rotation without intermediate memory access
  - Bi-linear scaling algorithm with cropping and letterboxing
  - Alpha-blend, BITBLT, color-keying
  - Memory efficient block-based rotation engine
  - Supports up to eight overlays
- Integrated TV-Out Support
  - Integrated PAL/NTSC TV-encoder fully pipelined to display controller's D1 resolution output stream
  - Integrated low-power 10-bit Video DAC (VDAC) for composite analog video output.
- Data Co-Processor (DCP)
  - AES 128-bit encryption/decryption
  - SHA-1 hashing
  - High-speed memory copy
- Three Universal Asynchronous Receiver-Transmitters (UARTs)
  - Two high-speed application UARTs operating up to 3.25 Mb/s with hardware flow control and dual DMA.
  - Debug UART operates at up to 115Kb/s using programmed I/O.
- I^2C Master/Slave
  - DMA control of an entire EEPROM or other device read/write transaction without CPU intervention
- Dual Synchronous Serial Ports (for SPI, MMC, SDIO, Triflash)
  - Up to 52MHz external SSP clock for all modes, including SPI
  - 1-bit, 4-bit and 8-bit MMC/SD/SDIO modes
  - Compliant with SDIO Rev. 2.0
  - SPI with single, dual and quad modes.
- Four-Channel 16-Bit Timer with Rotary Decoder
- Five-Channel Pulse Width Modulator (PWM)

- Real-Time Clock
  - Alarm clock can turn the system on.
  - Uses the existing 24-MHz XTAL for low cost or optional low power crystal (32.768 kHz or 32.0 kHz), customer-selectable via OTP.
- SPDIF Transmitter
- Dual Serial Audio Interface (SAIF), Three Stereo Pairs
  - Full-duplex stereo transmit and stereo receive operations
  - Cell phone baseband processor connection and external ADCs and DACs
  - Bluetooth hands-free connection
  - Analog I/O for peripheral bus breakouts
  - I^2S, left-justified, right-justified, and non-standard formats
- Customer-Programmable One-Time-Programmable (OTP) ROM via Integrated eFuse Block
  - Resistor-less boot mode selection
  - 128-bit boot mode crypto key
  - Boot mode specification of NAND characteristics for device that the customer is soldering to the board. This means no more costly delays waiting for new device support in the boot ROM.
  - Fully software-programmable and accessible
- Flexible I/O Pins
  - All digital pins have drive-strength controls as described in Section 37.2.2.1, “Pin Drive Strength Selection.”
  - Most non-EMI digital pins have general-purpose input/output (GPIO) mode.
- Offered in 128-Pin Low-Profile Quad Flat Pack (LQFP), and 169-Pin Ball Grid Array (BGA)

1.2 i.MX23 Product Features

The i.MX23 offers long battery life, minimal external components through integration, high processing performance, and excellent software development and debug support. The i.MX23 is especially suited for multi-media applications requiring audio/video decode and rich display support. These requirements are achieved via the high-performance CPU, pixel processing and integrated display and TV-Out hardware.

Figure 1-1 shows a block diagram of a typical system based on the i.MX23.

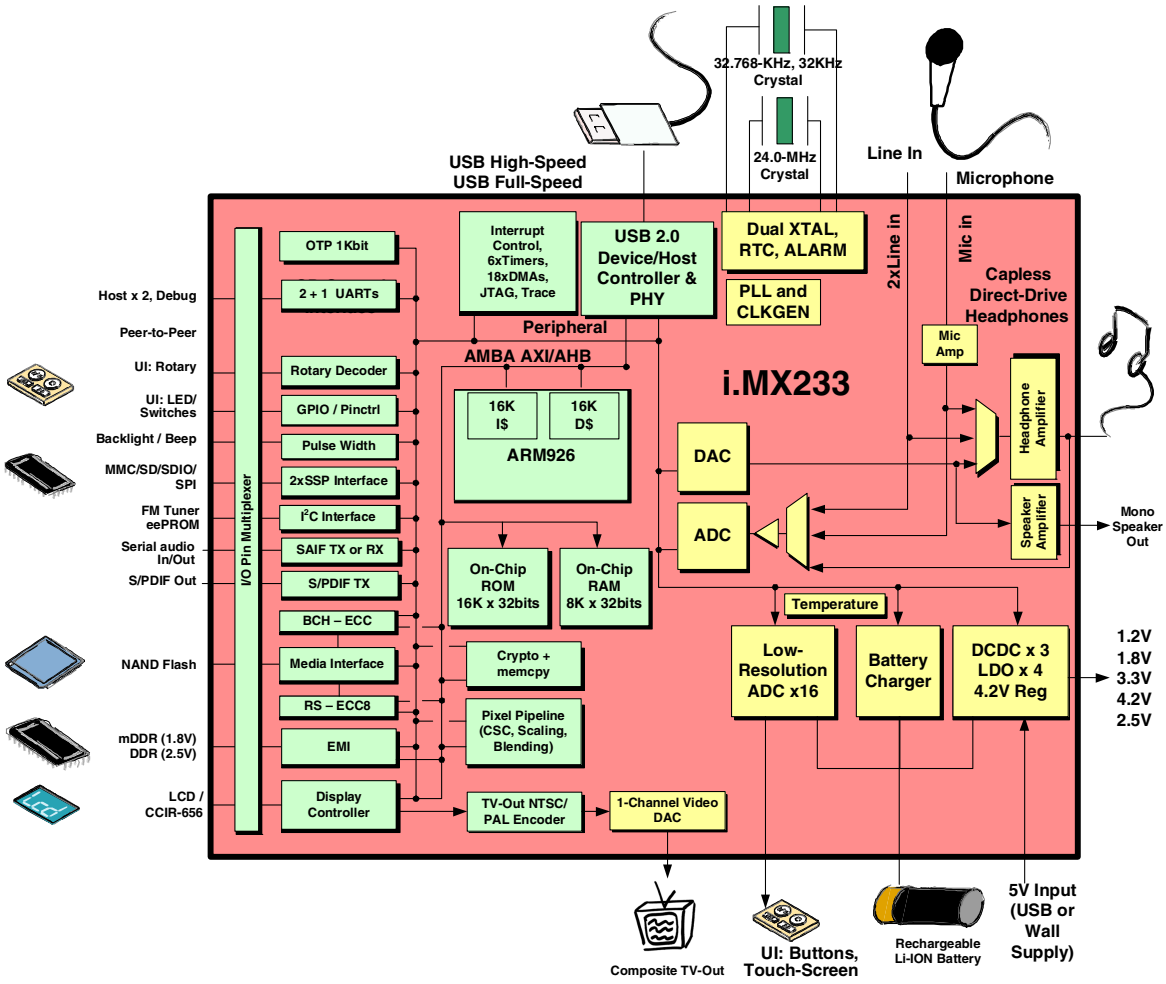


Figure 1-1. System Block Diagram

The i.MX23 features low power consumption to enable long battery life in portable applications. The integrated power management unit includes a high-efficiency, on-chip DC-DC converter. The power management unit also includes an intelligent battery charger for Li-Ion cells and is designed to support adaptive voltage control (AVC), which can reduce system power consumption by half. AVC also allows the chip to operate at a higher peak CPU operating frequency than typical voltage control systems. The DC-DC converters and the clock generator can be reprogrammed on-the-fly to trade off power versus performance dynamically.

To provide the maximum application flexibility, the i.MX23 integrates a wide range of I/O ports. It can efficiently interface to nearly any type of flash memory, serial peripheral bus, or LCD. It is also ready for advanced connectivity applications such as Bluetooth and WiFi via its integrated 4-bit SDIO controller and high-speed (3.25 Mb/s) UARTs.

The i.MX23 also integrates an entire suite of analog components, including a high-resolution audio codec with headphone amplifier, 16-channel 12-bit ADC, 10-bit Video DAC, Mono Speaker Amplifier, high-current battery charger, linear regulators for 5-V operation, high-speed USB Host PHY, and various system monitoring and infrastructure systems.

An ARM 926 EJ-S CPU with 32 Kbytes of on-chip SRAM and an integrated memory management unit provides the processing power needed to support advanced features such as audio cross-fading, as well as still picture and video decoding.

Execution always begins in on-chip ROM after reset, unless overridden by the debugger. A number of devices are programmed only at initialization or application state change, such as DC-DC converter voltages, clock generator settings, etc. Certain other devices either operate in the crystal clock domain or have significant portions that operate in the crystal clock domain, e.g., ADC, DAC, PLL, etc. These devices operate on a slower speed asynchronous peripheral bus. Write posting in the ARM core, additional write post buffering in the peripheral AHB, and set/clear operations at the device registers make these operations efficient.

1.2.1 ARM 926 Processor Core

The on-chip RISC processor core is an ARM, Ltd. 926EJ-S. This CPU implements the ARM v5TE instruction set architecture. The ARM9EJ-S has two instructions sets, a 32-bit instruction set used in the ARM state and a 16-bit instruction set used in Thumb state. The core offers the choice of running in the ARM state or the Thumb state or a mix of the two. This enables optimization for both code density and performance. The ARM CPU is described in Chapter 3, “ARM CPU Complex.”

The ARM RISC CPU is the central controller for the entire i.MX23 SOC, as shown in Figure 1-2. The ARM 926 core includes two AHB masters:

- AHB1—Used for instruction fetches
- AHB2—Used for data load/stores, page table accesses, DMA traffic, etc.

1.2.2 System Buses

The i.MX23 uses buses based on ARM’s Advanced Microcontroller Bus Architecture (AMBA) for the on-chip peripherals. The AMBA2 specification outlines two bus types: AHB and APB. The AMBA3 specification additionally outlines the AXI fabric.

- AXI is the highest-performance AMBA bus that supports de-coupled R/W channels, multiple outstanding transactions, and out-of-order data capability. This leads to higher performance and more efficient use of external memory.
- AHB is a higher-performance bus that supports multiple masters such as the CPU and DMA controllers.
- The APB is a lower-speed peripheral bus.

As shown in Figure 1-2, the i.MX23 uses a three-layer AHB, a high-performance AXI segment and two APBs: APBH and APBX. The APB buses are enhanced to include byte-write capability.

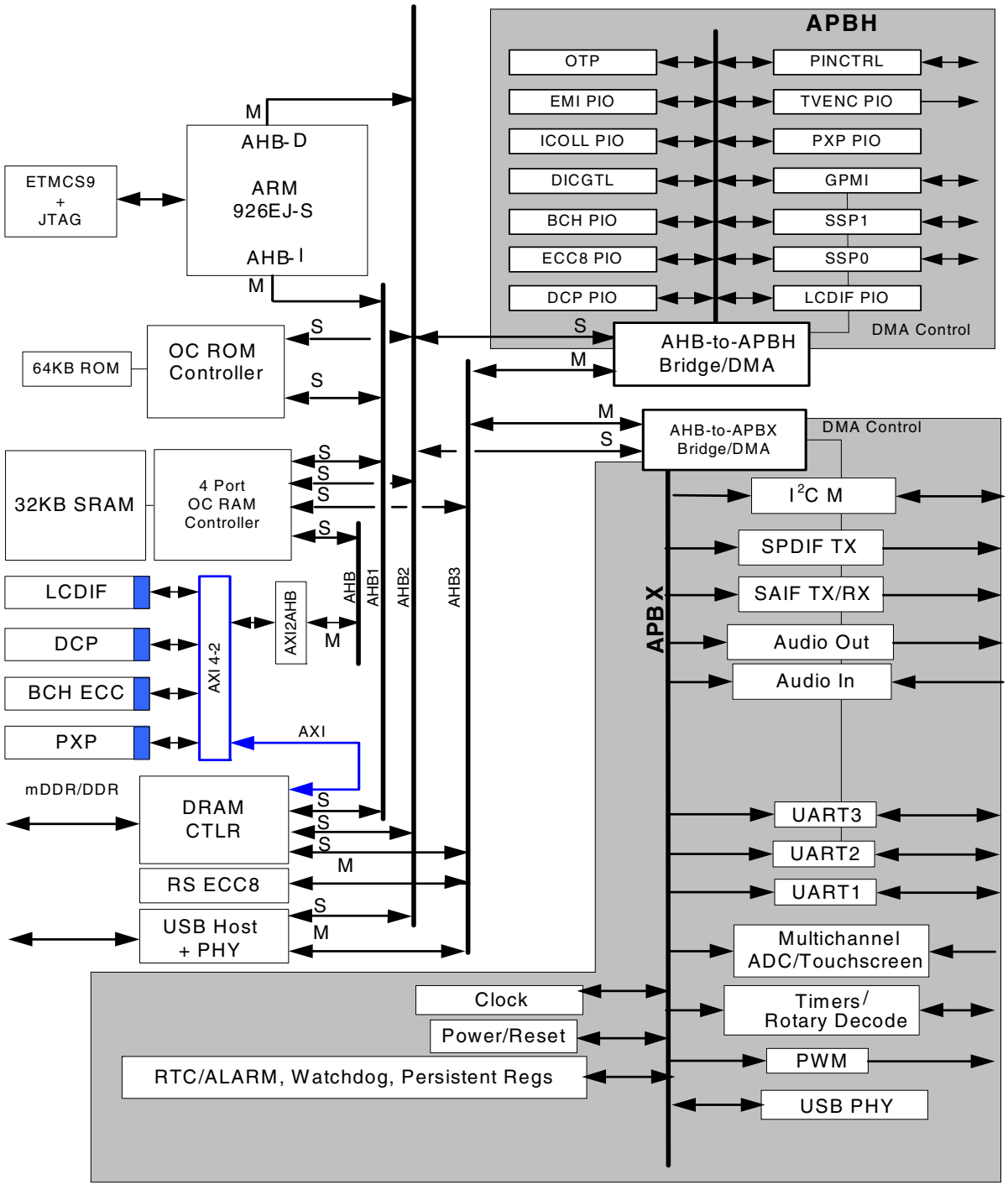


Figure 1-2. i.MX23 SoC Block Diagram

1.2.2.1 AXI Bus

The AXI bus-segment on i.MX23 provides several high-bandwidth/performance-critical peripherals a tightly-coupled and efficient interface to Port-0 of the external memory controller. The peripherals are as follows:

- DCP (Crypto/Memcpy)
- PXP (Pixel Processing Pipeline)
- LCDIF (Display Controller)
- BCH-ECC Engine

This connection also allows for a path to the On-Chip RAM and EMI as shown in the figure. The AXI bus-segment allows for each peripheral to issue multiple outstanding transactions allowing for higher-performance and more efficient memory bus usage.

1.2.2.2 AHB Bus

The AHB is the main high-performance system bus and is implemented in three layers, as follows:

- Layer 1 (AHB1)—CPU instruction access to OCRAM, OCROM, EMI
- Layer 2 (AHB2)—CPU data access to OCRAM, OCROM, all bridges, USB slaves, EMI
- Layer 3 (AHB3)—APBH DMA, APBX DMA, RS-ECC8 and USB masters, EMI

The ARM926 instruction bus (AHB1) is a single-master layer, as is the ARM926 data bus (AHB2). The other two layers have multiple masters, as shown in Figure 1-2.

The ARM926 data bus connects to the all slaves in the system, including RAMs, ROMs, bridge slaves, and USB slaves. The APB peripherals can act as AHB slaves through the AHB-APB bridge. The AHB has seven slaves:

- USB slave
- On-chip RAM
- On-chip ROM
- Two APB bridges
- External memory

Each layer of the AHB bus allows one active transaction at a time. A transaction is initiated by a master, controlled by an arbiter, and serviced by the slave at the corresponding address. A transaction can be as short as a single byte, or as long as a CPU cache line (32 bytes). For the USB, a transaction can be much longer, up to 512 bytes on its AHB layer. For more information, refer to the AMBA 2.0 specification.

1.2.2.3 APB Buses

There are two APB peripheral buses on the i.MX23:

- The APBH bus runs completely synchronously to the AHB’s HCLK.

- The APBX bus runs in the independent XCLK clock domain that can be slowed down significantly for power reduction.

The “H” in APB*H* denotes that the APBH is synchronous to HCLK, as compared to APB*X*, which runs on the crystal-derived XCLK. Figure 1-2 shows which blocks are controlled by which bus.

See Section 1.2.7, “DMA Controller,” for more information about these peripheral buses and their DMA bridges.

1.2.3 On-Chip RAM and ROM

The i.MX23 includes 8Kx32-bit on-chip RAM implemented as a single physical bank with four AHB slave ports. Each access to the on-chip RAM requires at a minimum two HCLK cycles.

The i.MX23 also includes 16Kx32-bit words of on-chip mask-programmable ROM. The ROM contains intitialization code written by Freescale. to handle the initial boot and hardware initialization. Software in this ROM offers a large number of boot configuration options, including manufacturing boot modes for burn-in and tester operation.

Other boot modes are responsible for loading application code from off-chip into the on-chip RAM. It supports initial program loading from a number of sources:

- NAND flash devices
- I^2C master mode from EEPROM devices
- USB recovery mode

At power-on time, the first instruction executed by the ARM core comes from this ROM. The reset boot vector is located at 0xFFFF0000. The on-chip boot code includes a firmware recovery mode. If the device fails to boot from NAND flash, or hard drive, for example, the device will attempt to boot from a PC host connected to its USB port.

The on-chip RAM and ROM run on the AHB HCLK domain.

Figure 1-3 shows the memory map for the AHB2 devices.

1.2.4 External Memory Interface

The i.MX23 supports off-chip DRAM storage via the EMI controller, which is connected to the four internal AHB/AXI busses.

The EMI supports multiple external memory types, including:

- 1.8-V Mobile DDR
- Standard 2.5V DDR1

The DRAM controller supports up to two external chip-select signal for the i.MX23 platform. Programmable registers within the DRAM controller allow great flexibility for device timings, low-power operation, and performance tuning. Note the diffences between the two package options:

- The 128-pin LQFP has 1 chip enable.
- The 169-pin BGA has 2 chip enables.

The EMI uses two primary clocks: the AHB bus HCLK and the DRAM source clock EMI_CLK. The maximum specified frequencies for these two clocks can be found in Chapter 2, “Characteristics and Specifications. The memory controller operates at frequencies that are asynchronous to the rest of the i.MX23.

The EMI consists of two major components:

- DRAM controller
- Delay compensation circuitry (DCC)

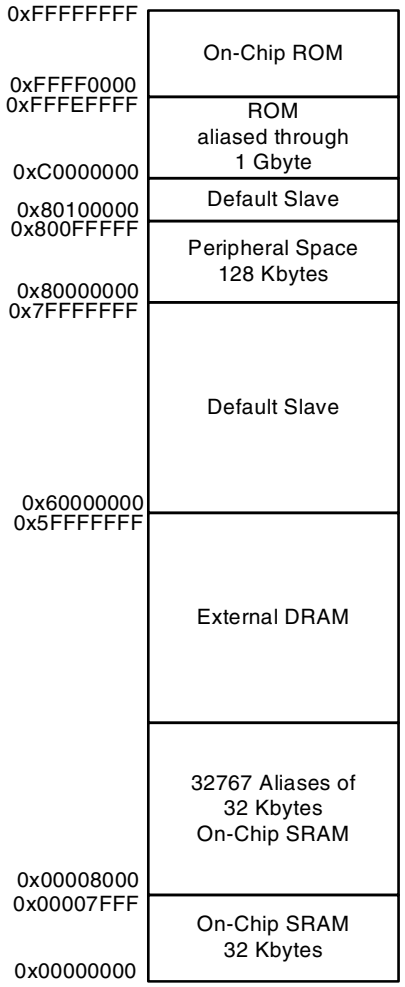


Figure 1-3. Physical Memory Map

1.2.5 On-Chip One-Time-Programmable (OCOTP) ROM

The i.MX23 contains 1024 bits (1Kb) of OTP ROM. The OTP is segmented into four distinct physical banks. Each bank is further divided logically into eight 32-bit words. The OTP serves several functions:

- Housing of hardware and software capability bits (copied into shadow registers)
- Housing of Freescale operations and unique-ID fields.
- Housing the customer-programmable cryptography key
- Four words for customer general use
- A 32-bit word is dedicated to controller read and write locking of the various OTP regions (copied into a shadow register)
- Storage of various ROM configuration bits

Access to the OTP is done through a memory-mapped APBH slave interface. Each of the 32 words is memory-mapped on APBH for the purposes of reading (requires a bank-opening sequence). Writing to the OTP is done through an address and data interface, where software provides the OTP word number (one of 32) and a programming mask.

For more information, see Chapter 7, “On-Chip OTP (OCOTP) Controller.”

1.2.6 Interrupt Collector

The i.MX23 contains a 128-bit vectored interrupt collector for the CPU’s IRQ input and a separate non-vectored interrupt collection mechanism for the CPU’s FIQ input. Each interrupt can be assigned to one of four levels of priority. The interrupt collector supports nesting of interrupts that preempt an interrupt service routine running at a lower priority level. Each of the 128 interrupts is assigned its own 32-bit programming register and can be set for HW source IRQ, SW source IRQ or HW source FIQ.

The interrupt collector is described in Chapter 5, “Interrupt Collector.”

1.2.7 DMA Controller

Many peripherals on the i.MX23 use direct memory access (DMA) transfers. Some peripherals, such as the USB controller, make highly random accesses to system memory for a large number of descriptor, queue heads, and packet payload transfers. This highly random access nature is supported by integrating a dedicated DMA into the USB controller and connecting it directly to the high-speed AHB bus.

Similarly, the RS-ECC8 error correction engine, the DCP (crypto/memcpy), BCH-ECC and LCD controller devices contain their own bus masters to allow for more random accesses to system memory.

Other peripherals have a small number of highly sequential transactions, for example the ADC or DAC streams, SPDIF transmitter, etc. These devices share a centralized address generation and data transfer function that allows them to share a single shared master on the AHB.

As mentioned previously, there are two AMBA peripheral buses on the i.MX23:

- The APBH bus runs completely synchronously to the AHB's HCLK.
- The APBX bus runs in an independent XLKC clock domain that can be slowed down significantly for power reduction.

See Chapter 10, "AHB-to-APBH Bridge with DMA," and Chapter 11, "AHB-to-APBX Bridge with DMA," for more detailed information. Note that the AHB HCLK can run up to 133 MHz.

The two bridge DMAs are controlled through linked DMA command lists. The CPU sets up the DMA command chains before starting the DMA. The DMA command chains include set-up information for a peripheral and associated DMA channel. The DMA controller reads the DMA command, writes any peripheral set up, tells the peripheral to start running and then transfers data, all without CPU intervention. The CPU can add commands to the end of a chain to keep data moving without interventions.

The linked DMA command architecture offloads most of the real-time aspects of I/O control from the CPU to the DMA controller. This provides better system performance, while allowing longer interrupt latency tolerances for the CPU.

1.2.8 Clock Generation Subsystem

The i.MX23 uses several different clock domains to provide clocks to the various subsystems, as shown in Figure 4-1. These clocks are either derived from the 24-MHz crystal or from the integrated high-speed PLL. The PLL output is fixed at 480 MHz.

More details about the system clock architecture can be found in Chapter 4, "Clock Generation and Control."

The system includes a real-time clock that can use either the 24-MHz system crystal or an optional low power crystal oscillator running at either 32.768 kHz or 32.0 kHz (customer-configurable via OTP). An integrated watchdog reset timer is also available for automatic recovery from errant code execution.

See Chapter 23, "Real-Time Clock, Alarm, Watchdog, Persistent Bits," for more information about these features.

1.2.9 Power Management Unit

The i.MX23 contains a sophisticated power management unit (PMU), including an integrated DC-DC converter, four linear regulators and a regulated 4.2V output. The PMU can operate from a Li-Ion battery using the DC-DC converter or a 5-V supply using the linear regulators and can automatically switch between them without interrupting operation. The PMU includes circuits for battery and system voltage brownout detection, as well as on-chip temperature, digital speed, and process monitoring.

The integrated PMU converter can be used to provide programmable power for the device as well as the entire application on up to five rails:

- VDDIO (nominal 3.3V) – DC-DC or linear-regulator from 5V
- VDDD (nominal 1.2V) – DC-DC or linear-regulator from VDDA
- VDDA (nominal 1.8V) – DC-DC or linear-regulator from VDDIO
- VDDM (nominal 2.5V) – linear-regulator from VDDIO
- VDD4P2 (nominal 4.2V) – when connected to 5V source

The 4.2V regulated output also allows for programmable current limits:

- Total load plus battery charge current (5V Limit)
- Battery Charge current
- Load current – for both on-chip and off-chip circuits

The 4.2V circuit is capable of adjusting distribution of current supply between the load and the battery-charger depending on programmed current-limits and load conditions. For example, when charging the battery, and exceeding the 5V current limit, the 4.2V regulator will steal current from the battery-charger circuit and divert it to the load circuit.

The converter can be configured to operate from standard Li-Ion battery chemistries up to 4.2 volts. These converters use off-chip reactive components (L/C) in a pulse-width or frequency-modulated DC-DC converter.

The real-time clock includes an alarm function that can be used to “wake-up” the DC-DC converters, which will then wake up the rest of the system.

The power subsystem is described in Chapter 32, “Power Supply.”

1.2.10 USB Interface

The chip includes a high-speed Universal Serial Bus (USB) version 2.0 controller and integrated USB transceiver macrocell interface (UTMI) PHY. The i.MX23 device interface can be attached to USB 2.0 hosts and hubs running in the USB 2.0 high-speed mode at 480 Mbits per second. It can be attached to USB 2.0 full-speed interfaces at 12 Mbits per second.

The USB controller and integrated PHY support high-speed Host modes for peer-to-peer file interchange. The i.MX23 has a high-current PWM channel that can be used with low-cost external components to generate up to 8 mA of 5 volts on the Host VBUS for Host session initiation. The USB controller can also be configured as a high-speed host.

The USB subsystem is designed to make efficient use of system resources within the i.MX23. It contains a random-access DMA engine that reduces the interrupt load on the system and reduces the total bus bandwidth that must be dedicated to servicing the five on-chip physical endpoints.

It is a dynamically configured port that can support up to five endpoints, each of which may be configured for bulk, interrupt, or isochronous transfers. The USB configuration information is read from on-chip memory via the USB controller's DMA.

See Chapter 8, "USB High-Speed Host/Device Controller," and Chapter 9, "Integrated USB 2.0 PHY," for more information.

1.2.11 General-Purpose Media Interface (GPMI)

The chip includes a general-purpose media interface (GPMI) controller that supports NAND devices (all packages).

The NAND flash interface provides a state machine that provides all of the logic necessary to perform DMA functions between on-chip or off-chip RAM and up to four NAND flash devices. The controller and DMA are sophisticated enough to manage the sharing of a single 16 bit wide data bus among four NAND devices, without detailed CPU intervention. This allows the i.MX23 to provide unprecedented levels of NAND performance.

The general-purpose media interface can be described as two fairly independent devices in one. The three operating modes are integrated into one overall state machine that can freely intermix cycles to different device types on the media interface. There are four chip selects on the media interface. Each chip select can be programmed to have a different type device installed.

The GPMI pin timings are based on a dedicated clock divider from the PLL, allowing the CPU clock divider to change without affecting the GPMI.

See Chapter 13, "General-Purpose Media Interface (GPMI)," for more information.

1.2.12 Hardware Acceleration for ECC for Robust External Storage

The hardware ECC accelerator provides a forward error-correction function for improving the reliability of various storage media that may be attached to the i.MX23. Modern high-density NAND flash devices presume the existence of forward error-correction algorithms to correct some soft and/or hard bit errors within the device, allowing for higher device yields and, therefore, lower NAND device costs.

The i.MX23 contains two separate Error Correction Code (ECC) hardware engines implemented the following algorithms:

- Reed-Solomon – Provides 4 or 8 bits/symbol correction (RS-ECC8). This is the same engine found in previous SoC products.
- Bose Ray-Choudhury Hocquenghem – Provides up to 20-bits correction (BCH-ECC)

Both engines are tightly coupled to the GPMI and are for mutually exclusive use with completely separate programming models and DMA structures. The BCH engine supersedes the RS-ECC8 except in

allowing for backwards compatibility of legacy NAND software drivers written specifically for the RS-ECC8.

1.2.12.1 Reed-Solomon ECC Engine

The RS-ECC module consists of two different error correcting code processors:

- Four-symbol error correcting (9 bits/symbol) Reed-Solomon encoder/decoder
- Eight-symbol error correcting (9 bits/symbol) Reed-Solomon encoder/decoder

The Reed-Solomon modes are used for storage elements that have a higher native defect probability, such as MLC NAND. It can correct up to four 9-bit symbols over a 512-byte block in a 2-Kbyte paged device or up to eight 9-bit symbols over a 512-byte block in a 4-Kbyte paged device.

Both of these error correction encoder/decoders use DMA transfers to move data from system memory completely in parallel with the CPU performing other useful work.

For storage read transfers, the ECC8 controller uses its AHB bus master to transfer data directly to system memory. In addition, the ECC8 automatically corrects errors in the read data buffers in system memory without CPU assistance.

The ECC8 includes one more significant enhancement, namely, it provides four-symbol error correction for the 9 or 16 byte metadata stored in the redundant area of the NAND device.

See Chapter 14, “8-Symbol Correcting ECC Accelerator (ECC8),” for more information.

1.2.12.2 Bose Ray-Choudhury Hocquenghem ECC Engine

The Bose, Ray-Chaudhuri, Hocquenghem (BCH) Encoder and Decoder module is capable of correcting from 2 to 20 single bit errors within a block of data no larger than about 900 bytes (512 bytes is typical) in applications such as protecting data and resources stored on modern NAND flash devices. The correction level in the BCH block is programmable to provide flexibility for varying applications and configurations of flash page size. The design can be programmed to encode protection of 2, 4, 8, 10, 12, 14, 16, 18, or 20 bit errors when writing flash and to correct the corresponding number of errors on decode. The correction level when decoding MUST be programmed to the same correction level as was used during the encode phase.

BCH-codes are a type of block-code, which implies that all error-correction is performed over a block of N-symbols. The BCH operation will be performed over GF(2^{13} = 8192), which is the Galois Field consisting of 8191 one-bit symbols. BCH encoding (or encode for any block-code) can be performed by two algorithms: systematic encoding or multiplicative encoding. Systematic encoding is the process of reading all the symbols which constitute a block, dividing continuously these symbols by the generator polynomial for the GF(8192) and appending the resulting ***t*** parity symbols to the block to create a BCH codeword (where ***t*** is the number of correctable bits).

The BCH sits on the AXI fabric with close coupling to both the GPMI and external memory controller.

See Chapter 15, “20-BIT Correcting ECC Accelerator (BCH).”

1.2.13 Data Co-Processor (DCP)—Memory Copy, Crypto, and Color-Space Converter

The i.MX23 SOC contains a data co-processor consisting of four virtual channels. Each channel is essentially a memory-to-memory copy engine. The linked list control structure can be used to move byte-aligned blocks of data from a source to a destination. In the process of copying from one place to another, the DCP can be programmed to encrypt or decrypt the block using AES-128 in one of several chaining modes. An SHA-1 hash can be calculated as part of the memory-copy operation.

See Chapter 16, “Data Co-Processor (DCP),” for more information.

1.2.14 Mixed Signal Audio Subsystem

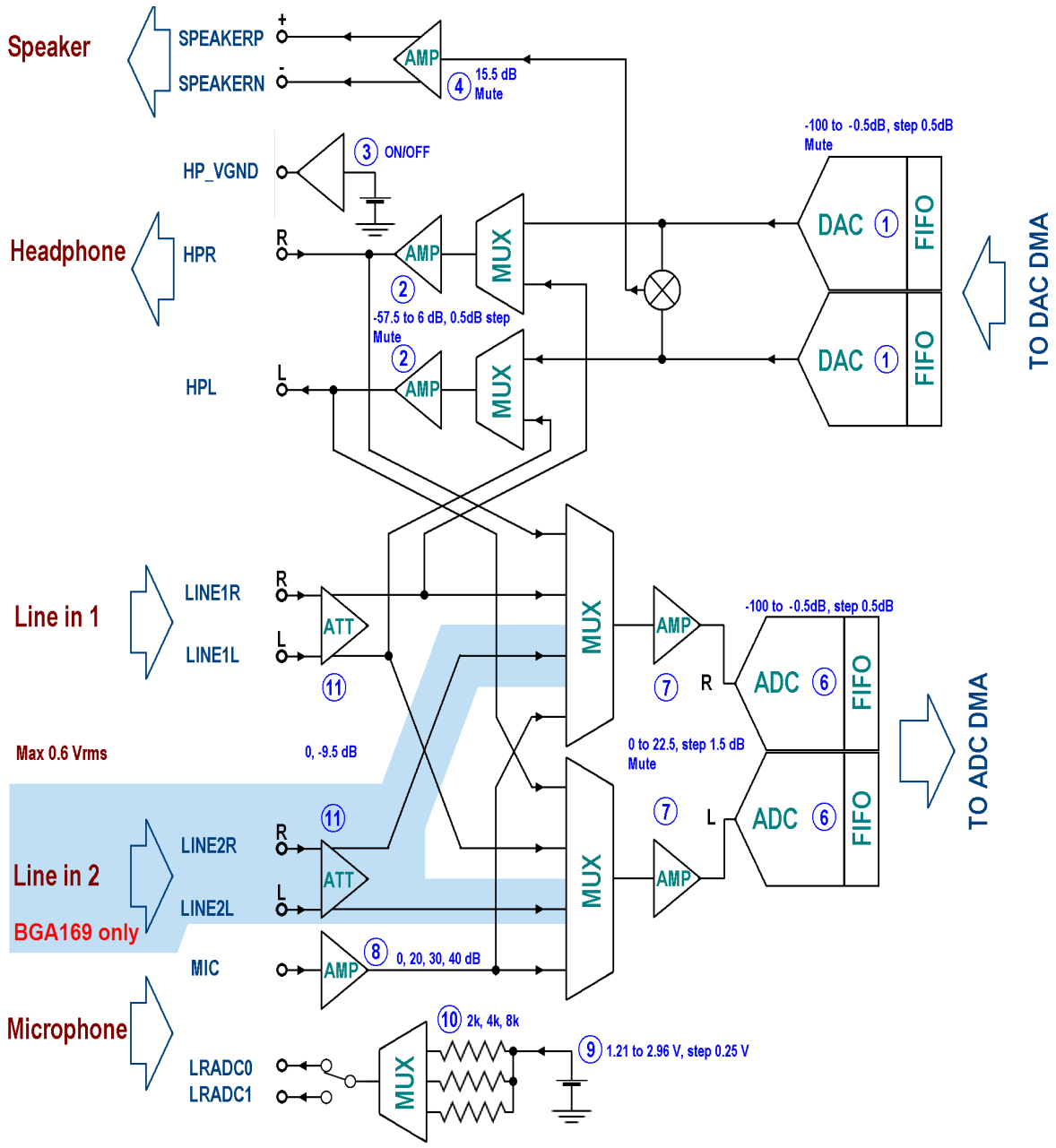
The i.MX23 contains an integrated high-quality mixed signal audio subsystem, including high-quality sigma delta D/A and A/D converters, as shown in Figure 1-4.

The chip includes a low-noise headphone driver that allows it to directly drive low-impedance (16Ω) headphones. The direct drive, or “capless” mode, removes the need for large expensive DC blocking capacitors in the headphone circuit. The headphone power amplifier can detect headphone shorts and report them via the interrupt collector. A digitally programmable master volume control allows user control of the headphone volume. Use of the headphone amplifier volume control is recommended as the digital control may reduce SNR performance. Annoying clicks and pops are eliminated by zero-crossing updates in the volume/mute circuits and by headphone driver startup and shutdown circuits.

The microphone circuit has a mono-to-stereo programmable gain pre-amp and an optional microphone bias generator.

Also integrated is a class A-B mono speaker amplifier which must be powered from a sufficiently high-enough current 4.2V source such as the battery. The speaker amplifier can support up to 2W rms of output assuming a 4.2V supply and a 4Ω speaker load.

These features are described in Chapter 28, “AUDIOIN/ADC,” and Chapter 29, “AUDIOOUT/DAC.”



Notes:
1. HW_AUDIOOUT_DACVOLUME: Digital volume control. -100 dB to -0.5 dB in 0.5 dB steps.
2. HW_AUDIOOUT_HPVOL: Analog volume control. -57.5 dB to 6 dB in 0.5 dB steps.
3. HW_AUDIOOUT_PWDN: Enable capless headphone common amplifier.
4. HW_AUDIOOUT_SPEAKERCTRL: Analog control for speaker amplifier, fixed gain of 9.5 dB from each DAC, 15.5dB total.
5. HW_AUDIOIN_ADCVOLUME: Digital volume control. -100 dB to -0.5 dB in 0.5 dB steps.
6. HW_AUDIOIN_ADCVOL: Analog volume control that controls the ADC gain block. 0 dB to 22.5 dB gain in 1.5 dB steps.
7. HW_AUDIOIN_MICLINE_MICGAIN: Analog volume control that controls the microphone amplifier. 0, 20, 30, or 40 dB gain.
8. HW_AUDIOOUT_MICLINE_MIC_BIAS: Mic bias voltage. 1.21V to 2.96V in 0.25V steps.
9. HW_AUDIOOUT_MICLINE_MIC_RESISTOR, HW_AUDIOOUT_MICLINE_MIC_SELECT.
10. HW_AUDIOIN_MICLINE_DIVIDE_LINE1/2.

Figure 1-4. Mixed Signal Audio Elements

1.2.15 Master Digital Control Unit (DIGCTL)

The master digital control unit (DIGCTL) provides control registers for a number of blocks that do not have their own AHB or APB slaves, notably the on-chip RAM and on-chip ROM controllers. In addition, it provides control registers for the DRAM controller output clock shifting. It also provides several security features, including an entropy register, as well as the JTAG shield.

See Chapter 6, “Digital Control and On-Chip RAM,” for more information.

1.2.16 Synchronous Serial Port (SSP)

The i.MX23 SOC contains two integrated synchronous serial ports, SSPs. Each SSP supports a wide range of synchronous serial interfaces, including:

- 1-bit, 4-bit, or 8-bit high-speed MMC/SD/SDIO
- Motorola (1-bit) and Winbond (1, 2 and 4-bit) SPI with up to 3 slave selects
- TI SSI

Each SSP has a dedicated DMA channel and a dedicated clock divider from the PLL.

See Chapter 21, “Synchronous Serial Ports (SSP),” for more information about these features.

1.2.17 I^2C Interface

The chip contains a two-wire SMB/I^2C bus interface. It can act as a master on the SMB interface. The on-chip ROM supports boot operations from I^2C EEPROMs.

See Chapter 25, “I2C Interface,” for more information.

1.2.18 General-Purpose Input/Output (GPIO)

The i.MX23 contains 95 GPIO pins in the 169-pin package and 64 GPIO pins in the 128-pin package. Most digital pins (except for EMI pins) that are available for specific functions are also available as GPIO pins if they are not otherwise used in a particular application.

See Chapter 37, “Pin Control and GPIO,” for more information

1.2.19 Display Processing

The i.MX23 display processing and output consists of four distinct modules as shown in Figure 1-5. These are:

- Pixel Processing Pipeline - dedicated AXI bus master.
- Display Controller (LCDIF) - dedicated AXI bus master.
- PAL/NTSC TV-Encoder - direct feed from LCDIF output
- 10-bit Video DAC for analog composite output - direct feed from TV-Encoder

This allows for all post video-decode pixel processing to be handled in hardware with minimal CPU intervention. Multiple pixel formats and display configurations are also supported.

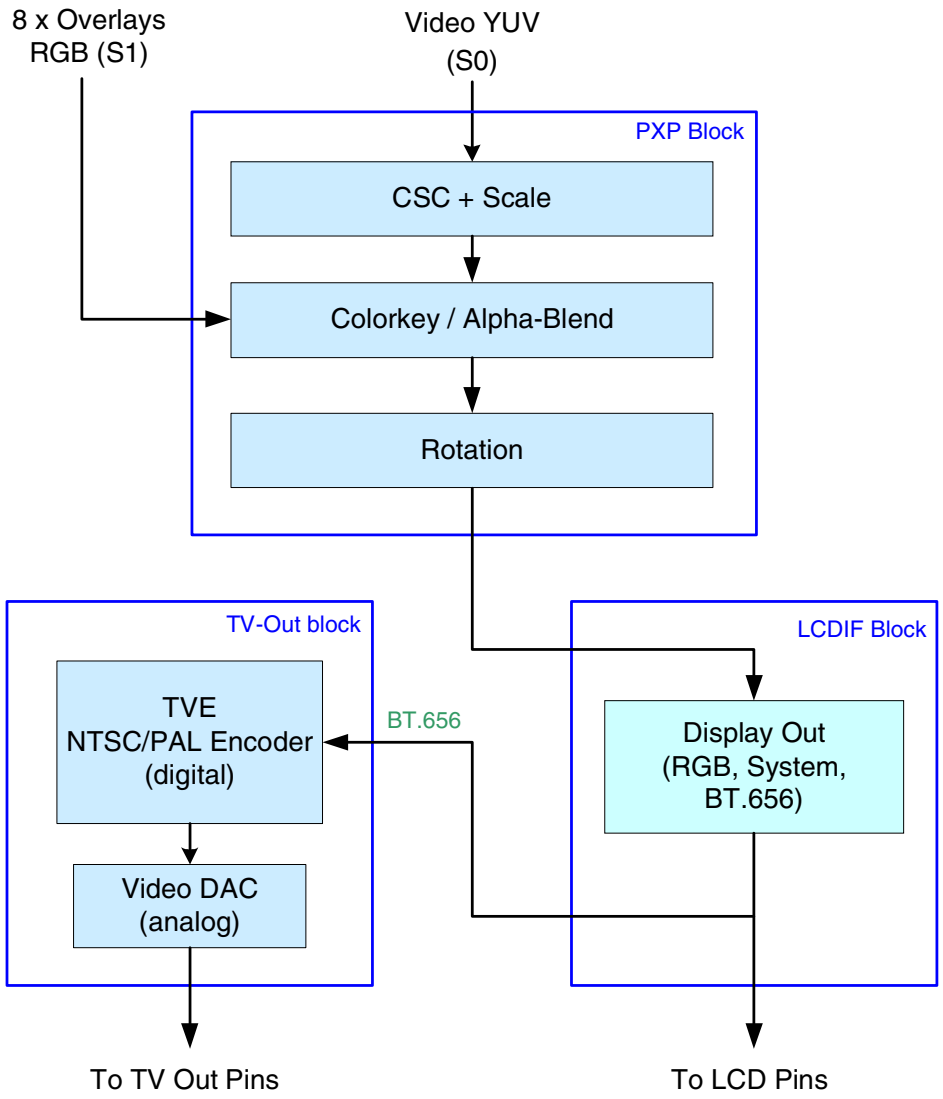


Figure 1-5. Display Processing Sub-System

1.2.19.1 Display Controller / LCD Interface (LCDIF)

The i.MX23 Display Controller (LCDIF) includes:

- AMBA AXI master mode allows for high-performance operation from external memory. This also includes an increase to a 128x32-bit internal latency buffer which features an under-flow recovery mechanism.
- Supports 24-bit full color parallel RGB (DOTCK) mode. Able to drive up to VGA (640x480) full color displays at refresh rates up to 60Hz.
- Supports full 24-bit system mode (8080/6080/VSYNC/WSYNC). Read-mode is not supported.

- ITU-R BT.656 compliant D1 digital video output mode with on-the-fly RGB to YCbCr color-space-conversion. This output also feeds the integrated TV-Encoder
- Supports wide variety of input and output formats allowing for conversion between input and output (e.g., RGB565 input to RGB888 output). Also supports packed pixel formats.

See Chapter 18, "LCD Interface (LCDIF)," for more information.

1.2.19.2 Pixel Processing Pipeline (PXP)

The PXP performs all necessary post display frame pre-processing in hardware with minimal memory overhead. In a video-centric system such as the i.MX23, this allows for the CPU to have maximum processing bandwidth for video-decode operation. The PXP operation and features can be described as follows:

- The *background* image (e.g. decoded video frames) is read from external memory into separate Y/U/V buffers as 8x8 pixel macroblocks. These buffers are then fed into a color-space converter (e.g. YUV to RGB) followed by the scaling engine which utilizes an advanced bi-linear weighted scaling algorithm. The scaling operation is defined in terms of the output image (via programmable offsets and cropping registers). The output of the scaler is fed into yet another internal buffer called S0. If the background image is already in the RGB color-space it is assumed to be scaled appropriately for the required output format and can thus be read directly into the internal S0 buffer. In order to maintain efficient use of external memory, only the relevant (visible) portion of the background image is fetched.
- The scaled RGB image (in the internal S0 buffer) can be blended with up to eight programmable *overlays*. The co-ordinates of the overlays can once again be described in terms of the resultant output image. Each overlay can have a either a global programmable opacity or a per-pixel resolution if constructed with ARGB color-space. In addition to this, each overlay can have a relative priority level such that when constructing the output image, the PXP only fetches the visible overlay in the current 8x8 macroblock. The overlays are fetched into the internal S1 buffer. Alpha blending is performed on the S0 and S1 buffers to generate the blended output into the internal S3 buffer. Other operations such as BITBLT and color-keying can also be performed at this stage.
- The final stage of the PXP operation is the rotator which can perform flips and 90, 180 and 270 rotations. The rotator operates on the 8x8 pixel macroblocks in the S3 buffer to maximize external memory fetch efficiency. It writes 8x8 macroblocks to external memory in this final stage.

It should be noted that the PXP supersedes all pixel operations of the DCP.

See Chapter 17, "Pixel Pipeline (PXP)," for more information on the PXP.

1.2.19.3 PAL/NTSC TV-Encoder

The PAL/NTSC TV-Encoder is part of the integrated TV-Out functionality of i.MX23. The encoder takes input directly from the LCDIF without intermediate memory access. In order to utilize the TV-Out path, the LCDIF must be configured to output the ITU-R BT.656/BT.601 D1 digital video stream mode. This stream is synchronized to the internal 108MHz clock of the TV-Encoder. After this point, the block

encodes the stream into a format suitable for the Video DAC. Before being sent to the video DAC, the output of the TV-Encoder is passed through a pixel interpolating filter which helps to lessen the requirements for off-chip video filtering.

See Chapter 19, “TV-Out NTSC/PAL Encoder.” for more information on the TV-Encoder.

1.2.19.4 Video DAC

The i.MX23 includes a fully integrated low-power 10-bit Video DAC which takes the direct output from the TV-Encoder to generate a compliant analog composite analog video signal (CVBS). Also supported are optional source termination and automatic jack detection (via interrupt) allowing the Video DAC to be enabled/disabled automatically.

See Chapter 20, “Video DAC,” for more information.

1.2.20 SPDIF Transmitter

The i.MX23 includes a Sony-Philips Digital Interface Format (SPDIF) transmitter. It supports sample rates independently from the A/D and D/A sample rates so that all three can run simultaneously. The SPDIF has a dedicated DMA channel. The SPDIF has its own clock divider from the PLL. See Chapter 30, “SPDIF Transmitter,” for more information.

1.2.21 Dual Serial Audio Interfaces

The BGA169 package of the i.MX23 includes two serial audio interfaces (SAIF), each with three stereo pairs. The pin multiplexing scheme for i.MX23 allows a stereo transmitter on one device and a stereo receiver to be connected to external devices, either D/A and A/D converters or to a host processor, such as a cell phone or BlueTooth controller.

See Chapter 31, “Serial Audio Interface (SAIF) (BGA169 Only).”

1.2.22 Timers and Rotary Decoder

An automatic rotary decoder function is integrated into the chip. Two digital inputs are monitored to determine which is leading and by how much. In addition, the hardware automatically determines the period for rotary inputs. There are four timers to provide timer functionality based on different clock inputs.

See Chapter 22, “Timers and Rotary Decoder,” for more information.

1.2.23 UARTs

Three UARTs, similar to a 16550 UART, are provided—two for application use and one for debug use. The application UARTs are a high-speed devices capable of running up to 3.25 Mbits per second with

16-byte receive and transmit FIFOs. The application UARTs supports DMA and flow control (CTS/RTS). The debug UART does not use DMA channels.

See Chapter 26, “Application UART,” and Chapter 27, “Debug UART,” for more information.

1.2.24 Low-Resolution ADC, Touch-Screen Interface, and Temperature Sensor

The LRADC provides 16 “physical” channels of 12-bit resolution analog-to-digital conversion. Only 8 “virtual” channels can be used at one time, but those 8 channels can be mapped to any of the 16 physical channels. Some physical channels have dedicated inputs:

- Channel 15—VDD5V
- Channel 14—Bandgap reference
- Channel 13—USB_DN
- Channel 12—USB_DP
- Channel 10 and 11—Reserved
- Channel 8 and 9—Internal temperature sensing
- Channel 7—Battery
- Channel 6—VDDIO

The USB_DN/DP inputs can only be sampled with the LRADC in non-USB mode (see HW_USBPHY_CTRL_DATA_ON_LRADC).

The remaining six channels are available for other uses and can be used for resistive button sense, touch-screens, or other analog input. Channels 0 and 1 have integrated current sources to drive external temperature monitor thermistors. Channels 2–5 have integrated drivers for resistive touch-screens. The LRADC provides typical performance of 12-bit no-missing-codes, 9-bit/~56dB SNR, and 1% absolute accuracy (limited by the bandgap reference).

See Chapter 33, “Low-Resolution ADC and Touch-Screen Interface,” for more information.

1.2.25 Pulse Width Modulator (PWM) Controller

The i.MX23 contains five PWM output controllers that can be used in place of GPIO pins. Applications include LED and backlight brightness control. Independent output control of each phase allows 0, 1, or high impedance to be independently selected for the active and inactive phases. Individual outputs can be run in lock step with guaranteed non-overlapping portions for differential drive applications.

See Chapter 24, “Pulse-Width Modulator (PWM) Controller,” for more information.

1.2.26 Real-Time Clock, Alarm, Watchdog, Persistent Bits

The i.MX23 supports real-time clock, alarm clock, watchdog reset, persistent bits and millisecond counter. The RTC system can be powered from the battery 5 V supply. The clock sources for these functions are selectable between 32 kHz, 32.768 kHz or 24 MHz crystals.

See Chapter 23, “Real-Time Clock, Alarm, Watchdog, Persistent Bits,” for more information.

Chapter 2
Characteristics and Specifications

This chapter describes the characteristics and specifications of the i.MX23 and includes sections on absolute maximum ratings, recommended operating conditions, and DC characteristics.

2.1 Absolute Maximum Ratings

Table 2-1. Absolute Maximum Ratings

Parameter	Min	Max	Units
Storage Temperature	–40	125	°C
Battery Pin - BATT, VDD4P2V	–0.3	4.242	V
5-Volt Source Pin - VDD5V (transient, t < 30ms, duty cycle < 0.05%)	-0.3	7.00	V
5-Volt Source Pin - VDD5V (static)	–0.3	6.00	V
PSWITCH (Note 1)	–0.3	VDDXTAL + 1.575	V
Analog Supply Voltage—VDDA	–0.3	2.10	V
Speaker Amplifier Supply Voltage—VDDS	–0.3	4.242	V
Digital Core Supply Voltage —VDDD	–0.3	1.575	V
Non-EMI Digital I/O Supply—VDDIO	–0.3	3.63	V
EMI Digital I/O Supply—VDDIO.EMI	–0.3	3.63	V
DC-DC Converter—DCDC_BATT (Note 2)	–0.3	BATT	V
Input Voltage on Any Digital I/O Pin Relative to Ground	–0.3	VDDIO+0.3	V
Input Voltage on USB_DP and USB_DN Pins Relative to Ground (Note 3)	–0.3	3.63	V
Input Voltage on Any Analog I/O Pin Relative to Ground	–0.3	VDDA+0.3	V

1 VDDIO can be applied to PSWITCH through a 10 kΩ resistor. This is necessary in order to enter the chip's firmware recovery mode. (The on-chip circuitry prevents the actual voltage on the pin from exceeding acceptable levels.)

2 Application should include a Schottky diode between BATT and VDD4P2.

3 USB_DN and USB_DP can tolerate 5V for up to 24 hours. Note that while 5V is applied to USB_DN or USB_DP, LRADC readings can be corrupted.

Table 2-2. Electro-Static Discharge Immunity

169-Pin BGA & 128-Pin LQFP Packages	Tested Level
Human Body Model (HBM)	2 kV
Charge Device Model (CDM)	500 V

2.2 Recommended Operating Conditions

Table 2-3. Recommended Power Supply Operating Conditions

Parameter	Min	Typ	Max	Units
Analog Core Supply Voltage—VDDA	1.62	-	2.10	V
Digital Core Supply Voltage—VDDD *Specification dependent on frequency.* (Notes **1, 4**)	1.00	-	1.55	V
Non-EMI Digital I/O Supply Voltage—VDDIO	2.90	-	3.575	V
EMI Digital I/O Supply Voltage—VDDIO.EMI	1.8	-	3.25	V
Battery / DCDC Input Voltage - BATT, DCDC_BATT (Note **2**)	2.6	-	4.242	V
Speaker Supply Voltage - VDDS	2.7	-	4.242	V
VDD5V Supply Voltage (5V current < 100 ma)	4.40	5.00	5.25	V
VDD5V Supply Voltage (5V current >= 100 ma)	4.75	5.00	5.25	V
Offstate Current (Note **3**): • 32-kHz RTC off, BATT = 4.2 V • 32-kHz RTC on, BATT = 4.2 V	 - -	 11 13.5	 30 30	 µA µA

1 For optimum USB jitter performance, VDDD = 1.35 V or greater.

2 This requires software to program the RTC_PERSIST0_SPARE<31:28> bits. Minimum without the software programming is 2.9V

3 When the real-time clock is enabled, the chip consumes additional current when in the OFF state to keep the crystal oscillator and the real-time clock running.

4 VDDD supply minimum voltage includes 75 mV guardband.

Table 2-4. Operating Temperature Conditions

Parameter	Min	Typ	Max	Units
Commercial Ambient Operating Temperature Range, T_A (Note **1, 2**)	–10	-	70	°C
Commercial Junction Temperature Range, T_J (Note **1, 2**)	–10	-	85	°C
Industrial Ambient Operating Temperature Range, T_A (Note **1, 2**)	–40	-	85	°C
Industrial Junction Temperature Range, T_J (Note **1, 2**)	–40	-	105	°C
Package Thermal Impedance, 128-Pin LQFP, Θ_{JA} (Note **3**)	-	-	43	°C/W
Package Thermal Impedance, 169-Pin BGA, Θ_{JA} (Note **3**)	-	-	44	°C/W

1 In most systems designs, battery and display specifications will limit the operating range to well within these specifications. Most battery manufacturers recommend enabling battery charge only when the ambient temperature is between 0° and 40°C. To ensure that battery charging does not occur outside the recommended temperature range, the player ambient temperature may be monitored by connecting a thermistor to the LRADC0 or LRADC1 pin on the i.MX23.

2 Maximum Ambient Operating Temperature may be limited due to on-chip power dissipation. $T_{A\ (MAX)} <= T_J - (\Theta_{JA} \times P_D)$ where:

- T_J = Maximum Junction Temperature
- Θ_{JA} = Package Thermal Impedance
- P_D = Total On-chip Power Dissipation = PSpeakerAmp + PVDD4P2 + PBatteryCharger + PDCDC + PLinearRegulators +PInternal. Note that depending on the application, some of these power dissipation terms may not appliy.
- PSpeakerAmp = Speaker Amp On-Chip Power Dissipation = ~1W (regardless of output amplitude)
- PVDD4P2 = VDD4P2 On-Chip Power Dissipation = (VDD5V - VDD4P2) x IDD4P2
- PBatteryCharger = Battery Charger On-Chip Power Dissipation = (VDD5V - BATT) x ICHARGE
- PDCDC = DC-DC Converter On-Chip Power Dissipation = (BATT x DCDC Input Current) x (1 - efficiency)
- PLinearRegulators = Linear Regulator On-Chip Power Dissipation = (VDD5V-VDDIO) x (IDDIO + IDDM + IDDA + IDDD) + (VDDIO - VDDM) x IDDM + (VDDIO - VDDA) x (IDDA + IDDD) + (VDDA - VDDD) x IDDD
- PInternal = Internal Digital On-Chip Power Dissipation = ~VDDD x IDDD

3 Assumes 4-layer PCB and still air. Actual thermal performance may vary based on board and enclosure composition and design.

Table 2-5. Recommended Analog Operating Conditions

Parameter	Min	Typ	Max	Units
Low Resolution ADC:				
• Input Impedance (CH0 - CH5)	>1	-	-	MΩ
• Absolute Accuracy		+/- 1.5		%
On-Die Temperature Sensor:				
• Absolute Accuracy		+/- 1.5		%
Microphone:				
• Full-Scale Input Voltage (MIC_GAIN=40 dB) (Note **4**, **5**)	0.0052	0.0055	0.0057	Vrms
• Input Resistance	75	100	125	kΩ
• Idle SNR (Note **8**)	59	66	-	dB FS
• THD+N at -3dBFS	-53	-63	-	dB
Line Inputs:				
• Full-Scale Input Voltage to ADC (Note **1, 4**, **5**)	0.52	0.54	0.56	Vrms
• Full-Scale Input Voltage to HP with 0dB Gain (Note **1, 4**)	0.52	0.59	0.61	Vrms
• Input Resistance (Line-to-Headphone mode) (Note **2**)	37.5	50	62.5	kΩ
• Input Resistance (Line-to-ADC mode) (Note **2**)	14.1	18.75	23.4	kΩ
• LineIn-to-HP SNR Idle Channel (Note **2**)	97	100	103	dB FS
• ADC SNR Idle Channel (Note **2**)	80	87	89	dB FS
• ADC –60 dB Dynamic Range (Note **2**)	80	87	89	dB FS
• ADC THD+N at -3dBFS	-73	-80	-	dB
Headphone:				
• Full-Scale Output Voltage (VDDA = 1.8 V, 16 Ω load) (Note **6**)	0.46	0.52	0.54	Vrms
• Output Resistance	0.1	0.3	0.5	Ω
• Crosstalk between Input Channels (16 Ω load) (Note **7**)	-65	-70	-	dB
• THD+N (16 Ω load) (Note **7, 9**)	-80	-84	-	dB
• THD+N (10 kΩ load) (Note **7, 9**)	-83	-86	-	dB
• DAC SNR Idle Channel (Note **2, 7, 8**)	95	97	-	dB FS
• DAC –60 dB Dynamic Range (Note **2, 7, 8**)	95	97	-	dB FS
• Output Frequency Response (20Hz - 20kHz, 1 kHz = 0dB)	-1	-	+1	dB FS
• Channel Balance (Level Difference between L-ch and R-ch)	-0.2	0.04	0.2	dB
Speaker Amplifier:				
• Full-Scale Output Voltage (VDDS = 4.2 V, 8 Ω load)	2.6	2.7	2.8	Vrms
• Output Offset Voltage	0	15	45	mV
• Maximum Out Power (VDDS = 4.2 V, 8 Ω load, 1% THD)	0.85	0.9	1	W
• Maximum Out Power (VDDS = 3.0 V, 8 Ω load, 1% THD)	0.425	0.45	0.55	W
• Maximum Out Power (VDDS = 4.2 V, 4 Ω load, 10% THD)	1.6	1.75	1.9	W
• Maximum Out Power (VDDS = 4.2 V, 4 Ω load, 1% THD)	1.275	1.45	1.6	W
• Maximum Out Power (VDDS = 3.0 V, 4 Ω load, 1% THD)	0.6	0.7	0.8	W
• SNR (VDDS = 4.2 V, A-Weighted, 8 Ω load)	90	95	-	dB
• Speaker VDDS Active Current (no signal) (Note **3**)	-	1.3 + I_{offset}	45 mV/Ω	mA
• Speaker VDDS Leakage Current (VDDS=4.2V, Speaker = OFF)	-	0.1	0.6	uA
• THD+N (4 Ω load, -2dB signal)	-50	-55	-	dB
• THD+N (8 Ω load, -2dB signal)	-55	-61	-	dB
• PSRR at 217-1000 Hz	60	75	-	dB

1 1 Vrms requires external resistor divider.

2 Measured “A weighted” over a 20-Hz to a 20-kHz bandwidth, relative to full scale output voltage (when VDDIO = 3.3 V and VDDA = 2.1 V).

3 I_{offset} = offset voltage/speaker impedance.

4 Maximum input that achieves -40dB THD+N

5 ADC gain = 0 dB

6 Maximum output that achieves at least -66dB THD+N into 16 Ω load

7 Measured at -1dB FS, where fullscale achieves at least -66dB THD+N into a 16 Ω load

8 SNR and Dynamic range measurements made in capless headphone mode with DCDC converters running and JTAG disconnected.

9 Applies for both LINE IN and DAC IN sources.

2.3 DC Characteristics

Table 2-6. PSWITCH Input Characteristics

Parameter	HW_PWR_STS _PSWITCH	Min	Max	Units
PSWITCH LOW LEVEL (See Note 3)	0x00	0.00	0.30	V
PSWITCH MID LEVEL & STARTUP (See Note 1, 3)	0x01	0.65	1.50	V
PSWITCH HIGH LEVEL (See Note 2, 3)	0x11	2.1	VDDXTAL + 1.575	V

1 A MID LEVEL PSWITCH state can be generated by connecting the VDDXTAL output of the SOC to PSWITCH through a switch.

2 PSWITCH acts like a high impedance input (>300 kΩ) when the voltage applied to it is less than 1.5V. However, above 1.5V it becomes lower impedance. To simplify design, it is recommended that a 10 kΩ resistor to VDDIO be applied to PSWITCH to set the HIGH LEVEL state.

3 Consult the reference schematics for recommended PSWITCH button circuitry.

Table 2-7. Power Supply Characteristics

Parameter	Min	Typ	Max	Units
VDDXTAL Voltage Reference				
Output Voltage		1.0		V
Linear Regulators	**Min**	**Typ**	**Max**	**Units**
Output Voltage Accuracy (VDDIO, VDDA, VDDM, VDDD) (See Note 1)	-1		+3	%
VDDIO Maximum Output Current (VDDIO=3.325V, VDD5V=4.75V) (See Note 2, 8)	240			mA
VDDIO Maximum Output Current (VDDIO=3.325V, VDD5V=4.40V) (See Note 2, 8)	175			mA
VDDM Maximum Output Current (VDDM=2.5V, VDD5V=4.75V) (See Note 2, 8)	240			mA
VDDM Maximum Output Current (VDDM=2.5V, VDD5V=4.40V) (See Note 2, 8)	230			mA
VDDA Maximum Output Current (VDDA=1.8V, VDD5V=4.75V) (See Note 2, 8)	235			mA
VDDA Maximum Output Current (VDDA=1.8V, VDD5V=4.75V) (See Note 2, 8)	225			mA
VDDD Maximum Output Current (VDDD=1.2V, VDD5V=4.75V) (See Note 2, 8)	260			mA
VDDD Maximum Output Current (VDDD=1.2V, VDD5V=4.75V) (See Note 2, 8)	245			mA
VDDIO Output Impedance (See Note 9)	-	115	-	mΩ
VDDM Output Impedance (See Note 9)	-	335	-	mΩ
VDDA Output Impedance (See Note 9)	-	105	-	mΩ
VDDD Output Impedance (See Note 9)	-	175	-	mΩ
DCDC Converters	**Min**	**Typ**	**Max**	**Units**
Output Voltage Accuracy (DCDC_VDDIO, DCDC_VDDA, DCDC_VDDD) (See Note 1)	-1		+3	%
DCDC Converters (3.0V < BATT < 4.242V)	**Min**	**Typ**	**Max**	**Units**
DCDC_VDDD Maximum Output Current (VDDD=1.55V) (See Note 3, 7)	250			mA
DCDC_VDDA Maximum Output Current (VDDA=1.8V) (See Note 3, 7)	200			mA

Table 2-7. Power Supply Characteristics (continued)

DCDC_VDDIO Maximum Output Current (VDDIO=3.30V, VDDD=1.55V) (See Note 3, 4, 7)	135			mA
DCDC_VDDIO Maximum Output Current (VDDIO=3.20V, VDDD=1.55V) (See Note 3, 4, 7)	120			mA
DCDC_VDDIO Maximum Output Current (VDDIO=3.15V, VDDD=1.55V) (See Note 3, 4, 7)	95			mA
DCDC_VDDIO Maximum Output Current (VDDIO=3.15V, VDDD=1.375V) (See Note 3, 4, 7)	175			mA
DCDC Converters (3.3V < BATT < 4.242V)	**Min**	**Typ**	**Max**	**Units**
DCDC_VDDD Maximum Output Current (VDDD=1.55V, 3.3V < BATT < 4.242V) (See Note 3, 7)	250			mA
DCDC_VDDA Maximum Output Current (VDDA=1.8V, 3.3V < BATT < 4.242V) (See Note 3, 7)	200			mA
DCDC_VDDIO Maximum Output Current (VDDIO=3.30V, VDDD=1.55V) (See Note 3, 4, 7)	215			mA
DCDC_VDDIO Maximum Output Current (VDDIO=3.20V, VDDD=1.55V) (See Note 3, 4, 7)	175			mA
DCDC_VDDIO Maximum Output Current (VDDIO=3.15V, VDDD=1.55V) (See Note 3, 4, 7)	135			mA
DCDC_VDDIO Maximum Output Current (VDDIO=3.15V, VDDD=1.375V) (See Note 3, 4, 7)	265			mA
DCDC Converters (4.0V < BATT < 4.242V)	**Min**	**Typ**	**Max**	**Units**
DCDC_VDDIO Maximum Output Current (VDDIO=3.30V, VDDD=1.55V) (See Note 3, 4, 7)	355			mA
DCDC_VDDIO Maximum Output Current (VDDIO=3.20V, VDDD=1.55V) (See Note 3, 4, 7)	340			mA
DCDC_VDDIO Maximum Output Current (VDDIO=3.15V, VDDD=1.55V) (See Note 3, 4, 7)	325			mA
DCDC_VDDIO Maximum Output Current (VDDIO=3.15V, VDDD=1.375V) (See Note 3, 4, 7)	420			mA
VDD4P2 Regulated Output	**Min**	**Typ**	**Max**	**Units**
VDD4P2 Output Voltage Accuracy (TARGET=4.2V) (Note 1)	-2		+1	%
VDD4P2 Maximum Output Current (VDD5V=5.00V, ILIMIT=780mA) (See Note 5)	725			mA
VDD4P2 Maximum Output Current (VDD5V=4.75V, ILIMIT=780mA) (See Note 5)	605			mA
VDD4P2 Maximum Output Current (VDD5V=4.40V, ILIMIT=780mA) (See Note 5)	325			mA
VDD4P2 Output Impedance (See Note 9)		90		mΩ
VDD4P2 Output Current Limit Accuracy (VDD5V=4.75V, ILIMIT=480mA) (See Note 6)	466	480	505	mA
VDD4P2 Output Current Limit Accuracy (VDD5V=4.75V, ILIMIT=100mA) (See Note 6)	95	100	105	mA
Battery Charger	**Min**	**Typ**	**Max**	**Units**
Final Charge Voltage Accuracy (TARGET=4.2V)	-2		+1	%

1 No Load.

2 Output regulated within 100 mV of target voltage.

3 DCDC Double FETs Enabled, Inductor Value = 15µH.

4 Assumes simultaneous load of IDDD = 250 mA@1.55V and IDDA = 200 mA@1.8V.

5 Output regulated within 300 mV of target voltage.

6 Untuned.

7 The DCDC Converter is a triple output buck converter. The maximum output current capability of each output of the converter is dependent on the loads on the other two outputs. For a given output, it may be possible to achieve a maximum output current higher than that specificed by ensuring the load on the other outputs is well below the maximum.

8 Because the internal linear regulators are cascaded, it is not possible to simultaneously operate the VDDIO, VDDA, VDDM, and VDDD linear regulators at the maximum specified load current. For example, the VDDIO linear regulator provides current to both the VDDIO 3.3 V supply rail as well as the VDDM and VDDA linear regulator inputs. Likewise, the VDDA linear regulator provides current to both the 1.8 V supply rail as well as the VDDD linear regulator input. The application designer should ensure the following two conditions are met:

- (VDDIO Load Current + VDDM Load Current + VDDA Load Current) < VDDIO Maximum Output Current
- (VDDA Load Current + VDDD Load Current) < VDDA Maximum Output Current

9) The output impedance value is measured on a PCB evaluation board and includes the PCB trace impedance.

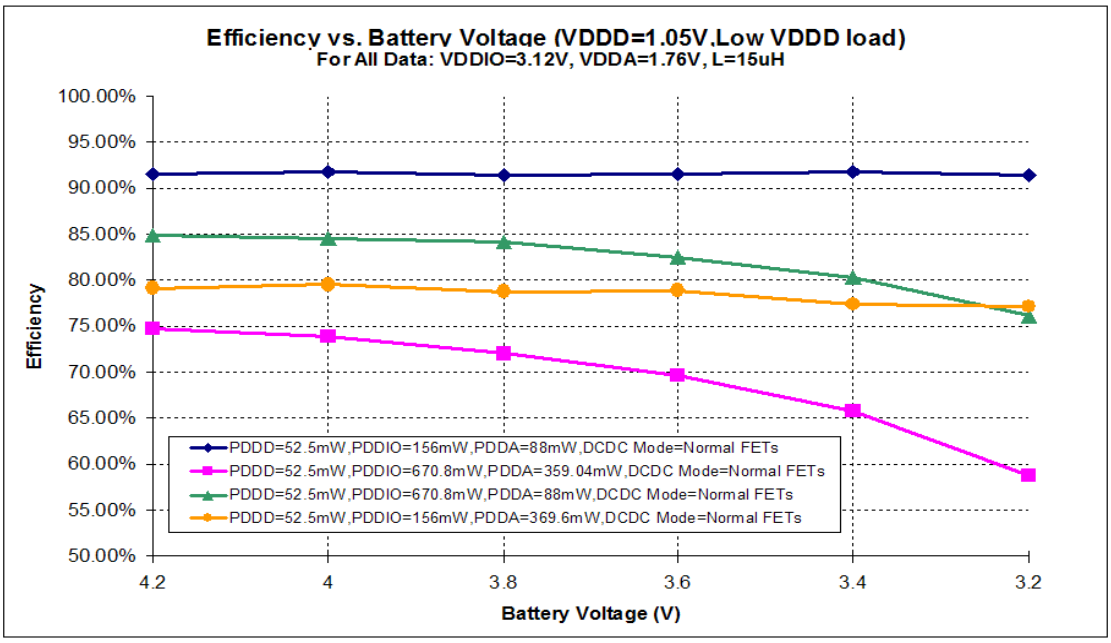


Figure 2-1. DCDC Efficiency vs. Battery Voltage (VDDD=1.05V, Low VDDD Load)

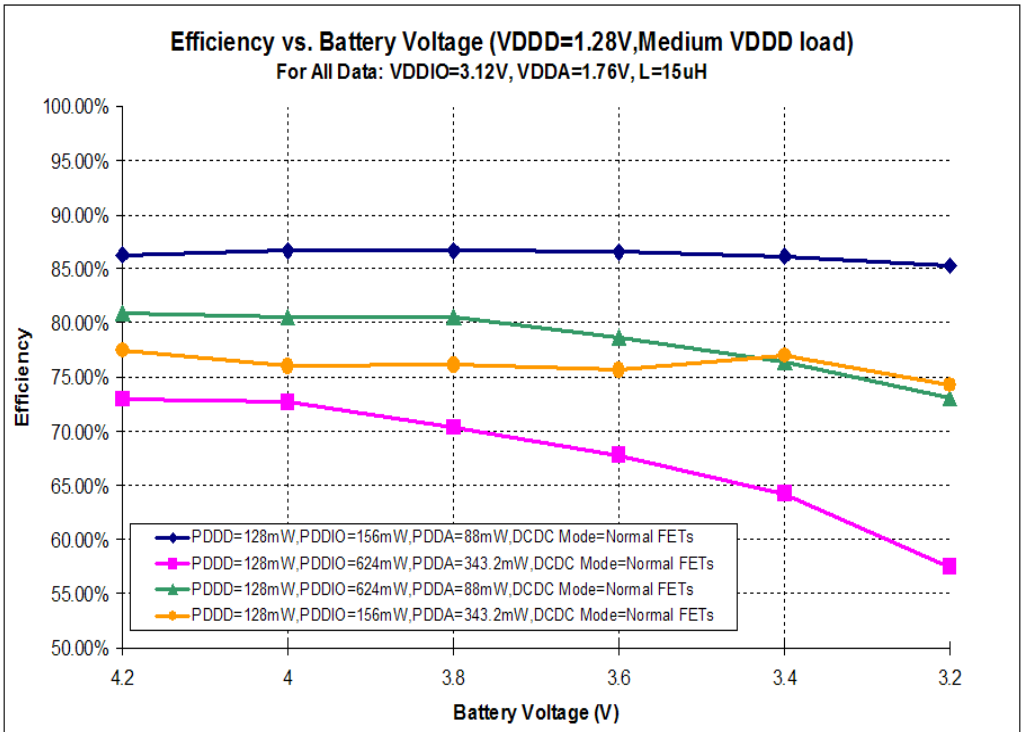


Figure 2-2. DCDC Efficiency vs. Battery Voltage (VDDD=1.28V, Medium VDDD Load)

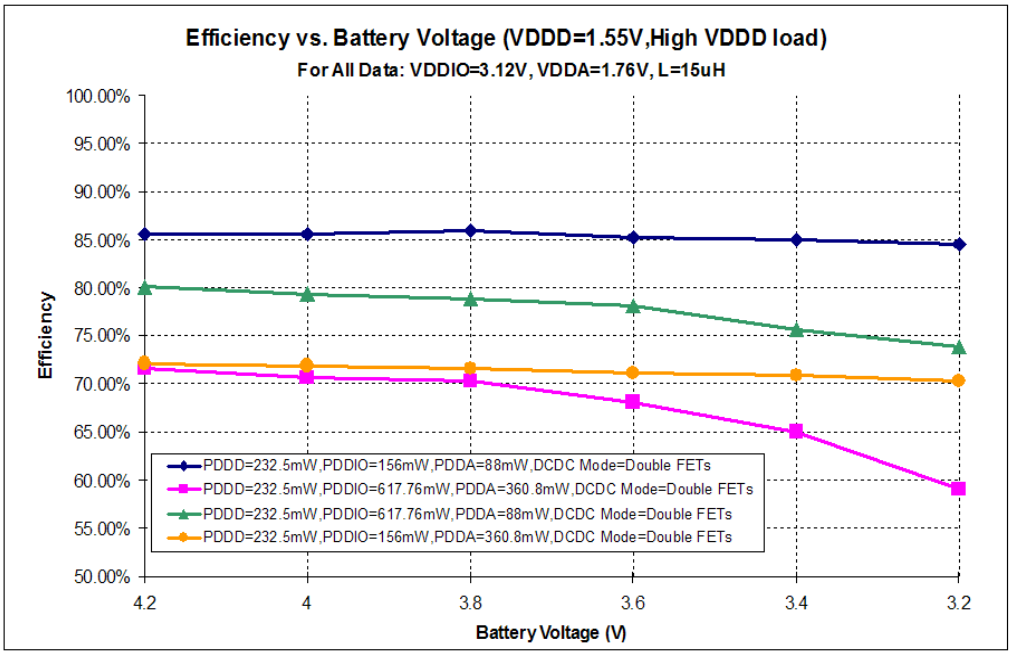


Figure 2-3. DCDC Efficiency vs. Battery Voltage (VDDD=1.55V, High VDDD Load)

Table 2-8. Non-EMI Digital Pin DC Characteristics

		VDDIO = 3.3 V		
Parameter	**Name**	**Min**	**Max**	**Units**
Non-EMI Regular & High Drive I/O Input Voltage	VIH	2.00	VDDIO	V
	VIL	-	0.80	V
Non-EMI Regular & High Drive I/O Output Voltage	VOH	0.8 * VDDIO	-	V
	VOL	-	0.40	V
Non-EMI Regular I/O Output Current (see notes 1 and 6 of Table 2-10)	IOH - 4mA	3.60	-	mA
	IOH - 8mA	7.20	-	mA
	IOH - 12mA	10.80	-	mA
	IOL - 4mA	-3.60	-	mA
	IOL - 8mA	-7.20	-	mA
	IOL - 12mA	-10.80	-	mA
Non-EMI High Drive I/O (PWM4) Output Current (see notes 2 and 6 of Table 2-10)	IOH - 8mA	-6.50	-	mA
	IOH - 16mA	-11.00	-	mA
	IOH - 24mA	-16.80	-	mA
	IOL - 8mA	-8.00	-	mA
	IOL - 16mA	-14.50	-	mA
	IOL - 24mA	-19.00	-	mA
External Pull-Up / Pull-Down Resistor Value Required to Overdrive Internal Gate Keeper		-	50	kΩ
Internal Pull-Up Resistor Accuracy	-20	-	+20	%

.

Table 2-9. EMI Digital Pin DC Characteristics

		VDDIO.EMI = 2.5 V		VDDIO.EMI = 1.8 V		Unit
Parameter	**Name**	**Min**	**Max**	**Min**	**Max**	
EMI I/O Input Voltage	VIH	(VDDIO.EMI / 2) + 0.2	VDDIO.EMI	(VDDIO.EMI / 2) + 0.2	VDDIO.EMI	V
	VIL	-	(VDDIO.EMI / 2) - 0.2	-	(VDDIO.EMI / 2) - 0.2	V
EMI I/O Output Voltage	VOH	0.7 * VDDIO.EMI	-	0.8 * VDDIO.EMI	-	V
	VOL	-	0.40	-	0.2 * VDDIO.EMI	V
EMI I/O Output Current (See notes 1 and 6 of Table 2-10)	IOH - 4 mA	3.00	-	3.00	-	mA
	IOH - 8 mA	6.00	-	5.00	-	mA
	IOH - 12 mA (see note 3 of Table 2-10)	10.00	-	8.50	-	mA
	IOH - 16 mA (see note 3 of Table 2-10)	14.00	-	11.00	-	mA
	IOL - 4 mA	-4.00	-	-3.50	-	mA
	IOL - 8 mA	-8.00	-	-7.00	-	mA
	IOL - 12 mA (see note 3 of Table 2-10)	-12.00	-	-10.50	-	mA
	IOL - 16 mA (see note 3 of Table 2-10)	-16.00	-	-13.50	-	mA

Table 2-10. External Devices Supported by the EMI

DRAM Device	**Max Load** (see notes 4 and 5)	**Pad Voltage**
DDR	15 pF	2.5 V
mDDR	15 pF	1.8 V

1 The stronger the driver mode, the noisier the on-chip power supply. The use of a stronger drive mode must be limited to only a few pins. The majority of GPIO drivers must be set in 4-mA mode.

2 High-drive I/O has a high current source/sink capability. However, it is not meant as high-speed I/O - the driver turns on slowly to reduce L*di/dt power supply noise.

3 The EMI I/O pad pre-drivers are powered from VDDIO rather than VDDIO.EMI. This causes the higher EMI I/O drive strengths at 2.5 V and 1.8 V to have a dependency on the VDDIO voltage. For 2.5 V and 1.8 V EMI I/O 12 mA & 16 mA drive strengths, VDDIO should equal 3.3 V or higher.

[4] Max load includes capacitive load due to PCB traces, pad capacitance and driver self-loading.

[5] Setting is for worst case. Freescale's EMI interface uses less powerful drivers than those typically used in mDDR devices. A possible transmission-line effect on the PC board must be suppressed by minimizing the trace length combined with Freescale's slower edge-rate drivers. The i.MX23 provides up to 16 mA programmable drive strength. However, the 16-mA mode is an experimental mode. With the 16-mA mode, the EMI function may be impaired by simultaneous switching output (SSO) noise. In general, the stronger the driver mode, the noisier the on-chip power supply. Freescale recommends not using a stronger driver mode than is required. Because on-chip power and ground noise is proportional to the inductance of its return path, users should make their best effort to reduce inductance between the EMI power and ground balls and the PC board power and ground planes.

[6] IOH is the maximum output current at which the VOH specification is met. IOL is the maximum input current at which the VOL specification is met.

2.3.1 Recommended Operating Conditions for Specific Clock Targets

NOTE

At this time, all data is preliminary and subject to change without notice.

Table 2-11. System Clocks

Name	Min. Freq. (MHz)	Max. Freq. (MHz)	Description
clk_gpmi	-	102.858	General Purpose memory interface clock domain
clk_ssp	-	102.858	Internal SSP Interface clock.
External SSP Clock	-	51.429	External SSP clock.

Table 2-12. Recommended Operating States - 169BGA Package

VDDD (V)	VDDD Brown-out (V)	HW_ DIGCTRL ARMCACHE (note 1)	CPUCLK / clk_p Frequency (MHz)	HW_ CLKCTRL CPU_DIV_CPU	HW_ CLKCTRL FRAC_ CPUFRC / PFD	AHBCLK / clk_h Frequency (MHz)	HW_ CLKCTRL HBUS_DIV	EMICLK / clk_emi Frequency (MHz)	HW_ CLKCTRL EMI_ DIV_EMI	HW_ CLKCTRL FRAC_ EMIFRAC	SUPPORTED DRAM
1.050	0.975		24.00			24.00	1	24.00			DDR, mDDR
1.050	0.975	11	64.00	5	27	64.00	1	64.00	5	27	DDR, mDDR
1.275	1.175	00	261.82	1	33	130.91	2	130.91	2	33	DDR, mDDR
1.375	1.275	00	360.00	1	24	120.00	3	120.00	3	24	DDR, mDDR
1.475	1.375	00	392.73	1	22	196.36	2	130.91	2	33	DDR, mDDR
1.550	1.450	00	454.74	1	19	151.58	3	151.58	3	19	mDDR
1.550	1.450	00	454.74	1	19	151.58	3	130.91	2	33	DDR

1 All timing control bit fields in HW_DIGCTRL_ARMCACHE should be set to the same value.

Table 2-13. Recommended Operating States - 128QFP Package

VDDD (V)	VDDD Brown-out (V)	HW_ DIGCTRL ARMCACHE (note 1)	CPUCLK / clk_p Frequency (MHz)	HW_ CLKCTRL CPU_DIV_CPU	HW_ CLKCTRL FRAC_ CPUFRC / PFD	AHBCLK / clk_h Frequency (MHz)	HW_ CLKCTRL HBUS_DIV	EMICLK / clk_emi Frequency (MHz)	HW_ CLKCTRL EMI_ DIV_EMI	HW_ CLKCTRL FRAC_ EMIFRAC	SUPPORTED DRAM
1.050	0.975		24.00			24.00	1	24.00			mDDR
1.050	0.975	11	64.00	5	27	64.00	1	64.00	5	27	DDR, mDDR
1.275	1.175	00	261.82	1	33	130.91	2	130.91	2	33	DDR, mDDR
1.375	1.275	00	360.00	1	24	120.00	3	120.00	3	24	DDR, mDDR
1.475	1.375	00	392.73	1	22	196.36	2	130.91	2	33	DDR, mDDR
1.550	1.450	00	454.74	1	19	151.58	3	130.91	2	33	DDR, mDDR

Table 2-14. Recommended Operating Conditions - CPU Clock (clk_p)

Minimum VDDD (V)	Minimum $VDDD_{Brown\text{-}out}$ (V)	HW_DIGCTRL ARMCACHE (note 1)	HW_CLKCTRL FRAC_CPUFRC / PFD	CPUCLK / clk_p Frequency max (MHz)
1.050	0.975	11	25 - 35	64.00
1.225	1.125	00	18 - 35	278.71
1.375	1.275	00	18 - 35	360.00
1.450	1.350	00	18 - 35	392.73
1.550	1.450	00	18 - 35	454.74

1 All timing control bit fields in HW_DIGCTRL_ARMCACHE should be set to the same value.

Table 2-15. Recommended Operating Conditions - AHB Clock (clk_h)

Minimum VDDD (V)	Minimum $VDDD_{Brown\text{-}out}$ (V)	HW_DIGCTRL ARMCACHE (note 1)	HW_CLKCTRL FRAC_CPUFRC / PFD	AHBCLK / clk_h Frequency max (MHz)
1.050	0.975	11	25 - 35	64.00
1.275	1.175	00	18 - 35	130.91
1.350	1.250	00	18 - 35	160.00
1.475	1.375	00	18 - 35	196.36
1.525	1.425	00	18 - 35	205.71

1 All timing control bit fields in HW_DIGCTRL_ARMCACHE should be set to the same value.

Table 2-16. Frequency vs. Voltage for EMICLK - 169-Pin BGA Package

Minimum VDDD (V)	Minimum $VDDD_{Brownout}$ (V)	EMICLK Fmax (MHz)	
		DDR (note 1)	mDDR (note 2)
1.55	1.45	130.91	151.58

Table 2-16. Frequency vs. Voltage for EMICLK - 169-Pin BGA Package (continued)

Minimum VDDD (V)	Minimum $VDDD_{Brownout}$ (V)	EMICLK Fmax (MHz)	
		DDR (note 1)	mDDR (note 2)
1.45	1.35	130.91	151.58
1.30	1.20	130.91	151.58
1.20	1.10	130.91	151.58
1.05	0.975	96.00	96.00

1) DDR EMICLK maximum is valid for the following conditions: Temp ≤ 105°, 2.4V ≤ VDDM ≤ 2.5V, 3.2V ≤ VDDIO ≤ 3.3V, Drive strength ≥ 12mA

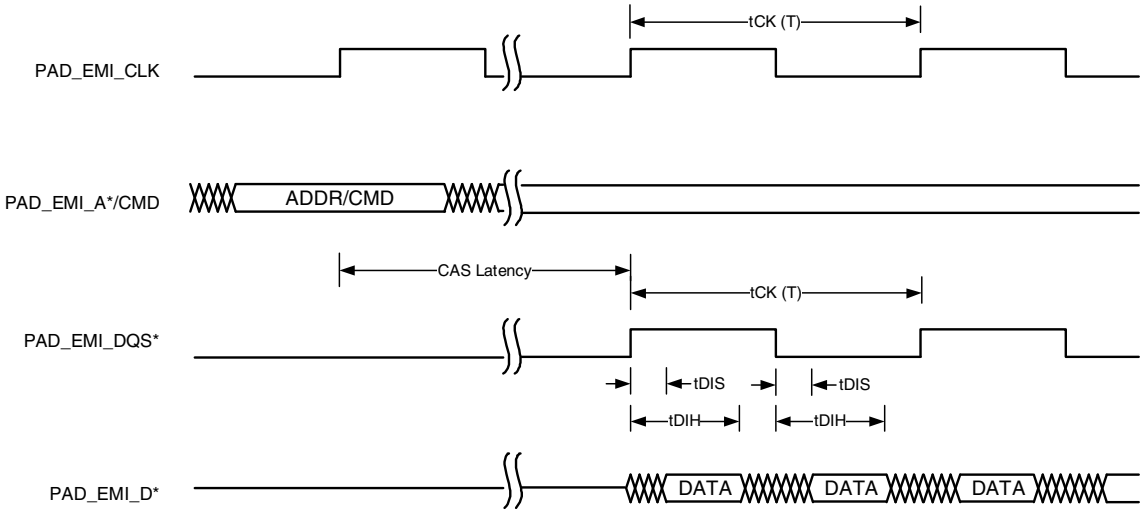
Table 2-17. Frequency vs. Voltage for EMICLK - 128-Pin LQFP Package

Minimum VDDD (V)	Minimum $VDDD_{Brownout}$ (V)	EMICLK Fmax (MHz)	
		DDR (note 1)	mDDR (note 2)
1.55	1.45	130.91	130.91
1.45	1.35	130.91	130.91
1.30	1.20	130.91	130.91
1.20	1.10	130.91	130.91
1.05	0.975	96.00	96.00

2) mDDR EMICLK maximum is valid for the following conditions: Temp ≤ 105°, 1.7V ≤ VDDA ≤ 1.9V, 3.2V ≤ VDDIO ≤ 3.3V, Drive strength ≥ 12mA

2.4 AC Characteristics

2.4.1 EMI Electrical Specifications



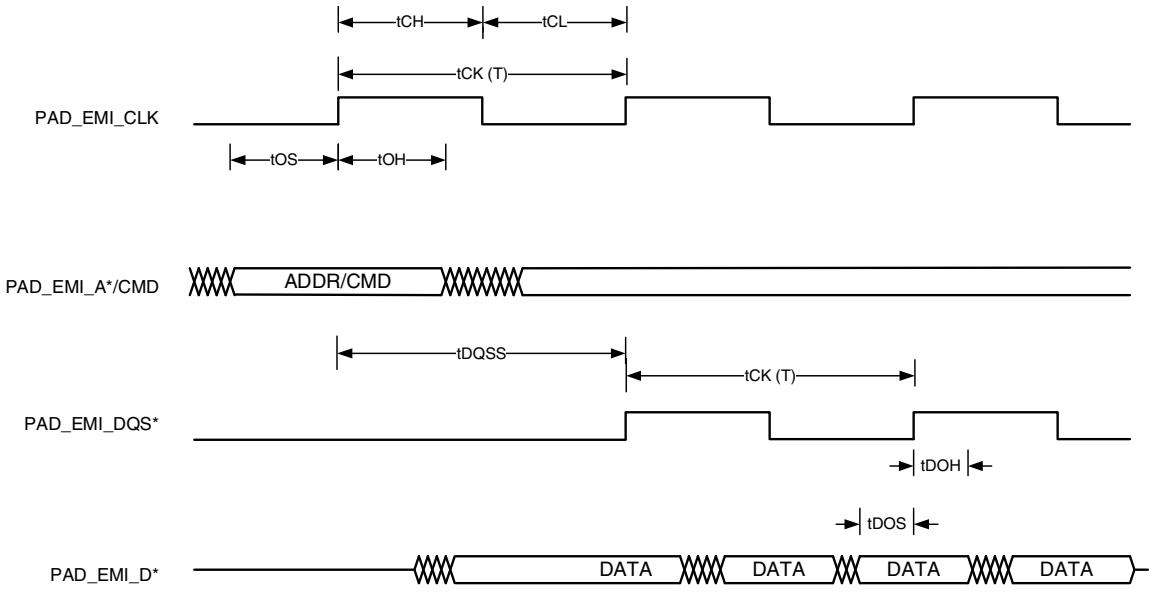
Assumptions
==========
VDDD PVT : 1.08V, SS, 125C Junction
VDDA IO PVT : 1.62V, SS, 125C Junction (1.8V setting, but WCS IO voltage)
IO Drive Strength = 4mA, Cap Load = 15pF on all pins
DQS has pull-downs on board (never goes high-Z), but DQ has keepers disabled.

DQS In Delay chain setting = 4 taps WCS, 13 taps BCS (approx ¼ cycle, ie approx 0x20)

Note that the SoC creates an internal delay on the DQS relative to DQ, so data launched from the DRAM on the rising edge of the DQS will set-up to the rising edge of the DQS, and will hold to the previous edge.

Legend
======
tDIS = Data Input Max Setup Time relative to DQS = 0.25T - 0.85 (e.g. at 151.58MHz, tDQSQ cannot exceed 0.25*(1000/151.58) - 0.85 = 0.8ns)
tDIH = Data Input Minimum Hold Time relative to DQS = 0.25T + 0.75 (e.g. at 151.58MHz, tQH must be at least 0.25*(1000/151.58) + 0.75 = 2.4ns)

Figure 2-4. i.MX23 EMI mDDR DRAM Input AC Timing



Assumptions
==========
VDDD PVT : 1.08V, SS, 125C Junction (unless otherwise noted)
VDDA IO PVT : 1.62V, SS, 125C Junction (1.8V setting, but WCS IO voltage)
IO Drive Strength = 4mA, Cap Load = 15pF on all pins
DQS has pull-downs on board (never goes high-Z), but DQ has keepers disabled.

DQS Out Delay chain setting = 0
DQS Write Clock Delay chain setting = 5 taps (approx ¼ cycle, ie approximately 0x20)
Clock Delay line setting = 5 (this also works at BCS PVT and gives best CK/DQS skew)

Legend
======
tCK = T = DRAM Clock Cycle Time = @ VDDD=1.55V 6.6ns (min), @ VDDD=1V 7.639ns (min)
tCH = DRAM Clock High Pulse = T/2 to T/2 - 0.37ns
tCL = DRAM Clock Low Pulse = T/2 to T/2 + 0.37ns
tOS = Addr/Cmd output setup to CK rising = T/2 - 0.96ns (min)
tOH = Addr/Cmd output hold to CK rising = T/2 - 1.51ns (min)

tDQSS = Write command valid to first DQS latching transition = T to T+0.1

tDOS = DQ to DQS setup time = T/4 - 0.485ns (min)
tDOH = DQ to DQS hold time = T/4 - 0.365ns (min)

Figure 2-5. i.MX23 EMI mDDR DRAM Output AC Timing

2.4.2 I²C Electrical Specifications

Figure 2-6 depicts the timing for the I²C module. Table 2-18 lists the I²C module timing parameters.

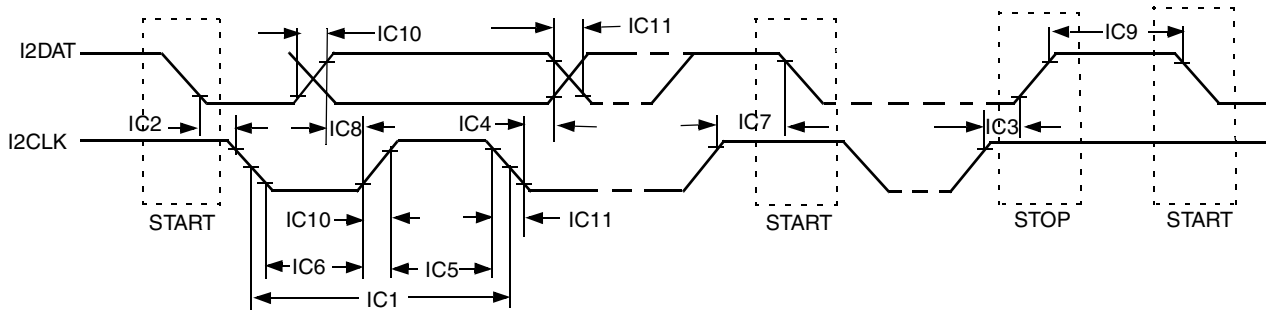


Figure 2-6. I²C Bus Timing Diagram

Table 2-18. I²C Timing Parameters

ID	Parameter	Min	Max	Unit
	I²C Input Timing			
IC1	I2CLK cycle time [a]	8	-	clk_x cycles
IC2	Hold time (repeated) START condition	1	-	clk_x cycles
IC3	Set-up time for STOP condition	1	-	clk_x cycles
IC4	Data hold time	1	-	clk_x cycles
IC5	HIGH Period of I2CLK Clock	4	-	clk_x cycles
IC6	LOW Period of the I2CLK Clock	4	-	clk_x cycles
IC7	Set-up time for a repeated START condition	1	-	clk_x cycles
IC8	Data set-up time	1	-	clk_x cycles
IC9	Bus free time between a STOP and START condition	4	-	clk_x cycles
	I²C Output Timing			
IC1	I2CLK cycle time [b]	high_count + low_count + 6	-	clk_x cycles
IC2	Hold time (repeated) START condition	leadin_count	-	clk_x cycles
IC3	Set-up time for STOP condition	rcv_count + 6	-	clk_x cycles
IC4	Data hold time	xmit_count	-	clk_x cycles
IC5	HIGH Period of I2CLK Clock	high_count	-	clk_x cycles
IC6	LOW Period of the I2CLK Clock	low_count	-	clk_x cycles
IC7	Set-up time for a repeated START condition	bus_free + 7	-	clk_x cycles
IC8	Data set-up time	low_count - xmit_count	-	clk_x cycles
IC9	Bus free time between a STOP and START condition	bus_free + 7 - rcv_count	-	clk_x cycles
IC10	Rise time of both I2DAT and I2CLK signals	0.15*Cb	0.17*Cb	ns

Table 2-18. I²C Timing Parameters (continued)

ID	Parameter	Min	Max	Unit
IC11	Fall time of both I2DAT and I2CLK signals	0.16*Cb	0.17*Cb	ns
IC12	Capacitive load for each bus line (C_b) [c]	5	100	pF

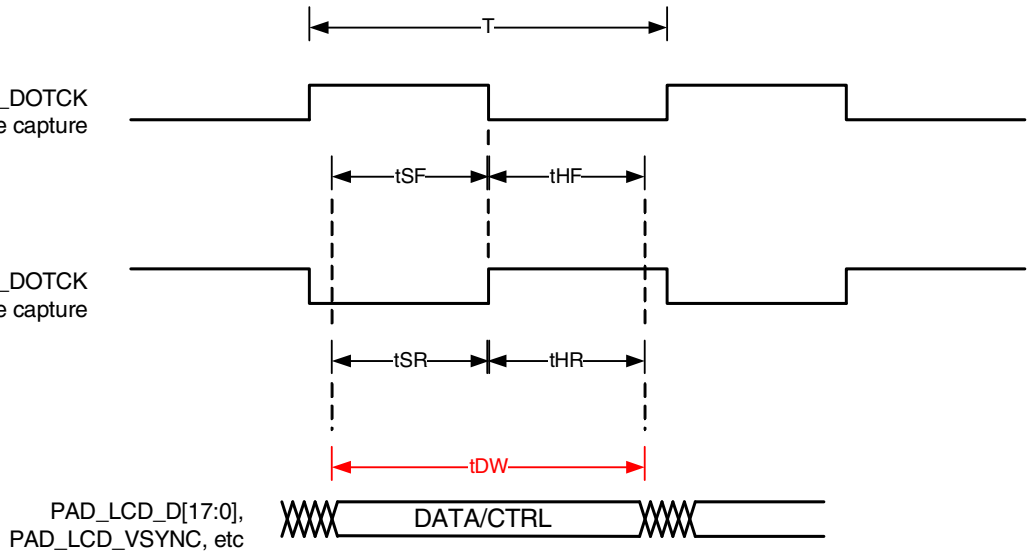
[a] The clk_x period is programmed as a divide with respect to the xtal clock. The divide value can be >= 1.

[b] All I2C output timings are determined by PIO register values. These values are multiplied by the programmable clk_x period.

[c] C_b = total capacitance of one bus line in pF.

2.4.3 LCD AC Output Electrical Specifications

Figure 2-7 depicts the AC output timing for the LCD module. Table 2-19 lists the LCD module timing parameters.



Notes:
T = LCD interface clock period
I/O Drive Strength = 4mA
I/O Voltage = 3.3V
Cck = Capacitance load on DOTCK pad
Cd = Capacitance load on DATA/CTRL pad

Figure 2-7. LCD AC Output Timing Digram

Table 2-19. LCD AC Output Timing Parameters

ID	Parameter	
tSF	Data setup for falling edge	DOTCK = T/2 – 1.97ns + 0.15*Cck – 0.19*Cd
tHF	Data hold for falling edge	DOTCK = T/2 + 0.29ns + 0.09*Cd – 0.10*Cck
tSR	Data setup for rising edge	DOTCK = T/2 – 2.09ns + 0.18*Cck – 0.19*Cd
tHR	Data hold for rising edge	DOTCK = T/2 + 0.40ns + 0.09*Cd – 0.10*Cck
tDW	Data valid window	tDW = T – 1.45ns

Chapter 3
ARM CPU Complex

This chapter describes the ARM CPU included on the i.MX23 and includes sections on the processor core, the JTAG debugger, and the embedded trace macrocell (ETM) interface.

3.1 ARM 926 Processor Core

The on-chip Reduced Instruction Set Computer (RISC) processor core is an ARM, Ltd. 926EJ-S. This CPU implements the ARM v5TE instruction set architecture, which includes enhanced DSP instructions.

The ARM9EJ-S has two instruction sets: a 32-bit instruction set used in the ARM state and a 16-bit instruction set used in Thumb state. The core offers the choice of running in the ARM state or the Thumb state or a mix of the two. This enables optimization for both code density and performance.

A block diagram of the ARM926EJ-S core is shown in Figure 3-1.

See http://www.arm.com/documentation/ARMProcessor_Cores/index.html to download the following ARM documentation on the ARM926EJ-S core:

- ARM926EJ-S Technical Reference Manual, DDI0198D
- ARM926EJ-S Development Chip Reference Manual, DDI0287A

The ARM9 core has a total of 37 programmer-visible registers, including 31 general-purpose 32-bit registers, six 32-bit status registers, and a 32-bit program counter, as shown in Figure 3-2. In ARM state, 16 general-purpose registers and one or two status registers are accessible at any one time. In privileged modes, mode-specific banked registers become available.

The ARM state register set contains 16 directly addressable registers, r0 through r15. An additional register, the current program status register (CPSR), contains condition code flags and the current mode bits. Registers r0–r13 are general-purpose registers used to hold data and address values, with R13 being used as a stack pointer. R14 is used as the subroutine link register (lr) to hold the return address. Register r15 holds the program counter (PC).

The Thumb state register set is a subset of the ARM register set. The programmer has access to eight general-purpose registers, r0–r7, the PC (ARM r15), the stack pointer (ARM r13), the link register (ARM r14), and the cpsr.

Exceptions arise whenever the normal flow of program execution has to be temporarily suspended, for example, to service an interrupt from a peripheral. Before attempting to handle an exception, the ARM

core preserves the current processor state, so that the original program can resume when the handler is finished.

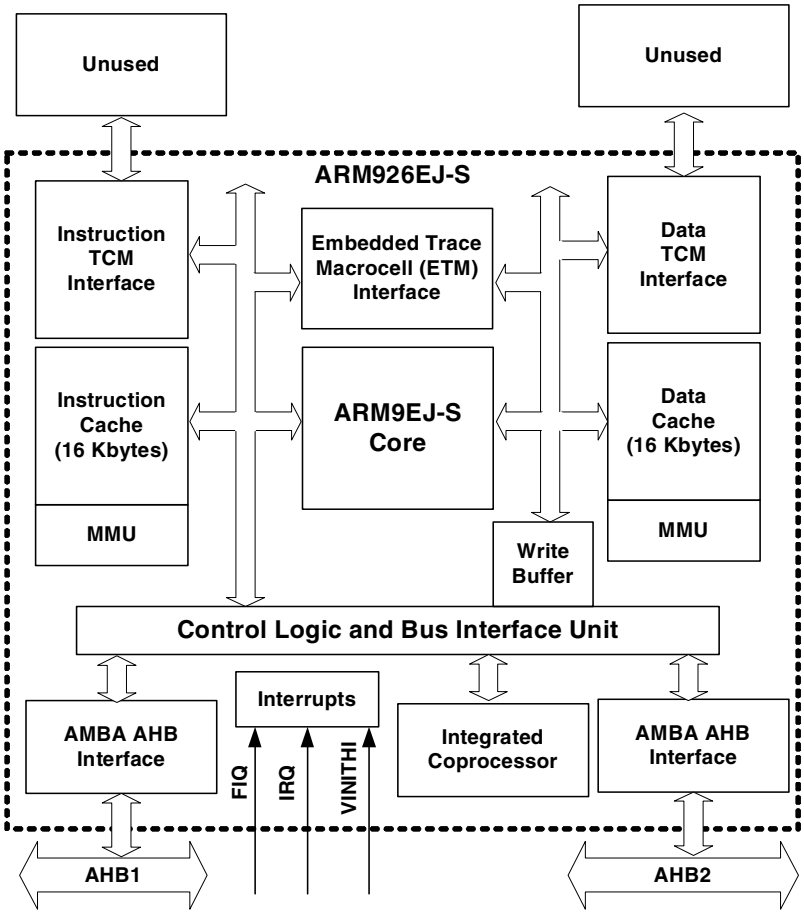


Figure 3-1. ARM926 RISC Processor Core

The following exceptions are recognized by the core:

- SWI—Software interrupt
- UNDEF—Undefined instruction
- PABT—Instruction prefetch abort
- FIQ—Fast peripheral interrupt
- IRQ—Normal peripheral interrupt
- DABT—Data abort
- RESET—Reset
- BKPT—Breakpoint

The vector table pointing to these interrupts can be located at physical address 0x00000000 or 0xFFFF0000. The i.MX23 maps its 64-Kbyte on-chip ROM to the address 0xFFFF0000 to 0xFFFFFFFF. The core is hardwired to use the high address vector table at hard reset (core port VINITHI =1).

The ARM 926 core includes a 16-Kbyte instruction cache and 16-Kbyte data cache and has two master interfaces to the AMBA AHB, as shown in Figure 3-1.

The i.MX23 always operates in little-endian mode.

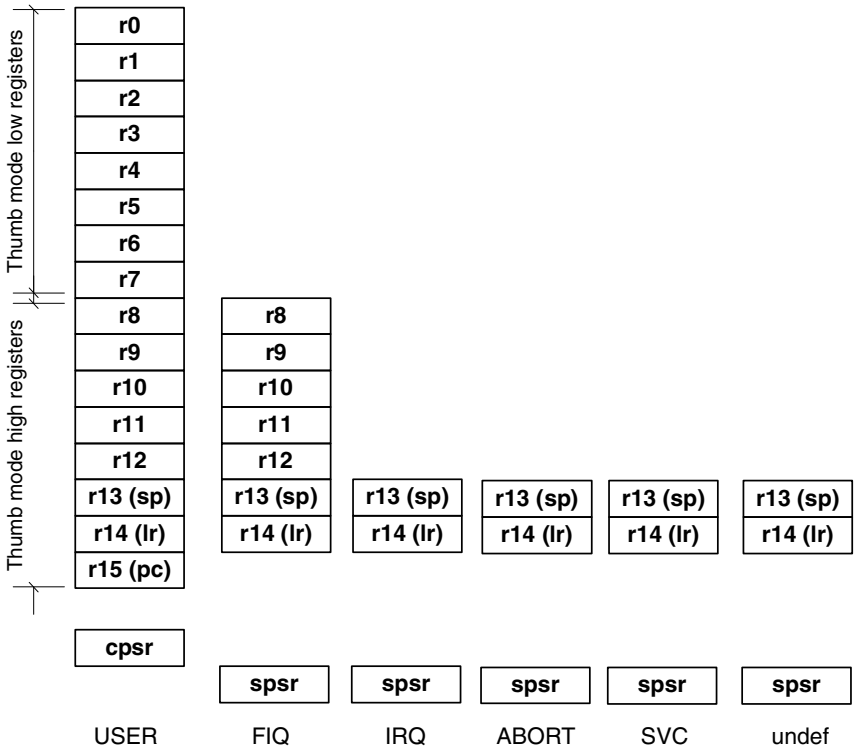


Figure 3-2. ARM Programmable Registers

3.2 JTAG Debugger

The TAP controller of the ARM core in the i.MX23 performs the standard debugger instructions.

3.2.1 JTAG READ ID

The TAP controller returns the following 32-bit data value in response to a JTAG READ ID instruction: 0x0792_64F3

3.2.2 JTAG Hardware Reset

The JTAG reset instruction can be accomplished by writing 0xDEADC0DE to ETM address 0x70. The ETM is on scan chain 6. The bitstream is 0xF0DEADC0DE.

The digital wide reset does not affect the DC-DC converters or the contents of the persistent registers in the analog side of the RTC.

3.2.3 JTAG Interaction with CPUCLK

Because the JTAG clock is sampled from the processor clock CPUCLK, there are cases in which the behavior of CPUCLK affects the ability to make use of JTAG. Specifically, the JTAG block will not function as expected if:

- CPUCLK is stalled due to an interrupt
- CPUCLK is less than 3x the JTAG clock
- CPUCLK is disabled for any reason

3.3 Embedded Trace Macrocell (ETM) Interface (169BGA-only)

The i.MX23 includes a stand-alone ARM CoreSight Embedded Trace Macrocell, ETM9CSSingle, which provides instruction trace and data trace for the ARM9 microprocessor. For more details see the CoreSight ETM9 Technical Reference Manual. Also, see the pin list in Chapter 36, “Pin Descriptions,” for pinout information.

Chapter 4
Clock Generation and Control

4.1 Overview

The clock control module, or CLKCTRL, generates the clock domains for all components in the i.MX23 system. The crystal clock or PLL clock are the two fundamental sources used to produce all the clock domains. For lower performance and reduced power consumption, the crystal clock is selected. The PLL is selected for higher performance requirements but requires increased power consumption. In most cases, when the PLL is used as the source, a phase fractional divider (PFD) can be programmed to reduce the PLL clock frequency by up to a factor of 2.

The PLL and PFD clocks are used as reference clock sources to drive digital clock dividers in the clock control module. These reference clocks, or ref_<clock>, drive the digital clock dividers in CLKCTRL. The digital clock dividers have three modes of operation, integer divide mode, fractional divide mode, and gated clock mode. The details of these three modes will be described to understand which mode should be selected to achieve the desired frequency.

All programming control for system clocks are contained in the CLKCTRL module. All clock domains have a programmable clock frequency to meet application requirements. Also, all analog clock control programming is done indirectly through the CLKCTRL module. This contains the complexity of overall system clock selection to a single device. Also, the hardware used to generate all clock domains is replicated. Following is a description of all clock domains in the i.MX23 system.

4.2 Clock Structure

The reference clocks are used in CLKCTRL as fundamental clock sources to produce clock domains throughout the system. A reference clock can be either the crystal clock, 480Mhz PLL, or PFD output from the analog module. The selected reference clock is used by a digital clock divider to produce the desired clock domain. The table below summarizes all available reference clocks used within the CLKCTRL and all clock domains used in the system. The diagram that follows depicts all clock domains and how they are connected within the CLKCTRL module. This should provide a reference for how clocks are generated within the i.MX23 system.

4.2.1 Table of System Clocks

Table 4-1 summarizes the clocks produced by the clock control module.

Table 4-1. System Clocks

NAME	REFERENCE	DIVIDE /FREQ	DESCRIPTION
Reference Clocks.			
ref_xtal	xtal_24m /ring_24m	1	This is the muxed select between the internal ring oscillator and the external crystal.
ref_cpu	PLL	9 phase	The 9 phase fractional divider output used as the reference for the CPU clock divider.
ref_emi	PLL	9 phase	The 9 phase fractional divider output used as the reference for the EMI clock divider.
ref_io	PLL	9 phase	The 9 phase fractional divider output used as the reference for the GPMI, SSP, and IR clock dividers.
ref_pix	PLL	9 phase	The 9 phase fractional divider output used as the reference for the PIX clock divider.
ref_vid	PLL	9 phase	The 9 phase fractional divider output used as the reference for the clk_tv108m clock divider.
ref_pll	PLL	1	This is the raw PLL output used as the reference for the SAIF clock divider.
Divided clock domains referenced from PLL or Xtal clock.			
clk_p	ref_xtal /ref_cpu	10/6 bits	ARM core clock.
clk_h	clk_p	5 bits	AHB/APBH clock domain. clk_h is a gated branch of the clk_p domain.
clk_etm	ref_xtal /ref_cpu	6/6 bits	ARM etm clock.
clk_emi	ref_xtal /ref_emi /ref_cpu	4/6 bits	External DDR interface clock.
clk_ssp	ref_xtal /ref_io	8 bits	SSP interface clock.
clk_gpmi	ref_xtal /ref_io	8 bits	General purpose memory interface clock domain.
clk_irov/ir	ref_io /clk_irov	9/10 bits	Over sample IR clock and IR data bit clock. The IROV clock has the ref_io as its reference. The IR clock domain uses the clk_irov domain as its reference.
clk_spdif /clk_pcmspdif	ref_xtal /ref_io		Clk_spdif is an intermediate clock that drives the clk_pcmspdif fractional clock divider.
clk_pix	ref_xtal /ref_pix	DDA	External display interface clock. Its reference is the xtal or fractional divider output that drives a DDA fractional divider.
clk_saif	ref_xtal /ref_pll	DDA	Serial Audio Interface clock domain. Its reference is the PLL clock output which drives a DDA fractional divider.
Divided clock domains referenced from Xtal clock.			
clk_x	ref_xtal	10 bits	APBX clock domain.
clk_uart	ref_xtal	2 bits	UART clock domain.
Fixed clock domains.			
clk_xtal24m	ref_xtal	24Mhz	Used for the DRI, filter, and analog 24Mhz clock domains.
clk_32k	ref_xtal32k /ref_xtal	32khz	Fixed 32khz clock domain. The reference is either the 32kHz crystal or the 24Mhz crystal and divides by 768 to produce 32kHhz.

Table 4-1. System Clocks (continued)

NAME	REFERENCE	DIVIDE /FREQ	DESCRIPTION
clk_adc	ref_xtal	2khz	Fixed 2khz clock domain.
clk_tv108m_ng	ref_vid	108Mhz	Fixed 108Mhz clk domain.
clk_tv54m	int_108m	54Mhz	Fixed 54Mhz clk domain. The reference is a gated clock on the internal fixed 108Mhz clock.
clk_tv27m	int_108m	27Mhz	Fixed 27Mhz clk domain. The reference is a gated clock on the internal fixed 108Mhz clock.
clk_tvenc_fifo	Int_108m	54Mhz/27Mhz	Selectable between 54MHz and 27MHz with control bit from tvenc block. The reference is a gated clock on the internal fixed 108Mhz clock.

4.2.2 Logical Diagram of Clock Domains

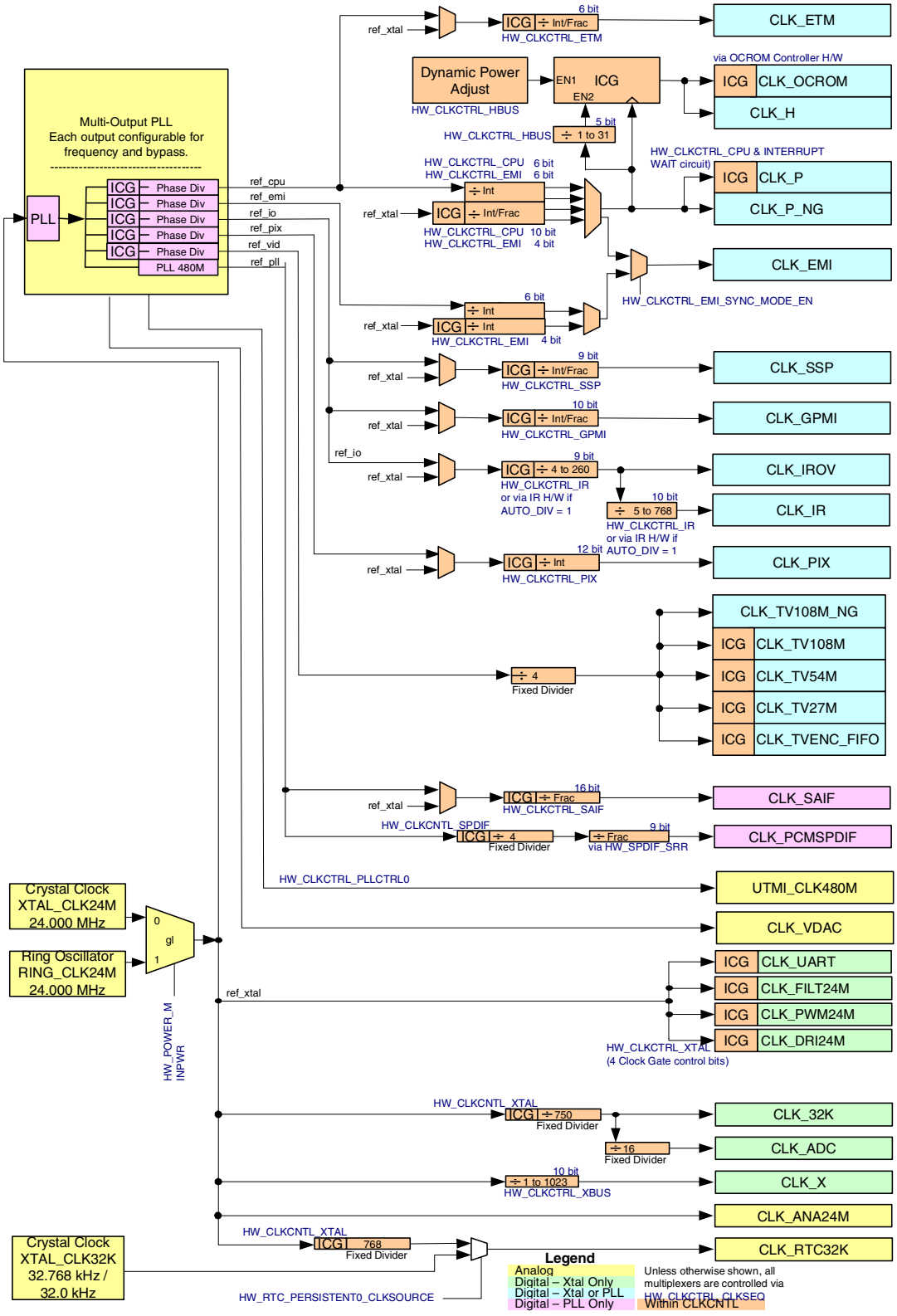


Figure 4-1. Logical Diagram of Clock Domains

4.2.3 Clock Domain Description

All major functional clock domains/branches have trunk level clock gating for power management. The intent is to gate clock domains off when modules for certain applications are not necessary. This clock gating is instantiated using an ICG element from the standard cell library. Software will have to enable the clock domain that drives on chip devices where trunk level clock gating is implemented.

The location of ICG elements to gate clock domains is not systematic. Since most of the clock structures throughout the system are unique, the location of ICGs for clock tree power reduction differs from one domain to the next. The location of these ICGs to gate off clock domains is apparent in the clock structure diagram.

All clock domains are asynchronous unless noted otherwise.

4.2.3.1 CLK_P, CLK_H

The clk_p domain is the used to drive the integrated ARM9 core. The reference for clk_p can be either ref_xtal or ref_cpu. The reference ref_cpu drives a 6 bit clock divider to provide a maximum divide down of the reference clock by 2^6. The reference ref_xtal drives a 10 bit clock divider to provide a maximum divide down of the selected reference clock by 2^10. All of the ARM core and SoC components on the clk_h branch are considered to be on the clk_p domain. clk_h is actually a branch of the clk_p domain. So, clk_h runs synchronous to clk_p.

The clk_h domain can be programmed to any divided ratio with respect to the clk_p domain depending on performance and power requirements. A dynamic clock frequency management controller monitors system performance requirements and scales the clk_h frequency to meet performance needs. When the CPU or support components require data transfer to/from system memory, the frequency manager scales the clk_h domain to meet the system performance requirements. Also, when the system is quiesed, the clk_h frequency is reduced to save power.

Clk_h has a 5 bit divider that divides the clk_p domain to produce the clk_h domain. The frequency for clk_h can be clk_p/32 <= clk_h <= clk_p. Two divide modes exist for the clk_h branch:

- Integer divide. In this mode, the value programmed in the hw_clkctrl_hbusclkctrl.div field represents an integer divide value.
- Fractional divide. In this mode, the value programmed in the hw_clkctrl_hbusclkctrl.div field represents a binary fraction. When the accumulation of the current count and the programmed divide value carry out of the most significant bit, a clk_h pulse is generated. For example, to achieve an 8:3 clk_p:clk_h clock ratio, set the div field to 0.01100 which represents (0*1/2) + (1*1/4) + (1*1/8) + (0*1/16) + (0*1/32). Note, fractional divide can not be used when clk_emi is synchronous with clk_h.

The clk_h branch can be further divided by the dynamic clock frequency adjustment logic, (hw_clkctrl_emi_sync_mode_en = 0 only). When all the system clk_h components are not busy and their respective busy signals are inactive, the clk_h branch is further divided down by the value in the

hw_clkctrl_hbusclkctrl register. The frequency reduction of the clk_h branch saves overall power con¬sumption. Note, the dynamic clock frequency adjustment logic should not be enabled when clk_emi is synchronous with clk_h.

4.2.3.2 CLK_EMI

The external memory interface domain is called clk_emi. This clock can be asynchronous to clk_h to achieve the highest possible clock rate for the EMI interface, or synchronously to minimize the incurred latency for CPU access to external DRAM. This option is provided to tradeoff the optimization of performace for systems that are dependent on memory access latency or throughput.

When the hw_clkctrl_emi_sync_mode_en bit is set to 1, clk_h is synchronous and edge aligned with the emi clock and clk_p. The emi clock dividers will set the frequency of clk_h and clk_emi domains when synchronous mode is selected. In synchronous mode, the dynamic clock frequency adjust logic should be disabled. This is required since DRAM devices cannot operate correctly with changing clock frequencies.

4.2.3.3 System Clocks

All reference clock domains used in the CLKCTRL are driven by replicated instances of the PFD pre dividers in the analog module. These PFD reference clocks drive replicated instances of a single digital clock divider design to create all system clocks. The following sections describe the digital clock dividers features and how they can be used to create clocks throughout the system. The CLKCTRL structural diagram should be used with the digital clock divider description to understand how clocks are generated in the i.MX23 system.

4.3 CLKCTRL Digital Clock Divider

The digital clock divider that is used to drive all functional clock domains has three modes of operation. These are:

- integer divide mode
- fractional divide mode
- gated clock divide mode

These modes are described in the following three sections.

4.3.1 Integer Clock Divide Mode

Each divider has the capability to divide an input reference frequency by a fixed integer value. This is the most common mode that will be used to select a particular clock frequency. For a desired clock frequency, first try to select a PFD reference clock frequency AND an integer clock divide value to achieve the desired clock domain frequency. This mode is selected when the respective “frac_en” field in the clock control register is logic 0. The divide value will be in the range of 1 to 2^N. When programming

the DIV field to 1, the reference clock for the domain is passed and the clock domain assumes the same frequency as the reference domain. When a value of 2 is programmed, the clock domain frequency will be half the reference clock frequency. The maximum divide value depends on the number of bits each digital clock divider implements. This is different for each digital clock divider. The number of bits implemented for each divider is indicated by each DIV field that controls each clock domain. Divide by zero is NOT a valid programming value for the DIV field of any clock control PIO register.

4.3.2 Fractional Clock Divide Mode

This mode is used to divide a reference clock in the range of 2 < div < 2^N. The fractional clock divider in the CLKCTRL module implements a fractional counter to approximate a divided clock with respect to the selected reference frequency. The accuracy of the output clock is depen¬dent on the extent of the bits used to implement the fractional counter. The reference clock frequency and the fractional divide value must both be selected to achieve the desired output frequency.

This mode is enabled when setting the FRAC_EN field of the respective clock domain control register to logic 1 AND the most significant bit of the DIV field is logic 0. Do NOT use this mode to divide the reference clock domain by an integer value (such as 4, 8, etc). Use the integer divide mode to achieve the best results for dividing by an integer.

NOTE

It is important to note that the nearest rising or falling edge of the input reference clock frequency is used to approximate the rising edge of the output clock domain. So, the output clock frequency will jitter based on the input reference clock frequency and the programmed fractional divide value.

4.3.2.1 Fractional Clock Divide Example, Divide by 3.5

As an example, if the desired divide value is 3.5, the digital approximation of 1/3.5 is 0.01001001 using an 8 bit fractional approximation. The most significant bit of the “div” field in this case is logic 0, so the fractional divide mode is selected. The following sequence indicates the first 8 values of the fractional clock divider. Notice the accumulated count is simply the current value incremented by the value programmed in the “div” field on each cycle.

1. 0.01001001
2. 0.10010010
3. 0.11011011
4. 1.00100100 (carry out of MSB initiates an output clock edge)
5. 0.01101101
6. 0.10110110
7. 0.11111111
8. 1.01001000 (output edge initiated)

When the carry out of the fractional count is one, a rising edge output pulse is initiated and the remainder of the accumulator is preserved. The sub fractional accumulated value is considered to determine if the output edge should occur on the falling edge of the reference clock or the rising edge of the reference clock to minimize output clock jitter

4.3.2.1.1 Fractional ClockDivide Example, Divide by 3/8

This example uses a 3 bit fractional accumulator to divide the reference clock input by 3/8. There are 3 output clock edges produced for every 8 input reference clock edges. An output edge is generated on every cycle that the fractional accumulator carries out of the most significant bit. Notice when the fractional component is .01, the output edge is shifted and generated off the falling edge of the input reference clock. This is done to produce the best output duty cycle that can be achieved based on the input reference clock frequency.

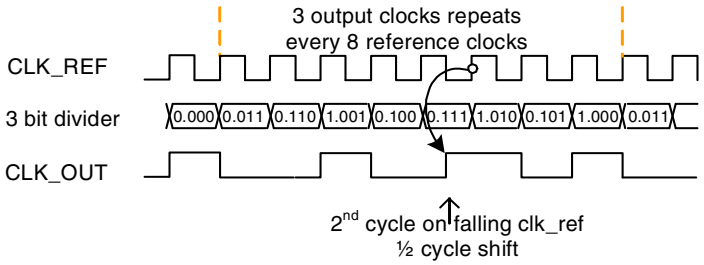


Figure 4-2. Fractional Clock divide; 3/8 example

4.3.3 Gated Clock Divide Mode

This mode is selected when the reference clock frequency is divided by a range of 1 < div < 2. To select this mode, program the FRAC_EN field to logic 1 and progarm the DIV field with the most significant bit set to logic 1. In this case, the reference clock is enabled/disabled on a cycle by cycle basis to pass to the output clock domain. Essentially, the reference clock is gated on or off depending on the cary out bit of the fractional count accumulator. This option is useful to divide the 24 MHz clock to a range between 12 to 24 MHz. The effective period is equal to the reference period since the output clock is a gated version of the reference clock. For example, a divide value of 4/3 will allow 3 consecutive pulses of the reference clock to propagate and will then gate off a single refer¬ence clock cycle. The edge to edge timing is effectively equal to the reference clock.

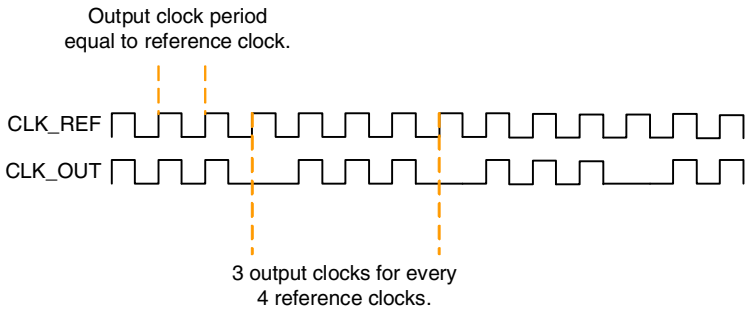


Figure 4-3. Divide Range 1 < div < 2

4.4 Clock Frequency Management

Clock frequency selection for some domains can be a function of multiple reference clock sources and divide parameters that are set in the CLKCTRL PIO control registers. The most extreme case is using a programmable fractional PLL clock divider, a multiplexer that selects either the xtal clock or fractional PLL clock as a source to drive the CLKCTRL divider, and the divide value for the CLKCTRL divider itself. When programming a selected frequency, the sequence of events to achieve a given frequency must maintain the integrity of the system as a whole. During a clock system context switch, intermediate clock frequencies for selected domains cannot be faster than the sub system or I/O interface is designed to support.

It is expected that the sequence of events when a clock domain is tuned to a desired frequency be managed by software using a hardware status polling mechanism. Each parameter has an associated enable bit so that all the divide parameters can be programmed in advance of the parameters taking effect. A single register, hw_clkctrl_clkseq, contains all the enable bits that cause the divide parameters to take effect. The enable bits can be set, the busy bits can be polled for each parameter, and thus the enable/busy sequencing via software control can manage the tuning of clock frequencies throughout the system.

4.5 Analog Clock Control

Analog clock control is performed indirectly through PIO accessable registers in the CLKCTRL module. The analog circuits that are controlled via CLKCTRL PIO access are the PLL and all instances of the phase fractional dividers, or PFDs.

4.6 CPU and EMI Clock Programming

A defined protocol is necessary for selecting clock frequencies and root sources for driving the clk_p and clk_emi domains. These two clock structures are unique in that they each implement a separate divider, one referenced by xtal clock and a second referenced by a PLL/PFD structure. The “roots” of these clocks must be programmed in order of the sources furthest from the trunk first. Elements in the clock roots should subsequently be configured along the root path up to the desired clock trunk. The programming sequence to go from a clock that is referenced from the xtal clock to the PLL is outlined below. This is the

case when the device is in low power operation and there exists the need for higher clock rates to meet the demands of a more compute-intensive application.

1. The crystal is the current source for the CPU or EMI clock domain.
2. Enable the PLL.
3. Wait for PLL lock.
4. Program and enable the PFD with the desired configuration.
5. Clear the PFD clock gate to establish the desired reference clock frequency.
6. Program the CLKCTRL clock divider register (EMI or CPU) that uses the PLL/PFD as its reference clock.
7. Switch the bypass to *off* (select PLL, not crystal).

The requirement is that the roots of the clock are configured and stable before elements higher up in the tree are programmed. This will allow the roots to stabilize before selected as a valid source to drive a clock trunk/tree. If this sequence is not honored, unpredictable frequencies can occur which may violate the maximum operating frequency of components on the respective clock trees. Be sure to gate off the clock paths directly downstream from the PLL before powering off the PLL.

When clk_emi is operating in synchronous mode, the following requirements must be maintained:

- The clk_p divide value is less than or equal to the clk_emi divide value.
- The clk_emi divide value must be divisable by the clk_p divide value. An example of possible clk_p:clk_emi divide values would be, but not limited, to 1:1, 1:2, 1:3, 2:2, 2:4, 2:6, 3:3, 3:6, 3:9.

4.7 Chip Reset

Two PIO accessible soft reset bits exists to establish the initial state of the device. These bits are called HW_CLKCTRL_RESET_CHIP and HW_CLKCTRL_RESET_DIG. Set¬ting these bits will result in a chip wide reset cycle. When setting the DIG software reset bit, the digital logic is reset with the exception of the power module and the DCDC converter control logic. The CHIP software reset bit also initiates the full reset cycle and the power and DCDC con¬verter logic are also reset. These two soft reset bits are themselves reset during a soft reset sequence.

See Figure 4-4 for reference of the functionality of these two reset bits.

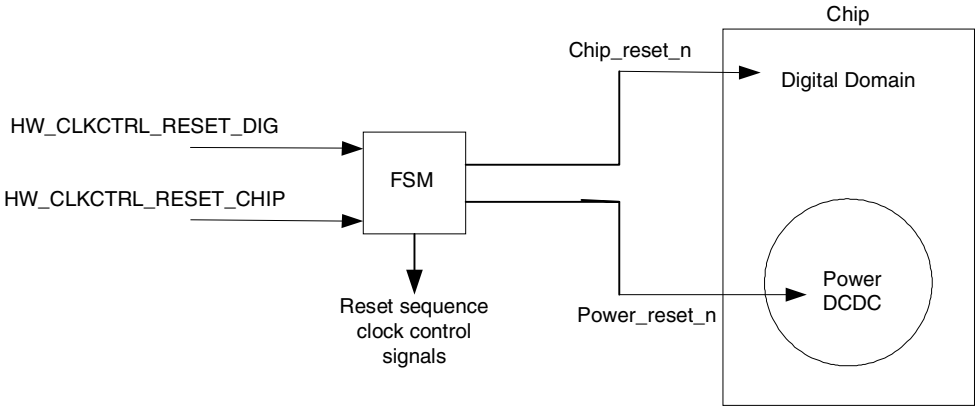


Figure 4-4. Reset Logic Functional Diagram

4.8 Programmable Registers

This section includes the programmable registers supported in the Clock Controller Module.

4.8.1 PLL Control Register 0 Description

The PLL Control Register 0 programs the 480 MHz PLL and the USB-clock enables.

HW_CLKCTRL_PLLCTRL0	0x000
HW_CLKCTRL_PLLCTRL0_SET	0x004
HW_CLKCTRL_PLLCTRL0_CLR	0x008
HW_CLKCTRL_PLLCTRL0_TOG	0x00C

Table 4-2. HW_CLKCTRL_PLLCTRL0

31 30	29 28	27 26	25 24	23 22	21 20	19	18	17	16	15 14 13 12	11 10 09 08	07 06 05 04	03 02 01 00
RSRVD6	LFR_SEL	RSRVD5	CP_SEL	RSRVD4	DIV_SEL	RSRVD3	EN_USB_CLKS	RSRVD2	POWER	RSRVD1			

Table 4-3. HW_CLKCTRL_PLLCTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	**RSRVD6**	RO	0x0	Always set to zero (0).
29:28	**LFR_SEL**	RW	0x0	TEST MODE FOR INTERNAL USE ONLY. Adjusts loop filter resistor. DEFAULT = 0x0 Default loop filter resistor TIMES_2 = 0x1 Doubles the loop filter resistor TIMES_05 = 0x2 Halves the loop filter resistor UNDEFINED = 0x3 Undefined

Table 4-3. HW_CLKCTRL_PLLCTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
27:26	**RSRVD5**	RO	0x0	Always set to zero (0).
25:24	**CP_SEL**	RW	0x0	TEST MODE FOR INTERNAL USE ONLY. Adjusts charge pump current DEFAULT = 0x0 Default charge pump current TIMES_2 = 0x1 Doubles charge pump current TIMES_05 = 0x2 Halves the charge pump current UNDEFINED = 0x3 Undefined
23:22	**RSRVD4**	RO	0x0	Always set to zero (0).
21:20	**DIV_SEL**	RW	0x0	TEST MODE FOR INTERNAL USE ONLY. This field is currently NOT supported. DEFAULT = 0x0 PLL frequency is 480 MHz LOWER = 0x1 Lower the PLL fequency from 480MHz to 384MHz LOWEST = 0x2 Lower the PLL fequency from 480MHz to 288MHz UNDEFINED = 0x3 Undefined
19	**RSRVD3**	RO	0x0	Always set to zero (0).
18	**EN_USB_CLKS**	RW	0x0	0: 8-phase PLL outputs for USB PHY are powered down. If set to 1, 8-phase PLL outputs for USB PHY are powered up. Additionally, the UTMICLK120_GATE and UTMICLK30_GATE must be deasserted in the UTMI phy to enable USB operation.
17	**RSRVD2**	RO	0x0	Always set to zero (0).
16	**POWER**	RW	0x0	PLL Power On (0 = PLL off; 1 = PLL On). Allow 10 us after turning the PLL on before using the PLL as a clock source. This is the time the PLL takes to lock to 480 MHz.
15:0	**RSRVD1**	RO	0x0	Always set to zero (0).

DESCRIPTION:

The PLL Control Register 0 programs the 480 MHz PLL and the USB-clock enables.

EXAMPLE:

```
HW_CLKCTRL_PLLCTRL0_WR(BF_CLKCTRL_PLLCTRL0_POWER(1)); // enable PLL
wait_10us; // Wait 10 us to let PLL lock before using it
```

4.8.2 PLL Control Register 1 Description

PLL Lock Control Register

HW_CLKCTRL_PLLCTRL1 0x010

Table 4-4. HW_CLKCTRL_PLLCTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
LOCK	FORCE_LOCK	RSRVD1														LOCK_COUNT															

Table 4-5. HW_CLKCTRL_PLLCTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**LOCK**	RO	0x0	PLL Lock bit. 1=PLL Locked. 0=PLL Unlocked.
30	**FORCE_LOCK**	RW	0x0	Force the PLL Lock sequence to start. 1=Enable Force Lock. This bit is not self clearing.
29:16	**RSRVD1**	RO	0x0	Reserved - Always set to zero (0).
15:0	**LOCK_COUNT**	RO	0x0	Status of the PLL lock count. The PLL lock bit will assert when the count reaches 0x4B0. The lock count is driven off of xtal, so the 50us.

DESCRIPTION:

The lock count is driven off of xtal. So after the PLL is powered on, the PLL Lock should be asserted after 50us.

EXAMPLE:

```
HW_CLKCTRL_PLLCTRL1_WR(BF_CLKCTRL_PLLCTRL1_FORCE_LOCK(1)); // force pll lock sequence
HW_CLKCTRL_PLLCTRL1_WR(BF_CLKCTRL_PLLCTRL1_FORCE_LOCK(0)); // clear force pll lock
```

4.8.3 CPU Clock Control Register Description

The CPUCLK Clock Control Register provides controls for generating the ARM CPUCLK.

HW_CLKCTRL_CPU	0x020
HW_CLKCTRL_CPU_SET	0x024
HW_CLKCTRL_CPU_CLR	0x028
HW_CLKCTRL_CPU_TOG	0x02c

Table 4-6. HW_CLKCTRL_CPU

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD6		BUSY_REF_XTAL	BUSY_REF_CPU	RSVD5	DIV_XTAL_FRAC_EN	DIV_XTAL										RSVD4			INTERRUPT_WAIT	RSVD3	RSVD2	RSVD1				DIV_CPU					

Table 4-7. HW_CLKCTRL_CPU Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	**RSVD6**	RO	0x0	Always set to zero (0).
29	**BUSY_REF_XTAL**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28	**BUSY_REF_CPU**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
27	**RSVD5**	RO	0x0	Always set to zero (0).
26	**DIV_XTAL_FRAC_EN**	RW	0x0	1 = Enable fractional divide. 0 = Enable integer divide.

Table 4-7. HW_CLKCTRL_CPU Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25:16	**DIV_XTAL**	RW	0x001	This field controls the divider connected to the crystal reference clock that drives the CLK_P domain when bypass is selected. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.
15:13	**RSVD4**	RO	0x0	Always set to zero (0).
12	**INTERRUPT_WAIT**	RW	0x0	Gate off CLK_P while waiting for an interrupt.
11	**RSVD3**	RO	0x0	Always set to zero (0).
10	**RSVD2**	RW	0x0	Program this field to 0x0.
9:6	**RSVD1**	RO	0x0	Always set to zero (0).
5:0	**DIV_CPU**	RW	0x01	This field controls the divider connected to the ref_cpu reference clock that drives the CLK_P domain when bypass is NOT selected. For changes to this field to take effect, the ref_cpu reference clock must be running. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

DESCRIPTION:

Controls for the ARM 926 clock divider. Note: Do not write register space when busy bit(s) are high.

EXAMPLE:

```
HW_CLKCTRL_CPU_WR(BF_CLKCTRL_DIV_CPU(12)); // 480 MHz / 12 = 40 MHz
```

4.8.4 AHB, APBH Bus Clock Control Register Description

The AHB, APBH Bus Clock Control Register provides controls for CLK_H generation when HW_CLKCTRL_EMI_SYNC_MODE_EN = 0.

HW_CLKCTRL_HBUS	0x030
HW_CLKCTRL_HBUS_SET	0x034
HW_CLKCTRL_HBUS_CLR	0x038
HW_CLKCTRL_HBUS_TOG	0x03c

Table 4-8. HW_CLKCTRL_HBUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4		BUSY	DCP_AS_ENABLE	PXP_AS_ENABLE	APBHDMA_AS_ENABLE	APBXDMA_AS_ENABLE	TRAFFIC_JAM_AS_ENABLE	TRAFFIC_AS_ENABLE	CPU_DATA_AS_ENABLE	CPU_INSTR_AS_ENABLE	AUTO_SLOW_MODE	RSRVD2	SLOW_DIV			RSRVD1										DIV_FRAC_EN	DIV				

Table 4-9. HW_CLKCTRL_HBUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	**RSRVD4**	RO	0x0	Reserved
29	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28	**DCP_AS_ENABLE**	RW	0x0	Enable auto-slow mode based on DCP activity. 0 = Run at the programmed CLK_H frequency.
27	**PXP_AS_ENABLE**	RW	0x0	Enable auto-slow mode based on PXP activity. 0 = Run at the programmed CLK_H frequency.
26	**APBHDMA_AS_ENABLE**	RW	0x0	Enable auto-slow mode based on APBH DMA activity. 0 = Run at the programmed CLK_H frequency.
25	**APBXDMA_AS_ENABLE**	RW	0x0	Enable auto-slow mode based on APBX DMA activity. 0 = Run at the programmed CLK_H frequency.
24	**TRAFFIC_JAM_AS_ENABLE**	RW	0x0	Enable auto-slow mode when less than three masters are trying to use the AHB. More than three active masters will engage the default mode. 0 = Run at the programmed CLK_H frequency.
23	**TRAFFIC_AS_ENABLE**	RW	0x0	Enable auto-slow mode based on AHB master activity. 0 = Run at the programmed CLK_H frequency.
22	**CPU_DATA_AS_ENABLE**	RW	0x0	Enable auto-slow mode based on with CPU Data access to AHB. 0 = Run at the programmed CLK_H frequency.
21	**CPU_INSTR_AS_ENABLE**	RW	0x0	Enable auto-slow mode based on with CPU Instruction access to AHB. 0 = Run at the programmed CLK_H frequency.
20	**AUTO_SLOW_MODE**	RW	0x0	Enable CLK_H auto-slow mode. When this is set, then CLK_H will run at the slow rate until one of the fast mode events has occurred. Note: The AUTO_SLOW_MODE bit must be cleared before writing to the SLOW_DIV bitfield.
19	**RSRVD2**	RO	0x0	Reserved
18:16	**SLOW_DIV**	RW	0x0	Slow mode divide ratio. Sets the ratio of CLK_H fast rate to the slow rate. Note: The AUTO_SLOW_MODE bit must be cleared before writing to the SLOW_DIV bitfield. BY1 = 0x0 Slow mode divide ratio = 1 BY2 = 0x1 Slow mode divide ratio = 2 BY4 = 0x2 Slow mode divide ratio = 4 BY8 = 0x3 Slow mode divide ratio = 8 BY16 = 0x4 Slow mode divide ratio = 16 BY32 = 0x5 Slow mode divide ratio = 32
15:6	**RSRVD1**	RO	0x0	Reserved
5	**DIV_FRAC_EN**	RW	0x0	1 = Enable fractional divide. 0 = Enable integer divide.
4:0	**DIV**	RW	0x01	CLK_P-to-CLK_H divide ratio. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

DESCRIPTION:

This register controls the clock divider that generates the CLK_H, the clock used by the AHB and APBH buses, when HW_CLKCTRL_EMI_SYNC_MODE_EN = 0.

Note: Do not write register space when busy bit(s) are high.

EXAMPLE:

```
HW_CLKCTRL_HBUS_WR(BF_CLKCTRL_HBUS_DIV(2)); // set CLK_H to half the ARM clock (CLK_P) frequency
```

4.8.5 APBX Clock Control Register Description

The APBX Clock Control Register provides control of the CLK_X clock divider.

HW_CLKCTRL_XBUS	0x040

Table 4-10. HW_CLKCTRL_XBUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BUSY	RSVD2																				RSVD1	DIV									

Table 4-11. HW_CLKCTRL_XBUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
30:11	**RSVD2**	RO	0x0	Always set to zero (0).
10	**RSVD1**	RW	0x0	Program this field to 0x0.
9:0	**DIV**	RW	0x001	This field controls the CLK_X divide ratio. CLK_X is sourced from the 24-MHz XTAL through this divider. Do NOT divide by 0.

DESCRIPTION:

This register controls the clock divider that generates the CLK_X, the clock used by the APBX bus.

Note: Do not write register space when busy bit(s) are high.

EXAMPLE:

```
HW_CLKCTRL_XBUS_WR(BF_CLKCTRL_XBUS_DIV(4)); // set apbx xbus clock to 1/4 the 24.0MHz crystal clock
frequency
```

4.8.6 XTAL Clock Control Register Description

The XTAL control register provides gating control for clocks sourced from the 24-MHz XTAL clock domain.

HW_CLKCTRL_XTAL	0x050
HW_CLKCTRL_XTAL_SET	0x054
HW_CLKCTRL_XTAL_CLR	0x058
HW_CLKCTRL_XTAL_TOG	0x05C

Table 4-12. HW_CLKCTRL_XTAL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UART_CLK_GATE	FILT_CLK24M_GATE	PWM_CLK24M_GATE	DRI_CLK24M_GATE	DIGCTRL_CLK1M_GATE	TIMROT_CLK32K_GATE	RSRVD1																								DIV_UART	

Table 4-13. HW_CLKCTRL_XTAL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**UART_CLK_GATE**	RW	0x0	If set to 1, fixed 24-MHz clock for the UART, CLK_UART, is gated off.
30	**FILT_CLK24M_GATE**	RW	0x1	If set to 1, fixed 24-MHz clock for the Digital Filter, CLK_FILT24M, is gated off.
29	**PWM_CLK24M_GATE**	RW	0x1	If set to 1, fixed 24-MHz clock for the PWM, CLK_PWM24M, is gated off.
28	**DRI_CLK24M_GATE**	RW	0x1	If set to 1, fixed 24-MHz clock for the Digital Radio Interface (DRI), CLK_DRI24M, is gated off.
27	**DIGCTRL_CLK1M_GATE**	RW	0x0	If set to 1, fixed 1-MHz clock for DIGCTRL, CLK_1M, is gated off.
26	**TIMROT_CLK32K_GATE**	RW	0x0	If set to 1, fixed 32-kHz clock for the TIMROT block, CLK_32K, is gated off.
25:2	**RSRVD1**	RO	0x0	Always set to zero (0).
1:0	**DIV_UART**	RW	0x1	Reserved - Always set to one (1)

DESCRIPTION:

This register controls various fixed-rate divider clocks working off the 24.0-MHz crystal clock.

EXAMPLE:

```
HW_CLKCTRL_XTAL_WR(BF_CLKCTRL_XTAL_UART_CLK_GATE(0)|BF_CLKCTRL_XTAL_DRI_CLK24M_GATE(1));
```

4.8.7 PIX (LCDIF) Clock Control Register Description

The PIX control register provides control for LCDIF clock generation.

HW_CLKCTRL_PIX 0x060

Table 4-14. HW_CLKCTRL_PIX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	RSRVD2	BUSY	RSRVD1																DIV_FRAC_EN	DIV											

Table 4-15. HW_CLKCTRL_PIX Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	CLK_PIX Gate. If set to 1, CLK_PIX is gated off. 0: CLK_PIX is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30	**RSRVD2**	RO	0x0	Always set to zero (0).
29	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28:13	**RSRVD1**	RO	0x0	Always set to zero (0).
12	**DIV_FRAC_EN**	RW	0x0	Reserved - Always set to zero (0).
11:0	**DIV**	RW	0x1	The Pixel clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_pix) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0. Do not divide by more than 255.

DESCRIPTION:

This register controls the divider that generates the PIX (LCDIF) clock.

Note: Do not write register space when busy bit(s) are high.

EXAMPLE:

```
HW_CLKCTRL_PIX_WR(BF_CLKCTRL_PIX_DIV(40));
```

4.8.8 Synchronous Serial Port Clock Control Register Description

The SSP control register provides control for SSP clock generation.

HW_CLKCTRL_SSP 0x070

Table 4-16. HW_CLKCTRL_SSP

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	RSVD3	BUSY	RSVD2																			RSVD1	DIV								

Table 4-17. HW_CLKCTRL_SSP Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	CLK_SSP Gate. If set to 1, CLK_SSP is gated off. 0: CLK_SSP is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30	**RSVD3**	RO	0x0	Always set to zero (0).
29	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28:10	**RSVD2**	RO	0x0	Always set to zero (0).
9	**RSVD1**	RW	0x0	Program this field to 0x0.
8:0	**DIV**	RW	0x1	The synchronous serial port clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_io) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

DESCRIPTION:

This register controls the clock divider that generates the clock for the synchronous serial port (SSP), CLK_SSP.

Note: Do not write register space when busy bit(s) are high.

EXAMPLE:

```
HW_CLKCTRL_SSP_WR(BF_CLKCTRL_SSP_DIV(40));
```

4.8.9 General-Purpose Media Interface Clock Control Register Description

The GPMI control register provides control for GPMI clock generation.

HW_CLKCTRL_GPMI 0x080

Table 4-18. HW_CLKCTRL_GPMI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	RSVD3	BUSY	RSVD2																		RSVD1	DIV									

Table 4-19. HW_CLKCTRL_GPMI Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	CLK_GPMI Gate. If set to 1, CLK_GPMI is gated off. 0: CLK_GPMI is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30	**RSVD3**	RO	0x0	Always set to zero (0).
29	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28:11	**RSVD2**	RO	0x0	Always set to zero (0).
10	**RSVD1**	RW	0x0	Program this field to 0x0.
9:0	**DIV**	RW	0x1	The GPMI clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_io) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

DESCRIPTION:

This register controls the divider that generates the General-Purpose Media Interface (GPMI) clock, CLK_GPMI.

Note: Do not write register space when busy bit(s) are high.

EXAMPLE:

```
HW_CLKCTRL_GPMI_WR(BF_CLKCTRL_GPMI_DIV(40));
```

4.8.10 SPDIF Clock Control Register Description

The SPDIF control register provides control for SPDIF clock generation.

HW_CLKCTRL_SPDIF 0x090

Table 4-20. HW_CLKCTRL_SPDIF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	RSRVD																														

Table 4-21. HW_CLKCTRL_SPDIF Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	CLK_PCMSPDIF Gate. If set to 1, CLK_PCMSPDIF is gated off. 0: CLK_PCMSPDIF is not gated. When this bit is modified, or when it is high, the SPDIF rate change field should not change its value. The SPDIF rate change field can change ONLY when this clock gate bit field is low.
30:0	**RSRVD**	RO	0x0	Always set to zero (0).

DESCRIPTION:

This register controls the clock gate on the SPDIF clock, CLK_PCMSPDIF.

EXAMPLE:

```
HW_CLKCTRL_SPDIF_WR(BF_CLKCTRL_SPDIF_CLKGATE(1));
```

4.8.11 EMI Clock Control Register Description

The EMi control register provides control for External Memory Interface clock generation.

HW_CLKCTRL_EMI 0x0a0

Table 4-22. HW_CLKCTRL_EMI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	SYNC_MODE_EN	BUSY_REF_XTAL	BUSY_REF_EMI	BUSY_REF_CPU	BUSY_SYNC_MODE	RSVD5								RSVD4	RSVD3	RSVD2					DIV_XTAL				RSVD1		DIV_EMI					

Table 4-23. HW_CLKCTRL_EMI Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	CLK_EMI crystal divider Gate. If set to 1, the EMI_CLK divider that is sourced by the crystal reference clock, ref_xtal, is gated off. 0: CLK_EMI crystal divider is not gated
30	**SYNC_MODE_EN**	RW	0x0	If set to 1, EMI_CLK is synchronous with the APBH clock. If set to 0, EMI_CLK is aysnchronous. In synchronous operation, the EMI clock dividers control both EMI_CLK and APBH clock. Both xtal and ref_cpu must be active to switch between asynchronous/synchronous operation.
29	**BUSY_REF_XTAL**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains. This bit is valid when HW_CLKCTRL_EMI_SYNC_MODE_EN = 0.
28	**BUSY_REF_EMI**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
27	**BUSY_REF_CPU**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains. This bit is valid when HW_CLKCTRL_EMI_SYNC_MODE_EN = 1.
26	**BUSY_SYNC_MODE**	RO	0x0	This read-only bit field returns a one when there is a change in HW_CLKCTRL_EMI_SYNCE_MODE_EN or when there is a change in HW_CLKCTRL_CLKSEQ_BYPASS_CPU and HW_CLKCTRL_EMI_SYNCE_MODE_EN is set. When this bit returns a one, do not change the CPU or EMI divider values.
25:18	**RSVD5**	RO	0x0	Always set to zero (0).
17	**RSVD4**	RO	0x0	Program this field to 0x0.
16	**RSVD3**	RW	0x0	Program this field to 0x0.
15:12	**RSVD2**	RO	0x0	Always set to zero (0).
11:8	**DIV_XTAL**	RW	0x1	This field controls the divider connected to the crystal reference clock, ref_xtal, that drives the CLK_EMI domain when bypass IS selected. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.
7:6	**RSVD1**	RO	0x0	Always set to zero (0).
5:0	**DIV_EMI**	RW	0x1	This field controls the divider connected to the ref_emi reference clock that drives the CLK_EMI domain when bypass IS NOT selected. For changes to this field to take effect, the ref_emi reference clock must be running. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

DESCRIPTION:

This register controls the clock dividers that generate the External Memory Interface (EMI) clock.

Note: Do not write register space when busy bit(s) are high.

EXAMPLE:

```
HW_CLKCTRL_EMI_WR(BF_CLKCTRL_EMI_DIV_XTAL(1));
```

4.8.12 SAIF Clock Control Register Description

The SAIF control register provides control for SAIF clock generation.

HW_CLKCTRL_SAIF 0x0c0

Table 4-24. HW_CLKCTRL_SAIF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	RSRVD2	BUSY	RSRVD1												DIV_FRAC_EN	DIV															

Table 4-25. HW_CLKCTRL_SAIF Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	CLK_SAIF Gate. If set to 1, CLK_SAIF is gated off. 0: CLK_SAIF is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30	**RSRVD2**	RO	0x0	Always set to zero (0).
29	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28:17	**RSRVD1**	RO	0x0	Always set to zero (0).
16	**DIV_FRAC_EN**	RW	0x0	Reserved - Always set to one (1) - Notice this is not the reset value.
15:0	**DIV**	RW	0x1	The SAIF clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_pll) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

DESCRIPTION:

This register controls the divider that generates the Serial Audio Interface (SAIF) clock.

Note: Do not write register space when busy bit(s) are high.

EXAMPLE:

```
HW_CLKCTRL_SAIF_WR(BF_CLKCTRL_SAIF_DIV(40));
```

4.8.13 TV Encode Clock Control Register Description

The TV control register provides control for TV Encoder clock generation.

HW_CLKCTRL_TV 0x0d0

Table 4-26. HW_CLKCTRL_TV

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLK_TV108M_GATE	CLK_TV_GATE	RSRVD																													

Table 4-27. HW_CLKCTRL_TV Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLK_TV108M_GATE**	RW	0x1	If set to 1, fixed 108-MHz clock for the TV Component Video is gated off. 0: CLK_TV108M, is not gated.
30	**CLK_TV_GATE**	RW	0x1	If set to 1, fixed 54-MHz and 27-MHz clocks for the TV Encoder are gated off. 0: CLK_TV54M and CLK_TV27M, are not gated.
29:0	**RSRVD**	RO	0x0	Always set to zero (0).

DESCRIPTION:

This register controls various video divider clocks.

EXAMPLE:

```
HW_CLKCTRL_CLK_TV108M_GATE_WR(BF_CLKCTRL_CLK_TV108M_GATE(0));
```

4.8.14 ETM Clock Control Register Description

The ETM control register provides control for ETM clock generation.

HW_CLKCTRL_ETM 0x0e0

Table 4-28. HW_CLKCTRL_ETM

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	RSRVD2	BUSY	RSRVD1																						DIV_FRAC_EN	DIV					

Table 4-29. HW_CLKCTRL_ETM Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	CLK_ETM Gate. If set to 1, CLK_ETM is gated off. 0: CLK_ETM is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30	**RSRVD2**	RO	0x0	Always set to zero (0).
29	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28:7	**RSRVD1**	RO	0x00000	Always set to zero (0).
6	**DIV_FRAC_EN**	RW	0x0	Reserved - Always set to zero (0).
5:0	**DIV**	RW	0x01	The Pixel clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_cpu) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

DESCRIPTION:

This register controls the divider that generates the ETM clock.

Note: Do not write register space when busy bit(s) are high.

EXAMPLE:

```
HW_CLKCTRL_ETM_WR(BF_CLKCTRL_ETM_DIV(4));
```

4.8.15 Fractional Clock Control Register Description

The FRAC control register provides control for PFD clock generation. NOTE: Only byte accesses are supported. When using DWORD accesses to this regiester, the PFD update sequence will commence for all four PFDs controlled by this register. This may be undersirable if only one of the PFD divide values need to be updated. Only access individual bytes within this register in a single PIO access.

HW_CLKCTRL_FRAC	0x0f0
HW_CLKCTRL_FRAC_SET	0x0f4
HW_CLKCTRL_FRAC_CLR	0x0f8
HW_CLKCTRL_FRAC_TOG	0x0fC

Table 4-30. HW_CLKCTRL_FRAC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATEIO	IO_STABLE	IOFRAC						CLKGATEPIX	PIX_STABLE	PIXFRAC						CLKGATEEMI	EMI_STABLE	EMIFRAC						CLKGATECPU	CPU_STABLE	CPUFRAC					

Table 4-31. HW_CLKCTRL_FRAC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATEIO**	RW	0x1	IO Clock Gate. If set to 1, the IO fractional divider clock (reference PLL ref_io) is off (power savings). 0: IO fractional divider clock is enabled.
30	**IO_STABLE**	RO	0x0	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
29:24	**IOFRAC**	RW	0x12	This field controls the IO clocks fractional divider. The resulting frequency shall be 480 * (18/IOFRAC) where IOFRAC = 1-35.
23	**CLKGATEPIX**	RW	0x1	PIX Clock Gate. If set to 1, the PIX fractional divider clock (reference PLL ref_pix) is off (power savings). 0: PIX fractional divider clock is enabled.
22	**PIX_STABLE**	RO	0x0	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
21:16	**PIXFRAC**	RW	0x12	This field controls the pixel clock fractional divider. The resulting frequency shall be 480 * (18/PIXFRAC) where PIXFRAC = 1-35.
15	**CLKGATEEMI**	RW	0x1	EMI Clock Gate. If set to 1, the EMI fractional divider clock (reference PLL ref_emi) is off (power savings). 0: EMI fractional divider clock is enabled.
14	**EMI_STABLE**	RO	0x0	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divide should become stable quickly enough that this field will never need to be used by either device driver or application code. This value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13:8	**EMIFRAC**	RW	0x12	This field controls the EMI clock fractional divider. The resulting frequency shall be 480 * (18/EMIFRAC) where EMIFRAC = 1-35.
7	**CLKGATECPU**	RW	0x1	CPU Clock Gate. If set to 1, the CPU fractional divider clock (reference PLL ref_cpu) is off (power savings). 0: CPU fractional divider clock is enabled.

Table 4-31. HW_CLKCTRL_FRAC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
6	**CPU_STABLE**	RO	0x0	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divide should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
5:0	**CPUFRAC**	RW	0x12	This field controls the CPU clock fractional divider. The resulting frequency shall be 480 * (18/CPUFRAC) where CPUFRAC = 1-35.

DESCRIPTION:

This register controls the 9-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers.

EXAMPLE:

```
HW_CLKCTRL_FRAC_WR(BF_CLKCTRL_FRAC_CPUFRAC(4));
```

4.8.16 Fractional Clock Control Register 1 Description

The FRAC1 control register provides control for PFD clock generation.

HW_CLKCTRL_FRAC1	0x100
HW_CLKCTRL_FRAC1_SET	0x104
HW_CLKCTRL_FRAC1_CLR	0x108
HW_CLKCTRL_FRAC1_TOG	0x10C

Table 4-32. HW_CLKCTRL_FRAC1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATEVID	VID_STABLE	RSRVD1																													

Table 4-33. HW_CLKCTRL_FRAC1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATEVID**	RW	0x1	432 MHz PLL Clock Gate. If set to 1, the 432 MHz fractional divider clock (reference PLL ref_vid) is off (power savings). 0: IO fractional divider clock is enabled.
30	**VID_STABLE**	RO	0x0	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
29:0	**RSRVD1**	RO	0x0	Always set to zero (0).

DESCRIPTION:

This register controls the 9-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers.

EXAMPLE:

```
HW_CLKCTRL_FRAC1_WR(BF_CLKCTRL_FRAC1_CLKGATEVID(0));
```

4.8.17 Clock Frequency Sequence Control Register Description

The CLKSEQ control register provides control for switching between XTAL and PLL clock generation.

HW_CLKCTRL_CLKSEQ	0x110
HW_CLKCTRL_CLKSEQ_SET	0x114
HW_CLKCTRL_CLKSEQ_CLR	0x118
HW_CLKCTRL_CLKSEQ_TOG	0x11c

Table 4-34. HW_CLKCTRL_CLKSEQ

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																							BYPASS_ETM	BYPASS_CPU	BYPASS_EMI	BYPASS_SSP	BYPASS_GPMI	BYPASS_IR	RSRVD0	BYPASS_PIX	BYPASS_SAIF

Table 4-35. HW_CLKCTRL_CLKSEQ Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:9	**RSRVD1**	RO	0x0	Always set to zero (0).
8	**BYPASS_ETM**	RW	0x1	ETM bypass select. 1 = Select ref_xtal path to generate the ETM clock domain. 0 = Select ref_cpu path to generate the ETM clock domain. PLL and 9-phase fractional divider must be configured when this bit is cleared.
7	**BYPASS_CPU**	RW	0x1	CPU bypass select. 1 = Select ref_xtal path to generate the CPU and APBH clock domains. 0 = Select ref_cpu path to generate the CPU and APBH clock domains. PLL and 9-phase fractional divider must be configured when this bit is cleared.
6	**BYPASS_EMI**	RW	0x1	EMI bypass select. 1 = Select ref_xtal path to generate the EMI clock domain. 0 = Select ref_emi path to generate the EMI clock domain. PLL and 9-phase fractional divider must be configured when this bit is cleared.
5	**BYPASS_SSP**	RW	0x1	SSP bypass select. 1 = Select ref_xtal path to generate the SSP clock domain. 0 = Select ref_io path to generate the SSP clock domain. PLL and 9-phase fractional divider must be configured when this bit is cleared.
4	**BYPASS_GPMI**	RW	0x1	GPMI bypass select. 1 = Select ref_xtal path to generate the GPMI clock domain. 0 = Select ref_io path to generate the GPMI clock domain. PLL and 9-phase fractional divider must be configured when this bit is cleared.
3	**BYPASS_IR**	RW	0x1	IR bypass select. 1 = Select ref_xtal path to generate the IR clock domain. 0 = Select ref_io path to generate the IR clock domain. PLL and 9-phase fractional divider must be configured when this bit is cleared.
2	**RSRVD0**	RO	0x0	Always set to zero (0).
1	**BYPASS_PIX**	RW	0x1	PIX bypass select. 1 = Select ref_xtal path to generate the PIX clock domain. 0 = Select ref_pix path to generate the PIX clock domain. PLL and 9-phase fractional divider must be configured when this bit is cleared.
0	**BYPASS_SAIF**	RW	0x1	Reserved - Always set to zero (0) - Notice this is not the reset value.

DESCRIPTION:

This register controls the selection of clock sources (ref_xtal or ref_*) for various clock dividers.

EXAMPLE:

```
HW_CLKCTRL_CLKSEQ_WR(BF_CLKCTRL_CLKSEQ_BYPASS_IR(1));
```

4.8.18 System Software Reset Register Description

The RESET control register provides control for soft reset.

HW_CLKCTRL_RESET 0x120

Table 4-36. HW_CLKCTRL_RESET

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD																														CHIP	DIG

Table 4-37. HW_CLKCTRL_RESET Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:2	**RSRVD**	RO	0x0	Always set to zero (0).
1	**CHIP**	RW	0x0	Setting this bit to a logic one will reset the ENTIRE chip, no exceptions. This bit will also be reset after the full chip reset cycle completes.
0	**DIG**	RW	0x0	Setting this bit to a logic one will reset the digital sections of the chip. The DCDC and power module will not be reset. This bit will also be reset after the reset cycle completes.

DESCRIPTION:

This register controls full chip reset generation.

EXAMPLE:

```
HW_CLKCTRL_RESET_WR(BF_CLKCTRL_RESET_ALL(1));
```

4.8.19 ClkCtrl Status Description

The STATUS control register provides read only status of the CPU frequecy limits.

HW_CLKCTRL_STATUS 0x130

Table 4-38. HW_CLKCTRL_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CPU_LIMIT	RSRVD																														

Table 4-39. HW_CLKCTRL_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	**CPU_LIMIT**	RO	0x00	CPU Limiting. 00: full cpu frequency, 01: limit cpu frequency to 411.43 MHz, 10: limit cpu frequency to 360 MHz, 11: limit cpu frequency to 320 MHz
29:0	**RSRVD**	RO	0x0	Always set to zero (0).

DESCRIPTION:

This register indicates the CPU Frequency limit.

EXAMPLE:

```
HW_CLKCTRL_STATUS_RD(BF_CLKCTRL_STATUS_CPU_LIMIT());
```

4.8.20 ClkCtrl Version Description

The VERSION control register is a read only status of the clkctrl block version.

HW_CLKCTRL_VERSION 0x140

Table 4-40. HW_CLKCTRL_VERSION

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MAJOR								MINOR								STEP															

Table 4-41. HW_CLKCTRL_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**MAJOR**	RO	0x4	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	**MINOR**	RO	0x0	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	**STEP**	RO	0x0	Fixed read-only value reflecting the stepping of the RTL version.

DESCRIPTION:

This register indicates the RTL version in use.

EXAMPLE:

```
HW_CLKCTRL_VERSION_RD(BF_CLKCTRL_VERSION_MAJOR());
```

CLKCTRL Block v4.0, Revision 1.48

Chapter 5
Interrupt Collector

This chapter describes the interrupt control features of the i.MX23 and includes sections on interrupt nesting, FIQ generation, and CPU wait-for-interrupt mode. Table 5-1 lists all of the interrupt sources available on the i.MX23. Programmable registers for interrupt generation and control are described in Section 5.4, “Programmable Registers.”

5.1 Overview

The ARM9 CPU core has two interrupt input lines, IRQ and FIQ. As shown in Figure 5-1, the Interrupt Collector (ICOLL) can steer any of 128 interrupt sources to either the the FIQn or IRQn lines of the ARM9 CPU.

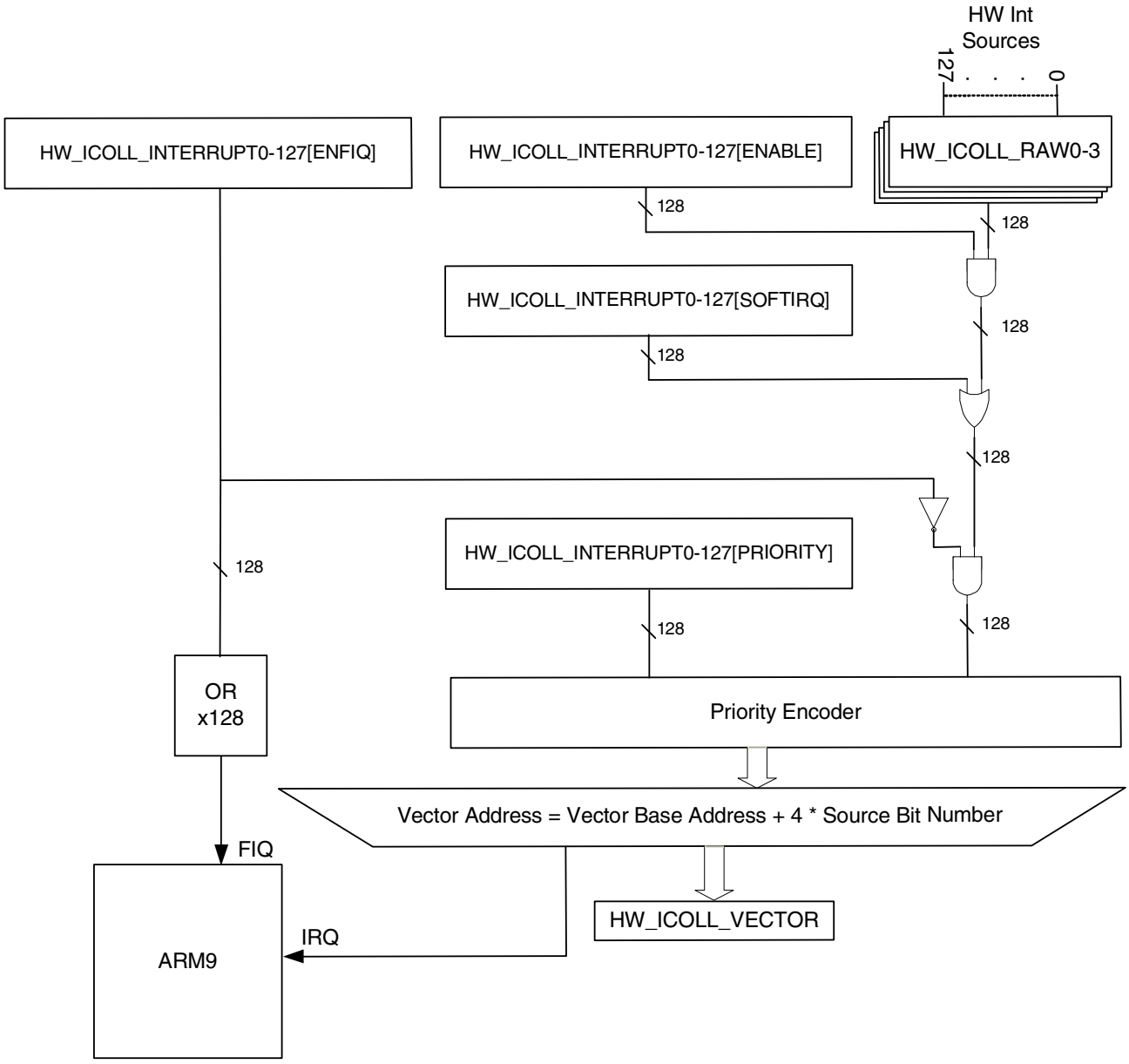


Figure 5-1. Interrupt Collector System Diagram

5.2 Operation

Within an individual interrupt request line (IRQ only), the ICOLL offers four-level priority (above base level) for each of its interrupt sources. Preemption of a lower priority interrupt by a higher priority is supported (interrupt nesting). Interrupts assigned to the same level are serviced in a strict linear priority order within level from lowest to highest interrupt source bit number. FIQ interrupts are not prioritized, nor are they vectorized. All interrupt lines can be configured as a FIQ. If more than one is routed to the FIQ, then they must be discriminated by software. It is highly recommended to reserve FIQ assignment to time critical events such as voltage brownouts or timers.

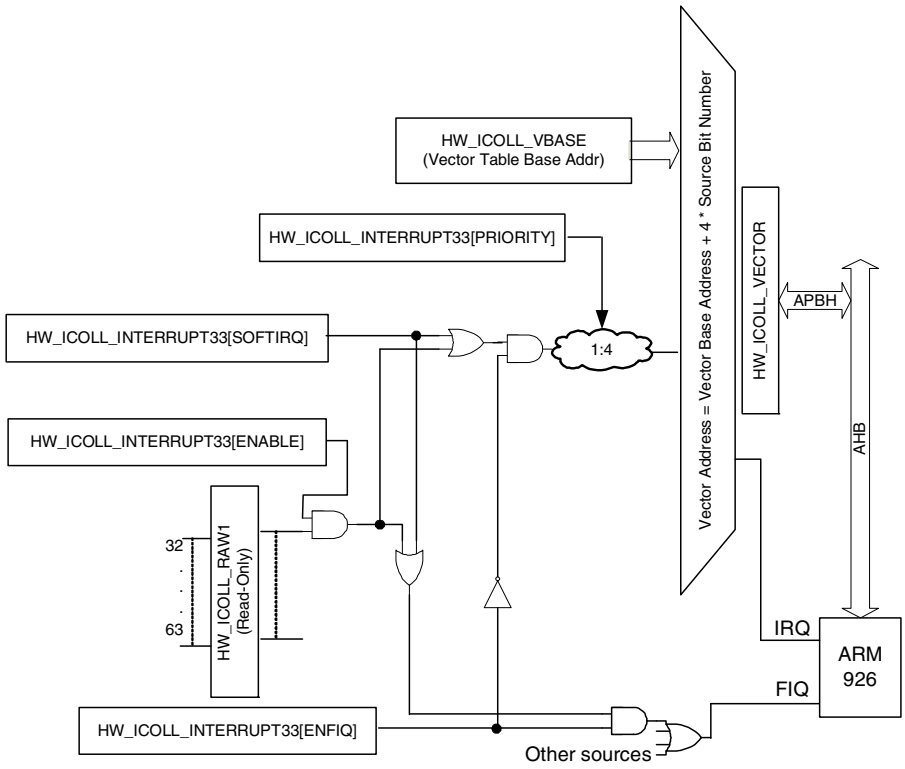


Figure 5-2. Interrupt Collector IRQ/FIQ Logic for Source 33

For a single interrupt source bit, there is an enable bit that gates it to the priority logic (HW_ICOLL_INTERRUPTn[ENABLE]). A software interrupt bit per source bit can be used to force an interrupt at the appropriate priority level directed to the corresponding vector address. Each source can be applied to one of four interrupt levels.

The enable bit, FIQ-enable, the software interrupt bit, and the two-bit priority level specification for each interrupt source bit are contained with a single programmable register for each interrupt. The path from any interrupt source to the FIQ or IRQ logic is shown in Figure 5-2 using HW_ICOLL_INTERRUPT33 as an example.

The data path for generating the vector address (readable by software) for the IRQ generation portion of the interrupt collector is implemented as a multicycle path, as shown in Figure 5-3. The interrupt sources are continuously sampled in the holding register until one or more arrive. The FSM causes the holding register to stop sampling while a vector address is computed. Each interrupt source bit is applied to one of four levels based on the two-bit priority specification of each source bit. When the holding register "closes," there can be more than one newly arrived source bit. Thus, the source bits could be assigned such that more than one interrupt level is requesting an interrupt. The pipeline first determines the highest level requesting interrupt service. All interrupt requests on that level are presented to the linear priority encoder. The result of this stage is a six-bit number corresponding to the source bit number of the highest priority requesting an interrupt. This six-bit source number is used to compute the vector address as follows:

VectorAddress = VectorBase + (Pitch * SourceBitNumber)

Pitch = 4,8, 12,16,20,24, or 28 as desired, see HW_CTRL_VECTOR_PITCH.

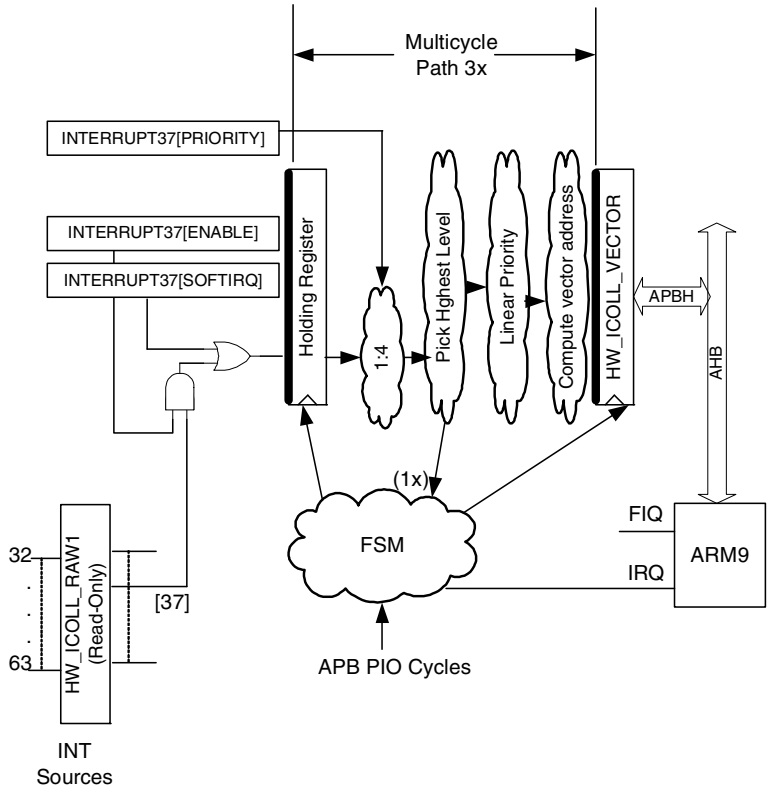


Figure 5-3. IRQ Control Flow

5.2.1 Nesting of Multi-Level IRQ Interrupts

There are a number of very important interactions between the interrupt collector's FSM and the interrupt service routine (ISR) running on the CPU. See Figure 5-4 for the following discussion.

As soon as the interrupt source is recognized in the holding register, the FSM delays two clocks, then grabs the vector address and asserts IRQ to the CPU. As soon as possible after the CPU enters the interrupt service routine, it must notify the interrupt collector. Software indicates the in-service state by writing to the HW_ICOLL_VECTOR register. The contents of the data bus on this write do not matter. Optionally, firmware can enable the ARM read side-effect mode. In this case, the in-service state is indicated as a side effect of having read the HW_ICOLL_VECTOR register at the exception vector (0xFFFF0018). At this point, the FSM reopens the holding register and scans for new interrupt sources. Any such IRQ sources are presented to the CPU, provided that they are at a level higher than any currently in-service level.

Whenever the ARM CPU takes an IRQ exception, it turns off the IRQ enable in the CPU status register (CSR), as shown in Figure 5-4. If a higher priority interrupt is pending at this point, then another IRQ exception is taken.

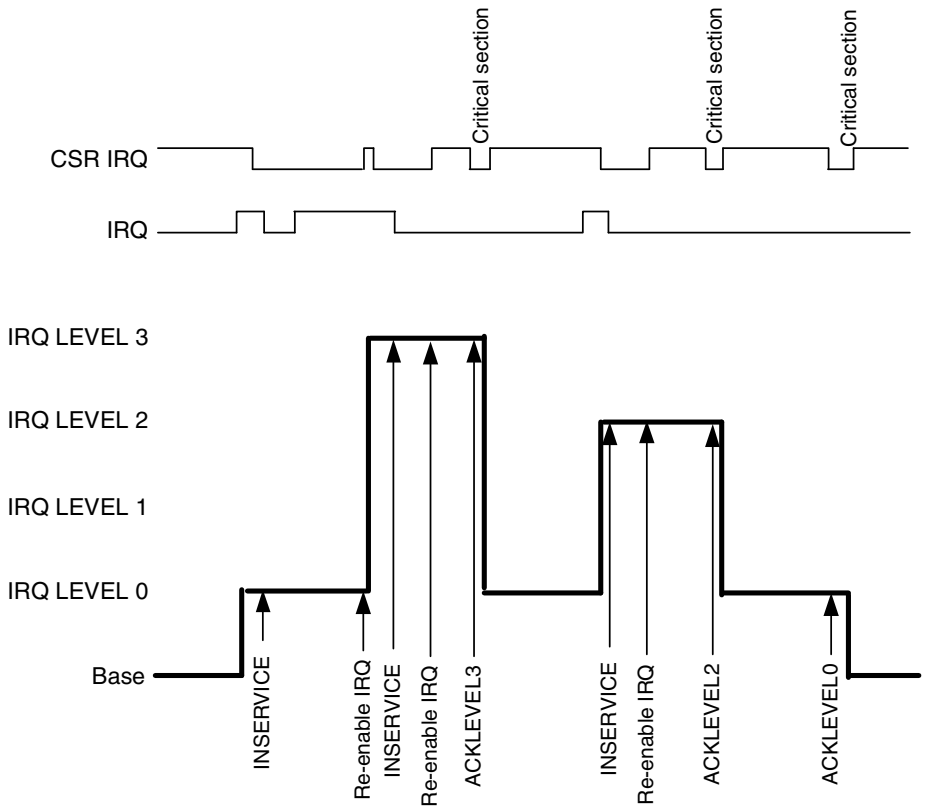


Figure 5-4. Nesting of Multi-Level IRQ Interrupts

The example in Figure 5-4 shows going from the base to a level 0 ISR. When the ISR at level 0 was ready, it enabled IRQ interrupts. At this point, it nests IRQ interrupts up to a level 3 interrupt. The level 3 ISR marks its in-service state, which causes the interrupt collector to open the holding register to search for new interrupt sources. In this example, none comes in, so the level 3 ISR completes. As part of the return process, the ISR disables IRQ interrupts, then acknowledges the level 3 service state. This is accomplished by writing the level number (3 in this case) to the interrupt collector's Level Acknowledge register. The interrupt collector resets the in-service bit for level 3. If this enables an IRQ at level 3, then it asserts IRQ and goes through the nesting process again. Since IRQ exceptions are masked in the level 3 ISR, this nesting does not take place until the level 3 ISR returns from interrupt. This return automatically re-enables IRQ exceptions. At this point, another exception could occur.

Figure 5-4 shows a second nesting of the IRQ interrupt by the arrival of a level 2 interrupt source bit. Finally, the figure shows the point at which the level 0 ISR enters its critical section (masks IRQ) and acknowledges level 0 to the interrupt collector and returns from interrupt.

The FSM reverts to its "BASE" level state waiting for an interrupt request to arrive in the holding register. The waveform for the IRQ mask in the CPU status register (CSR) and the waveform for the IRQ input to the CPU as they relate to the interrupt collector action are shown in Figure 5-4.

WARNING: There is an inherent race condition between notifying the interrupt collector that an ISR has been entered and having that ISR re-enable IRQ exceptions in the CSR. The in-service notification can take a number of cycles to percolate through the write buffer, through the AHB and APB bridge and into the interrupt collector where it removes the IRQ assertion to the CPU. This ICOLL IRQ must be deasserted before the CSR IRQ on the CPU is re-enabled or the CPU will see a phantom interrupt. This is why the ARM vectored interrupt controller provides this in service notification as a read side effect of the vector address read. Alternatively, the ISR can read the interrupt collector's CSR. The value received is unimportant, but the time required to do the read ensures that the write data has arrived at the interrupt collector. If firmware uses this method, it should allow clocks after the read for the FSM and for the CPU to recognize that the IRQ has been deasserted.

5.2.2 FIQ Generation

On i.MX23, all interrupt sources can be configured as FIQ. This is controlled via the HW_ICOLL_INTERRUPTn[ENFIQ] register bit as shown in Figure 5-2. When enabled to the FIQ, the software interrupt associated with these bits can be used to generate the FIQ from these sources for test purposes. When an interrupt source is programmed as an FIQ, and IRQ cannot be generated from that source.

5.2.3 Interrupt Sources

Table 5-1 lists all of the interrupt sources on the i.MX23. Use hw_irq.h to access these bits.

Table 5-1. i.MX23 Interrupt Sources

INTERRUPT SOURCE	SRC	VECTOR	DESCRIPTION
DEBUG_UART	0	0x0000	Non DMA on the debug UART
COMMS_RX,COMMS_TX	1	0x0004	JTAG debug communications port
SSP2_ERROR	2	0x0008	SSP2 device-level error and status
VDD5V	3	0x000C	IRQ on 5V connect or disconnect. Shared with DCDC status, Linear Regulator status, PSWITCH, and Host 4.2V
HEADPHONE_SHORT	4	0x0010	HEADPHONE_SHORT
DAC_DMA	5	0x0014	DAC DMA channel
DAC_ERROR	6	0x0018	DAC FIFO buffer underflow
ADC_DMA	7	0x001C	ADC DMA channel
ADC_ERROR	8	0x0020	ADC FIFO buffer overflow
SPDIF_DMA,SAIF2_DMA	9	0x0024	SPDIF DMA channel, SAIF2 DMA channel
SPDIF_ERROR, SAIF1_IRQ, SAIF2_IRQ	10	0x0028	SPDIF, SAIF1, SAIF2 FIFO underflow/overflow

Table 5-1. i.MX23 Interrupt Sources (continued)

INTERRUPT SOURCE	SRC	VECTOR	DESCRIPTION
USB_CTRL	11	0x002C	USB controller
USB_WAKEUP	12	0x0030	USB wakeup. Also ARC core to remain suspended.
GPMI_DMA	13	0x0034	From DMA channel for GPMI
SSP1_DMA	14	0x0038	From DMA channel for SSP1
SSP_ERROR	15	0x003C	SSP1 device-level error and status
GPIO0	16	0x0040	GPIO bank 0 interrupt
GPIO1	17	0x0044	GPIO bank 1 interrupt
GPIO2	18	0x0048	GPIO bank 2 interrupt
SAIF1_DMA	19	0x004C	SAIF1 DMA channel
SSP2_DMA	20	0x0050	From DMA channel for SSP2
ECC8_IRQ	21	0x0054	ECC8 completion interrupt
RTC_ALARM	22	0x0058	RTC alarm event
UARTAPP_TX_DMA	23	0x005C	Application UART1 transmitter DMA
UARTAPP_INTERNAL	24	0x0060	Application UART1 internal error
UARTAPP_RX_DMA	25	0x0064	Application UART1 receiver DMA
I2C_DMA	26	0x0068	From DMA channel for I^2C
I2C_ERROR	27	0x006C	From I^2C device detected errors and line conditions
TIMER0	28	0x0070	TIMROT Timer0, recommend to set as FIQ.
TIMER1	29	0x0074	TIMROT Timer1, recommend to set as FIQ.
TIMER2	30	0x0078	TIMROT Timer2, recommend to set as FIQ.
TIMER3	31	0x007C	TIMROT Timer3, recommend to set as FIQ.
BATT_BRNOUT	32	0x0080	Power module battery brownout detect, recommend to set as FIQ.
VDDD_BRNOUT	33	0x0084	Power module VDDD brownout detect, recommend to set as FIQ.
VDDIO_BRNOUT	34	0x0088	Power module VDDIO brownout detect, recommend to set as FIQ.
VDD18_BRNOUT	35	0x008C	Power module VDD18 brownout detect, recommend to set as FIQ.
TOUCH_DETECT	36	0x0090	Touch detection.
LRADC_CH0	37	0x0094	Channel 0 complete.
LRADC_CH1	38	0x0098	Channel 1 complete.
LRADC_CH2	39	0x009C	Channel 2 complete.
LRADC_CH3	40	0x00A0	Channel 3 complete.
LRADC_CH4	41	0x00A4	Channel 4 complete.

Table 5-1. i.MX23 Interrupt Sources (continued)

INTERRUPT SOURCE	SRC	VECTOR	DESCRIPTION
LRADC_CH5	42	0x00A8	Channel 5 complete.
LRADC_CH6	43	0x00AC	Channel 6 complete.
LRADC_CH7	44	0x00B0	Channel 7 complete.
LCDIF_DMA	45	0x00B4	From DMA channel for LCDIF.
LCDIF_ERROR	46	0x00B8	LCDIF error.
DIGCTL_DEBUG_TRAP	47	0x00BC	AHB arbiter debug trap.
RTC_1MSEC	48	0x00C0	RTC 1 ms tick interrupt.
RSVD	49	0x00C4	Reserved
RSVD	50	0x00C8	Reserved
GPMI	51	0x00CC	From GPMI internal error and status IRQ.
RSVD	52	0x00D0	Reserved
DCP_VMI	53	0x00D4	DCP Channel 0 virtual memory page copy.
DCP	54	0x00D8	DCP
RSVD	55	0x00DC	Reserved.
BCH	56	0x00E0	BCH consolidated Interrupt
PXP	57	0x00E4	Pixel Pipeline consolidated Interrupt
UARTAPP2_TX_DMA	58	0x00E8	Application UART2 transmitter DMA
UARTAPP2_INTERNAL	59	0x00EC	Application UART2 internal error
UARTAPP2_RX_DMA	60	0x00F0	Application UART2 receiver DMA
VDAC_DETECT	61	0x00F4	Video dac, jack presence auto-detect
RSVD	62	0x00F8	Reserved.
RSVD	63	0x00FC	Reserved.
VDD5V_DROOP	64	0x0100	5V Droop, recommend to be set as FIQ.
DCDC4P2_BO	65	0x0104	4.2V regulated supply brown-out, recommend to be set as FIQ.
RSVD	66-1278	0x0108-01FC	Reserved.

5.2.4 CPU Wait-for-Interrupt Mode

To enable wait-for-interrupt mode, two distinct actions are required by the programmer:

Set the INTERRUPT_WAIT bit in the HW_CLKCTRL_CPUCLKCTRL register. This must be done via a RMW operation. For example:

```
uclkctrl  = HW_CLKCTRL_CPUCLKCTRL_RD();
uclkctrl |= BM_CLKCTRL_CPUCLKCTRL_INTERRUPT_WAIT;
HW_CLKCTRL_CPUCLKCTRL_WR(uclkctrl);
```

8. After setting the INTERRUPT_WAIT bit, a coprocessor instruction is required.

```
asm (
            // Note: R0 is used in the following example, but any usual
            // <Rd> register may be used.
"mov R0, 0;" // Rd SBZ (should be zero)
"mcr p15,0,r0,c7,c0,4;"//Drain write buffers, idle CPU clock & processor,
                       // and stop processor at this instruction
"nop");      // The lr sent to handler points here after RTI
```

The coprocessor instruction sequence above enables an internal gating signal. This internal signal guarantees that write buffers are drained and ensures that the processor is in an idle state. On execution of the MCR coprocessor instruction, the CPU clock is stopped and the processor halts on the instruction—waiting for an interrupt to occur.

The INTERRUPT_WAIT bit can be thought of as a Wait-for-Interrupt enable bit. Therefore, it must be set prior to execution of the MCR instruction. It is recommended that, when the Wait-for-Interrupt mode is to be used, the INTERRUPT_WAIT bit be set at initialization time and left on.

With the INTERRUPT_WAIT bit set, after execution of the MCR WFI command, the processor halts on the MCR instruction. When an interrupt or FIQ occurs, the MCR instruction completes and the IRQ or FIQ handler is entered normally. The return link that is passed to the handler is automatically adjusted by the above MCR instruction, such that a normal return from interrupt results in continuing execution at the instruction immediately following the MCR. That is, the LR will contain the address of the MCR instruction plus eight, such that a typical return from interrupt instruction (e.g., subs pc, LR, 4) will return to the instruction immediately following the MCR (the NOP in the example above).

Whenever the CPU is stopped because the clock control HW_CLKCTRL_ CPUCLKCTRL_INTERRUPT_WAIT bit is set and the MCR WFI instruction is executed, the CPU stops until an interrupt occurs. The actual condition that wakes up the CPU is determined by ORing together all enabled interrupt requests including those that are directed to the FIQ CPU input. The ICOLL_BUSY output signal from the ICOLL communicates this information to the clock control. This function does not pass through the normal ICOLL state machine. It starts the CPU clock as soon as an enabled interrupt arrives.

5.3 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 39.3.10, “Correct Way to Soft Reset a Block, for additional information on using the SFTRST and CLKGATE bit fields.

5.4 Programmable Registers

The following registers provide interrupt generation and control for the i.MX23.

5.4.1 Interrupt Collector Interrupt Vector Address Register Description

This register is can be read by the Interrupt Service Routine using a load PC instruction. The priority logic presents the vector address of the next IRQ interrupt to be processed by the CPU. The vector address is held until a new ISR is entered..

HW_ICOLL_VECTOR	0x000
HW_ICOLL_VECTOR_SET	0x004
HW_ICOLL_VECTOR_CLR	0x008
HW_ICOLL_VECTOR_TOG	0x00C

Table 5-2. HW_ICOLL_VECTOR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
IRQVECTOR																														RSRVD1	

Table 5-3. HW_ICOLL_VECTOR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:2	**IRQVECTOR**	RW	0x0	This register presents the vector address for the interrupt currently active on the CPU IRQ input. Writing to this register notifies the interrupt collector that the interrupt service routine for the current interrupt has been entered (alternatively when ARM_RSE_MODE is set, reading this register is required).
1:0	**RSRVD1**	RO	0x0	Always write zeroes to this field.

DESCRIPTION:

This register mediates the vectored interrupt collectors interface with the CPU when it enteres the IRQ exception trap. The exception trap should have a LDPC instruction from this address.

EXAMPLE:

```
LDPC HW_ICOLL_VECTOR_ADDR; IRQ exception at 0xffff0018
```

5.4.2 Interrupt Collector Level Acknowledge Register Description

The Interrupt Collector Level Acknowledge Register is used by software to indicate the completion of an interrupt on a specific level.

HW_ICOLL_LEVELACK	0x010

Table 5-4. HW_ICOLL_LEVELACK

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																												IRQLEVELACK			

Table 5-5. HW_ICOLL_LEVELACK Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:4	**RSRVD1**	RO	0x0	Any value can be written to this bitfield. Writes are ignored.
3:0	**IRQLEVELACK**	RW	0x0	This bitfield is written by the processor to acknowledge the completion of an interrupt. The value written must correspond to the priority level of the completed interrupt LEVEL0 = 0x1 level 0 LEVEL1 = 0x2 level 1 LEVEL2 = 0x4 level 2 LEVEL3 = 0x8 level 3

DESCRIPTION:

This register is written to advance the ICOLL internal irq state machine. It advances from an in-service on a level state to the next pending interrupt level or to the idle state. This register is written at the very end of an interrupt service routine. If nesting is used then the CPU irq must be turned on before writing to this register to avoid a race condition in the CPU interrupt hardware. WARNING: the value written to the level ack register is decoded not binary, i.e. 8, 4, 2, 1.

EXAMPLE:

```
HW_ICOLL_LEVELACK_WR(HW_ICOLL_LEVELACK__LEVEL3);
```

5.4.3 Interrupt Collector Control Register Description

The Interrupt Collector Control Register provides overall control of interrupts being routed to the CPU. This register is not at offset zero from the block base because that location is needed for single 32 bit instructions to be placed in the exception vector location.

HW_ICOLL_CTRL	0x020
HW_ICOLL_CTRL_SET	0x024
HW_ICOLL_CTRL_CLR	0x028
HW_ICOLL_CTRL_TOG	0x02C

Table 5-6. HW_ICOLL_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSRVD3						VECTOR_PITCH			BYPASS_FSM	NO_NESTING	ARM_RSE_MODE	FIQ_FINAL_ENABLE	IRQ_FINAL_ENABLE	RSRVD1																	

Table 5-7. HW_ICOLL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	When set to one, this bit causes a soft reset to the entire interrupt collector. This bit must be turned off for normal operation. RUN = 0x0 Allow the interrupt collector to operate normally. IN_RESET = 0x1 Hold the interrupt collector in its reset state.
30	**CLKGATE**	RW	0x1	When set to one, this bit causes all clocks within the interrupt collector to be gated off. WARNING: Do not set this bit at the same time as SFTRST. Doing so, causes the softreset to have no effect. Setting SFTRST will cause the CLKGATE bit to set automatically four clocks later. RUN = 0x0 Enable clocks for normal operation of interrupt collector. NO_CLOCKS = 0x1 disable clocking within the interrupt collector.
29:24	**RSRVD3**	RO	0x0	Always write zeroes to this bitfield.
23:21	**VECTOR_PITCH**	RW	0x0	When an interrupt occurs one of the 128 input requests becomes the winning bit number, i.e. 0 to 127. This bit field selects one of eight constant multiplier values to multiply the winning bit number. The multiplied bit number is added to the vector table base to become the vector address. 0x0 and 0x1 yield a multiplier of 4 bytes. 0x2 yields a multiplier of 8 bytes while 0x3 yields a multiplier of 12 bytes, i.e. (8 + 4) bytes per step. DEFAULT_BY4 = 0x0 one word pitch BY4 = 0x1 one word pitch BY8 = 0x2 two word pitch BY12 = 0x3 three word pitch BY16 = 0x4 four word pitch BY20 = 0x5 five word pitch BY24 = 0x6 six word pitch BY28 = 0x7 seven word pitch
20	**BYPASS_FSM**	RW	0x0	Set this bit to one to bypass the FSM control of the request holding register and the vector address. With this bit set to one, the vector address register is continuously updated as interrupt requests come in. Turn off all enable bits and walk a one through the software interrupts, observing the vector address changes. Set to zero for normal operation. This control is included as a test mode, and is not intended for use by a real application. NORMAL = 0x0 Normal BYPASS = 0x1 no FSM handshake with CPU

Table 5-7. HW_ICOLL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19	**NO_NESTING**	RW	0x0	Set this bit to one disable interrupt level nesting, i.e. higher priority interrupt interrupting lower priority. For normal operation, set this bit to zero. NORMAL = 0x0 Normal NO_NEST = 0x1 no support for interrupt nesting
18	**ARM_RSE_MODE**	RW	0x0	Set this bit to one enable the ARM-style read side effect associated with the vector address register. In this mode, interrupt inservice is signalled by the read of the HW_ICOLL_VECTOR register to acquire the interrupt vector address. Set this bit to zero for normal operation, in which the ISR signals inservice explicitly by means of a write to the HW_ICOLL_VECTOR register.
17	**FIQ_FINAL_ENABLE**	RW	0x1	Set this bit to one to enable the final FIQ output to the CPU. Set this bit to zero for testing the interrupt collector without causing actual CPU interrupts. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
16	**IRQ_FINAL_ENABLE**	RW	0x1	Set this bit to one to enable the final IRQ output to the CPU. Set this bit to zero for testing the interrupt collector without causing actual CPU interrupts. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
15:0	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.

DESCRIPTION:

This register handles the overall control of the interrupt collector, including soft reset and clock gate. In addition, it handles state machine variations like NO_NESTING and ARM read side effect processing on the vector address register.

EXAMPLE:

```
HW_ICOLL_CTRL_CLR(BM_ICOLL_CTRL_SFTRST | BM_ICOLL_CTRL_SFTRST );
```

5.4.4 Interrupt Collector Interrupt Vector Base Address Register Description

This register is used by the priority logic to generate a unique vector address for each of the 80 interrupt request lines coming into the interrupt collector. The vector address is formed by multiply the interrupt bit number by 4 and adding it to the vector base address.

HW_ICOLL_VBASE	0x040
HW_ICOLL_VBASE_SET	0x044
HW_ICOLL_VBASE_CLR	0x048
HW_ICOLL_VBASE_TOG	0x04C

Table 5-8. HW_ICOLL_VBASE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TABLE_ADDRESS																														RSRVD1	

Table 5-9. HW_ICOLL_VBASE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:2	**TABLE_ADDRESS**	RW	0x0	This bitfield holds the upper 30 bits of the base address of the vector table.
1:0	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.

DESCRIPTION:

This register provides a mechanism to specify the base address of the interrupt vector table. It is used in the compuation of the value supplied in HW_ICOLL_VECTOR register.

EXAMPLE:

```
HW_ICOLL_VBASE_WR(pInterruptVectorTable);
```

5.4.5 Interrupt Collector Status Register Description

Read only view into various internal states, including the Vector number of the current interupt.

HW_ICOLL_STAT 0x070

Table 5-10. HW_ICOLL_STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																									VECTOR_NUMBER						

Table 5-11. HW_ICOLL_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:7	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
6:0	**VECTOR_NUMBER**	RO	0x7F	Vector number of current interrupt. Multiply by 4 * HW_ICOLL_CTRL[VECTOR_PITCH] and add to vector base address to obtain the value in HW_ICOLL_VECTOR.

DESCRIPTION:

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE:

```
if(HW_ICOLL_STAT_VECTOR_NUMBER_READ() == 0x00000017) ISR_vector_23(); // ISR for vector 23 decimal,
17 hex
```

5.4.6 Interrupt Collector Raw Interrupt Input Register 0 Description

The lower 32 interrupt hardware-source states are visible in this read-only register.

HW_ICOLL_RAW0	0x0A0
HW_ICOLL_RAW0_SET	0x0A4
HW_ICOLL_RAW0_CLR	0x0A8
HW_ICOLL_RAW0_TOG	0x0AC

Table 5-12. HW_ICOLL_RAW0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RAW_IRQS																															

Table 5-13. HW_ICOLL_RAW0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RAW_IRQS**	RO	0x0	read-only view of the lower 32 hardware interrupt request bits.

DESCRIPTION:

This register provides a read-only view of the raw interrupt request lines coming from various parts of the chip. Its purpose is to improve diagnostic observability. Note that these only capture the state of hardware interrupt sources.

EXAMPLE:

```
ulTest = HW_ICOLL_RAW0.RAW_IRQS;
```

5.4.7 Interrupt Collector Raw Interrupt Input Register 1 Description

Interrupt hardware-source states 32-63 are visible in this read-only register.

HW_ICOLL_RAW1	0x0B0
HW_ICOLL_RAW1_SET	0x0B4
HW_ICOLL_RAW1_CLR	0x0B8

HW_ICOLL_RAW1_TOG 0x0BC

Table 5-14. HW_ICOLL_RAW1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RAW_IRQS																															

Table 5-15. HW_ICOLL_RAW1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RAW_IRQS**	RO	0x0	read-only view of hardware interrupt request bits 32-63.

DESCRIPTION:

This register provides a read-only view of the raw interrupt request lines coming from various parts of the chip. The purpose is to improve diagnostic observability. Note that these only capture the state of hardware interrupt sources.

EXAMPLE:

```
ulTest = HW_ICOLL_RAW0.RAW_IRQS;
```

5.4.8 Interrupt Collector Raw Interrupt Input Register 2 Description

Interrupt hardware-source states 64-95 are visible in this read-only register.

HW_ICOLL_RAW2 0x0C0
HW_ICOLL_RAW2_SET 0x0C4
HW_ICOLL_RAW2_CLR 0x0C8
HW_ICOLL_RAW2_TOG 0x0CC

Table 5-16. HW_ICOLL_RAW2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RAW_IRQS																															

Table 5-17. HW_ICOLL_RAW2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RAW_IRQS**	RO	0x0	read-only view of hardware interrupt request bits 64-95.

DESCRIPTION:

This register provides a read-only view of the raw interrupt request lines coming from various parts of the chip. The purpose is to improve diagnostic observability. Note that these only capture the state of hardware interrupt sources.

EXAMPLE:

```
ulTest = HW_ICOLL_RAW0.RAW_IRQS;
```

5.4.9 Interrupt Collector Raw Interrupt Input Register 3 Description

Interrupt hardware-source states 96-127 are visible in this read-only register.

HW_ICOLL_RAW3	0x0D0
HW_ICOLL_RAW3_SET	0x0D4
HW_ICOLL_RAW3_CLR	0x0D8
HW_ICOLL_RAW3_TOG	0x0DC

Table 5-18. HW_ICOLL_RAW3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RAW_IRQS																															

Table 5-19. HW_ICOLL_RAW3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RAW_IRQS**	RO	0x0	read-only view of hardware interrupt request bits 96-127.

DESCRIPTION:

This register provides a read-only view of the raw interrupt request lines coming from various parts of the chip. The purpose is to improve diagnostic observability. Note that these only capture the state of hardware interrupt sources.

EXAMPLE:

```
ulTest = HW_ICOLL_RAW0.RAW_IRQS;
```

5.4.10 Interrupt Collector Interrupt Register 0 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT0	0x120
HW_ICOLL_INTERRUPT0_SET	0x124
HW_ICOLL_INTERRUPT0_CLR	0x128
HW_ICOLL_INTERRUPT0_TOG	0x12C

Table 5-20. HW_ICOLL_INTERRUPT0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY		

Table 5-21. HW_ICOLL_INTERRUPT0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT0_SET(0,0x00000001);
```

5.4.11 Interrupt Collector Interrupt Register 1 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT1	0x130
HW_ICOLL_INTERRUPT1_SET	0x134
HW_ICOLL_INTERRUPT1_CLR	0x138
HW_ICOLL_INTERRUPT1_TOG	0x13C

Table 5-22. HW_ICOLL_INTERRUPT1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-23. HW_ICOLL_INTERRUPT1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT1_SET(0,0x00000001);
```

5.4.12 Interrupt Collector Interrupt Register 2 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT2	0x140
HW_ICOLL_INTERRUPT2_SET	0x144
HW_ICOLL_INTERRUPT2_CLR	0x148
HW_ICOLL_INTERRUPT2_TOG	0x14C

Table 5-24. HW_ICOLL_INTERRUPT2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-25. HW_ICOLL_INTERRUPT2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT2_SET(0,0x00000001);
```

5.4.13 Interrupt Collector Interrupt Register 3 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT3	0x150
HW_ICOLL_INTERRUPT3_SET	0x154
HW_ICOLL_INTERRUPT3_CLR	0x158
HW_ICOLL_INTERRUPT3_TOG	0x15C

Table 5-26. HW_ICOLL_INTERRUPT3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-27. HW_ICOLL_INTERRUPT3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT3_SET(0,0x00000001);
```

5.4.14 Interrupt Collector Interrupt Register 4 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT4	0x160
HW_ICOLL_INTERRUPT4_SET	0x164
HW_ICOLL_INTERRUPT4_CLR	0x168
HW_ICOLL_INTERRUPT4_TOG	0x16C

Table 5-28. HW_ICOLL_INTERRUPT4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-29. HW_ICOLL_INTERRUPT4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT4_SET(0,0x00000001);
```

5.4.15 Interrupt Collector Interrupt Register 5 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT5	0x170
HW_ICOLL_INTERRUPT5_SET	0x174
HW_ICOLL_INTERRUPT5_CLR	0x178
HW_ICOLL_INTERRUPT5_TOG	0x17C

Table 5-30. HW_ICOLL_INTERRUPT5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-31. HW_ICOLL_INTERRUPT5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT5_SET(0,0x00000001);
```

5.4.16 Interrupt Collector Interrupt Register 6 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT6	0x180
HW_ICOLL_INTERRUPT6_SET	0x184
HW_ICOLL_INTERRUPT6_CLR	0x188
HW_ICOLL_INTERRUPT6_TOG	0x18C

Table 5-32. HW_ICOLL_INTERRUPT6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-33. HW_ICOLL_INTERRUPT6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT6_SET(0,0x00000001);
```

5.4.17 Interrupt Collector Interrupt Register 7 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT7	0x190
HW_ICOLL_INTERRUPT7_SET	0x194
HW_ICOLL_INTERRUPT7_CLR	0x198
HW_ICOLL_INTERRUPT7_TOG	0x19C

Table 5-34. HW_ICOLL_INTERRUPT7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-35. HW_ICOLL_INTERRUPT7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT7_SET(0,0x00000001);
```

5.4.18 Interrupt Collector Interrupt Register 8 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT8	0x1A0
HW_ICOLL_INTERRUPT8_SET	0x1A4
HW_ICOLL_INTERRUPT8_CLR	0x1A8
HW_ICOLL_INTERRUPT8_TOG	0x1AC

Table 5-36. HW_ICOLL_INTERRUPT8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-37. HW_ICOLL_INTERRUPT8 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT8_SET(0,0x00000001);
```

5.4.19 Interrupt Collector Interrupt Register 9 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT9	0x1B0
HW_ICOLL_INTERRUPT9_SET	0x1B4
HW_ICOLL_INTERRUPT9_CLR	0x1B8
HW_ICOLL_INTERRUPT9_TOG	0x1BC

Table 5-38. HW_ICOLL_INTERRUPT9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-39. HW_ICOLL_INTERRUPT9 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT9_SET(0,0x00000001);
```

5.4.20 Interrupt Collector Interrupt Register 10 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT10	0x1C0
HW_ICOLL_INTERRUPT10_SET	0x1C4
HW_ICOLL_INTERRUPT10_CLR	0x1C8
HW_ICOLL_INTERRUPT10_TOG	0x1CC

Table 5-40. HW_ICOLL_INTERRUPT10

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-41. HW_ICOLL_INTERRUPT10 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT10_SET(0,0x00000001);
```

5.4.21 Interrupt Collector Interrupt Register 11 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT11	0x1D0
HW_ICOLL_INTERRUPT11_SET	0x1D4
HW_ICOLL_INTERRUPT11_CLR	0x1D8
HW_ICOLL_INTERRUPT11_TOG	0x1DC

Table 5-42. HW_ICOLL_INTERRUPT11

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-43. HW_ICOLL_INTERRUPT11 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT11_SET(0,0x00000001);
```

5.4.22 Interrupt Collector Interrupt Register 12 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT12	0x1E0
HW_ICOLL_INTERRUPT12_SET	0x1E4
HW_ICOLL_INTERRUPT12_CLR	0x1E8
HW_ICOLL_INTERRUPT12_TOG	0x1EC

Table 5-44. HW_ICOLL_INTERRUPT12

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-45. HW_ICOLL_INTERRUPT12 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT12_SET(0,0x00000001);
```

5.4.23 Interrupt Collector Interrupt Register 13 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT13	0x1F0
HW_ICOLL_INTERRUPT13_SET	0x1F4
HW_ICOLL_INTERRUPT13_CLR	0x1F8
HW_ICOLL_INTERRUPT13_TOG	0x1FC

Table 5-46. HW_ICOLL_INTERRUPT13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-47. HW_ICOLL_INTERRUPT13 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT13_SET(0,0x00000001);
```

5.4.24 Interrupt Collector Interrupt Register 14 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT14	0x200
HW_ICOLL_INTERRUPT14_SET	0x204
HW_ICOLL_INTERRUPT14_CLR	0x208
HW_ICOLL_INTERRUPT14_TOG	0x20C

Table 5-48. HW_ICOLL_INTERRUPT14

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-49. HW_ICOLL_INTERRUPT14 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT14_SET(0,0x00000001);
```

5.4.25 Interrupt Collector Interrupt Register 15 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT15	0x210
HW_ICOLL_INTERRUPT15_SET	0x214
HW_ICOLL_INTERRUPT15_CLR	0x218
HW_ICOLL_INTERRUPT15_TOG	0x21C

Table 5-50. HW_ICOLL_INTERRUPT15

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-51. HW_ICOLL_INTERRUPT15 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT15_SET(0,0x00000001);
```

5.4.26 Interrupt Collector Interrupt Register 16 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT16	0x220
HW_ICOLL_INTERRUPT16_SET	0x224
HW_ICOLL_INTERRUPT16_CLR	0x228
HW_ICOLL_INTERRUPT16_TOG	0x22C

Table 5-52. HW_ICOLL_INTERRUPT16

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-53. HW_ICOLL_INTERRUPT16 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT16_SET(0,0x00000001);
```

5.4.27 Interrupt Collector Interrupt Register 17 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT17	0x230
HW_ICOLL_INTERRUPT17_SET	0x234
HW_ICOLL_INTERRUPT17_CLR	0x238
HW_ICOLL_INTERRUPT17_TOG	0x23C

Table 5-54. HW_ICOLL_INTERRUPT17

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-55. HW_ICOLL_INTERRUPT17 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT17_SET(0,0x00000001);
```

5.4.28 Interrupt Collector Interrupt Register 18 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT18	0x240
HW_ICOLL_INTERRUPT18_SET	0x244
HW_ICOLL_INTERRUPT18_CLR	0x248
HW_ICOLL_INTERRUPT18_TOG	0x24C

Table 5-56. HW_ICOLL_INTERRUPT18

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-57. HW_ICOLL_INTERRUPT18 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT18_SET(0,0x00000001);
```

5.4.29 Interrupt Collector Interrupt Register 19 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT19	0x250
HW_ICOLL_INTERRUPT19_SET	0x254
HW_ICOLL_INTERRUPT19_CLR	0x258
HW_ICOLL_INTERRUPT19_TOG	0x25C

Table 5-58. HW_ICOLL_INTERRUPT19

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-59. HW_ICOLL_INTERRUPT19 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT19_SET(0,0x00000001);
```

5.4.30 Interrupt Collector Interrupt Register 20 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT20	0x260
HW_ICOLL_INTERRUPT20_SET	0x264
HW_ICOLL_INTERRUPT20_CLR	0x268
HW_ICOLL_INTERRUPT20_TOG	0x26C

Table 5-60. HW_ICOLL_INTERRUPT20

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-61. HW_ICOLL_INTERRUPT20 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT20_SET(0,0x00000001);
```

5.4.31 Interrupt Collector Interrupt Register 21 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT21	0x270
HW_ICOLL_INTERRUPT21_SET	0x274
HW_ICOLL_INTERRUPT21_CLR	0x278
HW_ICOLL_INTERRUPT21_TOG	0x27C

Table 5-62. HW_ICOLL_INTERRUPT21

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-63. HW_ICOLL_INTERRUPT21 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT21_SET(0,0x00000001);
```

5.4.32 Interrupt Collector Interrupt Register 22 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT22	0x280
HW_ICOLL_INTERRUPT22_SET	0x284
HW_ICOLL_INTERRUPT22_CLR	0x288
HW_ICOLL_INTERRUPT22_TOG	0x28C

Table 5-64. HW_ICOLL_INTERRUPT22

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-65. HW_ICOLL_INTERRUPT22 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT22_SET(0,0x00000001);
```

5.4.33 Interrupt Collector Interrupt Register 23 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT23	0x290
HW_ICOLL_INTERRUPT23_SET	0x294
HW_ICOLL_INTERRUPT23_CLR	0x298
HW_ICOLL_INTERRUPT23_TOG	0x29C

Table 5-66. HW_ICOLL_INTERRUPT23

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-67. HW_ICOLL_INTERRUPT23 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT23_SET(0,0x00000001);
```

5.4.34 Interrupt Collector Interrupt Register 24 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT24	0x2A0
HW_ICOLL_INTERRUPT24_SET	0x2A4
HW_ICOLL_INTERRUPT24_CLR	0x2A8
HW_ICOLL_INTERRUPT24_TOG	0x2AC

Table 5-68. HW_ICOLL_INTERRUPT24

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-69. HW_ICOLL_INTERRUPT24 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT24_SET(0,0x00000001);
```

5.4.35 Interrupt Collector Interrupt Register 25 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT25	0x2B0
HW_ICOLL_INTERRUPT25_SET	0x2B4
HW_ICOLL_INTERRUPT25_CLR	0x2B8
HW_ICOLL_INTERRUPT25_TOG	0x2BC

Table 5-70. HW_ICOLL_INTERRUPT25

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-71. HW_ICOLL_INTERRUPT25 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT25_SET(0,0x00000001);
```

5.4.36 Interrupt Collector Interrupt Register 26 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT26	0x2C0
HW_ICOLL_INTERRUPT26_SET	0x2C4
HW_ICOLL_INTERRUPT26_CLR	0x2C8
HW_ICOLL_INTERRUPT26_TOG	0x2CC

Table 5-72. HW_ICOLL_INTERRUPT26

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-73. HW_ICOLL_INTERRUPT26 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT26_SET(0,0x00000001);
```

5.4.37 Interrupt Collector Interrupt Register 27 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT27	0x2D0
HW_ICOLL_INTERRUPT27_SET	0x2D4
HW_ICOLL_INTERRUPT27_CLR	0x2D8
HW_ICOLL_INTERRUPT27_TOG	0x2DC

Table 5-74. HW_ICOLL_INTERRUPT27

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-75. HW_ICOLL_INTERRUPT27 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT27_SET(0,0x00000001);
```

5.4.38 Interrupt Collector Interrupt Register 28 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT28	0x2E0
HW_ICOLL_INTERRUPT28_SET	0x2E4
HW_ICOLL_INTERRUPT28_CLR	0x2E8
HW_ICOLL_INTERRUPT28_TOG	0x2EC

Table 5-76. HW_ICOLL_INTERRUPT28

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-77. HW_ICOLL_INTERRUPT28 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT28_SET(0,0x00000001);
```

5.4.39 Interrupt Collector Interrupt Register 29 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT29	0x2F0
HW_ICOLL_INTERRUPT29_SET	0x2F4
HW_ICOLL_INTERRUPT29_CLR	0x2F8
HW_ICOLL_INTERRUPT29_TOG	0x2FC

Table 5-78. HW_ICOLL_INTERRUPT29

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-79. HW_ICOLL_INTERRUPT29 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT29_SET(0,0x00000001);
```

5.4.40 Interrupt Collector Interrupt Register 30 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT30	0x300
HW_ICOLL_INTERRUPT30_SET	0x304
HW_ICOLL_INTERRUPT30_CLR	0x308
HW_ICOLL_INTERRUPT30_TOG	0x30C

Table 5-80. HW_ICOLL_INTERRUPT30

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-81. HW_ICOLL_INTERRUPT30 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT30_SET(0,0x00000001);
```

5.4.41 Interrupt Collector Interrupt Register 31 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT31	0x310
HW_ICOLL_INTERRUPT31_SET	0x314
HW_ICOLL_INTERRUPT31_CLR	0x318
HW_ICOLL_INTERRUPT31_TOG	0x31C

Table 5-82. HW_ICOLL_INTERRUPT31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-83. HW_ICOLL_INTERRUPT31 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT31_SET(0,0x00000001);
```

5.4.42 Interrupt Collector Interrupt Register 32 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT32	0x320
HW_ICOLL_INTERRUPT32_SET	0x324
HW_ICOLL_INTERRUPT32_CLR	0x328
HW_ICOLL_INTERRUPT32_TOG	0x32C

Table 5-84. HW_ICOLL_INTERRUPT32

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-85. HW_ICOLL_INTERRUPT32 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT32_SET(0,0x00000001);
```

5.4.43 Interrupt Collector Interrupt Register 33 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT33	0x330
HW_ICOLL_INTERRUPT33_SET	0x334
HW_ICOLL_INTERRUPT33_CLR	0x338
HW_ICOLL_INTERRUPT33_TOG	0x33C

Table 5-86. HW_ICOLL_INTERRUPT33

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-87. HW_ICOLL_INTERRUPT33 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT33_SET(0,0x00000001);
```

5.4.44 Interrupt Collector Interrupt Register 34 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT34	0x340
HW_ICOLL_INTERRUPT34_SET	0x344
HW_ICOLL_INTERRUPT34_CLR	0x348
HW_ICOLL_INTERRUPT34_TOG	0x34C

Table 5-88. HW_ICOLL_INTERRUPT34

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-89. HW_ICOLL_INTERRUPT34 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT34_SET(0,0x00000001);
```

5.4.45 Interrupt Collector Interrupt Register 35 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT35	0x350
HW_ICOLL_INTERRUPT35_SET	0x354
HW_ICOLL_INTERRUPT35_CLR	0x358
HW_ICOLL_INTERRUPT35_TOG	0x35C

Table 5-90. HW_ICOLL_INTERRUPT35

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-91. HW_ICOLL_INTERRUPT35 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT35_SET(0,0x00000001);
```

5.4.46 Interrupt Collector Interrupt Register 36 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT36	0x360
HW_ICOLL_INTERRUPT36_SET	0x364
HW_ICOLL_INTERRUPT36_CLR	0x368
HW_ICOLL_INTERRUPT36_TOG	0x36C

Table 5-92. HW_ICOLL_INTERRUPT36

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-93. HW_ICOLL_INTERRUPT36 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT36_SET(0,0x00000001);
```

5.4.47 Interrupt Collector Interrupt Register 37 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT37	0x370
HW_ICOLL_INTERRUPT37_SET	0x374
HW_ICOLL_INTERRUPT37_CLR	0x378
HW_ICOLL_INTERRUPT37_TOG	0x37C

Table 5-94. HW_ICOLL_INTERRUPT37

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-95. HW_ICOLL_INTERRUPT37 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT37_SET(0,0x00000001);
```

5.4.48 Interrupt Collector Interrupt Register 38 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT38	0x380
HW_ICOLL_INTERRUPT38_SET	0x384
HW_ICOLL_INTERRUPT38_CLR	0x388
HW_ICOLL_INTERRUPT38_TOG	0x38C

Table 5-96. HW_ICOLL_INTERRUPT38

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-97. HW_ICOLL_INTERRUPT38 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT38_SET(0,0x00000001);
```

5.4.49 Interrupt Collector Interrupt Register 39 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT39	0x390
HW_ICOLL_INTERRUPT39_SET	0x394
HW_ICOLL_INTERRUPT39_CLR	0x398
HW_ICOLL_INTERRUPT39_TOG	0x39C

Table 5-98. HW_ICOLL_INTERRUPT39

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-99. HW_ICOLL_INTERRUPT39 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT39_SET(0,0x00000001);
```

5.4.50 Interrupt Collector Interrupt Register 40 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT40	0x3A0
HW_ICOLL_INTERRUPT40_SET	0x3A4
HW_ICOLL_INTERRUPT40_CLR	0x3A8
HW_ICOLL_INTERRUPT40_TOG	0x3AC

Table 5-100. HW_ICOLL_INTERRUPT40

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-101. HW_ICOLL_INTERRUPT40 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT40_SET(0,0x00000001);
```

5.4.51 Interrupt Collector Interrupt Register 41 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT41	0x3B0
HW_ICOLL_INTERRUPT41_SET	0x3B4
HW_ICOLL_INTERRUPT41_CLR	0x3B8
HW_ICOLL_INTERRUPT41_TOG	0x3BC

Table 5-102. HW_ICOLL_INTERRUPT41

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-103. HW_ICOLL_INTERRUPT41 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT41_SET(0,0x00000001);
```

5.4.52 Interrupt Collector Interrupt Register 42 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT42	0x3C0
HW_ICOLL_INTERRUPT42_SET	0x3C4
HW_ICOLL_INTERRUPT42_CLR	0x3C8
HW_ICOLL_INTERRUPT42_TOG	0x3CC

Table 5-104. HW_ICOLL_INTERRUPT42

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-105. HW_ICOLL_INTERRUPT42 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT42_SET(0,0x00000001);
```

5.4.53 Interrupt Collector Interrupt Register 43 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT43	0x3D0
HW_ICOLL_INTERRUPT43_SET	0x3D4
HW_ICOLL_INTERRUPT43_CLR	0x3D8
HW_ICOLL_INTERRUPT43_TOG	0x3DC

Table 5-106. HW_ICOLL_INTERRUPT43

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-107. HW_ICOLL_INTERRUPT43 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT43_SET(0,0x00000001);
```

5.4.54 Interrupt Collector Interrupt Register 44 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT44	0x3E0
HW_ICOLL_INTERRUPT44_SET	0x3E4
HW_ICOLL_INTERRUPT44_CLR	0x3E8
HW_ICOLL_INTERRUPT44_TOG	0x3EC

Table 5-108. HW_ICOLL_INTERRUPT44

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-109. HW_ICOLL_INTERRUPT44 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT44_SET(0,0x00000001);
```

5.4.55 Interrupt Collector Interrupt Register 45 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT45	0x3F0
HW_ICOLL_INTERRUPT45_SET	0x3F4
HW_ICOLL_INTERRUPT45_CLR	0x3F8
HW_ICOLL_INTERRUPT45_TOG	0x3FC

Table 5-110. HW_ICOLL_INTERRUPT45

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-111. HW_ICOLL_INTERRUPT45 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT45_SET(0,0x00000001);
```

5.4.56 Interrupt Collector Interrupt Register 46 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT46	0x400
HW_ICOLL_INTERRUPT46_SET	0x404
HW_ICOLL_INTERRUPT46_CLR	0x408
HW_ICOLL_INTERRUPT46_TOG	0x40C

Table 5-112. HW_ICOLL_INTERRUPT46

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-113. HW_ICOLL_INTERRUPT46 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT46_SET(0,0x00000001);
```

5.4.57 Interrupt Collector Interrupt Register 47 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT47	0x410
HW_ICOLL_INTERRUPT47_SET	0x414
HW_ICOLL_INTERRUPT47_CLR	0x418
HW_ICOLL_INTERRUPT47_TOG	0x41C

Table 5-114. HW_ICOLL_INTERRUPT47

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-115. HW_ICOLL_INTERRUPT47 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT47_SET(0,0x00000001);
```

5.4.58 Interrupt Collector Interrupt Register 48 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT48	0x420
HW_ICOLL_INTERRUPT48_SET	0x424
HW_ICOLL_INTERRUPT48_CLR	0x428
HW_ICOLL_INTERRUPT48_TOG	0x42C

Table 5-116. HW_ICOLL_INTERRUPT48

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-117. HW_ICOLL_INTERRUPT48 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT48_SET(0,0x00000001);
```

5.4.59 Interrupt Collector Interrupt Register 49 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT49	0x430
HW_ICOLL_INTERRUPT49_SET	0x434
HW_ICOLL_INTERRUPT49_CLR	0x438
HW_ICOLL_INTERRUPT49_TOG	0x43C

Table 5-118. HW_ICOLL_INTERRUPT49

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-119. HW_ICOLL_INTERRUPT49 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT49_SET(0,0x00000001);
```

5.4.60 Interrupt Collector Interrupt Register 50 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT50	0x440
HW_ICOLL_INTERRUPT50_SET	0x444
HW_ICOLL_INTERRUPT50_CLR	0x448
HW_ICOLL_INTERRUPT50_TOG	0x44C

Table 5-120. HW_ICOLL_INTERRUPT50

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-121. HW_ICOLL_INTERRUPT50 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT50_SET(0,0x00000001);
```

5.4.61 Interrupt Collector Interrupt Register 51 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT51	0x450
HW_ICOLL_INTERRUPT51_SET	0x454
HW_ICOLL_INTERRUPT51_CLR	0x458
HW_ICOLL_INTERRUPT51_TOG	0x45C

Table 5-122. HW_ICOLL_INTERRUPT51

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-123. HW_ICOLL_INTERRUPT51 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT51_SET(0,0x00000001);
```

5.4.62 Interrupt Collector Interrupt Register 52 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT52	0x460
HW_ICOLL_INTERRUPT52_SET	0x464
HW_ICOLL_INTERRUPT52_CLR	0x468
HW_ICOLL_INTERRUPT52_TOG	0x46C

Table 5-124. HW_ICOLL_INTERRUPT52

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-125. HW_ICOLL_INTERRUPT52 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT52_SET(0,0x00000001);
```

5.4.63 Interrupt Collector Interrupt Register 53 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT53	0x470
HW_ICOLL_INTERRUPT53_SET	0x474
HW_ICOLL_INTERRUPT53_CLR	0x478
HW_ICOLL_INTERRUPT53_TOG	0x47C

Table 5-126. HW_ICOLL_INTERRUPT53

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-127. HW_ICOLL_INTERRUPT53 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT53_SET(0,0x00000001);
```

5.4.64 Interrupt Collector Interrupt Register 54 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT54	0x480
HW_ICOLL_INTERRUPT54_SET	0x484
HW_ICOLL_INTERRUPT54_CLR	0x488
HW_ICOLL_INTERRUPT54_TOG	0x48C

Table 5-128. HW_ICOLL_INTERRUPT54

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-129. HW_ICOLL_INTERRUPT54 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT54_SET(0,0x00000001);
```

5.4.65 Interrupt Collector Interrupt Register 55 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT55	0x490
HW_ICOLL_INTERRUPT55_SET	0x494
HW_ICOLL_INTERRUPT55_CLR	0x498
HW_ICOLL_INTERRUPT55_TOG	0x49C

Table 5-130. HW_ICOLL_INTERRUPT55

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-131. HW_ICOLL_INTERRUPT55 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT55_SET(0,0x00000001);
```

5.4.66 Interrupt Collector Interrupt Register 56 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT56	0x4A0
HW_ICOLL_INTERRUPT56_SET	0x4A4
HW_ICOLL_INTERRUPT56_CLR	0x4A8
HW_ICOLL_INTERRUPT56_TOG	0x4AC

Table 5-132. HW_ICOLL_INTERRUPT56

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-133. HW_ICOLL_INTERRUPT56 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT56_SET(0,0x00000001);
```

5.4.67 Interrupt Collector Interrupt Register 57 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT57	0x4B0
HW_ICOLL_INTERRUPT57_SET	0x4B4
HW_ICOLL_INTERRUPT57_CLR	0x4B8
HW_ICOLL_INTERRUPT57_TOG	0x4BC

Table 5-134. HW_ICOLL_INTERRUPT57

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-135. HW_ICOLL_INTERRUPT57 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT57_SET(0,0x00000001);
```

5.4.68 Interrupt Collector Interrupt Register 58 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT58	0x4C0
HW_ICOLL_INTERRUPT58_SET	0x4C4
HW_ICOLL_INTERRUPT58_CLR	0x4C8
HW_ICOLL_INTERRUPT58_TOG	0x4CC

Table 5-136. HW_ICOLL_INTERRUPT58

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-137. HW_ICOLL_INTERRUPT58 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT58_SET(0,0x00000001);
```

5.4.69 Interrupt Collector Interrupt Register 59 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT59	0x4D0
HW_ICOLL_INTERRUPT59_SET	0x4D4
HW_ICOLL_INTERRUPT59_CLR	0x4D8
HW_ICOLL_INTERRUPT59_TOG	0x4DC

Table 5-138. HW_ICOLL_INTERRUPT59

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-139. HW_ICOLL_INTERRUPT59 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT59_SET(0,0x00000001);
```

5.4.70 Interrupt Collector Interrupt Register 60 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT60	0x4E0
HW_ICOLL_INTERRUPT60_SET	0x4E4
HW_ICOLL_INTERRUPT60_CLR	0x4E8
HW_ICOLL_INTERRUPT60_TOG	0x4EC

Table 5-140. HW_ICOLL_INTERRUPT60

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-141. HW_ICOLL_INTERRUPT60 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT60_SET(0,0x00000001);
```

5.4.71 Interrupt Collector Interrupt Register 61 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT61	0x4F0
HW_ICOLL_INTERRUPT61_SET	0x4F4
HW_ICOLL_INTERRUPT61_CLR	0x4F8
HW_ICOLL_INTERRUPT61_TOG	0x4FC

Table 5-142. HW_ICOLL_INTERRUPT61

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-143. HW_ICOLL_INTERRUPT61 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT61_SET(0,0x00000001);
```

5.4.72 Interrupt Collector Interrupt Register 62 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT62	0x500
HW_ICOLL_INTERRUPT62_SET	0x504
HW_ICOLL_INTERRUPT62_CLR	0x508
HW_ICOLL_INTERRUPT62_TOG	0x50C

Table 5-144. HW_ICOLL_INTERRUPT62

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-145. HW_ICOLL_INTERRUPT62 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT62_SET(0,0x00000001);
```

5.4.73 Interrupt Collector Interrupt Register 63 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT63	0x510
HW_ICOLL_INTERRUPT63_SET	0x514
HW_ICOLL_INTERRUPT63_CLR	0x518
HW_ICOLL_INTERRUPT63_TOG	0x51C

Table 5-146. HW_ICOLL_INTERRUPT63

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-147. HW_ICOLL_INTERRUPT63 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT63_SET(0,0x00000001);
```

5.4.74 Interrupt Collector Interrupt Register 64 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT64	0x520
HW_ICOLL_INTERRUPT64_SET	0x524
HW_ICOLL_INTERRUPT64_CLR	0x528
HW_ICOLL_INTERRUPT64_TOG	0x52C

Table 5-148. HW_ICOLL_INTERRUPT64

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-149. HW_ICOLL_INTERRUPT64 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT64_SET(0,0x00000001);
```

5.4.75 Interrupt Collector Interrupt Register 65 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT65	0x530
HW_ICOLL_INTERRUPT65_SET	0x534
HW_ICOLL_INTERRUPT65_CLR	0x538
HW_ICOLL_INTERRUPT65_TOG	0x53C

Table 5-150. HW_ICOLL_INTERRUPT65

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-151. HW_ICOLL_INTERRUPT65 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT65_SET(0,0x00000001);
```

5.4.76 Interrupt Collector Interrupt Register 66 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT66	0x540
HW_ICOLL_INTERRUPT66_SET	0x544
HW_ICOLL_INTERRUPT66_CLR	0x548
HW_ICOLL_INTERRUPT66_TOG	0x54C

Table 5-152. HW_ICOLL_INTERRUPT66

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-153. HW_ICOLL_INTERRUPT66 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT66_SET(0,0x00000001);
```

5.4.77 Interrupt Collector Interrupt Register 67 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT67	0x550
HW_ICOLL_INTERRUPT67_SET	0x554
HW_ICOLL_INTERRUPT67_CLR	0x558
HW_ICOLL_INTERRUPT67_TOG	0x55C

Table 5-154. HW_ICOLL_INTERRUPT67

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-155. HW_ICOLL_INTERRUPT67 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT67_SET(0,0x00000001);
```

5.4.78 Interrupt Collector Interrupt Register 68 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT68	0x560
HW_ICOLL_INTERRUPT68_SET	0x564
HW_ICOLL_INTERRUPT68_CLR	0x568
HW_ICOLL_INTERRUPT68_TOG	0x56C

Table 5-156. HW_ICOLL_INTERRUPT68

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-157. HW_ICOLL_INTERRUPT68 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT68_SET(0,0x00000001);
```

5.4.79 Interrupt Collector Interrupt Register 69 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT69	0x570
HW_ICOLL_INTERRUPT69_SET	0x574
HW_ICOLL_INTERRUPT69_CLR	0x578
HW_ICOLL_INTERRUPT69_TOG	0x57C

Table 5-158. HW_ICOLL_INTERRUPT69

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-159. HW_ICOLL_INTERRUPT69 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT69_SET(0,0x00000001);
```

5.4.80 Interrupt Collector Interrupt Register 70 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT70	0x580
HW_ICOLL_INTERRUPT70_SET	0x584
HW_ICOLL_INTERRUPT70_CLR	0x588
HW_ICOLL_INTERRUPT70_TOG	0x58C

Table 5-160. HW_ICOLL_INTERRUPT70

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-161. HW_ICOLL_INTERRUPT70 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT70_SET(0,0x00000001);
```

5.4.81 Interrupt Collector Interrupt Register 71 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT71	0x590
HW_ICOLL_INTERRUPT71_SET	0x594
HW_ICOLL_INTERRUPT71_CLR	0x598
HW_ICOLL_INTERRUPT71_TOG	0x59C

Table 5-162. HW_ICOLL_INTERRUPT71

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-163. HW_ICOLL_INTERRUPT71 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT71_SET(0,0x00000001);
```

5.4.82 Interrupt Collector Interrupt Register 72 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT72	0x5A0
HW_ICOLL_INTERRUPT72_SET	0x5A4
HW_ICOLL_INTERRUPT72_CLR	0x5A8
HW_ICOLL_INTERRUPT72_TOG	0x5AC

Table 5-164. HW_ICOLL_INTERRUPT72

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-165. HW_ICOLL_INTERRUPT72 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT72_SET(0,0x00000001);
```

5.4.83 Interrupt Collector Interrupt Register 73 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT73	0x5B0
HW_ICOLL_INTERRUPT73_SET	0x5B4
HW_ICOLL_INTERRUPT73_CLR	0x5B8
HW_ICOLL_INTERRUPT73_TOG	0x5BC

Table 5-166. HW_ICOLL_INTERRUPT73

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-167. HW_ICOLL_INTERRUPT73 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT73_SET(0,0x00000001);
```

5.4.84 Interrupt Collector Interrupt Register 74 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT74	0x5C0
HW_ICOLL_INTERRUPT74_SET	0x5C4
HW_ICOLL_INTERRUPT74_CLR	0x5C8
HW_ICOLL_INTERRUPT74_TOG	0x5CC

Table 5-168. HW_ICOLL_INTERRUPT74

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-169. HW_ICOLL_INTERRUPT74 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT74_SET(0,0x00000001);
```

5.4.85 Interrupt Collector Interrupt Register 75 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT75	0x5D0
HW_ICOLL_INTERRUPT75_SET	0x5D4
HW_ICOLL_INTERRUPT75_CLR	0x5D8
HW_ICOLL_INTERRUPT75_TOG	0x5DC

Table 5-170. HW_ICOLL_INTERRUPT75

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-171. HW_ICOLL_INTERRUPT75 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT75_SET(0,0x00000001);
```

5.4.86 Interrupt Collector Interrupt Register 76 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT76	0x5E0
HW_ICOLL_INTERRUPT76_SET	0x5E4
HW_ICOLL_INTERRUPT76_CLR	0x5E8
HW_ICOLL_INTERRUPT76_TOG	0x5EC

Table 5-172. HW_ICOLL_INTERRUPT76

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-173. HW_ICOLL_INTERRUPT76 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT76_SET(0,0x00000001);
```

5.4.87 Interrupt Collector Interrupt Register 77 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT77	0x5F0
HW_ICOLL_INTERRUPT77_SET	0x5F4
HW_ICOLL_INTERRUPT77_CLR	0x5F8
HW_ICOLL_INTERRUPT77_TOG	0x5FC

Table 5-174. HW_ICOLL_INTERRUPT77

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-175. HW_ICOLL_INTERRUPT77 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT77_SET(0,0x00000001);
```

5.4.88 Interrupt Collector Interrupt Register 78 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT78	0x600
HW_ICOLL_INTERRUPT78_SET	0x604
HW_ICOLL_INTERRUPT78_CLR	0x608
HW_ICOLL_INTERRUPT78_TOG	0x60C

Table 5-176. HW_ICOLL_INTERRUPT78

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-177. HW_ICOLL_INTERRUPT78 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT78_SET(0,0x00000001);
```

5.4.89 Interrupt Collector Interrupt Register 79 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT79	0x610
HW_ICOLL_INTERRUPT79_SET	0x614
HW_ICOLL_INTERRUPT79_CLR	0x618
HW_ICOLL_INTERRUPT79_TOG	0x61C

Table 5-178. HW_ICOLL_INTERRUPT79

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-179. HW_ICOLL_INTERRUPT79 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT79_SET(0,0x00000001);
```

5.4.90 Interrupt Collector Interrupt Register 80 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT80	0x620
HW_ICOLL_INTERRUPT80_SET	0x624
HW_ICOLL_INTERRUPT80_CLR	0x628
HW_ICOLL_INTERRUPT80_TOG	0x62C

Table 5-180. HW_ICOLL_INTERRUPT80

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-181. HW_ICOLL_INTERRUPT80 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT80_SET(0,0x00000001);
```

5.4.91 Interrupt Collector Interrupt Register 81 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT81	0x630
HW_ICOLL_INTERRUPT81_SET	0x634
HW_ICOLL_INTERRUPT81_CLR	0x638
HW_ICOLL_INTERRUPT81_TOG	0x63C

Table 5-182. HW_ICOLL_INTERRUPT81

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-183. HW_ICOLL_INTERRUPT81 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT81_SET(0,0x00000001);
```

5.4.92 Interrupt Collector Interrupt Register 82 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT82	0x640
HW_ICOLL_INTERRUPT82_SET	0x644
HW_ICOLL_INTERRUPT82_CLR	0x648
HW_ICOLL_INTERRUPT82_TOG	0x64C

Table 5-184. HW_ICOLL_INTERRUPT82

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-185. HW_ICOLL_INTERRUPT82 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT82_SET(0,0x00000001);
```

5.4.93 Interrupt Collector Interrupt Register 83 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT83	0x650
HW_ICOLL_INTERRUPT83_SET	0x654
HW_ICOLL_INTERRUPT83_CLR	0x658
HW_ICOLL_INTERRUPT83_TOG	0x65C

Table 5-186. HW_ICOLL_INTERRUPT83

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-187. HW_ICOLL_INTERRUPT83 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT83_SET(0,0x00000001);
```

5.4.94 Interrupt Collector Interrupt Register 84 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT84	0x660
HW_ICOLL_INTERRUPT84_SET	0x664
HW_ICOLL_INTERRUPT84_CLR	0x668
HW_ICOLL_INTERRUPT84_TOG	0x66C

Table 5-188. HW_ICOLL_INTERRUPT84

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-189. HW_ICOLL_INTERRUPT84 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT84_SET(0,0x00000001);
```

5.4.95 Interrupt Collector Interrupt Register 85 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT85	0x670
HW_ICOLL_INTERRUPT85_SET	0x674
HW_ICOLL_INTERRUPT85_CLR	0x678
HW_ICOLL_INTERRUPT85_TOG	0x67C

Table 5-190. HW_ICOLL_INTERRUPT85

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-191. HW_ICOLL_INTERRUPT85 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT85_SET(0,0x00000001);
```

5.4.96 Interrupt Collector Interrupt Register 86 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT86	0x680
HW_ICOLL_INTERRUPT86_SET	0x684
HW_ICOLL_INTERRUPT86_CLR	0x688
HW_ICOLL_INTERRUPT86_TOG	0x68C

Table 5-192. HW_ICOLL_INTERRUPT86

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-193. HW_ICOLL_INTERRUPT86 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT86_SET(0,0x00000001);
```

5.4.97 Interrupt Collector Interrupt Register 87 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT87	0x690
HW_ICOLL_INTERRUPT87_SET	0x694
HW_ICOLL_INTERRUPT87_CLR	0x698
HW_ICOLL_INTERRUPT87_TOG	0x69C

Table 5-194. HW_ICOLL_INTERRUPT87

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-195. HW_ICOLL_INTERRUPT87 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT87_SET(0,0x00000001);
```

5.4.98 Interrupt Collector Interrupt Register 88 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT88	0x6A0
HW_ICOLL_INTERRUPT88_SET	0x6A4
HW_ICOLL_INTERRUPT88_CLR	0x6A8
HW_ICOLL_INTERRUPT88_TOG	0x6AC

Table 5-196. HW_ICOLL_INTERRUPT88

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-197. HW_ICOLL_INTERRUPT88 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT88_SET(0,0x00000001);
```

5.4.99 Interrupt Collector Interrupt Register 89 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT89	0x6B0
HW_ICOLL_INTERRUPT89_SET	0x6B4
HW_ICOLL_INTERRUPT89_CLR	0x6B8
HW_ICOLL_INTERRUPT89_TOG	0x6BC

Table 5-198. HW_ICOLL_INTERRUPT89

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-199. HW_ICOLL_INTERRUPT89 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT89_SET(0,0x00000001);
```

5.4.100 Interrupt Collector Interrupt Register 90 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT90	0x6C0
HW_ICOLL_INTERRUPT90_SET	0x6C4
HW_ICOLL_INTERRUPT90_CLR	0x6C8
HW_ICOLL_INTERRUPT90_TOG	0x6CC

Table 5-200. HW_ICOLL_INTERRUPT90

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-201. HW_ICOLL_INTERRUPT90 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT90_SET(0,0x00000001);
```

5.4.101 Interrupt Collector Interrupt Register 91 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT91	0x6D0
HW_ICOLL_INTERRUPT91_SET	0x6D4
HW_ICOLL_INTERRUPT91_CLR	0x6D8
HW_ICOLL_INTERRUPT91_TOG	0x6DC

Table 5-202. HW_ICOLL_INTERRUPT91

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-203. HW_ICOLL_INTERRUPT91 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT91_SET(0,0x00000001);
```

5.4.102 Interrupt Collector Interrupt Register 92 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT92	0x6E0
HW_ICOLL_INTERRUPT92_SET	0x6E4
HW_ICOLL_INTERRUPT92_CLR	0x6E8
HW_ICOLL_INTERRUPT92_TOG	0x6EC

Table 5-204. HW_ICOLL_INTERRUPT92

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-205. HW_ICOLL_INTERRUPT92 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT92_SET(0,0x00000001);
```

5.4.103 Interrupt Collector Interrupt Register 93 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT93	0x6F0
HW_ICOLL_INTERRUPT93_SET	0x6F4
HW_ICOLL_INTERRUPT93_CLR	0x6F8
HW_ICOLL_INTERRUPT93_TOG	0x6FC

Table 5-206. HW_ICOLL_INTERRUPT93

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-207. HW_ICOLL_INTERRUPT93 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT93_SET(0,0x00000001);
```

5.4.104 Interrupt Collector Interrupt Register 94 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT94	0x700
HW_ICOLL_INTERRUPT94_SET	0x704
HW_ICOLL_INTERRUPT94_CLR	0x708
HW_ICOLL_INTERRUPT94_TOG	0x70C

Table 5-208. HW_ICOLL_INTERRUPT94

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-209. HW_ICOLL_INTERRUPT94 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT94_SET(0,0x00000001);
```

5.4.105 Interrupt Collector Interrupt Register 95 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT95	0x710
HW_ICOLL_INTERRUPT95_SET	0x714
HW_ICOLL_INTERRUPT95_CLR	0x718
HW_ICOLL_INTERRUPT95_TOG	0x71C

Table 5-210. HW_ICOLL_INTERRUPT95

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-211. HW_ICOLL_INTERRUPT95 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT95_SET(0,0x00000001);
```

5.4.106 Interrupt Collector Interrupt Register 96 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT96	0x720
HW_ICOLL_INTERRUPT96_SET	0x724
HW_ICOLL_INTERRUPT96_CLR	0x728
HW_ICOLL_INTERRUPT96_TOG	0x72C

Table 5-212. HW_ICOLL_INTERRUPT96

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-213. HW_ICOLL_INTERRUPT96 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT96_SET(0,0x00000001);
```

5.4.107 Interrupt Collector Interrupt Register 97 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT97	0x730
HW_ICOLL_INTERRUPT97_SET	0x734
HW_ICOLL_INTERRUPT97_CLR	0x738
HW_ICOLL_INTERRUPT97_TOG	0x73C

Table 5-214. HW_ICOLL_INTERRUPT97

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-215. HW_ICOLL_INTERRUPT97 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT97_SET(0,0x00000001);
```

5.4.108 Interrupt Collector Interrupt Register 98 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT98	0x740
HW_ICOLL_INTERRUPT98_SET	0x744
HW_ICOLL_INTERRUPT98_CLR	0x748
HW_ICOLL_INTERRUPT98_TOG	0x74C

Table 5-216. HW_ICOLL_INTERRUPT98

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-217. HW_ICOLL_INTERRUPT98 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT98_SET(0,0x00000001);
```

5.4.109 Interrupt Collector Interrupt Register 99 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT99	0x750
HW_ICOLL_INTERRUPT99_SET	0x754
HW_ICOLL_INTERRUPT99_CLR	0x758
HW_ICOLL_INTERRUPT99_TOG	0x75C

Table 5-218. HW_ICOLL_INTERRUPT99

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-219. HW_ICOLL_INTERRUPT99 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT99_SET(0,0x00000001);
```

5.4.110 Interrupt Collector Interrupt Register 100 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT100	0x760
HW_ICOLL_INTERRUPT100_SET	0x764
HW_ICOLL_INTERRUPT100_CLR	0x768
HW_ICOLL_INTERRUPT100_TOG	0x76C

Table 5-220. HW_ICOLL_INTERRUPT100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-221. HW_ICOLL_INTERRUPT100 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT100_SET(0,0x00000001);
```

5.4.111 Interrupt Collector Interrupt Register 101 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT101	0x770
HW_ICOLL_INTERRUPT101_SET	0x774
HW_ICOLL_INTERRUPT101_CLR	0x778
HW_ICOLL_INTERRUPT101_TOG	0x77C

Table 5-222. HW_ICOLL_INTERRUPT101

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-223. HW_ICOLL_INTERRUPT101 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT101_SET(0,0x00000001);
```

5.4.112 Interrupt Collector Interrupt Register 102 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT102	0x780
HW_ICOLL_INTERRUPT102_SET	0x784
HW_ICOLL_INTERRUPT102_CLR	0x788
HW_ICOLL_INTERRUPT102_TOG	0x78C

Table 5-224. HW_ICOLL_INTERRUPT102

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-225. HW_ICOLL_INTERRUPT102 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT102_SET(0,0x00000001);
```

5.4.113 Interrupt Collector Interrupt Register 103 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT103	0x790
HW_ICOLL_INTERRUPT103_SET	0x794
HW_ICOLL_INTERRUPT103_CLR	0x798
HW_ICOLL_INTERRUPT103_TOG	0x79C

Table 5-226. HW_ICOLL_INTERRUPT103

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-227. HW_ICOLL_INTERRUPT103 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT103_SET(0,0x00000001);
```

5.4.114 Interrupt Collector Interrupt Register 104 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT104	0x7A0
HW_ICOLL_INTERRUPT104_SET	0x7A4
HW_ICOLL_INTERRUPT104_CLR	0x7A8
HW_ICOLL_INTERRUPT104_TOG	0x7AC

Table 5-228. HW_ICOLL_INTERRUPT104

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-229. HW_ICOLL_INTERRUPT104 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT104_SET(0,0x00000001);
```

5.4.115 Interrupt Collector Interrupt Register 105 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT105	0x7B0
HW_ICOLL_INTERRUPT105_SET	0x7B4
HW_ICOLL_INTERRUPT105_CLR	0x7B8
HW_ICOLL_INTERRUPT105_TOG	0x7BC

Table 5-230. HW_ICOLL_INTERRUPT105

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-231. HW_ICOLL_INTERRUPT105 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT105_SET(0,0x00000001);
```

5.4.116 Interrupt Collector Interrupt Register 106 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT106	0x7C0
HW_ICOLL_INTERRUPT106_SET	0x7C4
HW_ICOLL_INTERRUPT106_CLR	0x7C8
HW_ICOLL_INTERRUPT106_TOG	0x7CC

Table 5-232. HW_ICOLL_INTERRUPT106

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-233. HW_ICOLL_INTERRUPT106 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT106_SET(0,0x00000001);
```

5.4.117 Interrupt Collector Interrupt Register 107 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT107	0x7D0
HW_ICOLL_INTERRUPT107_SET	0x7D4
HW_ICOLL_INTERRUPT107_CLR	0x7D8
HW_ICOLL_INTERRUPT107_TOG	0x7DC

Table 5-234. HW_ICOLL_INTERRUPT107

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-235. HW_ICOLL_INTERRUPT107 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT107_SET(0,0x00000001);
```

5.4.118 Interrupt Collector Interrupt Register 108 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT108	0x7E0
HW_ICOLL_INTERRUPT108_SET	0x7E4
HW_ICOLL_INTERRUPT108_CLR	0x7E8
HW_ICOLL_INTERRUPT108_TOG	0x7EC

Table 5-236. HW_ICOLL_INTERRUPT108

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-237. HW_ICOLL_INTERRUPT108 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT108_SET(0,0x00000001);
```

5.4.119 Interrupt Collector Interrupt Register 109 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT109	0x7F0
HW_ICOLL_INTERRUPT109_SET	0x7F4
HW_ICOLL_INTERRUPT109_CLR	0x7F8
HW_ICOLL_INTERRUPT109_TOG	0x7FC

Table 5-238. HW_ICOLL_INTERRUPT109

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-239. HW_ICOLL_INTERRUPT109 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT109_SET(0,0x00000001);
```

5.4.120 Interrupt Collector Interrupt Register 110 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT110	0x800
HW_ICOLL_INTERRUPT110_SET	0x804
HW_ICOLL_INTERRUPT110_CLR	0x808
HW_ICOLL_INTERRUPT110_TOG	0x80C

Table 5-240. HW_ICOLL_INTERRUPT110

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-241. HW_ICOLL_INTERRUPT110 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT110_SET(0,0x00000001);
```

5.4.121 Interrupt Collector Interrupt Register 111 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT111	0x810
HW_ICOLL_INTERRUPT111_SET	0x814
HW_ICOLL_INTERRUPT111_CLR	0x818
HW_ICOLL_INTERRUPT111_TOG	0x81C

Table 5-242. HW_ICOLL_INTERRUPT111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-243. HW_ICOLL_INTERRUPT111 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT111_SET(0,0x00000001);
```

5.4.122 Interrupt Collector Interrupt Register 112 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT112	0x820
HW_ICOLL_INTERRUPT112_SET	0x824
HW_ICOLL_INTERRUPT112_CLR	0x828
HW_ICOLL_INTERRUPT112_TOG	0x82C

Table 5-244. HW_ICOLL_INTERRUPT112

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-245. HW_ICOLL_INTERRUPT112 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT112_SET(0,0x00000001);
```

5.4.123 Interrupt Collector Interrupt Register 113 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT113	0x830
HW_ICOLL_INTERRUPT113_SET	0x834
HW_ICOLL_INTERRUPT113_CLR	0x838
HW_ICOLL_INTERRUPT113_TOG	0x83C

Table 5-246. HW_ICOLL_INTERRUPT113

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-247. HW_ICOLL_INTERRUPT113 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT113_SET(0,0x00000001);
```

5.4.124 Interrupt Collector Interrupt Register 114 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT114	0x840
HW_ICOLL_INTERRUPT114_SET	0x844
HW_ICOLL_INTERRUPT114_CLR	0x848
HW_ICOLL_INTERRUPT114_TOG	0x84C

Table 5-248. HW_ICOLL_INTERRUPT114

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-249. HW_ICOLL_INTERRUPT114 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT114_SET(0,0x00000001);
```

5.4.125 Interrupt Collector Interrupt Register 115 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT115	0x850
HW_ICOLL_INTERRUPT115_SET	0x854
HW_ICOLL_INTERRUPT115_CLR	0x858
HW_ICOLL_INTERRUPT115_TOG	0x85C

Table 5-250. HW_ICOLL_INTERRUPT115

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-251. HW_ICOLL_INTERRUPT115 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT115_SET(0,0x00000001);
```

5.4.126 Interrupt Collector Interrupt Register 116 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT116	0x860
HW_ICOLL_INTERRUPT116_SET	0x864
HW_ICOLL_INTERRUPT116_CLR	0x868
HW_ICOLL_INTERRUPT116_TOG	0x86C

Table 5-252. HW_ICOLL_INTERRUPT116

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-253. HW_ICOLL_INTERRUPT116 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT116_SET(0,0x00000001);
```

5.4.127 Interrupt Collector Interrupt Register 117 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT117	0x870
HW_ICOLL_INTERRUPT117_SET	0x874
HW_ICOLL_INTERRUPT117_CLR	0x878
HW_ICOLL_INTERRUPT117_TOG	0x87C

Table 5-254. HW_ICOLL_INTERRUPT117

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-255. HW_ICOLL_INTERRUPT117 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT117_SET(0,0x00000001);
```

5.4.128 Interrupt Collector Interrupt Register 118 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT118	0x880
HW_ICOLL_INTERRUPT118_SET	0x884
HW_ICOLL_INTERRUPT118_CLR	0x888
HW_ICOLL_INTERRUPT118_TOG	0x88C

Table 5-256. HW_ICOLL_INTERRUPT118

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-257. HW_ICOLL_INTERRUPT118 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT118_SET(0,0x00000001);
```

5.4.129 Interrupt Collector Interrupt Register 119 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT119	0x890
HW_ICOLL_INTERRUPT119_SET	0x894
HW_ICOLL_INTERRUPT119_CLR	0x898
HW_ICOLL_INTERRUPT119_TOG	0x89C

Table 5-258. HW_ICOLL_INTERRUPT119

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-259. HW_ICOLL_INTERRUPT119 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT119_SET(0,0x00000001);
```

5.4.130 Interrupt Collector Interrupt Register 120 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT120	0x8A0
HW_ICOLL_INTERRUPT120_SET	0x8A4
HW_ICOLL_INTERRUPT120_CLR	0x8A8
HW_ICOLL_INTERRUPT120_TOG	0x8AC

Table 5-260. HW_ICOLL_INTERRUPT120

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-261. HW_ICOLL_INTERRUPT120 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT120_SET(0,0x00000001);
```

5.4.131 Interrupt Collector Interrupt Register 121 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT121	0x8B0
HW_ICOLL_INTERRUPT121_SET	0x8B4
HW_ICOLL_INTERRUPT121_CLR	0x8B8
HW_ICOLL_INTERRUPT121_TOG	0x8BC

Table 5-262. HW_ICOLL_INTERRUPT121

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-263. HW_ICOLL_INTERRUPT121 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT121_SET(0,0x00000001);
```

5.4.132 Interrupt Collector Interrupt Register 122 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT122	0x8C0
HW_ICOLL_INTERRUPT122_SET	0x8C4
HW_ICOLL_INTERRUPT122_CLR	0x8C8
HW_ICOLL_INTERRUPT122_TOG	0x8CC

Table 5-264. HW_ICOLL_INTERRUPT122

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-265. HW_ICOLL_INTERRUPT122 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT122_SET(0,0x00000001);
```

5.4.133 Interrupt Collector Interrupt Register 123 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT123	0x8D0
HW_ICOLL_INTERRUPT123_SET	0x8D4
HW_ICOLL_INTERRUPT123_CLR	0x8D8
HW_ICOLL_INTERRUPT123_TOG	0x8DC

Table 5-266. HW_ICOLL_INTERRUPT123

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-267. HW_ICOLL_INTERRUPT123 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT123_SET(0,0x00000001);
```

5.4.134 Interrupt Collector Interrupt Register 124 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT124	0x8E0
HW_ICOLL_INTERRUPT124_SET	0x8E4
HW_ICOLL_INTERRUPT124_CLR	0x8E8
HW_ICOLL_INTERRUPT124_TOG	0x8EC

Table 5-268. HW_ICOLL_INTERRUPT124

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-269. HW_ICOLL_INTERRUPT124 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT124_SET(0,0x00000001);
```

5.4.135 Interrupt Collector Interrupt Register 125 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT125	0x8F0
HW_ICOLL_INTERRUPT125_SET	0x8F4
HW_ICOLL_INTERRUPT125_CLR	0x8F8
HW_ICOLL_INTERRUPT125_TOG	0x8FC

Table 5-270. HW_ICOLL_INTERRUPT125

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-271. HW_ICOLL_INTERRUPT125 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT125_SET(0,0x00000001);
```

5.4.136 Interrupt Collector Interrupt Register 126 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT126	0x900
HW_ICOLL_INTERRUPT126_SET	0x904
HW_ICOLL_INTERRUPT126_CLR	0x908
HW_ICOLL_INTERRUPT126_TOG	0x90C

Table 5-272. HW_ICOLL_INTERRUPT126

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-273. HW_ICOLL_INTERRUPT126 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT126_SET(0,0x00000001);
```

5.4.137 Interrupt Collector Interrupt Register 127 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT127	0x910
HW_ICOLL_INTERRUPT127_SET	0x914
HW_ICOLL_INTERRUPT127_CLR	0x918
HW_ICOLL_INTERRUPT127_TOG	0x91C

Table 5-274. HW_ICOLL_INTERRUPT127

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

Table 5-275. HW_ICOLL_INTERRUPT127 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT127_SET(0,0x00000001);
```

5.4.138 Interrupt Collector Debug Register 0 Description

The contents of this register will be defined as the hardware is developed.

HW_ICOLL_DEBUG	0x1120
HW_ICOLL_DEBUG_SET	0x1124
HW_ICOLL_DEBUG_CLR	0x1128
HW_ICOLL_DEBUG_TOG	0x112C

Table 5-276. HW_ICOLL_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
INSERVICE				LEVEL_REQUESTS				REQUESTS_BY_LEVEL				RSRVD2		FIQ	IRQ	RSRVD1						VECTOR_FSM									

Table 5-277. HW_ICOLL_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**INSERVICE**	RO	0x0	read-only view of the Inservice bits used for nesting IRQs. LEVEL0 = 0x1 LEVEL0 LEVEL1 = 0x2 LEVEL1 LEVEL2 = 0x4 LEVEL2 LEVEL3 = 0x8 LEVEL3
27:24	**LEVEL_REQUESTS**	RO	0x0	read-only view of the requsts by priority level for the current IRQ. LEVEL0 = 0x1 LEVEL0 LEVEL1 = 0x2 LEVEL1 LEVEL2 = 0x4 LEVEL2 LEVEL3 = 0x8 LEVEL3
23:20	**REQUESTS_BY_LEVEL**	RO	0x0	read-only view of the requsts by priority level for the current IRQ. LEVEL0 = 0x1 LEVEL0 LEVEL1 = 0x2 LEVEL1 LEVEL2 = 0x4 LEVEL2 LEVEL3 = 0x8 LEVEL3
19:18	**RSRVD2**	RO	0x0	Always write zeroes to this bitfield.
17	**FIQ**	RO	0x0	Read-Only View of the FIQ output to the CPU. NO_FIQ_REQUESTED = 0x0 No FIQ Requested FIQ_REQUESTED = 0x1 FIQ Requested
16	**IRQ**	RO	0x0	Read-Only View of the FIQ output to the CPU. NO_IRQ_REQUESTED = 0x0 No IRQ Requested IRQ_REQUESTED = 0x1 IRQ Requested
15:10	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
9:0	**VECTOR_FSM**	RO	0x0	Empty description. FSM_IDLE = 0x000 FSM_IDLE FSM_MULTICYCLE1 = 0x001 FSM_MULTICYCLE1 FSM_MULTICYCLE2 = 0x002 FSM_MULTICYCLE2 FSM_PENDING = 0x004 FSM_PENDING FSM_MULTICYCLE3 = 0x008 FSM_MULTICYCLE3 FSM_MULTICYCLE4 = 0x010 FSM_MULTICYCLE4 FSM_ISR_RUNNING1 = 0x020 FSM_ISR_RUNNING1 FSM_ISR_RUNNING2 = 0x040 FSM_ISR_RUNNING2 FSM_ISR_RUNNING3 = 0x080 FSM_ISR_RUNNING3 FSM_MULTICYCLE5 = 0x100 FSM_MULTICYCLE5 FSM_MULTICYCLE6 = 0x200 FSM_MULTICYCLE6

DESCRIPTION:

This register provides diagnostic visibility into the IRQ request state machine and its various inputs.

EXAMPLE:

```
if (BF_RD(ICOLL_DEBUG, LEVEL_REQUESTS) != HW_ICOLL_DEBUG_LEVEL_REQUESTS__LEVEL3)
Error();
TPRINTF(TP_MED, ("ICOLL INSERVICE = 0x%x
BF_RD(ICOLL_DEBUG, INSERVICE)));
TPRINTF(TP_MED, ("ICOLL STATE = 0x%x
VECTOR_FSM)));
```

5.4.139 Interrupt Collector Debug Read Register 0 Description

This register always returns a known read value for debug purposes.

HW_ICOLL_DBGREAD0	0x1130
HW_ICOLL_DBGREAD0_SET	0x1134
HW_ICOLL_DBGREAD0_CLR	0x1138
HW_ICOLL_DBGREAD0_TOG	0x113C

Table 5-278. HW_ICOLL_DBGREAD0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VALUE																															

Table 5-279. HW_ICOLL_DBGREAD0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**VALUE**	RO	0xECA94567	Fixed read-only value.

DESCRIPTION:

This register is used to test the read mux paths on the APBH.

EXAMPLE:

```
if (HW_ICOLL_DBGREAD0_RD != 0xECA94567)
Error();
```

5.4.140 Interrupt Collector Debug Read Register 1 Description

This register always returns a known read value for debug purposes.

HW_ICOLL_DBGREAD1	0x1140
HW_ICOLL_DBGREAD1_SET	0x1144
HW_ICOLL_DBGREAD1_CLR	0x1148
HW_ICOLL_DBGREAD1_TOG	0x114C

Table 5-280. HW_ICOLL_DBGREAD1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VALUE																															

Table 5-281. HW_ICOLL_DBGREAD1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**VALUE**	RO	0x1356DA98	Fixed read-only value.

DESCRIPTION:

This register is used to test the read mux paths on the APBH.

EXAMPLE:

```
if (HW_ICOLL_DBGREAD1_RD != 0x1356DA98)
Error();
```

5.4.141 Interrupt Collector Debug Flag Register Description

The Interrupt Collector debug flag register is used to post diagnostic state into simulation.

HW_ICOLL_DBGFLAG	0x1150
HW_ICOLL_DBGFLAG_SET	0x1154

HW_ICOLL_DBGFLAG_CLR	0x1158
HW_ICOLL_DBGFLAG_TOG	0x115C

Table 5-282. HW_ICOLL_DBGFLAG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																FLAG															

Table 5-283. HW_ICOLL_DBGFLAG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
15:0	**FLAG**	RW	0x0	This debug facility is probably temporary.

DESCRIPTION:

This register provides a posting register to synchronize C program execution and the internal simulation environment.

EXAMPLE:

```
BF_WR(ICOLL_DBGFLAG, FLAG, 3);
// ... do some diagnostic action
BF_WR(ICOLL_DBGFLAG, FLAG, 4);
// ... do some more diagnostic actions
BF_WR(ICOLL_DBGFLAG, FLAG, 5);
```

5.4.142 Interrupt Collector Debug Read Request Register 0 Description

read-only view into the low 32 bits of the request holding register.

HW_ICOLL_DBGREQUEST0	0x1160
HW_ICOLL_DBGREQUEST0_SET	0x1164
HW_ICOLL_DBGREQUEST0_CLR	0x1168
HW_ICOLL_DBGREQUEST0_TOG	0x116C

Table 5-284. HW_ICOLL_DBGREQUEST0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 5-285. HW_ICOLL_DBGREQUEST0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Low 32 bits of the request holding register.

DESCRIPTION:

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE:

```
if (HW_ICOLL_DBGREQUESTn_RD(0) != 0x00000000)
Error();
```

5.4.143 Interrupt Collector Debug Read Request Register 1 Description

read-only view into bits 32-63 of the request holding register.

HW_ICOLL_DBGREQUEST1	0x1170
HW_ICOLL_DBGREQUEST1_SET	0x1174
HW_ICOLL_DBGREQUEST1_CLR	0x1178
HW_ICOLL_DBGREQUEST1_TOG	0x117C

Table 5-286. HW_ICOLL_DBGREQUEST1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 5-287. HW_ICOLL_DBGREQUEST1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Bits 32-63 of the request holding register.

DESCRIPTION:

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE:

```
if (HW_ICOLL_DBGREQUESTn_RD(n) != 0x00000000)
Error();
```

5.4.144 Interrupt Collector Debug Read Request Register 2 Description

read-only view into bits 64-95 of the request holding register.

HW_ICOLL_DBGREQUEST2	0x1180
HW_ICOLL_DBGREQUEST2_SET	0x1184
HW_ICOLL_DBGREQUEST2_CLR	0x1188
HW_ICOLL_DBGREQUEST2_TOG	0x118C

Table 5-288. HW_ICOLL_DBGREQUEST2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 5-289. HW_ICOLL_DBGREQUEST2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Bits 64-95 of the request holding register.

DESCRIPTION:

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE:

```
if (HW_ICOLL_DBGREQUESTn_RD(n) != 0x00000000)
Error();
```

5.4.145 Interrupt Collector Debug Read Request Register 3 Description

read-only view into bits 96-127 of the request holding register.

HW_ICOLL_DBGREQUEST3	0x1190
HW_ICOLL_DBGREQUEST3_SET	0x1194
HW_ICOLL_DBGREQUEST3_CLR	0x1198
HW_ICOLL_DBGREQUEST3_TOG	0x119C

Table 5-290. HW_ICOLL_DBGREQUEST3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 5-291. HW_ICOLL_DBGREQUEST3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Bits 96-127 of the request holding register.

DESCRIPTION:

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE:

```
if (HW_ICOLL_DBGREQUESTn_RD(n) != 0x00000000)
Error();
```

5.4.146 Interrupt Collector Version Register Description

This register always returns a known read value for debug purposes it indicates the version of the block.

HW_ICOLL_VERSION	0x11E0

Table 5-292. HW_ICOLL_VERSION

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MAJOR								MINOR								STEP															

Table 5-293. HW_ICOLL_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**MAJOR**	RO	0x03	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	**MINOR**	RO	0x01	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	**STEP**	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

DESCRIPTION:

This register indicates the RTL version in use.

EXAMPLE:

```
if (HW_ICOLL_VERSION.B.MAJOR != 3)
Error();
```

ICOLL Block v3.1, Revision 1.50

5.4.147

Chapter 6
Digital Control and On-Chip RAM

This chapter describes the digital control block and the on-chip RAM features of the i.MX23. It includes sections on controlling the SRAM, performance monitors, high-entropy pseudo-random number seed, and free-running microseconds counter. Programmable registers for the block are described in Section 6.4, “Programmable Registers.”

6.1 Overview

The digital control block provides overall control of various items within the top digital block of the chip, including the on-chip RAM controls and HCLK performance counter, as shown in Figure 6-1.

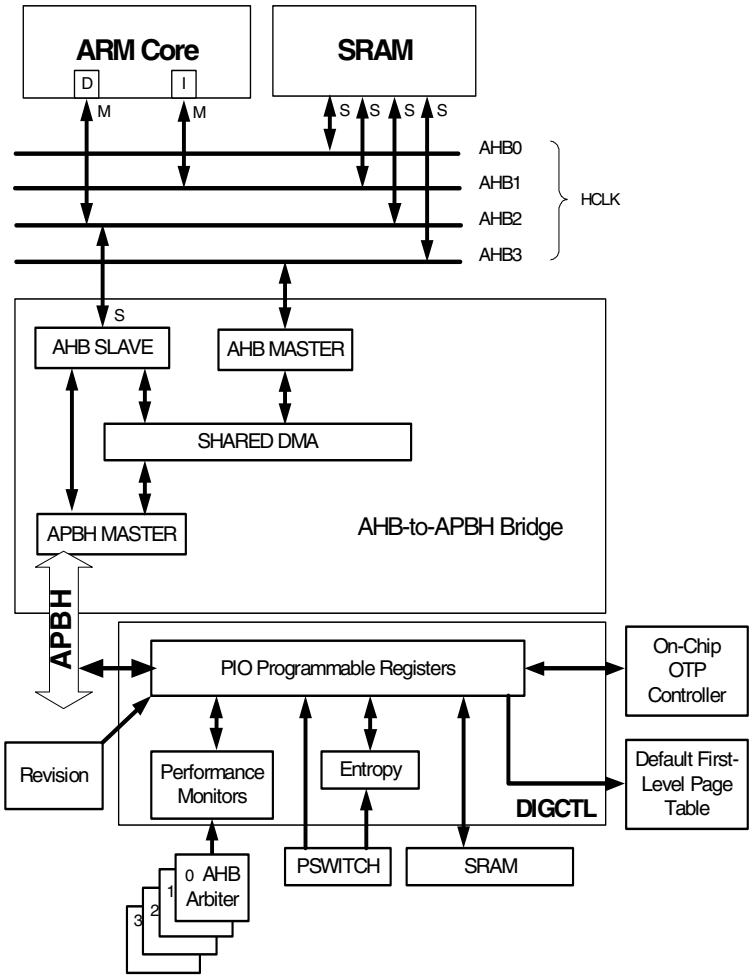


Figure 6-1. Digital Control (DIGCTL) Block Diagram

The on-chip RAM is constructed from an array of six-transistor dynamic RAM bit cells. The repair functions of this SRAM are controlled by registers in the DIGCTL block.

6.2 SRAM Controls

The on-chip RAM is a compiled RAM cell. It is implemented in one segment of 32 Kbytes. The memory is addressed as shown in Figure 6-2.

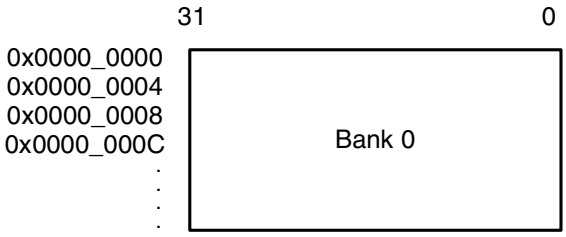


Figure 6-2. On-Chip RAM Partitioning

A 32-bit AHB address is arranged as shown in Table 6-1.

Table 6-1. On-Chip RAM Address Bits

AHB ADDR BITS	USAGE	DESCRIPTION
29:2	Address	Selects one of 32K words in a bank.
-	Bank Address	Tied 2'b00.
1:0	Byte Address	Selects/masks out specific bytes within a word.

Accessing on-chip RAM requires only one initial wait state for arbitration. There is a single cycle access for writes, and two cycle (one wait state) for reads. Other wait states happen whenever there is a read immediately following a write and also when a master is waiting to access a bank because some other master is accessing the same bank on a cycle.

The i.MX23 contains a simple 32-bit word RAM repair scheme. The purpose of this scheme is to address single-bit errors in the on-chip RAM. To enable the repair, HW_DIGCTL_RAMCTRL_RAM_REPAIR_EN must be set. Once this bit is set, the RAM controller replaces any access to the word address specified in HW_DIGCTL_RAMREPAIR_ADDR with a 32-bit redundant hardware memory. The actual repair enable and address must be read from OTP. Because these registers must be loaded by software, ROM boot code reads the OTP to see if the repair enable bit is set. If the repair enable bit is set, the ROM boot code sets HW_DIGCTL_RAMCTRL_RAM_REPAIR_EN and copies the 16-bit word address from OTP to HW_DIGCTL_RAMREPAIR_ADDR.

6.3 Miscellaneous Controls

The digital control block also contains a number of other miscellaneous functions, as detailed in this section.

6.3.1 Performance Monitoring

The digital control block contains several registers for system bus performance monitoring, including HW_DIGCTL_HCLKCOUNT, which counts HCLK rising edges. This register counts at a variable rate as HW_CLKCTRL_HBUS_AUTO_SLOW_ MODE is enabled.

In addition, there exists a performance monitoring register for each AHB layer (L0–L3). The HW_DIGCTL_L(n)_AHB_DATA_STALLED and HW_DIGCTL_L(n)_AHB_ACTIVE_CYCLES registers can be used to measure AHB bus utilization. The Stalled register counts all cycles in which any device has an outstanding and unfulfilled bus operation in flight. The Active Cycles register counts the number of data transfer cycles. Subtract cycles from stalls to determine under utilized bus cycles. These counters can be used to tune the performance of the HCLK frequency for specific activities. In addition, these monitors can be forced to focus on specific masters (which connect to that layer). See the HW_DIGCTL_AHB_STATS_SELECT bit description for details.

6.3.2 High-Entropy PRN Seed

A 32-bit entropy register begins running a pseudo-random number algorithm from the time reset is removed until the PSWITCH is released by the user. This high-entropy value can be used as the seed for other pseudo-random number generators.

6.3.3 Write-Once Register

A 32-bit write-once register holds a runtime-derived locked seed. Once written, it cannot be changed until the next chip wide reset event. The contents of this register are frequently derived from the entropy register.

6.3.4 Microseconds Counter

A 32-bit free-running microseconds counter provides fine-grain real-time control. Its period is determined by dividing the 24.0-MHz crystal oscillator by 24. Thus, its frequency does not change as HCLK, XCLK, and the processor clock frequency are changed.

6.4 Programmable Registers

The following registers provide control of all programmable elements of the digital control block.

6.4.1 DIGCTL Control Register Description

The DIGCTL Control Register provides overall control of various functions throughout the digital portion of the chip.

HW_DIGCTL_CTRL	0x000
HW_DIGCTL_CTRL_SET	0x004
HW_DIGCTL_CTRL_CLR	0x008
HW_DIGCTL_CTRL_TOG	0x00C

Table 6-2. HW_DIGCTL_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3	XTAL24M_GATE	TRAP_IRQ	RSVD2		CACHE_BIST_TMODE	LCD_BIST_CLKEN	LCD_BIST_START	DCP_BIST_CLKEN	DCP_BIST_START	ARM_BIST_CLKEN	USB_TESTMODE	ANALOG_TESTMODE	DIGITAL_TESTMODE	ARM_BIST_START	UART_LOOPBACK	SAIF_LOOPBACK	SAIF_CLKMUX_SEL		SAIF_CLKMST_SEL	SAIF_ALT_BITCLK_SEL	RSVD1	SY_ENDIAN	SY_SFTRST	SY_CLKGATE	USE_SERIAL_JTAG	TRAP_IN_RANGE	TRAP_ENABLE	DEBUG_DISABLE	USB_CLKGATE	JTAG_SHIELD	LATCH_ENTROPY

Table 6-3. HW_DIGCTL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSVD3**	RO	0x0	Reserved.
30	**XTAL24M_GATE**	RW	0x0	If set to 1, disable the Digital Control Microseconds counter, STRB1MHZ. If set to 0, enable the Digital Control Microseconds counter..
29	**TRAP_IRQ**	RW	0x0	This bit is set when an AHB access occurs to the range defined by the TRAP_ADDR registers below and the trap function is enabled with the TRAP_ENABLE bit.
28:27	**RSVD2**	RO	0x0	Reserved.
26	**CACHE_BIST_TMODE**	RW	0x0	Set this bit to enable the Cache BIST test mode.
25	**LCD_BIST_CLKEN**	RW	0x0	Set this bit to enable the LCD memory BIST clock.
24	**LCD_BIST_START**	RW	0x0	Set this bit to start the LCD memory BIST.
23	**DCP_BIST_CLKEN**	RW	0x0	Set this bit to enable the DCP memory BIST clock.
22	**DCP_BIST_START**	RW	0x0	Set this bit to start the DCP memory BIST.
21	**ARM_BIST_CLKEN**	RW	0x0	Set this bit to enable the ARM BIST clock.
20	**USB_TESTMODE**	RW	0x0	Set this bit to get into USB test mode.
19	**ANALOG_TESTMODE**	RW	0x0	Set this bit to get into analog test mode.
18	**DIGITAL_TESTMODE**	RW	0x0	Set this bit to get into digital test mode.
17	**ARM_BIST_START**	RW	0x0	Set this bit to start the ARM cache BIST controller.
16	**UART_LOOPBACK**	RW	0x0	Set this bit to loop the two AUARTs back on themselves in a null modem configuration (as well as connect AUART1 to DUART). NORMAL = 0x0 No loopback. LOOPIT = 0x1 Internally connect AUART1 TX to AUART2 RX and DUART RX, also connect AUART2 TX to AUART1 RX (note that DUART TX is unaffected).
15	**SAIF_LOOPBACK**	RW	0x0	Set this bit to loop SAIF1 to SAIF2 and SAIF2 to SAIF1. To use SAIF loopback, configure one SAIF for transmit and the other for receive. Because this bit connects SAIF1 output to SAIF2 input and SAIF2 output to SAIF1 input, it does not matter which of the two ports is configured for TX and the other for RX. Either configuration will produce an internal TX to RX loopback. Note that SAIF_CLKMST_SEL is ignored when loopback is enabled. NORMAL = 0x0 No loopback. LOOPIT = 0x1 Loop SAIF1 and SAIF2 back to each other.

Table 6-3. HW_DIGCTL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
14:13	**SAIF_CLKMUX_SEL**	RW	0x0	Selects the muxed pins and directions for the SAIF1 and SAIF2 master clock (MCLK), bit clock (BITCLK), and left/right sample clock (LRCLK). MCLK is an optional output and when used, is connected to the SAIF_MCLK_BITCLK muxed pin output. BITCLK can be either an input or output and can be connected to either the SAIF_MCLK_BITCLK muxed pin or the SAIF_ALT_BITCLK muxed pin. LRCLK can also be either an input or output and is connected to the SAIF_LRCLK muxed pin. When either MCLK, MCLK/BITCLK, or MCLK/BITCLK/LRCLK are configured to be outputs, the SAIF_CLKMST_SEL bit is used to determine which of the two SAIFs drives these clocks. Note that only one of the two SAIFs can be clock master at a time (READ_MODE=0 and/or SLAVE_MODE=0). When MCLK is not used and BITCLK/LRCLK are both inputs, one or both SAIFs must be configured as RX clock slaves (READ_MODE=1 and SLAVE_MODE=1). Valid configurations for SAIF_CLKMUX_SEL, as well as SAIF1 and SAIF2 SLAVE_MODE and READ_MODE bits, are: 1) one SAIF port is TX or RX master and controls BITCLK/LRCLK (both to the pins and to the other SAIF), and MCLK can optionally be an output while the other SAIF is an RX slave; 2) both ports are in RX slave mode and are driven by the BITCLK/LRCLK pin inputs, and MCLK can optionally be an output; 3) only one SAIF port is used as a TX or RX master during a given time, and SAIF_CLKMST_SEL is configured to give control of MCLK/BITCLK/LRCLK to the active port; or 4) only one of the two ports is used as an RX slave. See the table earlier in this chapter for a complete list of SAIF_CLKMUX_SEL/SAIF_CLKMST_SEL options, as well as SAIF1 and SAIF2 SLAVE_MODE/READ_MODE configurations. Note that SAIF_CLKMUX_SEL is ignored when SAIF_LOOPBACK=1. Also note that when the SAIF_ALT_BITCLK pinmux is selected to input/output BITCLK, 6-channel mode cannot be used since the SAIF2_SDATA2 pin is used for the alternate BITCLK. MBL_CLK_OUT = 0x0 MCLK output to SAIF_MCLK_BITCLK, BITCLK output to SAIF_ALT_BITCLK, LRCLK output to SAIF_LRCLK. BL_CLK_OUT = 0x1 BITCLK output to SAIF_MCLK_BITCLK, LRCLK output to SAIF_LRCLK. M_CLK_OUT_BL_CLK_IN = 0x2 MCLK output to SAIF_MCLK_BITCLK, BITCLK input to SAIF_ALT_BITCLK, LRCLK input to SAIF_LRCLK. BL_CLK_IN = 0x3 BITCLK input to SAIF_MCLK_BITCLK, LRCLK input to SAIF_LRCLK.

Table 6-3. HW_DIGCTL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
12	**SAIF_CLKMST_SEL**	RW	0x0	Selects whether SAIF1 or SAIF2 drives MCLK/BITCLK/LRCLK when they are configured as outputs via SAIF_CLKMUX_SEL. Note that the selected SAIF must be configured in clock master mode (READ_MODE=0 and/or SLAVE_MODE=0). This bit must also be configured when SAIF_LOOPBACK=1 to determine the clock master. SAIF1_MST = 0x0 SAIF1 clocks are used to output MCLK/BITCLK/LRCLK. SAIF2_MST = 0x1 SAIF2 clocks are used to output MCLK/BITCLK/LRCLK.
11	**SAIF_ALT_BITCLK_SEL**	RW	0x0	When the master SAIF (as selected by SAIF_CLKMST_SEL) requires all three clocks (MCLK/BITCLK/LRCLK), which is selected by programming SAIF_CLKMUX_SEL = 00 or 10, this bit selects whether SAIF2_SDATA2 or LCD_D16 is used to input/output BITCLK. 0 = SAIF2_SDATA2 pinmux pin selected for BITCLK. 1 = LCD_D16 pin selected for BITCLK. Note that this bit is ignored when SAIF_CLKMUX_SEL=01 or 11. Also note that the corresponding pin selected for BITCLK must have its MUXXEL bit field correctly programmed in the pin control block.
10	**RSVD1**	RO	0x0	Reserved.
9	**SY_ENDIAN**	RW	0x1	Setting this bit to 1 configures the SY to run in big endian mode, clearing this bit to 0 configures the SY to run in little endian mode.
8	**SY_SFTRST**	RW	0x1	Setting this bit to 1 forces a reset to the entire SY. SY_SFTRST has no effect on SY_CLKGATE. Also, the SY_SFTRST bit may be written when SY_CLKGATE=1. This bit must be cleared to 0 for normal SY operation.
7	**SY_CLKGATE**	RW	0x1	This bit gates the clocks to the SY to save power when the clocks are not in use. When set to 1, this bit gates off the clocks to the block. When this bit is cleared to 0, the block receives its clock for normal operation.
6	**USE_SERIAL_JTAG**	RW	0x0	Selects whether the one-wire serial JTAG interface or the alternative six-wire parallel JTAG interface is used. 0 = Parallel six-wire JTAG is enabled and is mapped to a collection of module pins that must be enabled by programming their MUXSEL bits in the pin control block. 1 = Serial JTAG is enabled and uses the dedicated DEBUG pin. The ROM bootcode writes this field prior to enabling JTAG, selecting which type of JTAG pin signaling to use. OLD_JTAG = 0x0 Use six-wire parallel JTAG mode. SERIAL_JTAG = 0x1 Use one-wire serial JTAG mode.

Table 6-3. HW_DIGCTL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
5	**TRAP_IN_RANGE**	RW	0x0	Determines whether the debug trap function causes a match when the master address is inside (low-address <= current-address <= high-address) the specified range. 0 = The trap occurs when the master address falls outside of the range. 1 = The check is inside the range.
4	**TRAP_ENABLE**	RW	0x0	Enables the AHB arbiter debug trap functions. When a trap occurs and this bit is set, an interrupt is sent to the ARM core.
3	**DEBUG_DISABLE**	RW	0x0	Set this bit to disable the ARM core's debug logic (for power savings). This bit must remain 0 following power-on reset for normal JTAG debugger operation of the ARM core. When set to 1, it gates off the clocks to the ARM core's debug logic. Once this bit is set, the part must undergo a power-on reset to re-enable debug operation. Manually clearing this bit via a write after it has been set produces unknown results.
2	**USB_CLKGATE**	RW	0x1	This bit must be cleared to 0 for normal operation of the USB controller. When set to 1, it gates off the clocks to the USB controller. USB_CLKGATE can be set during suspend to gate the USB clock during suspend. If this is gated, then the USB controller Reset Received bit (HW_USBCTRL_USBSTS_URI) should be not be polled for reset during suspend; use the HW_USBPHY_CTRL_RESUME_IRQ bit instead. RUN = 0x0 Allow USB to operate normally. NO_CLKS = 0x1 Do not clock USB gates in order to minimize power consumption.
1	**JTAG_SHIELD**	RW	0x0	0 = The JTAG debugger is enabled. 1 = The JTAG debugger is disabled. NORMAL = 0x0 JTAG debugger enabled. SHIELDS_UP = 0x1 JTAG debugger disabled.
0	**LATCH_ENTROPY**	RW	0x0	Setting this bit latches the current value of the entropy register into HW_DIGCTL_ENTROPY_VALUE. This can be used get a stable value on players that do not deassert the PSWITCH while powered up.

DESCRIPTION:

This register controls various functions throughout the digital portion of the chip.

EXAMPLE:

```
HW_DIGCTL_CTRL_CLR(BM_DIGCTL_CTRL_USB_CLKGATE); // enable USB clock
```

6.4.2 DIGCTL Status Register Description

The DIGCTL Status Register reports status for the digital control block.

HW_DIGCTL_STATUS	0x010
HW_DIGCTL_STATUS_SET	0x014
HW_DIGCTL_STATUS_CLR	0x018
HW_DIGCTL_STATUS_TOG	0x01C

Table 6-4. HW_DIGCTL_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
USB_HS_PRESENT	USB_OTG_PRESENT	USB_HOST_PRESENT	USB_DEVICE_PRESENT	RSVD2																	DCP_BIST_FAIL	DCP_BIST_PASS	DCP_BIST_DONE	LCD_BIST_FAIL	LCD_BIST_PASS	LCD_BIST_DONE	JTAG_IN_USE	PACKAGE_TYPE			WRITTEN

Table 6-5. HW_DIGCTL_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**USB_HS_PRESENT**	RO	0x1	This read-only bit returns a 1 when USB high-speed mode is present.
30	**USB_OTG_PRESENT**	RO	0x1	This read-only bit returns a 1 when USB on-the-go (OTG) functionality is present.
29	**USB_HOST_PRESENT**	RO	0x1	This read-only bit returns a 1 when USB host functionality is present.
28	**USB_DEVICE_PRESENT**	RO	0x1	This read-only bit returns a 1 when USB device functionality is present.
27:11	**RSVD2**	RO	0x0	Reserved.
10	**DCP_BIST_FAIL**	RO	0x0	This read-only bit is a 1 if the DCP memory BIST returns a failure.
9	**DCP_BIST_PASS**	RO	0x0	This read-only bit is a 1 if the DCP memory BIST returns a pass.
8	**DCP_BIST_DONE**	RO	0x0	This read-only bit is a 1 if the DCP memory BIST has completed.
7	**LCD_BIST_FAIL**	RO	0x0	This read-only bit is a 1 if the LCD memory BIST returns a failure.
6	**LCD_BIST_PASS**	RO	0x0	This read-only bit is a 1 if the LCD memory BIST returns a pass.
5	**LCD_BIST_DONE**	RO	0x0	This read-only bit is a 1 if the LCD memory BIST has completed.
4	**JTAG_IN_USE**	RO	0x0	This read-only bit is a 1 if JTAG debugger usage has been detected.
3:1	**PACKAGE_TYPE**	RO	0x0	This read-only bit field returns the pin count and package type. 000=169BGA, 011=128TQFP, all others=Reserved.
0	**WRITTEN**	RO	0x0	Set to 1 by any successful write to the HW_DIGCTL_WRITEONCE register.

DESCRIPTION:

The DIGCTL Status Register provides a read-only view to various input conditions and internal states.

EXAMPLE:

```
if(HW_DIGCTL_STATUS.PACKAGE_TYPE)
{
```

```
// do 100-pin package things
}
```

6.4.3 Free-Running HCLK Counter Register Description

The Free-Running HCLK Counter Register is available for performance metrics.

HW_DIGCTL_HCLKCOUNT	0x020
HW_DIGCTL_HCLKCOUNT_SET	0x024
HW_DIGCTL_HCLKCOUNT_CLR	0x028
HW_DIGCTL_HCLKCOUNT_TOG	0x02C

Table 6-6. HW_DIGCTL_HCLKCOUNT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-7. HW_DIGCTL_HCLKCOUNT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RO	0x0	This counter counts up from reset using HCLK. The count is valid for HCLK frequencies greater than 2 MHz.

DESCRIPTION:

This counter increments once per HCLK rising edge.

EXAMPLE:

```
StartTime = HW_DIGCTL_HCLKCOUNT;
// Do something you want timed here
EndTime = HW_DIGCTL_HCLKCOUNT;
Duration = EndTime - StartTime; // make sure to handle rollover in a real application
```

6.4.4 On-Chip RAM Control Register Description

The On-Chip RAM Control Register holds on-chip SRAM control bit fields.

HW_DIGCTL_RAMCTRL	0x030
HW_DIGCTL_RAMCTRL_SET	0x034
HW_DIGCTL_RAMCTRL_CLR	0x038
HW_DIGCTL_RAMCTRL_TOG	0x03C

Table 6-8. HW_DIGCTL_RAMCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1																				SPEED_SELECT				RSVD0							RAM_REPAIR_EN

Table 6-9. HW_DIGCTL_RAMCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	**RSVD1**	RO	0x0	Reserved.
11:8	**SPEED_SELECT**	RW	0x0	Speed select for 16Kx32 OCRAM instances. Recommended value is 0x0. To be used for characterization. This value may have to be modified to allow lower voltage operation.
7:1	**RSVD0**	RO	0x0	Reserved.
0	**RAM_REPAIR_EN**	RW	0x0	Enable word repair for OCRAM, using the address specified in HW_DIGCTL_RAMREPAIR.

DESCRIPTION:

This register controls various parts of the on-chip RAM, including the repair state machine that shifts the repair configuration data into the SRAM macro-cell.

EXAMPLE:

```
HW_DIGCTL_RAMCTRL_SET(BM_DIGCTL_RAMCTRL_REPAIR_TRANSMIT); // Start the efuse state machine
```

6.4.5 On-Chip RAM Repair Address Register Description

HW_DIGCTL_RAMREPAIR	0x040
HW_DIGCTL_RAMREPAIR_SET	0x044
HW_DIGCTL_RAMREPAIR_CLR	0x048
HW_DIGCTL_RAMREPAIR_TOG	0x04C

Table 6-10. HW_DIGCTL_RAMREPAIR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1																ADDR															

Table 6-11. HW_DIGCTL_RAMREPAIR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSVD1**	RO	0x0	Reserved.
15:0	**ADDR**	RW	0x0	Word repair address for OCRAM. Must be read from OTP and copied to this register. The repair is enabled when HW_DIGCTL_RAMCTRL_RAM_REPAIR_EN is set.

DESCRIPTION:

The On-Chip RAM Repair Address Register holds repair address for the on-chip SRAM. The value must be read from the OTP and copied here.

EXAMPLE:

```
HW_DIGCTL_RAMREPAIR.ADDR= 0xBADA; // read modify write is ok
```

6.4.6 On-Chip ROM Control Register Description

HW_DIGCTL_ROMCTRL	0x050
HW_DIGCTL_ROMCTRL_SET	0x054
HW_DIGCTL_ROMCTRL_CLR	0x058
HW_DIGCTL_ROMCTRL_TOG	0x05C

Table 6-12. HW_DIGCTL_ROMCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0																												RD_MARGIN			

Table 6-13. HW_DIGCTL_ROMCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:4	**RSVD0**	RO	0x0	Reserved.
3:0	**RD_MARGIN**	RW	0x2	This field is used for setting the read margin for the on-chip ROM. It programs the sense amp differential setting and allows the trade-off between speed and robustness. This field should not be changed unless instructed by Freescale.

DESCRIPTION:

The On-Chip ROM Control Register provides settings for the OCROM.

EXAMPLE:

```
Empty Example.
```

6.4.7 Software Write-Once Register Description

The Software Write-Once Register hold the value used in software certification management.

HW_DIGCTL_WRITEONCE 0x060

Table 6-14. HW_DIGCTL_WRITEONCE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 6-15. HW_DIGCTL_WRITEONCE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0xA5A5A5A5	This field can be written only one time. The contents are not used by hardware.

DESCRIPTION:

This register is used to hold a portion of a certificate that is not mutable after software initialization.

EXAMPLE:

```
HW_DIGCTL_WRITEONCE.U = my_certificate;
```

6.4.8 Entropy Register Description

HW_DIGCTL_ENTROPY 0x090

Table 6-16. HW_DIGCTL_ENTROPY

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VALUE																															

Table 6-17. HW_DIGCTL_ENTROPY Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**VALUE**	RO	0x0	This read-only bit field always reads back the results of an entropy calculation. It is used to randomize the seeds for random number generators.

DESCRIPTION:

The Entropy register is a read-only test value register.

EXAMPLE:

```
while(HW_DIGCTL_STATUS.PSWITCH != 0)
{
//wait for pswitch to go away
}
HW_DIGCTL_WRITEONCE.BITS = rand(HW_DIGCTL_ENTROPY.VALUE);
```

6.4.9 Entropy Latched Register Description

HW_DIGCTL_ENTROPY_LATCHED 0x0A0

Table 6-18. HW_DIGCTL_ENTROPY_LATCHED

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VALUE																															

Table 6-19. HW_DIGCTL_ENTROPY_LATCHED Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**VALUE**	RO	0x0	When the LATCH_ENTROPY bit in the HW_DIGCTL_CTRL register is set to 1, the value of the HW_DIGCTL_ENTROPY register is latched into this register. This can be used to latch a stable random value on players where the PSWITCH is not deasserted after power-up.

DESCRIPTION:

The Entropy Latched Register is a read-only test value register.

EXAMPLE:

```
Empty Example.
```

6.4.10 SJTAG Debug Register Description

The SJTAG Debug Register controls various debug points within the SJTAG block and provides read-only views into the SJTAG state machines.

HW_DIGCTL_SJTAGDBG 0x0B0
HW_DIGCTL_SJTAGDBG_SET 0x0B4
HW_DIGCTL_SJTAGDBG_CLR 0x0B8
HW_DIGCTL_SJTAGDBG_TOG 0x0BC

Table 6-20. HW_DIGCTL_SJTAGDBG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								SJTAG_STATE								RSVD1					SJTAG_TDO	SJTAG_TDI	SJTAG_MODE	DELAYED_ACTIVE				ACTIVE	SJTAG_PIN_STATE	SJTAG_DEBUG_DATA	SJTAG_DEBUG_OE

Table 6-21. HW_DIGCTL_SJTAGDBG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSVD2**	RO	0x0	Reserved.
26:16	**SJTAG_STATE**	RO	0x2	This bitfield shows the state of the sjtag_state flip-flops inside the SJTAG controller. These bits implement the second state machine in the SJTAG block.
15:11	**RSVD1**	RO	0x0	Reserved.
10	**SJTAG_TDO**	RO	0x0	This bit shows the state of the ARM JTAG TDO signal as seen inside the SJTAG controller.
9	**SJTAG_TDI**	RO	0x0	This bit shows the state of the JTAG TDI capture FF inside the SJTAG controller.
8	**SJTAG_MODE**	RO	0x0	This bit shows the state of the JTAG mode capture FF inside the SJTAG controller.
7:4	**DELAYED_ACTIVE**	RO	0x0	This bitfield shows the state of the delay_onewire_active_reg FF inside the SJTAG controller. These bits implement the first state machine in the SJTAG block.
3	**ACTIVE**	RO	0x0	This bit shows the state of the onewire_active_reg FF inside the SJTAG controller.
2	**SJTAG_PIN_STATE**	RO	0x0	This bit reflects the state of the input driver sampling the SJTAG pin. When HW_DIGCTL_CTRL_USE_SERIAL_JTAG is cleared to 0, external source can pull the SJTAG pin high without starting the SJTAG state machines. In this mode, the SJTAG_PIN_STATE bit is used to confirm continuity from the pad to the SJTAG block.
1	**SJTAG_DEBUG_DATA**	RW	0x0	When HW_DIGCTL_CTRL_USE_SERIAL_JTAG is cleared to 0, then the SJTAG pin is placed in a diagnostic mode. In that case, this bit controls the input to the pad data drive signal. If HW_DIGCTL_CTRL_SJTAG_DEBUG_OE is set to 1, then this bit also controls the state of the SJTAG pin itself.
0	**SJTAG_DEBUG_OE**	RW	0x0	When HW_DIGCTL_CTRL_USE_SERIAL_JTAG is cleared to 0, then the SJTAG pin is placed in a diagnostic mode. In that case, this bit controls the input to the pad data output enable signal. Setting this bit to 1 turns on the SJTAG pad and drives it to the state indicated by HW_DIGCTL_CTRL_SJTAG_DEBUG_DATA.

6.4.11 Digital Control Microseconds Counter Register Description

The Digital Control Microseconds Counter Register is a read-only test value register.

HW_DIGCTL_MICROSECONDS	0x0C0
HW_DIGCTL_MICROSECONDS_SET	0x0C4
HW_DIGCTL_MICROSECONDS_CLR	0x0C8
HW_DIGCTL_MICROSECONDS_TOG	0x0CC

Table 6-22. HW_DIGCTL_MICROSECONDS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VALUE																															

Table 6-23. HW_DIGCTL_MICROSECONDS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**VALUE**	RW	0x0	This register maintains a 32-bit counter that increments at a 1-microsecond rate. The 1-MHz clock driving this counter is derived from the 24.0-MHz crystal oscillator. The count value is not preserved over power-downs. The 32-bit value wraps in less than two hours.

DESCRIPTION:

This fixed-rate timer always increments at 24.0 MHz divided by 24 or 1.0 MHz. It does not generate an interrupt.

EXAMPLE:

```
StartTime = HW_DIGCTL_MICROSECONDS_RD();
EndTime = HW_DIGCTL_MICROSECONDS_RD();
ElapsedTime = StartTime - EndTime; // WARNING, handle rollover in real software
```

6.4.12 Digital Control Debug Read Test Register Description

The Digital Control Debug Read Test Register is a read-only test value register.

HW_DIGCTL_DBGRD 0x0D0

Table 6-24. HW_DIGCTL_DBGRD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COMPLEMENT																															

Table 6-25. HW_DIGCTL_DBGRD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COMPLEMENT**	RO	0x789ABCDE	This read-only bit field always reads back the one's complement of the value in HW_DIGCTL_DBG.

DESCRIPTION:

This register is used for debugging purposes.

EXAMPLE:

```
debug_value = HW_DIGCTL_DBGRD_RD();
```

6.4.13 Digital Control Debug Register Description

The Digital Control Debug Register is a read-only test value register.

HW_DIGCTL_DBG 0x0E0

Table 6-26. HW_DIGCTL_DBG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VALUE																															

Table 6-27. HW_DIGCTL_DBG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**VALUE**	RO	0x87654321	This read-only bit field always reads back the fixed value 0x87654321.

DESCRIPTION:

This register is used for debugging purposes.

EXAMPLE:

```
debug_value = HW_DIGCTL_DBG_RD();
```

6.4.14 SRAM BIST Control and Status Register Description

The SRAM BIST Control and Status Register provides overall control of the integrated BIST engine.

HW_DIGCTL_OCRAM_BIST_CSR 0x0F0
HW_DIGCTL_OCRAM_BIST_CSR_SET 0x0F4
HW_DIGCTL_OCRAM_BIST_CSR_CLR 0x0F8
HW_DIGCTL_OCRAM_BIST_CSR_TOG 0x0FC

Table 6-28. HW_DIGCTL_OCRAM_BIST_CSR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1																					BIST_DEBUG_MODE	BIST_DATA_CHANGE	BIST_CLKEN	RSVD0				FAIL	PASS	DONE	START

Table 6-29. HW_DIGCTL_OCRAM_BIST_CSR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:11	**RSVD1**	RO	0x0	Reserved.
10	**BIST_DEBUG_MODE**	RW	0x0	OCRAM BIST debug mode select
9	**BIST_DATA_CHANGE**	RW	0x0	OCRAM BIST data background select.
8	**BIST_CLKEN**	RW	0x0	Enable clock gate for OCRAM BIST.
7:4	**RSVD0**	RO	0x0	Reserved.
3	**FAIL**	RO	0x0	BIST has failed.

Table 6-29. HW_DIGCTL_OCRAM_BIST_CSR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**PASS**	RO	0x0	BIST has passed.
1	**DONE**	RO	0x0	BIST has completed.
0	**START**	RW	0x0	Initiate BIST of internal memory when high.

DESCRIPTION:

This register is used to start off the BIST operation on two RAMS in the DMA block. The status signals are returned after the BIST operation is completed to this register.

EXAMPLE:

```
To start the BIST operation, set HW_DIGCTL_1TRAM_BIST_CSR = 0x00000001.
After the BIST is completed and the test passes, the contents of
HW_DIGCTL_1TRAM_BIST_CSR will be 0x00000007, as the DONE and PASS flags
will be set.
```

6.4.15 SRAM Status Register 0 Description

The SRAM Status Register 0 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS0	0x110
HW_DIGCTL_OCRAM_STATUS0_SET	0x114
HW_DIGCTL_OCRAM_STATUS0_CLR	0x118
HW_DIGCTL_OCRAM_STATUS0_TOG	0x11C

Table 6-30. HW_DIGCTL_OCRAM_STATUS0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA00																															

Table 6-31. HW_DIGCTL_OCRAM_STATUS0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA00**	RO	0x0	This read-only bit field contains the fail data for the first fail in block 0.

DESCRIPTION:

This register contains fail data for the first fail in block 0.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS0_RD();
```

6.4.16 SRAM Status Register 1 Description

The SRAM Status Register 1 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS1	0x120
HW_DIGCTL_OCRAM_STATUS1_SET	0x124
HW_DIGCTL_OCRAM_STATUS1_CLR	0x128
HW_DIGCTL_OCRAM_STATUS1_TOG	0x12C

Table 6-32. HW_DIGCTL_OCRAM_STATUS1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA01																															

Table 6-33. HW_DIGCTL_OCRAM_STATUS1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA01**	RO	0x0	This read-only bit field contains the fail data for the second fail in block 0.

DESCRIPTION:

This register contains fail data for the second fail in block 0.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS1_RD();
```

6.4.17 SRAM Status Register 2 Description

SRAM Status Register 2 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS2	0x130
HW_DIGCTL_OCRAM_STATUS2_SET	0x134
HW_DIGCTL_OCRAM_STATUS2_CLR	0x138
HW_DIGCTL_OCRAM_STATUS2_TOG	0x13C

Table 6-34. HW_DIGCTL_OCRAM_STATUS2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA10																															

Table 6-35. HW_DIGCTL_OCRAM_STATUS2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA10**	RO	0x0	This read-only bit field contains the fail data for the first fail in block 1.

DESCRIPTION:

This register contains fail data for the first fail in block 1.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS2_RD();
```

6.4.18 SRAM Status Register 3 Description

RAM Status Register 3 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS3	0x140
HW_DIGCTL_OCRAM_STATUS3_SET	0x144

HW_DIGCTL_OCRAM_STATUS3_CLR 0x148
HW_DIGCTL_OCRAM_STATUS3_TOG 0x14C

Table 6-36. HW_DIGCTL_OCRAM_STATUS3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA11																															

Table 6-37. HW_DIGCTL_OCRAM_STATUS3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA11**	RO	0x0	This read-only bit field contains the fail data for the second fail in block 1.

DESCRIPTION:

This register contains fail data for the second fail in block 1.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS3_RD();
```

6.4.19 SRAM Status Register 4 Description

SRAM Status Register 4 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS4 0x150
HW_DIGCTL_OCRAM_STATUS4_SET 0x154
HW_DIGCTL_OCRAM_STATUS4_CLR 0x158
HW_DIGCTL_OCRAM_STATUS4_TOG 0x15C

Table 6-38. HW_DIGCTL_OCRAM_STATUS4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA20																															

Table 6-39. HW_DIGCTL_OCRAM_STATUS4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA20**	RO	0x0	This read-only bit field contains the fail data for the first fail in block 2.

DESCRIPTION:

This register contains fail data for the first fail in block 2.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS4_RD();
```

6.4.20 SRAM Status Register 5 Description

SRAM Status Register 5 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS5	0x160
HW_DIGCTL_OCRAM_STATUS5_SET	0x164
HW_DIGCTL_OCRAM_STATUS5_CLR	0x168
HW_DIGCTL_OCRAM_STATUS5_TOG	0x16C

Table 6-40. HW_DIGCTL_OCRAM_STATUS5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA21																															

Table 6-41. HW_DIGCTL_OCRAM_STATUS5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA21**	RO	0x0	This read-only bit field contains the fail data for the second fail in block 2.

DESCRIPTION:

This register contains fail data for the second fail in block 2.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS5_RD();
```

6.4.21 SRAM Status Register 6 Description

SRAM Status Register 6 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS6	0x170
HW_DIGCTL_OCRAM_STATUS6_SET	0x174
HW_DIGCTL_OCRAM_STATUS6_CLR	0x178
HW_DIGCTL_OCRAM_STATUS6_TOG	0x17C

Table 6-42. HW_DIGCTL_OCRAM_STATUS6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA30																															

Table 6-43. HW_DIGCTL_OCRAM_STATUS6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA30**	RO	0x0	This read-only bit field contains the fail data for the first fail in block 3.

DESCRIPTION:

This register contains fail data for the first fail in block 3.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS6_RD();
```

6.4.22 SRAM Status Register 7 Description

SRAM Status Register 7 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS7	0x180
HW_DIGCTL_OCRAM_STATUS7_SET	0x184
HW_DIGCTL_OCRAM_STATUS7_CLR	0x188
HW_DIGCTL_OCRAM_STATUS7_TOG	0x18C

Table 6-44. HW_DIGCTL_OCRAM_STATUS7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA31																															

Table 6-45. HW_DIGCTL_OCRAM_STATUS7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA31**	RO	0x0	This read-only bit field contains the fail data for the second fail in block 3.

DESCRIPTION:

This register contains fail data for the second fail in block 3.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS7_RD();
```

6.4.23 SRAM Status Register 8 Description

SRAM Status Register 8 is a read-only fail address register.

HW_DIGCTL_OCRAM_STATUS8	0x190
HW_DIGCTL_OCRAM_STATUS8_SET	0x194
HW_DIGCTL_OCRAM_STATUS8_CLR	0x198
HW_DIGCTL_OCRAM_STATUS8_TOG	0x19C

Table 6-46. HW_DIGCTL_OCRAM_STATUS8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3			FAILADDR01													RSVD2			FAILADDR00												

Table 6-47. HW_DIGCTL_OCRAM_STATUS8 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD3**	RO	0x0	This field is unused.
28:16	**FAILADDR01**	RO	0x0	This read-only bit field contains the failing address for the second fail in block 0.
15:13	**RSVD2**	RO	0x0	This field is unused.
12:0	**FAILADDR00**	RO	0x0	This read-only bit field contains the failing address for the first fail in block 0.

DESCRIPTION:

This register contains fail data for the first and second failures in block 0.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS8_RD();
```

6.4.24 SRAM Status Register 9 Description

SRAM Status Register 9 is a read-only fail address register.

HW_DIGCTL_OCRAM_STATUS9	0x1A0
HW_DIGCTL_OCRAM_STATUS9_SET	0x1A4
HW_DIGCTL_OCRAM_STATUS9_CLR	0x1A8
HW_DIGCTL_OCRAM_STATUS9_TOG	0x1AC

Table 6-48. HW_DIGCTL_OCRAM_STATUS9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3			FAILADDR11													RSVD2			FAILADDR10												

Table 6-49. HW_DIGCTL_OCRAM_STATUS9 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD3**	RO	0x0	This field is unused.
28:16	**FAILADDR11**	RO	0x0	This read-only bit field contains the failing address for the second fail in block 1.
15:13	**RSVD2**	RO	0x0	This field is unused.
12:0	**FAILADDR10**	RO	0x0	This read-only bit field contains the failing address for the first fail in block 1.

DESCRIPTION:

This register contains fail data for the first second failures in block 1.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS9_RD();
```

6.4.25 SRAM Status Register 10 Description

SRAM Status Register 10 is a read-only fail address register.

HW_DIGCTL_OCRAM_STATUS10	0x1B0
HW_DIGCTL_OCRAM_STATUS10_SET	0x1B4
HW_DIGCTL_OCRAM_STATUS10_CLR	0x1B8
HW_DIGCTL_OCRAM_STATUS10_TOG	0x1BC

Table 6-50. HW_DIGCTL_OCRAM_STATUS10

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3			FAILADDR21													RSVD2			FAILADDR20												

Table 6-51. HW_DIGCTL_OCRAM_STATUS10 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD3**	RO	0x0	This field is unused.
28:16	**FAILADDR21**	RO	0x0	This read-only bit field contains the failing address for the second fail in block 2.
15:13	**RSVD2**	RO	0x0	This field is unused.
12:0	**FAILADDR20**	RO	0x0	This read-only bit field contains the failing address for the first fail in block 2.

DESCRIPTION:

This register contains fail data for the first and second failures in block 2.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS10_RD();
```

6.4.26 SRAM Status Register 11 Description

SRAM Status Register 11 is a read-only fail address register.

HW_DIGCTL_OCRAM_STATUS11	0x1C0
HW_DIGCTL_OCRAM_STATUS11_SET	0x1C4
HW_DIGCTL_OCRAM_STATUS11_CLR	0x1C8
HW_DIGCTL_OCRAM_STATUS11_TOG	0x1CC

Table 6-52. HW_DIGCTL_OCRAM_STATUS11

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3				FAILADDR31												RSVD2				FAILADDR30											

Table 6-53. HW_DIGCTL_OCRAM_STATUS11 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD3**	RO	0x0	This field is unused.
28:16	**FAILADDR31**	RO	0x0	This read-only bit field contains the failing address for the second fail in block 3.
15:13	**RSVD2**	RO	0x0	This field is unused.
12:0	**FAILADDR30**	RO	0x0	This read-only bit field contains the failing address for the first fail in block 3.

DESCRIPTION:

This register contains fail data for the first and second failures in block 3.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS11_RD();
```

6.4.27 SRAM Status Register 12 Description

SRAM Status Register 12 is a read-only fail state register.

HW_DIGCTL_OCRAM_STATUS12	0x1D0
HW_DIGCTL_OCRAM_STATUS12_SET	0x1D4
HW_DIGCTL_OCRAM_STATUS12_CLR	0x1D8
HW_DIGCTL_OCRAM_STATUS12_TOG	0x1DC

Table 6-54. HW_DIGCTL_OCRAM_STATUS12

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3				FAILSTATE11				RSVD2				FAILSTATE10				RSVD1				FAILSTATE01				RSVD0				FAILSTATE00			

Table 6-55. HW_DIGCTL_OCRAM_STATUS12 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSVD3**	RO	0x0	This field is unused.
27:24	**FAILSTATE11**	RO	0x0	This read-only bit field contains the failing state for the second fail in block 1.

Table 6-55. HW_DIGCTL_OCRAM_STATUS12 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23:20	**RSVD2**	RO	0x0	This field is unused.
19:16	**FAILSTATE10**	RO	0x0	This read-only bit field contains the failing state for the first fail in block 1.
15:12	**RSVD1**	RO	0x0	This field is unused.
11:8	**FAILSTATE01**	RO	0x0	This read-only bit field contains the failing state for the second fail in block 0.
7:4	**RSVD0**	RO	0x0	This field is unused.
3:0	**FAILSTATE00**	RO	0x0	This read-only bit field contains the failing state for the first fail in block 0.

DESCRIPTION:

This register contains fail data for the first and second failures in blocks 0 and 1.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS12_RD();
```

6.4.28 SRAM Status Register 13 Description

SRAM Status Register 13 is a read-only fail state register.

HW_DIGCTL_OCRAM_STATUS13	0x1E0
HW_DIGCTL_OCRAM_STATUS13_SET	0x1E4
HW_DIGCTL_OCRAM_STATUS13_CLR	0x1E8
HW_DIGCTL_OCRAM_STATUS13_TOG	0x1EC

Table 6-56. HW_DIGCTL_OCRAM_STATUS13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3				FAILSTATE31				RSVD2				FAILSTATE30				RSVD1				FAILSTATE21				RSVD0				FAILSTATE20			

Table 6-57. HW_DIGCTL_OCRAM_STATUS13 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSVD3**	RO	0x0	This field is unused.
27:24	**FAILSTATE31**	RO	0x0	This read-only bit field contains the failing state for the second fail in block 3.
23:20	**RSVD2**	RO	0x0	This field is unused.
19:16	**FAILSTATE30**	RO	0x0	This read-only bit field contains the failing state for the first fail in block 3.
15:12	**RSVD1**	RO	0x0	This field is unused.
11:8	**FAILSTATE21**	RO	0x0	This read-only bit field contains the failing state for the second fail in block 2.

Table 6-57. HW_DIGCTL_OCRAM_STATUS13 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:4	**RSVD0**	RO	0x0	This field is unused.
3:0	**FAILSTATE20**	RO	0x0	This read-only bit field contains the failing state for the first fail in block 2.

DESCRIPTION:

This register contains fail data for the first and second failures in blocks 2 and 3.

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS0_RD();
```

6.4.29 Digital Control Scratch Register 0 Description

HW_DIGCTL_SCRATCH0 0x290

Table 6-58. HW_DIGCTL_SCRATCH0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PTR																															

Table 6-59. HW_DIGCTL_SCRATCH0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**PTR**	RW	0x0	

DESCRIPTION:

Scratch Pad Register 0.

EXAMPLE:

```
scratch_pad = (*void)HW_DIGCTL_SCRATCH0.PTR;
```

6.4.30 Digital Control Scratch Register 1 Description

HW_DIGCTL_SCRATCH1 0x2A0

Table 6-60. HW_DIGCTL_SCRATCH1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PTR																															

Table 6-61. HW_DIGCTL_SCRATCH1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**PTR**	RW	0x0	

DESCRIPTION:

Scratch Pad Register 1.

EXAMPLE:

```
scratch_pad = (*void)HW_DIGCTL_SCRATCH1.PTR;
```

6.4.31 Digital Control ARM Cache Register Description

This register provides the ARM cache RAM controls.

HW_DIGCTL_ARMCACHE 0x2B0

Table 6-62. HW_DIGCTL_ARMCACHE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4														VALID_SS		RSVD3		DRTY_SS		RSVD2		CACHE_SS		RSVD1		DTAG_SS		RSVD0		ITAG_SS	

Table 6-63. HW_DIGCTL_ARMCACHE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:18	**RSVD4**	RO	0x0	Reserved.
17:16	**VALID_SS**	RW	0x0	Timing Control for 64x24x1 RAMs (both instruction and data cache_valid arrays).
15:14	**RSVD3**	RO	0x0	Reserved.
13:12	**DRTY_SS**	RW	0x0	Timing Control for 128x8x1 RAM (DDRTY).
11:10	**RSVD2**	RO	0x0	Reserved.
9:8	**CACHE_SS**	RW	0x0	Timing Control for 1024x32x4 RAMs (both instruction and data cache arrays).
7:6	**RSVD1**	RO	0x0	Reserved.
5:4	**DTAG_SS**	RW	0x0	Timing Control for 256x22x4 RAM (DTAG).
3:2	**RSVD0**	RO	0x0	Reserved.
1:0	**ITAG_SS**	RW	0x0	Timing Control for 128x22x4 RAM (ITAG).

DESCRIPTION:

ARM Cache Control Register.

EXAMPLE:

```
cache_timing = HW_DIGCTL_ARMCACHE.CACHE_SS;
```

6.4.32 Debug Trap Range Low Address Description

The Debug Trap Range Low Address Register defines the lower bound for an address range that can be enabled to trigger an interrupt to the ARM core when an AHB cycle occurs within this range.

HW_DIGCTL_DEBUG_TRAP_ADDR_LOW 0x2C0

Table 6-64. HW_DIGCTL_DEBUG_TRAP_ADDR_LOW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDR																															

Table 6-65. HW_DIGCTL_DEBUG_TRAP_ADDR_LOW Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDR**	RW	0x0	This field contains the 32-bit lower address for the debug trap range.

DESCRIPTION:

This register sets the lower address that defines the debug trap function. When this function is enabled, any active AHB cycle on either Layer 0 or Layer 3 which accesses this range will trigger an interrupt to the ARM core.

6.4.33 Debug Trap Range High Address Description

The Debug Trap Range High Address Register defines the upper bound for an address range that can be enabled to trigger an interrupt to the ARM core when an AHB cycle occurs within this range.

HW_DIGCTL_DEBUG_TRAP_ADDR_HIGH 0x2D0

Table 6-66. HW_DIGCTL_DEBUG_TRAP_ADDR_HIGH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDR																															

Table 6-67. HW_DIGCTL_DEBUG_TRAP_ADDR_HIGH Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDR**	RW	0x0	This field contains the 32-bit upper address for the debug trap range.

DESCRIPTION:

This register sets the upper address that defines the debug trap function. When this function is enabled, any active AHB cycle on either Layer 0 or Layer 3 which accesses this range will trigger an interrupt to the ARM core.

6.4.34 Freescale Copyright Identifier Register Description

Read-only Freescale Copyright Identifier Register.

HW_DIGCTL_SGTL 0x300

Table 6-68. HW_DIGCTL_SGTL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COPYRIGHT																															

Table 6-69. HW_DIGCTL_SGTL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COPYRIGHT**	RO	0x6d676953	This read-only bit field contains the four bytes of the Freescale Copyright Identification String.

DESCRIPTION:

This register provides read-only access to the zero-terminated twelve-byte Freescale copyright identification string. This register behaves somewhat differently from all other APB registers in that it provides different read-back values at its three successive SCT bus addresses. The following binary values are read back at 0x300, 0x304, and 0x308 respectively:

0x6d676953 m,g,i,S at 0x300

0x6c655461 l,e,T,a at 0x304

0x00AEA92d 0x00, Registered Trademark (Æ), Copyright (©), hyphen (-) at 0x308

The debugger does a string compare on these 12 successive little endian bytes. Any chip that reads back these values is either a Freescale chip or it is a competitors chip that is violating Freescale registered trademarks and or copyrights.

EXAMPLE:

```
printf("%s", (char *)HW_DIGCTL_SGTL_ADDR);
```

6.4.35 Digital Control Chip Revision Register Description

Read-only chip revision register.

HW_DIGCTL_CHIPID 0x310

Table 6-70. HW_DIGCTL_CHIPID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PRODUCT_CODE																RSVD0								REVISION							

Table 6-71. HW_DIGCTL_CHIPID Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**PRODUCT_CODE**	RO	0x3780	This read-only bit field returns 0x3780, which identifies the generation from which the part is derived.
15:8	**RSVD0**	RO	0x0	Reserved.
7:0	**REVISION**	RO	0x00	This read-only bit field always reads back the mask revision level of the chip. 0x0 = TA1 0x1 = TA2 0x2 = TA3 0x3 = TA4

EXAMPLE:

```
FormatAndPrintChipID(HW_DIGCTL_CHIPID_PRODUCT_CODE,HW_DIGCTL_CHIPID_REVISION );
```

6.4.36 AHB Statistics Control Register Description

The AHB Statistics Control Register selects which AHB masters on each layer of the AHB subsystem are enabled to contribute to the statistics calculations.

HW_DIGCTL_AHB_STATS_SELECT 0x330

Table 6-72. HW_DIGCTL_AHB_STATS_SELECT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3				L3_MASTER_SELECT				RSVD2				L2_MASTER_SELECT				RSVD1				L1_MASTER_SELECT				RSVD0				L0_MASTER_SELECT			

Table 6-73. HW_DIGCTL_AHB_STATS_SELECT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSVD3**	RO	0x0	Reserved.
27:24	**L3_MASTER_SELECT**	RW	0x0	Set various bits of this bit field to one to enable performance monitoring in the AHB Layer 3 arbiter for the corresponding AHB master. APBH = 0x1 Select APBH DMA Master. APBX = 0x2 Select APBX DMA Master. USB = 0x4 Select USB Master.
23:20	**RSVD2**	RO	0x0	Reserved.
19:16	**L2_MASTER_SELECT**	RW	0x0	Set various bits of this bit field to one to enable performance monitoring in the AHB Layer 2 arbiter for the corresponding AHB master. ARM_D = 0x1 Select ARM D Master.
15:12	**RSVD1**	RO	0x0	Reserved.

Table 6-73. HW_DIGCTL_AHB_STATS_SELECT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11:8	**L1_MASTER_SELECT**	RW	0x0	Set various bits of this bit field to one to enable performance monitoring in the AHB Layer 1 arbiter for the corresponding AHB master. ARM_I = 0x1 Select ARM I Master.
7:4	**RSVD0**	RO	0x0	Reserved.
3:0	**L0_MASTER_SELECT**	RW	0x0	Set various bits of this bit field to one to enable performance monitoring in the AHB Layer 0 arbiter for the corresponding AHB master. ECC8 = 0x1 Select ECC8 Master. CRYPTO = 0x2 Select Crypto Master.

6.4.37 AHB Layer 0 Transfer Count Register Description

The AHB Layer 0 Transfer Count Register counts the number of AHB bus cycles during which a transfer is active on AHB Layer 0.

HW_DIGCTL_L0_AHB_ACTIVE_CYCLES 0x340

Table 6-74. HW_DIGCTL_L0_AHB_ACTIVE_CYCLES

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-75. HW_DIGCTL_L0_AHB_ACTIVE_CYCLES Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field contains the count of AHB bus cycles during which a master was active on the AHB Layer 0.

DESCRIPTION:

This field counts the number of AHB cycles in which a master was requesting a transfer, and the slave had not responded. This includes cycles in which it was requesting transfers but was not granted them, as well as cycles in which it was granted and driving the bus but the targeted slave was not ready. The master selects in HW_DIGCTL_AHB_STATS_SELECT_L0_MASTER_SELECT are used in the arbiter to mask which master's cycles are actually recorded here.

EXAMPLE:

```
NumberCycles = HW_DIGCTL_L0_AHB_ACTIVE_CYCLES_COUNT_RD();
```

6.4.38 AHB Layer 0 Performance Metric for Stalled Bus Cycles Register Description

Used for AHB bus utilization measurements, the AHB Performance Metric for Stalled Bus Cycles Register counts the number of stalled AHB cycles.

HW_DIGCTL_L0_AHB_DATA_STALLED 0x350

Table 6-76. HW_DIGCTL_L0_AHB_DATA_STALLED

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-77. HW_DIGCTL_L0_AHB_DATA_STALLED Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field counts the number of AHB cycles in which a master was stalled.

DESCRIPTION:

This counter increments on a data-phase of the AHB in which the HREADY signal is low, indicating a stalled data transfer.

EXAMPLE:

```
NumberStalledCycles = HW_DIGCTL_L0_AHB_DATA_STALLED_COUNT_RD();
```

6.4.39 AHB Layer 0 Performance Metric for Valid Bus Cycles Register Description

Used for AHB bus utilization measurements, the AHB Performance Metric for Valid Bus Cycles Register counts the number of actual AHB cycles in which a data transfer is completed.

HW_DIGCTL_L0_AHB_DATA_CYCLES 0x360

Table 6-78. HW_DIGCTL_L0_AHB_DATA_CYCLES

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-79. HW_DIGCTL_L0_AHB_DATA_CYCLES Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field contains the count of AHB bus cycles during which data was actually transferred from a master to a slave or from a slave to a master.

DESCRIPTION:

This field counts the number of AHB cycles in which a master completed a data transfer (a data-phase in which HREADY is high).

EXAMPLE:

```
StartTime = HW_DIGCTL_HCLKCOUNT_RD();
while(HW_DIGCTL_L0_AHB_DATA_CYCLES.COUNT less than 1000000)
{
// wait for a specific number of xfers
}
ElapsedTime = HW_DIGCTL_HCLKCOUNT_RD() - StartTime;
```

6.4.40 AHB Layer 1 Transfer Count Register Description

The AHB Layer 1 Transfer Count Register counts the number of AHB bus cycles during which a transfer is active on AHB Layer 1.

HW_DIGCTL_L1_AHB_ACTIVE_CYCLES 0x370

Table 6-80. HW_DIGCTL_L1_AHB_ACTIVE_CYCLES

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-81. HW_DIGCTL_L1_AHB_ACTIVE_CYCLES Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field contains the count of AHB bus cycles during which a master was active on the AHB Layer 1.

DESCRIPTION:

This field counts the number of AHB cycles in which a master was requesting a transfer, and the slave had not responded. This includes cycles in which it was requesting transfers but was not granted them, as well as cycles in which it was granted and driving the bus but the targeted slave was not ready. The master selects in HW_DIGCTL_AHB_STATS_SELECT_L1_MASTER_SELECT are used in the arbiter to mask which master's cycles are actually recorded here.

EXAMPLE:

```
NumberCycles = HW_DIGCTL_L1_AHB_ACTIVE_CYCLES_COUNT_RD();
```

6.4.41 AHB Layer 1 Performance Metric for Stalled Bus Cycles Register Description

Used for AHB bus utilization measurements, the AHB Performance Metric for Stalled Bus Cycles Register counts the number of stalled AHB cycles.

HW_DIGCTL_L1_AHB_DATA_STALLED 0x380

Table 6-82. HW_DIGCTL_L1_AHB_DATA_STALLED

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-83. HW_DIGCTL_L1_AHB_DATA_STALLED Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field counts the number of AHB cycles in which a master was stalled.

DESCRIPTION:

This counter increments on a data-phase of the AHB in which the HREADY signal is low, indicating a stalled data transfer.

EXAMPLE:

```
NumberStalledCycles = HW_DIGCTL_L1_AHB_DATA_STALLED_COUNT_RD();
```

6.4.42 AHB Layer 1 Performance Metric for Valid Bus Cycles Register Description

Used for AHB bus utilization measurements, the AHB Performance Metric for Valid Bus Cycles Register counts the number of actual AHB cycles in which a data transfer is completed.

HW_DIGCTL_L1_AHB_DATA_CYCLES 0x390

Table 6-84. HW_DIGCTL_L1_AHB_DATA_CYCLES

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-85. HW_DIGCTL_L1_AHB_DATA_CYCLES Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field contains the count of AHB bus cycles during which data was actually transferred from a master to a slave or from a slave to a master.

DESCRIPTION:

This field counts the number of AHB cycles in which a master completed a data transfer (a data-phase in which HREADY is high).

EXAMPLE:

```
StartTime = HW_DIGCTL_HCLKCOUNT_RD();
while(HW_DIGCTL_L1_AHB_DATA_CYCLES.COUNT less than 1000000)
{
// wait for a specific number of xfers
}
ElapsedTime = HW_DIGCTL_HCLKCOUNT_RD() - StartTime;
```

6.4.43 AHB Layer 2 Transfer Count Register Description

The AHB Layer 2 Transfer Count Register counts the number of AHB bus cycles during which a transfer is active on AHB Layer 2.

HW_DIGCTL_L2_AHB_ACTIVE_CYCLES 0x3A0

Table 6-86. HW_DIGCTL_L2_AHB_ACTIVE_CYCLES

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-87. HW_DIGCTL_L2_AHB_ACTIVE_CYCLES Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field contains the count of AHB bus cycles during which a master was active on the AHB Layer 2.

DESCRIPTION:

This field counts the number of AHB cycles in which a master was requesting a transfer, and the slave had not responded. This includes cycles in which it was requesting transfers but was not granted them, as well as cycles in which it was granted and driving the bus but the targeted slave was not ready. The master selects in HW_DIGCTL_AHB_STATS_SELECT_L2_MASTER_SELECT are used in the arbiter to mask which master's cycles are actually recorded here.

EXAMPLE:

```
NumberCycles = HW_DIGCTL_L2_AHB_ACTIVE_CYCLES_COUNT_RD();
```

6.4.44 AHB Layer 2 Performance Metric for Stalled Bus Cycles Register Description

Used for AHB bus utilization measurements, the AHB Performance Metric for Stalled Bus Cycles Register counts the number of stalled AHB cycles.

HW_DIGCTL_L2_AHB_DATA_STALLED 0x3B0

Table 6-88. HW_DIGCTL_L2_AHB_DATA_STALLED

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-89. HW_DIGCTL_L2_AHB_DATA_STALLED Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field counts the number of AHB cycles in which a master was stalled.

DESCRIPTION:

This counter increments on a data-phase of the AHB in which the HREADY signal is low, indicating a stalled data transfer.

EXAMPLE:

```
NumberStalledCycles = HW_DIGCTL_L2_AHB_DATA_STALLED_COUNT_RD();
```

6.4.45 AHB Layer 2 Performance Metric for Valid Bus Cycles Register Description

Used for AHB bus utilization measurements, the AHB Performance Metric for Valid Bus Cycles Register counts the number of actual AHB cycles in which a data transfer is completed.

HW_DIGCTL_L2_AHB_DATA_CYCLES 0x3C0

Table 6-90. HW_DIGCTL_L2_AHB_DATA_CYCLES

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-91. HW_DIGCTL_L2_AHB_DATA_CYCLES Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field contains the count of AHB bus cycles during which data was actually transferred from a master to a slave or from a slave to a master.

DESCRIPTION:

This field counts the number of AHB cycles in which a master completed a data transfer (a data-phase in which HREADY is high).

EXAMPLE:

```
StartTime = HW_DIGCTL_HCLKCOUNT_RD();
while(HW_DIGCTL_L2_AHB_DATA_CYCLES.COUNT less than 1000000)
{
// wait for a specific number of xfers
}
ElapsedTime = HW_DIGCTL_HCLKCOUNT_RD() - StartTime;
```

6.4.46 AHB Layer 3 Transfer Count Register Description

The AHB Layer 3 Transfer Count Register counts the number of AHB bus cycles during which a transfer is active on AHB Layer 3.

HW_DIGCTL_L3_AHB_ACTIVE_CYCLES 0x3D0

Table 6-92. HW_DIGCTL_L3_AHB_ACTIVE_CYCLES

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-93. HW_DIGCTL_L3_AHB_ACTIVE_CYCLES Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field contains the count of AHB bus cycles during which a master was active on the AHB Layer 3.

DESCRIPTION:

This field counts the number of AHB cycles in which a master was requesting a transfer, and the slave had not responded. This includes cycles in which it was requesting transfers but was not granted them, as well as cycles in which it was granted and driving the bus but the targeted slave was not ready. The master selects in HW_DIGCTL_AHB_STATS_SELECT_L3_MASTER_SELECT are used in the arbiter to mask which master's cycles are actually recorded here.

EXAMPLE:

```
NumberCycles = HW_DIGCTL_L3_AHB_ACTIVE_CYCLES_COUNT_RD();
```

6.4.47 AHB Layer 3 Performance Metric for Stalled Bus Cycles Register Description

Used for AHB bus utilization measurements, the AHB Performance Metric for Stalled Bus Cycles Register counts the number of stalled AHB cycles.

HW_DIGCTL_L3_AHB_DATA_STALLED 0x3E0

Table 6-94. HW_DIGCTL_L3_AHB_DATA_STALLED

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-95. HW_DIGCTL_L3_AHB_DATA_STALLED Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field counts the number of AHB cycles in which a master was stalled.

DESCRIPTION:

This counter increments on a data-phase of the AHB in which the HREADY signal is low, indicating a stalled data transfer.

EXAMPLE:

```
NumberStalledCycles = HW_DIGCTL_L3_AHB_DATA_STALLED_COUNT_RD();
```

6.4.48 AHB Layer 3 Performance Metric for Valid Bus Cycles Register Description

Used for AHB bus utilization measurements, the AHB Performance Metric for Valid Bus Cycles Register counts the number of actual AHB cycles in which a data transfer is completed.

HW_DIGCTL_L3_AHB_DATA_CYCLES 0x3F0

Table 6-96. HW_DIGCTL_L3_AHB_DATA_CYCLES

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-97. HW_DIGCTL_L3_AHB_DATA_CYCLES Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field contains the count of AHB bus cycles during which data was actually transferred from a master to a slave or from a slave to a master.

DESCRIPTION:

This field counts the number of AHB cycles in which a master completed a data transfer (a data-phase in which HREADY is high).

EXAMPLE:

```
StartTime = HW_DIGCTL_HCLKCOUNT_RD();
while(HW_DIGCTL_L3_AHB_DATA_CYCLES.COUNT less than 1000000)
{
// wait for a specific number of xfers
}
ElapsedTime = HW_DIGCTL_HCLKCOUNT_RD() - StartTime;
```

6.4.49 EMI CLK/CLKN Delay Adjustment Description

Used to adjust delay of the EMI_CLK and EMI_CLKN.

HW_DIGCTL_EMICLK_DELAY 0x500

Table 6-98. HW_DIGCTL_EMICLK_DELAY

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0																											NUM_TAPS			

Table 6-99. HW_DIGCTL_EMICLK_DELAY Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSVD0**	RO	0x0	Reserved.
4:0	**NUM_TAPS**	RW	0x0	Select one of 32 delay line taps to delay the EMI_CLK and EMI_CLKN pins

DESCRIPTION:

This register setting is useful in cases where additional setup time is required (at the expense of hold) for EMI Address/Command signals in SDRAM and mDDR modes. It can also be used to give additional write data setup time in SDRAM modes.

EXAMPLE:

```
Empty Example.
```

DIGCTL Block digctl, Revision 1.0

Chapter 7
On-Chip OTP (OCOTP) Controller

This chapter describes the on-chip OTP (OCOTP) controller included on the i.MX23. Programmable registers are described in Section 7.4, “Programmable Registers.”

7.1 Overview

The on-chip OTP controller (OCOTP) provides the following functions:

- Full memory-mapped (restricted) read access of 1 Kbit of on-chip OTP ROM.
- Data-register programming interface for the 1 Kbit of OTP.
- Generation of the chip hardware capability bus.
- Chip-level pin access to nonrestricted portions of OTP.

The OCOTP is connected to the APBH system peripheral bus and is accessible via the ARM core Data-AHB layer (Layer 2). Read accesses can be done at maximum HCLK frequency. Programming/writes can be performed at 24 MHz. The system diagram for the OCOTP is shown in Figure 7-1.

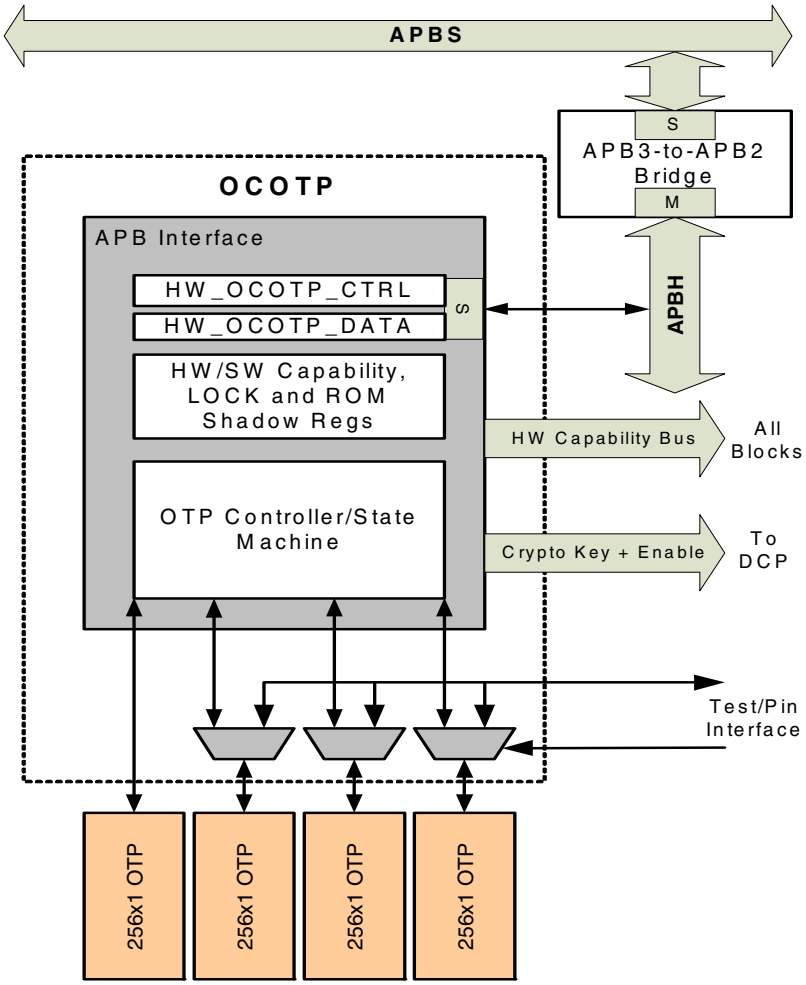


Figure 7-1. On-Chip OTP (OCOTP) Controller Block Diagram

7.2 Operation

The APB interface of the OCOTP provides two functions:

- Programmer-model access to registers (see Section 7.4, “Programmable Registers,” for register details). These operations require a bank opening sequence via HW_OCOTP_CTRL_RD_BANK_OPEN.
- Restricted 32-bit word write/program access to the 1-Kbit OTP

The OTP is divided into 32-bit words (32 in total). All the 32 words are memory-mapped to APBH addresses (for reads only). Writes require the use of HW_OCOTP_CTRL_ADDR. The customer view of the high-level OTP allocation is shown in Figure 7-2.

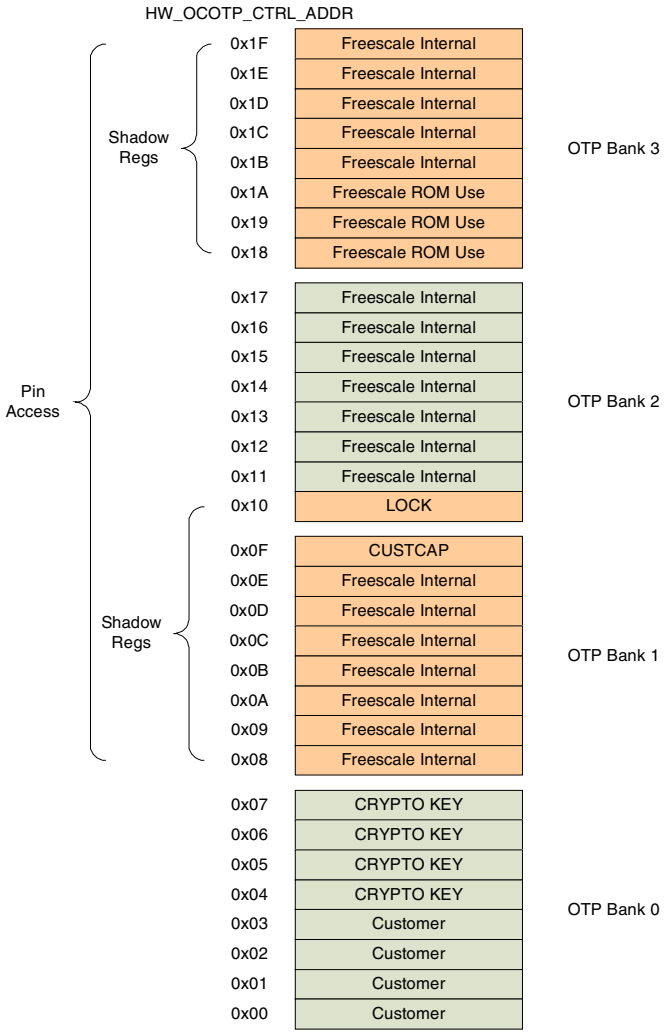


Figure 7-2. OCOTP Allocation

OTP reads and writes can be performed on 32-bit words only. For writes, the 32-bit word reflects the “write-mask”, such that bit fields with 0 will not be programmed and bit fields with 1 will be programmed.

For OTP random access, the programming interface consists of:

- HW_OCOTP_DATA—Data register (32- bit) for OTP programming (writes).
- HW_OCOTP_CTRL_ADDR—Address register (5-bit) for OTP programming (writes).
- HW_OCOTP_CTRL_BUSY—Programming/write request/status handshake bit.
- HW_OCOTP_CTRL_ERROR—Read/write access error status.
- HW_OCOTP_CTRL_RD_BANK_OPEN—Status of OTP read availability (reads).

7.2.1 Software Read Sequence

Reading OTP contents is relatively simple, because all OTP words are memory-mapped on the APB space (see Section 7.4, “Programmable Registers,” for details). These registers are read-only, except for the HW/SW capability shadow registers, which are writable until the appropriate LOCK bit in OTP is set.

Due to the fuse-read architecture, the OTP banks must be open before they can be read. This is accomplished as follows (the following does not apply to shadow registers, which can be read at any time).

1. Program the HCLK to a frequency up to the maximum allowable HCLK frequency. Note that this cannot exceed 200 MHz.
2. Check that HW_OCOTP_CTRL_BUSY and HW_OCOTP_CTRL_ERROR are clear.
3. Set HW_OCOTP_CTRL_RD_BANK_OPEN. This will kick the controller to put the fuses into read mode. The controller will set HW_OCOTP_CTRL_BUSY until the OTP contents are readable. Note that if there was a pending write (holding HW_OCOTP_CTRL_BUSY) and HW_OCOTP_CTRL_RD_BANK_OPEN was set, the controller would complete the write and immediately move into read operation (keeping HW_OCOTP_CTRL_BUSY set while the banks are being opened).
4. Poll for HW_OCOTP_CTRL_BUSY clear. When HW_OCOTP_CTRL_BUSY is clear and HW_OCOTP_CTRL_RD_BANK_OPEN is set, read the data from the appropriate memory-mapped address. Note that this is not necessary for registers that are shadowed. Reading before HW_OCOTP_CTRL_BUSY is cleared by the controller, will return 0xBADA_BADA and will result in the setting of HW_OCOTP_CTRL_ERROR. Because opening banks takes approximately 33 HCLK cycles, immediate polling for BUSY is not recommended.
5. Once accesses are complete, clear HW_OCOTP_CTRL_RD_BANK_OPEN. Leaving the banks open will cause current drain.

If data is accessed from a protected region (such as the crypto key, once a read LOCK bit has been set), the controller returns 0xBADA_BADA. In addition HW_OCOTP_CTRL_ERROR is set. It must be cleared by software before any new write access can be issued. Subsequent reads to unrestricted mapped OTP locations will still work successfully assuming that HW_OCOTP_CTRL_RD_BANK_OPEN is set and HW_OCOTP_CTRL_BUSY is clear.

It should be noted that after opening the banks, read latencies to OTP are “instant” (meaning they behave like regular reads from hardware registers), since parallel loading is used.

It should also be noted that setting HW_OCOTP_CTRL_RELOAD_SHADOWS to reload shadow registers does not set HW_OCOTP_CTRL_RD_BANK_OPEN. HW_OCOTP_CTRL_RD_BANK_OPEN can only be set and cleared by software. Forced reloading of shadows is covered in Section 7.2.4, “Shadow Registers and Hardware Capability Bus.”

7.2.2 Software Write Sequence

In order to avoid erroneous code performing erroneous writes to OTP, a special unlocking sequence is required for writes.

1. Program HCLK to 24 MHz. OTP writes do not work at frequencies above 24 MHz.
2. Set the VDDIO voltage to 2.8 V (using HW_POWER_VDDIOCTRL_TRG). The VDDIO voltage is used to program OTP. Incorrect voltage and frequency settings will result in the OTP being programmed with incorrect values.
3. Check that HW_OCOTP_CTRL_BUSY and HW_OCOTP_CTRL_ERROR are clear. Overlapped accesses are not supported by the controller. Any pending write must be completed before a write access can be requested. In addition, the banks cannot be open for reading, so HW_OCOTP_CTRL_RD_BANK_OPEN must also be clear. If the write is done following a previous write, the postamble wait period of 2 µs must be followed after the clearing of HW_OCOTP_CTRL_BUSY (see Section 7.2.3, “Write Postamble,”).
4. Write the requested address to HW_OCOTP_CTRL_ADDR and program the unlock code into HW_OCOTP_CTRL_WR_UNLOCK. This must be programmed for each write access. The lock code is documented in the register description. Both the unlock code and address can be written in the same operation.
5. Write the data to HW_OCOTP_DATA. This automatically sets HW_OCOTP_CTRL_BUSY and clears HW_OCOTP_CTRL_WR_UNLOCK. In this case, the data is a programming mask. Bit fields with ones will result in that OTP bit being set. Only the controller can clear HW_OCOTP_CTRL_BUSY. The controller will use the mask to program a 32-bit word in the OTP per the address in ADDR. At the same time that the write is accepted, the controller makes an internal copy of HW_OCOTP_CTRL_ADDR that cannot be updated until the next write sequence is initiated. This copy guarantees that erroneous writes to HW_OCOTP_CTRL_ADDR will not affect an active write operation. It should also be noted that, during the programming, HW_OCOTP_DATA will shift right (with zero fill). This shifting is required to program the OTP serially. During the write operation, HW_OCOTP_DATA cannot be modified.
6. Once complete, the controller clears BUSY. Beyond this, the 2-µs postamble requirement must be met before submitting any further OTP operations (see Section 7.2.3, “Write Postamble,”). A write request to a protected region will result in no OTP access and no setting of HW_OCOTP_CTRL_BUSY. In addition, HW_OCOTP_CTRL_ERROR will be set. It must be cleared by software before any new write access can be issued.

It should be noted that write latencies to OTP are in the order of 10s to 100s of microseconds per word. Write latencies will vary based on the location of the word within the OTP bank. Once a write is initiated, HW_OCOTP_DATA is shifted one bit per every 32 HCLK cycles.

Given:

8 words per OTP bank
32 bits per word
tHCLK is the HCLK clock period
n word locations (where $0 \leq n \leq 7$)

Then, the approximate write latency for a given word is:

*tHCLK * 32 * 32 * n*

In addition to this latency, software must allow for the 2-µs postamble (using HW_DIGCTL_MICROSECONDS), as described in Section 7.2.3, “Write Postamble,”

7.2.3 Write Postamble

Due to internal electrical characteristics of the OTP during writes, all OTP operations following a write must be separated by 2 µs after the clearing of HW_OCOTP_CTRL_BUSY following the write. This guarantees programming voltages on-chip to reach a steady state when exiting a write sequence. This includes reads, shadow reloads, or other writes. A recommended software sequence to meet the postamble requirements is as follows:

1. Issue the write and poll for BUSY (as per Section 7.2.2, “Software Write Sequence,”).
2. Once BUSY is clear, use HW_DIGCTL_MICROSECONDS to wait 2 µs.
3. Perform the next OTP operation.

7.2.4 Shadow Registers and Hardware Capability Bus

The on-chip customer hardware capability bus is generated using a direct connection to the HW_OCOTP_CUSTCAP shadow register. The bits are copied from the OTP on reset. They can be modified until HW_OCOTP_LOCK_CUSTCAP_SHADOW is set.

The user can force a reload of the shadow registers (including HW_OCOTP_LOCK) without having to reset the device, which is useful for debugging code. To force a reload:

- Set HW_OCOTP_CTRL_RELOAD_SHADOWS.
- Wait for HW_OCOTP_CTRL_BUSY and HW_OCOTP_CTRL_RELOAD_SHADOWS to be cleared by the controller.
- Attempting to write to the shadow registers while the shadows are being reloaded will result in the setting of HW_OCOTP_CTRL_ERROR. In addition, the register will not take the attempted write (yielding to the reload instead).
- Attempting to write to a shadow register that is locked will result in the setting of HW_OCOTP_CTRL_ERROR.

HW_OCOTP_CTRL_RELOAD_SHADOWS can be set at any time. There is no need to wait for HW_OCOTP_CTRL_BUSY or HW_OCOTP_CTRL_ERROR to be clear.

- In the case of HW_OCOTP_CTRL_BUSY being set due to an active write, the controller will perform the bank opening and shadow reloading immediately after the completion of the write.
- In the case where HW_OCOTP_CTRL_RD_BANK_OPEN is set, the shadow reload will be performed immediately after the banks are closed by software (by clearing HW_OCOTP_CTRL_RD_BANK_OPEN). It should be noted that BUSY will take approximately 33 HCLK cycles to clear, so polling for HW_OCOTP_CTRL_BUSY immediately after clearing HW_OCOTP_CTRL_RD_BANK_OPEN is not recommended.

- In all cases, the controller will clear HW_OCOTP_CTRL_RELOAD_SHADOWS after the successful completion of the operation.

7.3 Behavior During Reset

The OCOTP is always active. The shadow registers described in Section 7.4, “Programmable Registers,” automatically load the appropriate OTP contents after reset is deasserted. During this load-time HW_OCOTP_CTRL_BUSY is set. The load time is approximately 32 HCLK cycles after the deassertion of reset. These shadow registers can be reloaded as described in Section 7.2.4, “Shadow Registers and Hardware Capability Bus.”

7.4 Programmable Registers

The following registers are available for programmer access and control of the OCOTP.

7.4.1 OTP Controller Control Register Description

The OCOTP Control and Status Register specifies the copy state, as well as the control required for random access of the OTP memory

HW_OCOTP_CTRL	0x000
HW_OCOTP_CTRL_SET	0x004
HW_OCOTP_CTRL_CLR	0x008
HW_OCOTP_CTRL_TOG	0x00C

Table 7-1. HW_OCOTP_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
WR_UNLOCK																RSRVD2		RELOAD_SHADOWS	RD_BANK_OPEN	RSRVD1		ERROR	BUSY	RSRVD0				ADDR			

Table 7-2. HW_OCOTP_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**WR_UNLOCK**	RW	0x0	Write 0x3E77 to enable OTP write accesses. NOTE: This register must be unlocked on a write-by-write basis (a write is initiated when HW_OCOTP_DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to HW_OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY). KEY = 0x3E77 Key needed to unlock HW_OCOTP_DATA register.
15:14	**RSRVD2**	RO	0x0	These bits always read back zero.
13	**RELOAD_SHADOWS**	RW	0x0	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically open banks (but will not set RD_BANK_OPEN) and set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller. There is no need to set RD_BANK_OPEN to force the reload. If RD_BANK_OPEN is already set, its still possible to set RELOAD_SHADOWS. In this case, the shadow registers will only be updated upon the clearing of RD_BANK_OPEN.
12	**RD_BANK_OPEN**	RW	0x0	Set to open the all the OTP banks for reading. When set, the controller sets BUSY to allow time for the banks to become available (approximatly 32 HCLK cycles later at which time the controller will clear BUSY). Once BUSY is clear, the various OTP words are accessible via their memory mapped address. Note that OTP words which are shadowed, can be read at anytime and will not be affected by RD_BANK_OPEN. This bit must be cleared after reading is complete. Keeping the OTP banks open causes additional current draw. BUSY must be clear before this setting will take affect. If there is a write transaction pending (holding BUSY), then the bank opening sequence will begin automatically upon the previous transaction clears BUSY. Note that if a read is performed from non-shadowed locations without RD_BANK_OPEN, ERROR will be set
11:10	**RSRVD1**	RO	0x0	These bits always read back zero.
9	**ERROR**	RW	0x0	Set by the controller when either an access to a locked region is requested or a read is requested from non-shadowed efuse locations without the banks being open. Must be cleared before any further write access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.

Table 7-2. HW_OCOTP_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	**BUSY**	RO	0x0	OTP controller status bit. When active, no new write access or bank open operations (including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete (for writes), or the banks are open (for reads). After reset (or after setting RELOAD_SHADOWS), this bit is set by the controller until the HW/SW and LOCK registers are successfully copied, after which time it is automatically cleared by the controller.
7:5	**RSRVD0**	RO	0x0	These bits always read back zero.
4:0	**ADDR**	RW	0x0	OTP write access address register. Specifies one of 32 word address locations (0x00 - 0x1F). If a valid write is accepted by the controller (see HW_OCOTP_DATA for details on what constitutes a valid write), the controller makes an internal copy of this value (to avoid the OTP programming being corrupted). This internal copy will not update until the write access is complete.

DESCRIPTION:

The OCOTP Control and Status Register provides the necessary software interface for performing read and write operations to the On-Chip OTP (One-Time Programmable ROM). The control fields such as WR_UNLOCK, ADDR and BUSY/ERROR may be used in conjuction with the HW_OCOTP_DATA register to perform write operations. Read operations are performed via the direct memory mapped registers. In the cases where OTP values are shadowed into local memory storage, the memory mapped location can be read directly. In the cases where the OTP values are not shadowed into local memory, the read-preparation sequence involving RD_BANK_OPEN and BUSY/ERROR fields must be used before performing the read.

EXAMPLE:

```
Empty Example.
```

7.4.2 OTP Controller Write Data Register Description

The OCOTP Data Register is used for OTP Programming

HW_OCOTP_DATA 0x010

Table 7-3. HW_OCOTP_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATA																															

Table 7-4. HW_OCOTP_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATA**	RW	0x0	Used to initiate a write to OTP. Please see the "Software Write Sequence" section for operating details.

DESCRIPTION:

This register is used in conjuction with HW_OCOTP_CTRL to perform one-time writes to the OTP. Please see the "Software Write Sequence" section for operating details.

EXAMPLE:

```
Empty Example.
```

7.4.3 Value of OTP Bank0 Word0 (Customer) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_CUST0 0x020

Table 7-5. HW_OCOTP_CUST0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-6. HW_OCOTP_CUST0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 0, word 0 (ADDR = 0x00)

DESCRIPTION:

Non-shadowed memory mapped access to OTP Bank 0, word 0 (ADDR = 0x00).

EXAMPLE:

```
Empty Example.
```

7.4.4 Value of OTP Bank0 Word1 (Customer) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_CUST1 0x030

Table 7-7. HW_OCOTP_CUST1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-8. HW_OCOTP_CUST1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 0, word 1 (ADDR = 0x01)

DESCRIPTION:

Non-shadowed memory mapped access to OTP Bank 0, word 1 (ADDR = 0x01).

EXAMPLE:

```
Empty Example.
```

7.4.5 Value of OTP Bank0 Word2 (Customer) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_CUST2 0x040

Table 7-9. HW_OCOTP_CUST2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-10. HW_OCOTP_CUST2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 0, word 2 (ADDR = 0x02)

DESCRIPTION:

Non-shadowed memory mapped access to OTP Bank 0, word 2 (ADDR = 0x02).

EXAMPLE:

```
Empty Example.
```

7.4.6 Value of OTP Bank0 Word3 (Customer) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_CUST3 0x050

Table 7-11. HW_OCOTP_CUST3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-12. HW_OCOTP_CUST3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 0, word 3 (ADDR = 0x03)

DESCRIPTION:

Non-shadowed memory mapped access to OTP Bank 0, word 3 (ADDR = 0x03).

EXAMPLE:

```
Empty Example.
```

7.4.7 Value of OTP Bank0 Word4 (Crypto Key) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_CRYPTO0 0x060

Table 7-13. HW_OCOTP_CRYPTO0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-14. HW_OCOTP_CRYPTO0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 0, word 4 (ADDR = 0x04). If LOCK[CRYPTOKEY] is set, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR]

DESCRIPTION:

Non-shadowed memory mapped access to OTP Bank 0, word 4 (ADDR = 0x04).

EXAMPLE:

```
Empty Example.
```

7.4.8 Value of OTP Bank0 Word5 (Crypto Key) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and

HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_CRYPTO1 0x070

Table 7-15. HW_OCOTP_CRYPTO1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-16. HW_OCOTP_CRYPTO1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 0, word 5 (ADDR = 0x05). If LOCK[CRYPTOKEY] is set, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR]

DESCRIPTION:

Non-shadowed memory mapped access to OTP Bank 0, word 5 (ADDR = 0x05).

EXAMPLE:

```
Empty Example.
```

7.4.9 Value of OTP Bank0 Word6 (Crypto Key) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_CRYPTO2 0x080

Table 7-17. HW_OCOTP_CRYPTO2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-18. HW_OCOTP_CRYPTO2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 0, word 6 (ADDR = 0x06). If LOCK[CRYPTOKEY] is set, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR]

DESCRIPTION:

Non-shadowed memory mapped access to OTP Bank 0, word 6 (ADDR = 0x06).

EXAMPLE:

```
Empty Example.
```

7.4.10 Value of OTP Bank0 Word7 (Crypto Key) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_CRYPTO3 0x090

Table 7-19. HW_OCOTP_CRYPTO3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-20. HW_OCOTP_CRYPTO3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 0, word 7 (ADDR = 0x07). If LOCK[CRYPTOKEY] is set, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR]

DESCRIPTION:

Non-shadowed memory mapped access to OTP Bank 0, word 7 (ADDR = 0x07).

EXAMPLE:

```
Empty Example.
```

7.4.11 HW Capability Shadow Register 0 Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

HW_OCOTP_HWCAP0 0x0A0

Table 7-21. HW_OCOTP_HWCAP0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-22. HW_OCOTP_HWCAP0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Shadow register for HW capability bits 31:0 (copy of OTP bank 1, word 0 (ADDR = 0x08)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

DESCRIPTION:

Shadowed memory mapped access to OTP bank 1, word 0 (ADDR = 0x08).

EXAMPLE:

```
Empty Example.
```

7.4.12 HW Capability Shadow Register 1 Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

HW_OCOTP_HWCAP1 0x0B0

Table 7-23. HW_OCOTP_HWCAP1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-24. HW_OCOTP_HWCAP1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Shadow register for HW capability bits 63:32 (copy of OTP bank 1, word 1 (ADDR = 0x09)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

DESCRIPTION:

Shadowed memory mapped access to OTP bank 1, word 1 (ADDR = 0x09).

EXAMPLE:

```
Empty Example.
```

7.4.13 HW Capability Shadow Register 2 Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

HW_OCOTP_HWCAP2 0x0C0

Table 7-25. HW_OCOTP_HWCAP2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-26. HW_OCOTP_HWCAP2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Shadow register for HW capability bits 95:64 (copy of OTP bank 1, word 2 (ADDR = 0x0A)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

DESCRIPTION:

Shadowed memory mapped access to OTP bank 1, word 2 (ADDR = 0x0A).

EXAMPLE:

```
Empty Example.
```

7.4.14 HW Capability Shadow Register 3 Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

HW_OCOTP_HWCAP3 0x0D0

Table 7-27. HW_OCOTP_HWCAP3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-28. HW_OCOTP_HWCAP3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Shadow register for HW capability bits 127:96 (copy of OTP bank 1, word 3 (ADDR = 0x0B)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

DESCRIPTION:

Shadowed memory mapped access to OTP bank 1, word 3 (ADDR = 0x0B).

EXAMPLE:

```
Empty Example.
```

7.4.15 HW Capability Shadow Register 4 Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

HW_OCOTP_HWCAP4 0x0E0

Table 7-29. HW_OCOTP_HWCAP4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-30. HW_OCOTP_HWCAP4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Shadow register for HW capability bits 159:128 (copy of OTP bank 1, word 4 (ADDR = 0x0C)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

DESCRIPTION:

Shadowed memory mapped access to OTP bank 1, word 4 (ADDR = 0x0C).

EXAMPLE:

```
Empty Example.
```

7.4.16 HW Capability Shadow Register 5 Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

HW_OCOTP_HWCAP5 0x0F0

Table 7-31. HW_OCOTP_HWCAP5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-32. HW_OCOTP_HWCAP5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Shadow register for HW capability bits 191:160 (copy of OTP bank 1, word 5 (ADDR = 0x0D)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

DESCRIPTION:

Shadowed memory mapped access to OTP bank 1, word 5 (ADDR = 0x0D).

EXAMPLE:

```
Empty Example.
```

7.4.17 SW Capability Shadow Register Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

HW_OCOTP_SWCAP 0x100

Table 7-33. HW_OCOTP_SWCAP

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-34. HW_OCOTP_SWCAP Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Shadow register for SW capability bits 31:0 (copy of OTP bank 1, word 6 (ADDR = 0x0E)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

DESCRIPTION:

Shadowed memory mapped access to OTP bank 1, word 6 (ADDR = 0x0E).

EXAMPLE:

```
Empty Example.
```

7.4.18 Customer Capability Shadow Register Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

HW_OCOTP_CUSTCAP 0x110

Table 7-35. HW_OCOTP_CUSTCAP

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CUST_DISABLE_WMADRM9	CUST_DISABLE_JANUSDRM10	RSRVD1																									ENABLE_SJTAG_12MA_DRIVE	USE_PARALLEL_JTAG	RTC_XTAL_32768_PRESENT	RTC_XTAL_32000_PRESENT	RSRVD0	

Table 7-36. HW_OCOTP_CUSTCAP Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CUST_DISABLE_WMADRM9**	RW	0x0	Blow to disable WMA DRM9.
30	**CUST_DISABLE_JANUSDRM10**	RW	0x0	Blow to disable WMA Janus DRM10.
29:5	**RSRVD1**	RW	0x0	Reserved - do not blow these bits.
4	**ENABLE_SJTAG_12MA_DRIVE**	RW	0x0	Blow to force the 1-wire DEBUG (serial JTAG) pin to drive 12mA, the default is 8mA (see ENABLE_PJTAG_12MA_DRIVE in the ROM0 register for 6-wire parallel JTAG). This is a hardware override which will cause the drive select bits in the PINCTRL block to reset to the 12mA drive settings rather than the normal default of 8mA. The user is still free to reprogram these bits to other drive levels.
3	**USE_PARALLEL_JTAG**	RW	0x0	During JTAG boot mode, the ROM reads this bit, then inverts it, and writes the value to the HW_DIGCTL_CTRL_USE_SERIAL_JTAG bit. If this bit is one, indicating parallel JTAG mode is selected, a zero is written to the DIGCTL USE_SERIAL_JTAG bit which places the device into 6-wire JTAG mode, and if this bit is zero, a one instead is written causing the SJTAG block to switch to the 1-wire serial JTAG mode.
2	**RTC_XTAL_32768_PRESENT**	RW	0x0	Blow to indicate the presence of an optional 32.768KHz crystal off-chip.
1	**RTC_XTAL_32000_PRESENT**	RW	0x0	Blow to indicate the presence of an optional 32.000KHz crystal off-chip.
0	**RSRVD0**	RW	0x0	Reserved - do not blow these bits.

DESCRIPTION:

Shadowed memory mapped access to OTP bank 1, word 7 (ADDR = 0x0F).

EXAMPLE:

```
Empty Example.
```

7.4.19 LOCK Shadow Register OTP Bank 2 Word 0 Description

Shadow register for OCOTP Lock Status Value (ADDR = 0x10). Copy of the state of the OTP lock regions. Copied from the OTP upon reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

HW_OCOTP_LOCK 0x120

Table 7-37. HW_OCOTP_LOCK

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ROM7	ROM6	ROM5	ROM4	ROM3	ROM2	ROM1	ROM0	HWSW_SHADOW_ALT	CRYPTODCP_ALT	CRYPTOKEY_ALT	PIN	OPS	UN2	UN1	UN0	UNALLOCATED					ROM_SHADOW	CUSTCAP	HWSW	CUSTCAP_SHADOW	HWSW_SHADOW	CRYPTODCP	CRYPTOKEY	CUST3	CUST2	CUST1	CUST0

Table 7-38. HW_OCOTP_LOCK Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**ROM7**	RO	0x0	Status of ROM use region write lock bits (ADDR = 0x1F). When set, word 0x1F in the ROM region is locked.
30	**ROM6**	RO	0x0	Status of ROM use region write lock bits (ADDR = 0x1E). When set, word 0x1E in the ROM region is locked.
29	**ROM5**	RO	0x0	Status of ROM use region write lock bits (ADDR = 0x1D). When set, word 0x1D in the ROM region is locked.
28	**ROM4**	RO	0x0	Status of ROM use region write lock bits (ADDR = 0x1C). When set, word 0x1C in the ROM region is locked.
27	**ROM3**	RO	0x0	Status of ROM use region write lock bits (ADDR = 0x1B). When set, word 0x1B in the ROM region is locked.
26	**ROM2**	RO	0x0	Status of ROM use region write lock bits (ADDR = 0x1A). When set, word 0x1A in the ROM region is locked.
25	**ROM1**	RO	0x0	Status of ROM use region write lock bits (ADDR = 0x19). When set, word 0x19 in the ROM region is locked.
24	**ROM0**	RO	0x0	Status of ROM use region write lock bits (ADDR = 0x18). When set, word 0x18 in the ROM region is locked.
23	**HWSW_SHADOW_ALT**	RO	0x0	Status of alternate bit for HWSW_SHADOW lock
22	**CRYPTODCP_ALT**	RO	0x0	Status of alternate bit for CRYPTODCP lock
21	**CRYPTOKEY_ALT**	RO	0x0	Status of alternate bit for CRYPTOKEY lock
20	**PIN**	RO	0x0	Status of Pin access lock bit. When set, pin access is disabled.
19	**OPS**	RO	0x0	Status of SGTL-OPS region (ADDR = 0x11-0x14) write lock bit. When set, region is locked.
18	**UN2**	RO	0x0	Status of un-assigned (ADDR = 0x17) write-lock bit. When set, un-asigned word at OTP address 0x17 is locked.
17	**UN1**	RO	0x0	Status of un-assigned (ADDR = 0x16) write-lock bit. When set, un-asigned word at OTP address 0x16 is locked.

Table 7-38. HW_OCOTP_LOCK Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
16	**UN0**	RO	0x0	Status of un-assigned (ADDR = 0x15) write-lock bit. When set, un-asigned word at OTP address 0x15 is locked.
15:11	**UNALLOCATED**	RO	0x0	Value of un-used portion of LOCK word
10	**ROM_SHADOW**	RO	0x0	Status of ROM region shadow register lock. When set, over-ride of ROM-region shadow bits is blocked.
9	**CUSTCAP**	RO	0x0	Status of Customer Capability region (ADDR = 0x0F) write lock bit. When set, region is locked.
8	**HWSW**	RO	0x0	Status of HW/SW region (ADDR = 0x08-0x0E) write lock bit. When set, region is locked.
7	**CUSTCAP_SHADOW**	RO	0x0	Status of Customer Capability shadow register lock. When set, over-ride of customer capabality shadow bits is blocked.
6	**HWSW_SHADOW**	RO	0x0	Status of HW/SW Capability shadow register lock. When set, over-ride of HW/SW capabality shadow bits is blocked.
5	**CRYPTODCP**	RO	0x0	Status of read lock bit for DCP APB crypto access. When set, the DCP will disallow reads of its crypto keys via its APB interface.
4	**CRYPTOKEY**	RO	0x0	Status of crypto key region (ADDR = 0x04-0x07) read/write lock bit. When set, region is locked.
3	**CUST3**	RO	0x0	Status of customer region word (ADDR = 0x03) write lock bit. When set, the region is locked.
2	**CUST2**	RO	0x0	Status of customer region word (ADDR = 0x02) write lock bit. When set, the region is locked.
1	**CUST1**	RO	0x0	Status of customer region word (ADDR = 0x01) write lock bit. When set, the region is locked.
0	**CUST0**	RO	0x0	Status of customer region word (ADDR = 0x00) write lock bit. When set, the region is locked.

DESCRIPTION:

Shadowed memory mapped access to OTP bank 2, word 0 (ADDR = 0x10).

EXAMPLE:

```
Empty Example.
```

7.4.20 Value of OTP Bank2 Word1 (Freescale OPS0) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_OPS0 0x130

Table 7-39. HW_OCOTP_OPS0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-40. HW_OCOTP_OPS0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 2, word 1 (ADDR = 0x11)

DESCRIPTION:

Shadowed memory mapped access to OTP Bank 2, word 1 (ADDR = 0x11).

EXAMPLE:

```
Empty Example.
```

7.4.21 Value of OTP Bank2 Word2 (Freescale OPS1) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_OPS1 0x140

Table 7-41. HW_OCOTP_OPS1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-42. HW_OCOTP_OPS1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 2, word 2 (ADDR = 0x12)

DESCRIPTION:

Shadowed memory mapped access to OTP Bank 2, word 2 (ADDR = 0x12).

EXAMPLE:

```
Empty Example.
```

7.4.22 Value of OTP Bank2 Word3 (Freescale OPS2) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_OPS2 0x150

Table 7-43. HW_OCOTP_OPS2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-44. HW_OCOTP_OPS2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 2, word 3 (ADDR = 0x13)

DESCRIPTION:

Shadowed memory mapped access to OTP Bank 2, word 3 (ADDR = 0x13).

EXAMPLE:

```
Empty Example.
```

7.4.23 Value of OTP Bank2 Word4 (Freescale OPS3) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_OPS3 0x160

Table 7-45. HW_OCOTP_OPS3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-46. HW_OCOTP_OPS3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 2, word 4 (ADDR = 0x14)

DESCRIPTION:

Shadowed memory mapped access to OTP Bank 2, word 4 (ADDR = 0x14).

EXAMPLE:

```
Empty Example.
```

7.4.24 Value of OTP Bank2 Word5 (Unassigned0) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_UN0 0x170

Table 7-47. HW_OCOTP_UN0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-48. HW_OCOTP_UN0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 2, word 5 (ADDR = 0x15)

DESCRIPTION:

Non-shadowed memory mapped access to OTP Bank 2, word 5 (ADDR = 0x15).

EXAMPLE:

```
Empty Example.
```

7.4.25 Value of OTP Bank2 Word6 (Unassigned1) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_UN1 0x180

Table 7-49. HW_OCOTP_UN1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-50. HW_OCOTP_UN1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 2, word 6 (ADDR = 0x16)

DESCRIPTION:

Non-shadowed memory mapped access to OTP Bank 2, word 6 (ADDR = 0x16).

EXAMPLE:

```
Empty Example.
```

7.4.26 Value of OTP Bank2 Word7 (Unassigned2) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and

HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_UN2 0x190

Table 7-51. HW_OCOTP_UN2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-52. HW_OCOTP_UN2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Reflects value of OTP Bank 2, word 7 (ADDR = 0x17)

DESCRIPTION:

Non-shadowed memory mapped access to OTP Bank 2, word 7 (ADDR = 0x17).

EXAMPLE:

```
Empty Example.
```

7.4.27 Shadow Register for OTP Bank3 Word0 (ROM Use 0) Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

HW_OCOTP_ROM0 0x1A0

Table 7-53. HW_OCOTP_ROM0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BOOT_MODE								ENABLE_PJTAG_12MA_DRIVE	USE_PARALLEL_JTAG	SD_POWER_GATE_GPIO		SD_POWER_UP_DELAY				SD_BUS_WIDTH				SSP_SCK_INDEX				RSRVD3	DISABLE_SPI_NOR_FAST_READ	ENABLE_USB_BOOT_SERIAL_NUM	ENABLE_UNENCRYPTED_BOOT	SD_MBR_BOOT	RSRVD2	RSRVD1	RSRVD0

Table 7-54. HW_OCOTP_ROM0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**BOOT_MODE**	RW	0x0	Encoded boot mode.
23	**ENABLE_PJTAG_12MA_DRIVE**	RW	0x0	Blow to force the 6-wire PJTAG pins to drive 12mA, default is 4mA (note that SJTAG is fixed at 8mA). Blowing this bit causes the ROM to program all six parallel JTAG pins to drive 12mA via the pin control registers.
22	**USE_PARALLEL_JTAG**	RW	0x0	During JTAG boot mode, the ROM reads this bit, then inverts it, and writes the value to the HW_DIGCTL_CTRL_USE_SERIAL_JTAG bit. If this bit is one, indicating parallel JTAG mode is selected, a zero is written to the DIGCTL USE_SERIAL_JTAG bit which places the device into 6-wire JTAG mode, and if this bit is zero, a one instead is written causing the SJTAG block to switch to the 1-wire serial JTAG mode.
21:20	**SD_POWER_GATE_GPIO**	RW	0x0	SD card power gate GPIO pin select: 00 - PWM0, 01 - LCD_DOTCLK, 10 - PWM3, 11 - NO_GATE.
19:14	**SD_POWER_UP_DELAY**	RW	0x0	SD card power up delay required after enabling GPIO power gate: 000000 - 0 ms, 000001 - 10 ms, 000010 - 20 ms, 111111 - 630 ms.
13:12	**SD_BUS_WIDTH**	RW	0x0	SD card bus width: 00 - 4-bit, 01 - 1-bit, 10 - 8-bit, 11 - reserved.
11:8	**SSP_SCK_INDEX**	RW	0x0	Index to the SSP clock speed
7	**RSRVD3**	RW	0x0	Reserved - do not blow this bit.
6	**DISABLE_SPI_NOR_FAST_READ**	RW	0x0	Blow to disable SPI NOR fast reads which are used by default.
5	**ENABLE_USB_BOOT_SERIAL_NUM**	RW	0x0	Blow to enable USB boot serial number.
4	**ENABLE_UNENCRYPTED_BOOT**	RW	0x0	Blow to enable unencrypted boot modes.
3	**SD_MBR_BOOT**	RW	0x0	Blow to enable master boot record (MBR) boot mode for SD boot.
2	**RSRVD2**	RW	0x0	Reserved - do not blow this bit.
1	**RSRVD1**	RW	0x0	Reserved - do not blow this bit.
0	**RSRVD0**	RW	0x0	Reserved - do not blow this bit.

DESCRIPTION:

Shadowed memory mapped access to OTP Bank 3, word 0 (ADDR = 0x18).

EXAMPLE:

```
Empty Example.
```

7.4.28 Shadow Register for OTP Bank3 Word1 (ROM Use 1) Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

HW_OCOTP_ROM1 0x1B0

Table 7-55. HW_OCOTP_ROM1

31:30	29:28	27:26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11:10	09:08	07:04	03	02:01	00
RSRVD1	USE_ALT_GPMI_RDY3	USE_ALT_GPMI_CE3	USE_ALT_GPMI_RDY2	USE_ALT_GPMI_CE2	ENABLE_NAND3_CE_RDY_PULLUP	ENABLE_NAND2_CE_RDY_PULLUP	ENABLE_NAND1_CE_RDY_PULLUP	ENABLE_NAND0_CE_RDY_PULLUP	UNTOUCH_INTERNAL_SSP_PULLUP	SSP2_EXT_PULLUP	SSP1_EXT_PULLUP	SD_INCREASE_INIT_SEQ_TIME	SD_INIT_SEQ_2_ENABLE	SD_CMD0_DISABLE	SD_INIT_SEQ_1_DISABLE	USE_ALT_SSP1_DATA4-7	BOOT_SEARCH_COUNT	RSRVD0	NUMBER_OF_NANDS			

Table 7-56. HW_OCOTP_ROM1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	**RSRVD1**	RW	0x0	Reserved - do not blow this bit.
29:28	**USE_ALT_GPMI_RDY3**	RW	0x0	These bits are used by ROM NAND driver to enable one of 3 alternate pins for GPMI_RDY3. 00-GPMI_RDY3, 01-PWM2 and 10-LCD_DOTCK.
27:26	**USE_ALT_GPMI_CE3**	RW	0x0	These bits are used by ROM NAND driver to enable one of 4 alternate pins for GPMI_CE3. 00-GPMI_D15, 01-LCD_RESET, 10-SSP_DETECT and 11-ROTARYB.
25	**USE_ALT_GPMI_RDY2**	RW	0x0	If the bit is blown then ROM NAND driver will enable alternate pins for GPMI_RDY2.
24	**USE_ALT_GPMI_CE2**	RW	0x0	If the bit is blown then ROM NAND driver will enable alternate pins for GPMI_CE2.
23	**ENABLE_NAND3_CE_RDY_PULLUP**	RW	0x0	If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE3 and GPMI_RDY3 pins.
22	**ENABLE_NAND2_CE_RDY_PULLUP**	RW	0x0	If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE2 and GPMI_RDY2 pins.
21	**ENABLE_NAND1_CE_RDY_PULLUP**	RW	0x0	If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE1 and GPMI_RDY1 pins.
20	**ENABLE_NAND0_CE_RDY_PULLUP**	RW	0x0	If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE0 and GPMI_RDY0 pins.
19	**UNTOUCH_INTERNAL_SSP_PULLUP**	RW	0x0	If this bit is blown then internal pull-ups for SSP are neither enabled nor disabled. This bit is used only if external pull-ups are implemented and ROM1:18 and/or ROM1:17 are blown.
18	**SSP2_EXT_PULLUP**	RW	0x0	Blow to indicate external pull-ups implemented for SSP2.
17	**SSP1_EXT_PULLUP**	RW	0x0	Blow to indicate external pull-ups implemented for SSP1.
16	**SD_INCREASE_INIT_SEQ_TIME**	RW	0x0	Blow to increase the SD card initialization sequence time from 1ms (default) to 4ms.
15	**SD_INIT_SEQ_2_ENABLE**	RW	0x0	Blow to enable the second initialization sequence for SD boot.

Table 7-56. HW_OCOTP_ROM1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
14	**SD_CMD0_DISABLE**	RW	0x0	Cmd0 (reset cmd) is called by default to reset the SD card during startup. Blow this bit to not reset the card during SD boot.
13	**SD_INIT_SEQ_1_DISABLE**	RW	0x0	Blow to disable the first initialization sequence for SD.
12	**USE_ALT_SSP1_DATA4-7**	RW	0x0	This bit is blown to enable alternate pin use for SSP1 data lines 4-7.
11:8	**BOOT_SEARCH_COUNT**	RW	0x0	Number of 64 page blocks that should be read by the boot loader.
7:3	**RSRVD0**	RW	0x0	Reserved - do not blow these bits.
2:0	**NUMBER_OF_NANDS**	RW	0x0	Encoded value indicates number of external NAND devices (0 to 7). Zero indicates ROM will probe for the number of NAND devices connected in the system.

DESCRIPTION:

Shadowed memory mapped access to OTP Bank 3, word 1 (ADDR = 0x19).

EXAMPLE:

```
Empty Example.
```

7.4.29 Shadow Register for OTP Bank3 Word2 (ROM Use 2) Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

HW_OCOTP_ROM2 0x1C0

Table 7-57. HW_OCOTP_ROM2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
USB_VID																USB_PID															

Table 7-58. HW_OCOTP_ROM2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**USB_VID**	RW	0x0	USB Vendor ID.
15:0	**USB_PID**	RW	0x0	USB Product ID

DESCRIPTION:

Shadowed memory mapped access to OTP Bank 3, word 2 (ADDR = 0x1A).

EXAMPLE:

```
Empty Example.
```

7.4.30 Shadow Register for OTP Bank3 Word3 (ROM Use 3) Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

HW_OCOTP_ROM3 0x1D0

Table 7-59. HW_OCOTP_ROM3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																						RSRVD0									

Table 7-60. HW_OCOTP_ROM3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:10	**RSRVD1**	RW	0x0	Reserved - do not blow these bits.
9:0	**RSRVD0**	RW	0x0	SDK reserved bits.

DESCRIPTION:

Shadowed memory mapped access to OTP Bank 3, word 3 (ADDR = 0x1B).

EXAMPLE:

```
Empty Example.
```

7.4.31 Shadow Register for OTP Bank3 Word4 (ROM Use 4) Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

HW_OCOTP_ROM4 0x1E0

Table 7-61. HW_OCOTP_ROM4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-62. HW_OCOTP_ROM4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Shadow register for ROM-use word4 (Copy of OTP Bank 3, word 4 (ADDR = 0x1C)). These bits become read-only after the HW_OCOTP_LOCK[ROM_SHADOW] bit is set.

DESCRIPTION:

Shadowed memory mapped access to OTP Bank 3, word 4 (ADDR = 0x1C).

EXAMPLE:

```
Empty Example.
```

7.4.32 Shadow Register for OTP Bank3 Word5 (ROM Use 5) Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

HW_OCOTP_ROM5 0x1F0

Table 7-63. HW_OCOTP_ROM5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-64. HW_OCOTP_ROM5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Shadow register for ROM-use word5 (Copy of OTP Bank 3, word 5 (ADDR = 0x1D)). These bits become read-only after the HW_OCOTP_LOCK[ROM_SHADOW] bit is set.

DESCRIPTION:

Shadowed memory mapped access to OTP Bank 3, word 5 (ADDR = 0x1D).

EXAMPLE:

```
Empty Example.
```

7.4.33 Shadow Register for OTP Bank3 Word6 (ROM Use 6) Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

HW_OCOTP_ROM6 0x200

Table 7-65. HW_OCOTP_ROM6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 7-66. HW_OCOTP_ROM6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Shadow register for ROM-use word6 (Copy of OTP Bank 3, word 6 (ADDR = 0x1E)). These bits become read-only after the HW_OCOTP_LOCK[ROM_SHADOW] bit is set.

DESCRIPTION:

Shadowed memory mapped access to OTP Bank 3, word 6 (ADDR = 0x1E).

EXAMPLE:

```
Empty Example.
```

7.4.34 Shadow Register for OTP Bank3 Word7 (ROM Use 7) Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

HW_OCOTP_ROM7 0x210

Table 7-67. HW_OCOTP_ROM7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2																							ENABLE_SSP_12MA_DRIVE	RSRVD1				I2C_USE_400KHZ	ENABLE_ARM_ICACHE	RSRVD0	ENABLE_PIN_BOOT_CHECK

Table 7-68. HW_OCOTP_ROM7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:9	**RSRVD2**	RW	0x0	Reserved - do not blow these bits.
8	**ENABLE_SSP_12MA_DRIVE**	RW	0x0	Blow to force SSP pins to drive 12mA (default is 4mA)
7:4	**RSRVD1**	RW	0x0	Reserved - do not blow these bits.
3	**I2C_USE_400KHZ**	RW	0x0	Blow to force the I2C to be programmed by the boot loader to run at 400KHz (100KHz is the default)
2	**ENABLE_ARM_ICACHE**	RW	0x0	Blow to enable the ARM 926 ICache during boot
1	**RSRVD0**	RW	0x0	Reserved - do not blow this bit.
0	**ENABLE_PIN_BOOT_CHECK**	RW	0x1	Blow to enable boot loader to first test the LCD_RS pin to determine if pin boot mode is enabled. If this bit is blown, and LCD_RS is pulled high, then boot mode is determined by the state of LCD_D[6:0] pins. If this bit is not blown, skip testing the LCD_RS pin and go directly to determining boot mode by reading the state of LCD_D[6:0].

DESCRIPTION:

Shadowed memory mapped access to OTP Bank 3, word 7 (ADDR = 0x1F).

EXAMPLE:

```
Empty Example.
```

7.4.35 OTP Controller Version Register Description

This register always returns a known read value for debug purposes it indicates the version of the block.

HW_OCOTP_VERSION 0x220

Table 7-69. HW_OCOTP_VERSION

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MAJOR								MINOR								STEP															

Table 7-70. HW_OCOTP_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**MAJOR**	RO	0x01	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	**MINOR**	RO	0x04	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	**STEP**	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

DESCRIPTION:

This register indicates the RTL version in use.

EXAMPLE:

```
Empty Example.
```

OCOTP Block v1.4, Revision 1.21

Chapter 8
USB High-Speed Host/Device Controller

This chapter describes the USB high-speed controller included on the i.MX23. It includes sections on the PIO, DMA, and UTMI interfaces, along with USB controller flowcharts. Programmable USB controller registers are described in Section 8.6, "Programmable Registers." Descriptions for additional programmable registers mentioned in this chapter can be found in Section 4.8, "Programmable Registers," Section 6.4, "Programmable Registers," Section 9.4, "Programmable Registers, and Section 32.11, "Programmable Registers."

8.1 Overview

The i.MX23 includes a Universal Serial Bus (USB) version 2.0 controller capable of operating as either a USB device or a USB host, as shown in Figure 8-1. The USB controller is used to download digital music data or program code into external memory and to upload voice recordings from memory to the PC. Program updates can also be loaded into the flash memory area using the USB interface.

The USB device controller included on the i.MX23 supports five bi-directional endpoints: one control (for the default pipe) and four general purpose endpoints, each capable of operating in either IN, OUT, or both directions simultaneously. A typical portable device application defines 1 bulk-in, 1 bulk-out, and 1 interrupt in pipe.

As a USB host controller, it can enumerate and control USB devices attached to it. Using the USB Host Capability features, the USB controller can negotiate with another USB Host Capable system to be either the host or the device in a peer connection.

The USB controller operates either in full-speed mode or high-speed mode.

Refer to the USB Implementer's Forum website www.usb.org for detailed specifications and information on the USB protocol, timing and electrical characteristics.

The USB 2.0 controller comprises both a programmed I/O (PIO) interface and a DMA interface. Both of these interfaces are designed to meet an ARM Ltd. AMBA Hardware Bus (AHB). The AHB is used by the USB controller as a slave (PIO register accesses) and as a master (DMA memory accesses).

The USB 2.0 PHY is fully integrated on-chip and is described in Chapter 9, "Integrated USB 2.0 PHY," The PHY is controlled over the APBX peripheral bus.

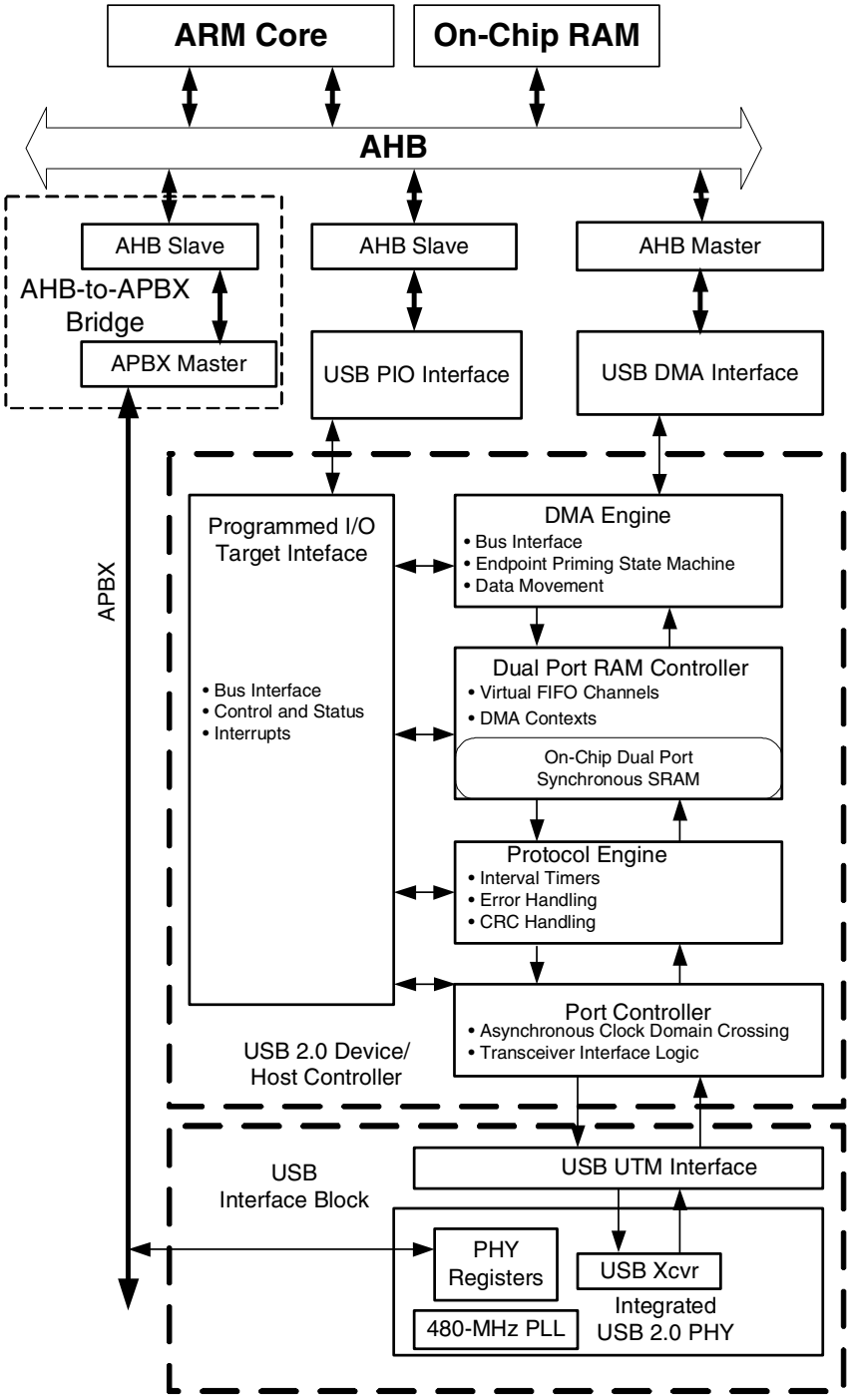


Figure 8-1. USB 2.0 Device Controller Block Diagram

8.2 USB Programmed I/O (PIO) Target Interface

The PIO interface is on an AHB slave of the USB controller. It allows the ARM processor to access the configuration, control, and status registers. There are identification registers for hardware configuration parameters and operational registers for control and status.

8.3 USB DMA Interface

The DMA is a master AHB interface that allows USB data to be transferred to/from the system memory. The data in memory is structured to implement a software framework supported by the controller. For a device controller, this structure is a linked-list interface that consists of queue heads and pointers that are transfer descriptors. The queue head is where transfers are managed. It has status information and location of the data buffers. The hardware controller's PIO registers enable the entire data structure, and once USB data is transferred between the host, the status of the transfer is updated in the queue head, with minimal latency to the system.

For a host controller, there is also a linked-list interface. It consists of a periodic frame list and pointers to transfer descriptors. The period frame list is a schedule of transfers. The frame list points to the data buffers through the transfer descriptors. The hardware controller's PIO registers enable the data structure and manage the transfers within a USB frame. The period frame list works as a sliding window of host transfers over time. As each transfer is completed, the status information is updated in the frame list.

The i.MX23 has the bandwidth to handle the data buffers in DRAM for both high-speed and full-speed USB transmissions. However, the queue heads (dQH) must be placed in on-chip RAM. A design limitation on burst size does not allow the queue heads to be placed in DRAM.

8.4 USB UTM Interface

The USB UTM interface on the i.MX23 implements the specification that allows USB controllers to interface with the USB PHY. Please refer to the *USB 2.0 Transceiver Macrocell Interface (UTMI) Specification, Version 1.05* and *UTMI+ Specification, Version 1.0* for additional details:
http://www.intel.com/technology/usb/spec.htm

8.4.1 Digital/Analog Loopback Test Mode

Since the UTM has to operate at high frequencies (480 MHz), it has a capacity to self-test. A pseudo-random number generator transmits data to the receive path, and data is compared for validity. In the digital loopback, the data transfer only resides in the UTM. It checks for sync, EOP, and bit-stuffing generation and data integrity. The analog loopback is the same as the digital loopback, but involves the analog PHY. This allows for checking of the and full-speed (FS) comparators and transmitters.

8.5 USB Controller Flowcharts

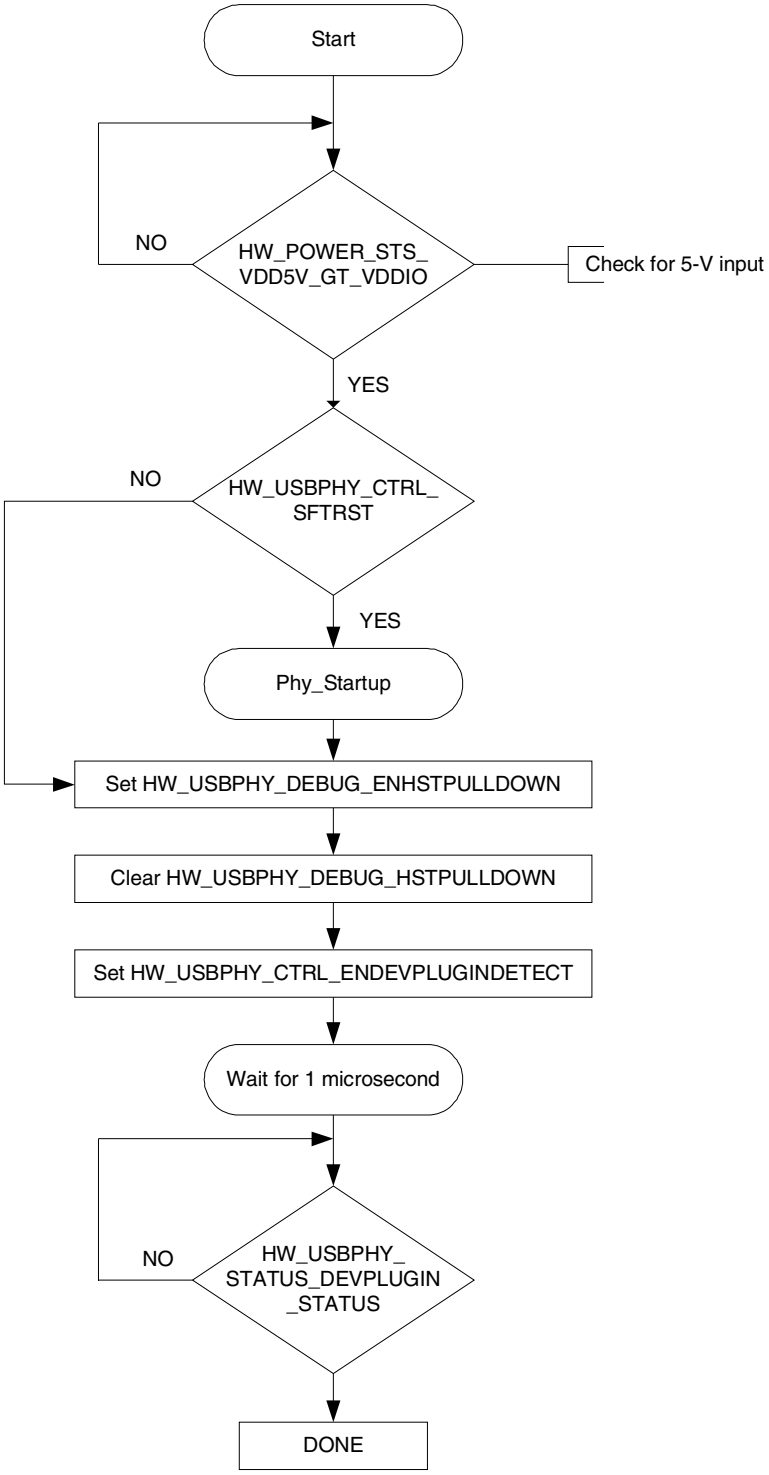


Figure 8-2. USB 2.0 Check_USB_Plugged_In Flowchart

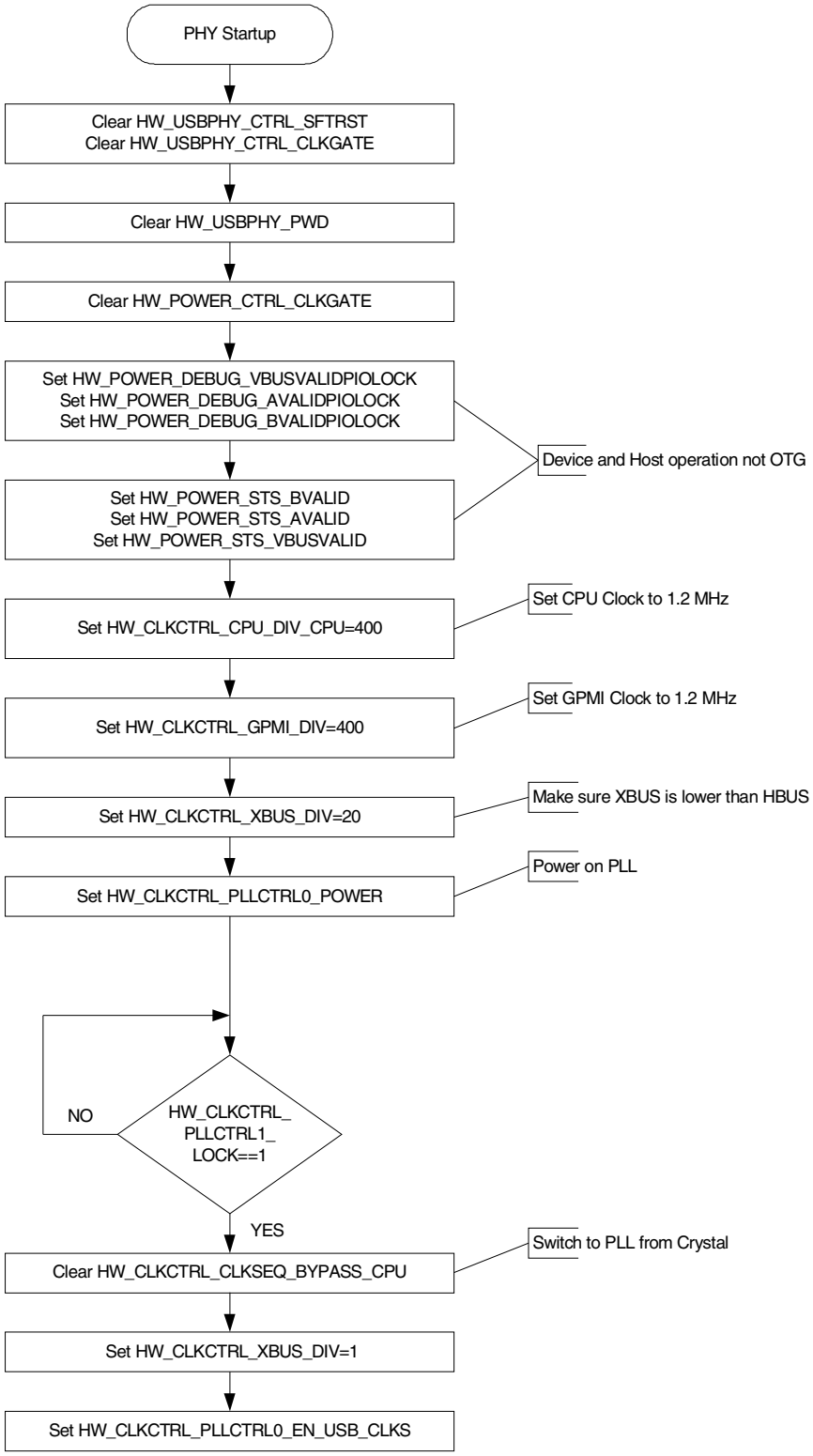


Figure 8-3. USB 2.0 USB PHY Startup Flowchart

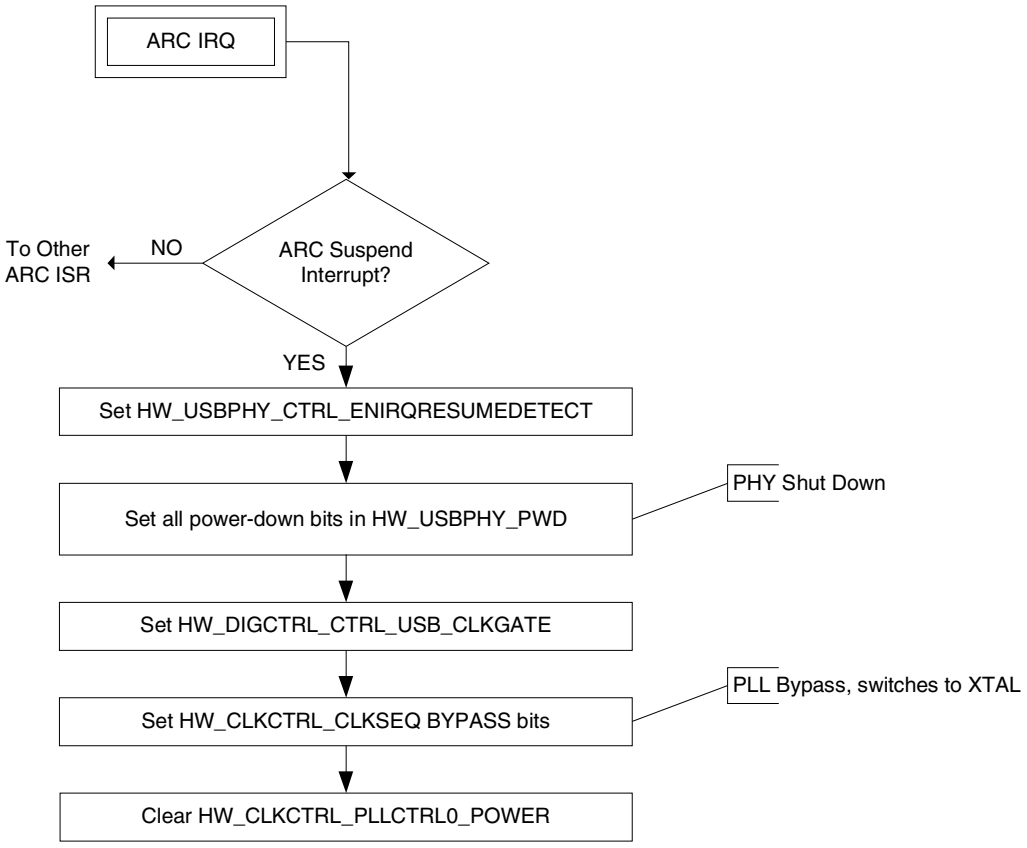


Figure 8-4. USB 2.0 PHY PLL Suspend Flowchart

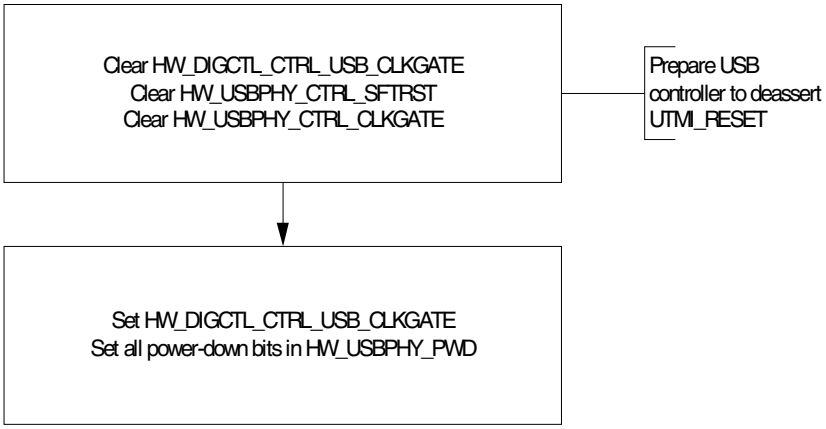


Figure 8-5. UTMI Powerdown

8.5.1 References

- *USB 2.0 Transceiver Macrocell Interface (UTMI) Specification*, Version 1.05, March 2001, Jon Lueker, Steve McGowan (Editor) Ken Oliver, Dean Warren. http://www.intel.com
- *VSI Alliance Virtual Component Interface Standard*, Version 2 (OCB 2 2.0), April 2001, On-Chip Bus Development Working Group. http://www.vsi.org
- *Universal Serial Bus Specification*, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org
- *On-The-Go Supplement to the USB 2.0 Specification*, Revision 1.0, Dec 2001, On-The-Go Working Group of the USB-IF. http://www.usb.org
- *Enhanced Host Controller Interface Specification for Universal Serial Bus*, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com
- *Universal Serial Bus Specification*, Revision 1.1, September 1998, Compaq, Intel, Microsoft, NEC. http://www.usb.org
- *AMBA Specification*, Revision 2.0, May 1999, ARM Limited. http://www.arm.com
- *UTMI+ Low Pin Interface (ULPI) Specification*, Revision 1.0 February 2004, ULPI Specification Organization. http://www.ulpi.org

8.6 Programmable Registers

This section includes the programmable registers supported in the USB high-speed Host controller core.

8.6.1 Identification Register Description

The Identification Register provides a simple way to determine if the USB-HS USB 2.0 core is provided in the system. The HW_USBCTRL_ID register identifies the USB-HS USB 2.0 core and its revision. The default value of this register is 0xE241FA05.

HW_USBCTRL_ID 0x000

Table 8-1. HW_USBCTRL_ID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CIVERSION			VERSION				REVISION				TAG					RSVD1		NID						RSVD0		ID						

Table 8-2. HW_USBCTRL_ID Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**CIVERSION**	RO	0x7	Identifies the Chip Idea product version of the USB-HS USB 2.0 core.
28:25	**VERSION**	RO	0x1	Identifies the version of the USB-HS USB 2.0 core release. (<version>.<revision><tag>)
24:21	**REVISION**	RO	0x2	Identifies the revision of the USB-HS USB 2.0 core release. (<version>.<revision><tag>)
20:16	**TAG**	RO	0x1	Identifies the tag of the USB-HS USB 2.0 core release. (<version>.<revision><tag>)
15:14	**RSVD1**	RO	0x3	Reserved.
13:8	**NID**	RO	0x3a	One's complement version of ID[5:0].
7:6	**RSVD0**	RO	0x0	Reserved.
5:0	**ID**	RO	0x05	Configuration number. This number is set to 0x05 and indicates that the peripheral is the USB-HS USB 2.0 core.

DESCRIPTION:

REV/ID

EXAMPLE:

```
Empty Example.
```

8.6.2 General Hardware Parameters Register Description

The default value of this register is 0x00000015.

HW_USBCTRL_HWGENERAL 0x004

Table 8-3. HW_USBCTRL_HWGENERAL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																					SM		PHYM			PHYW		BWT	CLKC		RT

Table 8-4. HW_USBCTRL_HWGENERAL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:11	**RSVD**	RO	0x0	Reserved.
10:9	**SM**	RO	0x0	PHY Serial Engine Type. Always 0 = Serial engine not present.
8:6	**PHYM**	RO	0x0	PHY Type. Always 1 = UTMI.
5:4	**PHYW**	RO	0x1	Data Interface Width to PHY. Always 1 = 16 bits.
3	**BWT**	RO	0x0	Reserved for internal testing. Always 0.

Table 8-4. HW_USBCTRL_HWGENERAL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2:1	**CLKC**	RO	0x2	USB Controller Clocking Method. Always 2 = Mixed clocked.
0	**RT**	RO	0x1	Reset Type. Always 1 = Synchronous

DESCRIPTION:

General Hardware Parameters

EXAMPLE:

```
Empty Example.
```

8.6.3 Host Hardware Parameters Register Description

The default value of this register is 0x10020001.

HW_USBCTRL_HWHOST 0x008

Table 8-5. HW_USBCTRL_HWHOST

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TTPER								TTASY								RSVD												NPORT			HC

Table 8-6. HW_USBCTRL_HWHOST Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**TTPER**	RO	0x10	Periodic contexts for hub TT.
23:16	**TTASY**	RO	0x02	Asynch contexts for hub TT.
15:4	**RSVD**	RO	0x0	Reserved.
3:1	**NPORT**	RO	0x0	Maximum downstream ports minus 1.
0	**HC**	RO	0x1	Host Capable. Always 0x1.

DESCRIPTION:

Host hardware params as defined in sys-level/core-config

EXAMPLE:

```
Empty Example.
```

8.6.4 Device Hardware Parameters Register Description

The default value of this register is 0x0000000B.

HW_USBCTRL_HWDEVICE 0x00c

Table 8-7. HW_USBCTRL_HWDEVICE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																										DEVEP					DC

Table 8-8. HW_USBCTRL_HWDEVICE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:6	**RSVD**	RO	0x0	Reserved.
5:1	**DEVEP**	RO	0x5	Maximum number of endpoints, which is 5.
0	**DC**	RO	0x1	Device Capable. Always 0x1.

DESCRIPTION:

device hardware params as defined in sys-level/core-config

EXAMPLE:

```
Empty Example.
```

8.6.5 TX Buffer Hardware Parameters Register Description

The default value of this register is 0x40060910.

HW_USBCTRL_HWTXBUF 0x010

Table 8-9. HW_USBCTRL_HWTXBUF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TXLCR	RSVD							TXCHANADD								TXADD								TXBURST							

Table 8-10. HW_USBCTRL_HWTXBUF Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**TXLCR**	RO	0x1	Always 0x1.
30:24	**RSVD**	RO	0x0	Reserved.
23:16	**TXCHANADD**	RO	0x06	Number of address bits for the TX buffer.
15:8	**TXADD**	RO	0x09	Always 0x9.
7:0	**TXBURST**	RO	0x10	Burst size for memory-to-TX-buffer transfers.

DESCRIPTION:

tx hardware buf params

EXAMPLE:

```
Empty Example.
```

8.6.6 RX Buffer Hardware Parameters Register Description

The default value of this register is 0x00000710.

HW_USBCTRL_HWRXBUF 0x014

Table 8-11. HW_USBCTRL_HWRXBUF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																RXADD								RXBURST							

Table 8-12. HW_USBCTRL_HWRXBUF Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSVD**	RO	0x0	Reserved.
15:8	**RXADD**	RO	0x07	Always 0x07.
7:0	**RXBURST**	RO	0x10	Burst size for RX buffer-to-memory transfers.

DESCRIPTION:

rx hardware buf params

EXAMPLE:

```
Empty Example.
```

8.6.7 General-Purpose Timer 0 Load (Non-EHCI-Compliant) Register Description

The host/device controller driver can measure time-related activities using this timer register. This register is not part of the standard EHCI controller. This register contains the timer duration or load value. See the GPTIMER0CTRL (Non-EHCI) for a description of the timer functions.

HW_USBCTRL_GPTIMER0LD 0x080

Table 8-13. HW_USBCTRL_GPTIMER0LD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0								GPTLD																							

Table 8-14. HW_USBCTRL_GPTIMER0LD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD0**	RO	0x0	Reserved.
23:0	**GPTLD**	RW	0x0	General-Purpose Timer Load Value. This field is the value to be loaded into the GPTCNT countdown timer on a reset action. This value in this register represents the time in microseconds minus 1 for the timer duration. Example: for a one-millisecond timer, load 1000 Note: Max value is 0xFFFFFF or 16.777215 seconds.

DESCRIPTION:

General Purpose Timer #0 Load Register

EXAMPLE:

```
Empty Example.
```

8.6.8 General-Purpose Timer 0 Control (Non-EHCI-Compliant) Register Description

The host/device controller driver can measure time-related activities using this timer register. This register is not part of the standard EHCI controller. This register contains the control for the timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two modes supported by this timer, the first is a one-shot and the second is a looped count that is described in the register table below. When the timer counter value transitions to 0, an interrupt can be generated through the use of the timer interrupts in the USBTS and USBINTR registers.

HW_USBCTRL_GPTIMER0CTRL 0x084

Table 8-15. HW_USBCTRL_GPTIMER0CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
GPTRUN	GPTRST	RSVD0					GPTMODE	GPTCNT																							

Table 8-16. HW_USBCTRL_GPTIMER0CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**GPTRUN**	RW	0x0	General-Purpose Timer Run. This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT except if it stops or starts counting. STOP = 0 Timer stop. RUN = 1 Timer run.
30	**GPTRST**	WO	0x0	General-Purpose Timer Reset. Writing a 1 to this bit will reload the GPTCNT with the value in GPTLD. NOACTION = 0 No action. LOADCOUNTER = 1 Load counter value.
29:25	**RSVD0**	RO	0x0	Reserved.
24	**GPTMODE**	RW	0x0	General-Purpose Timer Mode. This bit selects between a single timer countdown and a looped count down. In one-shot mode, the timer will count down to 0, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to 0, generate an interrupt, and automatically reload the counter to begin again. ONESHOT = 0 One shot. REPEAT = 1 Repeat.
23:0	**GPTCNT**	RO	0x0	General-Purpose Timer Counter. This field is the value of the running timer.

DESCRIPTION:

General Purpose Timer #0 Control Register

EXAMPLE:

```
Empty Example.
```

8.6.9 General-Purpose Timer 1 Load (Non-EHCI-Compliant) Register Description

Same as GPTIMER0LD description.

HW_USBCTRL_GPTIMER1LD 0x088

Table 8-17. HW_USBCTRL_GPTIMER1LD

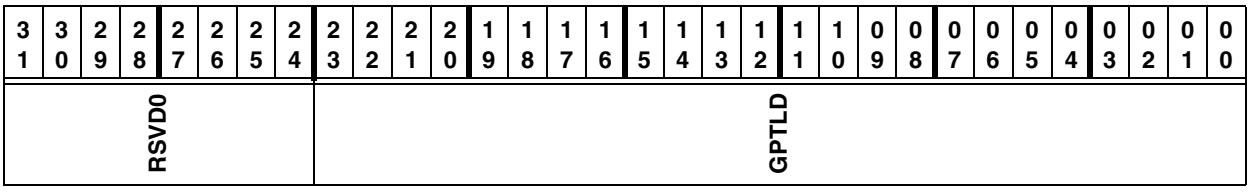


Table 8-18. HW_USBCTRL_GPTIMER1LD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD0**	RO	0x0	Reserved.
23:0	**GPTLD**	RW	0x0	General-Purpose Timer Load Value. This field is the value to be loaded into the GPTCNT countdown timer on a reset action. This value in this register represents the time in microseconds minus 1 for the timer duration. Example: for a one-millisecond timer, load 1000 Note: Max value is 0xFFFFFF or 16.777215 seconds.

DESCRIPTION:

General Purpose Timer #1 Load Register

EXAMPLE:

```
Empty Example.
```

8.6.10 General-Purpose Timer 1 Control (Non-EHCI-Compliant) Register Description

Same as GPTIMER0CTRL description.

HW_USBCTRL_GPTIMER1CTRL 0x08c

Table 8-19. HW_USBCTRL_GPTIMER1CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
GPTRUN	GPTRST	RSVD0					GPTMODE	GPTCNT																							

Table 8-20. HW_USBCTRL_GPTIMER1CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**GPTRUN**	RW	0x0	General-Purpose Timer Run. This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT except if it stops or starts counting. STOP = 0 Timer stop. RUN = 1 Timer run.
30	**GPTRST**	WO	0x0	General-Purpose Timer Reset. Writing a 1 to this bit will reload the GPTCNT with the value in GPTLD. NOACTION = 0 No action. LOADCOUNTER = 1 Load counter value.
29:25	**RSVD0**	RO	0x0	Reserved.

Table 8-20. HW_USBCTRL_GPTIMER1CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
24	**GPTMODE**	RW	0x0	General-Purpose Timer Mode. This bit selects between a single timer countdown and a looped count down. In one-shot mode, the timer will count down to 0, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to 0, generate an interrupt, and automatically reload the counter to begin again. ONESHOT = 0 One shot. REPEAT = 1 Repeat.
23:0	**GPTCNT**	RO	0x0	General-Purpose Timer Counter. This field is the value of the running timer.

DESCRIPTION:

General Purpose Timer #1 Control Register

EXAMPLE:

```
Empty Example.
```

8.6.11 System Bus Configuration (Non-EHCI-Compliant) Register Description

This register controls the AMBA system bus Master/Slave interfaces.

HW_USBCTRL_SBUSCFG 0x090

Table 8-21. HW_USBCTRL_SBUSCFG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																													AHBBRST		

Table 8-22. HW_USBCTRL_SBUSCFG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:3	**RSVD**	RO	0x00	Reserved.
2:0	**AHBBRST**	RW	0x0	AMBA AHB BURST. This field selects the following options for the m_hburst signal of the AMBA master interface: U_INCR = 0x0 INCR burst of unspecified length. S_INCR4 = 0x1 INCR4, non-multiple transfers of INCR4 will be decomposed into singles. S_INCR8 = 0x2 INCR8, non-multiple transfers of INCR8 will be decomposed into INCR4 or singles. S_INCR16 = 0x3 INCR16, non-multiple transfers of INCR16 will be decomposed into INCR8, INCR4 or singles. RESERVED = 0x4 This value is reserved and should not be used. U_INCR4 = 0x5 INCR4, non-multiple transfers of INCR4 will be decomposed into smaller unspecified length bursts. U_INCR8 = 0x6 INCR8, non-multiple transfers of INCR8 will be decomposed into smaller unspecified length bursts. U_INCR16 = 0x7 INCR16, non-multiple transfers of INCR16 will be decomposed into smaller unspecified length bursts.

DESCRIPTION:

AHB System Bus Configuration Register

EXAMPLE:

```
Empty Example.
```

8.6.12 Capability Length and HCI Version (EHCI-Compliant) Register Description

This register contains the Capability Length and HCI Version Register.

HW_USBCTRL_CAPLENGTH 0x100

Table 8-23. HW_USBCTRL_CAPLENGTH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
HCIVERSION																RSVD								CAPLENGTH							

Table 8-24. HW_USBCTRL_CAPLENGTH Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**HCIVERSION**	RO	0x0100	Contains a BCD encoding of the EHCI revision number supported by the host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.
15:8	**RSVD**	RO	0x00	Reserved.
7:0	**CAPLENGTH**	RO	0x40	Offset to add to register base address at beginning of the Operational Register.

DESCRIPTION:

Capability Length and HCI Version register

EXAMPLE:

```
Empty Example.
```

8.6.13 Host Control Structural Parameters (EHCI-Compliant with Extensions) Register Description

Port-steering logic capabilities are described in this register. The default value of this register is 0x00010011.

HW_USBCTRL_HCSPARAMS 0x104

Table 8-25. HW_USBCTRL_HCSPARAMS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2				N_TT				N_PTT				RSVD1			PI	N_CC				N_PCC				RSVD0			PPC	N_PORTS			

Table 8-26. HW_USBCTRL_HCSPARAMS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSVD2**	RO	0x0	Reserved.
27:24	**N_TT**	RO	0x0	Number of Transaction Translators (N_TT). Indicates the number of embedded transaction translators associated with the USB2.0 host controller. This in a non-EHCI field to support embedded TT.
23:20	**N_PTT**	RO	0x0	Number of Ports per Transaction Translator (N_PTT). Indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. This in a non-EHCI field to support embedded TT.
19:17	**RSVD1**	RO	0x0	Reserved.
16	**PI**	RO	0x1	Port Indicators (P INDICATOR). Indicates whether the ports support port indicator control. When set to 1, the port status and control registers include a read/writable field for controlling the state of the port indicator.
15:12	**N_CC**	RO	0x0	Number of Companion Controller (N_CC). Indicates the number of companion controllers associated with this USB2.0 host controller. A 0 in this field indicates there are no internal Companion Controllers. Port-ownership hand-off is not supported. A value larger than 0 in this field indicates there are companion USB host controller(s). Port-ownership hand-offs are supported. High- and Full-speed devices are supported on the host controller root ports.
11:8	**N_PCC**	RO	0x0	Number of Ports per Companion Controller. Indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC.
7:5	**RSVD0**	RO	0x0	Reserved.

Table 8-26. HW_USBCTRL_HCSPARAMS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4	**PPC**	RO	0x1	Port Power Control. Indicates whether the host controller implementation includes port power control. A 1 indicates the ports have port power switches. A 0 indicates the ports do not have port power switches. The value of this field affects the functionality of the Port Power field in each port status and control register.
3:0	**N_PORTS**	RO	0x1	Number of downstream ports. Specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register. Valid values are in the range of 0x11-0xF. A 0 in this field is undefined.

DESCRIPTION:

host controller structural params

EXAMPLE:

```
Empty Example.
```

8.6.14 Host Control Capability Parameters (EHCI-Compliant) Register Description

This register identifies multiple mode control (time-base bit functionality) addressing capability. The default value of this register is 0x00000006.

HW_USBCTRL_HCCPARAMS 0x108

Table 8-27. HW_USBCTRL_HCCPARAMS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2																EECP								IST				RSVD0	ASP	PFL	ADC

Table 8-28. HW_USBCTRL_HCCPARAMS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSVD2**	RO	0x0	Reserved.
15:8	**EECP**	RO	0x0	EHCI Extended Capabilities Pointer. Default = 0. This optional field indicates the existence of a capabilities list. A value of 0x00 indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 0x40 or greater if implemented to maintain the consistency of the PCI header defined for this class of device.
7:4	**IST**	RO	0x0	Isochronous Scheduling Threshold. Indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit 7 is 0, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit 7 is a 1, then host software assumes the host controller may cache an isochronous data structure for an entire frame.
3	**RSVD0**	RO	0x0	Reserved.
2	**ASP**	RO	0x1	Asynchronous Schedule Park Capability. Default = 1. If this bit is set to a 1, then the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register.
1	**PFL**	RO	0x1	Programmable Frame List Flag. If this bit is set to 0, then the system software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to 0. If set to a 1, then the system software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K page boundary. This requirement ensures that the frame list is always physically contiguous.
0	**ADC**	RO	0x0	64-bit Addressing Capability. No 64-bit addressing capability is supported.

DESCRIPTION:

host controller capability params

EXAMPLE:

```
Empty Example.
```

8.6.15 Device Interface Version Number (Non-EHCI-Compliant) Register Description

The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

HW_USBCTRL_DCIVERSION 0x120

Table 8-29. HW_USBCTRL_DCIVERSION

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																DCIVERSION															

Table 8-30. HW_USBCTRL_DCIVERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSVD**	RO	0x0	Reserved.
15:0	**DCIVERSION**	RW	0x1	Two-byte BCD encoding of the interface version number.

DESCRIPTION:

device interface version

EXAMPLE:

```
Empty Example.
```

8.6.16 Device Control Capability Parameters (Non-EHCI-Compliant) Register Description

These fields describe the overall host/device capability of the controller. The default value of this register is 0x00000185.

HW_USBCTRL_DCCPARAMS 0x124

Table 8-31. HW_USBCTRL_DCCPARAMS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1																							HC	DC	RSVD2		DEN				

Table 8-32. HW_USBCTRL_DCCPARAMS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:9	**RSVD1**	RO	0x0	Reserved.
8	**HC**	RO	0x1	Host Capable. When this bit is 1, this controller is capable of operating as an EHCI-compatible USB 2.0 host controller.
7	**DC**	RO	0x1	Device Capable. When this bit is 1, this controller is capable of operating as a USB 2.0 device.
6:5	**RSVD2**	RO	0x0	Reserved.
4:0	**DEN**	RO	0x5	Device Endpoint Number. This field indicates the number of endpoints built into the device controller, which is 5.

DESCRIPTION:

device controller capability params

EXAMPLE:

```
Empty Example.
```

8.6.17 USB Command Register Description

The serial bus host/device controller executes the command indicated in this register. * Default Value:0x00080B00 (Host mode), 0x00080000 (Device mode)

HW_USBCTRL_USBCMD 0x140

Table 8-33. HW_USBCTRL_USBCMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3								ITC								FS2	ATDTW	SUTW	RSVD2	ASPE	RSVD1	ASP		LR	IAA	ASE	PSE	FS1	FS0	RST	RS

Table 8-34. HW_USBCTRL_USBCMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD3**	RO	0x0	Reserved.
23:16	**ITC**	RW	0x8	Interrupt Threshold Control. Default 0x08. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are: IMM = 0x0 Immediate (no threshold). 1_MICROFRAME = 0x1 1_MICROFRAME. 2_MICROFRAME = 0x2 2_MICROFRAME. 4_MICROFRAME = 0x4 4_MICROFRAME. 8_MICROFRAME = 0x8 8_MICROFRAME. 16_MICROFRAME = 0x10 16_MICROFRAME. 32_MICROFRAME = 0x20 32_MICROFRAME. 64_MICROFRAME = 0x40 64_MICROFRAME.

Table 8-34. HW_USBCTRL_USBCMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15	**FS2**	RW	0x0	Bit 2 of Frame List Size field. See definition of bit FS0 for the complete definition.
14	**ATDTW**	RW	0x0	Add dTD TripWire (device mode only). This bit is used as a semaphore to ensure the proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit shall also be cleared by hardware when is state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized.
13	**SUTW**	RW	0x0	Setup TripWire (device mode only). This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (See USBMODE) then there exists a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists.
12	**RSVD2**	RO	0x0	Reserved.
11	**ASPE**	RW	0x0	Asynchronous Schedule Park Mode Enable (OPTIONAL). This bit defaults to 0x1. Software uses this bit to enable or disable Park mode. When this bit is 1, Park mode is enabled. When this bit is a 0, Park mode is disabled. This field is set to 1 in host mode; 0 in device mode.
10	**RSVD1**	RO	0x0	Reserved.
9:8	**ASP**	RW	0x0	Asynchronous Schedule Park Mode Count (OPTIONAL). This field defaults to 0x3 and is R/W. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. See Section 4.10.3.2 of the EHCI specification for full operational details. Valid values are 0x1-0x3. Software must not write a 0 to this bit as this will result in undefined behavior. This field is set to 0x3 in host mode; 0x0 in device mode.
7	**LR**	RW	0x0	Light Host/Device Controller Reset (OPTIONAL). Not Implemented. This field will always be 0.

Table 8-34. HW_USBCTRL_USBCMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
6	**IAA**	RW	0x0	Interrupt on Async Advance Doorbell. This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is 1, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to 0 after it has set the Interrupt on Sync Advance status bit in the USBSTS register to 1. Software should not write a 1 to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a 1 to this bit when device mode is selected will have undefined results.
5	**ASE**	RW	0x0	Asynchronous Schedule Enable. Default 0. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit.
4	**PSE**	RW	0x0	Periodic Schedule Enable. Default Ob. This bit controls whether the host controller skips processing the Periodic Schedule. 0 = Do not process the Periodic Schedule 1 = Use the PERIODICLISTBASE register to access the Periodic Schedule. Only the host controller uses this bit.
3	**FS1**	RW	0x0	Bit 1 of Frame List Size field. See definition of bit FS0 for the complete definition.
2	**FS0**	RW	0x0	Bit 0 of Frame List Size field. The Frame List Size field (FS2, FS1, FS0) specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3 and 2. Default is 000b. 000b = 1024 ELEMENTS (4096 bytes) Default value. 001b = 512_ELEMENTS (2048 bytes). 010b = 256_ELEMENTS (1024 bytes). 011b = 128_ELEMENTS (512 bytes). 100b = 64_ELEMENTS (256 bytes). 101b = 32_ELEMENTS (128 bytes). 110b = 16_ELEMENTS (64 bytes). 111b = 8_ELEMENTS (32 bytes). Only the host controller uses this field.

Table 8-34. HW_USBCTRL_USBCMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**RST**	RW	0x0	Controller Reset (RESET). Software uses this bit to reset the controller. This bit is set to 0 by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a 0 to this register. Host Controller: When software writes a 1 to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a 1 when the HCHalted bit in the USBSTS register is a 0. Attempting to reset an actively running host controller will result in undefined behavior. Device Controller: When software writes a 1 to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a 1 to this bit when the device is in the attached state is not recommended, since the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.
0	**RS**	RW	0x0	Run/Stop (RS). Default 0. 1 = Run. 0 = Stop. Host Controller: When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a 1. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a 1 to this field unless the host controller is in the Halted state (i.e., HCHalted in the USBSTS register is a 1). Device Controller: Writing a 1 to this bit will cause the device controller to enable a pullup on D+ and initiate an attach event. This control bit is not directly connected to the pullup enable, as the pullup will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this will cause a detach event.

DESCRIPTION:

command

EXAMPLE:

```
Empty Example.
```

8.6.18 USB Status Register Description

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus. Software clears certain bits in this register by writing a 1 to them. * Default Value:0x00001000 (Host mode), 0x00000000 (Device mode)

HW_USBCTRL_USBSTS 0x144

Table 8-35. HW_USBCTRL_USBSTS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD5						TI1	TI0	RSVD4				UPI	UAI	RSVD3	NAKI	AS	PS	RCL	HCH	RSVD2	ULPII	RSVD1	SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	UI

Table 8-36. HW_USBCTRL_USBSTS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSVD5**	RO	0x0	Reserved.
25	**TI1**	RW	0x0	General-Purpose Timer Interrupt 1 (GPTINT1). This bit is set when the counter in the GPTIMER1CTRL (Non-EHCI) register transitions to 0. Writing a 1 to this bit will clear it.
24	**TI0**	RW	0x0	General-Purpose Timer Interrupt 0 (GPTINT0). This bit is set when the counter in the GPTIMER0CTRL (Non-EHCI) register transitions to 0. Writing a 1 to this bit will clear it.
23:20	**RSVD4**	RO	0x0	Reserved.
19	**UPI**	RW	0x0	USB Host Periodic Interrupt (USBHSTPERINT). This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be 0.

Table 8-36. HW_USBCTRL_USBSTS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
18	**UAI**	RW	0x0	USB Host Asynchronous Interrupt (USBHSTASYNCINT). This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be 0.
17	**RSVD3**	RO	0x0	Reserved.
16	**NAKI**	RO	0x0	NAK Interrupt Bit. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared.
15	**AS**	RO	0x0	Asynchronous Schedule Status. This bit reports the current real status of the Asynchronous Schedule. When set to 0 the asynchronous schedule status is disabled and if set to 1 the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). Only used by the host controller.
14	**PS**	RO	0x0	Periodic Schedule Status. 0 = Default. This bit reports the current real status of the Periodic Schedule. When set to 0 the periodic schedule is disabled, and if set to 1 the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). Only used by the host controller.
13	**RCL**	RO	0x0	Reclamation. 0 = Default. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller; 0 in device mode.

Table 8-36. HW_USBCTRL_USBSTS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
12	**HCH**	RW	0x0	HC Halted. 1 = Default. This bit is a 0 whenever the Run/Stop bit is a 1. The Host Controller sets this bit to 1 after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error). Only used by the host controller; 0 in device mode.
11	**RSVD2**	RO	0x0	Reserved.
10	**ULPII**	RW	0x0	Not present in this implementation.
9	**RSVD1**	RO	0x0	Reserved.
8	**SLI**	RW	0x0	DC Suspend. 0 = Default. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller.
7	**SRI**	RW	0x0	SOF Received. 0 = Default. When the device controller detects a Start Of (micro) Frame, this bit will be set to a 1. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125ms in HS mode and will be synchronized to the actual SOF that is received. Since the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit.
6	**URI**	RW	0x0	USB Reset Received. 0 = Default. When the device controller detects a USB Reset and enters the default state, this bit will be set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. NOTE: This bit should not normally be used to detect reset during suspend, as this block will normally be clock-gated during that time. Use HW_USBPHY_CTRL_RESUME_IRQ, instead.
5	**AAI**	RW	0x0	Interrupt on Async Advance. 0 = Default. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a 1 to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller.

Table 8-36. HW_USBCTRL_USBSTS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4	**SEI**	RW	0x0	System Error. This bit is not used in this implementation and will always be set to 0.
3	**FRI**	RW	0x0	Frame List Rollover. The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FHINDEX [12] toggles. Only used by the host controller.
2	**PCI**	RW	0x0	Port Change Detect. The Host Controller sets this bit to a 1 when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a 1 when the port controller enters the full or high-speed operational state. When the port controller exits the full or highspeed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively. This bit is not EHCI compatible.
1	**UEI**	RW	0x0	USB Error Interrupt (USBERRINT). When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. See Section 4.15.1 in the EHCI specification for a complete list of host error interrupt conditions. The device controller detects resume signaling only.
0	**UI**	RW	0x0	USB Interrupt (USBINT). This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

DESCRIPTION:

status

EXAMPLE:

```
Empty Example.
```

8.6.19 USB Interrupt Enable Register Description

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt is active. The USB Status register (USBSTS) still shows interrupt sources even if they are disabled by the USBINTR register, allowing polling of interrupt events by the software.

HW_USBCTRL_USBINTR 0x148

Table 8-37. HW_USBCTRL_USBINTR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD5						TIE1	TIE0	RSVD4				UPIE	UAIE	RSVD3	NAKE	RSVD2					ULPIE	RSVD1	SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE

Table 8-38. HW_USBCTRL_USBINTR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSVD5**	RO	0x0	Reserved.
25	**TIE1**	RW	0x0	General-Purpose Timer Interrupt Enable 1. When this bit is a 1, and the GPTINT1 bit in the USBSTS register is a 1, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the GPTINT1 bit.
24	**TIE0**	RW	0x0	General-Purpose Timer Interrupt Enable 0. When this bit is a 1, and the GPTINT0 bit in the USBSTS register is a 1, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the GPTINT0 bit.
23:20	**RSVD4**	RO	0x0	Reserved.
19	**UPIE**	RW	0x0	USB Host Periodic Interrupt Enable. When this bit is a 1, and the USBHSTPERINT bit in the USBSTS register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHSTPERINT bit.
18	**UAIE**	RW	0x0	USB Host Asynchronous Interrupt Enable. RW 0x0 When this bit is a 1, and the USBHSTASYNCINT bit in the USBSTS register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHSTASYNCINT bit.
17	**RSVD3**	RO	0x0	Reserved.
16	**NAKE**	RW	0x0	NAK Interrupt Enable. This bit is set by software if it wants to enable the hardware interrupt for the NAK Interrupt bit. If both this bit and the corresponding NAK Interrupt bit are set, a hardware interrupt is generated.
15:11	**RSVD2**	RO	0x0	Reserved.
10	**ULPIE**	RW	0x0	ULPI Enable. Not used in this implementation.
9	**RSVD1**	RO	0x0	Reserved.

Table 8-38. HW_USBCTRL_USBINTR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	**SLE**	RW	0x0	Sleep Enable. When this bit is a 1, and the DCSuspend bit in the USBSTS register transitions, the device controller will issue an interrupt. The interrupt is acknowledged by software writing a 1 to the DCSuspend bit. Only used by the device controller.
7	**SRE**	RW	0x0	SOF Received Enable. When this bit is a 1, and the SOF Received bit in the USBSTS register is a 1, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the SOF Received bit.
6	**URE**	RW	0x0	USB Reset Enable. When this bit is a 1, and the USB Reset Received bit in the USBSTS register is a 1, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller.
5	**AAE**	RW	0x0	Interrupt on Async Advance Enable. When this bit is a 1, and the Interrupt on Async Advance bit in the USBSTS register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller.
4	**SEE**	RW	0x0	System Error Enable. When this bit is a 1, and the System Error bit in the USBSTS register is a 1, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the System Error bit.
3	**FRE**	RW	0x0	Frame List Rollover Enable. When this bit is a 1, and the Frame List Rollover bit in the USBSTS register is a 1, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller.
2	**PCE**	RW	0x0	Port Change Detect Enable. When this bit is a 1, and the Port Change Detect bit in the USBSTS register is a 1, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port Change Detect bit.
1	**UEE**	RW	0x0	USB Error Interrupt Enable. When this bit is a 1, and the USBERRINT bit in the USBSTS register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register.
0	**UE**	RW	0x0	USB Interrupt Enable. When this bit is a 1, and the USBINT bit in the USBSTS register is a 1, the host/device controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit.

DESCRIPTION:

interrupt enables

EXAMPLE:

```
Empty Example.
```

8.6.20 USB Frame Index Register Description

This register is used by the host controller to index the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the USBCMD register. This register must be written as a DWord. Byte writes produce-undefined results. This register cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit. A write to this register while the Run/Stop bit is set to a 1 produces undefined results. Writes to this register also affect the SOF value. In device mode this register is Read-Only and, the device controller updates the FRINDEX [13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX [13:3] will be checked against the SOF marker. If FRINDEX [13:3] is different from the SOF marker, FRINDEX [13:3] will be set to the SOF value and FRINDEX [2:0] will be set to 0 (i.e., SOF for 1 ms frame). If FRINDEX [13:3] is equal to the SOF value, FRINDEX [2:0] will be incremented (i.e., SOF for 125 us micro-frame.) * The default value of this register is undefined (free-running counter).

HW_USBCTRL_FRINDEX 0x14c

Table 8-39. HW_USBCTRL_FRINDEX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																		FRINDEX											UINDEX		

Table 8-40. HW_USBCTRL_FRINDEX Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:14	**RSVD**	RO	0x0	Reserved.
13:3	**FRINDEX**	RO	0x0	Frame List Current Index. Read/write in host mode. Read in device mode. The value in this register increments at the end of each time frame (e.g., micro-frame). Bits [N: 3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. In device mode, the value is the current frame number of the last frame transmitted. It is not used as an index. N_12 = 12 FRAME LIST SIZE = 1024, USBCMD = 3'b000. N_11 = 11 FRAME LIST SIZE = 512, USBCMD = 3'b001. N_10 = 10 FRAME LIST SIZE = 256, USBCMD = 3'b010. N_9 = 9 FRAME LIST SIZE = 128, USBCMD = 3'b011. N_8 = 8 FRAME LIST SIZE = 64, USBCMD = 3'b100. N_7 = 7 FRAME LIST SIZE = 32, USBCMD = 3'b101. N_6 = 6 FRAME LIST SIZE = 16, USBCMD = 3'b110. N_5 = 5 FRAME LIST SIZE = 8, USBCMD = 3'b111.
2:0	**UINDEX**	RW	0x0	Current Microframe.

DESCRIPTION:

frame index

EXAMPLE:

```
Empty Example.
```

8.6.21 Frame List Base Address Register (Host Controller mode) Description

In Host Controller mode, this 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. This is a read/write register. Writes must be DWORD writes.

HW_USBCTRL_PERIODICLISTBASE 0x154

Table 8-41. HW_USBCTRL_PERIODICLISTBASE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PERBASE																				RSVD											

Table 8-42. HW_USBCTRL_PERIODICLISTBASE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	**PERBASE**	RW	0x0	Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.
11:0	**RSVD**	RO	0x0	Reserved. Must be written as zeros. During runtime, the values of these bits are undefined.

DESCRIPTION:

PERIODIC LIST BASE (host-controller)

EXAMPLE:

```
Empty Example.
```

8.6.22 USB Device Address Register (Device Controller mode) Description

In Device Controller mode, the upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor. The USBADRA is used to accelerate the SET_ADDRESS sequence by allowing the DCD to preset the USBADR register before the status phase of the SET_ADDRESS descriptor. This is a read/write register. Writes must be DWORD writes.

HW_USBCTRL_DEVICEADDR 0x154

Table 8-43. HW_USBCTRL_DEVICEADDR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
USBADR							USBADRA	RSVD																							

Table 8-44. HW_USBCTRL_DEVICEADDR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**USBADR**	RW	0x0	Device Address. These bits correspond to the USB device address.
24	**USBADRA**	RW	0x0	Device Address Advance. Default=0. When this bit is `0', any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1. IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2. OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3. Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD cannot write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
23:0	**RSVD**	RO	0x0	Reserved. Must be written as zeros. During runtime, the values of these bits are undefined.

DESCRIPTION:

DEVICE-ADDR

EXAMPLE:

```
Empty Example.
```

8.6.23 Next Asynchronous Address Register (Host Controller mode) Description

In Host Controller mode, this 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a 0 when read.

HW_USBCTRL_ASYNCLISTADDR 0x158

Table 8-45. HW_USBCTRL_ASYNCLISTADDR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ASYBASE																												RSVD			

Table 8-46. HW_USBCTRL_ASYNCLISTADDR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**ASYBASE**	RW	0x0	Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.
4:0	**RSVD**	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.

DESCRIPTION:

ASYNC-LIST-ADDR (host-controller)

EXAMPLE:

```
Empty Example.
```

8.6.24 Endpoint List Address Register (Device Controller mode) Description

In Device Controller mode, this register contains the address of the top of the endpoint list in system memory. Bits [10:0] of this register cannot be modified by the system software and will always return a 0 when read. The memory structure referenced by this physical memory pointer is assumed 64-byte. This is a read/write register. Writes must be DWORD writes.

HW_USBCTRL_ENDPOINTLISTADDR 0x158

Table 8-47. HW_USBCTRL_ENDPOINTLISTADDR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
EPBASE																					RSVD										

Table 8-48. HW_USBCTRL_ENDPOINTLISTADDR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:11	**EPBASE**	RW	0x0	Endpoint List Pointer (Low). These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). (i.e., one queue head per endpoint and direction.)
10:0	**RSVD**	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.

DESCRIPTION:

EndPoint List Address

EXAMPLE:

```
Empty Example.
```

8.6.25 Embedded TT Asynchronous Buffer Status and Control Register (Host Controller mode) Description

This register contains parameters needed for internal TT operations. This register is not used in the device controller operation. This is a read/write register. Writes must be DWORD writes.

HW_USBCTRL_TTCTRL 0x15c

Table 8-49. HW_USBCTRL_TTCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1	TTHA							RSVD2																							

Table 8-50. HW_USBCTRL_TTCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSVD1**	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.
30:24	**TTHA**	RW	0x0	Internal TT Hub Address Representation. Default is 0 (Read/Write). This field is used to match against the Hub Address field in QH and siTD to determine if the packet is routed to the internal TT for directly attached FS/LS devices. If the Hub Address in the QH or siTD does not match this address then the packet will be broadcast on the High Speed ports destined for a downstream High Speed hub with the address in the QH/siTD.
23:0	**RSVD2**	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.

DESCRIPTION:

TT (internal operation) Control

EXAMPLE:

```
Empty Example.
```

8.6.26 Programmable Burst Size Register Description

This register is used to control dynamically change the burst size used during data movement on the initiator (master) interface. This is a read/write register. Writes must be DWORD writes. The default value is 0x00001010.

HW_USBCTRL_BURSTSIZE 0x160

Table 8-51. HW_USBCTRL_BURSTSIZE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																TXPBURST								RXPBURST							

Table 8-52. HW_USBCTRL_BURSTSIZE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSVD**	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.
15:8	**TXPBURST**	RW	0x10	Programmable TX Burst Length. This register represents the maximum length of a the burst in 32-bit words while moving data from system memory to the USB bus.
7:0	**RXPBURST**	RW	0x10	Programmable RX Burst Length. This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory.

DESCRIPTION:

controls (dynamically) burst size for usb->ahb

EXAMPLE:

```
Empty Example.
```

8.6.27 Host Transmit Pre-Buffer Packet Timing Register Description

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system. Definitions: T0 = Standard packet overhead T1 = Time to send data payload Tff = Time to

fetch packet into TX FIFO up to specified level. Ts = Total Packet Flight Time (send-only) packet Ts = T0 + T1 Tp = Total Packet Time (fetch and send) packet Tp = Tff + T0 + T1 Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure Tp remains before the end of the (micro)frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the (micro)frame is < Ts then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a mark will be made the scheduler health counter to note the occurrence of a "back-off" event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the TSCHHEALTH (Tff) described below. This is a read/write register. Writes must be DWORD writes. The default value of this register is 0x00000000.

HW_USBCTRL_TXFILLTUNING 0x164

Table 8-53. HW_USBCTRL_TXFILLTUNING

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2										TXFIFOTHRES						RSVD1			TXSCHEALTH					RSVD0	TXSCHOH						

Table 8-54. HW_USBCTRL_TXFILLTUNING Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:22	**RSVD2**	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.
21:16	**TXFIFOTHRES**	RW	0x0	FIFO Burst Threshold. This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
15:13	**RSVD1**	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.

Table 8-54. HW_USBCTRL_TXFILLTUNING Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
12:8	**TXSCHEALTH**	RW	0x0	Scheduler Health Counter. This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31.
7	**RSVD0**	RO	0x0	Reserved. This bit is reserved and its value has no effect on operation.
6:0	**TXSCHOH**	RW	0x0	Scheduler Overhead. This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode for OTG & SPH. The time unit represented in this register is 6.333us when a device is connected in Full Speed Mode for OTG & SPH. The time unit represented in this register is always 1.267 in the MPH product.

DESCRIPTION:

TX Fill Tuning

EXAMPLE:

```
Empty Example.
```

8.6.28 Inter-Chip Control Register Description

This register is present but not used in this implementation.

HW_USBCTRL_IC_USB 0x16c

Table 8-55. HW_USBCTRL_IC_USB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																												IC_ENABLE	IC_VDD		

Table 8-56. HW_USBCTRL_IC_USB Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:4	**RSVD**	RO	0x0	Reserved.
3	**IC_ENABLE**	RW	0x0	Inter-Chip Transceiver Enable. These bits enables the InterChip transceiver for each port (for the MPH case). To enable the interface, the bits PTS must be set to 0b11 in the PORTSCx. Writing a '1' to each bit selects the IC_USB interface for that port. If the Controller is not MultiPort, IC8 to IC2 will be '0' and Read-Only.
2:0	**IC_VDD**	RW	0x0	Inter-Chip Transceiver Voltage. Selects the voltage being supplied to the peripheral through each port (MPH case). VOLTAGE_NONE = 0x0 . VOLTAGE_1_0 = 0x1 . VOLTAGE_1_2 = 0x2 . VOLTAGE_1_5 = 0x3 . VOLTAGE_1_8 = 0x4 . VOLTAGE_3_0 = 0x5 . RESERVED0 = 0x6 . RESERVED1 = 0x7 .

DESCRIPTION:

This register enables and controls the IC_USB FS/LS transceiver.

EXAMPLE:

```
Empty Example.
```

8.6.29 ULPI Viewport Register Description

This register is present but not used in this implementation.

HW_USBCTRL_ULPI 0x170

Table 8-57. HW_USBCTRL_ULPI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ULPIWU	ULPIRUN	ULPIRW	RSVD0	ULPISS	ULPIPORT			ULPIADDR								ULPIDATRD								ULPIDATWR							

Table 8-58. HW_USBCTRL_ULPI Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**ULPIWU**	RW	0x0	Not used. Read as 0.
30	**ULPIRUN**	RW	0x0	Not used. Read as 0.
29	**ULPIRW**	RW	0x0	Not used. Read as 0.
28	**RSVD0**	RO	0x0	Not used. Read as 0.
27	**ULPISS**	RO	0x0	Not used. Read as 0.
26:24	**ULPIPORT**	RW	0x0	Not used. Read as 0.
23:16	**ULPIADDR**	RW	0x0	Not used. Read as 0.

Table 8-58. HW_USBCTRL_ULPI Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:8	**ULPIDATRD**	RO	0x0	Not used. Read as 0.
7:0	**ULPIDATWR**	RW	0x0	Not used. Read as 0.

DESCRIPTION:

ULPI control

EXAMPLE:

```
Empty Example.
```

8.6.30 Endpoint NAK Register Description

HW_USBCTRL_ENDPTNAK 0x178

Table 8-59. HW_USBCTRL_ENDPTNAK

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1											EPTN					RSVD0											EPRN				

Table 8-60. HW_USBCTRL_ENDPTNAK Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	**RSVD1**	RO	0x0	Reserved.
20:16	**EPTN**	RW	0x0	TX Endpoint NAK. Each TX endpoint has one bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. EPTN[4] = Endpoint 4 EPTN[3] = Endpoint 3 EPTN[2] = Endpoint 2 EPTN[1] = Endpoint 1 EPTN[0] = Endpoint 0
15:5	**RSVD0**	RO	0x0	Reserved.
4:0	**EPRN**	RW	0x0	RX Endpoint NAK. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. EPRN[4] = Endpoint 4 EPRN[3] = Endpoint 3 EPRN[2] = Endpoint 2 EPRN[1] = Endpoint 1 EPRN[0] = Endpoint 0

DESCRIPTION:

NAK-sent indicator

EXAMPLE:

```
Empty Example.
```

8.6.31 Endpoint NAK Enable Register Description

HW_USBCTRL_ENDPTNAKEN 0x17c

Table 8-61. HW_USBCTRL_ENDPTNAKEN

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1											EPTNE					RSVD0											EPRNE				

Table 8-62. HW_USBCTRL_ENDPTNAKEN Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	**RSVD1**	RO	0x0	Reserved.
20:16	**EPTNE**	RW	0x0	TX Endpoint NAK Enable. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. EPTNE[4] = Endpoint 4 EPTNE[3] = Endpoint 3 EPTNE[2] = Endpoint 2 EPTNE[1] = Endpoint 1 EPTNE[0] = Endpoint 0
15:5	**RSVD0**	RO	0x0	Reserved.
4:0	**EPRNE**	RW	0x0	RX Endpoint NAK Enable. Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. EPRNE[4] = Endpoint 4 EPRNE[3] = Endpoint 3 EPRNE[2] = Endpoint 2 EPRNE[1] = Endpoint 1 EPRNE[0] = Endpoint 0

DESCRIPTION:

NAK-sent indicator enable

EXAMPLE:

```
Empty Example.
```

8.6.32 Port Status and Control 1 Register Description

Host Controller: A host controller must implement one to eight port registers. The number of port registers implemented by a particular instantiation of a host controller is documented in the HCSPARAMs register and is fixed at 1 in this implementation. Software uses this information as an input parameter to determine

how many ports need service. This register is only reset when power is initially applied or in response to a controller reset. The initial conditions of a port are: - No device connected - Port disabled If the port has port power control, this state remains until software applies power to the port by setting port power to 1. Device Controller: A device controller must implement only port register 1 and it does not support power control. Port control in device mode is only used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling and allows software to put the PHY into low power suspend mode and disable the PHY clock. * Default Value: 00010000000000000000XX0000000000b (Host mode) 00010000000000000001XX0000000100b (Device mode) X = Unknown

HW_USBCTRL_PORTSC1 0x184

Table 8-63. HW_USBCTRL_PORTSC1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PTS		STS	PTW	PSPD		SRT	PFSC	PHCD	WKOC	WKDS	WKCN	PTC				PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS

Table 8-64. HW_USBCTRL_PORTSC1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	**PTS**	RW	0x0	Parallel Transceiver Select. For this implementation, always set to 00b for UTMI. UTMI = 0 UTMI/UTMI+. PHIL = 1 Phillips-Classic. ULPI = 2 ULPI. SERIAL = 3 Serial/1.1FS.
29	**STS**	RW	0x0	Serial Transceiver Select. Always 0.
28	**PTW**	RW	0x1	Parallel Transceiver Width. This bit is always 0, indicating an 8-bit (60-MHz) UTMI interface.
27:26	**PSPD**	RW	0x0	Port Speed. This register field indicates the speed at which the port is operating. For high-speed mode operation in the host controller and high-speed/fullspeed operation in the device controller, the port routing steers data to the protocol engine. This bit is not defined in the EHCI specification. FULL = 0 Full Speed. HIGH = 2 High Speed.
25	**SRT**	RW	0x0	Reserved.
24	**PFSC**	RW	0x0	Port Force Full Speed Connect. Default = 0. Writing this bit to a 1 will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as high-speed. This is useful for testing full-speed configurations with a high-speed host, hub or device. This bit is not defined in the EHCI specification. This bit is for debugging purposes.

Table 8-64. HW_USBCTRL_PORTSC1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23	**PHCD**	RW	0x0	PHY Low Power Suspend - Clock Disable (PLPSCD). Default = 0. Writing this bit to a 1 will disable the PHY clock. Writing a 0 enables it. Reading this bit will indicate the status of the PHY clock. In Device Mode: The PHY can be put into Low Power Suspend running (USBCMD Run/Stop=0) or the host has signaled suspend (PORTSC SUSPEND=1). Lowpower suspend will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the device controller driver must clear this bit. In Host Mode: The PHY can be put into Low Power Suspend device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software. This bit is not defined in the EHCI specification.
22	**WKOC**	RW	0x0	Wake on Over-current Enable (WKOC_E). Default = 0. Writing this bit to a 1 enables the port to be sensitive to over-current conditions as wake-up events. This field is 0 if Port Power (PP) is 0.
21	**WKDS**	RW	0x0	Wake on Disconnect Enable (WKDSCNNT_E). Default=0. Writing this bit to a 1 enables the port to be sensitive to device disconnects as wake-up events. This field is 0 if Port Power (PP) is 0 or in device mode.
20	**WKCN**	RW	0x0	Wake on Connect Enable (WKCNNT_E). Default=0. Writing this bit to a 1 enables the port to be sensitive to device connects as wake-up events. This field is 0 if Port Power (PP) is 0 or in device mode.
19:16	**PTC**	RW	0x0	Port Test Control. Default = 0000b. Any other value than 0 indicates that the port is operating in test mode. Refer to Chapter 9 of the USB Specification Revision 2.0 for details on each test mode. The TEST_FORCE_ENABLE_FS and TEST_FORCE_ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the TEST_FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_DISABLE will allow the port state machines to progress normally from that point. Note: Low speed operations are not supported. TEST_DISABLE = 0 Disable. TEST_J_STATE = 1 J-State. TEST_K_STATE = 2 K-State. TEST_J_SE0_NAK = 3 Host:SE0/Dev:NAK. TEST_PACKET = 4 Test-Packet. TEST_FORCE_ENABLE_HS = 5 Force-Enable-HS. TEST_FORCE_ENABLE_FS = 6 Force-Enable-FS. TEST_FORCE_ENABLE_LS = 7 Force-Enable-LS.

Table 8-64. HW_USBCTRL_PORTSC1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:14	**PIC**	RW	0x0	Port Indicator Control. Default = 0. Refer to the USB Specification Revision 2.0 for a description on how these bits are to be used. OFF = 0 OFF. AMBER = 1 Amber. GREEN = 2 Green. UNDEF = 3 undefined.
13	**PO**	RW	0x0	Port Owner. Port owner handoff is not implemented in this design, therefore this bit will always read back as 0. The EHCI definition is include here for reference: Default = 0. This bit unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is 0. System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a 1 to this bit when the attached device is not a high-speed device. A 1 in this bit means that an internal companion controller owns and controls the port.
12	**PP**	RW	0x0	Port Power (PP). This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e., PP equals a 0), the port is nonfunctional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port, the PP bit in each affected port may be transitioned by the host controller driver from a 1 to a 0 (removing power from the port). This feature is implemented in the host/OTG controller (PPC = 1). In a device implementation, port power control is not necessary.
11:10	**LS**	RW	0x0	Line Status. These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The bit encodings are listed below. In Host Mode: The use of linestate by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In Device Mode: The use of linestate by the device controller driver is not necessary. SE0 = 0 SE0. K_STATE = 1 K. J_STATE = 2 J. UNDEF = 3 Undefined.

Table 8-64. HW_USBCTRL_PORTSC1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
9	**HSP**	RW	0x0	High-Speed Port. Default = 0. When the bit is 1, the host/device connected to the port is in high-speed mode and if set to 0, the host/device connected to the port is not in a high-speed mode. Note: HSP is redundant with PSPD(27:26) but will remain in the design for compatibility. This bit is not defined in the EHCI specification.
8	**PR**	RW	0x0	Port Reset This field is 0 if Port Power (PP) is 0. In Host Mode: (Read/Write). 1 = Port is in Reset. 0 = Port is not in Reset. Default 0. When software writes a 1 to this bit, the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to 0 after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a 0 after the reset duration is timed in the driver. In Device Mode: This bit is a Read-Only status bit. Device reset from the USB bus is also indicated in the USBSTS register.

Table 8-64. HW_USBCTRL_PORTSC1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7	**SUSP**	RW	0x0	Suspend In Host Mode: (Read/Write) 0 = Port not in suspend state. 1 = Port in suspend state. Default = 0. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits Port State 0x Disable 10 Enable 11 Suspend When in suspend state, the downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to 0 when software sets the Force Port Resume bit to 0. The host controller ignores a write of 0 to this bit. If host software sets this bit to a 1 when the port is not enabled (i.e., Port enabled bit is a 0) the results are undefined. This field is 0 if Port Power (PP) is 0 in host mode. In Device Mode: (Read-Only) 1 = Port in suspend state. 0 = Port not in suspend state. Default=0. In device mode, this bit is a Read-Only status bit.

Table 8-64. HW_USBCTRL_PORTSC1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
6	**FPR**	RW	0x0	Force Port Resume. 0 = No resume (K-state) detected/driven on port. 1 = Resume detected/driven on port. Default = 0. In Host Mode: Software sets this bit to 1 to drive resume signaling. The Host Controller sets this bit to 1 if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a 1 because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to 1. This bit will automatically change to 0 after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a 0 after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a 1. This bit will remain a 1 until the port has switched to the high-speed idle. Writing a 0 has no effect because the port controller will time the resume operation and clear the bit when the port control state switches to HS or FS idle. This field is 0 if Port Power (PP) is 0 in host mode. This bit is not-EHCI compatible. In Device Mode: After the device has been in Suspend State for 5 ms or more, software must set this bit to 1 to drive resume signaling before clearing. The Device Controller will set this bit to 1 if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a 1 because a J-to-K transition has been detected, the Port Change Detect bit in the USBSTS register is also set to 1.
5	**OCC**	RW	0x0	Over-Current Change. 0 = Default. 1 = This bit gets set to 1 when there is a change to Over-Current Active. Software clears this bit by writing a 1 to this bit position. For host/OTG implementations, the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device-only implementations, this bit shall always be 0.

Table 8-64. HW_USBCTRL_PORTSC1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4	**OCA**	RW	0x0	Over-Current Active. 0 = This port does not have an over-current condition. 1 = This port currently has an over-current condition. Default = 0. This bit will automatically transition from 1 to 0 when the over current condition is removed. For host/OTG implementations the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device-only implementations this bit shall always be 0.
3	**PEC**	RW	0x0	Port Enable/Disable Change. 0 = No change. 1 = Port enabled/disabled status has changed. Default = 0. In Host Mode: For the root hub, this bit gets set to a 1 only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a 1 to it. This field is 0 if Port Power (PP) is 0. In Device Mode: The device port is always enabled. (This bit will be 0)
2	**PE**	RW	0x0	Port Enabled/Disabled. 0 = Disable. 1 = Enable. Default = 0. In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a 1 to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0) downstream propagation of data is blocked except for reset. This field is 0 if Port Power (PP) is 0 in host mode. In Device Mode: The device port is always enabled. (This bit will be 1)

Table 8-64. HW_USBCTRL_PORTSC1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**CSC**	RW	0x0	Connect Status Change. 0 = No change. 1 = Change in Current Connect Status. Default = 0. In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a 1 to it. This field is 0 if Port Power (PP) is 0 in host mode. In Device Mode: This bit is undefined in device controller mode.
0	**CCS**	RW	0x0	Current Connect Status. In Host Mode: 0 = No device is present. 1 = Device is present on port. Default = 0. This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is 0 if Port Power (PP) is 0 in host mode. In Device Mode: 0 = Not Attached. 1 = Attached. Default = 0. A 1 indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A 0 indicates that the device did not attach successfully or was forcibly disconnected by the software writing a 0 to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended.

DESCRIPTION:

port status and control

EXAMPLE:

```
Empty Example.
```

8.6.33 OTG Status and Control Register Description

Host Controller: A host controller implements one On-The-Go (OTG) Status and Control register corresponding to Port 0 of the host controller. The OTGSC register has four sections: OTG Interrupt Enables (Read/Write) OTG Interrupt Status (Read/Write to Clear) OTG Status Inputs (Read-Only) OTG Controls (Read/Write) The status inputs are debounced using a 1-ms time constant. Values on the status

inputs that do not persist for more than 1 ms will not cause an update of the status input register, or cause an OTG interrupt. See also USBMODE register. The default value of this register is 0x00000120.

HW_USBCTRL_OTGSC 0x1a4

Table 8-65. HW_USBCTRL_OTGSC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2	DPIE	ONEMSE	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	RSVD1	DPIS	ONEMSS	BSEIS	BSVIS	ASVIS	AVVIS	IDIS	RSVD0	DPS	ONEMST	BSE	BSV	ASV	AVV	ID	HABA	HADP	IDPU	DP	OT	HAAR	VC	VD

Table 8-66. HW_USBCTRL_OTGSC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSVD2**	RO	0x0	Reserved.
30	**DPIE**	RW	0x0	Data Pulse Interrupt Enable
29	**ONEMSE**	RW	0x0	1 Millisecond Timer Interrupt Enable
28	**BSEIE**	RW	0x0	B Session End Interrupt Enable. Setting this bit enables the B session end interrupt.
27	**BSVIE**	RW	0x0	B Session Valid Interrupt Enable. Setting this bit enables the B session valid interrupt.
26	**ASVIE**	RW	0x0	A Session Valid Interrupt Enable. Setting this bit enables the A session valid interrupt.
25	**AVVIE**	RW	0x0	A VBus Valid Interrupt Enable. Setting this bit enables the A VBus valid interrupt.
24	**IDIE**	RW	0x0	USB ID Interrupt Enable. Setting this bit enables the USB ID interrupt.
23	**RSVD1**	RO	0x0	Reserved.
22	**DPIS**	RW	0x0	Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0).PortPower = Off (0). Software must write a 1 to clear this bit.
21	**ONEMSS**	RW	0x0	1 Millisecond Timer Interrupt Status. This bit is set once every millisecond. Software must write a 1 to clear this bit.
20	**BSEIS**	RW	0x0	B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software must write a 1 to clear this bit.
19	**BSVIS**	RW	0x0	B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software must write a 1 to clear this bit.
18	**ASVIS**	RW	0x0	A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software must write a 1 to clear this bit.

Table 8-66. HW_USBCTRL_OTGSC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
17	**AVVIS**	RW	0x0	A VBus Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software must write a 1 to clear this bit.
16	**IDIS**	RW	0x0	USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software must write a 1 to clear this bit.
15	**RSVD0**	RO	0x0	Reserved.
14	**DPS**	RW	0x0	Data Bus Pulsing Status. A 1 indicates data bus pulsing is being detected on the port.
13	**ONEMST**	RW	0x0	1 Millisecond Timer Toggle. This bit toggles once per millisecond.
12	**BSE**	RO	0x0	B Session End. Indicates VBus is below the B session end threshold.
11	**BSV**	RO	0x0	B Session Valid. Indicates VBus is above the B session valid threshold.
10	**ASV**	RO	0x0	A Session Valid. Indicates VBus is above the A session valid threshold.
9	**AVV**	RO	0x0	A VBus Valid. Indicates VBus is above the A VBus valid threshold.
8	**ID**	RO	0x1	USB ID. 0 = A device. 1 = B device.
7	**HABA**	RW	0x0	Hardware Assist B-Disconnect to A-connect. 0 = Disabled. 1 = Enable automatic B-disconnect to A-connect sequence.
6	**HADP**	RW	0x0	Hardware Assist Data-Pulse 1 = Start Data Pulse Sequence.
5	**IDPU**	RW	0x1	ID Pullup. This bit provide control over the ID pullup resister. 0 = off. 1 = on (default). When this bit is 0, the ID input will not be sampled.
4	**DP**	RW	0x0	Data Pulsing. Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
3	**OT**	RW	0x0	OTG Termination. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM.
2	**HAAR**	RW	0x0	Hardware Assist Auto-Reset. 0 = Disabled. 1 = Enable automatic reset after connect on host port.
1	**VC**	RW	0x0	VBUS Charge. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
0	**VD**	RW	0x0	VBUS_Discharge. Setting this bit causes VBus to discharge through a resistor.

DESCRIPTION:

OTG status/control

EXAMPLE:

```
Empty Example.
```

8.6.34 USB Device Mode Register Description

Default Value:0x00000000 (implementation OTGmode not selected).

HW_USBCTRL_USBMODE 0x1a8

Table 8-67. HW_USBCTRL_USBMODE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																										VBPS	SDIS	SLOM	ES	CM	

Table 8-68. HW_USBCTRL_USBMODE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:6	**RSVD**	RO	0x0	Reserved.
5	**VBPS**	RW	0x0	Vbus Power Select 0 = Output is 0. 1 = Output is 1. This bit is connected to the vbus_pwr_select output and can be used for any generic control but is named to be used by logic that selects between an on-chip Vbus power source (charge pump) and an off-chip source in systems when both are available.

Table 8-68. HW_USBCTRL_USBMODE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4	**SDIS**	RW	0x0	Stream Disable Mode. 0 = Inactive (default). 1 = Active. In Device Mode: Setting to a 1 disables double priming on both RX and TX for low bandwidth systems. This mode, when enabled, ensures that the RX and TX buffers are sufficient to contain an entire packet, so that the usual double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note: In High Speed Mode, all packets received will be responded to with a NYET handshake when stream disable is active. In Host Mode: Setting to a 1 ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB. Note: Time duration to pre-fill the FIFO becomes significative when stream disable is active. See TXFILLTUNING and TXTTFILLTUNING to characterize the adjustments needed for the scheduler when using this feature. Note: The use of this feature substantially limits of the overall USB performance that can be achieved.
3	**SLOM**	RW	0x0	Setup Lockout Mode. In device mode, this bit controls behavior of the setup lock mechanism. 0 = Setup Lockouts On (default). 1 = Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in USBCMD).

Table 8-68. HW_USBCTRL_USBMODE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**ES**	RW	0x0	Endian Select. This bit can change the byte ordering of the transfer buffers to match the host microprocessor bus architecture. The bit fields in the microprocessor interface and the DMA data structures (including the setup buffer within the device QH) are unaffected by the value of this bit, because they are based upon 32-bit words. 0 = Little Endian (default): First byte referenced in least significant byte of 32-bit word. 1 = Big Endian: First byte referenced in most significant byte of 32-bit word.
1:0	**CM**	RW	0x0	Controller Mode. Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability, the controller will default to an idle state and will need to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register. IDLE = 0x0 IDLE. DEVICE = 0x2 DEVICE. HOST = 0x3 HOST.

DESCRIPTION:

device mode

EXAMPLE:

```
Empty Example.
```

8.6.35 Endpoint Setup Status Register Description

HW_USBCTRL_ENDPTSETUPSTAT 0x1ac

Table 8-69. HW_USBCTRL_ENDPTSETUPSTAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																											ENDPTSETUPSTAT				

Table 8-70. HW_USBCTRL_ENDPTSETUPSTAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSVD**	RO	0x0	Reserved.
4:0	**ENDPTSETUPSTAT**	RW	0x0	Setup Endpoint Status. For every setup transaction that is received, a corresponding bit in this register is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to a respective bit after it has read the setup data from Queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lock out mechanism is engaged. This register is only used in device mode.

DESCRIPTION:

endpoint setup status

EXAMPLE:

```
Empty Example.
```

8.6.36 Endpoint Prime Register Description

This register is used in device mode only.

HW_USBCTRL_ENDPTPRIME 0x1b0

Table 8-71. HW_USBCTRL_ENDPTPRIME

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1											PETB					RSVD0											PERB				

Table 8-72. HW_USBCTRL_ENDPTPRIME Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	**RSVD1**	RO	0x0	Reserved.
20:16	**PETB**	RW	0x0	Prime Endpoint Transmit Buffer. For each endpoint, a corresponding bit is used to request that a buffer be prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a 1 to the corresponding bit when posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed. Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PETB[4] = Endpoint 4. PETB[3] = Endpoint 3. PETB[2] = Endpoint 2. PETB[1] = Endpoint 1. PETB[0] = Endpoint 0.
15:5	**RSVD0**	RO	0x0	Reserved.
4:0	**PERB**	RW	0x0	Prime Endpoint Receive Buffer. For each endpoint, a corresponding bit is used to request a buffer be prepared for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a 1 to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed. Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PERB[4] = Endpoint 4. PERB[3] = Endpoint 3. PERB[2] = Endpoint 2. PERB[1] = Endpoint 1. PERB[0] = Endpoint 0.

DESCRIPTION:

endpoint prime request

EXAMPLE:

```
Empty Example.
```

8.6.37 Endpoint Flush Register Description

This register is used in device-mode only.

HW_USBCTRL_ENDPTFLUSH 0x1b4

Table 8-73. HW_USBCTRL_ENDPTFLUSH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1											FETB					RSVD0											FERB				

Table 8-74. HW_USBCTRL_ENDPTFLUSH Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	**RSVD1**	RO	0x0	Reserved.
20:16	**FETB**	RW	0x0	Flush Endpoint Transmit Buffer. Writing a 1 to a bit(s) in this register will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for 1 of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful. FETB[4] = Endpoint 4. FETB[3] = Endpoint 3. FETB[2] = Endpoint 2. FETB[1] = Endpoint 1. FETB[0] = Endpoint 0.
15:5	**RSVD0**	RO	0x0	Reserved.
4:0	**FERB**	RW	0x0	Flush Endpoint Receive Buffer. Writing a 1 to a bit(s) will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful. FERB[4] = Endpoint 4. FERB[3] = Endpoint 3. FERB[2] = Endpoint 2. FERB[1] = Endpoint 1. FERB[0] = Endpoint 0.

DESCRIPTION:

endpoint flush request

EXAMPLE:

```
Empty Example.
```

8.6.38 Endpoint Status Register Description

This register is used in device mode only.

HW_USBCTRL_ENDPTSTAT 0x1b8

Table 8-75. HW_USBCTRL_ENDPTSTAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1											ETBR					RSVD0											ERBR				

Table 8-76. HW_USBCTRL_ENDPTSTAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	**RSVD1**	RO	0x0	Reserved.
20:16	**ETBR**	RO	0x0	Endpoint Transmit Buffer Ready. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a 1 by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ETBR[4] = Endpoint 4. ETBR[3] = Endpoint 3. ETBR[2] = Endpoint 2. ETBR[1] = Endpoint 1. ETBR[0] = Endpoint 0.
15:5	**RSVD0**	RO	0x0	Reserved.
4:0	**ERBR**	RO	0x0	Endpoint Receive Buffer Ready. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a 1 by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ERBR[4] = Endpoint 4. ERBR[3] = Endpoint 3. ERBR[2] = Endpoint 2. ERBR[1] = Endpoint 1. ERBR[0] = Endpoint 0.

DESCRIPTION:

endpoint ready

EXAMPLE:

```
Empty Example.
```

8.6.39 Endpoint Complete Register Description

This register is used in device-mode only.

HW_USBCTRL_ENDPTCOMPLETE 0x1bc

Table 8-77. HW_USBCTRL_ENDPTCOMPLETE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1											ETCE					RSVD0												ERCE			

Table 8-78. HW_USBCTRL_ENDPTCOMPLETE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	**RSVD1**	RO	0x0	Reserved.
20:16	**ETCE**	RW	0x0	Endpoint Transmit Complete Event. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a 1 will clear the corresponding bit in this register. ETCE[4] = Endpoint 4. ETCE[3] = Endpoint 3. ETCE[2] = Endpoint 2. ETCE[1] = Endpoint 1. ETCE[0] = Endpoint 0.
15:5	**RSVD0**	RO	0x0	Reserved.
4:0	**ERCE**	RW	0x0	Endpoint Receive Complete Event. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a 1 will clear the corresponding bit in this register. ERCE[4] = Endpoint 4. ERCE[3] = Endpoint 3. ERCE[2] = Endpoint 2. ERCE[1] = Endpoint 1. ERCE[0] = Endpoint 0.

DESCRIPTION:

endpoint complete

EXAMPLE:

```
Empty Example.
```

8.6.40 Endpoint Control 0 Register Description

Every Device will implement Endpoint0 as a control endpoint. The default value of this register is 0x00800080.

HW_USBCTRL_ENDPTCTRL0 0x1c0

Table 8-79. HW_USBCTRL_ENDPTCTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD6								TXE	RSVD5			TXT		RSVD4	TXS	RSVD3								RXE	RSVD2			RXT		RSVD1	RXS

Table 8-80. HW_USBCTRL_ENDPTCTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD6**	RO	0x0	Reserved.
23	**TXE**	RW	0x1	TX Endpoint Enable. 1 = Enabled. Endpoint0 is always enabled.
22:20	**RSVD5**	RO	0x0	Reserved. Bit reserved and should be read as zeroes.
19:18	**TXT**	RW	0x0	TX Endpoint Transmit Type. Endpoint0 is fixed as a Control endpoint. CONTROL = 0 Control.
17	**RSVD4**	RO	0x0	Reserved.
16	**TXS**	RW	0x0	Endpoint Stall. 0 = Endpoint OK (default). 1 = Endpoint Stalled. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Note: There is a slight delay (50 clocks max.) between the ENDPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, then follow this procedure: Continually write this stall bit until it is set OR until a new SETUP has been received by checking the associated ENDPTSETUPSTAT bit.

Table 8-80. HW_USBCTRL_ENDPTCTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:8	**RSVD3**	RO	0x0	Reserved. Bit reserved and should be read as zeroes.
7	**RXE**	RW	0x1	RX Endpoint Enable. 1 = Enabled. Endpoint0 is always enabled.
6:4	**RSVD2**	RO	0x0	Reserved. Bit reserved and should be read as zeroes.
3:2	**RXT**	RW	0x0	RX Endpoint Receive Type. Endpoint0 is fixed as a Control endpoint. CONTROL = 0 Control.
1	**RSVD1**	RO	0x0	Reserved.
0	**RXS**	RW	0x0	RX Endpoint Stall. 0 = Endpoint OK (default). 1 = Endpoint Stalled. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Note: There is a slight delay (50 clocks max.) between the ENDPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, then follow this procedure: Continually write this stall bit until it is set OR until a new SETUP has been received by checking the associated ENDPTSETUPSTAT bit.

DESCRIPTION:

endpoint control registers [0-n]

EXAMPLE:

```
Empty Example.
```

8.6.41 Endpoint Control 1 Register Description

Register HW_USBCTRL_ENDPTCTRL1 is the control register for endpoint 1 in a device. CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (i.e., bulk type). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

HW_USBCTRL_ENDPTCTRL1 0x1c4

Table 8-81. HW_USBCTRL_ENDPTCTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD6								TXE	TXR	TXI	RSVD5	TXT		TXD	TXS	RSVD3								RXE	RXR	RXI	RSVD2	RXT		RXD	RXS

Table 8-82. HW_USBCTRL_ENDPTCTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD6**	RO	0x0	Reserved.
23	**TXE**	RW	0x0	TX Endpoint Enable. 0 = Disabled (default). 1 = Enabled. An endpoint should be enabled only after it has been configured.
22	**TXR**	RW	0x0	TX Data Toggle Reset. Write 1 to reset PID sequence. Whenever a configuration event is received for this endpoint, software must write a 1 to this bit in order to synchronize the data PIDs between the host and device.
21	**TXI**	RW	0x0	TX Data Toggle Inhibit. 0 = PID Sequencing Enabled (default). 1 = PID Sequencing Disabled. This bit is only used for test and should always be written as 0. Writing a 1 to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20	**RSVD5**	RO	0x0	Reserved.
19:18	**TXT**	RW	0x0	TX Endpoint Transmit Type. CONTROL = 0 Control. ISO = 1 Isochronous. BULK = 2 Bulk. INT = 3 Interrupt.
17	**TXD**	RW	0x0	TX Endpoint Data Source. 0 = Dual Port Memory Buffer/DMA Engine (default). Should always be written as 0.

Table 8-82. HW_USBCTRL_ENDPTCTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
16	**TXS**	RW	0x0	Endpoint Stall. 0 = Endpoint OK. 1 = Endpoint Stalled. This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. Note (control endpoint types only): There is a slight delay (50 clocks max.) between the ENDPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, Should the DCD observe that the stall bit is not set after writing a 1 to it, then follow this procedure: Continually write this stall bit until it is set OR until a new SETUP has been received by checking the associated ENDPTSETUPSTAT bit.
15:8	**RSVD3**	RO	0x0	Reserved.
7	**RXE**	RW	0x0	RX Endpoint Enable. 0 = Disabled (default). 1 = Enabled. An Endpoint should be enabled only after it has been configured.
6	**RXR**	RW	0x0	Data Toggle Reset. Write 1 to reset PID Sequence. Whenever a configuration event is received for this endpoint, software must write a 1 to this bit in order to synchronize the data PIDs between the host and device.
5	**RXI**	RW	0x0	RX Data Toggle Inhibit. 0 = Disabled (default). 1 = Enabled. This bit is only used for test and should always be written as 0. Writing a 1 to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4	**RSVD2**	RO	0x0	Reserved.
3:2	**RXT**	RW	0x0	RX Endpoint Receive Type. CONTROL = 0 Control. ISO = 1 Isochronous. BULK = 2 Bulk. INT = 3 Interrupt.

Table 8-82. HW_USBCTRL_ENDPTCTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**RXD**	RW	0x0	RX Endpoint Data Sink. 0 = Dual Port Memory Buffer/DMA Engine (default). Should always be written as 0.
0	**RXS**	RW	0x0	RX Endpoint Stall. 0 = Endpoint OK (default). 1 = Endpoint Stalled. This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. Note (control endpoint types only): There is a slight delay (50 clocks maximum) between the ENDPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, then follow this procedure: Continually write this stall bit until it is set OR until a new SETUP has been received by checking the associated ENDPTSETUPSTAT bit.

DESCRIPTION:

endpoint control [0-n]

EXAMPLE:

```
Empty Example.
```

8.6.42 Endpoint Control 2 Register Description

Register HW_USBCTRL_ENDPTCTRL2 is the control register for endpoint 2 in a device. See the bit field defintions and descriptions of register HW_USBCTRL_ENDPTCTRL1. CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (i.e., bulk type). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

HW_USBCTRL_ENDPTCTRL2 0x1c8

Table 8-83. HW_USBCTRL_ENDPTCTRL2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD6								TXE	TXR	TXI	RSVD5	TXT		TXD	TXS	RSVD3								RXE	RXR	RXI	RSVD2	RXT		RXD	RXS

Table 8-84. HW_USBCTRL_ENDPTCTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD6**	RO	0x0	
23	**TXE**	RW	0x0	
22	**TXR**	RW	0x0	
21	**TXI**	RW	0x0	
20	**RSVD5**	RO	0x0	
19:18	**TXT**	RW	0x0	CONTROL = 0 Control ISO = 1 Isochronous BULK = 2 Bulk INT = 3 Int
17	**TXD**	RW	0x0	
16	**TXS**	RW	0x0	
15:8	**RSVD3**	RO	0x0	
7	**RXE**	RW	0x0	
6	**RXR**	RW	0x0	
5	**RXI**	RW	0x0	
4	**RSVD2**	RO	0x0	
3:2	**RXT**	RW	0x0	CONTROL = 0 Control ISO = 1 Isochronous BULK = 2 Bulk INT = 3 Int
1	**RXD**	RW	0x0	
0	**RXS**	RW	0x0	

EXAMPLE:

```
Empty Example.
```

8.6.43 Endpoint Control 3 Register Description

Register HW_USBCTRL_ENDPTCTRL3 is the control register for endpoint 3 in a device. See the bit field defintions and descriptions of register HW_USBCTRL_ENDPTCTRL1. CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (i.e., bulk type). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

HW_USBCTRL_ENDPTCTRL3 0x1cc

Table 8-85. HW_USBCTRL_ENDPTCTRL3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD6								TXE	TXR	TXI	RSVD5	TXT		TXD	TXS	RSVD3								RXE	RXR	RXI	RSVD2	RXT		RXD	RXS

Table 8-86. HW_USBCTRL_ENDPTCTRL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD6**	RO	0x0	
23	**TXE**	RW	0x0	
22	**TXR**	RW	0x0	
21	**TXI**	RW	0x0	
20	**RSVD5**	RO	0x0	

Table 8-86. HW_USBCTRL_ENDPTCTRL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:18	**TXT**	RW	0x0	CONTROL = 0 Control ISO = 1 Isochronous BULK = 2 Bulk INT = 3 Int
17	**TXD**	RW	0x0	
16	**TXS**	RW	0x0	
15:8	**RSVD3**	RO	0x0	
7	**RXE**	RW	0x0	
6	**RXR**	RW	0x0	
5	**RXI**	RW	0x0	
4	**RSVD2**	RO	0x0	
3:2	**RXT**	RW	0x0	CONTROL = 0 Control ISO = 1 Isochronous BULK = 2 Bulk INT = 3 Int
1	**RXD**	RW	0x0	
0	**RXS**	RW	0x0	

EXAMPLE:

```
Empty Example.
```

8.6.44 Endpoint Control 4 Register Description

Register HW_USBCTRL_ENDPTCTRL4 is the control register for endpoint 4 in a device. See the bit field defintions and descriptions of register HW_USBCTRL_ENDPTCTRL1. CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (i.e., bulk type). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

HW_USBCTRL_ENDPTCTRL4 0x1d0

Table 8-87. HW_USBCTRL_ENDPTCTRL4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD6								TXE	TXR	TXI	RSVD5	TXT		TXD	TXS	RSVD3								RXE	RXR	RXI	RSVD2	RXT		RXD	RXS

Table 8-88. HW_USBCTRL_ENDPTCTRL4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD6**	RO	0x0	
23	**TXE**	RW	0x0	
22	**TXR**	RW	0x0	
21	**TXI**	RW	0x0	
20	**RSVD5**	RO	0x0	
19:18	**TXT**	RW	0x0	CONTROL = 0 Control ISO = 1 Isochronous BULK = 2 Bulk INT = 3 Int
17	**TXD**	RW	0x0	
16	**TXS**	RW	0x0	

Table 8-88. HW_USBCTRL_ENDPTCTRL4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:8	**RSVD3**	RO	0x0	
7	**RXE**	RW	0x0	
6	**RXR**	RW	0x0	
5	**RXI**	RW	0x0	
4	**RSVD2**	RO	0x0	
3:2	**RXT**	RW	0x0	CONTROL = 0 Control ISO = 1 Isochronous BULK = 2 Bulk INT = 3 Int
1	**RXD**	RW	0x0	
0	**RXS**	RW	0x0	

EXAMPLE:

```
Empty Example.
```

USBCTRL Block v1.4, Revision 1.11

8.6.45

Chapter 9
Integrated USB 2.0 PHY

This chapter describes the integrated USB 2.0 full-speed and high-speed PHY available on the i.MX23. Programmable registers are described in Section 9.4, “Programmable Registers.”

9.1 Overview

The i.MX23 contains an integrated USB 2.0 PHY macrocell capable of connecting to PC host systems at the USB full-speed (FS) rate of 12 Mbits/s or at the USB 2.0 high-speed (HS) rate of 480 Mbits/s. See Figure 9-1 for a block diagram of the PHY. The integrated PHY provides a standard UTM interface. The USB_DP and USB_DN pins connect directly to a USB device connector.

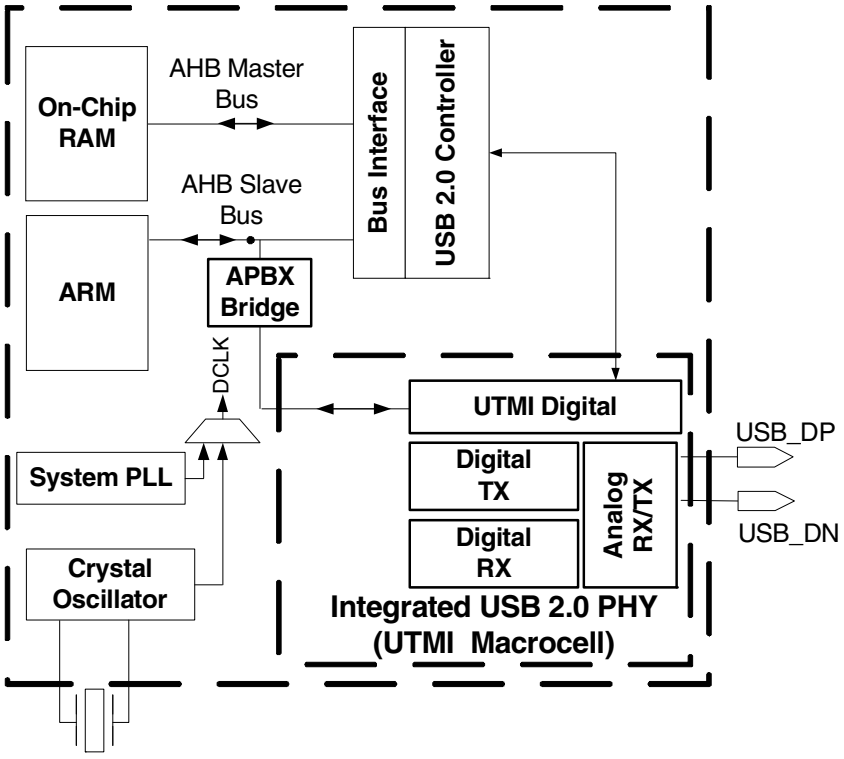


Figure 9-1. USB 2.0 PHY Block Diagram

The following subsections describe the external interfaces, internal interfaces, major blocks, and programmable registers that comprise the integrated USB 2.0 PHY.

9.2 Operation

The UTM provides a 16-bit interface to the USB controller. This interface is clocked at 30 MHz.

- The digital portions of the USBPHY block include the UTMI, digital transmitter, digital receiver, and the programmable registers.
- The analog transceiver section comprises an analog receiver and an analog transmitter, as shown in Figure 9-2.

9.2.1 UTMI

The UTMI block handles the line_state bits, reset buffering, suspend distribution, transceiver speed selection, and transceiver termination selection. The PLL supplies a 120-MHz signal to all of the digital logic. The UTMI block does a final divide-by-four to develop the 30-MHz clock used in the interface.

9.2.2 Digital Transmitter

The digital transmitter receives the 16-bit transmit data from the USB controller and handles the tx_valid, tx_validh and tx_ready handshake. In addition, it contains the transmit serializer that converts the 16-bit parallel words at 30 MHz to a single bitstream at 480 Mbit for high-speed or 12 Mbit for full-speed. It does this while implementing the bit-stuffing algorithm and the NRZI encoder that are used to remove the DC component from the serial bitstream. The output of this encoder is sent to the full-speed (FS) or high-speed (HS) drivers in the analog transceiver section's transmitter block.

9.2.3 Digital Receiver

The digital receiver receives the raw serial bitstream from the full speed (FS) differential transceiver, and a 9X, 480-MHz sampled data from the high speed (HS) differential transceiver. As the phase of the USB host transmitter shifts relative to the local PLL, the receiver section's HS DLL tracks these changes to give a reliable sample of the incoming 480-Mbit/s bitstream. Since this sample point shifts relative to the PLL phase used by the digital logic, a rate-matching elastic buffer is provided to cross this clock domain boundary. Once the bitstream is in the local clock domain, an NRZI decoder and bit unstuffer restore the original payload data bitstream and pass it to a deserializer and holding register. The receive state machine handles the rx_valid, rx_validh, and handshake with the USB controller. The handshake is not interlocked, in that there is no rx_ready signal coming from the controller. The controller must take each 16-bit value as presented by the PHY. The receive state machine provides an rx_active signal to the controller that indicates when it is inside a valid packet (SYNC detected, etc.).

9.2.4 Analog Receiver

The analog receiver comprises five differential receivers, two single-ended receivers, and a 9X, 480-MHz HS data sampling module, as shown in Figure 9-2 and described further in this section.

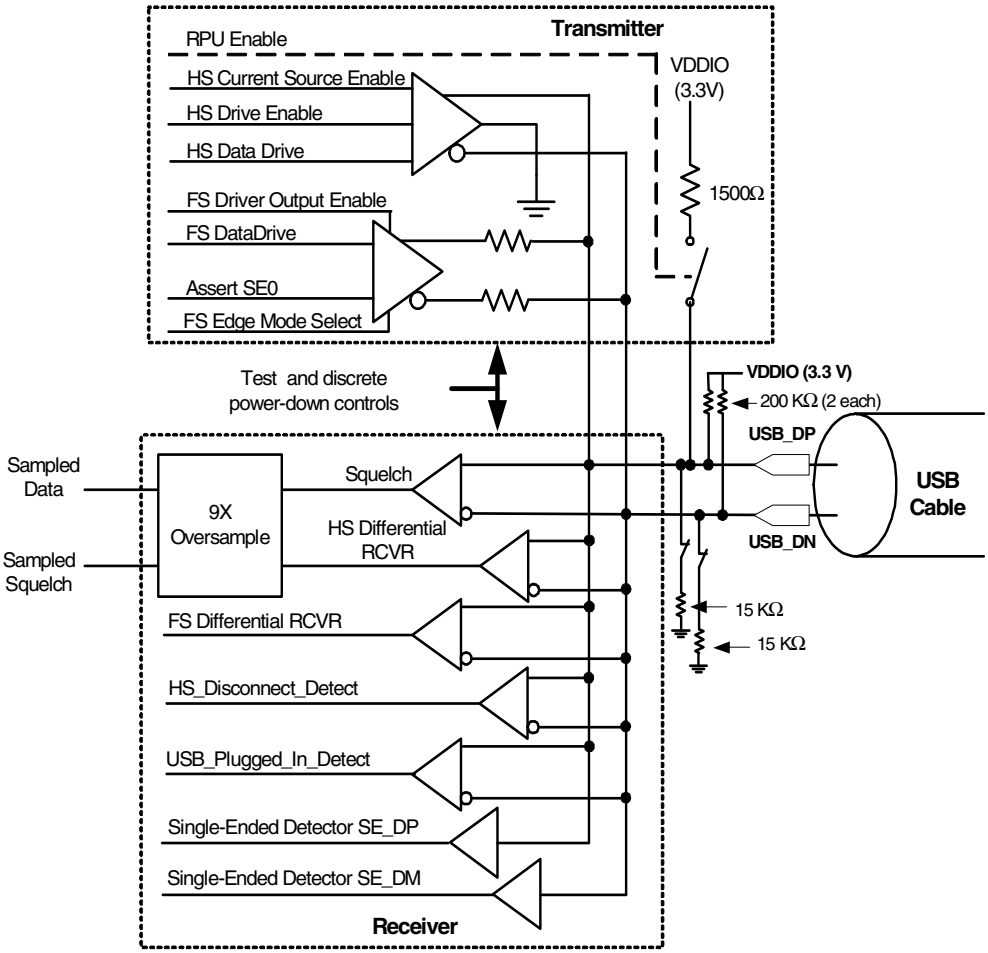


Figure 9-2. USB 2.0 PHY Analog Transceiver Block Diagram

9.2.4.1 HS Differential Receiver

The high-speed differential receiver is both a differential analog receiver and threshold comparator. Its output is a one if the differential signal is greater than a 0-V threshold. Its output is 0, otherwise. Its purpose is to discriminate the ± 400-mV differential voltage resulting from the high-speed drivers current flow into the dual 45Ω terminations found on each leg of the differential pair. The envelope or squelch detector, described below, ensures that the differential signal has sufficient magnitude to be valid. The HS differential receiver tolerates up to 500 mV of common mode offset.

9.2.4.2 Squelch Detector

The squelch detector is a differential analog receiver and threshold comparator. Its output is a 1 if the differential magnitude is less than a nominal 100-mV threshold. Its output is 0, otherwise. Its purpose is to invalidate the HS differential receiver when the incoming signal is simply too low to receive reliably.

9.2.4.3 FS Differential Receiver

The full-speed differential receiver is both a differential analog receiver and threshold comparator. The crossover voltage falls between 1.3 V and 2.0 V. Its output is a 1 when the USB_DP line is above the crossover point and the USB_DN line is below the crossover point.

9.2.4.4 HS Disconnect Detector

This host-side function is not used in i.MX23 applications, but is included to make a complete UTMI macrocell. It is a differential analog receiver and threshold comparator. Its output is a 1 if the differential magnitude is greater than a nominal 575-mV threshold. Its output is 0, otherwise.

9.2.4.5 USB Plugged-In Detector

The USB plugged-in detector looks for both USB_DP and USB_DN to be high. There is a pair of large on-chip pullup resistors (200KΩ) that hold both USB_DP and USB_DN high when the USB cable is not attached. The USB plugged-in detector signals a 0 in this case.

When in device mode, the host/hub interface that is *upstream* from the i.MX23 contains a 15KΩ pull-down resistor that easily overrides the 200KΩ pullup. When plugged in, at least one signal in the pair will be low, which will force the plugged-in detector's output high.

9.2.4.6 Single-Ended USB_DP Receiver

The single-ended USB_DP receiver output is high whenever the USB_DP input is above its nominal 1.8-V threshold.

9.2.4.7 Single-Ended USB_DN Receiver

The single-ended USB_DN receiver output is high whenever the USB_DN input is above its nominal 1.8-V threshold.

9.2.4.8 9X Oversample Module

The 9X oversample module uses nine identically spaced phases of the 480-MHz clock to sample a high speed bit data. The squelch signal is sampled only 1X.

9.2.5 Analog Transmitter

The analog transmitter comprises two differential drivers: one for high-speed signaling and one for full-speed signaling. It also contains the switchable 1.5KΩ pullup resistor. See Figure 9-2.

9.2.5.1 Switchable High-Speed 45Ω Termination Resistors

High-speed current mode differential signaling requires good 90Ω differential termination at each end of the USB cable. This results from switching in 45Ω terminating resistors from each signal line to ground at each end of the cable. Because each signal is parallel terminated with 45Ω at each end, each driver sees a 22.5Ω load. This is much too low of a load impedance for full-speed signaling levels—hence the need for switchable high-speed terminating resistors. Switchable trimming resistors are provided to tune the actual termination resistance of each device, as shown in Figure 9-3. The HW_USBPHY_TX_TXCAL45DP bit field, for example, allows one of 16 trimming resistor values to be placed in parallel with the 45Ω terminator on the USB_DP signal.

9.2.5.2 Full-Speed Differential Driver

The full-speed differential drivers are essentially “open drain” low-impedance pulldown devices that are switched in a differential mode for full-speed signaling, i.e., either one or the other device is turned on to signal the “J” state or the “K” state. These drivers are both turned on, simultaneously, for high-speed signaling. This has the effect of switching in both 45Ω terminating resistors. The tx_fs_hiz signal originates in the digital transmitter section. The hs_term signal that also controls these drivers comes from the UTMI.

9.2.5.3 High-Speed Differential Driver

The high-speed differential driver receives a 17.78-mA current from the constant current source and essentially steers it down either the USB_DP signal or the USB_DN signal or alternatively to ground. This current will produce approximately a 400-mV drop across the 22.5Ω termination seen by the driver when it is steered onto one of the signal lines. The approximately 17.78-mA current source is referenced back to the integrated voltage-band-gap circuit. The Iref, IBias, and V to I circuits are shared with the integrated battery charger.

9.2.5.4 Switchable 1.5KΩ USB_DP Pullup Resistor

The i.MX23 contains a switchable 1.5KΩ pullup resistor on the USB_DP signal. This resistor is switched on to tell the host/hub controller that a full-speed-capable device is on the USB cable, powered on, and ready. This resistor is switched off at power-on reset so the host does not recognize a USB device until processor software enables the announcement of a full-speed device.

9.2.5.5 Switchable 15KΩ USB_DP Pulldown Resistor

The i.MX23 contains a switchable 15KΩ pulldown resistor on both USB_DP and USB_DN signals. This is used in host mode to tell the device controller that a host is present.

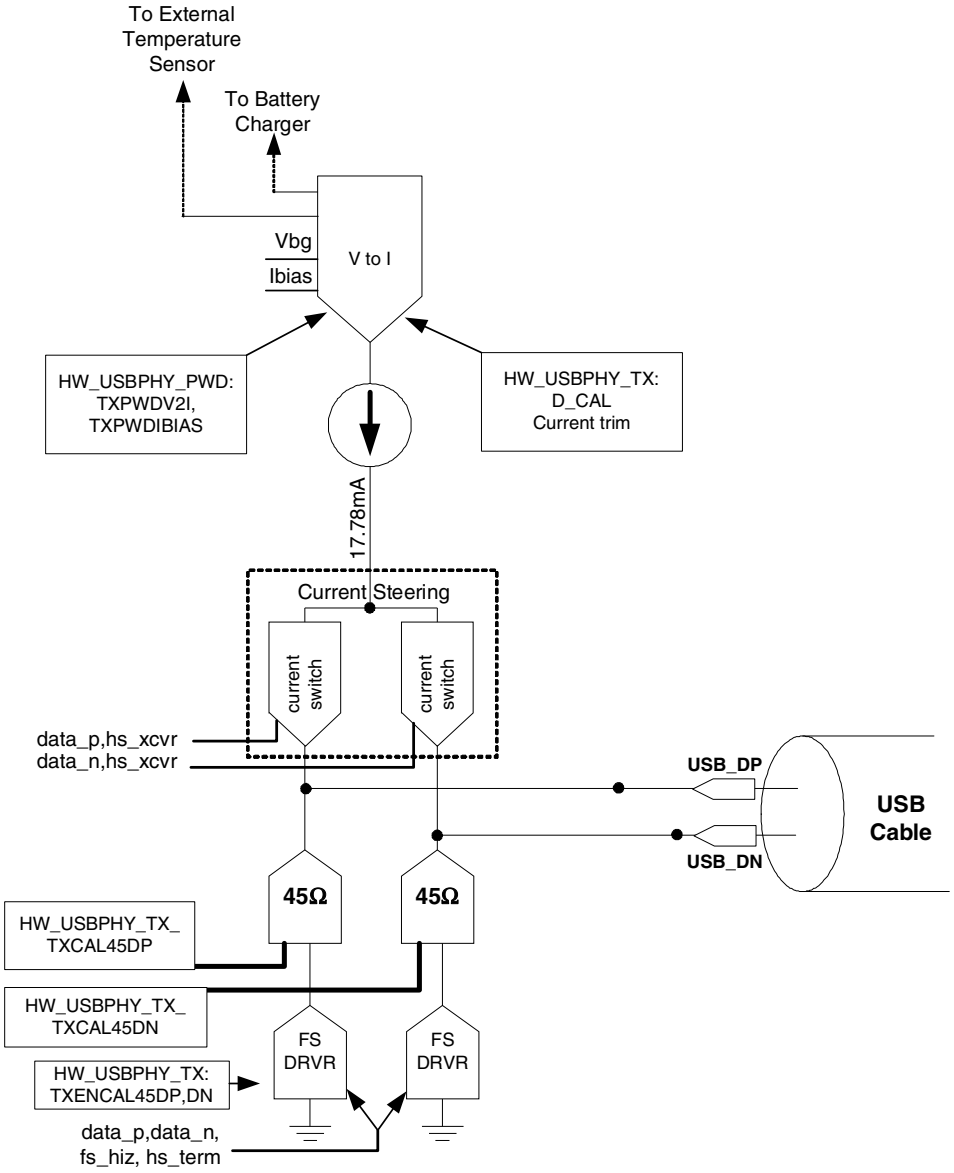


Figure 9-3. USB 2.0 PHY Transmitter Block Diagram

Table 9-1 summarizes the response of the PHY analog transmitter to various states of UTMI input and key transmit/receive state machine states.

Table 9-1. USB PHY Terminator States

UTMI OPMODE	UTM TERM	UTM XCVR	T/R	Function	45 Ω HIZ	1500 Ω HIZ	
00=Normal	0	0	X	HS	0	1	
	1	1	T	FS	0	0	
	1	1	R	FS	1	0	SUSPEND
	1	0	R	CHIRP	1	0	
	1	0	T	CHIRP	1	0	
	0	1	X	DISCONNECT	1	1	
01=NoDrive	0	0	T	HS	1	1	
	0	0	R	HS	1	1	
	1	1	X	FS	1	1	
	1	0	X	CHIRP	1	1	
	0	1	X	DISCONNECT	1	1	POR
10=NoNRZI NoBitStuff	0	0	X	HS	0	1	
	1	1	T	FS	0	0	
	1	1	R	FS	1	0	
	1	0	R	CHIRP	1	0	
	1	0	T	CHIRP	1	0	
	0	1	X	DISCONNECT	1	1	
11= Invalid	0	0	T	HS	1	1	
	0	0	R	HS	1	1	
	1	1	X	FS	1	1	
	1	0	X	CHIRP	1	1	
	0	1	X	DISCONNECT	1	1	

9.2.6 Recommended Register Configuration for USB Certification

The register settings in this section are recommended for passing USB certification.

The following settings lower the J/K levels to certifiable limits:

HW_USBPHY_TX_TXCAL45DP = 0x0
HW_USBPHY_TX_TXCAL45DN = 0x0
HW_USBPHY_TX_D_CAL = 0x7

The following settings help lower jitter in extreme conditions, for example, during heavy SDRAM usage with worst-case bit patterns:

HW_AUDIOOUT_REFCTRL_VDDXTAL_TO_VDDD = 0x1
HW_CLKCTRL_PLLCTRL0_CP_SEL = 0x2

Note that HW_AUDIOOUT_REFCTRL_VDDXTAL_TO_VDDD is controlled by the SFTRST and CLKGATE bits in the AUDIOIN block.

9.3 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 39.3.10, "Correct Way to Soft Reset a Block," for additional information on using the SFTRST and CLKGATE bit fields.

9.4 Programmable Registers

The USB 2.0 integrated PHY contains the following directly programmable registers.

9.4.1 USB PHY Power-Down Register Description

The USB PHY Power-Down Register provides overall control of the PHY power state.

HW_USBPHY_PWD	0x000
HW_USBPHY_PWD_SET	0x004
HW_USBPHY_PWD_CLR	0x008
HW_USBPHY_PWD_TOG	0x00c

Table 9-2. HW_USBPHY_PWD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2											RXPWDRX	RXPWDDIFF	RXPWD1PT1	RXPWDENV	RSVD1				TXPWDV2I	TXPWDIBIAS	TXPWDFS	RSVD0										

Table 9-3. HW_USBPHY_PWD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	**RSVD2**	RO	0x0	Reserved.
20	**RXPWDRX**	RW	0x1	0 = Normal operation. 1 = Power-down the entire USB PHY receiver block except for the full-speed differential receiver.
19	**RXPWDDIFF**	RW	0x1	0 = Normal operation. 1 = Power-down the USB high-speed differential receiver.
18	**RXPWD1PT1**	RW	0x1	0 = Normal operation. 1 = Power-down the USB full-speed differential receiver.
17	**RXPWDENV**	RW	0x1	0 = Normal operation. 1 = Power-down the USB high-speed receiver envelope detector (squelch signal).
16:13	**RSVD1**	RO	0x0	Reserved.

Table 9-3. HW_USBPHY_PWD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
12	**TXPWDV2I**	RW	0x1	0 = Normal operation. 1 = Power-down the USB PHY transmit V-to-I converter and the current mirror. Note that these circuits are shared with the battery charge circuit. Setting this to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down.
11	**TXPWDIBIAS**	RW	0x1	0 = Normal operation. 1 = Power-down the USB PHY current bias block for the transmitter. This bit should be set only when the USB is in suspend mode. This effectively powers down the entire USB transmit path. Note that these circuits are shared with the battery charge circuit. Setting this bit to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down.
10	**TXPWDFS**	RW	0x1	0 = Normal operation. 1 = Power-down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-impedance output.
9:0	**RSVD0**	RO	0x0	Reserved.

DESCRIPTION:

This register is used to control the USB PHY power state. See bits description for details.

EXAMPLE:

```
Empty Example.
```

9.4.2 USB PHY Transmitter Control Register Description

The USB PHY Transmitter Control Register handles the transmit controls.

HW_USBPHY_TX	0x010
HW_USBPHY_TX_SET	0x014
HW_USBPHY_TX_CLR	0x018
HW_USBPHY_TX_TOG	0x01c

Table 9-4. HW_USBPHY_TX

31 30 29	28 27 26	25	24	23 22	21	20	19 18 17 16	15 14	13	12	11 10 09 08	07 06 05 04	03 02 01 00
RSVD5	USBPHY_TX_EDGECTRL	USBPHY_TX_SYNC_INVERT	USBPHY_TX_SYNC_MUX	RSVD4	TXENCAL45DP	RSVD3	TXCAL45DP	RSVD2	TXENCAL45DN	RSVD1	TXCAL45DN	RSVD0	D_CAL

Table 9-5. HW_USBPHY_TX Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD5**	RO	0x0	Reserved.
28:26	**USBPHY_TX_EDGECTRL**	RW	0x4	Controls the edge-rate of the current sensing transistors used in HS transmit. NOT FOR CUSTOMER USE.
25	**USBPHY_TX_SYNC_INVERT**	RW	0x0	NOT FOR CUSTOMER USE. While in testmode enables clock jitter analysis by resyncing data to the USB_DP and USB_DM pins. 0 = no Sync, 1 = Sync. When EMI clock is ungated, the USB data is resynced with EMI clock (for EMI jitter analysis), otherwise the USB clock is used for resync.
24	**USBPHY_TX_SYNC_MUX**	RW	0x0	Enables multiplexer to synchronize data from the USB_DP and USB_DM pins 0 = No sync, 1 = Sync. NOT FOR CUSTOMER USE.
23:22	**RSVD4**	RO	0x0	Reserved.
21	**TXENCAL45DP**	RW	0x0	This bit is not used and must remain cleared.
20	**RSVD3**	RO	0x0	Reserved.
19:16	**TXCAL45DP**	RW	0x6	Decode to select a 45-Ohm resistance to the USB_DP output pin. Maximum resistance = 0000. Resistance is centered by design at 0110.
15:14	**RSVD2**	RO	0x0	Reserved.
13	**TXENCAL45DN**	RW	0x0	This bit is not used and must remain cleared.
12	**RSVD1**	RO	0x0	Reserved.
11:8	**TXCAL45DN**	RW	0x6	Decode to select a 45-Ohm resistance to the USB_DN output pin. Maximum resistance = 0000. Resistance is centered by design at 0110.
7:4	**RSVD0**	RO	0x0	Reserved.
3:0	**D_CAL**	RW	0x7	Resistor Trimming Code: 0000 = 0.16% 0111 = Nominal 1111 = +25%

DESCRIPTION:

This register is used to control several items related USB Phy transmitter.

EXAMPLE:

```
Empty Example.
```

9.4.3 USB PHY Receiver Control Register Description

The USB PHY Receiver Control Register handles receive path controls.

HW_USBPHY_RX	0x020
HW_USBPHY_RX_SET	0x024
HW_USBPHY_RX_CLR	0x028
HW_USBPHY_RX_TOG	0x02c

Table 9-6. HW_USBPHY_RX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2									RXDBYPASS	RSVD1															DISCONADJ			RSVD0	ENVADJ		

Table 9-7. HW_USBPHY_RX Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:23	**RSVD2**	RO	0x0	Reserved.
22	**RXDBYPASS**	RW	0x0	0 = Normal operation. 1 = Use the output of the USB_DP single-ended receiver in place of the full-speed differential receiver. This test mode is intended for lab use only.
21:7	**RSVD1**	RO	0x0	Reserved.
6:4	**DISCONADJ**	RW	0x0	The DISCONADJ field adjusts the trip point for the disconnect detector: 0000 = Trip-Level Voltage is 0.57500 V 0001 = Trip-Level Voltage is 0.56875 V 0010 = Trip-Level Voltage is 0.58125 V 0011 = Trip-Level Voltage is 0.58750 V 01XX = Reserved 1XXX = Reserved
3	**RSVD0**	RO	0x0	Reserved.
2:0	**ENVADJ**	RW	0x0	The ENVADJ field adjusts the trip point for the envelope detector. 0000 = Trip-Level Voltage is 0.12500 V 0001 = Trip-Level Voltage is 0.10000 V 0010 = Trip-Level Voltage is 0.13750 V 0011 = Trip-Level Voltage is 0.15000 V 01XX = Reserved 1XXX = Reserved

DESCRIPTION:

This register is used to control several items related USB Phy receiver

EXAMPLE:

```
Empty Example.
```

9.4.4 USB PHY General Control Register Description

The USB PHY General Control Register handles Host controls. This register also includes interrupt enables and connectivity detect enables and results.

HW_USBPHY_CTRL	0x030
HW_USBPHY_CTRL_SET	0x034
HW_USBPHY_CTRL_CLR	0x038
HW_USBPHY_CTRL_TOG	0x03c

Table 9-8. HW_USBPHY_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	UTMI_SUSPENDM	HOST_FORCE_LS_SE0	RSVD3														DATA_ON_LRADC	DEVPLUGIN_IRQ	ENIRQDEVPLUGIN	RESUME_IRQ	ENIRQRESUMEDETECT	RSVD2	ENOTGIDDETECT	RSVD1	DEVPLUGIN_POLARITY	ENDEVPLUGINDETECT	HOSTDISCONDETECT_IRQ	ENIRQHOSTDISCON	ENHOSTDISCONDETECT	RSVD0

Table 9-9. HW_USBPHY_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Writing a 1 to this bit will soft-reset the HW_USBPHY_PWD, HW_USBPHY_TX, HW_USBPHY_RX, and HW_USBPHY_CTRL registers.
30	**CLKGATE**	RW	0x1	Gate UTMI Clocks. Clear to 0 to run clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated.
29	**UTMI_SUSPENDM**	RO	0x0	Used by the PHY to indicate a powered-down state. If all the power-down bits in the HW_USBPHY_PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.
28	**HOST_FORCE_LS_SE0**	RW	0x0	Forces the next FS packet that is transmitted to have a EOP with low-speed timing. This bit is used in host mode for the resume sequence. After the packet is transferred, this bit is cleared. The design can use this function to force the LS SE0 or use the HW_USBPHY_CTRL_UTMI_SUSPENDM to trigger this event when leaving suspend. This bit is used in conjunction with HW_USBPHY_DEBUG_HOST_RESUME_DEBUG.
27:14	**RSVD3**	RO	0x0	Reserved.

Table 9-9. HW_USBPHY_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
13	**DATA_ON_LRADC**	RW	0x0	Enables the LRADC to monitor USB_DP and USB_DM. This is for use in non-USB modes only.
12	**DEVPLUGIN_IRQ**	RW	0x0	Indicates that the device is connected. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
11	**ENIRQDEVPLUGIN**	RW	0x0	Enables interrupt for the detection of connectivity to the USB line.
10	**RESUME_IRQ**	RW	0x0	Indicates that the host is sending a wake-up after suspend. This bit is also set on a reset during suspend. Use this bit to wake up from suspend for either the resume or the reset case. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
9	**ENIRQRESUMEDETECT**	RW	0x0	Enables interrupt for detection of a non-J state on the USB line. This should only be enabled after the device has entered suspend mode.
8	**RSVD2**	RO	0x0	Reserved.
7	**ENOTGIDDETECT**	RW	0x0	Enables circuit to detect resistance of MiniAB ID pin.
6	**RSVD1**	RO	0x0	Reserved.
5	**DEVPLUGIN_POLARITY**	RW	0x0	For device mode, if this bit is cleared to 0, then it trips the interrupt if the device is plugged in. If set to 1, then it trips the interrupt if the device is unplugged.
4	**ENDEVPLUGINDETECT**	RW	0x0	For device mode, enables 200-KOhm pullups for detecting connectivity to the host.
3	**HOSTDISCONDETECT_IRQ**	RW	0x0	Indicates that the device has disconnected in high-speed mode. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
2	**ENIRQHOSTDISCON**	RW	0x0	Enables interrupt for detection of disconnection to Device when in high-speed host mode. This should be enabled after ENDEVPLUGINDETECT is enabled.
1	**ENHOSTDISCONDETECT**	RW	0x0	For host mode, enables high-speed disconnect detector. This signal allows the override of enabling the detection that is normally done in the UTMI controller. The UTMI controller enables this circuit whenever the host sends a start-of-frame packet. Due to a on chip issue (Errata #2791), software must pay attention to when to assert the ENHOSTDISCONDETECT bit in HW_USBPHY_CTRL register: 1. Only set HW_USBPHY_CTRL.ENHOSTDISCONDETECT during high speed host mode. 2. Do not set HW_USBPHY_CTRL.ENHOSTDISCONDETECT during the reset and speed negotiation period. 3. Do not set HW_USBPHY_CTRL.ENHOSTDISCONDETECT during host suspend/resume sequence.
0	**RSVD0**	RO	0x0	Reserved.

DESCRIPTION:

This register is used to control high-level items related USB PHY.

EXAMPLE:

```
Empty Example.
```

9.4.5 USB PHY Status Register Description

The USB PHY Status Register holds results of IRQ and other detects.

HW_USBPHY_STATUS 0x040

Table 9-10. HW_USBPHY_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4																					RESUME_STATUS	RSVD3	OTGID_STATUS	RSVD2	DEVPLUGIN_STATUS	RSVD1		HOSTDISCONDETECT_STATUS	RSVD0		

Table 9-11. HW_USBPHY_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:11	**RSVD4**	RO	0x0	Reserved.
10	**RESUME_STATUS**	RO	0x0	Indicates that the host is sending a wake-up after suspend and has triggered an interrupt.
9	**RSVD3**	RO	0x0	Reserved.
8	**OTGID_STATUS**	RW	0x0	Indicates the results of ID pin on MiniAB plug. False (0) is when ID resistance is less than Ra_Plug_ID, indicating host (A) side. True (1) is when ID resistance is greater than Rb_Plug_ID, indicating device (B) side.
7	**RSVD2**	RO	0x0	Reserved.
6	**DEVPLUGIN_STATUS**	RO	0x0	Indicates that the device has been connected on the USB_DP and USB_DM lines.
5:4	**RSVD1**	RO	0x0	Reserved.
3	**HOSTDISCONDETECT_STATUS**	RO	0x0	Indicates that the device has disconnected while in high-speed host mode.
2:0	**RSVD0**	RO	0x0	Reserved.

DESCRIPTION:

This register is a status register and contains IRQ and other status.

EXAMPLE:

```
Empty Example.
```

9.4.6 USB PHY Debug Register Description

This register is used to debug the USB PHY.

HW_USBPHY_DEBUG	0x050
HW_USBPHY_DEBUG_SET	0x054
HW_USBPHY_DEBUG_CLR	0x058
HW_USBPHY_DEBUG_TOG	0x05c

Table 9-12. HW_USBPHY_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3	CLKGATE	HOST_RESUME_DEBUG	SQUELCHRESETLENGTH				ENSQUELCHRESET	RSVD2			SQUELCHRESETCOUNT					RSVD1			ENTX2RXCOUNT	TX2RXCOUNT			RSVD0		ENHSTPULLDOWN		HSTPULLDOWN		DEBUG_INTERFACE_HOLD	OTGIDPIOLOCK

Table 9-13. HW_USBPHY_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSVD3**	RO	0x0	Reserved.
30	**CLKGATE**	RW	0x1	Gate Test Clocks. Clear to 0 for running clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated.
29	**HOST_RESUME_DEBUG**	RW	0x1	Choose to trigger the host resume SE0 with HOST_FORCE_LS_SE0 = 0 or UTMI_SUSPEND = 1.
28:25	**SQUELCHRESETLENGTH**	RW	0xf	Duration of RESET in terms of the number of 480-MHz cycles.
24	**ENSQUELCHRESET**	RW	0x1	Set bit to allow squelch to reset high-speed receive.
23:21	**RSVD2**	RO	0x0	Reserved.
20:16	**SQUELCHRESETCOUNT**	RW	0x18	Delay in between the detection of squelch to the reset of high-speed RX.
15:13	**RSVD1**	RO	0x0	Reserved.
12	**ENTX2RXCOUNT**	RW	0x0	Set this bit to allow a countdown to transition in between TX and RX.
11:8	**TX2RXCOUNT**	RW	0x0	Delay in between the end of transmit to the beginning of receive. This is a Johnson count value and thus will count to 8.
7:6	**RSVD0**	RO	0x0	Reserved.
5:4	**ENHSTPULLDOWN**	RW	0x0	Set bit 5 to 1 to override the control of the USB_DP 15-KOhm pulldown. Set bit 4 to 1 to override the control of the USB_DM 15-KOhm pulldown. Clear to 0 to disable.

Table 9-13. HW_USBPHY_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3:2	**HSTPULLDOWN**	RW	0x0	Set bit 3 to 1 to pull down 15-KOhm on USB_DP line. Set bit 2 to 1 to pull down 15-KOhm on USB_DM line. Clear to 0 to disable.
1	**DEBUG_INTERFACE_HOLD**	RW	0x0	Use holding registers to assist in timing for external UTMI interface.
0	**OTGIDPIOLOCK**	RW	0x0	Once OTG ID is determined from HW_USBPHY_STATUS_OTGID_STATUS, use this to hold the value. This is to save power for the comparators that are used to determine the ID status.

DESCRIPTION:

Register not intended for customer use.

EXAMPLE:

```
Empty Example.
```

9.4.7 UTMI Debug Status Register 0 Description

The UTMI Debug Status Register 0 holds multiple views for counters and status of state machines. This is used in conjunction with the HW_USBPHY_DEBUG1.DBG_ADDRESS field to choose which function to view. The default is described in the bit fields below and is used to count errors.

HW_USBPHY_DEBUG0_STATUS 0x060

Table 9-14. HW_USBPHY_DEBUG0_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SQUELCH_COUNT						UTMI_RXERROR_FAIL_COUNT										LOOP_BACK_FAIL_COUNT																

Table 9-15. HW_USBPHY_DEBUG0_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**SQUELCH_COUNT**	RO	0x0	Running count of the squelch reset instead of normal end for HS RX.

Table 9-15. HW_USBPHY_DEBUG0_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25:16	**UTMI_RXERROR_FAIL_COUNT**	RO	0x0	Running count of the UTMI_RXERROR.
15:0	**LOOP_BACK_FAIL_COUNT**	RO	0x0	Running count of the failed pseudo-random generator loopback. Each time entering testmode, counter goes to "900d" and will count up for every detected packet failure in digital/analog loopback tests.

DESCRIPTION:

Register not intended for customer use.

EXAMPLE:

```
Empty Example.
```

9.4.8 UTMI Debug Status Register 1 Description

Chooses the muxing of the debug register to be shown in HW_USBPHY_DEBUG0_STATUS.

HW_USBPHY_DEBUG1	0x070
HW_USBPHY_DEBUG1_SET	0x074
HW_USBPHY_DEBUG1_CLR	0x078
HW_USBPHY_DEBUG1_TOG	0x07c

Table 9-16. HW_USBPHY_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1																	ENTAILADJVD		ENTX2TX	RSVD0								DBG_ADDRESS			

Table 9-17. HW_USBPHY_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:15	**RSVD1**	RO	0x0	Reserved.
14:13	**ENTAILADJVD**	RW	0x0	Delay increment of the rise of squelch: 00 = Delay is nominal 01 = Delay is +20% 10 = Delay is -20% 11 = Delay is -40%
12	**ENTX2TX**	RW	0x1	This bit has no function in this system.
11:4	**RSVD0**	RO	0x0	Reserved.
3:0	**DBG_ADDRESS**	RW	0x0	Chooses the multiplexing of the debug register to be shown in HW_USBPHY_DEBUG0_STATUS.

DESCRIPTION:

Register not intended for customer use.

EXAMPLE:

```
Empty Example.
```

9.4.9 UTMI RTL Version Description

Fields for RTL Version.

HW_USBPHY_VERSION	0x080

Table 9-18. HW_USBPHY_VERSION

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MAJOR								MINOR								STEP															

Table 9-19. HW_USBPHY_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**MAJOR**	RO	0x4	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	**MINOR**	RO	0x0	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	**STEP**	RO	0x0	Fixed read-only value reflecting the stepping of the RTL version.

DESCRIPTION:

This register indicates the RTL version in use.

EXAMPLE:

```
Empty Example.
```

9.4.10 USB PHY IP Block Register Description

The USB PHY IP Block Register IS FOR INTERNAL USE ONLY. It provides control of miscellaneous control bits found in other non-USB PIO control blocks that affects USB operations.

HW_USBPHY_IP	0x090
HW_USBPHY_IP_SET	0x094
HW_USBPHY_IP_CLR	0x098
HW_USBPHY_IP_TOG	0x09c

Table 9-20. HW_USBPHY_IP

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1							DIV_SEL		LFR_SEL		CP_SEL		TSTI_TX_DP	TSTI_TX_DM	ANALOG_TESTMODE	RSVD0													EN_USB_CLKS	PLL_LOCKED	PLL_POWER

Table 9-21. HW_USBPHY_IP Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD1**	RO	0x0	Reserved.
24:23	**DIV_SEL**	RW	0x0	TEST MODE FOR INTERNAL USE ONLY. This field is currently NOT supported. These bits came from the clkctrl PIO control block (clkctrl_pllctrl0_div_sel). DEFAULT = 0x0 PLL frequency is 480 MHz LOWER = 0x1 Lower the PLL fequency from 480MHz to 384MHz LOWEST = 0x2 Lower the PLL fequency from 480MHz to 288MHz UNDEFINED = 0x3 Undefined
22:21	**LFR_SEL**	RW	0x0	TEST MODE FOR INTERNAL USE ONLY. Adjusts loop filter resistor. These bits came from the clkctrl PIO control block (clkctrl_pllctrl0_lfr_sel). DEFAULT = 0x0 Default loop filter resistor TIMES_2 = 0x1 Doubles the loop filter resistor TIMES_05 = 0x2 Halves the loop filter resistor UNDEFINED = 0x3 Undefined
20:19	**CP_SEL**	RW	0x0	TEST MODE FOR INTERNAL USE ONLY. Adjusts charge pump current. These bits came from the clkctrl PIO control block (clkctrl_pllctrl0_cp_sel). DEFAULT = 0x0 Default charge pump current TIMES_2 = 0x1 Doubles charge pump current TIMES_05 = 0x2 Halves the charge pump current UNDEFINED = 0x3 Undefined
18	**TSTI_TX_DP**	RW	0x0	Analog testmode bit. Drives value on the DP pad. Default value is 1'b0. This bit came from the test control module.
17	**TSTI_TX_DM**	RW	0x0	Analog testmode bit. Drives value on the DM pad. Default value is 1'b0. This bit came from the test control module.
16	**ANALOG_TESTMODE**	RW	0x0	Analog testmode bit. Set to 0 for normal operation. Set to 1 for engineering debug of analog PHY block. This bit came from the test control module.
15:3	**RSVD0**	RO	0x0	Reserved.
2	**EN_USB_CLKS**	RW	0x0	If set to 0, 9-phase PLL outputs for USB PHY are powered down. If set to 1, 9-phase PLL outputs for USB PHY are powered up. Additionally, the UTMICLK120_GATE and UTMICLK30_GATE must be deasserted in the UTMI phy to enable USB operation. This bit came from the clkctrl PIO control block (clkctrl_pllctrl0_en_usb_clks).

Table 9-21. HW_USBPHY_IP Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**PLL_LOCKED**	RW	0x0	Software controlled bit to indicate when the USB PLL has locked. Software needs to wait 10 us after enabling the PLL POWER bit (0) before asserting this bit. If set to 0, tells the UTMI module that the USB PLL has not locked. If set to 1, tells the UTMI module that the USB PLL has locked. Software should clear this bit prior to turning off the USB PLL. This bit came from the clkctrl module.
0	**PLL_POWER**	RW	0x0	USB PLL Power On (0 = PLL off; 1 = PLL On). Allow 10 us after turning the PLL on before using the PLL as a clock source. This is the time the PLL takes to lock to 480 MHz. This bit came from the clkctrl PIO control block (clkctrl_pllctrl0_power).

DESCRIPTION:

This register contains control bits in other non AUSTIN USB applications.

EXAMPLE:

```
Empty Example.
```

USBPHY Block v4.0, Revision 1.52

9.4.11

Chapter 10
AHB-to-APBH Bridge with DMA

This chapter describes the AHB-to-APBH bridge on the i.MX23, along with its central DMA function and implementation examples. Programmable registers are described in Section 10.5, “Programmable Registers.”

10.1 Overview

The AHB-to-APBH bridge provides the i.MX23 with an inexpensive peripheral attachment bus running on the AHB’s HCLK. (The “H” in APBH denotes that the APB*H* is synchronous to HCLK, as compared to APB*X*, which runs on the crystal-derived XCLK.)

As shown in Figure 10-1, the AHB-to-APBH bridge includes the AHB-to-APB PIO bridge for memory-mapped I/O to the APB devices, as well as a central DMA facility for devices on this bus and a vectored interrupt controller for the ARM926 core. Each one of the APB peripherals, including the vectored interrupt controller, are documented in their own chapters elsewhere in this document.

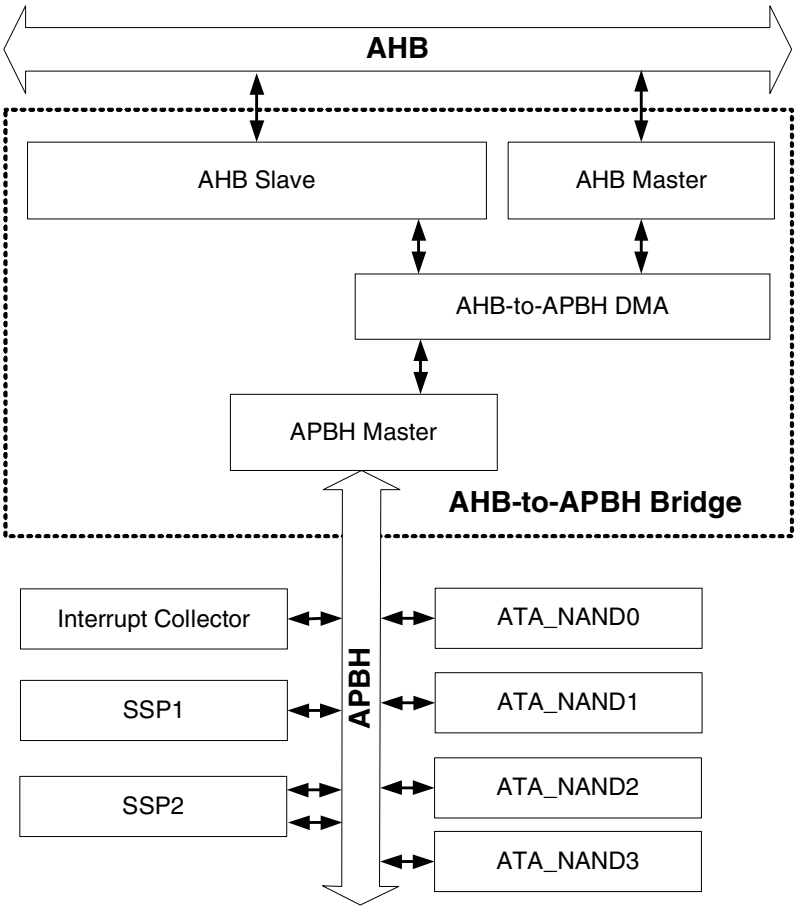


Figure 10-1. AHB-to-APBH Bridge DMA Block Diagram

The DMA controller uses the APBH bus to transfer read and write data to and from each peripheral. There is no separate DMA bus for these devices. Contention between the DMA's use of the APBH bus and the AHB-to-APB bridge functions' use of the APBH is mediated by internal arbitration logic. For contention between these two units, the DMA is favored and the AHB slave will report "not ready" via its HREADY output until the bridge transfer can complete. The arbiter tracks repeated lockouts and inverts the priority, guaranteeing the CPU every fourth transfer on the APB.

10.2 AHBH DMA

The DMA supports eight channels of DMA services, as shown in Table 10-1. The shared DMA resource allows each independent channel to follow a simple chained command list. Command chains are built up using the general structure, as shown in Figure 10-2.

Table 10-1. APBH DMA Channel Assignments

aPBH DMA Channel #	USAGE
0	Reserved
1	SSP1
2	SSP2
3	Reserved
4	NAND_DEVICE0
5	NAND_DEVICE1
6	NAND_DEVICE2
7	NAND_DEVICE3

A single command structure or channel command word specifies a number of operations to be performed by the DMA in support of a given device. Thus, the CPU can set up large units of work, chaining together many DMA channel command words, pass them off to the DMA, and have no further concern for the device until the DMA completion interrupt occurs. The goal here, as with the entire design of the i.MX23, is to have enough intelligence in the DMA and the devices to keep the interrupt frequency from any device below 1-kHz (arrival intervals longer than 1 ms).

Thus, a single command structure can issue 32-bit PIO write operations to key registers in the associated device using the same APB bus and controls that it uses to write DMA data bytes to the device. For example, this allows a chain of operations to be issued to the ATANAND controller to send NAND command bytes, address bytes, and data transfers where the command and address structure is completely under software control, but the administration of that transfer is handled autonomously by the DMA. Each DMA structure can have from 0 to 15 PIO words appended to it. The #PIOWORDs field, if non-zero, instructs the DMA engine to copy these words to the APB, beginning at PADDR = 0x0000 and incrementing its PADDR for each cycle.

The DMA master generates only normal read/write transfers to the APBH. It does *not* generate SCT set, clear, or toggle transfers.

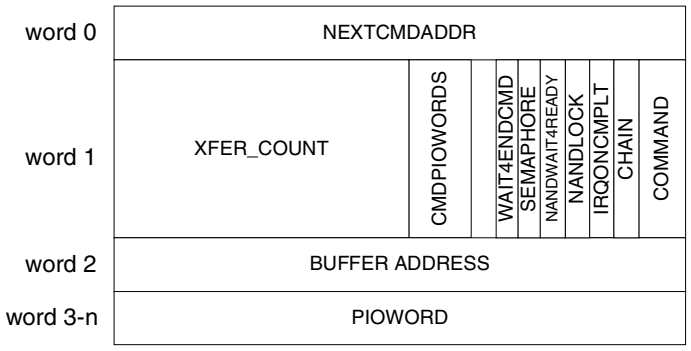


Figure 10-2. AHB-to-APBH Bridge DMA Channel Command Structure

Once any requested PIO words have been transferred to the peripheral, the DMA examines the two-bit command field in the channel command structure. Table 10-2 shows the four commands implemented by the DMA.

Table 10-2. APBH DMA Commands

DMA COMMAND	USAGE
00	NO_DMA_XFER. Perform any requested PIO word transfers, but terminate the command before any DMA transfer.
01	DMA_WRITE. Perform any requested PIO word transfers, then perform a DMA transfer from the peripheral for the specified number of bytes.
10	DMA_READ. Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.
11	DMA_SENSE. Perform any requested PIO word transfers, then perform a conditional branch to the next chained device. Follow the NEXTCMD_ADDR pointer if the peripheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true. This command becomes a no-operation for any channel other than a GPMI channel.

DMA_WRITE operations copy data bytes to system memory (on-chip RAM or SDRAM) from the associated peripheral. Each peripheral has a target PADDR value that it expects to receive DMA bytes. This association is synthesized in the DMA. The DMA_WRITE transfer uses the BUFFER_ADDDRESS word in the command structure to point to the beginning byte to write data from the peripheral.

DMA_READ operations copy data bytes to the APB peripheral from system memory. The DMA engine contains a shared byte aligner that aligns bytes from system memory to or from the peripherals. Peripherals always assume little-endian-aligned data arrives or departs on their 32-bit APB. The DMA_READ transfer uses the BUFFER_ADDRESS word in the command structure to point to the DMA data buffer to be read by the DMA_READ command.

The NO_DMA_XFER command is used to write PIO words to a device without performing any DMA data byte transfers. This command is useful in such applications as activating the NAND devices CHECKSTATUS operation. The check status command in the NAND peripheral reads a status byte from the NAND device, performs an XOR and MASK against an expected value supplied as part of the PIO transfer. Once the read check completes (see Section 10.3.1, “NAND Read Status Polling Example,”), the NO_DMA_XFER command completes. The result in the peripheral is that its PSENSE line is driven by the results of the comparison. The sense flip-flop is only updated by CHECKSTATUS for the device that is executed. At some future point, the chain contains a DMA command structure with the fourth and final command value, i.e., the DMA_SENSE command.

As each DMA command completes, it triggers the DMA to load the next DMA command structure in the chain. The normal flow list of DMA commands is found by following the NEXTCMD_ADDR pointer in the DMA command structure. The DMA_SENSE command uses the DMA buffer pointer word of the command structure in a slightly different way. Namely, it points to an alternate DMA command structure chain or list. The DMA_SENSE command examines the sense line of the associated peripheral. If the

sense line is “true,” then the DMA follows the standard list found whose next command is found from the pointer in the NEXTCMD_ADDR word of the command structure. If the sense line is “false,” then the DMA follows the alternate list whose next command is found from the pointer in the DMA Buffer Pointer word of the DMA_SENSE command structure (see Figure 10-3). The sense command ignores the CHAIN bit, so that both pointers must be valid when the DMA comes to sense command.

If the wait-for-end-command bit (WAIT4ENDCMD) is set in a command structure, then the DMA channel waits for the device to signal completion of a command by toggling the ENDCMCD signal before proceeding to load and execute the next command structure. The semaphore is decremented after the end command is seen.

A detailed bit-field view of the DMA command structure is shown in Table 10-3, which shows a field that specifies the number of bytes to be transferred by this DMA command. The transfer-count mechanism is duplicated in the associated peripheral, either as an implied or specified count in the peripheral.

Table 10-3. DMA Channel Command Word in System Memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															
Number DMA Bytes to Transfer																Number PIO Words to Write								WAIT4ENDCMD	DECREMENT SEMAPHORE	NANDwAIT4READY	NANDLOCK	IRQ_COMPLETE	CHAIN	COMMAND		
DMA Buffer or Alternate CCW																															
Zero or More PIO Words to Write to the Associated Peripheral Starting at its Base Address on the APBH Bus																															

Figure 10-3 also shows the CHAIN bit in bit 2 of the second word of the command structure. This bit is set to 1 if the NEXT_COMMAND_ADDRESS contains a pointer to another DMA command structure. If a null pointer (0) is loaded into the NEXT_COMMAND_ADDRESS, it will not be detected by the DMA hardware. Only the CHAIN bit indicates whether a valid list exists beyond the current structure.

If the IRQ_COMPLETE bit is set in the command structure, then the last act of the DMA before loading the next command is to set the interrupt status bit corresponding to the current channel. The sticky interrupt request bit in the DMA CSR remains set until cleared by software. It can be used to interrupt the CPU.

The NAND_LOCK bit is monitored by the DMA channel arbiter. Once a NAND channel ([7:4]) succeeds in the arbiter with its NAND_LOCK bit set, then the arbiter will ignore the other three NAND channels until a command is completed in which the NAND_LOCK is not set. Notice that the semantic here is that the NAND_LOCK state is to limit scheduling of a non-locked DMA. A DMA channel can go

from unlocked to locked in the arbiter at the beginning of a command when the NAND_LOCK bit is set. When the last DMA command of an atomic sequence is completed, the lock should be removed. To accomplish this, the last command does not have the NAND_LOCK bit. It is still locked in the atomic state within the arbiter when the command starts, so that it is the only NAND command that can be executed. At the end, it drops from the atomic state within the arbiter.

The NAND_WAIT4READY bit also has a special use for DMA channels [7:4], i.e., the NAND device channels. The NAND device supplies a sample of the ready line for the NAND device. This ready value is used to hold off of a command with this bit set until the ready line is asserted to 1. Once the arbiter sees a command with a wait-for-ready set, it holds off that channel until ready is asserted.

will continue without waiting for the interrupt.

Receiving an IRQ for HALTONTERMINATE (HOT) is a new feature in the APBH/X DMA descriptor that allows certain peripheral block (e.g. GPMI, SSP, I2C) to signal to the DMA engine that an error has occurred. In prior chips, if a block stalled due to an error, the only practical way to discover this in s/w was via a timer of some sort, or to poll the block. Now, an HOT signal is sent from the peripheral to the DMA engine and causes an IRQ after terminating the DMA descriptor being executed. Note not all peripheral block support this termination feature.

Therefore, it is recommended that software use this signal as follows:

- Always set HALTONTERMINATE to 1 in a DMA descriptor. That way, if a peripheral signals HOT, the transfer will end, leaving the peripheral block and the DMA engine synchronized (but at the end of a command).
- When an IRQ from an APBH/X channel is received, and the IRQ is determined to be due to an error (as opposed to an IRQONCOMPLETE interrupt) the software should:
  - — Reset the channel.
  - — Determine the error from error reporting in the peripheral block, then manage the error in the peripheral that is attached to that channel in whatever appropriate way exists for that device (software recovery, device reset, block reset, etc).

Each channel has an eight-bit counting semaphore that controls whether it is in the run or idle state. When the semaphore is non-zero, the channel is ready to run and process commands and DMA transfers. Whenever a command finishes its DMA transfer, it checks the DECREMENT_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 as a result, then the channel enters the IDLE state and remains there until the semaphore is incremented by software. When the semaphore goes to non-zero and the channel is in its IDLE state, then it uses the value in the HW_APBH_CHn_NXTCMDAR (next command address register) to fetch a pointer to the next command to process. NOTE: This is a double indirect case. This method allows software to append to a running command list under the protection of the counting semaphore.

To start processing the first time, software creates the command list to be processed. It writes the address of the first command into the HW_APBH_CHn_NXTCMDAR register, and then writes a 1 to the counting semaphore in HW_APBH_CHn_SEMA. The DMA channel loads HW_APBH_CHn_CURCMDAR

register and then enters the normal state machine processing for the next command. When software writes a value to the counting semaphore, it is added to the semaphore count by hardware, protecting the case where both hardware and software are trying to change the semaphore on the same clock edge.

Software can examine the value of HW_APBH_CHn_CURCMDAR at any time to determine the location of the command structure currently being processed.

10.3 Implementation Examples

10.3.1 NAND Read Status Polling Example

Figure 10-3 shows a more complicated scenario. This subset of a NAND device workload shows that the first two command structures are used during the data-write phase of an NAND device write operation (CLE and ALE transfers omitted for clarity).

- After writing the data, one must wait until the NAND device status register indicates that the write charge has been transferred. This is built into the workload using a check status command in the NAND in a loop created from the next two DMA command structures.
- The NO_DMA_TRANSFER command is shown here performing the read check, followed by a DMA_SENSE command to branch the DMA command structure list, based on the status of a bit in the external NAND device.

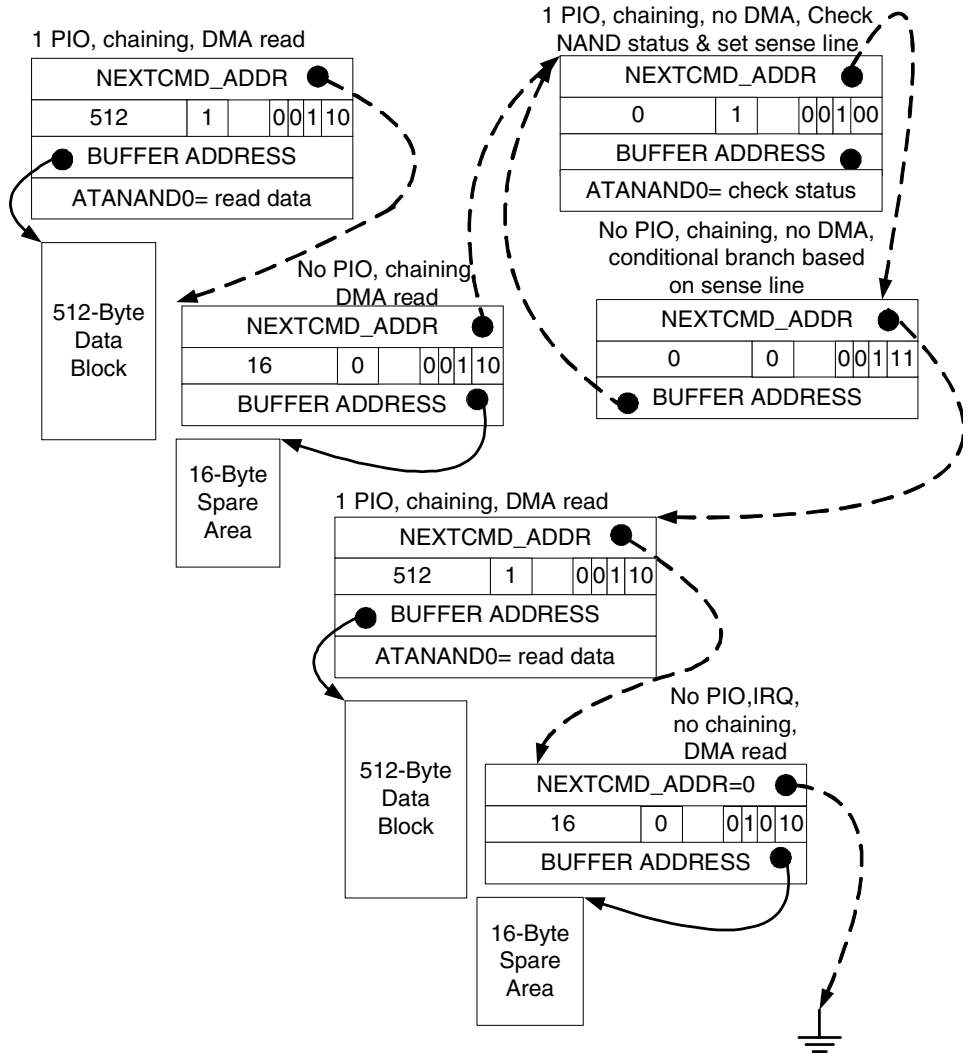


Figure 10-3. AHB-to-APBH Bridge DMA NAND Read Status Polling with DMA Sense Command

The example in Figure 10-3 shows the workload continuing immediately to the next NAND page transfer. However, one could perform a second sense operation to see if an error occurred after the write. One could then point the sense command alternate branch at a NO_DMA_XFER command with the interrupt bit set. If the CHAIN bit is not set on this failure branch, then the CPU is interrupted immediately, and the channel process is also immediately terminated in the presence of a workload-detected NAND error bit.

Note that each word of the three-word DMA command structure corresponds to a PIO register of the DMA that is accessible on the APBH bus. Normally, the DMA copies the next command structure onto these registers for processing at the start of each command by following the value of the pointer previously loaded into the NEXTCMD_ADDR register.

To start DMA processing for the first command, initialize the PIO registers of the desired channel, as follows:

- First, load the next command address register with a pointer to the first command to be loaded.
- Then, write a 1 to the counting semaphore register. This causes the DMA to schedule the targeted channel for DMA command structure load, as if it just finished its previous command.

10.4 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 39.3.10, "Correct Way to Soft Reset a Block," for additional information on using the SFTRST and CLKGATE bit fields.

10.5 Programmable Registers

This section describes the programmable registers of the AHB-to-APBH bridge block.

10.5.1 AHB to APBH Bridge Control and Status Register 0 Description

The APBH CTRL 0 provides overall control of the AHB to APBH bridge and DMA.

HW_APBH_CTRL0	0x000
HW_APBH_CTRL0_SET	0x004
HW_APBH_CTRL0_CLR	0x008
HW_APBH_CTRL0_TOG	0x00C

Table 10-4. HW_APBH_CTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	AHB_BURST8_EN	APB_BURST4_EN	RSVD0				RESET_CHANNEL								CLKGATE_CHANNEL								FREEZE_CHANNEL							

Table 10-5. HW_APBH_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Set this bit to zero to enable normal APBH DMA operation. Set this bit to one (default) to disable clocking with the APBH DMA and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the APBH DMA block to its default state.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29	**AHB_BURST8_EN**	RW	0x1	Set this bit to one (default) to enable AHB 8-beat burst. Set to zero to disable 8-beat burst on AHB interface.
28	**APB_BURST4_EN**	RW	0x0	Set this bit to one (default) to enable apb master do a 4 continous writes when a device request a burst dma. Set to zero will treat a burst dma request as 4 individual request.
27:24	**RSVD0**	RO	0x000000	Reserved, always set to zero.
23:16	**RESET_CHANNEL**	RW	0x0	Setting a bit in this field causes the DMA controller to take the corresponding channel through its reset state. The bit is reset after the channel resources are cleared. SSP1 = 0x02 SSP2 = 0x04 ATA = 0x10 NAND0 = 0x10 NAND1 = 0x20 NAND2 = 0x40 NAND3 = 0x80
15:8	**CLKGATE_CHANNEL**	RW	0x00	These bit must be set to zero for normal operation of each channel. When set to one they gate off the individual clocks to the channels. SSP1 = 0x02 SSP2 = 0x04 ATA = 0x10 NAND0 = 0x10 NAND1 = 0x20 NAND2 = 0x40 NAND3 = 0x80
7:0	**FREEZE_CHANNEL**	RW	0x0	Setting a bit in this field will freeze the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When frozen, the channel is denied access to the central DMA resources. SSP1 = 0x02 SSP2 = 0x04 ATA = 0x10 NAND0 = 0x10 NAND1 = 0x20 NAND2 = 0x40 NAND3 = 0x80

DESCRIPTION:

This register contains module softreset, clock gating, channel clock gating/freeze bits.

EXAMPLE:

```
Empty Example.
```

10.5.2 AHB to APBH Bridge Control and Status Register 1 Description

The APBH CTRL one provides overall control of the interrupts generated by the AHB to APBH DMA.

HW_APBH_CTRL1	0x010
HW_APBH_CTRL1_SET	0x014
HW_APBH_CTRL1_CLR	0x018
HW_APBH_CTRL1_TOG	0x01C

Table 10-6. HW_APBH_CTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1								CH7_CMDCMPLT_IRQ_EN	CH6_CMDCMPLT_IRQ_EN	CH5_CMDCMPLT_IRQ_EN	CH4_CMDCMPLT_IRQ_EN	CH3_CMDCMPLT_IRQ_EN	CH2_CMDCMPLT_IRQ_EN	CH1_CMDCMPLT_IRQ_EN	CH0_CMDCMPLT_IRQ_EN	RSVD0								CH7_CMDCMPLT_IRQ	CH6_CMDCMPLT_IRQ	CH5_CMDCMPLT_IRQ	CH4_CMDCMPLT_IRQ	CH3_CMDCMPLT_IRQ	CH2_CMDCMPLT_IRQ	CH1_CMDCMPLT_IRQ	CH0_CMDCMPLT_IRQ

Table 10-7. HW_APBH_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD1**	RO	0x00000000	Reserved, always set to zero.
23	**CH7_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA channel 7.
22	**CH6_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA channel 6.
21	**CH5_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA channel 5.
20	**CH4_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA channel 4.
19	**CH3_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA channel 3.
18	**CH2_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA channel 2.
17	**CH1_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA channel 1.
16	**CH0_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA channel 0.
15:8	**RSVD0**	RO	0x00000000	Reserved, always set to zero.
7	**CH7_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA channel 7. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
6	**CH6_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA channel 6. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
5	**CH5_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA channel 5. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

Table 10-7. HW_APBH_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4	**CH4_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA channel 4. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
3	**CH3_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA channel 3. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
2	**CH2_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA channel 2. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
1	**CH1_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA channel 1. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
0	**CH0_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA channel 0. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

DESCRIPTION:

This register contains the per channel interrupt status bits and the per channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE:

```
BF_WR(APBH_CTRL1, CH5_CMDCMPLT_IRQ, 0); // use bitfield write macro
BF_APBH_CTRL1.CH5_CMDCMPLT_IRQ = 0; // or, assign to register struct's bitfield
```

10.5.3 AHB to APBH Bridge Control and Status Register 2 Description

The APBH CTRL 2 provides channel error interrupts generated by the AHB to APBH DMA.

HW_APBH_CTRL2	0x020
HW_APBH_CTRL2_SET	0x024
HW_APBH_CTRL2_CLR	0x028
HW_APBH_CTRL2_TOG	0x02C

Table 10-8. HW_APBH_CTRL2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1								CH7_ERROR_STATUS	CH6_ERROR_STATUS	CH5_ERROR_STATUS	CH4_ERROR_STATUS	CH3_ERROR_STATUS	CH2_ERROR_STATUS	CH1_ERROR_STATUS	CH0_ERROR_STATUS	RSVD0								CH7_ERROR_IRQ	CH6_ERROR_IRQ	CH5_ERROR_IRQ	CH4_ERROR_IRQ	CH3_ERROR_IRQ	CH2_ERROR_IRQ	CH1_ERROR_IRQ	CH0_ERROR_IRQ

Table 10-9. HW_APBH_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD1**	RO	0x00000000	Reserved, always set to zero.
23	**CH7_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 7. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
22	**CH6_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 6. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
21	**CH5_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 5. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
20	**CH4_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 4. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
19	**CH3_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 3. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
18	**CH2_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 2. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
17	**CH1_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 1. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
16	**CH0_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 0. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
15:8	**RSVD0**	RO	0x00000000	Reserved, always set to zero.

Table 10-9. HW_APBH_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7	**CH7_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 7. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
6	**CH6_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 6. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
5	**CH5_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 5. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
4	**CH4_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 4. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
3	**CH3_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 3. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
2	**CH2_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 2. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
1	**CH1_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
0	**CH0_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

DESCRIPTION:

This register contains the per channel interrupt status bits and the per channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE:

```
BF_WR(APBH_CTRL1, CH5_CMDCMPLT_IRQ, 0); // use bitfield write macro
BF_APBH_CTRL1.CH5_CMDCMPLT_IRQ = 0; // or, assign to register struct's bitfield
```

10.5.4 AHB to APBH DMA Device Assignment Register Description

This register allows reassignment of the APBH device connected to the DMA Channels.

HW_APBH_DEVSEL 0x030

Table 10-10. HW_APBH_DEVSEL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CH7				CH6				CH5				CH4				CH3				CH2				CH1				CH0			

Table 10-11. HW_APBH_DEVSEL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**CH7**	RO	0x0	Reserved.
27:24	**CH6**	RO	0x0	Reserved.
23:20	**CH5**	RO	0x0	Reserved.
19:16	**CH4**	RO	0x0	Reserved.
15:12	**CH3**	RO	0x0	Reserved.
11:8	**CH2**	RO	0x0	Reserved.
7:4	**CH1**	RO	0x0	Reserved.
3:0	**CH0**	RO	0x0	Reserved.

DESCRIPTION:

This register contains channel mux sel bits. N/A for apbh bridge dma.

EXAMPLE:

```
Empty Example.
```

10.5.5 APBH DMA Channel 0 Current Command Address Register Description

The APBH DMA channel 0 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH0_CURCMDAR 0x040

Table 10-12. HW_APBH_CH0_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-13. HW_APBH_CH0_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 0.

DESCRIPTION:

APBH DMA Channel 0 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty Example.
```

10.5.6 APBH DMA Channel 0 Next Command Address Register Description

The APBH DMA Channel 0 Next Command Address register contains the address of the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to 1 in the DMA command word to process command lists.

HW_APBH_CH0_NXTCMDAR 0x050

Table 10-14. HW_APBH_CH0_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-15. HW_APBH_CH0_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 0.

DESCRIPTION:

APBH DMA Channel 0 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 0 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty Example.
```

10.5.7 APBH DMA Channel 0 Command Register Description

The APBH DMA Channel 0 command register specifies the DMA transaction to perform for the current command chain item.

HW_APBH_CH0_CMD 0x060

Table 10-16. HW_APBH_CH0_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND		

Table 10-17. HW_APBH_CH0_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the DMA device. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the DMA device, starting with the base PIO address of the DMA device register and incrementing from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before executing the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 10-17. HW_APBH_CH0_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH0_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers 10- Read transfer 11- SENSE NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty Example.
```

10.5.8 APBH DMA Channel 0 Buffer Address Register Description

The APBH DMA Channel 0 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBH_CH0_BAR 0x070

Table 10-18. HW_APBH_CH0_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 10-19. HW_APBH_CH0_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty Example.
```

10.5.9 APBH DMA Channel 0 Semaphore Register Description

The APBH DMA Channel 0 semaphore register is used to synchronize the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH0_SEMA 0x080

Table 10-20. HW_APBH_CH0_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 10-21. HW_APBH_CH0_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty Example.
```

10.5.10 AHB to APBH DMA Channel 0 Debug Information Description

This register gives debug visibility into the APBH DMA Channel 0 state machine and controls.

HW_APBH_CH0_DEBUG1 0x090

Table 10-22. HW_APBH_CH0_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE				

Table 10-23. HW_APBH_CH0_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27	**SENSE**	RO	0x0	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-7, this bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26	**READY**	RO	0x0	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-7, this bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25	**LOCK**	RO	0x0	This bit is reserved for this Channel and always reads 0. For Channels 4-7, this bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.

Table 10-23. HW_APBH_CH0_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 0 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 0.

EXAMPLE:

```
Empty example.
```

10.5.11 AHB to APBH DMA Channel 0 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 0.

HW_APBH_CH0_DEBUG2 0x0A0

Table 10-24. HW_APBH_CH0_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 10-25. HW_APBH_CH0_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 0.

EXAMPLE:

```
Empty example.
```

10.5.12 APBH DMA Channel 1 Current Command Address Register Description

The APBH DMA Channel 1 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH1_CURCMDAR 0x0B0

Table 10-26. HW_APBH_CH1_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-27. HW_APBH_CH1_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 1.

DESCRIPTION:

APBH DMA Channel 1 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBH_CHn_CURCMDAR_RD(1); // read the whole register, since there
is only one field
pCurCmd = (hw_apbh_chn_cmd_t *) BF_RDn(APBH_CHn_CURCMDAR, 1, CMD_ADDR); // or, use multi-register
bitfield read macro
pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBH_CHn_CURCMDAR(1).CMD_ADDR; // or, assign from bitfield of
indexed register's struct
```

10.5.13 APBH DMA Channel 1 Next Command Address Register Description

The APBH DMA Channel 1 Next Command Address register contains the address of the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to 1 in the DMA command word to process command lists.

HW_APBH_CH1_NXTCMDAR 0x0C0

Table 10-28. HW_APBH_CH1_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-29. HW_APBH_CH1_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 1.

DESCRIPTION:

APBH DMA Channel 1 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 1 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
HW_APBH_CHn_NXTCMDAR_WR(1, (reg32_t) pCommandTwoStructure); // write the entire register, since
there is only one field
BF_WRn(APBH_CHn_NXTCMDAR, 1, (reg32_t) pCommandTwoStructure); // or, use multi-register bitfield
write macro
HW_APBH_CHn_NXTCMDAR(1).CMD_ADDR = (reg32_t) pCommandTwoStructure; // or, assign to bitfield of
indexed register's struct
```

10.5.14 APBH DMA Channel 1 Command Register Description

The APBH DMA Channel 1 command register specifies the cycle to perform for the current command chain item.

HW_APBH_CH1_CMD 0x0D0

Table 10-30. HW_APBH_CH1_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 10-31. HW_APBH_CH1_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the SSP1 device. A value of 0 indicates a 64 KBytes trasnfer size.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the SSP1, starting with the base PIO address of the SSP1 control register and incrementing from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 10-31. HW_APBH_CH1_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH1_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, i.e. data sent from the SSP1 (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty Example.
```

10.5.15 APBH DMA Channel 1 Buffer Address Register Description

The APBH DMA Channel 1 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBH_CH1_BAR 0x0E0

Table 10-32. HW_APBH_CH1_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 10-33. HW_APBH_CH1_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
hw_apbh_chn_bar_t dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

10.5.16 APBH DMA Channel 1 Semaphore Register Description

The APBH DMA Channel 1 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH1_SEMA 0x0F0

Table 10-34. HW_APBH_CH1_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 10-35. HW_APBH_CH1_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
BF_WR(APBH_CHn_SEMA, 1, INCREMENT_SEMA, 2); // increment semaphore by two
current_sema = BF_RD(APBH_CHn_SEMA, 1, PHORE); // get instantaneous value
```

10.5.17 AHB to APBH DMA Channel 1 Debug Information Description

This register gives debug visibility into the APBH DMA Channel 1 state machine and controls.

HW_APBH_CH1_DEBUG1 0x100

Table 10-36. HW_APBH_CH1_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 10-37. HW_APBH_CH1_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27	**SENSE**	RO	0x0	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-7, this bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26	**READY**	RO	0x0	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-7, this bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25	**LOCK**	RO	0x0	This bit is reserved for this Channel and always reads 0. For Channels 4-7, this bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflect the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflect the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflect the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflect the current state of the DMA Channel's Write FIFO Empty signal.

Table 10-37. HW_APBH_CH1_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20	**WR_FIFO_FULL**	RO	0x0	This bit reflect the current state of the DMA Channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 1 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 1.

EXAMPLE:

```
Empty example.
```

10.5.18 AHB to APBH DMA Channel 1 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 1.

HW_APBH_CH1_DEBUG2 0x110

Table 10-38. HW_APBH_CH1_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 10-39. HW_APBH_CH1_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 1.

EXAMPLE:

```
Empty example.
```

10.5.19 APBH DMA Channel 2 Current Command Address Register Description

The APBH DMA Channel 2 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH2_CURCMDAR 0x120

Table 10-40. HW_APBH_CH2_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-41. HW_APBH_CH2_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 2.

DESCRIPTION:

APBH DMA Channel 2 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

10.5.20 APBH DMA Channel 2 Next Command Address Register Description

The APBH DMA Channel 2 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBH_CH2_NXTCMDAR 0x130

Table 10-42. HW_APBH_CH2_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-43. HW_APBH_CH2_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 2.

DESCRIPTION:

APBH DMA Channel 2 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 0 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty Example.
```

10.5.21 APBH DMA Channel 2 Command Register Description

The APBH DMA Channel 2 command register specifies the cycle to perform for the current command chain item.

HW_APBH_CH2_CMD 0x140

Table 10-44. HW_APBH_CH2_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 10-45. HW_APBH_CH2_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the SSP2 device. A value of 0 indicates a 64 KBytes transfer size.
15:12	**CMDWORDS**	RO	0x00	This field contains the number of command words to send to the SSP2, starting with the base PIO address of the SSP2 control register and incrementing from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 10-45. HW_APBH_CH2_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH2_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, i.e. data sent from the APBH Device (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

10.5.22 APBH DMA Channel 2 Buffer Address Register Description

The APBH DMA Channel 2 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBH_CH2_BAR 0x150

Table 10-46. HW_APBH_CH2_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 10-47. HW_APBH_CH2_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

10.5.23 APBH DMA Channel 2 Semaphore Register Description

The APBH DMA Channel 2 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH2_SEMA 0x160

Table 10-48. HW_APBH_CH2_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 10-49. HW_APBH_CH2_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

10.5.24 AHB to APBH DMA Channel 2 Debug Information Description

This register gives debug visibility into the APBH DMA Channel 2 state machine and controls.

HW_APBH_CH2_DEBUG1 0x170

Table 10-50. HW_APBH_CH2_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE				

Table 10-51. HW_APBH_CH2_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27	**SENSE**	RO	0x0	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-7, this bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26	**READY**	RO	0x0	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-7, this bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25	**LOCK**	RO	0x0	This bit is reserved for this Channel and always reads 0. For Channels 4-7, this bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflect the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflect the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflect the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflect the current state of the DMA Channel's Write FIFO Empty signal.

Table 10-51. HW_APBH_CH2_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20	**WR_FIFO_FULL**	RO	0x0	This bit reflect the current state of the DMA Channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 2 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 2.

EXAMPLE:

```
Empty example.
```

10.5.25 AHB to APBH DMA Channel 2 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 2.

HW_APBH_CH2_DEBUG2 0x180

Table 10-52. HW_APBH_CH2_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 10-53. HW_APBH_CH2_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 2.

EXAMPLE:

```
Empty example.
```

10.5.26 APBH DMA Channel 3 Current Command Address Register Description

The APBH DMA Channel 3 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH3_CURCMDAR 0x190

Table 10-54. HW_APBH_CH3_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-55. HW_APBH_CH3_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 3.

DESCRIPTION:

APBH DMA Channel 3 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

10.5.27 APBH DMA Channel 3 Next Command Address Register Description

The APBH DMA Channel 3 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBH_CH3_NXTCMDAR 0x1A0

Table 10-56. HW_APBH_CH3_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-57. HW_APBH_CH3_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 3.

DESCRIPTION:

APBH DMA Channel 3 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 3 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

10.5.28 APBH DMA Channel 3 Command Register Description

The APBH DMA Channel 3 command register specifies the cycle to perform for the current command chain item.

HW_APBH_CH3_CMD 0x1B0

Table 10-58. HW_APBH_CH3_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 10-59. HW_APBH_CH3_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the DMA device egister. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the DMA device, starting with the base PIO address of the DMA device and increment from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 10-59. HW_APBH_CH3_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH3_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, i.e. data sent from the APBH device (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

10.5.29 APBH DMA Channel 3 Buffer Address Register Description

The APBH DMA Channel 3 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBH_CH3_BAR 0x1C0

Table 10-60. HW_APBH_CH3_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 10-61. HW_APBH_CH3_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

10.5.30 APBH DMA Channel 3 Semaphore Register Description

The APBH DMA Channel 3 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH3_SEMA 0x1D0

Table 10-62. HW_APBH_CH3_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 10-63. HW_APBH_CH3_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

10.5.31 AHB to APBH DMA Channel 3 Debug Information Description

This register gives debug visibility into the APBH DMA Channel 3 state machine and controls.

HW_APBH_CH3_DEBUG1 0x1E0

Table 10-64. HW_APBH_CH3_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE				

Table 10-65. HW_APBH_CH3_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27	**SENSE**	RO	0x0	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-7, this bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26	**READY**	RO	0x0	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-7, this bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25	**LOCK**	RO	0x0	This bit is reserved for this Channel and always reads 0. For Channels 4-7, this bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflect the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x0	This bit reflect the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflect the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x0	This bit reflect the current state of the DMA Channel's Write FIFO Empty signal.

Table 10-65. HW_APBH_CH3_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20	**WR_FIFO_FULL**	RO	0x0	This bit reflect the current state of the DMA Channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 3 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 3.

EXAMPLE:

```
Empty example.
```

10.5.32 AHB to APBH DMA Channel 3 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 3.

HW_APBH_CH3_DEBUG2 0x1F0

Table 10-66. HW_APBH_CH3_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 10-67. HW_APBH_CH3_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 3.

EXAMPLE:

```
Empty example.
```

10.5.33 APBH DMA Channel 4 Current Command Address Register Description

The APBH DMA Channel 4 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH4_CURCMDAR 0x200

Table 10-68. HW_APBH_CH4_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-69. HW_APBH_CH4_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 4.

DESCRIPTION:

APBH DMA Channel 4 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

10.5.34 APBH DMA Channel 4 Next Command Address Register Description

The APBH DMA Channel 4 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBH_CH4_NXTCMDAR 0x210

Table 10-70. HW_APBH_CH4_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-71. HW_APBH_CH4_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 4.

DESCRIPTION:

APBH DMA Channel 4 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 4 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

10.5.35 APBH DMA Channel 4 Command Register Description

The APBH DMA Channel 4 command register specifies the cycle to perform for the current command chain item.

HW_APBH_CH4_CMD 0x220

Table 10-72. HW_APBH_CH4_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 10-73. HW_APBH_CH4_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI ATANAND_0 device HW_GPMI_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the GPMI, starting with the base PIO address of the GPMI (HW_GPMI_CTRL0) and increment from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 10-73. HW_APBH_CH4_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH4_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, i.e. data sent from the GPMI (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

10.5.36 APBH DMA Channel 4 Buffer Address Register Description

The APBH DMA Channel 1 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBH_CH4_BAR 0x230

Table 10-74. HW_APBH_CH4_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 10-75. HW_APBH_CH4_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

10.5.37 APBH DMA Channel 4 Semaphore Register Description

The APBH DMA Channel 4 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH4_SEMA 0x240

Table 10-76. HW_APBH_CH4_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 10-77. HW_APBH_CH4_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

10.5.38 AHB to APBH DMA Channel 4 Debug Information Description

This register gives debug visibility into the APBH DMA Channel 4 state machine and controls.

HW_APBH_CH4_DEBUG1 0x250

Table 10-78. HW_APBH_CH4_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE				

Table 10-79. HW_APBH_CH4_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27	**SENSE**	RO	0x0	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26	**READY**	RO	0x0	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25	**LOCK**	RO	0x0	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.

Table 10-79. HW_APBH_CH4_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 4 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts. WAIT_READY = 0x1F When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 4.

EXAMPLE:

```
Empty example.
```

10.5.39 AHB to APBH DMA Channel 4 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 4.

HW_APBH_CH4_DEBUG2 0x260

Table 10-80. HW_APBH_CH4_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 10-81. HW_APBH_CH4_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 4.

EXAMPLE:

```
Empty example.
```

10.5.40 APBH DMA Channel 5 Current Command Address Register Description

The APBH DMA Channel 5 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH5_CURCMDAR 0x270

Table 10-82. HW_APBH_CH5_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-83. HW_APBH_CH5_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 5.

DESCRIPTION:

APBH DMA Channel 5 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

10.5.41 APBH DMA Channel 5 Next Command Address Register Description

The APBH DMA Channel 5 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBH_CH5_NXTCMDAR 0x280

Table 10-84. HW_APBH_CH5_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-85. HW_APBH_CH5_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 5.

DESCRIPTION:

APBH DMA Channel 5 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 5 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

10.5.42 APBH DMA Channel 5 Command Register Description

The APBH DMA Channel 5 command register specifies the cycle to perform for the current command chain item.

HW_APBH_CH5_CMD 0x290

Table 10-86. HW_APBH_CH5_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 10-87. HW_APBH_CH5_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI ATANAND_1 device HW_GPMI_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the GPMI, starting with the base PIO address of the GPMI (HW_GPMI_CTRL0) and increment from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 10-87. HW_APBH_CH5_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH5_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, i.e. data sent from the GPMI (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

10.5.43 APBH DMA Channel 5 Buffer Address Register Description

The APBH DMA Channel 5 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBH_CH5_BAR 0x2A0

Table 10-88. HW_APBH_CH5_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 10-89. HW_APBH_CH5_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

10.5.44 APBH DMA Channel 5 Semaphore Register Description

The APBH DMA Channel 5 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH5_SEMA 0x2B0

Table 10-90. HW_APBH_CH5_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 10-91. HW_APBH_CH5_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

10.5.45 AHB to APBH DMA Channel 5 Debug Information Description

This register gives debug visibility into the APBH DMA Channel 5 state machine and controls.

HW_APBH_CH5_DEBUG1 0x2C0

Table 10-92. HW_APBH_CH5_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 10-93. HW_APBH_CH5_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27	**SENSE**	RO	0x0	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26	**READY**	RO	0x0	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25	**LOCK**	RO	0x0	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.

Table 10-93. HW_APBH_CH5_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 5 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts. WAIT_READY = 0x1F When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 5.

EXAMPLE:

```
Empty example.
```

10.5.46 AHB to APBH DMA Channel 5 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 5.

HW_APBH_CH5_DEBUG2 0x2D0

Table 10-94. HW_APBH_CH5_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 10-95. HW_APBH_CH5_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 5.

EXAMPLE:

```
Empty example.
```

10.5.47 APBH DMA Channel 6 Current Command Address Register Description

The APBH DMA Channel 6 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH6_CURCMDAR 0x2E0

Table 10-96. HW_APBH_CH6_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-97. HW_APBH_CH6_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 6.

DESCRIPTION:

APBH DMA Channel 6 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

10.5.48 APBH DMA Channel 6 Next Command Address Register Description

The APBH DMA Channel 6 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBH_CH6_NXTCMDAR 0x2F0

Table 10-98. HW_APBH_CH6_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-99. HW_APBH_CH6_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 6.

DESCRIPTION:

APBH DMA Channel 6 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 6 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

10.5.49 APBH DMA Channel 6 Command Register Description

The APBH DMA Channel 6 command register specifies the cycle to perform for the current command chain item.

HW_APBH_CH6_CMD 0x300

Table 10-100. HW_APBH_CH6_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 10-101. HW_APBH_CH6_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI ATANAND_2 device HW_GPMI_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the GPMI, starting with the base PIO address of the GPMI (HW_GPMI_CTRL0) and increment from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 10-101. HW_APBH_CH6_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH6_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, i.e. data sent from the GPMI (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

10.5.50 APBH DMA Channel 6 Buffer Address Register Description

The APBH DMA Channel 6 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBH_CH6_BAR 0x310

Table 10-102. HW_APBH_CH6_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 10-103. HW_APBH_CH6_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

10.5.51 APBH DMA Channel 6 Semaphore Register Description

The APBH DMA Channel 6 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH6_SEMA 0x320

Table 10-104. HW_APBH_CH6_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 10-105. HW_APBH_CH6_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

10.5.52 AHB to APBH DMA Channel 6 Debug Information Description

This register gives debug visibility into the APBH DMA Channel 6 state machine and controls.

HW_APBH_CH6_DEBUG1 0x330

Table 10-106. HW_APBH_CH6_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE				

Table 10-107. HW_APBH_CH6_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27	**SENSE**	RO	0x0	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26	**READY**	RO	0x0	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25	**LOCK**	RO	0x0	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.

Table 10-107. HW_APBH_CH6_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 6 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts. WAIT_READY = 0x1F When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 6.

EXAMPLE:

```
Empty example.
```

10.5.53 AHB to APBH DMA Channel 6 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 6.

HW_APBH_CH6_DEBUG2 0x340

Table 10-108. HW_APBH_CH6_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 10-109. HW_APBH_CH6_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 6.

EXAMPLE:

```
Empty example.
```

10.5.54 APBH DMA Channel 7 Current Command Address Register Description

The APBH DMA Channel 7 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH7_CURCMDAR 0x350

Table 10-110. HW_APBH_CH7_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-111. HW_APBH_CH7_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 7.

DESCRIPTION:

APBH DMA Channel 7 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

10.5.55 APBH DMA Channel 7 Next Command Address Register Description

The APBH DMA Channel 7 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBH_CH7_NXTCMDAR 0x360

Table 10-112. HW_APBH_CH7_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 10-113. HW_APBH_CH7_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 7.

DESCRIPTION:

APBH DMA Channel 7 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 7 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

10.5.56 APBH DMA Channel 7 Command Register Description

The APBH DMA Channel 7 command register specifies the cycle to perform for the current command chain item.

HW_APBH_CH7_CMD 0x370

Table 10-114. HW_APBH_CH7_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 10-115. HW_APBH_CH7_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI ATANAND_3 device HW_GPMI_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the GPMI, starting with the base PIO address of the GPMI (HW_GPMI_CTRL0) and increment from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 10-115. HW_APBH_CH7_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH7_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, i.e. data sent from the GPMI (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

10.5.57 APBH DMA Channel 7 Buffer Address Register Description

The APBH DMA Channel 7 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBH_CH7_BAR 0x380

Table 10-116. HW_APBH_CH7_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 10-117. HW_APBH_CH7_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

10.5.58 APBH DMA Channel 7 Semaphore Register Description

The APBH DMA Channel 7 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH7_SEMA 0x390

Table 10-118. HW_APBH_CH7_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 10-119. HW_APBH_CH7_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

10.5.59 AHB to APBH DMA Channel 7 Debug Information Description

This register gives debug visibility into the APBH DMA Channel 7 state machine and controls.

HW_APBH_CH7_DEBUG1 0x3A0

Table 10-120. HW_APBH_CH7_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 10-121. HW_APBH_CH7_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27	**SENSE**	RO	0x0	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26	**READY**	RO	0x0	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25	**LOCK**	RO	0x0	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.

Table 10-121. HW_APBH_CH7_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 7 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts. WAIT_READY = 0x1F When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 7.

EXAMPLE:

```
Empty example.
```

10.5.60 AHB to APBH DMA Channel 7 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 7.

HW_APBH_CH7_DEBUG2 0x3B0

Table 10-122. HW_APBH_CH7_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 10-123. HW_APBH_CH7_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 7.

EXAMPLE:

```
Empty example.
```

10.5.61 APBH Bridge Version Register Description

This register always returns a known read value for debug purposes it indicates the version of the block.

HW_APBH_VERSION 0x3F0

Table 10-124. HW_APBH_VERSION

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MAJOR								MINOR								STEP															

Table 10-125. HW_APBH_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**MAJOR**	RO	0x02	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	**MINOR**	RO	0x00	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	**STEP**	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

DESCRIPTION:

This register indicates the RTL version in use.

EXAMPLE:

```
if (HW_APBH_VERSION.B.MAJOR != 1)
Error();
```

APBH Block v2.0, Revision 1.57

Chapter 11
AHB-to-APBX Bridge with DMA

This chapter describes the AHB-to-APBX bridge on the i.MX23, along with its central DMA function and implementation examples. Programmable registers are described in Section 11.5, “Programmable Registers.”

11.1 Overview

The AHB-to-APBX bridge provides the i.MX23 with an inexpensive peripheral attachment bus running on the AHB’s XCLK. (The “X” in APBX denotes that the APB*X* runs on a crystal-derived clock, as compared to APB*H*, which is synchronous to HCLK.)

As shown in Figure 12-1, the AHB-to-APBX bridge includes the AHB-to-APB PIO bridge for memory-mapped I/O to the APB devices, as well a central DMA facility for devices on this bus and a vectored interrupt controller for the ARM926 core. Each one of the APB peripherals are documented in their own chapters elsewhere in this document.

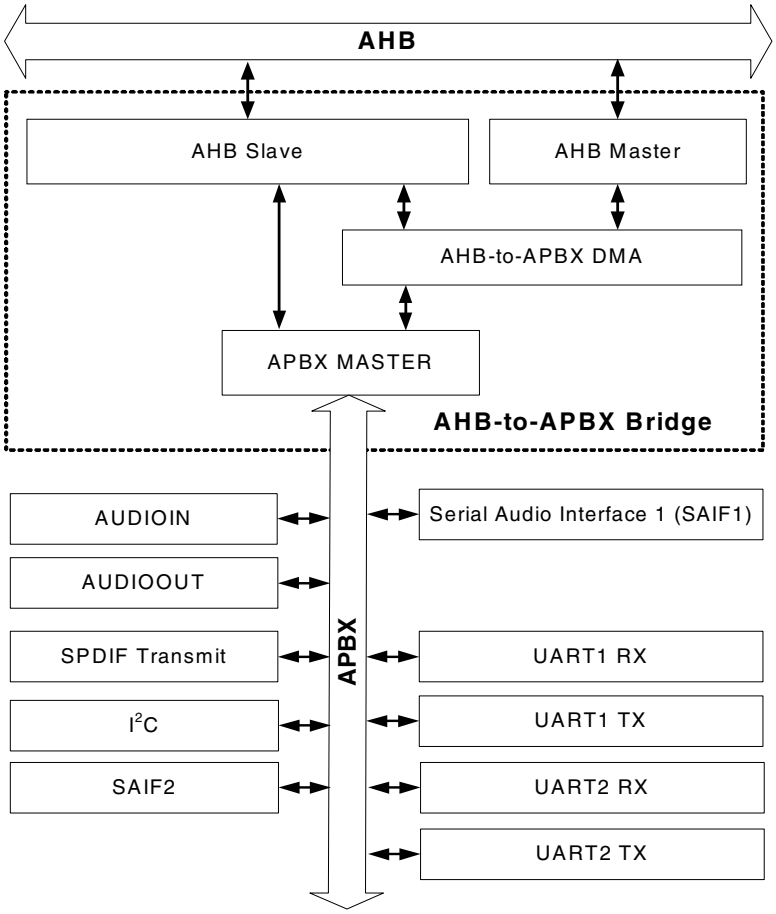


Figure 11-1. AHB-to-APBX Bridge DMA Block Diagram

The DMA controller uses the APBX bus to transfer read and write data to and from each peripheral. There is no separate DMA bus for these devices. Contention between the DMA's use of the APBX bus and AHB-to-APB bridge functions' use of the APBX is mediated by internal arbitration logic. For contention between these two units, the DMA is favored and the AHB slave will report not ready via its HREADY output until the bridge transfer completes. The arbiter tracks repeated lockouts and inverts the priority, so that the CPU is guaranteed every fourth transfer on the APB.

11.2 APBX DMA

The DMA supports sixteen channels of DMA services, as shown in Table 11-1. The shared DMA resource allows each independent channel to follow a simple chained command list. Command chains are built up using the DMA command structure, as shown in Figure 11-2.

Table 11-1. APBX DMA Channel Assignments

APBX DMA Channel #	USAGE
0	Audio ADCs
1	Audio DACs
2	SPDIF TX
3	I^2C
4	SAIF1
5	Reserved
6	UART1 RX, IrDA RX
7	UART1 TX, IrDA TX
8	UART2 RX
9	UART2 TX
10	SAIF2
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved

A single command structure or channel command word specifies a number of operations to be performed by the DMA in support of a given device. Thus, the CPU can set up large units of work, chaining together many DMA channel command words, pass them off to the DMA and have no further concern for the device until the DMA completion interrupt occurs. The i.MX23 is designed to have enough intelligence in the DMA and the devices to keep the interrupt frequency from any device below 1 kHz (arrival intervals longer than 1 ms).

Thus, a single command structure can issue 32-bit PIO write operations to key registers in the associated device using the same APB bus and controls it uses to write DMA data bytes to the device. For example, this allows a chain of operations to be issued to the serial audio interface to send command bytes, address bytes, and data transfers, where the command and address structure is completely under software control, but the administration of that transfer is handled autonomously by the DMA.

Each DMA structure can have from 0 to 15 PIO words appended to it. The #PIOWORDs field, if non-zero, instructs the DMA engine to copy these words to the APB beginning at PADDR = 0x0000 and incrementing its PADDR for each cycle. (Note that for APBX DMA Channel 6, which is the UART/IrDA RX channel, the first PIO word in the DMA command is CTRL0. However, for APBX DMA Channel 7, which is the UART/IrDA TX, the first PIO word in a DMA command is CTRL1.)

The HW_APBX_DEVSEL_CHx bit fields allow reassignment of the APBX device connected to DMA channels 2, 6, and 7. The DMA channel can be programmed to enable an alternate channel owner—for example, SAIF2 instead of SPDIF for Channel 2. Note that the CHx bit fields can be set only once after the chip is reset. To have the DMA channel provide DMA for another device, the chip must be reset and the HW_APBX_DEVSEL register must be reprogrammed. Whichever device is selected for the DMA

channel remains the selected device until the chip is reset and the DMA channel selected for another device.

The DMA master generates only normal read/write transfers to the APBX. It does *not* generate set, clear, or toggle SCT transfers.

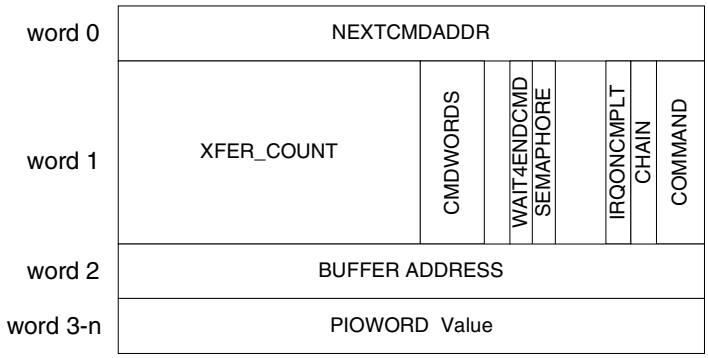


Figure 11-2. AHB-to-APBX Bridge DMA Channel Command Structure

Once any requested PIO words have been transferred to the peripheral, the DMA examines the two-bit command field in the channel command structure. Table 11-2 shows the four commands implemented by the DMA.

Table 11-2. APBX DMA Commands

DMA COMMAND	Usage
00	NO_DMA_XFER. Perform any requested PIO word transfers, but terminate the command before any DMA transfer.
01	DMA_WRITE. Perform any requested PIO word transfers, and then perform a DMA transfer from the peripheral for the specified number of bytes.
10	DMA_READ. Perform any requested PIO word transfers, and then perform a DMA transfer to the peripheral for the specified number of bytes.
11	Reserved

DMA_WRITE operations copy data bytes to system memory (on-chip RAM or SDRAM) from the associated peripheral. Each peripheral has a target PADDR value that it expects to receive DMA bytes. This association is synthesized in the DMA. The DMA_WRITE transfer uses the BUFFER_ADDDRESS word in the command structure to point to the beginning byte to write data from the peripheral.

DMA_READ operations copy data bytes to the APB peripheral from system memory. The DMA engine contains a shared byte aligner that aligns bytes from system memory to or from the peripherals. Peripherals always assume little-endian-aligned data arrives or departs on their 32-bit APB. The DMA_READ transfer uses the BUFFER_ADDRESS word in the command structure to point to the DMA data buffer to be read by the DMA_READ command.

The NO_DMA_XFER command is used to write PIO words to a device without performing any DMA data byte transfers.

As each DMA command completes, it triggers the DMA to load the next DMA command structure in the chain. The normal flow list of DMA commands is found by following the NEXTCMD_ADDR pointer in the DMA command structure. If the wait-for-end-command bit (WAIT4ENDCMD) is set in a command structure, then the DMA channel will wait for the device to signal completion of a command by toggling the apx_endcmcd signal before proceeding to load and execute the next command structure. The semaphore is decremented after the end command is seen.

A detailed bit-field view of the DMA command structure is shown in Table 11-3, which shows a field that specifies the number of bytes to be transferred by this DMA command. The transfer count mechanism is duplicated in the associated peripheral, either as an implied or specified count in the peripheral.

Table 11-3. DMA Channel Command Word in System Memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															
Number DMA Bytes to Transfer																Number PIO Words to Write				Reserved				WAIT4ENDCMD	DECREMENT SEMAPHORE	Reserved			IRQ_COMPLETE	CHAIN	COMMAND		
DMA Buffer or Alternate CCW																															
Zero or More PIO Words to Write to the Associated Peripheral Starting at its Base Address on the APBX Bus																															

Figure 11-3 shows the CHAIN bit in bit 2 of the second word of the command structure. This bit is set to 1 if the NEXT_COMMAND_ADDRESS contains a pointer to another DMA command structure. If a null pointer (0) is loaded into the NEXT_COMMAND_ADDRESS, it will not be detected by the DMA hardware. Only the CHAIN bit indicates whether a valid list exists beyond the current structure.

If the IRQ_COMPLETE bit is set in the command structure, then the last act of the DMA before loading the next command is to set the interrupt status bit corresponding to the current channel. The sticky interrupt request bit in the DMA CSR remains set until cleared by software. It can be used to interrupt the CPU.

Each channel has an eight-bit counting semaphore that controls whether it is in the run or idle state. When the semaphore is non-zero, the channel is ready to run and process commands and DMA transfers. Whenever a command finishes its DMA transfer, it checks the DECREMENT_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 as a result, then the channel enters the IDLE state and remains there until the semaphore is incremented by software. When the semaphore goes to

non-zero and the channel is in its IDLE state, then it uses the value in the HW_APBX_CHn_NXTCMDAR (next command address register) to fetch a pointer to the next command to process. NOTE: this is a double indirect case. This method allows software to append to a running command list under the protection of the counting semaphore.

Receiving an IRQ for HALTONTERMINATE (HOT) is a new feature in the APBH/X DMA descriptor that allows certain peripheral block (e.g. GPMI, SSP, I2C) to signal to the DMA engine that an error has occurred. In prior chips, if a block stalled due to an error, the only practical way to discover this in s/w was via a timer of some sort, or to poll the block. Now, an HOT signal is sent from the peripheral to the DMA engine and causes an IRQ after terminating the DMA descriptor being executed. Note not all peripheral block support this termination feature.

Therefore, it is recommended that s/w use this signal as follows:

- Always set HALTONTERMINATE to 1 in a DMA descriptor. That way, if a peripheral signals HOT, the transfer will end, leaving the peripheral block and the DMA engine synchronized (but at the end of a command).
- When an IRQ from an APBH/X channel is received, and the IRQ is determined to be due to an error (as opposed to an IRQONCOMPLETE interrupt) the software should:

1. reset the channel, and
2. determine the error from error reporting in the peripheral block, then manage the error in the peripheral that is attached to that channel in whatever appropriate way exists for that device (software recovery, device reset, block reset, etc).

To start processing the first time, software creates the command list to be processed. It writes the address of the first command into the HW_APBX_CHn_NXTCMDAR register, and then writes a 1 to the counting semaphore in HW_APBX_CHn_SEMA. The DMA channel loads HW_APBX_CHn_CURCMDAR register and then enters the normal state machine processing for the next command. When software writes a value to the counting semaphore, it is added to the semaphore count by hardware, protecting the case where both hardware and software are trying to change the semaphore on the same clock edge.

Software can examine the value of HW_APBX_CHn_CURCMDAR at any time to determine the location of the command structure that is currently being processed.

11.3 DMA Chain Example

The example in Figure 11-3 shows how to bring the basic items together to make a simple DMA chain to read PCM samples and send them out the Audio Output (DAC) using one DMA channel. This example shows three command structures linked together using their normal command list pointers. The first command writes a single PIO word to the HW_AUDIOOUT_CTRL0 register with a new word count for the DAC. This first command also performs a 512 byte DMA_READ operation to read the data block bytes into the DAC. A second and a third DMA command structure also performs a DMA_READ operation to handle circular buffer style outputs. The completion of each command structure generates an interrupt request. In addition, each command structure decrements the semaphore. If the decompression software

has not provided a buffer in a timely fashion, then the DMA will stall. Without the decrement semaphore interlocking, then the DMA will continue to output a stream of samples. In this mode, it is up to software to use the interrupts to synchronize outputs so that underruns do not occur.

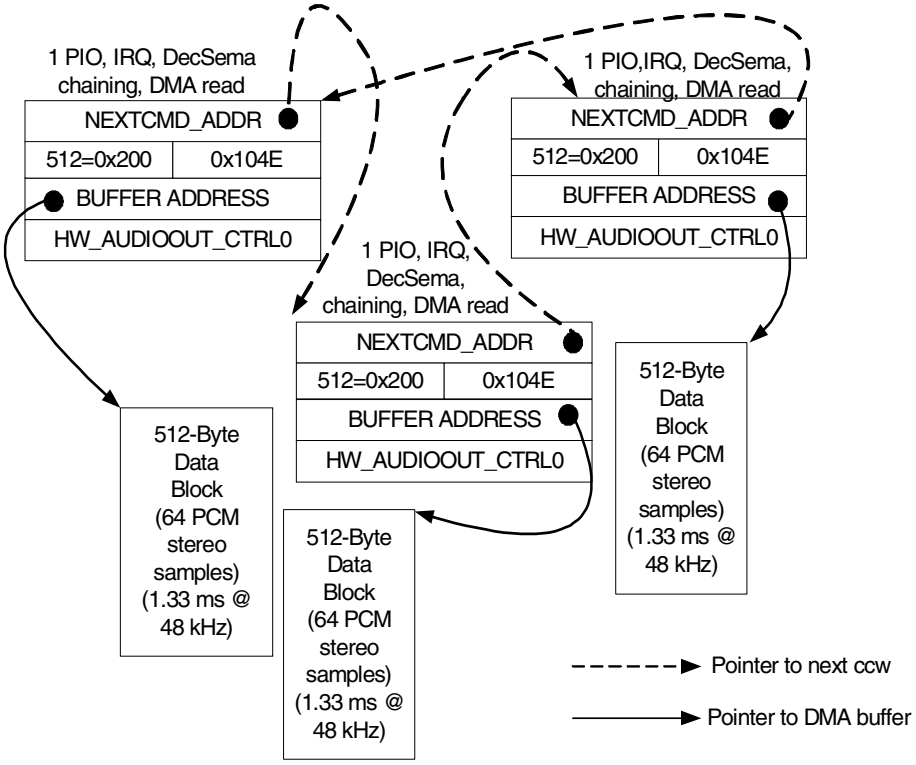


Figure 11-3. AHB-to-APBX Bridge DMA AUDIOOUT (DAC) Example Command Chain

Note that each word of the three-word DMA Command structure corresponds to a PIO register of the DMA that is accessible on the APBX bus. Normally, the DMA copies the next command structure onto these registers for processing at the start of each command by following the value of the pointer previously loaded into the NEXTCMD_ADDR register. In order to start DMA processing, for the first command, one must initialize the PIO registers of the desired channel. FIrst load the next command address register with a pointer to the first command to be loaded. Then write a 1 to the counting semaphore register. This causes the DMA to schedule the targeted channel for DMA command structure load, as if it just finished its previous command.

11.4 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 39.3.10, “Correct Way to Soft Reset a Block,” for additional information on using the SFTRST and CLKGATE bit fields.

11.5 Programmable Registers

This section describes the programmable registers of the AHB-to-APBX bridge block.

11.5.1 AHB to APBX Bridge Control Register 0 Description

The APBX CTRL 0 provides overall control and IRQ status of the AHB to APBX bridge and DMA.

HW_APBX_CTRL0	0x000
HW_APBX_CTRL0_SET	0x004
HW_APBX_CTRL0_CLR	0x008
HW_APBX_CTRL0_TOG	0x00C

Table 11-4. HW_APBX_CTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSVD0																													

Table 11-5. HW_APBX_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Set this bit to zero to enable normal APBX DMA operation. Set this bit to one (default) to disable clocking with the APBX DMA and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the APBX DMA block to its default state.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29:0	**RSVD0**	RO	0x000000	Reserved, always set to zero.

DESCRIPTION:

This register contains softreset, clock gating bits.

EXAMPLE:

```
No Example.
```

11.5.2 AHB to APBX Bridge Control Register 1 Description

The APBX CTRL 1 provides channel complete IRQ status of the AHB to APBX bridge and DMA.

HW_APBX_CTRL1	0x010
HW_APBX_CTRL1_SET	0x014
HW_APBX_CTRL1_CLR	0x018
HW_APBX_CTRL1_TOG	0x01C

Table 11-6. HW_APBX_CTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CH15_CMDCMPLT_IRQ_EN	CH14_CMDCMPLT_IRQ_EN	CH13_CMDCMPLT_IRQ_EN	CH12_CMDCMPLT_IRQ_EN	CH11_CMDCMPLT_IRQ_EN	CH10_CMDCMPLT_IRQ_EN	CH9_CMDCMPLT_IRQ_EN	CH8_CMDCMPLT_IRQ_EN	CH7_CMDCMPLT_IRQ_EN	CH6_CMDCMPLT_IRQ_EN	CH5_CMDCMPLT_IRQ_EN	CH4_CMDCMPLT_IRQ_EN	CH3_CMDCMPLT_IRQ_EN	CH2_CMDCMPLT_IRQ_EN	CH1_CMDCMPLT_IRQ_EN	CH0_CMDCMPLT_IRQ_EN	CH15_CMDCMPLT_IRQ	CH14_CMDCMPLT_IRQ	CH13_CMDCMPLT_IRQ	CH12_CMDCMPLT_IRQ	CH11_CMDCMPLT_IRQ	CH10_CMDCMPLT_IRQ	CH9_CMDCMPLT_IRQ	CH8_CMDCMPLT_IRQ	CH7_CMDCMPLT_IRQ	CH6_CMDCMPLT_IRQ	CH5_CMDCMPLT_IRQ	CH4_CMDCMPLT_IRQ	CH3_CMDCMPLT_IRQ	CH2_CMDCMPLT_IRQ	CH1_CMDCMPLT_IRQ	CH0_CMDCMPLT_IRQ

Table 11-7. HW_APBX_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CH15_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 15.
30	**CH14_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 14.
29	**CH13_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 13.
28	**CH12_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 12.
27	**CH11_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 11.
26	**CH10_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 10.
25	**CH9_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 9.
24	**CH8_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 8.
23	**CH7_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 7.
22	**CH6_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 6.
21	**CH5_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 5.
20	**CH4_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 4.
19	**CH3_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 3.
18	**CH2_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 2.
17	**CH1_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 1.
16	**CH0_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 0.

Table 11-7. HW_APBX_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15	**CH15_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 15. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
14	**CH14_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 14. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
13	**CH13_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 13. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
12	**CH12_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 12. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
11	**CH11_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 11. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
10	**CH10_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 10. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
9	**CH9_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 9. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
8	**CH8_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 8. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
7	**CH7_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 7. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
6	**CH6_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 6. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
5	**CH5_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 5. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
4	**CH4_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 4. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
3	**CH3_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 3. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

Table 11-7. HW_APBX_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CH2_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 2. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
1	**CH1_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
0	**CH0_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

DESCRIPTION:

This register contains the per channel interrupt status bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE:

```
BF_WR(APBX_CTRL1, CH5_CMDCMPLT_IRQ, 0); // use bitfield write macro
BF_APBX_CTRL1.CH5_CMDCMPLT_IRQ = 0; // or, assign to register struct's bitfield
```

11.5.3 AHB to APBX Bridge Control and Status Register 2 Description

The APBX CTRL 2 provides channel error interrupts generated by the AHB to APBX DMA.

HW_APBX_CTRL2	0x020
HW_APBX_CTRL2_SET	0x024
HW_APBX_CTRL2_CLR	0x028
HW_APBX_CTRL2_TOG	0x02C

Table 11-8. HW_APBX_CTRL2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CH15_ERROR_STATUS	CH14_ERROR_STATUS	CH13_ERROR_STATUS	CH12_ERROR_STATUS	CH11_ERROR_STATUS	CH10_ERROR_STATUS	CH9_ERROR_STATUS	CH8_ERROR_STATUS	CH7_ERROR_STATUS	CH6_ERROR_STATUS	CH5_ERROR_STATUS	CH4_ERROR_STATUS	CH3_ERROR_STATUS	CH2_ERROR_STATUS	CH1_ERROR_STATUS	CH0_ERROR_STATUS	CH15_ERROR_IRQ	CH14_ERROR_IRQ	CH13_ERROR_IRQ	CH12_ERROR_IRQ	CH11_ERROR_IRQ	CH10_ERROR_IRQ	CH9_ERROR_IRQ	CH8_ERROR_IRQ	CH7_ERROR_IRQ	CH6_ERROR_IRQ	CH5_ERROR_IRQ	CH4_ERROR_IRQ	CH3_ERROR_IRQ	CH2_ERROR_IRQ	CH1_ERROR_IRQ	CH0_ERROR_IRQ

Table 11-9. HW_APBX_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CH15_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 15. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
30	**CH14_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 14. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
29	**CH13_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 13. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
28	**CH12_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 12. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
27	**CH11_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 11. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
26	**CH10_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 10. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
25	**CH9_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 9. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
24	**CH8_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 8. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.

Table 11-9. HW_APBX_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23	**CH7_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 7. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
22	**CH6_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 6. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
21	**CH5_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 5. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
20	**CH4_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 4. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
19	**CH3_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 3. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
18	**CH2_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 2. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
17	**CH1_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 1. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
16	**CH0_ERROR_STATUS**	RO	0x0	Error status bit for APBX DMA Channel 0. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. TERMINATION = 0x0 An early termination from the device causes error IRQ. BUS_ERROR = 0x1 An AHB bus error causes error IRQ.
15	**CH15_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 15. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

Table 11-9. HW_APBX_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
14	**CH14_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 14. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
13	**CH13_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 13. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
12	**CH12_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 12. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
11	**CH11_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 11. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
10	**CH10_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 10. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
9	**CH9_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 9. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
8	**CH8_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 8. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
7	**CH7_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 7. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
6	**CH6_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 6. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
5	**CH5_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 5. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
4	**CH4_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 4. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
3	**CH3_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 3. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
2	**CH2_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 2. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

Table 11-9. HW_APBX_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**CH1_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
0	**CH0_ERROR_IRQ**	RW	0x0	Error interrupt status bit for APBX DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

DESCRIPTION:

This register contains the per channel bus error interrupt status bits and the per channel completion interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE:

```
Exmpty Example
```

11.5.4 AHB to APBX Bridge Channel Register Description

The APBX CHANNEL CTRL provides reset/freeze control of each DMA channel.

HW_APBX_CHANNEL_CTRL	0x030
HW_APBX_CHANNEL_CTRL_SET	0x034
HW_APBX_CHANNEL_CTRL_CLR	0x038
HW_APBX_CHANNEL_CTRL_TOG	0x03C

Table 11-10. HW_APBX_CHANNEL_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RESET_CHANNEL																FREEZE_CHANNEL															

Table 11-11. HW_APBX_CHANNEL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RESET_CHANNEL**	RW	0x0	Setting a bit in this field causes the DMA controller to take the corresponding channel through its reset state. The bit is reset after the channel resources are cleared. AUDIOIN = 0x0001 AUDIOOUT = 0x0002 SPDIF_TX = 0x0004 I2C = 0x0008 SAIF1 = 0x0010 UART0_RX = 0x0040 UART0_TX = 0x0080 UART1_RX = 0x0100 UART1_TX = 0x0200 SAIF2 = 0x0400
15:0	**FREEZE_CHANNEL**	RW	0x0	Setting a bit in this field will freeze the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When frozen, the channel is deined access to the central DMA resources. AUDIOIN = 0x0001 AUDIOOUT = 0x0002 SPDIF_TX = 0x0004 I2C = 0x0008 SAIF1 = 0x0010 UART0_RX = 0x0040 UART0_TX = 0x0080 UART1_RX = 0x0100 UART1_TX = 0x0200 SAIF2 = 0x0400

DESCRIPTION:

This register contains individual channel reset/freeze bits.

EXAMPLE:

```
Empty Example.
```

11.5.5 AHB to APBX DMA Device Assignment Register Description

This register allows reassignment of the APBX device connected to the DMA Channels.

HW_APBX_DEVSEL 0x040

Table 11-12. HW_APBX_DEVSEL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CH15		CH14		CH13		CH12		CH11		CH10		CH9		CH8		CH7		CH6		CH5		CH4		CH3		CH2		CH1		CH0	

Table 11-13. HW_APBX_DEVSEL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	**CH15**	RO	0x0	Reserved.
29:28	**CH14**	RO	0x0	Reserved.
27:26	**CH13**	RO	0x0	Reserved.
25:24	**CH12**	RO	0x0	Reserved.
23:22	**CH11**	RO	0x0	Reserved.

Table 11-13. HW_APBX_DEVSEL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
21:20	**CH10**	RO	0x0	Reserved.
19:18	**CH9**	RO	0x0	Reserved.
17:16	**CH8**	RO	0x0	Reserved.
15:14	**CH7**	RW	0x0	Reserved.
13:12	**CH6**	RW	0x0	Reserved.
11:10	**CH5**	RO	0x0	Reserved.
9:8	**CH4**	RO	0x0	Reserved.
7:6	**CH3**	RO	0x0	Reserved.
5:4	**CH2**	RO	0x0	Reserved.
3:2	**CH1**	RO	0x0	Reserved.
1:0	**CH0**	RO	0x0	Reserved.

DESCRIPTION:

This register provides a mechanism for assigning the device which is attached to DMA channels 6 and 7.

EXAMPLE:

```
Empty Example.
```

11.5.6 APBX DMA Channel 0 Current Command Address Register Description

The APBX DMA Channel 0 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH0_CURCMDAR 0x100

Table 11-14. HW_APBX_CH0_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-15. HW_APBX_CH0_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 0.

DESCRIPTION:

APBX DMA Channel 0 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBX_CHn_CURCMDAR_RD(0); // read the whole register, since there is only one field
pCurCmd = (hw_apbh_chn_cmd_t *) BF_RDn(APBX_CHn_CURCMDAR, 0, CMD_ADDR); // or, use multi-register bitfield read macro
pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBX_CHn_CURCMDAR(0).CMD_ADDR; // or, assign from bitfield of indexed register's struct
```

11.5.7 APBX DMA Channel 0 Next Command Address Register Description

The APBX DMA Channel 0 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH0_NXTCMDAR 0x110

Table 11-16. HW_APBX_CH0_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-17. HW_APBX_CH0_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 0.

DESCRIPTION:

APBX DMA Channel 0 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 0 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
HW_APBX_CHn_NXTCMDAR_WR(0, (reg32_t) pCommandTwoStructure); // write the entire register, since there is only one field
BF_WRn(APBX_CHn_NXTCMDAR, 0, (reg32_t) pCommandTwoStructure); // or, use multi-register bitfield write macro
HW_APBX_CHn_NXTCMDAR(0).CMD_ADDR = (reg32_t) pCommandTwoStructure; // or, assign to bitfield of indexed register's struct
```

11.5.8 APBX DMA Channel 0 Command Register Description

The APBX DMA Channel 0 command register specifies the DMA transaction to perform for the current command chain item.

HW_APBX_CH0_CMD 0x120

Table 11-18. HW_APBX_CH0_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-19. HW_APBX_CH0_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the ADC device HW_AUDIOIN_DATA. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the ADC, starting with the base PIO address of the ADC (HW_AUDIOIN_CTRL) and increment from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the ABPX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH0_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
hw_apbh_chn_cmd_t dma_cmd;
dma_cmd.XFER_COUNT = 512; // transfer 512 bytes
```

```
dma_cmd.COMMAND = BV_APBX_CHn_CMD_COMMAND__DMA_WRITE; // transfer to system memory from peripheral
device
dma_cmd.CHAIN = 1; // chain an additional command structure on to the list
dma_cmd.IRQONCMPLT = 1; // generate an interrupt on completion of this command structure
```

11.5.9 APBX DMA Channel 0 Buffer Address Register Description

The APBX DMA Channel 0 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH0_BAR 0x130

Table 11-20. HW_APBX_CH0_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-21. HW_APBX_CH0_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
hw_apbh_chn_bar_t dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

11.5.10 APBX DMA Channel 0 Semaphore Register Description

The APBX DMA Channel 0 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH0_SEMA 0x140

Table 11-22. HW_APBX_CH0_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-23. HW_APBX_CH0_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
BF_WR(APBX_CHn_SEMA, 0, INCREMENT_SEMA, 2); // increment semaphore by two
current_sema = BF_RD(APBX_CHn_SEMA, 0, PHORE); // get instantaneous value
```

11.5.11 AHB to APBX DMA Channel 0 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 0 state machine and controls.

HW_APBX_CH0_DEBUG1 0x150

Table 11-24. HW_APBX_CH0_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-25. HW_APBX_CH0_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 0 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 0.

EXAMPLE:

```
Empty example.
```

11.5.12 AHB to APBX DMA Channel 0 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 0.

HW_APBX_CH0_DEBUG2 0x160

Table 11-26. HW_APBX_CH0_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-27. HW_APBX_CH0_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 0.

EXAMPLE:

```
Empty example.
```

11.5.13 APBX DMA Channel 1 Current Command Address Register Description

The APBX DMA Channel 1 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH1_CURCMDAR 0x170

Table 11-28. HW_APBX_CH1_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-29. HW_APBX_CH1_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 1.

DESCRIPTION:

APBX DMA Channel 1 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBX_CHn_CURCMDAR_RD(1); // read the whole register, since there is only one field
pCurCmd = (hw_apbh_chn_cmd_t *) BF_RDn(APBX_CHn_CURCMDAR, 1, CMD_ADDR); // or, use multi-register bitfield read macro
pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBX_CHn_CURCMDAR(1).CMD_ADDR; // or, assign from bitfield of indexed register's struct
```

11.5.14 APBX DMA Channel 1 Next Command Address Register Description

The APBX DMA Channel 1 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH1_NXTCMDAR 0x180

Table 11-30. HW_APBX_CH1_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-31. HW_APBX_CH1_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 1.

DESCRIPTION:

APBX DMA Channel 1 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 1 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
HW_APBX_CHn_NXTCMDAR_WR(1, (reg32_t) pCommandTwoStructure); // write the entire register, since there is only one field
BF_WRn(APBX_CHn_NXTCMDAR, 1, (reg32_t) pCommandTwoStructure); // or, use multi-register bitfield write macro
HW_APBX_CHn_NXTCMDAR(1).CMD_ADDR = (reg32_t) pCommandTwoStructure; // or, assign to bitfield of indexed register's struct
```

11.5.15 APBX DMA Channel 1 Command Register Description

The APBX DMA Channel 1 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH1_CMD 0x190

Table 11-32. HW_APBX_CH1_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-33. HW_APBX_CH1_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the DAC device HW_AUDIOOUT_DATA. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the DAC, starting with the base PIO address of the DAC (HW_AUDIOOUT_CTRL) and increment from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 11-33. HW_APBX_CH1_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH1_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty Example.
```

11.5.16 APBX DMA Channel 1 Buffer Address Register Description

The APBX DMA Channel 1 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH1_BAR 0x1A0

Table 11-34. HW_APBX_CH1_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-35. HW_APBX_CH1_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
hw_apbh_chn_bar_t dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

11.5.17 APBX DMA Channel 1 Semaphore Register Description

The APBX DMA Channel 1 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH1_SEMA 0x1B0

Table 11-36. HW_APBX_CH1_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-37. HW_APBX_CH1_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
BF_WR(APBX_CHn_SEMA, 1, INCREMENT_SEMA, 2); // increment semaphore by two
current_sema = BF_RD(APBX_CHn_SEMA, 1, PHORE); // get instantaneous value
```

11.5.18 AHB to APBX DMA Channel 1 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 1 state machine and controls.

HW_APBX_CH1_DEBUG1 0x1C0

Table 11-38. HW_APBX_CH1_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-39. HW_APBX_CH1_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.

Table 11-39. HW_APBX_CH1_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 1 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 1.

EXAMPLE:

```
Empty example.
```

11.5.19 AHB to APBX DMA Channel 1 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 1.

HW_APBX_CH1_DEBUG2 0x1D0

Table 11-40. HW_APBX_CH1_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-41. HW_APBX_CH1_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 1.

EXAMPLE:

```
Empty example.
```

11.5.20 APBX DMA Channel 2 Current Command Address Register Description

The APBX DMA Channel 2 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH2_CURCMDAR 0x1E0

Table 11-42. HW_APBX_CH2_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-43. HW_APBX_CH2_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 2.

DESCRIPTION:

APBX DMA Channel 2 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.21 APBX DMA Channel 2 Next Command Address Register Description

The APBX DMA Channel 2 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH2_NXTCMDAR 0x1F0

Table 11-44. HW_APBX_CH2_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-45. HW_APBX_CH2_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 2.

DESCRIPTION:

APBX DMA Channel 2 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 0 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty Example.
```

11.5.22 APBX DMA Channel 2 Command Register Description

The APBX DMA Channel 2 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH2_CMD 0x200

Table 11-46. HW_APBX_CH2_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-47. HW_APBX_CH2_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the SPDIF or SAIF1 device. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the SPDIF, starting with the base PIO address of the SPDIF or SAIF1 and incrementing from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH2_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.23 APBX DMA Channel 2 Buffer Address Register Description

The APBX DMA Channel 2 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH2_BAR 0x210

Table 11-48. HW_APBX_CH2_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-49. HW_APBX_CH2_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.24 APBX DMA Channel 2 Semaphore Register Description

The APBX DMA Channel 2 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH2_SEMA 0x220

Table 11-50. HW_APBX_CH2_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-51. HW_APBX_CH2_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.25 AHB to APBX DMA Channel 2 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 2 state machine and controls.

HW_APBX_CH2_DEBUG1 0x230

Table 11-52. HW_APBX_CH2_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-53. HW_APBX_CH2_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device

Table 11-53. HW_APBX_CH2_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 2 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 2.

EXAMPLE:

```
Empty example.
```

11.5.26 AHB to APBX DMA Channel 2 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 2.

HW_APBX_CH2_DEBUG2 0x240

Table 11-54. HW_APBX_CH2_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-55. HW_APBX_CH2_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 2.

EXAMPLE:

```
Empty example.
```

11.5.27 APBX DMA Channel 3 Current Command Address Register Description

The APBX DMA Channel 3 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH3_CURCMDAR 0x250

Table 11-56. HW_APBX_CH3_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-57. HW_APBX_CH3_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 3.

DESCRIPTION:

APBX DMA Channel 3 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.28 APBX DMA Channel 3 Next Command Address Register Description

The APBX DMA Channel 3 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH3_NXTCMDAR 0x260

Table 11-58. HW_APBX_CH3_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-59. HW_APBX_CH3_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 3.

DESCRIPTION:

APBX DMA Channel 3 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 3 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.29 APBX DMA Channel 3 Command Register Description

The APBX DMA Channel 3 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH3_CMD 0x270

Table 11-60. HW_APBX_CH3_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-61. HW_APBX_CH3_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the I2C device HW_I2C_DATA. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the I2C, starting with the base PIO address of the I2C (HW_I2C_CTRL0) and increment from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 11-61. HW_APBX_CH3_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH3_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.30 APBX DMA Channel 3 Buffer Address Register Description

The APBX DMA Channel 3 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH3_BAR 0x280

Table 11-62. HW_APBX_CH3_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-63. HW_APBX_CH3_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.31 APBX DMA Channel 3 Semaphore Register Description

The APBX DMA Channel 3 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH3_SEMA 0x290

Table 11-64. HW_APBX_CH3_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-65. HW_APBX_CH3_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.32 AHB to APBX DMA Channel 3 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 3 state machine and controls.

HW_APBX_CH3_DEBUG1 0x2A0

Table 11-66. HW_APBX_CH3_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-67. HW_APBX_CH3_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.

Table 11-67. HW_APBX_CH3_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 3 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 3.

EXAMPLE:

```
Empty example.
```

11.5.33 AHB to APBX DMA Channel 3 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 3.

HW_APBX_CH3_DEBUG2 0x2B0

Table 11-68. HW_APBX_CH3_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-69. HW_APBX_CH3_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 3.

EXAMPLE:

```
Empty example.
```

11.5.34 APBX DMA Channel 4 Current Command Address Register Description

The APBX DMA Channel 4 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH4_CURCMDAR 0x2C0

Table 11-70. HW_APBX_CH4_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-71. HW_APBX_CH4_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 4.

DESCRIPTION:

APBX DMA Channel 4 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.35 APBX DMA Channel 4 Next Command Address Register Description

The APBX DMA Channel 4 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH4_NXTCMDAR 0x2D0

Table 11-72. HW_APBX_CH4_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-73. HW_APBX_CH4_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 4.

DESCRIPTION:

APBX DMA Channel 4 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 4 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.36 APBX DMA Channel 4 Command Register Description

The APBX DMA Channel 4 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH4_CMD 0x2E0

Table 11-74. HW_APBX_CH4_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-75. HW_APBX_CH4_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the SAIF1 device. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the SAIF1. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH4_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.37 APBX DMA Channel 4 Buffer Address Register Description

The APBX DMA Channel 4 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH4_BAR 0x2F0

Table 11-76. HW_APBX_CH4_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-77. HW_APBX_CH4_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device associate with this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.38 APBX DMA Channel 4 Semaphore Register Description

The APBX DMA Channel 4 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH4_SEMA 0x300

Table 11-78. HW_APBX_CH4_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-79. HW_APBX_CH4_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.39 AHB to APBX DMA Channel 4 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 4 state machine and controls.

HW_APBX_CH4_DEBUG1 0x310

Table 11-80. HW_APBX_CH4_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-81. HW_APBX_CH4_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device

Table 11-81. HW_APBX_CH4_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 4 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 4.

EXAMPLE:

```
Empty example.
```

11.5.40 AHB to APBX DMA Channel 4 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 4.

HW_APBX_CH4_DEBUG2 0x320

Table 11-82. HW_APBX_CH4_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-83. HW_APBX_CH4_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 4.

EXAMPLE:

```
Empty example.
```

11.5.41 APBX DMA Channel 5 Current Command Address Register Description

The APBX DMA Channel 5 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH5_CURCMDAR 0x330

Table 11-84. HW_APBX_CH5_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-85. HW_APBX_CH5_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 5.

DESCRIPTION:

APBX DMA Channel 5 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.42 APBX DMA Channel 5 Next Command Address Register Description

The APBX DMA Channel 5 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH5_NXTCMDAR 0x340

Table 11-86. HW_APBX_CH5_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-87. HW_APBX_CH5_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 5.

DESCRIPTION:

APBX DMA Channel 5 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 5 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.43 APBX DMA Channel 5 Command Register Description

The APBX DMA Channel 5 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH5_CMD 0x350

Table 11-88. HW_APBX_CH5_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-89. HW_APBX_CH5_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the DRI device HW_DRI_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the DRI, starting with the base PIO address of the DRI (HW_DRI_CTRL) and increment from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH5_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.44 APBX DMA Channel 5 Buffer Address Register Description

The APBX DMA Channel 5 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH5_BAR 0x360

Table 11-90. HW_APBX_CH5_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-91. HW_APBX_CH5_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.45 APBX DMA Channel 5 Semaphore Register Description

The APBX DMA Channel 5 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH5_SEMA 0x370

Table 11-92. HW_APBX_CH5_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-93. HW_APBX_CH5_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.46 AHB to APBX DMA Channel 5 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 5 state machine and controls.

HW_APBX_CH5_DEBUG1 0x380

Table 11-94. HW_APBX_CH5_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-95. HW_APBX_CH5_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.

Table 11-95. HW_APBX_CH5_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 5 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 5.

EXAMPLE:

```
Empty example.
```

11.5.47 AHB to APBX DMA Channel 5 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 5.

HW_APBX_CH5_DEBUG2 0x390

Table 11-96. HW_APBX_CH5_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-97. HW_APBX_CH5_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 5.

EXAMPLE:

```
Empty example.
```

11.5.48 APBX DMA Channel 6 Current Command Address Register Description

The APBX DMA Channel 6 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH6_CURCMDAR 0x3A0

Table 11-98. HW_APBX_CH6_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-99. HW_APBX_CH6_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 6.

DESCRIPTION:

APBX DMA Channel 6 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.49 APBX DMA Channel 6 Next Command Address Register Description

The APBX DMA Channel 6 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH6_NXTCMDAR 0x3B0

Table 11-100. HW_APBX_CH6_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-101. HW_APBX_CH6_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 6.

DESCRIPTION:

APBX DMA Channel 6 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 6 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.50 APBX DMA Channel 6 Command Register Description

The APBX DMA Channel 6 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH6_CMD 0x3C0

Table 11-102. HW_APBX_CH6_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-103. HW_APBX_CH6_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH6_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.51 APBX DMA Channel 6 Buffer Address Register Description

The APBX DMA Channel 6 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH6_BAR 0x3D0

Table 11-104. HW_APBX_CH6_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-105. HW_APBX_CH6_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.52 APBX DMA Channel 6 Semaphore Register Description

The APBX DMA Channel 6 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH6_SEMA 0x3E0

Table 11-106. HW_APBX_CH6_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-107. HW_APBX_CH6_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.53 AHB to APBX DMA Channel 6 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 6 state machine and controls.

HW_APBX_CH6_DEBUG1 0x3F0

Table 11-108. HW_APBX_CH6_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-109. HW_APBX_CH6_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device

Table 11-109. HW_APBX_CH6_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 6 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 6.

EXAMPLE:

```
Empty example.
```

11.5.54 AHB to APBX DMA Channel 6 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 6.

HW_APBX_CH6_DEBUG2 0x400

Table 11-110. HW_APBX_CH6_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-111. HW_APBX_CH6_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 6.

EXAMPLE:

```
Empty example.
```

11.5.55 APBX DMA Channel 7 Current Command Address Register Description

The APBX DMA Channel 7 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH7_CURCMDAR 0x410

Table 11-112. HW_APBX_CH7_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-113. HW_APBX_CH7_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 7.

DESCRIPTION:

APBX DMA Channel 7 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.56 APBX DMA Channel 7 Next Command Address Register Description

The APBX DMA Channel 7 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH7_NXTCMDAR 0x420

Table 11-114. HW_APBX_CH7_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-115. HW_APBX_CH7_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 7.

DESCRIPTION:

APBX DMA Channel 7 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 7 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.57 APBX DMA Channel 7 Command Register Description

The APBX DMA Channel 7 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH7_CMD 0x430

Table 11-116. HW_APBX_CH7_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-117. HW_APBX_CH7_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 11-117. HW_APBX_CH7_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH7_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.58 APBX DMA Channel 7 Buffer Address Register Description

The APBX DMA Channel 7 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH7_BAR 0x440

Table 11-118. HW_APBX_CH7_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-119. HW_APBX_CH7_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.59 APBX DMA Channel 7 Semaphore Register Description

The APBX DMA Channel 7 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH7_SEMA 0x450

Table 11-120. HW_APBX_CH7_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-121. HW_APBX_CH7_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.60 AHB to APBX DMA Channel 7 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 7 state machine and controls.

HW_APBX_CH7_DEBUG1 0x460

Table 11-122. HW_APBX_CH7_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-123. HW_APBX_CH7_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.

Table 11-123. HW_APBX_CH7_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 7 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 7.

EXAMPLE:

```
Empty example.
```

11.5.61 AHB to APBX DMA Channel 7 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 7.

HW_APBX_CH7_DEBUG2 0x470

Table 11-124. HW_APBX_CH7_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-125. HW_APBX_CH7_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 7.

EXAMPLE:

```
Empty example.
```

11.5.62 APBX DMA Channel 8 Current Command Address Register Description

The APBX DMA Channel 8 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH8_CURCMDAR 0x480

Table 11-126. HW_APBX_CH8_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-127. HW_APBX_CH8_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 8.

DESCRIPTION:

APBX DMA Channel 8 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.63 APBX DMA Channel 8 Next Command Address Register Description

The APBX DMA Channel 8 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH8_NXTCMDAR 0x490

Table 11-128. HW_APBX_CH8_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-129. HW_APBX_CH8_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 8.

DESCRIPTION:

APBX DMA Channel 8 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 8 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.64 APBX DMA Channel 8 Command Register Description

The APBX DMA Channel 8 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH8_CMD 0x4A0

Table 11-130. HW_APBX_CH8_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-131. HW_APBX_CH8_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH8_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included

with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.65 APBX DMA Channel 8 Buffer Address Register Description

The APBX DMA Channel 8 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH8_BAR 0x4B0

Table 11-132. HW_APBX_CH8_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-133. HW_APBX_CH8_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.66 APBX DMA Channel 8 Semaphore Register Description

The APBX DMA Channel 8 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH8_SEMA 0x4C0

Table 11-134. HW_APBX_CH8_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-135. HW_APBX_CH8_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.67 AHB to APBX DMA Channel 8 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 8 state machine and controls.

HW_APBX_CH8_DEBUG1 0x4D0

Table 11-136. HW_APBX_CH8_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-137. HW_APBX_CH8_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.

Table 11-137. HW_APBX_CH8_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 8 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 8.

EXAMPLE:

```
Empty example.
```

11.5.68 AHB to APBX DMA Channel 8 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 8.

HW_APBX_CH8_DEBUG2 0x4E0

Table 11-138. HW_APBX_CH8_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-139. HW_APBX_CH8_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 8.

EXAMPLE:

```
Empty example.
```

11.5.69 APBX DMA Channel 9 Current Command Address Register Description

The APBX DMA Channel 9 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH9_CURCMDAR 0x4F0

Table 11-140. HW_APBX_CH9_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-141. HW_APBX_CH9_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 9.

DESCRIPTION:

APBX DMA Channel 9 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.70 APBX DMA Channel 9 Next Command Address Register Description

The APBX DMA Channel 9 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH9_NXTCMDAR 0x500

Table 11-142. HW_APBX_CH9_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-143. HW_APBX_CH9_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 9.

DESCRIPTION:

APBX DMA Channel 9 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 9 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.71 APBX DMA Channel 9 Command Register Description

The APBX DMA Channel 9 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH9_CMD 0x510

Table 11-144. HW_APBX_CH9_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-145. HW_APBX_CH9_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH9_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.72 APBX DMA Channel 9 Buffer Address Register Description

The APBX DMA Channel 9 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH9_BAR 0x520

Table 11-146. HW_APBX_CH9_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-147. HW_APBX_CH9_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.73 APBX DMA Channel 9 Semaphore Register Description

The APBX DMA Channel 9 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH9_SEMA 0x530

Table 11-148. HW_APBX_CH9_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-149. HW_APBX_CH9_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.74 AHB to APBX DMA Channel 9 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 9 state machine and controls.

HW_APBX_CH9_DEBUG1 0x540

Table 11-150. HW_APBX_CH9_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-151. HW_APBX_CH9_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device

Table 11-151. HW_APBX_CH9_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 9 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 9.

EXAMPLE:

```
Empty example.
```

11.5.75 AHB to APBX DMA Channel 9 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 9.

HW_APBX_CH9_DEBUG2 0x550

Table 11-152. HW_APBX_CH9_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-153. HW_APBX_CH9_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 9.

EXAMPLE:

```
Empty example.
```

11.5.76 APBX DMA Channel 10 Current Command Address Register Description

The APBX DMA Channel 10 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH10_CURCMDAR 0x560

Table 11-154. HW_APBX_CH10_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-155. HW_APBX_CH10_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 10.

DESCRIPTION:

APBX DMA Channel 10 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.77 APBX DMA Channel 10 Next Command Address Register Description

The APBX DMA Channel 10 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH10_NXTCMDAR 0x570

Table 11-156. HW_APBX_CH10_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-157. HW_APBX_CH10_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 10.

DESCRIPTION:

APBX DMA Channel 10 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 10 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.78 APBX DMA Channel 10 Command Register Description

The APBX DMA Channel 10 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH10_CMD 0x580

Table 11-158. HW_APBX_CH10_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-159. HW_APBX_CH10_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 11-159. HW_APBX_CH10_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH10_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.79 APBX DMA Channel 10 Buffer Address Register Description

The APBX DMA Channel 10 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH10_BAR 0x590

Table 11-160. HW_APBX_CH10_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-161. HW_APBX_CH10_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.80 APBX DMA Channel 10 Semaphore Register Description

The APBX DMA Channel 10 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH10_SEMA 0x5A0

Table 11-162. HW_APBX_CH10_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-163. HW_APBX_CH10_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.81 AHB to APBX DMA Channel 10 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 10 state machine and controls.

HW_APBX_CH10_DEBUG1 0x5B0

Table 11-164. HW_APBX_CH10_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-165. HW_APBX_CH10_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.

Table 11-165. HW_APBX_CH10_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 10 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 10.

EXAMPLE:

```
Empty example.
```

11.5.82 AHB to APBX DMA Channel 10 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 10.

HW_APBX_CH10_DEBUG2 0x5C0

Table 11-166. HW_APBX_CH10_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-167. HW_APBX_CH10_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 10.

EXAMPLE:

```
Empty example.
```

11.5.83 APBX DMA Channel 11 Current Command Address Register Description

The APBX DMA Channel 11 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH11_CURCMDAR 0x5D0

Table 11-168. HW_APBX_CH11_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-169. HW_APBX_CH11_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 11.

DESCRIPTION:

APBX DMA Channel 11 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.84 APBX DMA Channel 11 Next Command Address Register Description

The APBX DMA Channel 11 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH11_NXTCMDAR 0x5E0

Table 11-170. HW_APBX_CH11_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-171. HW_APBX_CH11_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 11.

DESCRIPTION:

APBX DMA Channel 11 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 11 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.85 APBX DMA Channel 11 Command Register Description

The APBX DMA Channel 11 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH11_CMD 0x5F0

Table 11-172. HW_APBX_CH11_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-173. HW_APBX_CH11_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH11_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.86 APBX DMA Channel 11 Buffer Address Register Description

The APBX DMA Channel 11 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH11_BAR 0x600

Table 11-174. HW_APBX_CH11_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-175. HW_APBX_CH11_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.87 APBX DMA Channel 11 Semaphore Register Description

The APBX DMA Channel 11 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH11_SEMA 0x610

Table 11-176. HW_APBX_CH11_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-177. HW_APBX_CH11_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.88 AHB to APBX DMA Channel 11 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 11 state machine and controls.

HW_APBX_CH11_DEBUG1 0x620

Table 11-178. HW_APBX_CH11_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-179. HW_APBX_CH11_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device

Table 11-179. HW_APBX_CH11_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 11 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 11.

EXAMPLE:

```
Empty example.
```

11.5.89 AHB to APBX DMA Channel 11 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 11.

HW_APBX_CH11_DEBUG2 0x630

Table 11-180. HW_APBX_CH11_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-181. HW_APBX_CH11_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 11.

EXAMPLE:

```
Empty example.
```

11.5.90 APBX DMA Channel 12 Current Command Address Register Description

The APBX DMA Channel 12 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH12_CURCMDAR 0x640

Table 11-182. HW_APBX_CH12_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-183. HW_APBX_CH12_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 12.

DESCRIPTION:

APBX DMA Channel 12 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.91 APBX DMA Channel 12 Next Command Address Register Description

The APBX DMA Channel 12 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH12_NXTCMDAR 0x650

Table 11-184. HW_APBX_CH12_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-185. HW_APBX_CH12_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 12.

DESCRIPTION:

APBX DMA Channel 12 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 12 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.92 APBX DMA Channel 12 Command Register Description

The APBX DMA Channel 12 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH12_CMD 0x660

Table 11-186. HW_APBX_CH12_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-187. HW_APBX_CH12_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 11-187. HW_APBX_CH12_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH12_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.93 APBX DMA Channel 12 Buffer Address Register Description

The APBX DMA Channel 12 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH12_BAR 0x670

Table 11-188. HW_APBX_CH12_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-189. HW_APBX_CH12_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.94 APBX DMA Channel 12 Semaphore Register Description

The APBX DMA Channel 12 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH12_SEMA 0x680

Table 11-190. HW_APBX_CH12_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-191. HW_APBX_CH12_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.95 AHB to APBX DMA Channel 12 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 12 state machine and controls.

HW_APBX_CH12_DEBUG1 0x690

Table 11-192. HW_APBX_CH12_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-193. HW_APBX_CH12_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.

Table 11-193. HW_APBX_CH12_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 12 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 12.

EXAMPLE:

```
Empty example.
```

11.5.96 AHB to APBX DMA Channel 12 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 12.

HW_APBX_CH12_DEBUG2 0x6A0

Table 11-194. HW_APBX_CH12_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-195. HW_APBX_CH12_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 12.

EXAMPLE:

```
Empty example.
```

11.5.97 APBX DMA Channel 13 Current Command Address Register Description

The APBX DMA Channel 13 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH13_CURCMDAR 0x6B0

Table 11-196. HW_APBX_CH13_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-197. HW_APBX_CH13_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 13.

DESCRIPTION:

APBX DMA Channel 13 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.98 APBX DMA Channel 13 Next Command Address Register Description

The APBX DMA Channel 13 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH13_NXTCMDAR 0x6C0

Table 11-198. HW_APBX_CH13_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-199. HW_APBX_CH13_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 13.

DESCRIPTION:

APBX DMA Channel 13 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 13 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.99 APBX DMA Channel 13 Command Register Description

The APBX DMA Channel 13 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH13_CMD 0x6D0

Table 11-200. HW_APBX_CH13_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-201. HW_APBX_CH13_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH13_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.100 APBX DMA Channel 13 Buffer Address Register Description

The APBX DMA Channel 13 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH13_BAR 0x6E0

Table 11-202. HW_APBX_CH13_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-203. HW_APBX_CH13_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.101 APBX DMA Channel 13 Semaphore Register Description

The APBX DMA Channel 13 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH13_SEMA 0x6F0

Table 11-204. HW_APBX_CH13_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-205. HW_APBX_CH13_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.102 AHB to APBX DMA Channel 13 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 13 state machine and controls.

HW_APBX_CH13_DEBUG1 0x700

Table 11-206. HW_APBX_CH13_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-207. HW_APBX_CH13_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device

Table 11-207. HW_APBX_CH13_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 13 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 13.

EXAMPLE:

```
Empty example.
```

11.5.103 AHB to APBX DMA Channel 13 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 13.

HW_APBX_CH13_DEBUG2 0x710

Table 11-208. HW_APBX_CH13_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-209. HW_APBX_CH13_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 13.

EXAMPLE:

```
Empty example.
```

11.5.104 APBX DMA Channel 14 Current Command Address Register Description

The APBX DMA Channel 14 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH14_CURCMDAR 0x720

Table 11-210. HW_APBX_CH14_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-211. HW_APBX_CH14_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 14.

DESCRIPTION:

APBX DMA Channel 14 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.105 APBX DMA Channel 14 Next Command Address Register Description

The APBX DMA Channel 14 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH14_NXTCMDAR 0x730

Table 11-212. HW_APBX_CH14_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-213. HW_APBX_CH14_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 14.

DESCRIPTION:

APBX DMA Channel 14 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 14 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.106 APBX DMA Channel 14 Command Register Description

The APBX DMA Channel 14 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH14_CMD 0x740

Table 11-214. HW_APBX_CH14_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-215. HW_APBX_CH14_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words
11:9	**RSVD1**	RO	0x0	Reserved, always set to zero.
8	**HALTONTERMINATE**	RO	0x0	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

Table 11-215. HW_APBX_CH14_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH14_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.107 APBX DMA Channel 14 Buffer Address Register Description

The APBX DMA Channel 14 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH14_BAR 0x750

Table 11-216. HW_APBX_CH14_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-217. HW_APBX_CH14_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.108 APBX DMA Channel 14 Semaphore Register Description

The APBX DMA Channel 14 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH14_SEMA 0x760

Table 11-218. HW_APBX_CH14_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-219. HW_APBX_CH14_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.109 AHB to APBX DMA Channel 14 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 14 state machine and controls.

HW_APBX_CH14_DEBUG1 0x770

Table 11-220. HW_APBX_CH14_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-221. HW_APBX_CH14_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.

Table 11-221. HW_APBX_CH14_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 14 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 14.

EXAMPLE:

```
Empty example.
```

11.5.110 AHB to APBX DMA Channel 14 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 14.

HW_APBX_CH14_DEBUG2 0x780

Table 11-222. HW_APBX_CH14_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-223. HW_APBX_CH14_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 14.

EXAMPLE:

```
Empty example.
```

11.5.111 APBX DMA Channel 15 Current Command Address Register Description

The APBX DMA Channel 15 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH15_CURCMDAR 0x790

Table 11-224. HW_APBX_CH15_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-225. HW_APBX_CH15_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 15.

DESCRIPTION:

APBX DMA Channel 15 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.112 APBX DMA Channel 15 Next Command Address Register Description

The APBX DMA Channel 15 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH15_NXTCMDAR 0x7A0

Table 11-226. HW_APBX_CH15_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-227. HW_APBX_CH15_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 15.

DESCRIPTION:

APBX DMA Channel 15 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 15 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.113 APBX DMA Channel 15 Command Register Description

The APBX DMA Channel 15 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH15_CMD 0x7B0

Table 11-228. HW_APBX_CH15_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 11-229. HW_APBX_CH15_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH15_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.114 APBX DMA Channel 15 Buffer Address Register Description

The APBX DMA Channel 15 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH15_BAR 0x7C0

Table 11-230. HW_APBX_CH15_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 11-231. HW_APBX_CH15_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.115 APBX DMA Channel 15 Semaphore Register Description

The APBX DMA Channel 15 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH15_SEMA 0x7D0

Table 11-232. HW_APBX_CH15_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 11-233. HW_APBX_CH15_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.116 AHB to APBX DMA Channel 15 Debug Information Description

This register gives debug visibility into the APBX DMA Channel 15 state machine and controls.

HW_APBX_CH15_DEBUG1 0x7E0

Table 11-234. HW_APBX_CH15_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 11-235. HW_APBX_CH15_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request Signal from the APB device
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst Signal from the APB device

Table 11-235. HW_APBX_CH15_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 15 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 15.

EXAMPLE:

```
Empty example.
```

11.5.117 AHB to APBX DMA Channel 15 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 15.

HW_APBX_CH15_DEBUG2 0x7F0

Table 11-236. HW_APBX_CH15_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-237. HW_APBX_CH15_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 15.

EXAMPLE:

```
Empty example.
```

11.5.118 APBX Bridge Version Register Description

This register always returns a known read value for debug purposes it indicates the version of the block.

HW_APBX_VERSION 0x800

Table 11-238. HW_APBX_VERSION

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MAJOR								MINOR								STEP															

Table 11-239. HW_APBX_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**MAJOR**	RO	0x02	Fixed read-only value reflecting the MAJOR field of the RTL version.

Table 11-239. HW_APBX_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23:16	**MINOR**	RO	0x01	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	**STEP**	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

DESCRIPTION:

This register indicates the RTL version in use.

EXAMPLE:

```
if (HW_APBX_VERSION.B.MAJOR != 1)
Error();
```

APBX Block v2.1, Revision 1.30

11.5.119

Chapter 12
External Memory Interface (EMI)

This chapter describes the external memory interface (EMI) on the i.MX23. It describes the DRAM controller and EMI power management. Programmable registers for both the DRAM controller are described in Section 12.5, “Programmable Registers.”

12.1 Overview

The i.MX23 supports off-chip DRAM storage via the EMI controller, which is connected to the four internal AHB/AXI busses.

The EMI supports multiple external memory types, including:

- 1.8-V Mobile DDR
- Standard 2.5V DDR1

The DRAM controller supports up to two external chip-select signals for the i.MX23 platform. Programmable registers within the DRAM controller allow great flexibility for device timings, low-power operation, and performance tuning. Note the differences between the two package options:

- The 128-pin LQFP has 1 chip enable. Maximum DRAM supported is 64MB.
- The 169-pin BGA has 2 chip enables. Maximum DRAM supported is 128MB.

The EMI uses two primary clocks: the AHB bus HCLK and the DRAM source clock EMI_CLK. The maximum specified frequencies for these two clocks can be found in Chapter 2, “Characteristics and Specifications. The memory controller operates at frequencies that are asynchronous to the rest of the i.MX23.

The EMI consists of two major components:

- DRAM controller
- Delay compensation circuitry (DCC)

A block diagram of the external memory controller is shown in Figure 12-1.

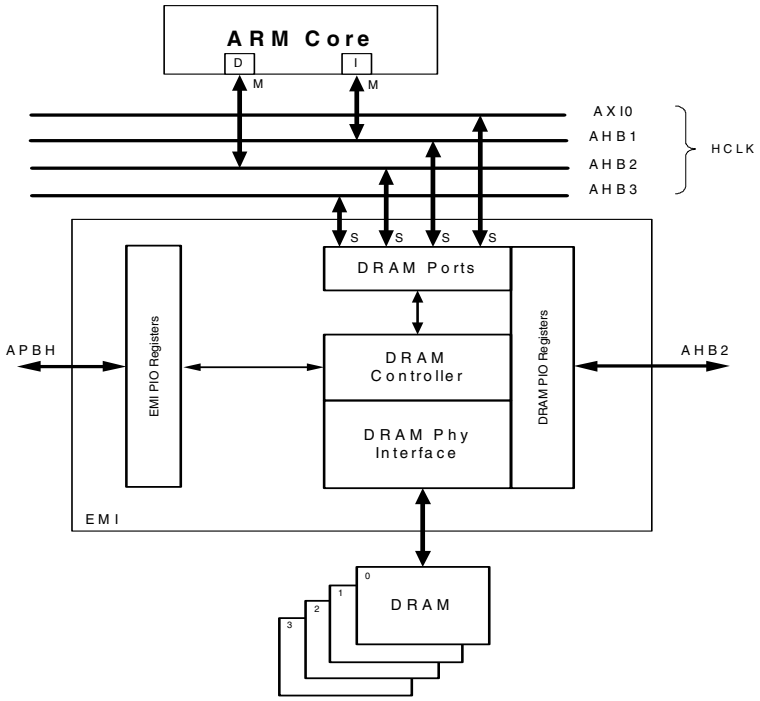


Figure 12-1. External Memory Interface (EMI) Top-Level Block Diagram

12.1.1 AHB Address Ranges

The EMI supports a 512-Mbyte DRAM address space at address 0x40000000. The 512-Mbyte DRAM address space is broken down within the DRAM controller as shown in Figure 12-2.

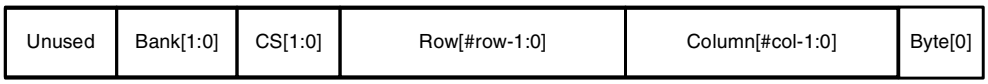


Figure 12-2. DRAM Controller AHB Address Breakdown

Note: This DRAM memory range is not available if the DRAM memory controller is not initialized. A memory access to this range without initializing the DRAM memory controller will result in a system bus hang or a bus error, depending on the state of the TRAP_INIT and TRAP_SR bits in the HW_EMI_CTRL register.

The DRAM controller has programmability to support variously sized DRAM devices. Thus, the number of rows and columns are programmable. In addition, the number of external devices that are in use is programmable, as well. With this organization, the DRAM chips form one large contiguous address space:

$$dram\_memory\_available = 2 * 2^{\#col} * 2^{\#row} * (\#\ dram\_devices) * (\#\ banks\_per\_device)$$

For example, with 10 column bits, 12 row bits, 1 external device and 4 banks per device, the total memory space available would be 32 Mbytes, as follows:

$$2 * 2^{10} * 2^{12} * 1 * 4 = 33{,}554{,}432 \text{ bytes}$$

12.2 DRAM Controller

The DRAM controller handles all of the accesses to the off-chip DRAM devices, including refresh cycles, entry into and exit from low-power modes, and data transfers. This controller supports the following devices:

- 1.8-V mobile DDR
- 2.5-V DDR1

The EMI also supports the connection of simple or multiple external devices with the matrix shown in Table 2-11. See Section 12.6, "EMI Memory Parameters and Register Settings," for configuration examples for DDR and mDDR devices.

The architecture of the DRAM controller is shown in Figure 12-3.

12.2.1 Delay Compensation Circuit (DCC)

The delay compensation circuit (DCC) controls the source-synchronous write and read clocks for data transfer to and from DRAM devices. It is responsible for synchronizing the inbound DRAM data using the DRAM clock (in bypass mode) or the DQS signals. This is done by implementing a series of buffers to delay the clock or DQS signals and then picking the correct tap from the buffer chain to use to sample the data.

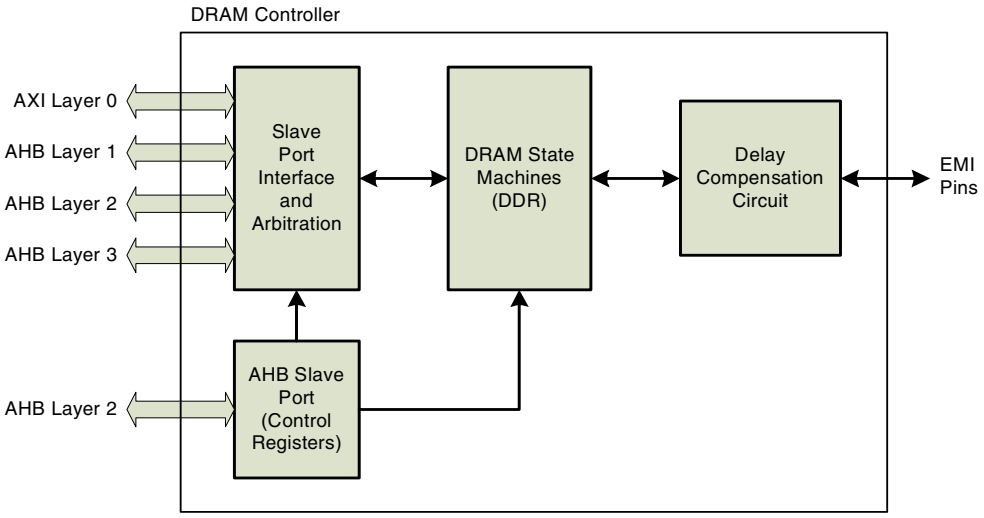


Figure 12-3. DRAM Controller Architecture

12.2.2 Address Mapping

The memory controller automatically maps user addresses to the DRAM memory in a contiguous block. Addressing starts at system address 0x40000000 and extends up to a maximum system address of 0x5FFFFFFF. This allows for a maximum of 512 Mbytes of DRAM storage. This mapping is accomplished by setting certain bit fields in the internal DRAM controller registers.

12.2.2.1 DDR Address Mapping Options

The address structure of DDR devices consists of these fields:

- Datapath
- Column
- Row
- Chip Select
- Bank

The DRAM controller extracts these fields from the lower 30 bits of the system address. The exact bit positions for each field are defined in the programmable registers of the controller. The order of extraction, however, is always fixed as:

Bank-Chip Select-Row-Column-Datapath

The maximum widths of each of these fields are fixed at:

- Bank = 2 bits
- Chip Select = 2 bits
- Row = 13 bits
- Column = 12 bits
- Datapath = 1 bit

The actual width of the column and row fields are programmable using the device address width bit fields (ADDR_PINS and COLUMN_SIZE) in the memory controller.

These maximum values, when combined, define the maximum 512-Mbyte addressable DRAM memory space. Figure 12-4 shows the positioning of the fields within the system address:

Note that practically, the maximum addressable external memory is limited to 128MB for the 169BGA and 64MB for the 128QFP packages. This is because most larger DRAMs require 14 row bits and the EMI controller supports a maximum of 13 row bits.

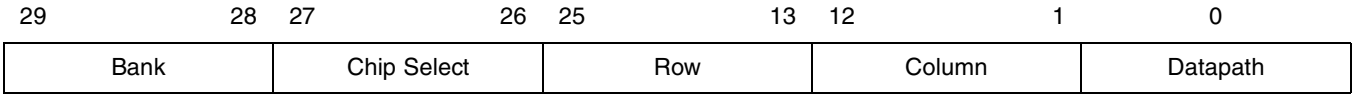


Figure 12-4. Memory Controller Memory Map: Maximum

The ADDR_PINS and COLUMN_SIZE bit fields can each range from their maximum values down to a minimum value defined only by the size of the attached device. This allows the memory controller to function with a wide variety of memory device sizes. The settings for the ADDR_PINS and COLUMN_SIZE bit fields control how the address map is used to decode the user address to the DRAM chip selects and row and column addresses. It is assumed that the values in these bit fields never exceed the maximum values of 13 rows and 12 columns. Using the example shown in Figure 12-4, if the memory

controller is wired to devices with 10 row pins and 11 column bits, the maximum accessible memory space would be reduced. The accessible memory space for this configuration is 64 Mbytes.

The address map for this configuration is shown in Figure 12-5. Note that address bits 26–29 are not used. These bits are ignored when generating the address to the DRAM devices.

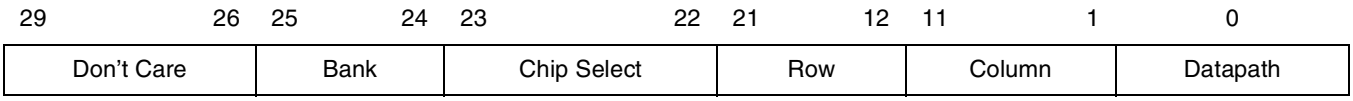


Figure 12-5. Example Memory Map: 10 Row Bits, 11 Column Bits

Note: The Chip Select, Row, Bank, and Column fields are used to address an entire 16-bit memory word. For example, for a read starting at byte address 0x1, the Datapath bit would be a 1 in order to address this byte directly. Reads and writes are 16-bit memory word-aligned if the Datapath bit is 0.

12.2.2.2 Memory Controller Address Control

The available number of accessible memory rows and columns is determined by comparing the maximum values configured with the values programmed into the HW_DRAM_CTL10_ADDR_PINS and HW_DRAM_CTL11_COLUMN_SIZE bit fields. Note that the ADDR_PINS and COLUMN_SIZE bit fields are represented as differences between the maximum configured value and the actual number of pins connected.

The number of connected chip selects and their connection orientation is based on the programming in the HW_DRAM_CTL14_CS_MAP bit field. Because the internal DRAM controller supports up to 4 memory chips, this field is structured to support either one, two, or four memory chips. However, because only 2 memory chip selects are pinned out on the 169BGA package and only 1 memory chip select is pinned out on the 128QFP package, not all CS_MAP configurations can be used.

Below are examples of valid system configurations for the CS_MAP bit field:

- CS_MAP = b0001: One memory device is connected to EMI_CE0n (configuration supported in 128LQFP and 169BGA).
- CS_MAP = b0010: One memory device is connected to EMI_CE1n (configuration supported in 169BGA only).
- CS_MAP = b0011: Two memory devices are connected - one to EMI_CE0n and one to EMI_CE1n. (configuration supported in 169BGA only).

12.2.2.3 Out-of-Range Address Checking

The memory controller is equipped with an out-of-range address checking feature that compares all incoming addresses against the addressable physical memory space. If a transaction is addressed to an out-of-range memory location, then bit 0 of the INT_STATUS bit field is set to 1 to alert the user of this

condition. The memory controller records the address, source ID, length and type of transaction that caused the out-of-range interrupt in the following bit fields:

HW_DRAM_CTL35_OUT_OF_RANGE_ADDR
HW_DRAM_CTL09_OUT_OF_RANGE_SOURCE_ID
HW_DRAM_CTL21_OUT_OF_RANGE_LENGTH
HW_DRAM_CTL09_OUT_OF_RANGE_TYPE

Reading the out-of-range bit fields initiates the memory controller to empty these bit fields and allow them to store out-of-range access information for future errors. The interrupt should be acknowledged by setting bit 0 of the HW_DRAM_CTL16_INT_ACK bit field to 1, which will in turn cause bit 0 of the HW_DRAM_CTL18_INT_STATUS bit field to be cleared to 0.

If a second out-of-range access occurs before the first out-of-range interrupt is acknowledged, then bit 1 of the INT_STATUS bit field is set to 1 to indicate that multiple out-of-range accesses have occurred. If the out-of-range bit fields have been read when the second out-of-range error occurs, then the details for this transaction are stored in the out-of-range bit fields. If they have not been read, then the details of the second error are lost.

Even though the address has been identified as erroneous, the memory controller will still process the read or write transaction. A read transaction will return random data which the user must receive to avoid stalling the memory controller. A write transaction will write the associated data to an unknown location in the memory array, potentially over-writing other stored data. The command cannot be aborted once accepted into the memory controller.

Note that there is no mechanism to indicate an IRQ to the ARM core when this condition occurs. These registers are provided for debugging convenience and can be used with the AHB arbiter debug trap function. To capture an out-of-range error, set an address range with the HW_DIGCTL_DEBUG_TRAP_ADDR_LOW/HIGH registers and enable the trap using HW_DIGCTL_CTRL_TRAP_ENABLE.

12.2.3 Read Data Capture

The read data capture logic is responsible for capturing the DQ outputs from the DRAM devices and passing the data back to the EMI clock domain. The DQS strobes used to capture data are delayed to ensure that the rising and falling edges of the strobes are in the middle of the valid window of data.

DDR (dual data rate) devices send a data strobe (DQS) signal coincident with the read data so that the read data can be reliably captured by the memory controller. The edges of this strobe are aligned with the data output by the DRAM devices. The board traces for the data and the associated data strobe signals should be routed with the same length allowing the rising and falling edges of the data strobe to arrive at the SOC pads.

A delayed version of the data strobe signal must be used to capture the data. The delay added to the data strobe signals should be such that the margin to capture the read data is maximized. Because the frequency of the data strobe signal is matched to the system clock, the delay is a relative number based on the period of the system clock. In the example shown in Figure 12-6, the delay is set to approximately 25% of the system clock. The delay compensation circuit keeps this relative delay constant so that the read data from the DRAM devices can be reliably captured.

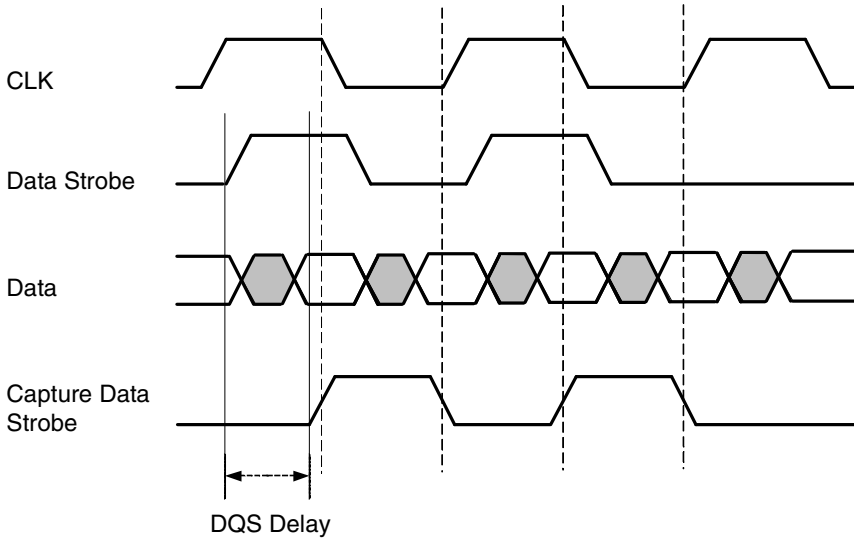


Figure 12-6. DQS Read Timing

12.2.3.1 DQS Gating Control

For the read path, the flight paths must be taken into consideration. There is a certain time lag from when the clock is sent from the memory controller to when the data and DQS signals are received at the memory controller from the memory. Since the DQS from the memory will be sent coincident with the data, and the data must be captured reliably, the DQS signal must be delayed so that it is centered in the data valid window (nominally approximately 1/4 cycle).

The DQS bus is a bidirectional bus that is driven by the memory controller on writes and the memory on reads. When neither device is driving the bus, DQS will remain in a high-impedance state. However, DQS is only relevant to the memory controller during reads in order to capture valid data. For this reason, the DQS signal from memory must be gated so that it is ignored at all other times. Gating of the DQS signal is shown in Figure 12-7.

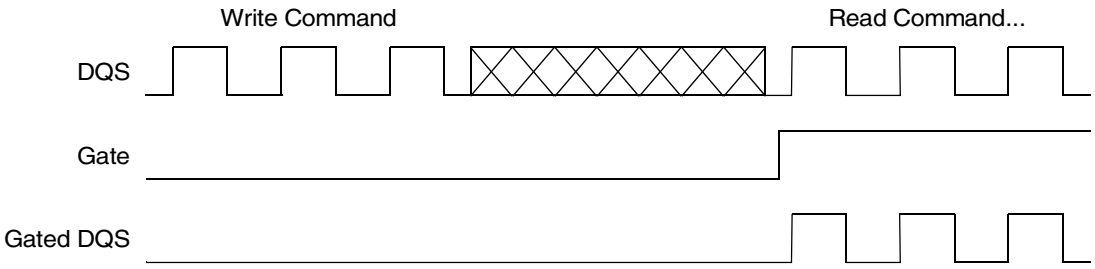


Figure 12-7. DQS Gating

The timing of when to start gating the DQS depends on the design itself, the flight time of the clock to memory, and the flight time of the data/DQS to the memory controller, as follows:

- If the round trip time is between ½ cycle and 1½ cycles, program the *caslat_lin* parameter equal to the *caslat* parameter.
- If the round trip time is less than ½ cycle, program the *caslat_lin* parameter one value less (which translates to ½ cycle) than the *caslat* parameter to open the gate ½ cycle sooner.
- If the round trip time is longer than 1½ cycles, program the *caslat_lin* parameter one value more (which translates to ½ cycle) than the *caslat* parameter to open the gate ½ cycle later.

In addition, the *caslat_lin_gate* parameter controls the opening of the gating signal. Nominally, *caslat_lin_gate* should have the same value as the *caslat_lin* parameter. However, to accommodate the skew of the memory devices, it may be necessary to open the gate a 1/2-cycle sooner or later. Adjusting the value of *caslat_lin_gate* modifies the gate opening by this factor.

There is a requirement that the DQS signals must be known and low when the memory controller is not driving. Because of the large variance in access times for the mobile devices, the gate for the DQS received by the memory controller must be active for longer than the period of time that the memory drives the DQS. Maintaining the DQS bus low when neither the memory controller nor the memory is driving ensures a clean DQS received by the memory controller.

12.2.3.2 mDDR Read Data Timing Registers

When using an mDDR external DRAM device, control of the read data timing is provided through multiple registers, as shown in Figure 12-8. First, the HW_DRAM_CTL04_DLL_BYPASS_MODE selects whether the DCC DLL circuitry is enabled or bypassed. Programming a 1 into this register disables the DLL auto-sync functionality and instead uses a fixed delay-chain select point programmed into the HW_DRAM_CTL19_DLL_DQS_DELAY_BYPASS1 and 0 bit fields. Programming a 0 into the DLL_BYPASS_MODE field enables the DLL auto-sync mode, utilizing the HW_DRAM_CTL18_DLL_DQS_DELAY_BYPASS1 and 0 values to define the percentage of the clock period of delay to add to the DQS inputs before being used as data capture controls.

The BYPASS_MODE or control bit is set based on the desired EMI_CLK frequency. At frequencies above 80 MHz, the BYPASS_MODE should be disabled, allowing the DLL to auto-sync. Frequencies below this point show enable the BYPASS_MODE.

12.2.4 Write Data Timing

DDR DRAM devices require that the DQS data strobe arrive at the DRAM devices within a certain window around the clock. Figure 12-8 describes this relationship. The value for tdqss is specified in fractions of a clock cycle. Most DRAM devices specify this value between ± 0.25 and 0.2 of a clock cycle. This translates to a valid window of between 0.4 and 0.5 of a clock cycle.

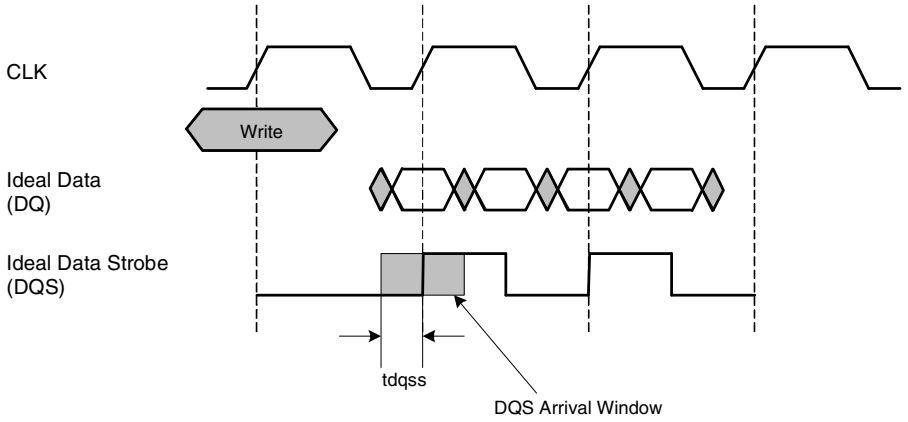


Figure 12-8. DRAM DQS Arrival Time Requirements

The data transfer timing from the memory controller to the DRAM for writes is similar to the read transfer from the DRAM devices to the memory controller. However, there are two differences:

- The DRAM devices expect the data strobe signal to be shifted by the memory controller to allow the DRAM the maximum margin for capturing the data with the data strobe signal.
- The first rising edge of the data strobe signal sent from the memory controller must occur near the rising edge of the clock at the DRAM. This is called the arrival window. DRAM devices typically specify this window as 0.8clk to 1.2clk. Refer to Figure 12-9 for details.

The DCC maintains two delay lines for sending write data and the write data strobe. The first delay line delays the main clock such that the write data strobe transition is as near to the clock edge at the DRAM as possible under typical operating conditions. The second delay line adjusts the clock that is used to output the write data. This clock should be adjusted to maximize the setup and hold requirements around the write strobe.

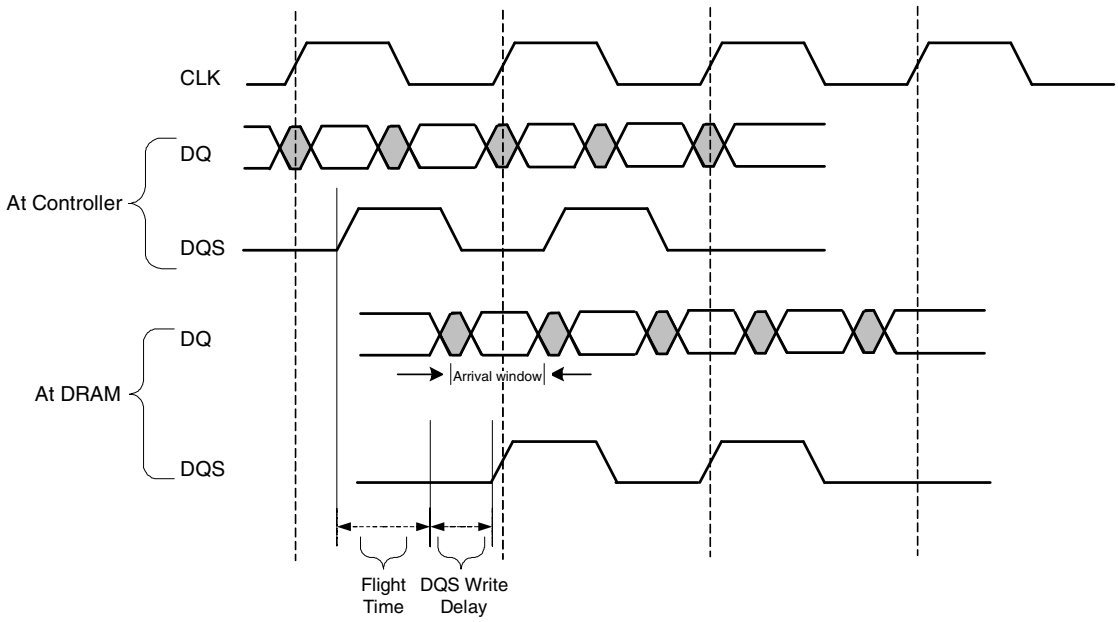


Figure 12-9. DQS Write Timing

Achieving the coincident data strobe signal arrival at a certain point in a clock cycle at the DRAM is a function of the generation of the data strobe signal and the physical delays in transmitting this signal from one point to another. Figure 12-10 illustrates this path in the memory controller.

The write data sent along with the data strobe must be aligned such that the strobe rises and falls within the valid region of the data with maximum setup and hold characteristics. This translates into the write data being clocked 1/4 cycle before the rising edge of the data strobe. This relationship is illustrated in Figure 12-10.

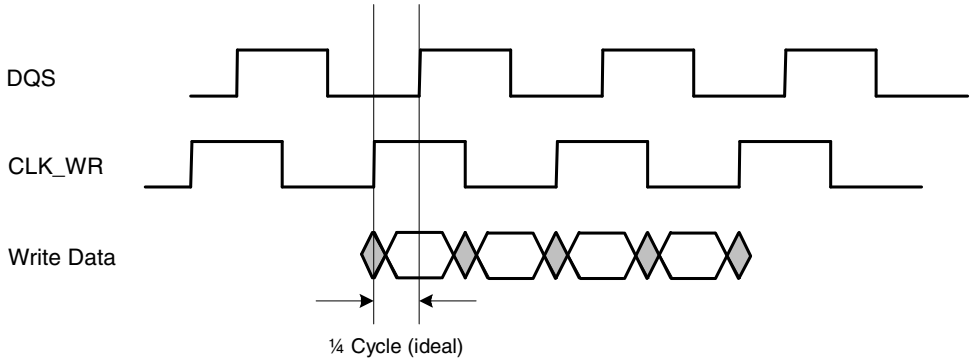


Figure 12-10. Write Data and DQS Relationship

The write data itself originates from a register within the core of the memory controller clocked by the EMI clock. Both the clk_wr and clk_dqs_out signals from the core clock are controlled by the programmable bit fields HW_DRAM_CTL20_WR_DQS_SHIFT and HW_DRAM_CTL19_DQS_OUT_SHIFT,

as shown in Figure 12-11. These bit fields allow these two clocks to be delayed a fixed percentage of the core clock, as illustrated by the example in Figure 12-12.

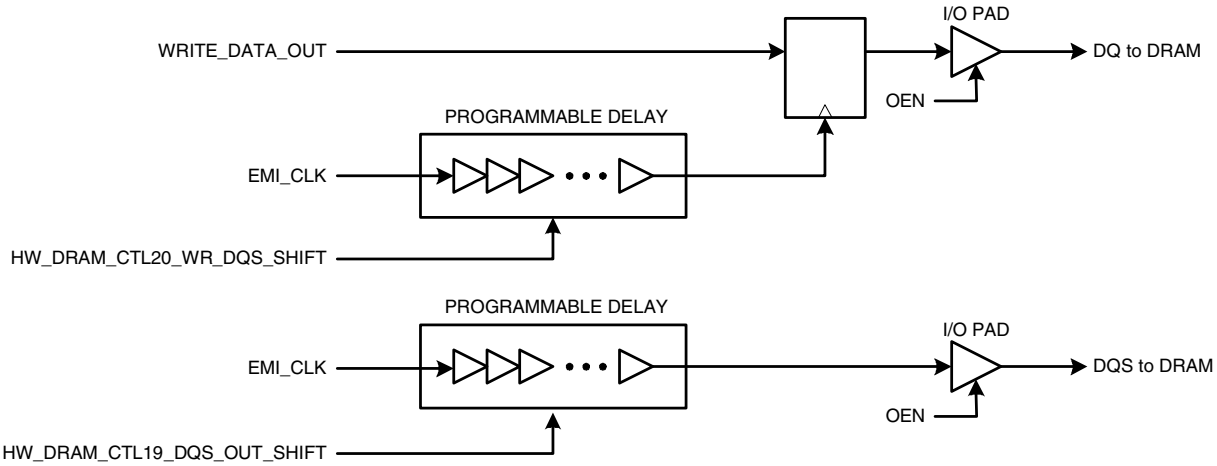


Figure 12-11. Write Data with Programmable Delays

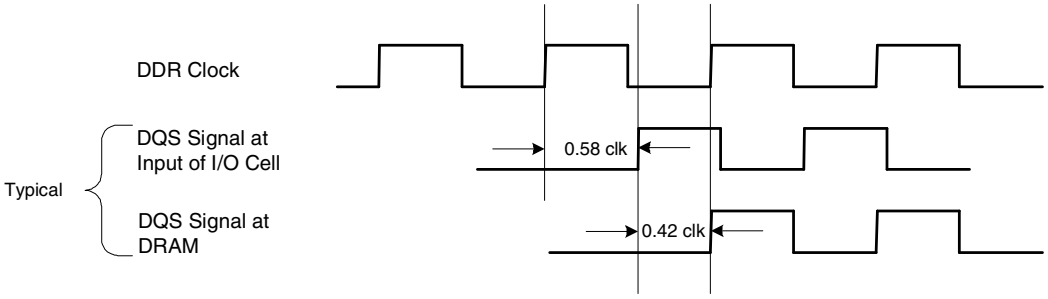


Figure 12-12. WR_DQS_SHIFT Delay Setting Example

12.2.5 DRAM Clock Programmable Delay

The i.MX23 DRAM controller uses an architecture where the address and control signals are launched from the negative edge of the internal EMI clock. The data, DQS, and DM signals are launched from the rising edge of that same clock. At certain higher clock frequencies, this architecture may cause issues with the timing of the signals at the DRAM device relative to the clock itself because the i.MX23 has less flight delay for the clock signals than the address and command signals.

To compensate for this situation, a programmable delay chain is available to delay the output clock to the DRAM device. The delay chain is illustrated in Figure 12-13. This chain consists of 32 delay taps. The delay is voltage-dependent. No other output signals are affected.

The control for this delay is located in the DIGCTL register space in HW_DIGCTL_EMICLK_DELAY_NUM_TAPS. By default, this delay value is 0. In practice, this is not

expected to be used, but it is available as a precaution against high board loads where the address and command signals may not have enough setup time relative to the DRAM clock at the device(s).

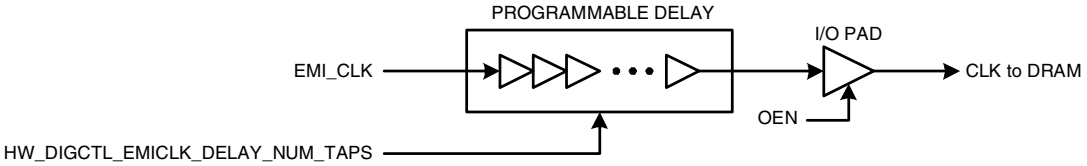


Figure 12-13. DRAM Clock Programmable Delay

12.2.6 Low-Power Operation

In many applications, it is desirable to minimize the power consumption of the memory controller and the memory devices. The memory controller provides various user-configurable low-power options to address power savings. In addition, a partial-array self-refresh option is included for mobile memory devices.

12.2.6.1 Low-Power Modes

There are five low-power modes available in the memory controller. The low-power modes are listed from least to most power saving.

Note: It is not possible to exit one low-power mode and enter another low-power mode simultaneously. The user should plan for a minimum delay between exit and entry between the two low-power modes of 15 cycles in which the memory controller must remain stable.

- **Mode 1: Memory Power-Down**—The memory controller sets the memory devices into power-down, which reduces the overall power consumption of the system, but has the least effect of all the low-power modes. In this mode, the memory controller and memory clocks are fully operational, but the CKE input bit to the memory devices is deasserted. The memory controller continues to monitor memory refresh needs and automatically brings the memory out of power-down to perform these refreshes. When a refresh is required, the CKE input bit to the memory devices is re-enabled. This action brings the memory devices out of power-down. Once the refresh has been completed, the memory devices are returned to power-down by deasserting the CKE input bit.
- **Mode 2: Memory Power-Down with Memory Clock Gating**—The memory controller sets the memory devices into power-down and gates off the clock to the memory devices. Refreshes are handled as in the Memory Power-Down mode (Mode 1), with the exception that gating on the memory clock is removed before asserting the CKE pin. After the refresh has been completed, the memory devices are returned to power-down with the clock gated. Before the memory devices are removed from power-down, the clock is gated on again. Although this mode is supported in both mobile and non-mobile memory devices, clock gating while in power-down is only allowed for mobile memory devices. Therefore, the memory controller will only attempt to gate the clock if it is configured for mobile device operation. For non-mobile memory devices in this low-power

mode, the memory controller operates identically to the Memory Power-Down mode without the clock gating (Mode 1).

- **Mode 3: Memory Self-Refresh**—The memory controller sets the memory devices into self-refresh. In this mode, the memory controller and memory clocks are fully operational and the CKE input bit to the memory devices is deasserted. Since the memory automatically refreshes its contents, the memory controller does not need to send explicit refreshes to the memory.
- **Mode 4: Memory Self-Refresh with Memory Clock Gating**—The memory controller sets the memory devices into self-refresh and gates off the clock to the memory devices. Before the memory devices are removed from self-refresh, the clock is gated on again.
- **Mode 5: Memory Self-Refresh with Memory and Controller Clock Gating**—This is the deepest low-power mode of the memory controller. The memory controller sets the memory devices into self-refresh and gates off the clock to the memory devices. In addition, the clock to the memory controller and the programming bit fields are gated off, except to a small portion of the DLL, which must remain active to maintain the lock. Before the memory devices are removed from self-refresh, the memory controller and memory clocks are gated on.

12.2.6.2 Low-Power Mode Control

The memory controller may enter and exit the various low-power modes in the following ways:

- **Automatic Entry**—When the memory controller is idle, four timing counters begin counting the cycles of inactivity. If any of the counters expires, the memory controller enters the low-power mode associated with that counter.
- **Manual Entry**—The user may initiate any low-power mode by setting the bit of the LOWPOWER_CONTROL bit field associated with the desired mode. The memory controller enters the selected low-power mode when it is has completed its current burst.
- **Hardware Entry**—If the memory pins are being shared between the memory controller and an external source, a handshaking interface is used to control bus activity. The Memory Self-Refresh mode (Mode 3) of the memory controller is used to facilitate the pin sharing.

Automatic and manual entry methods are both controlled by two bit fields: LOWPOWER_CONTROL and LOWPOWER_AUTO_ENABLE located in HW_DRAM_CTL16. The LOWPOWER_CONTROL bit field contains individual enable/disable bits for each low-power mode, and the LOWPOWER_AUTO_ENABLE bit field controls whether each mode is entered automatically or manually.

12.2.6.3 Automatic Entry

Automatic entry occurs if all of the following conditions are true:

- The hardware entry interface is not active or transitioning.
- The mode is programmed for automatic entry by setting the relevant bit in the LOWPOWER_AUTO_ENABLE bit field to 1.
- The particular mode is enabled in the LOWPOWER_CONTROL bit field.
- The memory controller is idle.
- The counter associated with this mode expires.

There are four counters in all to cover the five low-power modes. There are separate counters for each of the three memory self-refresh low-power modes (Modes 3, 4 and 5). Memory Power-Down mode (Mode 1) and Memory Power-Down with Memory Clock Gating mode (Mode 2) share the same counter.

The counters determine the number of idle cycles before entry into the associated low-power mode. All of these counters are re-initialized each time there is a new read or write transaction entering or executing in the memory controller. This ensures that the memory controller does not enter any of the low-power modes when active.

All five low-power modes can be entered through automatic entry and are exited automatically when any of the following conditions occur:

- A new read or write transaction appears at the memory controller interface.
- The memory controller must refresh the memory when in either of the power-down modes (Modes 1 or 2). After completing the memory refresh, the memory controller re-enters power-down.
- The counter for a deeper low-power mode expires. The memory controller must exit the current low-power mode in order to enter the deeper low-power mode. A minimum of 15 cycles occur between exit from one low-power mode before entering into the next low-power mode, even if the counters expire within 15 cycles of each other. Note that the memory controller does not enter a less deep low-power mode, regardless of which counters expire.

12.2.6.4 Manual “On-Demand” Entry

Manual entry occurs if all of the following conditions are true:

- The hardware entry interface is not active or transitioning.
- The mode is programmed for manual entry by clearing the relevant bit in the LOWPOWER_AUTO_ENABLE bit field to 0.
- The particular mode is set to 1 in the LOWPOWER_CONTROL bit field.

For manual entry, the LOWPOWER_CONTROL bit field triggers entry into the low-power modes. The memory controller does not need to be idle when a low-power mode bit is enabled. When a particular mode that is programmed for manual entry is enabled, the memory controller completes the current memory burst access, and then, regardless of the activity inside the memory controller or at the memory interface, it enters the selected low-power mode.

If new transactions enter the memory controller while it is in one of the low-power modes, they accumulate inside the memory controller’s command queue until the queue is full. Exit from a manually-entered low-power mode is also manual. Clearing the LOWPOWER_CONTROL bit field bits to 0 disables the low-power mode of the memory controller, and command processing resumes. In the deepest low-power mode (Mode 5), the clock to the programming registers module is gated off. However, manual low-power mode exit requires the user to clear the LOWPOWER_CONTROL bit field to 0, which is not possible when the clock is off. As a result, the user should not manually activate the deepest low-power mode. If Memory Self-Refresh with Memory and Controller Clock Gating mode (Mode 5) is entered manually, the device cannot be brought out of low-power mode again!

If a different LOWPOWER_CONTROL bit is set to 1 while in one of the low-power modes, or on clearing of the original bit to 0, the memory controller exits the current low-power mode. There will be at least a 15 cycle delay before the memory controller is fully operational or enters the new low-power mode.

NOTE: There is a deadlock possibility that exists when using the manual low-power mode entry. If a read cycle from the ARM core occurs to the DRAM when a manual low-power mode is active, the ARM cycle does not complete. There is no other device within the SOC that can deactivate the low-power mode. Thus, the system will be deadlocked. The same can occur with multiple write cycles that will fill the two-command deep write buffer of the memory controller.

12.2.6.5 Register Programming

The low-power modes of the memory controller are controlled through the LOWPOWER_CONTROL and LOWPOWER_AUTO_ENABLE bit fields in HW_DRAM_CTL16. These five-bit bit fields each contain one bit for controlling each low-power mode. The LOWPOWER_CONTROL bit field enables the associated low-power mode, and the LOWPOWER_AUTO_ENABLE bit field sets the entry method into that mode as manual or automatic. Table 12-1 shows the relationship between the five bits of the lowpower_control and lowpower_auto_enable bit fields and the various low-power modes.

Table 12-1. Low-Power Mode Bit Fields

Low-Power Mode	Enable	Entry
Memory Power-Down (Mode 1)	LOWPOWER_CONTROL [4] =1	LOWPOWER_AUTO_ENABLE [4] • 0 = Manual • 1 = Automatic
Memory Power-Down with Memory Clock Gating (Mode 2)	LOWPOWER_CONTROL [3] =1	LOWPOWER_AUTO_ENABLE [3] • 0 = Manual • 1 = Automatic
Memory Self-Refresh (Mode 3)	LOWPOWER_CONTROL [2] =1	LOWPOWER_AUTO_ENABLE [2] • 0 = Manual • 1 = Automatic
Memory Self-Refresh with Memory Clock Gating (Mode 4)	LOWPOWER_CONTROL [1] =1	LOWPOWER_AUTO_ENABLE [1] • 0 = Manual • 1 = Automatic
Memory Self-Refresh with Memory and Controller Clock Gating (Mode 5)	LOWPOWER_CONTROL [0] =1	LOWPOWER_AUTO_ENABLE [0] • 0 = Manual • 1 = Automatic

When a LOWPOWER_CONTROL bit field bit is set to 1 by the user, the memory controller checks the LOWPOWER_AUTO_ENABLE bit field.

- If the associated bit in the LOWPOWER_AUTO_ENABLE bit field is set to 1, then the memory controller watches the associated counter for expiration, and then enters that low-power mode.Table 12-2 shows the correlation between the low-power modes and the counters that control each mode’s automatic entry.

- If the associated bit in the LOWPOWER_AUTO_ENABLE bit field is cleared to 0, then the memory controller completes its current memory burst access and then enters the specified low-power mode.

Table 12-2. Low-Power Mode Counters

Low-Power Mode	Counter
Memory Power-Down (Mode 1)	HW_DRAM_CTL30_LOWPOWER_POWER_DOWN_CNT
Memory Power-Down with Memory Clock Gating (Mode 2)	HW_DRAM_CTL30_ LOWPOWER_POWER_DOWN_CNT
Memory Self-Refresh (Mode 3)	HW_DRAM_CTL31_LOWPOWER_SELF_REFRESH_CNT
Memory Self-Refresh with Memory Clock Gating (Mode 4)	HW_DRAM_CTL29_LOWPOWER_EXTERNAL_CNT
Memory Self-Refresh with Memory and Controller Clock Gating (Mode 5)	HW_DRAM_CTL29_LOWPOWER_INTERNAL_CNT

Note that the values in the LOWPOWER_AUTO_ENABLE bit field are only relevant when the associated LOWPOWER_CONTROL bit is set to 1.

Multiple bits of the LOWPOWER_CONTROL and LOWPOWER_AUTO_ENABLE bit fields can be set to 1 at the same time. When this happens, the memory controller always enters the deepest low-power mode of all the modes that are enabled. If the memory controller is already in one low-power mode when a deeper low-power mode is requested automatically or manually, it must first exit the current low-power mode, and then enter the deeper low-power mode. A minimum 15-cycle delay occurs before the second entry.

The timing for automatic entry into any of the low-power modes is based on the number of idle cycles that have elapsed in the memory controller. There are four counters related to the five low-power modes to determine when any particular low-power mode will be entered if the automatic entry option is chosen. The counters are also shown in Table 12-2. Since the two power-down modes share one counter, if the user wishes to enter Memory Power-Down mode (Mode 1) automatically, then the Memory Power-Down with Memory Clock Gating mode (Mode 2) must not be enabled.

12.2.6.6 Refresh Masking

Regular refresh commands are issued at the same intervals while the memory controller is operating normally, is idle, or is in any of the low-power modes. However, for memory arrays with multiple chip selects, the memory controller supports the ability to mask refreshes while in any of the low-power modes. By clearing bits of the HW_DRAM_CTL14_LOWPOWER_REFRESH_ENABLE bit field to 0, auto-refreshes will be masked for the associated chip selects. It is the user's responsibility to ensure that refreshes are not constantly masked, and that each chip select is refreshed periodically.

12.2.6.7 Mobile DDR Devices

When using a mobile device, the HW_DRAM_CTL05_EN_LOWPOWER_MODE bit field must be set to 1. This enables the memory controller to use the initialization sequence and EMRS addressing appropriate to mobile devices. When the EN_LOWPOWER_MODE bit field is cleared to 0, a standard DDRS-DRAM device may be used.

12.2.6.8 Partial Array Self-Refresh

For mobile devices, the memory controller is capable of supporting refreshes to subsections of the memory array. To facilitate this capability, separate bit fields are provided to supply the EMRS data for each chip select. These are EMRS_DATA_x bit fields, where X represents the chip select.

Having separate control bit fields for the EMRS data allows the individual chips to set their own masked refresh. The WRITE_MODEREG bit field controls the writing of this EMRS data into the registers. When WRITE_MODEREG is set to 1 initially, the EMRS register of chip select 0 will be written. Each subsequent setting of the WRITE_MODEREG bit field to 1 writes the EMRS register of the next chip select (1, 2, then 3).

Note that the memory controller does not check if operations attempt to access addresses outside of the refresh ranges set by the EMRS registers. Any accesses to these addresses may result in corrupt or lost data.

12.2.7 EMI Clock Frequency Change Requirements

Running the EMI block at different operational frequencies involves changing the DRAM controller timing registers and the EMI clock control registers (Chapter 4, “Clock Generation and Control,” for the EMI clock control registers). To change the EMI frequency safely without losing current memory state, the following steps are required:

1. Call code in non-cached, non-buffered OCRAM or ROM (code cannot be executing out of DRAM).
2. Software saves interrupt enable state and disables interrupts.
3. Software flushes instruction and data caches.
4. Software puts DRAM controller in self-refresh mode.
5. Software writes new DRAM controller timing register values (this step is optional, perform if necessary).
6. Software writes new clock frequency.
7. Software polls for EMI clock stability.
8. Software takes DRAM controller out of self-refresh mode.
9. Software restores saved interrupt enable state.
10. Return.

12.3 Power Management

The EMI has multiple levels of power management. Architectural power management is controlled by bits in the programmable registers. The DRAM controller also has automatic engagement of various power-saving modes that are documented in Section 12.2.6, “Low-Power Operation.”

The highest level of power-savings in the DRAM controller is achieved by enabling the EMI Clock Gate in the EMI Control register. When enabled, access to all PIO registers except the control register’s Soft Reset and Clock Gate bits is disabled, and the DRAM controller is completely shut down. The next step in power savings is the individual DRAM clock gate, which controls the state machines and associated logic. These are also available in the EMI Control Register.

Note: The DRAM control registers have a low-power setting, Level 5, that turns off the DRAM clock inside the controller. It is recommended that Level 5 of the DRAM low-power modes not be used. Instead, use Level 4, which will put the DRAM chip into a self-refresh mode and disable the external clock and CKE. Then, use the EMI Control Register clock gate for the entire EMI or the DRAM-only gate.

The DRAM controller interface, however, has multiple levels of low-power options available. They are, in increasing order of power savings:

1. Memory Power-Down—Controller and EMI_CLK are active, but EMI_CKE is pulled low to the memory devices.
2. Memory Power-Down with Memory Clock Gating—The controller clock remains active, but the EMI_CLK is gated off, and EMI_CKE is pulled low.
3. Memory Self-Refresh—The controller puts the memory devices into self-refresh mode. The controller clock and EMI_CLK remain active, but EMI_CKE is pulled low.
4. Memory Self-Refresh with Memory Clock Gating—The controller puts the memory devices into self-refresh mode. The controller clock remains active, but the EMI_CLK is gated off, and EMI_CKE is pulled low.
5. Memory Self-Refresh with Controller and Memory Clock Gating—The controller puts the memory devices into self-refresh mode. Then, the controller and EMI_CLK are gated off. The only clock that remains active is the clock to the DLL, which must remain active to maintain DLL lock.

The DRAM controller can be programmed to enter these modes automatically, or they can be entered manually via control register accesses. It is expected that Freescale software will set up the DRAM controller to automatically enter modes 1 and 2, but that modes 3–4 would be entered manually after specific requests from the software. Avoid using Level 5.

12.4 AXI/AHB Port Arbitration

The EMI port arbiter supports three operational arbitration modes. The arbiter is provided with PIO control fields, including a two bit HW_EMI_CTRL_ARB_MODE field, which selects one of the three modes. The three arbitration modes are described below.

12.4.1 Legacy Timestamp Mode

When commands are logged into one of the four command queue channels, they are issued a 6-bit sequential count or timestamp. The commands in these four independent channels are then granted access to the downstream controller placement queue in strict timestamp order. The grant decision is simple and is purely combinational in design. Waiting commands can be granted every cycle provided that the placement queue isn't full.

12.4.2 Timestamp/write-priority Hybrid Mode

This new arbitration mode consists of cycling through 2 different priority modes: the legacy timestamp mode (described above) and a new write priority mode. The arbiter first grants a requesting channel based on the timestamp scheme and then goes into the write priority mode. There it loops though all the high priority write channels (3 of the channels can be programmed as high priority write: AXI0, AHB2 and AHB3) a programmed number of iterations granting pending write operations only. It then goes back to timestamp priority mode to grant the operation with the next oldest timestamp. The cycle thus continues alternating between the timestamp and write priority modes. It is important to remember that in the write-priority mode one iteration means to loop through all high priority ports once granting ports with pending commands. And the order in which we consider (or scan through) each port is fixed. The hardware loops through the ports 2, 3 and 0 granting pending commands, in that order. This constitutes one iteration. And this is done repeatedly for the number of iterations programmed. The ordering was set by considering the importance, or priority, of writes of the various masters attached to these busses and therefore is chip specific.

The arbiter receives three 1-bit high priority write masks (HW_EMI_CTRL_HIGH_PRIORITY_WRITE) which select the ports to be given high priority write status. It is also be provided with a 3-bit HP write loop count (HW_EMI_CTRL_PRIORITY_WRITE_ITER), which indicates the maximum number of iterations through the write loop. The write loop will exit when there are no more high priority writes available or when the loop counter reaches the maximum loop count value. The high priority write loop could be skipped if no HP writes are pending. The maximum loop count allowed is 5. These two parameters are PIO programmable.

In practice, software would likely set the ARM data port to have high priority write status and the maximum loop counter would likely be programmed to a value of at least two. This would give highly preferential treatment to ARM data writes and ensure that they get additional commands into the memory controller's placement queue ahead of all other commands. By looping on the writes, we also enable the memory controller to get a stream of write operations that should improve efficiency. Since the arbiter moves back into the timestamp loop periodically, low-priority ports should still have reasonable access to the placement queue. If problems with starvation occur, the maximum loop counter should be programmed to a lower value.

12.4.3 Port Priority Mode

This mode requires the user to program the ports to have a highest to lowest priority. When multiple ports have commands pending, the port with the highest priority will always be granted without regard to timestamp. This mode does not address the problem of possible port starvation.

This mode doesn't inherently guarantee that a read cannot get out of order with a previously issued address-paired write and return stale data. The two previously defined modes guarantee this. However, based on the typical use case, this scenario should only occur on certain port pairs. So if care is taken when programming the port priorities this mode will avoid such errors in this typical case. This mode is very simple, efficient and fully programmable. For this mode it is recommended that the default value be used for the HW_EMI_CTRL_PORT_PRIORITY_ORDER field.

12.5 Programmable Registers

This section describes the programmable registers of the external memory interface (EMI).

12.5.1 EMI Control Register Description

EMI Interface Control Register.

HW_EMI_CTRL	0x000
HW_EMI_CTRL_SET	0x004
HW_EMI_CTRL_CLR	0x008
HW_EMI_CTRL_TOG	0x00C

Table 12-3. HW_EMI_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	RSVD6	TRAP_SR	TRAP_INIT	AXI_DEPTH		DLL_SHIFT_RESET	DLL_RESET	ARB_MODE		RSVD5	PORT_PRIORITY_ORDER					RSVD4	PRIORITY_WRITE_ITER			RSVD3	HIGH_PRIORITY_WRITE			RSVD2	MEM_WIDTH	RSVD1	RESET_OUT	RSVD0			

Table 12-4. HW_EMI_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Reset EMI register block. 0 = EMI controller is not reset. 1 = EMI controller is reset. Note: This soft reset only affects the EMI registers . There is also a soft-reset for the DRAM controller in the DRAM registers.
30	**RSVD6**	RO	0x0	Reserved
29	**TRAP_SR**	RW	0x0	When set, causes an AHB ERROR response on any access to the DRAM memory space if the DRAM controller is in Self-Refresh mode.
28	**TRAP_INIT**	RW	0x1	When set, causes an AHB ERROR response on any access to the DRAM memory space if the DRAM controller has not been initialized (specifically when START is not set).
27:26	**AXI_DEPTH**	RW	0x3	Specifies the number of commands allowed in the AXI port queue. ONE = 0x0 Allow only one command. TWO = 0x1 Allow two commands. THREE = 0x2 Allow three commands. FOUR = 0x3 Allow four commands.
25	**DLL_SHIFT_RESET**	RW	0x0	When set, forces the DRAM controller DLL startpoint shift logic into a reset state.
24	**DLL_RESET**	RW	0x0	When set, forces the DRAM controller into a reset state.
23:22	**ARB_MODE**	RW	0x0	This field sets the arbitration mode for the DRAM port controller. The supported arbitration schemes are: simple timestamp priority method; enhanced high-priority write and timestamp hibrid method; and port priority method. The programming options are: TIMESTAMP = 0x0 Timestamp Priority (37xx arbitration) WRITE_HYBRID = 0x1 Write Priority Hybrid PORT_PRIORITY = 0x2 Fixed Port Priority
21	**RSVD5**	RO	0x0	Reserved
20:16	**PORT_PRIORITY_ORDER**	RW	0x8	This field specifies the priority order 1-4 (1= highest priority) of the 4 arbitrated ports. The field values define the following order (highest to lowest): (NOTE: Values 0x18-0x1F select PORT1230.) PORT0123 = 0x00 Priority Order: AXI0, AHB1, AHB2, AHB3 PORT0312 = 0x01 Priority Order: AXI0, AHB3, AHB1, AHB2 PORT0231 = 0x02 Priority Order: AXI0, AHB2, AHB3, AHB1 PORT0321 = 0x03 Priority Order: AXI0, AHB3, AHB2, AHB1 PORT0213 = 0x04 Priority Order: AXI0, AHB2, AHB1, AHB3 PORT0132 = 0x05 Priority Order: AXI0, AHB1, AHB3, AHB2 PORT1023 = 0x06 Priority Order: AHB1, AXI0, AHB2, AHB3 PORT1302 = 0x07 Priority Order: AHB1, AHB3, AXI0, AHB2 PORT1230 = 0x08 Priority Order: AHB1, AHB2, AHB3, AXI0 PORT1320 = 0x09 Priority Order: AHB1, AHB3, AHB2, AXI0 PORT1203 = 0x0A Priority Order: AHB1, AHB2, AXI0, AHB3 PORT1032 = 0x0B Priority Order: AHB1, AXI0, AHB3, AHB2 PORT2013 = 0x0C Priority Order: AHB2, AXI0, AHB1, AHB3 PORT2301 = 0x0D Priority Order: AHB2, AHB3, AXI0, AHB1 PORT2130 = 0x0E Priority Order: AHB2, AHB1, AHB3, AXI0 PORT2310 = 0x0F Priority Order: AHB2, AHB3, AHB1, AXI0 PORT2103 = 0x10 Priority Order: AHB2, AHB1, AXI0, AHB3 PORT2031 = 0x11 Priority Order: AHB2, AXI0, AHB3, AHB1 PORT3012 = 0x12 Priority Order: AHB3, AXI0, AHB1, AHB2 PORT3201 = 0x13 Priority Order: AHB3, AHB2, AXI0, AHB1 PORT3120 = 0x14 Priority Order: AHB3, AHB1, AHB2, AXI0 PORT3210 = 0x15 Priority Order: AHB3, AHB2, AHB1, AXI0 PORT3102 = 0x16 Priority Order: AHB3, AHB1, AXI0, AHB2 PORT3021 = 0x17 Priority Order: AHB3, AXI0, AHB2, AHB1

Table 12-4. HW_EMI_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15	**RSVD4**	RO	0x0	Reserved
14:12	**PRIORITY_WRITE_ITER**	RW	0x4	When the hybrid port arbitration scheme is enabled, this field specifies how many times to iterate through the high-priority write phase. This field's range is 1-5 iterations. NOTE: The number of iterations is this field value plus 1.
11	**RSVD3**	RO	0x0	Reserved
10:8	**HIGH_PRIORITY_WRITE**	RW	0x0	Specifies which AHB ports to the EMI have high write priority when the enhanced memory arbitration scheme is enabled. When set bits 12-14 specify high priority for AHB0, AHB2 and AHB3 respectively. The ports are defined as follows: AXI0 = DCP/BCH/PXP port, AHB2 = ARM Data, AHB3 = USB/DMA/ECC8.
7	**RSVD2**	RO	0x0	Reserved
6	**MEM_WIDTH**	RW	0x1	0 = 8-bit memory. 1 = 16-bit memory.
5	**RSVD1**	RO	0x0	Reserved
4	**RESET_OUT**	RW	0x0	0 = Reset output is low. 1 = Reset output is high.
3:0	**RSVD0**	RO	0x0	Reserved

DESCRIPTION:

The EMI Control register is used to control several high-level items related to the EMI controller. This register should be used in conjunction with the DRAM register bits.

EXAMPLE:

```
Empty Example.
```

12.5.2 EMI Version Register Description

This register always returns a known read value for debug purposes. It indicates the version of the block.

HW_EMI_VERSION 0x0F0

Table 12-5. HW_EMI_VERSION

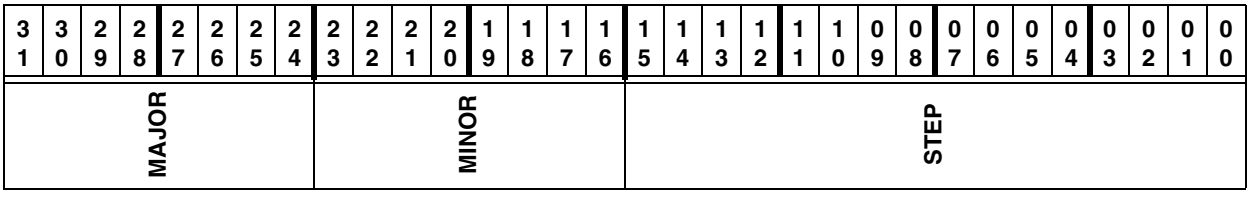


Table 12-6. HW_EMI_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**MAJOR**	RO	0x02	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	**MINOR**	RO	0x01	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	**STEP**	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

DESCRIPTION:

This register indicates the RTL version in use.

EXAMPLE:

```
Empty Example.
```

EMI Block v2.1, Revision 1

12.5.3 DRAM Control Register 00 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL00 0x000

Table 12-7. HW_DRAM_CTL00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4							AHB0_W_PRIORITY	RSVD3							AHB0_R_PRIORITY	RSVD2							AHB0_FIFO_TYPE_REG	RSVD1							ADDR_CMP_EN

Table 12-8. HW_DRAM_CTL00 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD4**	RO	0x0	Reserved.
24	**AHB0_W_PRIORITY**	RW	0x0	Priority of write commands from port 0. Sets the priority of write commands from AHB port 0 relative to the other AHB Ports. A value of 0 is the highest priority.
23:17	**RSVD3**	RO	0x0	Reserved.
16	**AHB0_R_PRIORITY**	RW	0x0	Priority of read commands from port 0. Sets the priority of read commands from AHB port 0 relative to the other AHB Ports. A value of 0 is the highest priority.

Table 12-8. HW_DRAM_CTL00 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:9	**RSVD2**	RO	0x0	Reserved.
8	**AHB0_FIFO_TYPE_REG**	RW	0x0	Clock domain relativity between port 0 and memory controller core. Sets the relativity of the clock domains between AHB port 0 and the memory controller core clock. 0 = Asynchronous 1 = Synchronous
7:1	**RSVD1**	RO	0x0	Reserved.
0	**ADDR_CMP_EN**	RW	0x0	Enable address collision detection for command queue placement logic. Enables address collision/data coherency detection as a condition when using the placement logic to fill the command queue. 0 = Disabled 1 = Enabled

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.4 DRAM Control Register 01 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL01 0x004

Table 12-9. HW_DRAM_CTL01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4							AHB2_FIFO_TYPE_REG	RSVD3							AHB1_W_PRIORITY	RSVD2							AHB1_R_PRIORITY	RSVD1							AHB1_FIFO_TYPE_REG

Table 12-10. HW_DRAM_CTL01 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD4**	RO	0x0	Reserved.
24	**AHB2_FIFO_TYPE_REG**	RW	0x0	Clock domain relativity between port 2 and memory controller core. Sets the relativity of the clock domains between AHB port 2 and the memory controller core clock. 0 = Asynchronous 1 = Synchronous
23:17	**RSVD3**	RO	0x0	Reserved.
16	**AHB1_W_PRIORITY**	RW	0x0	Priority of write commands from port 1. Sets the priority of write commands from AHB port 1 relative to the other AHB Ports. A value of 0 is the highest priority.
15:9	**RSVD2**	RO	0x0	Reserved.
8	**AHB1_R_PRIORITY**	RW	0x0	Priority of read commands from port 1. Sets the priority of read commands from AHB port 1 relative to the other AHB Ports. A value of 0 is the highest priority.
7:1	**RSVD1**	RO	0x0	Reserved.
0	**AHB1_FIFO_TYPE_REG**	RW	0x0	Clock domain relativity between port 1 and memory controller core. Sets the relativity of the clock domains between AHB port 1 and the memory controller core clock. 0 = Asynchronous 1 = Synchronous

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.5 DRAM Control Register 02 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL02 0x008

Table 12-11. HW_DRAM_CTL02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4							AHB3_R_PRIORITY	RSVD3							AHB3_FIFO_TYPE_REG	RSVD2							AHB2_W_PRIORITY	RSVD1							AHB2_R_PRIORITY

Table 12-12. HW_DRAM_CTL02 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD4**	RO	0x0	Reserved.
24	**AHB3_R_PRIORITY**	RW	0x0	Priority of read commands from port 3. Sets the priority of read commands from AHB port 3 relative to the other AHB Ports. A value of 0 is the highest priority.
23:17	**RSVD3**	RO	0x0	Reserved.
16	**AHB3_FIFO_TYPE_REG**	RW	0x0	Clock domain relativity between port 3 and memory controller core. Sets the relativity of the clock domains between AHB port 3 and the memory controller core clock. 0 = Asynchronous 1 = Synchronous
15:9	**RSVD2**	RO	0x0	Reserved.
8	**AHB2_W_PRIORITY**	RW	0x0	Priority of write commands from port 2. Sets the priority of write commands from AHB port 2 relative to the other AHB Ports. A value of 0 is the highest priority.
7:1	**RSVD1**	RO	0x0	Reserved.
0	**AHB2_R_PRIORITY**	RW	0x0	Priority of read commands from port 2. Sets the priority of read commands from AHB port 2 relative to the other AHB Ports. A value of 0 is the highest priority.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.6 DRAM Control Register 03 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL03 0x00C

Table 12-13. HW_DRAM_CTL03

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4							AUTO_REFRESH_MODE	RSVD3							AREFRESH	RSVD2							AP	RSVD1							AHB3_W_PRIORITY

Table 12-14. HW_DRAM_CTL03 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD4**	RO	0x0	Reserved.
24	**AUTO_REFRESH_MODE**	RW	0x0	Controls whether auto-refresh will be at next burst or next command boundary. Sets the mode for when the automatic refresh will occur. If auto_refresh_mode is set and a refresh is required to memory, the memory controller will delay this refresh until the end of the current transaction (if the transaction is fully contained inside a single page), or until the current transaction hits the end of the current page. 0 = Issue refresh on the next DRAM burst boundary, even if the current command is not complete. 1 = Issue refresh on the next command boundary.
23:17	**RSVD3**	RO	0x0	Reserved.
16	**AREFRESH**	WO	0x0	Initiate auto-refresh when specified by AUTO_REFRESH_MODE. Initiates an automatic refresh to the DRAM devices based on the setting of the AUTO_REFRESH_MODE bit field. If there are any open banks when this bit field is set, the memory controller will automatically close these banks before issuing the auto-refresh command. This bit field will always read back 0. 0 = No action 1 = Issue refresh to the DRAM devices
15:9	**RSVD2**	RO	0x0	Reserved.
8	**AP**	RW	0x0	Enable auto pre-charge mode of controller. Enables auto pre-charge mode for DRAM devices. NOTE: This bit field may not be modified after the START bit field has been asserted. 0 = Auto pre-charge mode disabled. Memory banks will stay open until another request requires this bank, the maximum open time (tras_max) has elapsed, or a refresh command closes all the banks. 1 = Auto pre-charge mode enabled. All read and write transactions must be terminated by an auto pre-charge command. If a transaction consists of multiple read or write bursts, only the last command is issued with an auto pre-charge.
7:1	**RSVD1**	RO	0x0	Reserved.
0	**AHB3_W_PRIORITY**	RW	0x0	Priority of write commands from port 3. Sets the priority of write commands from AHB port 3 relative to the other AHB Ports. A value of 0 is the highest priority.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.7 DRAM Control Register 04 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL04 0x010

Table 12-15. HW_DRAM_CTL04

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4							DLL_BYPASS_MODE	RSVD3							DLLLOCKREG	RSVD2							CONCURRENTAP	RSVD1							BANK_SPLIT_EN

Table 12-16. HW_DRAM_CTL04 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD4**	RO	0x0	Reserved.
24	**DLL_BYPASS_MODE**	RW	0x0	Enable the DLL bypass feature of the controller. Defines the behavior of the DLL bypass logic and establishes which set of delay parameters will be used. 0 = The values programmed in the DLL_DQS_DELAY_X, DQS_OUT_SHIFT, and WR_DQS_SHIFT are used. These parameters add fractional increments of the clock to the specified lines. 1 = The values programmed into the DLL_DQS_DELAY_BYPASS_X, DQS_OUT_SHIFT_BYPASS, and WR_DQS_SHIFT_BYPASS are used. These parameters specify the actual number of delay elements added to each of the lines. If the total delay time programmed into the delay parameters exceeds the number of delay elements in the delay chain, then the delay will be set to the maximum number of delay elements in the delay chain. 0 = Normal operational atuo-sync mode. 1 = Bypass the auto-sync DLL master delay line.
23:17	**RSVD3**	RO	0x0	Reserved.
16	**DLLLOCKREG**	RO	0x0	Status of DLL lock coming out of master delay. DLL lock/unlock.
15:9	**RSVD2**	RO	0x0	Reserved.
8	**CONCURRENTAP**	RW	0x0	Allow controller to issue commands to other banks while a bank is in auto pre-charge. Enables concurrent auto pre-charge. Some DRAM devices do not allow one bank to be auto pre-charged while another bank is reading or writing. The JEDEC standard allows concurrent auto pre-charge. Set this parameter for the DRAM device being used. 0 = Concurrent auto pre-charge disabled. 1 = Concurrent auto pre-charge enabled.

Table 12-16. HW_DRAM_CTL04 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:1	**RSVD1**	RO	0x0	Reserved.
0	**BANK_SPLIT_EN**	RW	0x0	Enable bank splitting for command queue placement logic. Enables bank splitting as a condition when using the placement logic to fill the command queue. 0 = Disabled 1 = Enabled

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.8 DRAM Control Register 05 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL05 0x014

Table 12-17. HW_DRAM_CTL05

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4							INTRPTREADA	RSVD3							INTRPTAPBURST	RSVD2							FAST_WRITE	RSVD1							EN_LOWPOWER_MODE

Table 12-18. HW_DRAM_CTL05 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD4**	RO	0x0	Reserved.
24	**INTRPTREADA**	RW	0x0	Allow the controller to interrupt a combined read with auto pre-charge command with another read command. Enables interrupting of a combined read with auto pre-charge command with another read command to the same bank before the first read command is completed. 0 = Disable interrupting the combined read with auto pre-charge command with another read command to the same bank. 1 = Enable interrupting the combined read with auto pre-charge command with another read command to the same bank.

Table 12-18. HW_DRAM_CTL05 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23:17	**RSVD3**	RO	0x0	Reserved.
16	**INTRPTAPBURST**	RW	0x0	Allow the controller to interrupt an auto pre-charge command with another command. Enables interrupting an auto pre-charge command with another command for a different bank. If enabled, the current operation will be interrupted. However, the bank will be pre-charged as if the current operation were allowed to continue. 0 = Disable interrupting an auto pre-charge operation on a different bank. 1 = Enable interrupting an auto pre-charge operation on a different bank.
15:9	**RSVD2**	RO	0x0	Reserved.
8	**FAST_WRITE**	RW	0x0	Sets when write commands are issued to DRAM devices. Controls when the write commands are issued to the DRAM devices. 0 = The memory controller will issue a write command to the DRAM devices when it has received enough data for one DRAM burst. In this mode, write data can be sent in any cycle relative to the write command. This mode also allows for multi-word write command data to arrive in non-sequential cycles. 1 = The memory controller will issue a write command to the DRAM devices after the first word of the write data is received by the memory controller. The first word can be sent at any time relative to the write command. In this mode, multi-word write command data must be available to the memory controller in sequential cycles.
7:1	**RSVD1**	RO	0x0	Reserved.
0	**EN_LOWPOWER_MODE**	RW	0x0	Enable low-power mode in controller. Enables the low-power mode of the memory controller. 0 = Disabled 1 = Enabled

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.9 DRAM Control Register 06 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL06 0x018

Table 12-19. HW_DRAM_CTL06

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4							POWER_DOWN	RSVD3							PLACEMENT_EN	RSVD2							NO_CMD_INIT	RSVD1							INTRPTWRITEA

Table 12-20. HW_DRAM_CTL06 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD4**	RO	0x0	Reserved.
24	**POWER_DOWN**	RW	0x0	Disable clock enable and set DRAMs in power-down state. When this bit field is written with a 1, the memory controller will complete processing of the current burst for the current transaction (if any), issue a pre-charge all command and then disable the clock enable signal to the DRAM devices. Any subsequent commands in the command queue will be suspended until this bit field is written with a 0. 0 = Enable full power state. 1 = Disable the clock enable and power down the memory controller.
23:17	**RSVD3**	RO	0x0	Reserved.
16	**PLACEMENT_EN**	RW	0x0	Enable placement logic for command queue. Enables using the placement logic to fill the command queue. 0 = Placement logic is disabled. The command queue is a straight FIFO. 1 = Placement logic is enabled. The command queue will be filled according to the placement logic factors.
15:9	**RSVD2**	RO	0x0	Reserved.
8	**NO_CMD_INIT**	RW	0x0	Disable DRAM commands until TDLL has expired during initialization. Disables DRAM commands until DLL initialization is complete and tdll has expired. 0 = Issue only REF and PRE commands during DLL initialization of the DRAM devices. 1 = Do not issue any type of command during DLL initialization of the DRAM devices.
7:1	**RSVD1**	RO	0x0	Reserved.
0	**INTRPTWRITEA**	RW	0x0	Allow the controller to interrupt a combined write with auto pre-charge command with another write command. Enables interrupting of a combined write with auto pre-charge command with another read or write command to the same bank before the first write command is completed. 0 = Disable interrupting a combined write with auto pre-charge command with another read or write command to the same bank. 1 = Enable interrupting a combined write with auto pre-charge command with another read or write command to the same bank.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.10 DRAM Control Register 07 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL07 0x01C

Table 12-21. HW_DRAM_CTL07

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4							RW_SAME_EN	RSVD3							REG_DIMM_ENABLE	RSVD2							RD2RD_TURN	RSVD1							PRIORITY_EN

Table 12-22. HW_DRAM_CTL07 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD4**	RO	0x0	Reserved.
24	**RW_SAME_EN**	RW	0x0	Enable read/write grouping for command queue placement logic. Enables read/write grouping as a condition when using the placement logic to fill the command queue. 0 = Disabled 1 = Enabled
23:17	**RSVD3**	RO	0x0	Reserved.
16	**REG_DIMM_ENABLE**	RW	0x0	Enable registered DIMM operation of the controller. Enables registered DIMM operations to control the address and command pipeline of the memory controller. 0 = Normal operation 1 = Enable registered DIMM operation
15:9	**RSVD2**	RO	0x0	Reserved.

Table 12-22. HW_DRAM_CTL07 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	**RD2RD_TURN**	RW	0x0	Enable insertion of addition turn around clock for back to back reads to different css. Adds an additional clock between back-to-back read operations. The extra clock is required for mobile DDR devices where tac_max > (period/2 + tac_min). Without this additional clock, the first read may drive DQS out at tac_max and the second read may drive DQS out at tac_min, resulting in a contention on the DQS line. 0 = Disabled 1 = Enabled
7:1	**RSVD1**	RO	0x0	Reserved.
0	**PRIORITY_EN**	RW	0x0	Enable priority for command queue placement logic. Enables priority as a condition when using the placement logic to fill the command queue. 0 = Disabled 1 = Enabled

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.11 DRAM Control Register 08 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL08 0x020

Table 12-23. HW_DRAM_CTL08

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4							TRAS_LOCKOUT	RSVD3							START	RSVD2							SREFRESH	RSVD1							SDR_MODE

Table 12-24. HW_DRAM_CTL08 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD4**	RO	0x0	Reserved.
24	**TRAS_LOCKOUT**	RW	0x0	Allow the controller to execute auto pre-charge commands before TRAS_MIN expires. Defines the tRAS lockout setting for the DRAM device. tRAS lockout allows the memory controller to execute auto pre-charge commands before the TRAS_MIN parameter has expired. 0 = tRAS lockout not supported by memory device. 1 = tRAS lockout supported by memory device.
23:17	**RSVD3**	RO	0x0	Reserved.
16	**START**	RW	0x0	Initiate command processing in the controller. With this bit field cleared to 0, the memory controller will not issue any commands to the DRAM devices or respond to any signal activity except for reading and writing bit fields. Once this bit field is set to 1, the memory controller will respond to inputs from the ASIC. When set, the memory controller begins its initialization routine. When the interrupt bit in the INT_STATUS bit field associated with completed initialization is set, the user may begin to submit transactions. 0 = Controller is not in active mode. 1 = Initiate active mode for the memory controller.
15:9	**RSVD2**	RO	0x0	Reserved.
8	**SREFRESH**	RW	0x0	Place DRAMs in self-refresh mode. When this bit field is written with a 1, the DRAM device(s) will be placed in self-refresh mode. For this, the current burst for the current transaction (if any) will complete, all banks will be closed, the self-refresh command will be issued to the DRAM, and the clock enable signal will be de-asserted. The system will remain in self-refresh mode until this bit field is written with a 0. The DRAM devices will return to normal operating mode after the self-refresh exit time (txsr) of the device and any DLL initialization time for the DRAM is reached. The memory controller will resume processing of the commands from the interruption point. This bit field will be updated with an assertion of the srefresh_enter pin, regardless of the behavior on the register interface. To disable self-refresh again after a srefresh_enter pin assertion, the user will need to clear the bit field to 0. 0 = Disable self-refresh mode. 1 = Initiate self-refresh of the DRAM devices.
7:1	**RSVD1**	RO	0x0	Reserved.
0	**SDR_MODE**	RW	0x0	Select SDR or DDR mode of the controller. Selects between SDR (single data rate) and DDR (dual data rate) modes. 0 = DDR mode 1 = SDR mode

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.12 DRAM Control Register 09 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL09 0x024

Table 12-25. HW_DRAM_CTL09

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4						OUT_OF_RANGE_TYPE		RSVD3						OUT_OF_RANGE_SOURCE_ID		RSVD2							WRITE_MODEREG	RSVD1							WRITEINTERP

Table 12-26. HW_DRAM_CTL09 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSVD4**	RO	0x0	Reserved.
25:24	**OUT_OF_RANGE_TYPE**	RO	0x0	Type of command that caused an Out-of-Range interrupt. Holds the type of command that caused an out-of-range interrupt request to the memory devices.
23:18	**RSVD3**	RO	0x0	Reserved.
17:16	**OUT_OF_RANGE_SOURCE_ID**	RO	0x0	Source ID of command that caused an Out-of-Range interrupt. Holds the Source ID of the command that caused an out-of-range interrupt request to the memory devices.
15:9	**RSVD2**	RO	0x0	Reserved.
8	**WRITE_MODEREG**	WO	0x0	Write EMRS data to the DRAMs. Supplies the EMRS data for each chip select to allow individual chips to set masked refreshing. When this bit field is written with a 1, the mode bit field(s) [EMRS register] within the DRAM devices will be written. Each subsequent write_modereg setting will write the EMRS register of the next chip select. This bit field will always read back as 0. The mode registers are automatically written at initialization of the memory controller. There is no need to initiate a mode register write after setting the START bit field in the memory controller unless some value in these registers needs to be changed after initialization. Note: This bit field may not be changed when the memory is in power-down mode (when the CKE input is de-asserted).

Table 12-26. HW_DRAM_CTL09 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:1	**RSVD1**	RO	0x0	Reserved.
0	**WRITEINTERP**	RW	0x0	Allow controller to interrupt a write bursts to the DRAMs with a read command. Defines whether the memory controller can interrupt a write burst with a read command. Some memory devices do not allow this functionality. 0 = The device does not support read commands interrupting write commands. 1 = The device does support read commands interrupting write commands.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.13 DRAM Control Register 10 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL10 0x028

Table 12-27. HW_DRAM_CTL10

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4					AGE_COUNT			RSVD3					ADDR_PINS			RSVD2						TEMRS		RSVD1							Q_FULLNESS	

Table 12-28. HW_DRAM_CTL10 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSVD4**	RO	0x0	Reserved.
26:24	**AGE_COUNT**	RW	0x0	Initial value of master aging-rate counter for command aging. Holds the initial value of the master aging-rate counter. When using the placement logic to fill the command queue, the command aging counters will be decremented one each time the master aging-rate counter counts down age_count cycles.
23:19	**RSVD3**	RO	0x0	Reserved.

Table 12-28. HW_DRAM_CTL10 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
18:16	**ADDR_PINS**	RW	0x0	Difference between number of address pins available and number being used. Defines the difference between the maximum number of address pins configured (13) and the actual number of pins being used. The user address is automatically shifted so that the user address space is mapped contiguously into the memory map based on the value of this bit field.
15:10	**RSVD2**	RO	0x0	Reserved.
9:8	**TEMRS**	RW	0x0	DRAM TEMRS parameter in cycles. Defines the DRAM extended mode parameter set time, in cycles.
7:2	**RSVD1**	RO	0x0	Reserved.
1:0	**Q_FULLNESS**	RW	0x0	Quantity that determines command queue full. Defines quantity of data that will be considered full for the command queue.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.14 DRAM Control Register 11 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL11 0x02C

Table 12-29. HW_DRAM_CTL11

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4					MAX_CS_REG			RSVD3					COMMAND_AGE_COUNT			RSVD2					COLUMN_SIZE			RSVD1					CASLAT		

Table 12-30. HW_DRAM_CTL11 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSVD4**	RO	0x0	Reserved.
26:24	**MAX_CS_REG**	RO	0x4	Maximum number of chip selects available. Defines the maximum number of chip selects for the memory controller as the log2 of the number of chip selects.

Table 12-30. HW_DRAM_CTL11 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23:19	**RSVD3**	RO	0x0	Reserved.
18:16	**COMMAND_AGE_COUNT**	RW	0x0	Initial value of individual command aging counters for command aging. Holds the initial value of the command aging counters associated with each command in the command queue. When using the placement logic to fill the command queue, the command aging counters decrement one each time the master aging-rate counter counts down age_count cycles.
15:11	**RSVD2**	RO	0x0	Reserved.
10:8	**COLUMN_SIZE**	RW	0x0	Difference between number of column pins available and number being used. Shows the difference between the maximum column width available (12) and the actual number of column pins being used. The user address is automatically shifted so that the user address space is mapped contiguously into the memory map based on the value of this bit field.
7:3	**RSVD1**	RO	0x0	Reserved.
2:0	**CASLAT**	RW	0x0	Encoded CAS latency sent to DRAMs during initialization. Sets the CAS (Column Address Strobe) latency encoding that the memory uses. The binary value programmed into this bit field is dependent on the memory device, since the same caslat value may have different meanings to different memories. This will be programmed into the DRAM devices at initialization. The CAS encoding will be specified in the DRAM spec sheet, and should correspond to the CASLAT_LIN bit field.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.15 DRAM Control Register 12 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL12 0x030

Table 12-31. HW_DRAM_CTL12

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3					TWR_INT			RSVD2					TRRD			OBSOLETE								RSVD1					TCKE		

Table 12-32. HW_DRAM_CTL12 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSVD3**	RO	0x0	Reserved.
26:24	**TWR_INT**	RW	0x0	DRAM TWR parameter in cycles. Defines the DRAM write recovery time, in cycles.
23:19	**RSVD2**	RO	0x0	Reserved.
18:16	**TRRD**	RW	0x0	DRAM TRRD parameter in cycles. Defines the DRAM activate to activate delay for different banks, in cycles.
15:8	**OBSOLETE**	RO	0x0	Reserved.
7:3	**RSVD1**	RO	0x0	Reserved.
2:0	**TCKE**	RW	0x0	Minimum CKE pulse width. Defines the minimum CKE pulse width, in cycles.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.16 DRAM Control Register 13 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL13 0x034

Table 12-33. HW_DRAM_CTL13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4				CASLAT_LIN_GATE				RSVD3				CASLAT_LIN				RSVD2				APREBIT				RSVD1				TWTR			

Table 12-34. HW_DRAM_CTL13 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSVD4**	RO	0x0	Reserved.
27:24	**CASLAT_LIN_GATE**	RW	0x0	Adjusts data capture gate open by half cycles. Adjusts the data capture gate open time by 1/2 cycle increments. This bit field is programmed differently than CASLAT_LIN when there are fixed offsets in the flight path between the memories and the memory controller for clock gating. When CASLAT_LIN_GATE is a larger value than CASLAT_LIN, the data capture window will become shorter. A CASLAT_LIN_GATE value smaller than CASLAT_LIN may have no effect on the data capture window, depending on the fixed offsets in the ASIC and the board. 0000 - 0010 = Reserved 0011 = 1.5 cycles 0100 = 2 cycles 0101 = 2.5 cycles 0110 = 3 cycles 0111 = 3.5 cycles 1000 = 4 cycles 1001 = Reserved 1010 = 5 cycles All other settings are Reserved
23:20	**RSVD3**	RO	0x0	Reserved.
19:16	**CASLAT_LIN**	RW	0x0	Sets latency from read command send to data receive from/to controller. Sets the CAS latency linear value in 1/2 cycle increments. This sets an internal adjustment for the delay from when the read command is sent from the memory controller to when data will be received back. The window of time in which the data is captured is a fixed length. The CASLAT_LIN bit field adjusts the start of this data capture window. Note: Not all linear values will be supported for the memory devices being used. Refer to the specification for the memory devices being used. 0000 - 0010 = Reserved 0011 = 1.5 cycles 0100 = 2 cycles 0101 = 2.5 cycles 0110 = 3 cycles 0111 = 3.5 cycles 1000 = 4 cycles 1001 = Reserved 1010 = 5 cycles All other settings are reserved
15:12	**RSVD2**	RO	0x0	Reserved.
11:8	**APREBIT**	RW	0x0	Location of the auto pre-charge bit in the DRAM address. Defines the location of the auto pre-charge bit in the DRAM address in decimal encoding.
7:3	**RSVD1**	RO	0x0	Reserved.
2:0	**TWTR**	RW	0x0	DRAM TWTR parameter in cycles. Sets the number of cycles needed to switch from a write to a read operation, as dictated by the DDR SDRAM specification.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.17 DRAM Control Register 14 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL14 0x038

Table 12-35. HW_DRAM_CTL14

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4				MAX_COL_REG				RSVD3				LOWPOWER_REFRESH_ENABLE				RSVD2				INITAREF				RSVD1				CS_MAP			

Table 12-36. HW_DRAM_CTL14 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSVD4**	RO	0x0	Reserved.
27:24	**MAX_COL_REG**	RO	0xd	Maximum width of column address in DRAMs. Defines the maximum width of column address in the DRAM devices. This value can be used to set the COLUMN_SIZE bit field. column_size = max_col_reg - <number of column bits in memory device>.
23:20	**RSVD3**	RO	0x0	Reserved.
19:16	**LOWPOWER_REFRESH_ENABLE**	RW	0x0	Enable refreshes during power down. Enables refreshes during power-down mode. 0 = Disabled 1 = Enabled
15:12	**RSVD2**	RO	0x0	Reserved.
11:8	**INITAREF**	RW	0x0	Number of auto-refresh commands to execute during DRAM initialization. Defines the number of auto-refresh commands needed by the DRAM devices to satisfy the initialization sequence.

Table 12-36. HW_DRAM_CTL14 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:4	**RSVD1**	RO	0x0	Reserved.
3:0	**CS_MAP**	RW	0x0	Sets the mask that determines which chip select pins are active. The user address chip select field will be mapped into the active chip selects indicated by this bit field in ascending order from lowest to highest. This allows the memory controller to map the entire contiguous user address into any group of chip selects. Bit 0 of this bit field corresponds to chip select [0]. Note that the number of chip selects, the number of bits set to 1 in this bit field, must be a power of 2 (2 raised to power of 0, 2 raised to power of 1, 2 raised to power of 2, etc.). NOTE: On the 169-pin BGA package, bits [3:2] of CS_MAP should always be 0. On the 128-pin LQFP, bits [3:1] of CS_MAP should always be 0.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.18 DRAM Control Register 15 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL15 0x03C

Table 12-37. HW_DRAM_CTL15

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4				TRP				RSVD3				TDAL				RSVD2				PORT_BUSY				RSVD1				MAX_ROW_REG			

Table 12-38. HW_DRAM_CTL15 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSVD4**	RO	0x0	Reserved.
27:24	**TRP**	RW	0x0	DRAM TRP parameter in cycles. Defines the DRAM pre-charge command time, in cycles.
23:20	**RSVD3**	RO	0x0	Reserved.

Table 12-38. HW_DRAM_CTL15 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:16	**TDAL**	RW	0x0	DRAM TDAL parameter in cycles. Defines the auto pre-charge write recovery time when auto pre-charge is enabled (ap is set), in cycles. This is defined internally as tRP (pre-charge time) + auto pre-charge write recovery time. Note that not all memories use this parameter. If tDAL is defined in the memory specification, then program this bit field to the specified value. If the memory does not specify a tDAL time, then program this bit field to tWR + tRP. DO NOT program this bit field with a value of 0x0 or the memory controller will not function properly when auto pre-charge is enabled.
15:12	**RSVD2**	RO	0x0	Reserved.
11:8	**PORT_BUSY**	RO	0x0	Per-port indicator that the controller is processing a command. Indicates that a port is actively processing a command. Each bit controls the corresponding port. 0 = Port is not busy. 1 = Port is busy.
7:4	**RSVD1**	RO	0x0	Reserved.
3:0	**MAX_ROW_REG**	RO	0xd	Maximum width of memory address bus. Defines the maximum width of the memory address bus (number of row bits) for the memory controller. This value can be used to set the ADDR_PINS bit field. ADDR_PINS = MAX_ROW_REG - <number of row bits in memory device>.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.19 DRAM Control Register 16 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL16 0x040

Table 12-39. HW_DRAM_CTL16

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4			TMRD					RSVD3			LOWPOWER_CONTROL					RSVD2			LOWPOWER_AUTO_ENABLE				RSVD1				INT_ACK			

Table 12-40. HW_DRAM_CTL16 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD4**	RO	0x0	Reserved.
28:24	**TMRD**	RW	0x00	DRAM TMRD parameter in cycles. Defines the DRAM mode register set command time, in cycles.
23:21	**RSVD3**	RO	0x0	Reserved.
20:16	**LOWPOWER_CONTROL**	RW	0x00	Controls entry into the low-power modes. Enables the individual low-power modes of the device. Bit 0 = Controls memory self-refresh with memory and controller clock gating mode (Mode 5). Reserved and should always be written to a 0. Gate the clock via the CLKCTRL clock-gate for the EMI instead. Bit 1 = Controls memory self-refresh with memory clock gating mode (Mode 4). Bit 2 = Controls memory self-refresh mode (Mode 3). Bit 3 = Controls memory power-down with memory clock gating mode (Mode 2). Bit 4 = Controls memory power-down mode (Mode 1). For all bits: 0 = Disabled. 1 = Enabled.
15:13	**RSVD2**	RO	0x0	Reserved.

Table 12-40. HW_DRAM_CTL16 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
12:8	**LOWPOWER_AUTO_ENABLE**	RW	0x00	Enables automatic entry into the low-power mode on idle. Enables automatic entry into the low-power modes of the memory controller. Bit 0 = Controls memory self-refresh with memory and controller clock gating mode (Mode 5). Reserved and should always be written to a 0. Gate the clock via the CLKCTRL clock-gate for the EMI instead. Bit 1 = Controls memory self-refresh with memory clock gating mode (Mode 4). Reserved and should always be written to a 0. Bit 2 = Controls memory self-refresh mode (Mode 3). Reserved and should always be written to a 0. Bit 3 = Controls memory power-down with memory clock gating mode (Mode 2). Bit 4 = Controls memory power-down mode (Mode 1). For all bits: 0 = Automatic entry into this mode is disabled. The user may enter this mode manually by setting the associated lowpower_control bit. 1 = Automatic entry into this mode is enabled. The mode will be entered automatically when the proper counters expire, and only if the associated lowpower_control bit is set.
7:4	**RSVD1**	RO	0x0	Reserved.
3:0	**INT_ACK**	WO	0x0	Clear mask of the INT_STATUS bit field. Controls the clearing of the INT_STATUS bit field. If any of the INT_ACK bits are set to a 1 the corresponding bit in the INT_STATUS bit field will be cleared to 0. Any INT_ACK bits written with a 0 will not alter the corresponding bit in the INT_STATUS bit field. This bit field will always read back as 0.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.20 DRAM Control Register 17 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL17 0x044

Table 12-41. HW_DRAM_CTL17

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DLL_START_POINT								DLL_LOCK								DLL_INCREMENT								RSVD1				TRC			

Table 12-42. HW_DRAM_CTL17 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**DLL_START_POINT**	RW	0x00	Initial delay count when searching for lock in master DLL. Sets the number of delay elements to place in the master delay line to start searching for lock in master DLL.
23:16	**DLL_LOCK**	RO	0x00	Number of delay elements in master DLL lock. Defines the actual number of delay elements used to capture one full clock cycle. This bit field is automatically updated every time a refresh operation is performed.
15:8	**DLL_INCREMENT**	RW	0x00	Number of elements to add to DLL_START_POINT when searching for lock. Defines the number of delay elements to recursively increment the DLL_START_POINT bit field with when searching for lock.
7:5	**RSVD1**	RO	0x0	Reserved.
4:0	**TRC**	RW	0x00	DRAM TRC parameter in cycles. Defines the DRAM period between active commands for the same bank, in cycles.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.21 DRAM Control Register 18 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL18 0x048

Table 12-43. HW_DRAM_CTL18

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4	DLL_DQS_DELAY_1							RSVD3	DLL_DQS_DELAY_0							RSVD2			INT_STATUS					RSVD1			INT_MASK				

Table 12-44. HW_DRAM_CTL18 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSVD4**	RO	0x0	Reserved.
30:24	**DLL_DQS_DELAY_1**	RW	0x00	Fraction of a cycle to delay the dqs signal from the DRAMs for dll_rd_dqs_slice 1 during reads. Sets the delay for the read_dqs signal from the DDR SDRAM devices for dll_rd_dqs_slice 1. This delay is used center the edges of the read_dqs signal so that the read data will be captured in the middle of the valid window in the I/O logic.Each increment of this bit field adds a delay of 1/128 of the system clock.
23	**RSVD3**	RO	0x0	Reserved.
22:16	**DLL_DQS_DELAY_0**	RW	0x00	Fraction of a cycle to delay the dqs signal from the DRAMs for dll_rd_dqs_slice 0 during reads. Sets the delay for the read_dqs signal from the DDR SDRAM devices for dll_rd_dqs_slice 0. This delay is used center the edges of the read_dqs signal so that the read data will be captured in the middle of the valid window in the I/O logic.Each increment of this bit field adds a delay of 1/128 of the system clock.
15:13	**RSVD2**	RO	0x0	Reserved.
12:8	**INT_STATUS**	RO	0x00	Status of interrupt features in the controller. Shows the status of all possible interrupts generated by the memory controller. The MSB is the result of a logical OR of all the lower bits. The INT_STATUS bits correspond to these interrupts: Bit 0 = A single access outside the defined PHYSICAL memory space detected. Bit 1 = Multiple accesses outside the defined PHYSICAL memory space detected. Bit 2 = DRAM initialization complete. Bit 3 = DLL unlock condition detected. Bit 4 = Logical OR of all lower bits.
7:5	**RSVD1**	RO	0x0	Reserved.
4:0	**INT_MASK**	RW	0x00	Mask for controller_int signals from the INT_STATUS bit field. Active-high mask bits that control the value of the memory controller_int signal on the ASIC interface. This mask is inverted and then logically AND'ed with the outputs of the INT_STATUS bit field.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.22 DRAM Control Register 19 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL19 0x04C

Table 12-45. HW_DRAM_CTL19

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DQS_OUT_SHIFT_BYPASS								RSVD1	DQS_OUT_SHIFT							DLL_DQS_DELAY_BYPASS_1								DLL_DQS_DELAY_BYPASS_0							

Table 12-46. HW_DRAM_CTL19 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**DQS_OUT_SHIFT_BYPASS**	RW	0x00	Sets the delay for the clk_dqs_out signal of the dll_wr_dqs_slice when the DLL is being bypassed. This is used to ensure correct data capture in the I/O logic. The value programmed into this bit field sets the actual number of delay elements in the clk_dqs_out line. If the total delay time programmed exceeds the number of delay elements in the delay chain, then the delay will be set internally to the maximum number of delay elements available.
23	**RSVD1**	RO	0x0	Reserved.
22:16	**DQS_OUT_SHIFT**	RW	0x00	Sets the delay for the clk_dqs_out signal of the dll_wr_dqs_slice to ensure correct data capture in the I/O logic. Each increment of this bit field adds a delay of 1/128 of the system clock.

Table 12-46. HW_DRAM_CTL19 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:8	**DLL_DQS_DELAY_BYPASS_1**	RW	0x00	Sets the delay for the read_dqs signal from the DDR SDRAM devices for dll_rd_dqs_slice 1 for reads when the DLL is being bypassed. This delay is used center the edges of the read_dqs signal so that the read data will be captured in the middle of the valid window in the I/O logic. The value programmed into this bit field sets the actual number of delay elements in the read_dqs line. If the total delay time programmed exceeds the number of delay elements in the delay chain, then the delay will be set internally to the maximum number of delay elements available.
7:0	**DLL_DQS_DELAY_BYPASS_0**	RW	0x00	Sets the delay for the read_dqs signal from the DDR SDRAM devices for dll_rd_dqs_slice 0 for reads when the DLL is being bypassed. This delay is used center the edges of the read_dqs signal so that the read data will be captured in the middle of the valid window in the I/O logic. The value programmed into this bit field sets the actual number of delay elements in the read_dqs line. If the total delay time programmed exceeds the number of delay elements in the delay chain, then the delay will be set internally to the maximum number of delay elements available.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.23 DRAM Control Register 20 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL20 0x050

Table 12-47. HW_DRAM_CTL20

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TRCD_INT								TRAS_MIN								WR_DQS_SHIFT_BYPASS								RSVD1	WR_DQS_SHIFT						

Table 12-48. HW_DRAM_CTL20 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**TRCD_INT**	RW	0x00	DRAM TRCD parameter in cycles. Defines the DRAM RAS to CAS delay, in cycles
23:16	**TRAS_MIN**	RW	0x00	DRAM TRAS_MIN parameter in cycles. Defines the DRAM minimum row activate time, in cycles.
15:8	**WR_DQS_SHIFT_BYPASS**	RW	0x00	Sets the delay for the clk_wr signal when the DLL is being bypassed. This is used to ensure correct data capture in the I/O logic. The value programmed into this bit field sets the actual number of delay elements in the clk_wr line. If the total delay time programmed exceeds the number of delay elements in the delay chain, then the delay will be set internally to the maximum number of delay elements available.
7	**RSVD1**	RO	0x0	Reserved.
6:0	**WR_DQS_SHIFT**	RW	0x00	Sets the delay for the clk_wr signal to ensure correct data capture in the I/O logic. Each increment of this bit field adds a delay of 1/128 of the system clock. The same delay will be added to the clk_dqs_out signal for each slice.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.24 DRAM Control Register 21 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL21 0x054

Table 12-49. HW_DRAM_CTL21

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
OBSOLETE								RSVD1						OUT_OF_RANGE_LENGTH										TRFC							

Table 12-50. HW_DRAM_CTL21 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**OBSOLETE**	RO	0x0	Reserved.
23:18	**RSVD1**	RO	0x0	Reserved.
17:8	**OUT_OF_RANGE_LENGTH**	RO	0x000	Length of command that caused an Out-of-Range interrupt. Holds the length of the command that caused an out-of-range interrupt request to the memory devices.
7:0	**TRFC**	RW	0x00	DRAM TRFC parameter in cycles. Defines the DRAM refresh command time, in cycles.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.25 DRAM Control Register 22 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL22 0x058

Table 12-51. HW_DRAM_CTL22

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2					AHB0_WRCNT											RSVD1					AHB0_RDCNT										

Table 12-52. HW_DRAM_CTL22 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSVD2**	RO	0x0	Reserved.
26:16	**AHB0_WRCNT**	RW	0x000	Number of bytes for an INCR WRITE command on port 0. Holds the number of bytes to send to the memory controller core from AHB port 0 for an INCR WRITE AHB command. The AHB logic will subdivide an INCR request into memory controller core commands of the size of this bit field. The logic will continue sending bursts of this size as the previous request has been transmitted by the AHB port. If the INCR command is terminated on an unnatural boundary, the logic will discard the unnecessary words. The value defined in this bit field should be a multiple of the number of bytes in the AHB port width. Clearing this bit field will cause the port to issue commands of 0 length to the controller core, which the core interprets as the pre-configured value of 1024 bytes.
15:11	**RSVD1**	RO	0x0	Reserved.
10:0	**AHB0_RDCNT**	RW	0x000	Number of bytes for an INCR READ command on port 0. Holds the number of bytes to return to AHB port 0 for an INCR READ AHB command. The AHB logic will subdivide an INCR request into memory controller core commands of the size of this bit field. The logic will continue requesting bursts of this size as soon as the previous request has been received by the AHB port. If the INCR command is terminated on an unnatural boundary, the logic will discard the unnecessary words. The value defined in this bit field should be a multiple of the number of bytes in the AHB port width. Clearing this bit field will cause the port to issue commands of 0 length to the controller core, which the core interprets as the pre-configured value of 1024 bytes.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.26 DRAM Control Register 23 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL23 0x05C

Table 12-53. HW_DRAM_CTL23

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2					AHB1_WRCNT											RSVD1					AHB1_RDCNT										

Table 12-54. HW_DRAM_CTL23 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSVD2**	RO	0x0	Reserved.
26:16	**AHB1_WRCNT**	RW	0x000	Number of bytes for an INCR WRITE command on port 1. Holds the number of bytes to send to the memory controller core from AHB port 0 for an INCR WRITE AHB command. The AHB logic will subdivide an INCR request into memory controller core commands of the size of this bit field. The logic will continue sending bursts of this size as the previous request has been transmitted by the AHB port. If the INCR command is terminated on an unnatural boundary, the logic will discard the unnecessary words. The value defined in this bit field should be a multiple of the number of bytes in the AHB port width. Clearing this bit field will cause the port to issue commands of 0 length to the controller core, which the core interprets as the pre-configured value of 1024 bytes.
15:11	**RSVD1**	RO	0x0	Reserved.
10:0	**AHB1_RDCNT**	RW	0x000	Number of bytes for an INCR READ command on port 1. Holds the number of bytes to return to AHB port 0 for an INCR READ AHB command. The AHB logic will subdivide an INCR request into memory controller core commands of the size of this bit field. The logic will continue requesting bursts of this size as soon as the previous request has been received by the AHB port. If the INCR command is terminated on an unnatural boundary, the logic will discard the unnecessary words. The value defined in this bit field should be a multiple of the number of bytes in the AHB port width. Clearing this bit field will cause the port to issue commands of 0 length to the controller core, which the core interprets as the pre-configured value of 1024 bytes.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.27 DRAM Control Register 24 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL24 0x060

Table 12-55. HW_DRAM_CTL24

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2					AHB2_WRCNT											RSVD1					AHB2_RDCNT										

Table 12-56. HW_DRAM_CTL24 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSVD2**	RO	0x0	Reserved.
26:16	**AHB2_WRCNT**	RW	0x000	Number of bytes for an INCR WRITE command on port 2. Holds the number of bytes to send to the memory controller core from AHB port 2 for an INCR WRITE AHB command. The AHB logic will subdivide an INCR request into memory controller core commands of the size of this bit field. The logic will continue sending bursts of this size as the previous request has been transmitted by the AHB port. If the INCR command is terminated on an unnatural boundary, the logic will discard the unnecessary words. The value defined in this bit field should be a multiple of the number of bytes in the AHB port width. Clearing this bit field will cause the port to issue commands of 0 length to the controller core, which the core interprets as the pre-configured value of 1024 bytes.
15:11	**RSVD1**	RO	0x0	Reserved.
10:0	**AHB2_RDCNT**	RW	0x000	Number of bytes for an INCR READ command on port 2. Holds the number of bytes to return to AHB port 2 for an INCR READ AHB command. The AHB logic will subdivide an INCR request into memory controller core commands of the size of this bit field. The logic will continue requesting bursts of this size as soon as the previous request has been received by the AHB port. If the INCR command is terminated on an unnatural boundary, the logic will discard the unnecessary words. The value defined in this bit field should be a multiple of the number of bytes in the AHB port width. Clearing this bit field will cause the port to issue commands of 0 length to the controller core, which the core interprets as the pre-configured value of 1024 bytes.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.28 DRAM Control Register 25 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL25 0x064

Table 12-57. HW_DRAM_CTL25

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2					AHB3_WRCNT											RSVD1					AHB3_RDCNT										

Table 12-58. HW_DRAM_CTL25 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSVD2**	RO	0x0	Reserved.
26:16	**AHB3_WRCNT**	RW	0x000	Number of bytes for an INCR WRITE command on port 3. Holds the number of bytes to send to the memory controller core from AHB port 3 for an INCR WRITE AHB command. The AHB logic will subdivide an INCR request into memory controller core commands of the size of this bit field. The logic will continue sending bursts of this size as the previous request has been transmitted by the AHB port. If the INCR command is terminated on an unnatural boundary, the logic will discard the unnecessary words. The value defined in this bit field should be a multiple of the number of bytes in the AHB port width. Clearing this bit field will cause the port to issue commands of 0 length to the controller core, which the core interprets as the pre-configured value of 1024 bytes.
15:11	**RSVD1**	RO	0x0	Reserved.
10:0	**AHB3_RDCNT**	RW	0x000	Number of bytes for an INCR READ command on port 3. Holds the number of bytes to return to AHB port 3 for an INCR READ AHB command. The AHB logic will subdivide an INCR request into memory controller core commands of the size of this bit field. The logic will continue requesting bursts of this size as soon as the previous request has been received by the AHB port. If the INCR command is terminated on an unnatural boundary, the logic will discard the unnecessary words. The value defined in this bit field should be a multiple of the number of bytes in the AHB port width. Clearing this bit field will cause the port to issue commands of 0 length to the controller core, which the core interprets as the pre-configured value of 1024 bytes.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.29 DRAM Control Register 26 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL26 0x068

Table 12-59. HW_DRAM_CTL26

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
OBSOLETE																RSVD1				TREF											

Table 12-60. HW_DRAM_CTL26 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**OBSOLETE**	RO	0x0	Reserved.
15:12	**RSVD1**	RO	0x0	Reserved.
11:0	**TREF**	RW	0x000	DRAM TREF parameter in cycles. Defines the DRAM cycles between refresh commands.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.30 DRAM Control Register 27 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL27 0x06C

Table 12-61. HW_DRAM_CTL27

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
OBSOLETE																															

Table 12-62. HW_DRAM_CTL27 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**OBSOLETE**	RO	0x0	Reserved.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.31 DRAM Control Register 28 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL28 0x070

Table 12-63. HW_DRAM_CTL28

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
OBSOLETE																															

Table 12-64. HW_DRAM_CTL28 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**OBSOLETE**	RO	0x0	Reserved.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.32 DRAM Control Register 29 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL29 0x074

Table 12-65. HW_DRAM_CTL29

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
LOWPOWER_INTERNAL_CNT																LOWPOWER_EXTERNAL_CNT															

Table 12-66. HW_DRAM_CTL29 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**LOWPOWER_INTERNAL_CNT**	RW	0x0000	Counts idle cycles to self-refresh with memory and controller clk gating. Counts the number of idle cycles before memory self-refresh with memory and controller clock gating low-power mode 5.
15:0	**LOWPOWER_EXTERNAL_CNT**	RW	0x0000	Counts idle cycles to self-refresh with memory clock gating. Counts the number of idle cycles before memory self-refresh with memory clock gating low-power mode 4.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.33 DRAM Control Register 30 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL30 0x078

Table 12-67. HW_DRAM_CTL30

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
LOWPOWER_REFRESH_HOLD																LOWPOWER_POWER_DOWN_CNT															

Table 12-68. HW_DRAM_CTL30 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**LOWPOWER_REFRESH_HOLD**	RW	0x0000	Re-Sync counter for DLL in Clock Gate Mode. Counts the re-synchronization cycles for the DLL in Clock Gate Mode.
15:0	**LOWPOWER_POWER_DOWN_CNT**	RW	0x0000	Counts idle cycles to memory power-down. Counts the number of idle cycles before memory power-down or power-down with memory clock gating low-power mode 1 or 2.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.34 DRAM Control Register 31 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL31 0x07C

Table 12-69. HW_DRAM_CTL31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TDLL																LOWPOWER_SELF_REFRESH_CNT															

Table 12-70. HW_DRAM_CTL31 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**TDLL**	RW	0x0000	DRAM TDLL parameter in cycles. Defines the DRAM DLL lock time, in cycles.
15:0	**LOWPOWER_SELF_REFRESH_CNT**	RW	0x0000	Counts idle cycles to memory self-refresh. Counts the number of cycles to the next memory self-refresh low-power mode 3.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.35 DRAM Control Register 32 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL32 0x080

Table 12-71. HW_DRAM_CTL32

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TXSNR																TRAS_MAX															

Table 12-72. HW_DRAM_CTL32 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**TXSNR**	RW	0x0000	DRAM TXSNR parameter in cycles. Defines the DRAM tXSNR parameter, in cycles.
15:0	**TRAS_MAX**	RW	0x0000	DRAM TRAS_MAX parameter in cycles. Defines the DRAM maximum row active time, in cycles.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.36 DRAM Control Register 33 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL33 0x084

Table 12-73. HW_DRAM_CTL33

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VERSION																TXSR															

Table 12-74. HW_DRAM_CTL33 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**VERSION**	RO	0x2041	Controller version number. Holds the version number for this controller.
15:0	**TXSR**	RW	0x0000	DRAM TXSR parameter in cycles. Defines the DRAM self-refresh exit time, in cycles.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.37 DRAM Control Register 34 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL34 0x088

Table 12-75. HW_DRAM_CTL34

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1								TINIT																							

Table 12-76. HW_DRAM_CTL34 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD1**	RO	0x0	Reserved.
23:0	**TINIT**	RW	0x000000	DRAM TINIT parameter in cycles. Defines the DRAM initialization time, in cycles.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.38 DRAM Control Register 35 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL35 0x08C

Table 12-77. HW_DRAM_CTL35

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1																OUT_OF_RANGE_ADDR															

Table 12-78. HW_DRAM_CTL35 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSVD1**	RO	0x0	Reserved.
30:0	**OUT_OF_RANGE_ADDR**	RO	0x00000000	Address of command that caused an Out-of-Range interrupt. Holds the address of the command that caused an out-of-range interrupt request to the memory devices.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.39 DRAM Control Register 36 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL36 0x090

Table 12-79. HW_DRAM_CTL36

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD5							PWRUP_SREFRESH_EXIT	RSVD4							ENABLE_QUICK_SREFRESH	RSVD3							RSVD2	RSVD1							ACTIVE_AGING

Table 12-80. HW_DRAM_CTL36 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD5**	RO	0x0	Reserved.
24	**PWRUP_SREFRESH_EXIT**	RW	0x0	Powerup via self-refresh instead of full memory initialization. Allows controller to exit power-down mode by executing a self-refresh instead of the full memory initialization. 0 = Disabled 1 = Enabled
23:17	**RSVD4**	RO	0x0	Reserved.
16	**ENABLE_QUICK_SREFRESH**	RW	0x0	Interrupts memory initialization to enter self-refresh mode. When this bit is set, the memory initialization sequence will be interrupted and self-refresh mode will be entered. 0 = Continue memory initialization. 1 = Interrupt memory initialization and enter self-refresh mode.
15:9	**RSVD3**	RO	0x0	Reserved.
8	**RSVD2**	RW	0x0	Program this field to 0x0.

Table 12-80. HW_DRAM_CTL36 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:1	**RSVD1**	RO	0x0	Reserved.
0	**ACTIVE_AGING**	RW	0x0	Enable aging of the active command. Enables aging of the active command as a condition when using the placement logic to fill the command queue. 0 = Disabled 1 = Enabled

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.40 DRAM Control Register 37 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL37 0x094

Table 12-81. HW_DRAM_CTL37

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
OBSOLETE								RSVD2								BUS_SHARE_TIMEOUT								RSVD1							TREF_ENABLE

Table 12-82. HW_DRAM_CTL37 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**OBSOLETE**	RO	0x0	Reserved.
23:18	**RSVD2**	RO	0x0	Reserved.
17:8	**BUS_SHARE_TIMEOUT**	RW	0x000	Wait time from when the memory controller needs the bus to asserting bus_timeout signal. Sets the wait time for the memory controller when the bus is being shared. This value is loaded into the bus share counter when a command is ready to communicate with the memory devices. When the counter expires, the bus_timeout signal will be asserted indicating to the external source that the memory controller is requesting the shared pins.

Table 12-82. HW_DRAM_CTL37 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:1	**RSVD1**	RO	0x0	Reserved.
0	**TREF_ENABLE**	RW	0x0	Issue self-refresh commands to the DRAMs every TREF cycles. Enables internal refresh commands. If command refresh mode is configured, then refresh commands will be issued based on the internal tref counter and any refresh commands sent through the command interface. 0 = Internal refresh commands disabled. 1 = Internal refresh commands enabled.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.41 DRAM Control Register 38 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL38 0x098

Table 12-83. HW_DRAM_CTL38

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2			EMRS2_DATA_0													RSVD1			EMRS1_DATA												

Table 12-84. HW_DRAM_CTL38 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD2**	RO	0x0	Reserved.
28:16	**EMRS2_DATA_0**	RW	0x0000	EMRS2 data for chip select 0. Holds the EMRS2 data written during DDRII initialization for chip select 0. The contents of this bit field will be programmed into the DRAM at initialization or when the write_modereg bit field is written with a 1. Consult the DRAM specification for the correct settings for this bit field.

Table 12-84. HW_DRAM_CTL38 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:13	**RSVD1**	RO	0x0	Reserved.
12:0	**EMRS1_DATA**	RW	0x0000	EMRS1 data. Holds the EMRS1 data written during DDRII initialization. The contents of this bit field will be programmed into the DRAM at initialization or when the WRITE_MODEREG bit field is written with a 1. Consult the DRAM specification for the correct settings for this bit field.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.42 DRAM Control Register 39 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL39 0x09C

Table 12-85. HW_DRAM_CTL39

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2			EMRS2_DATA_2													RSVD1			EMRS2_DATA_1												

Table 12-86. HW_DRAM_CTL39 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD2**	RO	0x0	Reserved.
28:16	**EMRS2_DATA_2**	RW	0x0000	EMRS2 data for chip select 2. Holds the EMRS2 data written during DDRII initialization for chip select 2. The contents of this bit field will be programmed into the DRAM at initialization or when the WRITE_MODEREG bit field is written with a 1. Consult the DRAM specification for the correct settings for this bit field.
15:13	**RSVD1**	RO	0x0	Reserved.
12:0	**EMRS2_DATA_1**	RW	0x0000	EMRS2 data for chip select 1. Holds the EMRS2 data written during DDRII initialization for chip select 1. The contents of this bit field will be programmed into the DRAM at initialization or when the WRITE_MODEREG bit field is written with a 1. Consult the DRAM specification for the correct settings for this bit field.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.43 DRAM Control Register 40 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL40 0x0A0

Table 12-87. HW_DRAM_CTL40

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TPDEX																RSVD1			EMRS2_DATA_3												

Table 12-88. HW_DRAM_CTL40 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**TPDEX**	RW	0x0000	DRAM TPDEX parameter in cycles. Defines the DRAM power-down exit command period, in cycles.
15:13	**RSVD1**	RO	0x0	Reserved.
12:0	**EMRS2_DATA_3**	RW	0x0000	EMRS2 data for chip select 3. Holds the EMRS2 data written during DDRII initialization for chip select 3. The contents of this bit field will be programmed into the DRAM at initialization or when the WRITE_MODEREG bit field is written with a 1. Consult the DRAM specification for the correct settings for this bit field.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

DRAM Block v2.1, Revision 1.50

12.6 EMI Memory Parameters and Register Settings

12.6.1 Mobile DDR (5 nsec) Parameters

Table 12-89. Frequency Dependent Parameters

Parameter	24 MHz	48 MHz	60 MHz	96 MHz	120 MHz	133 MHz	160 MHz
TCKE	0x0001	0x0001	0x0001	0x0001	0x0001	0x0001	0x0001
TDAL	0x03	0x03	0x03	0x04	0x04	0x04	0x05
TDLL	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000
TEMRS	0x02	0x02	0x02	0x02	0x02	0x02	0x02
TINIT	0x000012C1	0x00002582	0x00002EE5	0x00004B0D	0x00005DCA	0x00006665	0x00007D00
TMRD	0x02	0x02	0x02	0x02	0x02	0x02	0x02
TPDEX	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002
TRAS_MAX	0x0687	0x0D17	0x1060	0x1A3B	0x20CA	0x23CD	0x2BB6
TRAS_MIN	0x01	0x01	0x03	0x04	0x05	0x06	0x07
TRC	0x02	0x03	0x04	0x06	0x07	0x08	0x09
TRCD_INT	0x01	0x01	0x01	0x02	0x02	0x02	0x03
TREF	0x000000B3	0x0000016F	0x000001CC	0x000002E6	0x000003A1	0x000003F7	0x000004DA
TRFC	0x0003	0x0005	0x0006	0x000A	0x000C	0x000D	0x0010
TRP	0x01	0x01	0x01	0x02	0x02	0x02	0x03
TRRD	0x01	0x01	0x01	0x01	0x02	0x02	0x02
TWR_INT	0x02	0x02	0x02	0x02	0x02	0x02	0x02
TWTR	0x03	0x03	0x03	0x03	0x03	0x03	0x03
TXSNR	0x0003	0x0006	0x0008	0x000C	0x000F	0x0010	0x0014
TXSR	0x0004	0x0007	0x0009	0x000D	0x0011	0x0012	0x0016

12.6.1.1 Bypass Cutoff

Suggested bypass cutoff frequency is 60 MHz.

12.6.1.2 Bypass Mode Enabled

Table 12-90. Delays

Parameter	24 MHz	48 MHz	60 MHz
DLL_DQS_DELAY_BYPASS_0	0x24	0x13	0x0E
DLL_DQS_DELAY_BYPASS_1	0x24	0x13	0x0E
DQS_OUT_SHIFT_BYPASS	0x01	0x01	0x01
WR_DQS_SHIFT_BYPASS	0x23	0x12	0x0D

12.6.1.3 Bypass Mode Disabled

Table 12-91. DLL

Parameter	60 MHz	96 MHz	120 MHz	133 MHz
DLL_INCREMENT	TBD	TBD	TBD	TBD
DLL_START_POINT	TBD	TBD	TBD	TBD

Table 12-92. Delays

Parameter	Value
DLL_DQS_DELAY_1	0x20
DLL_DQS_DELAY_0	0x20
DQS_OUT_SHIFT	0x7F
WR_DQS_SHIFT	0x1C

12.6.1.4 Example Register Settings

```
////////////////////////////////////////////////////////////////////////////
//
// Filename: mobile_ddr_mt46h32m16lf_5_regs_3_160MHz.h
//
// Description: Initialization register values for EMI DRAM controller
//
////////////////////////////////////////////////////////////////////////////
//
// WARNING!  THIS FILE IS AUTOMATICALLY GENERATED.
//           DO NOT MODIFY THIS FILE DIRECTLY.
//
// SETTINGS USED TO GENERATE THIS FILE:
//
//            Burst: 4
//              CAS: 3
//             Freq: 160 MHz
//   Auto-Precharge: 0
//       DLL_Bypass: 0
//
////////////////////////////////////////////////////////////////////////////

{
    0x01010001,  /* 0x01 ahb0_w_priority 0x01 ahb0_r_priority 0x00 ahb0_fifo_type_reg
                    0x01 addr_cmp_en */
    0x00010100,  /* 0x00 ahb2_fifo_type_reg 0x01 ahb1_w_priority 0x01 ahb1_r_priority
                    0x00 ahb1_fifo_type_reg */
    0x01000101,  /* 0x01 ahb3_r_priority 0x00 ahb3_fifo_type_reg 0x01 ahb2_w_priority
                    0x01 ahb2_r_priority */
    0x00000001,  /* 0x00 auto_refresh_mode 0x00 arefresh 0x00 ap 0x01 ahb3_w_priority */
   0x00000101,  /* 0x00 dll_bypass_mode 0x00 dlllockreg 0x01 concurrentap 0x01 bank_split_en */
   0x00000001,  /* 0x00 intrptreada 0x00 intrptapburst 0x00 fast_write 0x01 en_lowpower_mode */
    0x00010000,  /* 0x00 power_down 0x01 placement_en 0x00 no_cmd_init 0x00 intrptwritea */
    0x01000101,  /* 0x01 rw_same_en 0x00 reg_dimm_enable 0x01 rd2rd_turn 0x01 priority_en */
    0x01000000,  /* 0x01 tras_lockout 0x00 start 0x00 srefresh 0x00 sdr_mode */
    0x00000001,  /* 0x00 out_of_range_type 0x00 out_of_range_source_id 0x00 write_modereg
                    0x01 writeinterp */
```

```
    0x07000200,  /* 0x07 age_count 0x00 addr_pins 000000_10 temrs 0x00 q_fullness */
    0x00070203,  /* 0x00 max_cs_reg 0x07 command_age_count 0x02 column_size 0x03 caslat */
    0x02020001,  /* 0x02 twr_int 0x02 trrd 0x0001 tcke */
    0x06060a03,  /* 0x06 caslat_lin_gate 0x06 caslat_lin 0x0a aprebit 0x03 twtr */
    0x00000201,  /* 0x00 max_col_reg 0x00 lowpower_refresh_enable 0x02 initaref 0x01 cs_map */
    0x03050000,  /* 0x03 trp 0x05 tdal 0x00 port_busy 0x00 max_row_reg */
    0x02000000,  /* 0x02 tmrd 0x00 lowpower_control 0x00 lowpower_auto_enable 0x00 int_ack */
    0x2d000d09,  /* 0x2d dll_start_point 0x00 dll_lock 0x0d dll_increment 0x09 trc */
    0x20200000,  /* 0x20 dll_dqs_delay_1 0x20 dll_dqs_delay_0 0x00 int_status 0x00 int_mask */
    0x02020f0f,  /* 0x02 dqs_out_shift_bypass 0x02 dqs_out_shift 0x0f dll_dqs_delay_bypass_1
                    0x0f dll_dqs_delay_bypass_0 */
    0x0307121c,  /* 0x03 trcd_int 0x07 tras_min 0x12 wr_dqs_shift_bypass 0x1c wr_dqs_shift */
    0x00000010,  /* 0x000000 out_of_range_length 0x10 trfc */
    0x00080008,  /* 0x0008 ahb0_wrcnt 0x0008 ahb0_rdcnt */
    0x00200020,  /* 0x0020 ahb1_wrcnt 0x0020 ahb1_rdcnt */
    0x00200020,  /* 0x0020 ahb2_wrcnt 0x0020 ahb2_rdcnt */
    0x00200020,  /* 0x0020 ahb3_wrcnt 0x0020 ahb3_rdcnt */
    0x000004da,  /* 0x000004da tref */
    0x00000000,  /* 0x00000000 */
    0x00000000,  /* 0x00000000 */
    0x00000000,  /* 0x0000 lowpower_internal_cnt 0x0000 lowpower_external_cnt */
    0x00000000,  /* 0x0000 lowpower_refresh_hold 0x0000 lowpower_power_down_cnt */
    0x00000000,  /* 0x0000 tdll 0x0000 lowpower_self_refresh_cnt */
    0x00142bb6,  /* 0x0014 txsnr 0x2bb6 tras_max */
    0x00000016,  /* 0x0000 version 0x0016 txsr */
    0x00007d00,  /* 0x00007d00 tinit */
    0x00000000,  /* 0x00000000 out_of_range_addr */
    0x00000101,  /* 0x00 pwrup_srefresh_exit 0x00 enable_quick_srefresh
                    0x01 bus_share_enable 0x01 active_aging */
    0x00040001,  /* 0x000400 bus_share_timeout 0x01 tref_enable */
    0x00400000,  /* 0x0040 emrs2_data_0 0x0000 emrs1_data */
    0x00400040,  /* 0x0040 emrs2_data_2 0x0040 emrs2_data_1 */
    0x00020040,  /* 0x0002 tpdex 0x0040 emrs2_data_3 */
},
```

12.6.2 Mobile DDR (6 nsec)

Table 12-93. Frequency Dependent Parameters

Parameter	24 MHz	48 MHz	60 MHz	96 MHz	120 MHz	133 MHz	167 MHz
TCKE	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002
TDAL	0x03	0x03	0x04	0x04	0x05	0x05	0x05
TDLL	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000
TEMRS	0x02	0x02	0x02	0x02	0x02	0x02	0x02
TINIT	0x000012C1	0x00002582	0x00002EE5	0x00004B0D	0x00005DCA	0x00006665	0x000081C7
TMRD	0x02	0x02	0x02	0x02	0x02	0x02	0x02
TPDEX	0x0001	0x0002	0x0002	0x0003	0x0004	0x0004	0x0005
TRAS_MAX	0x0687	0x0D17	0x1060	0x1A3B	0x20CA	0x23CD	0x2D62
TRAS_MIN	0x02	0x03	0x03	0x05	0x06	0x06	0x07
TRC	0x02	0x03	0x04	0x06	0x08	0x08	0x0A

Table 12-93. Frequency Dependent Parameters (continued)

Parameter	24 MHz	48 MHz	60 MHz	96 MHz	120 MHz	133 MHz	167 MHz
TRCD_INT	0x01	0x01	0x02	0x02	0x03	0x03	0x03
TREF	0x000000B3	0x0000016F	0x000001CC	0x000002E6	0x000003A1	0x000003F7	0x00000509
TRFC	0x02	0x04	0x05	0x07	0x09	0x0A	0x0C
TRP	0x01	0x01	0x02	0x02	0x03	0x03	0x03
TRRD	0x01	0x01	0x01	0x02	0x02	0x02	0x02
TWR_INT	0x02	0x02	0x02	0x02	0x02	0x02	0x02
TWTR	0x02	0x02	0x02	0x02	0x02	0x02	0x02
TXSNR	0x0003	0x0006	0x0008	0x000C	0x000F	0x0010	0x0014
TXSR	0x0003	0x0006	0x0008	0x000C	0x000F	0x0010	0x0014

12.6.2.1 Bypass Cutoff

Suggested bypass cutoff frequency is 60 MHz.

12.6.2.2 Bypass Mode Enabled

Table 12-94. Delays

Parameter	24 MHz	48 MHz	60 MHz
DLL_DQS_DELAY_BYPASS_0	0x1A	0x0D	0x0C
DLL_DQS_DELAY_BYPASS_1	0x1A	0x0D	0x0C
DQS_OUT_SHIFT_BYPASS	0x01	0x01	0x01
WR_DQS_SHIFT_BYPASS	0x12	0x12	0x12

12.6.2.3 Bypass Mode Disabled

Table 12-95. DLL

Parameter	60 MHz	96 MHz	120 MHz	133 MHz	167 MHz
DLL_INCREMENT	0x06	0x05	0x05	0x05	0x05
DLL_START_POINT	0x28	0x18	0x14	0x14	0x14

Table 12-96. Delays

PARAMETER	VALUE
DLL_DQS_DELAY_1	0x20
DLL_DQS_DELAY_0	0x20
DQS_OUT_SHIFT	0x7F
WR_DQS_SHIFT	0x20

12.6.2.4 Example Register Settings

```
//////////////////////////////////////////////////////////////////////////////
//
// Filename: mobile_ddr_mt46h16m16lf_6_regs_3_96MHz.h
//
// Description: Initialization register values for EMI DRAM controller
//
//////////////////////////////////////////////////////////////////////////////
//
// WARNING!  THIS FILE IS AUTOMATICALLY GENERATED.
//           DO NOT MODIFY THIS FILE DIRECTLY.
//
// SETTINGS USED TO GENERATE THIS FILE:
//
//            Burst: 4
//              CAS: 3
//             Freq: 96 MHz
//   Auto-Precharge: 0
//       DLL_Bypass: 0
//
//////////////////////////////////////////////////////////////////////////////

{
    0x01010001,  /* 0x01 ahb0_w_priority 0x01 ahb0_r_priority 0x00 ahb0_fifo_type_reg
                    0x01 addr_cmp_en */
    0x00010100,  /* 0x00 ahb2_fifo_type_reg 0x01 ahb1_w_priority 0x01 ahb1_r_priority
                    0x00 ahb1_fifo_type_reg */
    0x01000101,  /* 0x01 ahb3_r_priority 0x00 ahb3_fifo_type_reg 0x01 ahb2_w_priority
                    0x01 ahb2_r_priority */
    0x00000001,  /* 0x00 auto_refresh_mode 0x00 arefresh 0x00 ap 0x01 ahb3_w_priority */
   0x00000101,  /* 0x00 dll_bypass_mode 0x00 dlllockreg 0x01 concurrentap 0x01 bank_split_en */
   0x00000001,  /* 0x00 intrptreada 0x00 intrptapburst 0x00 fast_write 0x01 en_lowpower_mode */
    0x00010000,  /* 0x00 power_down 0x01 placement_en 0x00 no_cmd_init 0x00 intrptwritea */
    0x01000001,  /* 0x01 rw_same_en 0x00 reg_dimm_enable 0x00 rd2rd_turn 0x01 priority_en */
    0x01000000,  /* 0x01 tras_lockout 0x00 start 0x00 srefresh 0x00 sdr_mode */
    0x00000001,  /* 0x00 out_of_range_type 0x00 out_of_range_source_id 0x00 write_modereg
                    0x01 writeinterp */
    0x07000200,  /* 0x07 age_count 0x00 addr_pins 000000_10 temrs 0x00 q_fullness */
    0x00070303,  /* 0x00 max_cs_reg 0x07 command_age_count 0x03 column_size 0x03 caslat */
    0x02020002,  /* 0x02 twr_int 0x02 trrd 0x0002 tcke */
    0x06060a02,  /* 0x06 caslat_lin_gate 0x06 caslat_lin 0x0a aprebit 0x02 twtr */
    0x00000201,  /* 0x00 max_col_reg 0x00 lowpower_refresh_enable 0x02 initaref 0x01 cs_map */
    0x02040000,  /* 0x02 trp 0x04 tdal 0x00 port_busy 0x00 max_row_reg */
    0x02000000,  /* 0x02 tmrd 0x00 lowpower_control 0x00 lowpower_auto_enable 0x00 int_ack */
    0x18000606,  /* 01001110 dll_start_point 0x00 dll_lock 0x06 dll_increment 0x06 trc */
    0x15150000,  /* 0x15 dll_dqs_delay_1 0x15 dll_dqs_delay_0 0x00 int_status 0x00 int_mask */
    0x027f1a1a,  /* 0x02 dqs_out_shift_bypass 0x7f dqs_out_shift 0x1a dll_dqs_delay_bypass_1
                    0x1a dll_dqs_delay_bypass_0 */
    0x02051c12,  /* 0x02 trcd_int 0x05 tras_min 0x1c wr_dqs_shift_bypass 0x12 wr_dqs_shift */
    0x00000007,  /* 0x000000 out_of_range_length 0x07 trfc */
    0x00080008,  /* 0x0008 ahb0_wrcnt 0x0008 ahb0_rdcnt */
    0x00200020,  /* 0x0020 ahb1_wrcnt 0x0020 ahb1_rdcnt */
    0x00200020,  /* 0x0020 ahb2_wrcnt 0x0020 ahb2_rdcnt */
    0x00200020,  /* 0x0020 ahb3_wrcnt 0x0020 ahb3_rdcnt */
    0x000002e6,  /* 0x000002e6 tref */
    0x00000000,  /* 0x00000000 */
    0x00000000,  /* 0x00000000 */
```

```
    0x00000000,  /* 0x0000 lowpower_internal_cnt 0x0000 lowpower_external_cnt */
    0x00000000,  /* 0x0000 lowpower_refresh_hold 0x0000 lowpower_power_down_cnt */
    0x00000000,  /* 0x0000 tdll 0x0000 lowpower_self_refresh_cnt */
    0x000c1a3b,  /* 0x000c txsnr 0x1a3b tras_max */
    0x0000000c,  /* 0x0000 version 0x000c txsr */
    0x00004b0d,  /* 0x00004b0d tinit */
    0x00000000,  /* 0x00000000 out_of_range_addr */
    0x00000101,  /* 0x00 pwrup_srefresh_exit 0x00 enable_quick_srefresh
                    0x01 bus_share_enable 0x01 active_aging */
    0x00040001,  /* 0x000400 bus_share_timeout 0x01 tref_enable */
    0x00400000,  /* 0x0040 emrs2_data_0 0x0000 emrs1_data */
    0x00400040,  /* 0x0040 emrs2_data_2 0x0040 emrs2_data_1 */
    0x00030040,  /* 0x0003 tpdex 0x0040 emrs2_data_3 */
},
```

12.6.3 Mobile DDR (7.5 nsec)

Table 12-97. Frequency Dependent Parameters

Parameter	24 MHz	48 MHz	60 MHz	96 MHz	120 MHz	133 MHz
TCKE	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002
TDAL	0x03	0x04	0x04	0x05	0x05	0x05
TDLL	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000
TEMRS	0x02	0x02	0x02	0x02	0x02	0x02
TINIT	0x000012C1	0x00002582	0x00002EE5	0x00004B0D	0x00005DCA	0x00006665
TMRD	0x02	0x02	0x02	0x02	0x02	0x02
TPDEX	0x0001	0x0002	0x0002	0x0003	0x0004	0x0004
TRAS_MAX	0x0687	0x0D17	0x1060	0x1A3B	0x20CA	0x23CD
TRAS_MIN	0x02	0x03	0x03	0x05	0x06	0x06
TRC	0x02	0x04	0x05	0x08	0x0A	0x0A
TRCD_INT	0x01	0x02	0x02	0x03	0x03	0x03
TREF	0x00B3	0x016F	0x01CC	0x02E6	0x03A1	0x03F7
TRFC	0x02	0x04	0x05	0x07	0x09	0x0A
TRP	0x01	0x02	0x02	0x03	0x03	0x03
TRRD	0x01	0x01	0x01	0x02	0x02	0x02
TWR_INT	0x02	0x02	0x02	0x02	0x02	0x02
TWTR	0x02	0x02	0x02	0x02	0x02	0x02
TXSNR	0x0003	0x0006	0x0008	0x000C	0x000F	0x0010
TXSR	0x0003	0x0006	0x0008	0x000C	0x000F	0x0010

12.6.3.1 Bypass Cutoff

Suggested bypass cutoff frequency is 72 MHz.

12.6.3.2 Bypass Mode Enabled

Table 12-98. Delays

Parameter	24 MHz	48 MHz	60 MHz	72 MHz
DLL_DQS_DELAY_BYPASS_0	0x19	0x0D	0x0B	0x0C
DLL_DQS_DELAY_BYPASS_1	0x19	0x0D	0x0B	0x0C
DQS_OUT_SHIFT_BYPASS	0x01	0x01	0x01	0x01
WR_DQS_SHIFT_BYPASS	0x18	0x0D	0x0A	0x0C

12.6.3.3 Bypass Mode Disabled

Table 12-99. DLL

Parameter	60 MHz	96 MHz	120 MHz	133 MHz
DLL_INCREMENT	0x22	0x15	0x11	0x0F
DLL_START_POINT	0x7E	0x25	0x19	0x19

Table 12-100. Delays

Parameter	Value
DLL_DQS_DELAY_1	0x0D
DLL_DQS_DELAY_0	0x0D
DQS_OUT_SHIFT	0x7F
WR_DQS_SHIFT	0x20

12.6.3.4 Example Register Settings

```
////////////////////////////////////////////////////////////////////////////////
//
// Filename: mobile_ddr_mt46h16m16lf_7.5_regs_3_120MHz.h
//
// Description: Initialization register values for EMI DRAM controller
//
////////////////////////////////////////////////////////////////////////////////
//
// WARNING!  THIS FILE IS AUTOMATICALLY GENERATED.
//           DO NOT MODIFY THIS FILE DIRECTLY.
//
// SETTINGS USED TO GENERATE THIS FILE:
//
//            Burst: 4
//              CAS: 3
//             Freq: 120 MHz
//   Auto-Precharge: 0
//       DLL_Bypass: 0
//
////////////////////////////////////////////////////////////////////////////////

{
    0x01010001,  /* 0x01 ahb0_w_priority 0x01 ahb0_r_priority 0x00 ahb0_fifo_type_reg
                    0x01 addr_cmp_en */
```

```
    0x00010100,  /* 0x00 ahb2_fifo_type_reg 0x01 ahb1_w_priority 0x01 ahb1_r_priority
                    0x00 ahb1_fifo_type_reg */
    0x01000101,  /* 0x01 ahb3_r_priority 0x00 ahb3_fifo_type_reg 0x01 ahb2_w_priority
                    0x01 ahb2_r_priority */
    0x00000001,  /* 0x00 auto_refresh_mode 0x00 arefresh 0x00 ap 0x01 ahb3_w_priority */
   0x00000101,  /* 0x00 dll_bypass_mode 0x00 dlllockreg 0x01 concurrentap 0x01 bank_split_en */
   0x00000001,  /* 0x00 intrptreada 0x00 intrptapburst 0x00 fast_write 0x01 en_lowpower_mode */
    0x00010000,  /* 0x00 power_down 0x01 placement_en 0x00 no_cmd_init 0x00 intrptwritea */
    0x01000001,  /* 0x01 rw_same_en 0x00 reg_dimm_enable 0x00 rd2rd_turn 0x01 priority_en */
    0x01000000,  /* 0x01 tras_lockout 0x00 start 0x00 srefresh 0x00 sdr_mode */
    0x00000001,  /* 0x00 out_of_range_type 0x00 out_of_range_source_id 0x00 write_modereg
                    0x01 writeinterp */
    0x07000200,  /* 0x07 age_count 0x00 addr_pins 0x02 temrs 0x00 q_fullness */
    0x00070303,  /* 0x00 max_cs_reg 0x07 command_age_count 0x03 column_size 0x03 caslat */
    0x02020002,  /* 0x02 twr_int 0x02 trrd 0x0002 tcke */
    0x06060a02,  /* 0x06 caslat_lin_gate 0x06 caslat_lin 0x0a aprebit 0x02 twtr */
    0x00000201,  /* 0x00 max_col_reg 0x00 lowpower_refresh_enable 0x02 initaref 0x01 cs_map */
    0x03050000,  /* 0x03 trp 0x05 tdal 0x00 port_busy 0x00 max_row_reg */
    0x02000000,  /* 0x02 tmrd 0x00 lowpower_control 0x00 lowpower_auto_enable 0x00 int_ack */
    0x1900110a,  /* 0x19 dll_start_point 0x00 dll_lock 0x11 dll_increment 0x0a trc */
    0x1e1e0000,  /* 0x1e dll_dqs_delay_1 0x1e dll_dqs_delay_0 0x00 int_status 0x00 int_mask */
    0x01011919,  /* 0x01 dqs_out_shift_bypass 0x01 dqs_out_shift 0x19 dll_dqs_delay_bypass_1
                    0x19 dll_dqs_delay_bypass_0 */
    0x03061820,  /* 0x03 trcd_int 0x06 tras_min 0x18 wr_dqs_shift_bypass 0x20 wr_dqs_shift */
    0x00000009,  /* 0x000000 out_of_range_length 0x09 trfc */
    0x00080008,  /* 0x0008 ahb0_wrcnt 0x0008 ahb0_rdcnt */
    0x00200020,  /* 0x0020 ahb1_wrcnt 0x0020 ahb1_rdcnt */
    0x00200020,  /* 0x0020 ahb2_wrcnt 0x0020 ahb2_rdcnt */
    0x00200020,  /* 0x0020 ahb3_wrcnt 0x0020 ahb3_rdcnt */
    0x000003a1,  /* 0x000003a1 tref */
    0x00000000,  /* 0x00000000 */
    0x00000000,  /* 0x00000000 */
    0x00000000,  /* 0x0000 lowpower_internal_cnt 0x0000 lowpower_external_cnt */
    0x00000000,  /* 0x0000 lowpower_refresh_hold 0x0000 lowpower_power_down_cnt */
    0x00000000,  /* 0x0000 tdll 0x0000 lowpower_self_refresh_cnt */
    0x000f20ca,  /* 0x000f txsnr 0x20ca tras_max */
    0x0000000f,  /* 0x0000 version 0x000f txsr */
    0x00005dca,  /* 0x00005dca tinit */
    0x00000000,  /* 0x00000000 out_of_range_addr */
    0x00000101,  /* 0x00 pwrup_srefresh_exit 0x00 enable_quick_srefresh
                    0x01 bus_share_enable 0x01 active_aging */
    0x00040001,  /* 0x000400 bus_share_timeout 0x01 tref_enable */
    0x00400000,  /* 0x0040 emrs2_data_0 0x0000 emrs1_data */
    0x00400040,  /* 0x0040 emrs2_data_2 0x0040 emrs2_data_1 */
    0x00040040,  /* 0x0004 tpdex 0x0000 emrs2_data_3 */
},
```

12.6.4 DDR

Table 12-101. Frequency Dependent Parameters

Parameter	80 MHz	96 MHz	120 MHz	133 MHz	167 MHz
TCKE	0x0000	0x0000	0x0000	0x0000	0x0000
TDAL	0x04	0x04	0x04	0x04	0x05
TDLL	0x00C8	0x00C8	0x00C8	0x00C8	0x00C8
TEMRS	0x00	0x00	0x00	0x00	0x00
TINIT	0x00003E80	0x00004B0D	0x00005DCA	0x00006665	0x000081C7
TMRD	0x01	0x02	0x02	0x02	0x02
TPDEX	0x0001	0x0001	0x0001	0x0001	0x0001
TRAS_MAX	0x15D6	0x1A3B	0x20CA	0x23CD	0x2D62
TRAS_MIN	0x04	0x05	0x06	0x06	0x07
TRC	0x05	0x06	0x08	0x08	0x0A
TRCD_INT	0x02	0x02	0x02	0x02	0x03
TREF	0x00000269	0x000002E6	0x000003A1	0x000003F7	0x00000509
TRFC	0x0006	0x0007	0x0009	0x000A	0x000C
TRP	0x02	0x02	0x02	0x02	0x03
TRRD	0x01	0x02	0x02	0x02	0x02
TWR_INT	0x02	0x02	0x02	0x02	0x02
TWTR	0x01	0x01	0x01	0x01	0x01
TXSNR	0x0006	0x0008	0x000A	0x000A	0x000D
TXSR	0x00C8	0x00C8	0x00C8	0x00C8	0x00C8

12.6.4.1 Bypass Mode Disabled

NOTE: DDR should always be run with bypass disabled.

Table 12-102. DLL

Parameter	80 MHz	96 MHz	120 MHz	133 MHz	167 MHz
DLL_INCREMENT	0x1C	0x17	0x13	0x11	0x0D
DLL_START_POINT	0x20	0x2F	0x26	0x22	0x18

Table 12-103. Delays

Parameter	80 MHz	96 MHz	120 MHz	133 MHz	160 MHz
DLL_DQS_DELAY_1	0x1F	0x1F	0x1F	0x1F	0x1F
DLL_DQS_DELAY_0	0x1F	0x1F	0x1F	0x1F	0x1F
DQS_OUT_SHIFT	0x7F	0x7F	0x7F	0x7F	0x7F
WR_DQS_SHIFT	0x22	0x22	0x23	0x23	0x24

12.6.4.2 Example Register Settings

```
//////////////////////////////////////////////////////////////////////////////
//
// Filename: ddr_mt46v32m16_6t_regs_2.5_167MHz.h
//
// Description: Initialization register values for EMI DRAM controller
//
//////////////////////////////////////////////////////////////////////////////
//
// WARNING!  THIS FILE IS AUTOMATICALLY GENERATED.
//           DO NOT MODIFY THIS FILE DIRECTLY.
//
// SETTINGS USED TO GENERATE THIS FILE:
//
//            Burst: 4
//              CAS: 2.5
//             Freq: 167MHz
//   Auto-Precharge: 0
//       DLL_Bypass: 0
//
//////////////////////////////////////////////////////////////////////////////

{
    0x01010001,  /* 0x01 ahb0_w_priority 0x01 ahb0_r_priority 0x00 ahb0_fifo_type_reg
                    0x01 addr_cmp_en */
    0x00010100,  /* 0x00 ahb2_fifo_type_reg 0x01 ahb1_w_priority 0x01 ahb1_r_priority
                    0x00 ahb1_fifo_type_reg */
    0x01000101,  /* 0x01 ahb3_r_priority 0x00 ahb3_fifo_type_reg 0x01 ahb2_w_priority
                    0x01 ahb2_r_priority */
    0x00000001,  /* 0x00 auto_refresh_mode 0x00 arefresh 0x00 ap 0x01 ahb3_w_priority */
   0x00000101,  /* 0x00 dll_bypass_mode 0x00 dlllockreg 0x01 concurrentap 0x01 bank_split_en */
   0x00000000,  /* 0x00 intrptreada 0x00 intrptapburst 0x00 fast_write 0x00 en_lowpower_mode */
    0x00010000,  /* 0x00 power_down 0x01 placement_en 0x00 no_cmd_init 0x00 intrptwritea */
    0x01000001,  /* 0x01 rw_same_en 0x00 reg_dimm_enable 0x00 rd2rd_turn 0x01 priority_en */
    0x01000000,  /* 0x01 tras_lockout 0x00 start 0x00 srefresh 0x00 sdr_mode */
    0x00000001,  /* 0x00 out_of_range_type 0x00 out_of_range_source_id 0x00 write_modereg
                    0x01 writeinterp */
    0x07000200,  /* 0x07 age_count 0x00 addr_pins 0x02 temrs 0x00 q_fullness */
    0x00070206,  /* 0x00 max_cs_reg 0x07 command_age_count 0x02 column_size 0x06 caslat */
    0x02020000,  /* 0x02 twr_int 0x02 trrd 0x0000 tcke */
    0x05050a01,  /* 0x05 caslat_lin_gate 0x05 caslat_lin 0x0a aprebit 0x01 twtr */
   0x0000020f,  /* 0x00 max_col_reg 0x00 lowpower_refresh_enable 0x02 initaref 0x01 cs_map */
    0x03050000,  /* 0x03 trp 0x05 tdal 0x00 port_busy 0x00 max_row_reg */
    0x02000000,  /* 0x02 tmrd 0x00 lowpower_control 0x00 lowpower_auto_enable 0x00 int_ack */
    0x18000d0a,  /* 0x18 dll_start_point 0x00 dll_lock 0x0d dll_increment 0x0a trc */
   0x1f1f0000,  /* 0x1f dll_dqs_delay_1 0x1f dll_dqs_delay_0 0x00 int_status 0x00 int_mask */
    0x027f0f0f,  /* 0x02 dqs_out_shift_bypass 0x7f dqs_out_shift 0x0f dll_dqs_delay_bypass_1
                    0x0f dll_dqs_delay_bypass_0 */
    0x03071124,  /* 0x03 trcd_int 0x07 tras_min 0x11 wr_dqs_shift_bypass 0x24 wr_dqs_shift */
    0x0000000c,  /* 0x000000 out_of_range_length 00001100 trfc */
    0x00080008,  /* 0x0008 ahb0_wrcnt 0x0008 ahb0_rdcnt */
    0x00200020,  /* 0x0020 ahb1_wrcnt 0x0020 ahb1_rdcnt */
    0x00200020,  /* 0x0020 ahb2_wrcnt 0x0020 ahb2_rdcnt */
    0x00200020,  /* 0x0020 ahb3_wrcnt 0x0020 ahb3_rdcnt */
    0x00000509,  /* 0x00000509 tref */
    0x00000000,  /* 0x00000000 */
    0x00000000,  /* 0x00000000 */
```

```
    0x00000000,  /* 0x0000 lowpower_internal_cnt 0x0000 lowpower_external_cnt */
    0x00000000,  /* 0x0000 lowpower_refresh_hold 0x0000 lowpower_power_down_cnt */
    0x00c80000,  /* 0x00c8 tdll 0x0000 lowpower_self_refresh_cnt */
    0x000d2d62,  /* 0x000d txsnr 0x2d62 tras_max */
    0x000000c8,  /* 0x0000 version 0x00c8 txsr */
    0x000081c7,  /* 0x000081c7 tinit */
    0x00000000,  /* 0x00000000 out_of_range_addr */
    0x00000101,  /* 0x00 pwrup_srefresh_exit 0x00 enable_quick_srefresh
                    0x01 bus_share_enable 0x01 active_aging */
    0x00040001,  /* 0x000400 bus_share_timeout 0x01 tref_enable */
    0x00000000,  /* 0x0000 emrs2_data_0 0x0000 emrs1_data */
    0x00000000,  /* 0x0000 emrs2_data_2 0x0000 emrs2_data_1 */
    0x00010000,  /* 0x0001 tpdex 0x0000 emrs2_data_3 */
},
```

Chapter 13
General-Purpose Media Interface (GPMI)

This chapter describes the general-purpose media interface (GPMI) on the i.MX23. Programmable registers are described in Section 13.4, “Programmable Registers.”

13.1 Overview

The general-purpose media interface (GPMI) controller is a flexible interface to up to four NAND Flash. The NAND Flash mode has configurable address and command behavior, providing support for future devices not yet specified.

The GPMI resides on the APBH. The GPMI also provides an interface to the ECC8 and BCH modules to allow direct parity processing.

Registers are clocked on the HCLK (CLK_H) domain. The I/O and pin timing are clocked on a dedicated GPMICLK domain. GPMICLK can be set to maximize I/O performance.

Figure 13-1 shows a block diagram of the GPMI controller.

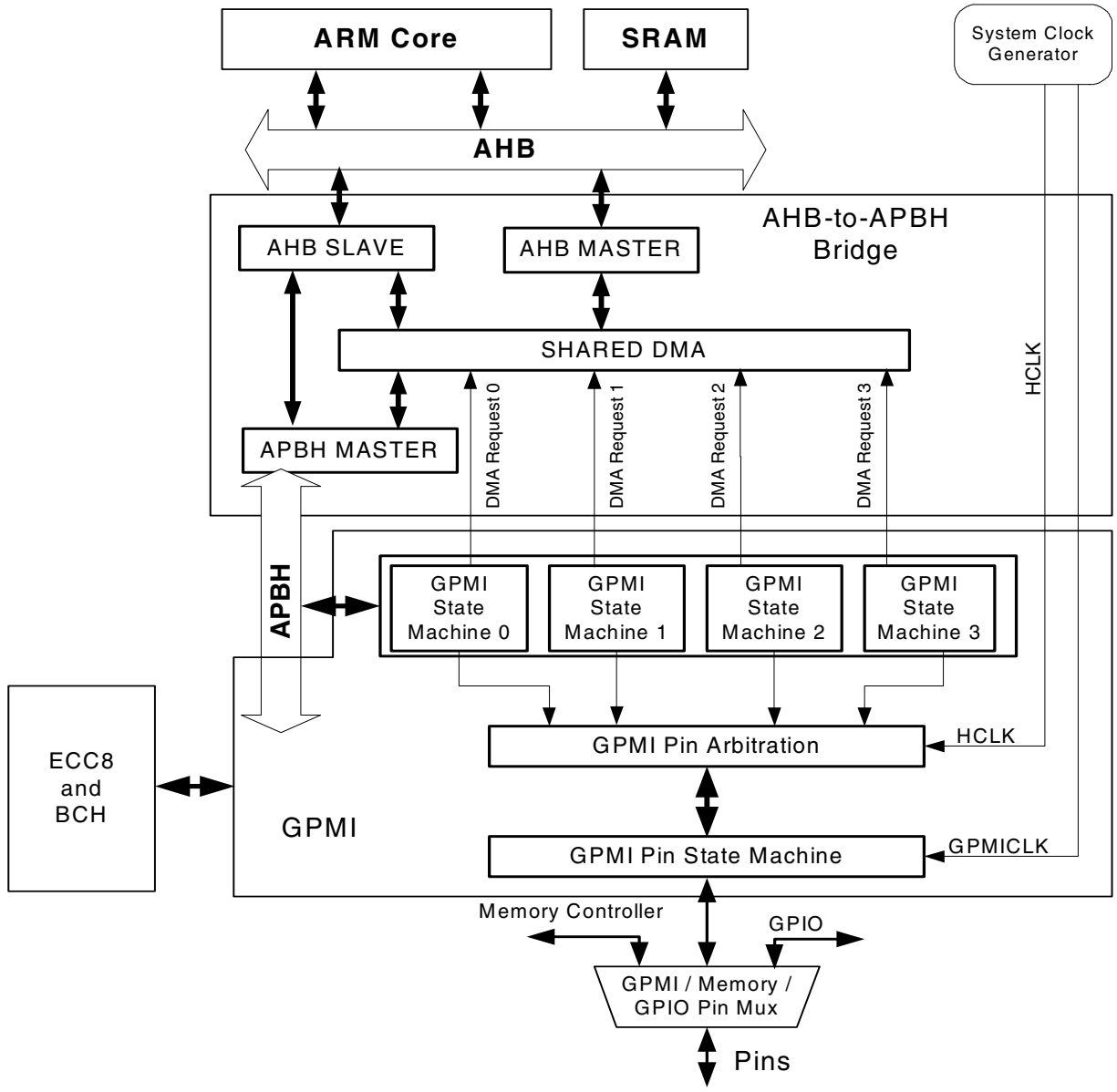


Figure 13-1. General-Purpose Media Interface Controller Block Diagram

13.2 GPMI NAND Flash Mode

The general-purpose media interface has several features to efficiently support NAND Flash:

- Individual chip select and ready/busy pins for four NAND Flash.
- Individual state machine and DMA channel for each chip select.
- Special command modes work with DMA controller to perform all normal NAND Flash functions without CPU intervention.

- Configurable timing based on a dedicated clock allows optimal balance of high NAND Flash performance and low system power.

Since current NAND Flash does not support multiple page read/write commands, the GPMI and DMA have been designed to handle complex multi-page operations without CPU intervention. The DMA uses a linked descriptor function with branching capability to automatically handle all of the operations needed to read/write multiple pages:

- **Data/Register Read/Write**—The GPMI can be programmed to read or write multiple cycles to the NAND Flash address, command or data registers.
- **Wait for NAND Flash Ready**—The GPMI's Wait-for-Ready mode can monitor the ready/busy signal of a single NAND Flash and signal the DMA when the device has become ready. It also has a time-out counter and can indicate to the DMA that a time-out error has occurred. The DMAs can conditionally branch to a different descriptor in the case of an error.
- **Check Status**—The Read-and-Compare mode allows the GPMI to check NAND Flash status against a reference. If an error is found, the GPMI can instruct the DMA to branch to an alternate descriptor, which attempts to fix the problem or asserts a CPU IRQ.

13.2.1 Multiple NAND Flash Support

The GPMI supports up to four NAND Flash chip selects, each with independent ready/busy signals. Since they share a data bus and control lines, the GPMI can only actively communicate with a single NAND Flash at a time. However, all NAND Flashes can concurrently perform internal read, write, or erase operations. With fast NAND Flash and software support for concurrent NAND Flash operations, this architecture allows the total throughput to approach the data bus speed, which can be as high as 80 MB/s (16-bit bus running at 40 MHz).

13.2.2 GPMI NAND Flash Timing and Clocking

The dedicated clock, GPMICLK, is used as a timing reference for NAND Flash I/O. Since various NAND Flashes have different timing requirements, GPMICLK may need to be adjusted for each application. While the actual pin timings are limited by the NAND Flash chips used, the GPMI can support data bus speeds of up to 33 MHz x 16 bits. The actual read/write strobe timing parameters are adjusted as indicated in the register descriptions in Section 13.4, "Programmable Registers." Refer to Chapter 4, "Clock Generation and Control," for more information about setting GPMICLK.

13.2.3 Basic NAND Flash Timing

Figure 13-2 illustrates the operation of the timing parameters in NAND Flash mode.

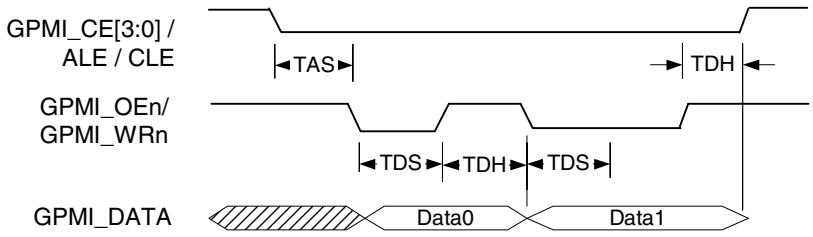


Figure 13-2. BASIC NAND Flash Timing

13.2.4 High-Speed NAND Flash Timing

In high-speed NAND Flashes, the read data may not be valid until after the read strobe (RDN) deasserts. This is the case when the minimum tDS is programmed to achieve higher bandwidth. The GPMI implements a feedback read strobe to sample the read data. The feedback read strobe can be delayed to support fast NAND Flash EDO (Extended Data Out) timing where the read strobe may deassert before the read data is valid, and read data is valid for some time after read strobe. NAND Flash EDO timings is applied typically for read cycle frequency above 33MHz. See Figure 13-3.

The GPMI provides control over the amount of delay applied to the feedback read strobe. This delay depends on the maximum read access time (tREA) of the NAND Flash and the read pulse width (tRP) used to access the NAND Flash. tRP is specified by HW_GPMI_TIMING0_DATA_SETUP register. When (tREA + 4ns) is less than tRP, no delay is required to sample to NAND Flash read data. (The 4ns provides adequate data setup time for the GPMI.) In this case set HW_GPMI_CTRL1_HALF_PERIOD=0; HW_GPMI_CTRL1_RDN_DELAY=0; HW_GPMI_CTRL1_DLL_ENABLE=0.

When (tREA + 4ns) is greater than or equal to tRP, a delay of the feedback read strobe is required to sample to NAND Flash read data. This delay is equal to the difference between these two timings:

DELAY= tREA+4ns - tRP.

Since the GPMI delay chain is limited to 16ns maximum, if DELAY > 16ns then increase tRP by increasing the value of HW_GPMI_TIMING0_DATA_SETUP until DELAY is less than or equal to 16ns.

The GPMI programming for this DELAY depends on the GPMICLK period. The GPMI DLL will not function properly if the GPMICLK period is greater than 32ns: disable the DLL if this is the case. If the GPMICLK period is greater than 16ns (and not greater than 32ns), set the HW_GPMI_CTRL1_HALF_PERIOD=1; This will cause the DLL reference period (RP) to be one-half of the GPMICLK period. If the GPMICLK period is 16ns or less then set the

HW_GPMI_CTRL1_HALF_PERIOD=0; This will cause the DLL reference period (RP) to be equal to the GPMICLK period. DELAY is a multiple (0 to 1.875) of RP.

The HW_GPMI_CTRL1_RDN_DELAY is encoded as a 1-bit integer and 3-bit fraction delay factor. See table below. DELAY is a multiple of the delay factor and the reference period:

Table 13-1.

HW_GPMI_CTRL1_RDN_DELAY	0	1	2	3	4	5	6	7
Delay Factor	0.000	0.125	0.250	0.375	0.500	0.625	0.750	0.875
HW_GPMI_CTRL1_RDN_DELAY	8	9	10	11	12	13	14	15
Delay Factor	1.000	1.125	1.250	1.375	1.500	1.625	1.750	1.875

— DELAY = DelayFactor x RP or

— DELAY = HW_GPMI_CTRL1_RDN_DELAY x 0.125 x RP.

Use this equation to calculate the value for HW_GPMI_CTRL1_RDN_DELAY. Then set HW_GPMI_CTRL1_DLL_ENABLE=1.

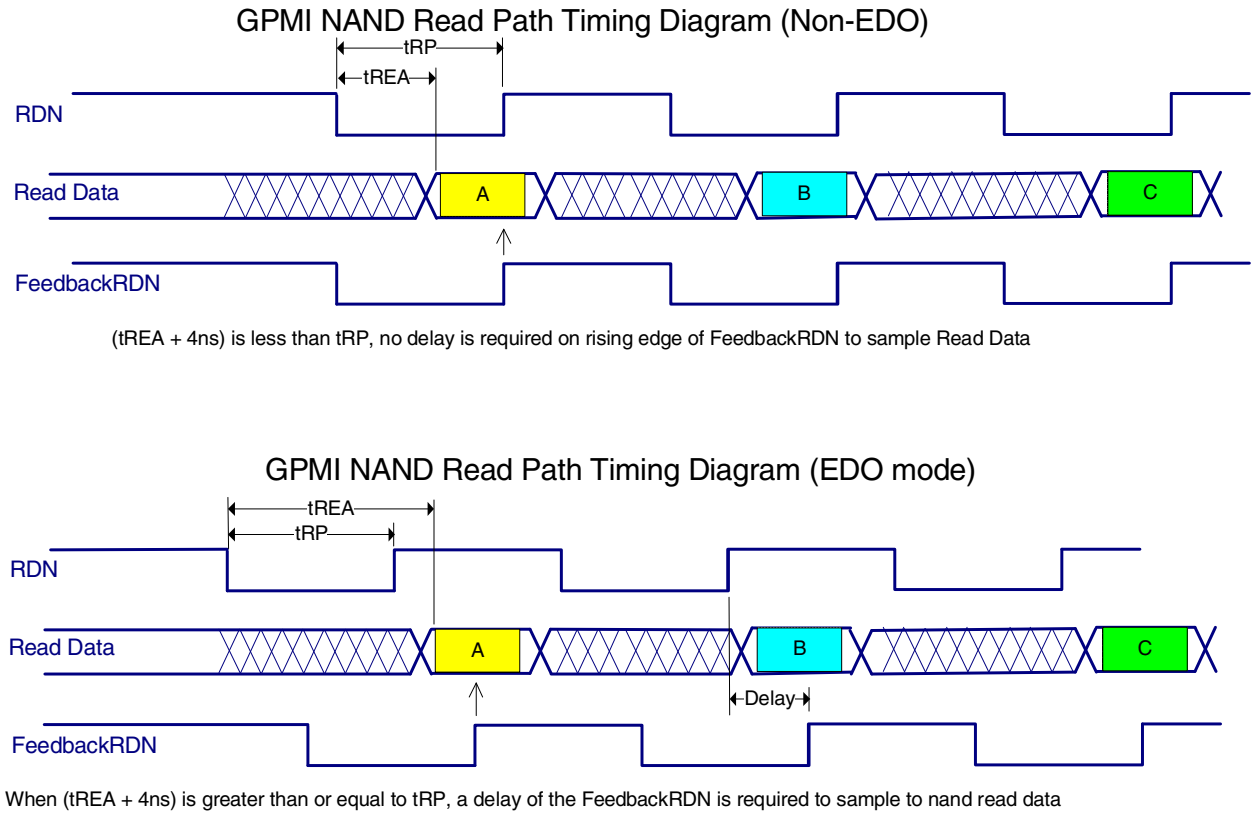


Figure 13-3. NAND Flash Read Path Timing

For example, a NAND Flash with tREAmax=20ns, tRPmin=12ns, and tRCmin=25ns (read cycle time) may be programmed as follows:

- GPMICLK clock frequency: Consider 480/6=80MHz which is 12.5ns clock period. This is too close to the minimum NAND Flash spec if we program the data setup and hold to 1 GPMICLK cycle. Consider 480/7=68.57MHz which is 14.58ns clock period. With data setup and hold set to 1, we have a tRP of 14.58ns and a tRC of 29.16ns (good margins).
- Since (tREA +4ns) is greater than tRP, required DELAY = tREA+4ns - tRP = 20 + 4 - 14.58ns = 9.42ns.
- HALF_PERIOD =0, since GPMICLK period is less than 16ns. So RP=GPMICLK period = 14.58ns.
- DELAY = HW_GPMI_CTRL1_RDN_DELAY x 0.125 x RP.
  9.42ns = HW_GPMI_CTRL1_RDN_DELAY x 0.125 x 14.58ns.
  HW_GPMI_CTRL1_RDN_DELAY = 5 (round off 5.169)

Note: *It is recommended that the drive strength of GPMI_RDn and GPMI_WRn output pins be set to 8 mA. This will reduce the transition time under heavy loads. Low transition times will be important when NAND Flash interface read and write cycle times are below 30 ns. The other GPMI pins may remain at 4 mA, since their frequency is only up to half that of GPMI_RDn and GPMI_WRn.*

13.2.5 NAND Flash Command and Address Timing Example

Figure 13-4 illustrates a command and address being sent to a NAND Flash.

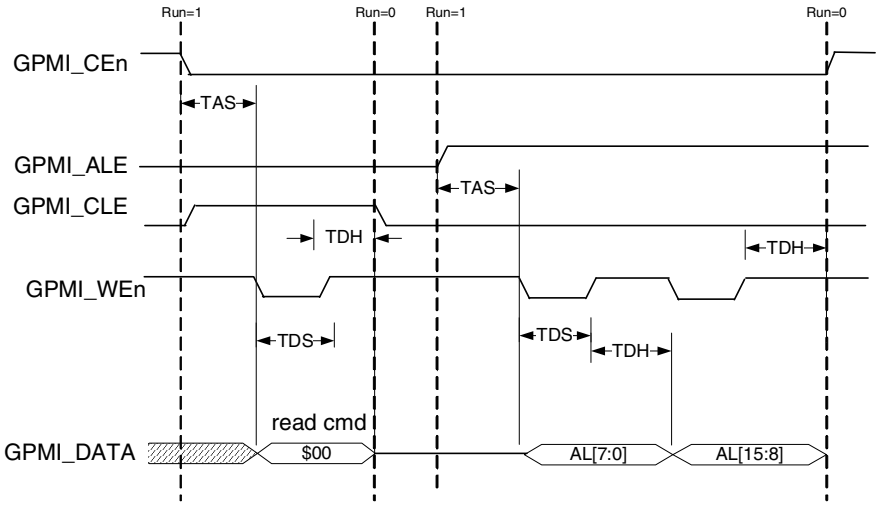


Figure 13-4. NAND Flash Command and Address Timing Example

13.2.6 Hardware BCH/ECC (ECC8) Interface

The GPMI provides an interface to the ECC8 module. This same interface is used by the BCH module when in BCH mode. This reduces the SoC bus traffic and the software involvement. When in BCH or

ECC8 mode, parity information is inserted on-the-fly during writes to 8-bit NAND Flash devices. (Note that ECC8 mode is not avaiable with 16-bit devices.) During NAND Flash reads, parity is checked and ECC processing is performed after each read block.

In ECC8 mode, during NAND Flash writes, each 512-byte block of payload data is sent to the ECC8 module at the same time it is sent to the NAND Flash. The ECC8 module returns the parity information, which is then appended to the block of data written to the NAND Flash. This is repeated for each block of data written to the NAND Flash. During NAND Flash reads, each block of payload data and parity is redirected to the ECC8 module for ECC processing and memory write, instead of DMA to memory. This works the same way for BCH mode.

To program the ECC8 for NAND Flash writes, remove the soft reset and clock gates from HW_ECC8_CTRL_SFTRST and HW_ECC8_CTRL_CLKGATE. The bulk ECC8 programming is actually applied to the GPMI via PIO operations embedded in its DMA command structures. This has a subtle implication when writing to the GPMI ECC8 registers: access to the these registers must be written in progressive register order. Thus, to write to the HW_GPMI_ECCCOUNT register, write first (in order) to registers HW_GPMI_CTRL0, HW_GPMI_COMPARE, and HW_GPMI_ECCCTRL before writing to HW_GPMI_ECCCOUNT. These additional register writes need to be accounted for in the CMDWORDS field of the respective DMA channel command register. See the descriptive text, flowcharts, and code examples in Section 14.2.1, “Reed-Solomon ECC Accelerator,” Section 14.2.2, “Reed-Solomon ECC Encoding for NAND Writes,” and Section 14.2.3, “Reed-Solomon ECC Decoding for NAND Reads” for more information about using GPMI registers to program the ECC8 function.

Note that the HW_GPMI_PAYLOAD and HW_GPMI_AUXILIARY pointers need to be word-aligned for proper ECC8 operation. If those pointers are non-word-aligned, then the ECC8 engine will not operate properly and could possibly corrupt system memory in the adjoining memory regions.

Note that when using DMA to read from the NAND, there are two possible interrupts:

- GPMI DMA completion interrupt
- ECC engine completion interrupt (only used when the data is read using ECC in the DMA descriptor)

When the NAND is read using ECC (as configured in the DMA descriptor for that read), the CPU must wait for **both** interrupts, and the interrupts may occur in either order. That is, the GPMI interrupt may happen before the ECC interrupt, or vice-versa. Software needs to wait until both interrupts have occurred to know that the data has been delivered by the DMA.

When the NAND is read without using ECC (as configured in the DMA descriptor for that read), the CPU only needs to wait for the GPMI interrupt. (Indeed, there will be no ECC interrupt, because the ECC engine is not in use.).

13.3 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 39.3.10, “Correct Way to Soft Reset a Block” for additional information on using the SFTRST and CLKGATE bit fields.

13.4 Programmable Registers

The following registers provide control for programmable elements of the GPMI module.

13.4.1 GPMI Control Register 0 Description

The GPMI control register 0 specifies the GPMI transaction to perform for the current command chain item.

HW_GPMI_CTRL0	0x000
HW_GPMI_CTRL0_SET	0x004
HW_GPMI_CTRL0_CLR	0x008
HW_GPMI_CTRL0_TOG	0x00C

Table 13-2. HW_GPMI_CTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RUN	RSVD3	TIMEOUT_IRQ_EN	RSVD2	COMMAND_MODE		WORD_LENGTH	LOCK_CS	CS		RSVD1			ADDRESS_INCREMENT	XFER_COUNT															

Table 13-3. HW_GPMI_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. This will not work if the CLKGATE bit is already set to '1'. CLKGATE must be cleared to '0' before issuing a soft reset. Also the GPMICLK must be running for this to work properly. RUN = 0x0 Allow GPMI to operate normally. RESET = 0x1 Hold GPMI in reset.
30	**CLKGATE**	RW	0x1	Set this bit zero for normal operation. Setting this bit to one (default), gates all of the block level clocks off for miniminizing AC energy consumption. RUN = 0x0 Allow GPMI to operate normally. NO_CLKS = 0x1 Do not clock GPMI gates in order to minimize power consumption.