

A12 Hardware Programming Reference Manual

SUMMARY DESCRIPTION

The A12 Hardware Programming Reference Manual (PRM) lists software-programmable registers accessible from the A12 CPU core, providing comprehensive information on each field of each register, as well as step-by-step programming instructions.

The PRM document also includes an overview of the system memory map, power-on configuration options, and available ARM interrupts.

This reference manual is intended for use with both the 15-mm x 15-mm A12 system-on-a-chip (SoC) package, as well as with the smaller-package A12m family of chips.

However, please note that a number of the features, pins, interfaces, registers, and programming steps described in this document apply to the 15-mm x 15-mm A12 package only and may differ for the reduced-size A12m package. This key information has been called-out textually and labeled with the  symbol for easy reference.

For a complete list of features for a specific A12 or A12m SoC, please refer to the relevant Ambarella datasheet.

KEY TOPICS

- Register programming information
- Power-on configuration (POC) detail
- Boot mode summary
- DSP subsystem overview
- Detailed peripheral interface information
- Memory allocation summary

CONTENTS

1. Introduction	1
2. System Boot and Clock Configuration	10
3. DSP Subsystem	17
4. DMA Engine Subsystem.....	20
5. Graphics DMA (GDMA).....	41
6. Video / Sensor Input Interface	57
7. Video Output	58
8. HDMI and CEC	59
9. I2S Interface.....	118
10. Ethernet Controller	134
11. USB Device Controller.....	173
12. USB Host Controller	191
13. Flash and SD Controllers	223
14. Interrupt Controller (VIC).....	305
15. GPIO	320
16. SSI / SPI	348
17. UART	388
18. IDC	407
19. ADC, PWC and RTC	412
20. InfraRed (IR) Remote Interface	441
21. Stepper, Micro-Stepper and PWM	444
22. Pulse Width Modulator (PWM).....	463
23. Interval Timers	464
24. Watch Dog Timer (WDT).....	470
25. Cryptography Engine	473
26. Ambarella Contact Information	484
27. Important Notice	485
28. Typographical Conventions.....	486
29. Revision History.....	488

The material in this document is for information only. Ambarella assumes no responsibility for errors or omissions and reserves the right to change, without notice, product specifications, operating characteristics, packaging, ordering, etc. Ambarella assumes no liability for damage resulting from the use of information contained in this document. All brands, product names and company names are trademarks of their respective owners. Further information, including additional disclaimers, appears in the Important Notice at the end of this document.

1. INTRODUCTION

This document is the Hardware Programming Reference Manual (PRM) for the A12 and A12m family of system-on-a-chip (SoC) processors from Ambarella. The manual begins with a brief overview of the chip and follows with essentials for system boot / power-on configuration ([Chapter 2](#)) and details on the DSP subsystem ([Chapter 3](#)). Registers and programming information for A12 peripherals are provided in subsequent chapters ([Chapter 4 +](#)).

This overview continues with:

- [\(Section 1.1\) Introduction: Scope of Document](#)
- [\(Section 1.2\) Introduction: Note for Users of A12m Chips](#)
- [\(Section 1.3\) Introduction: Functional Block Diagram](#)
- [\(Section 1.4\) Introduction: Feature List](#)
- [\(Section 1.5\) Introduction: Memory Map](#)

1.1 Introduction: Scope of Document

This manual contains programming reference materials intended to support A12-based camera design efforts. The reader is assumed to be generally familiar with the A12 or A12m chip feature list, pin map, and peripheral interfaces.

Please refer to the following documents for supporting information on the A12 family of processors.

- The A12 family of datasheets provides chip-specific information on performance features, peripheral interfaces, external pins, electrical characteristics, and package information.
- The “*A12 System Hardware*” document provides guidelines for power management, the implementation of various peripheral interfaces, and PCB layout.
- Micron SPI-NOR Specification: n25q_1gb_3v_65nm, n25q_512mb_1ce_3v_65nm

1.2 Introduction: Note for Users of A12m Chips

While this document applies to both the A12 and A12m packages, please note that a number of the features, pins, interfaces, registers, and programming steps described in these reference materials are applicable to the 15-mm x 15-mm A12 package only and may differ for the reduced-sized A12m package.

This key information has been called-out textually and labeled with the  symbol for easy reference.

Users of A12m SoCs are expected to take note of this key information in order to ensure that their software development efforts remain aligned with their specific A12m chip’s features and capabilities.

1.3 Introduction: Functional Block Diagram

The following is a functional block diagram showing the A12 peripherals according to bus location. Please contact an Ambarella representative for further information.

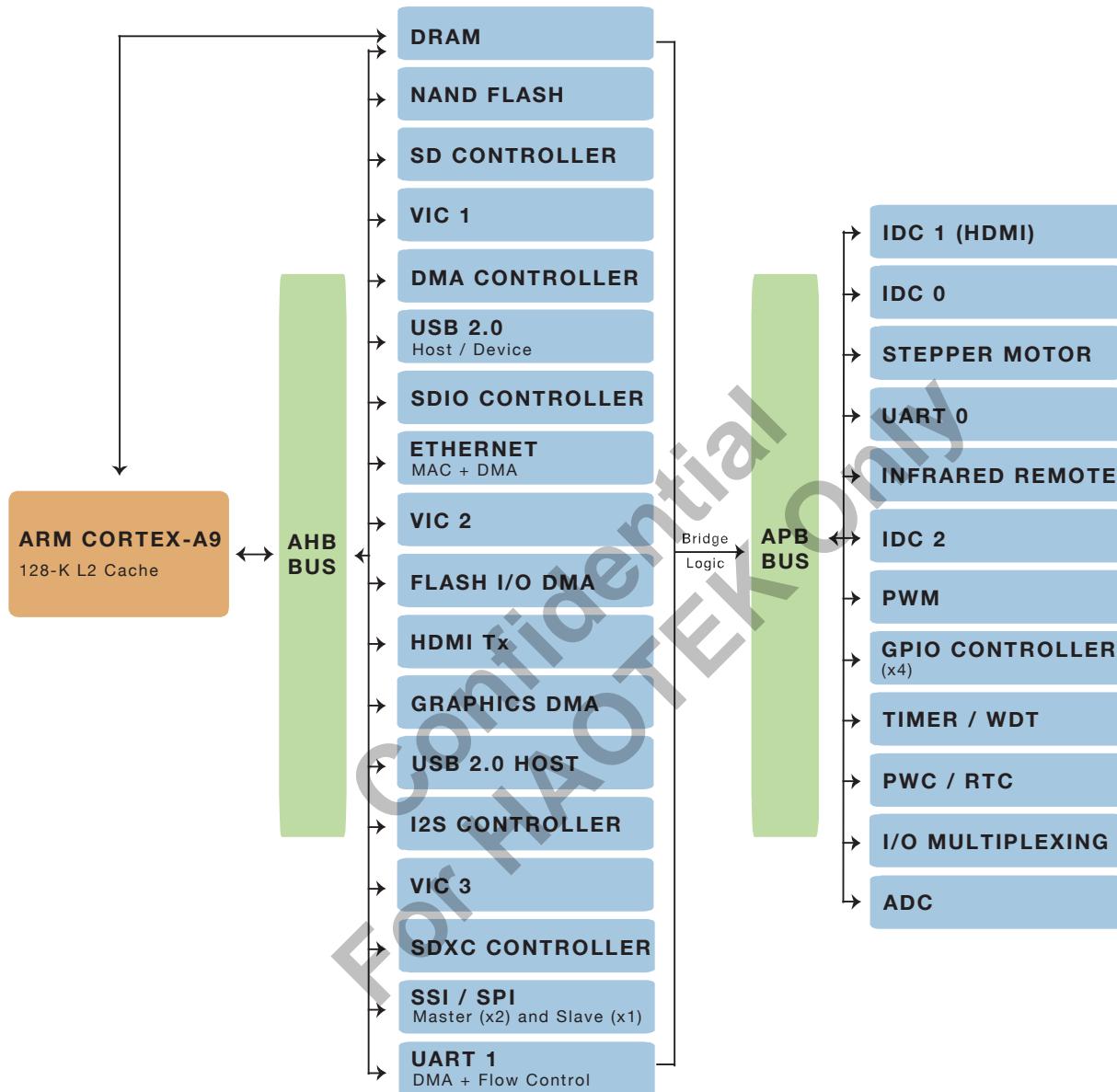


Figure 1-1. Functional Block Diagram for the A12 Chip.

Notes:

- The AHB bus (32-bit width) provides a high-performance interconnect between the ARM Cortex-A9, the SDRAM controller, and high-performance peripherals such as the DMA engine and I2S controllers. The AHB bus connects to an APB bus via bridge logic.
 - The APB bus (32-bit width) connects many lower-performance peripherals including timers and the stepper motor controllers.
- (i)** A12m packages may not include every functional block shown above.

1.4 Introduction: Feature List

A general list of features for the 15-mm x 15-mm A12 SoC is provided below.

Note that the smaller-package A12m family of chips does not include every feature listed in this section. For a comprehensive list of features for a specific Ambarella SoC, please refer to the relevant datasheet.

- Embedded single-core ARM Cortex-A9 CPU
 - Clock frequency up to 1.0 GHz
 - 32-KByte data / 32-KByte instruction cache
 - 128-KByte L2 cache
 - NEON SIMD engine
- DDR3 and DDR3L controller
 - Up to 600-MHz clock rate
 - 32-bit wide data bus
 - Maximum capacity of 4 Gbits (512 MBytes)
 - Includes support for the LPDDR2 and LPDDR3 low-power DDR interfaces
- Image pipeline
 - More than 480 MPixel/s processing rate
 - 64-MPixel maximum resolution
 - Fish-eye lens dewarping and barrel distortion correction
 - Black level correction
 - Dynamic and static defect pixel cluster correction
 - CFA crosstalk and fixed-pattern noise reduction
 - RGB Bayer demosaicing
 - Lens shading
 - 3D LUT color transform with gamma
 - Advanced motion-compensated sharpening
 - Advanced dynamic range (WDR and HDR) engine with multi-exposure fusion and motion artifact reduction
 - Per-pixel local exposure dynamic range enhancement
 - Tone mapping and global tone-curve adjustment
 - Chromatic aberration correction
 - Flexible APIs and image-tuning tools
 - Adjustable 3A; exposure, white balance and focus control (AE/AWB/AF)
 - Day/Night and DC/P iris control
 - Crop, mirror, flip, 90°/270° rotation
 - Image stabilization with rolling shutter compensation

- Video engine
 - H.264 MP/HP Level 5.1 and MJPEG codecs
 - Encode performance up to 1080p100
 - Advanced H.264 compression tools
 - I, IP, IBP modes ($M=1,2,3,4\dots$; IP, IBP, IBBP, IBBBP...)
 - High Profile with B-frames and hierarchical GOP
 - Up to three reference frames
 - Flexible rate control
 - CBR, VBR and Constant QP with max bitrate control
 - Macroblock-level adaptive quantization
 - 3D noise reduction (MCTF)
- Sensor/Video Input (VIN) interfaces
 - Two input channels with multiple serial input modes
 - Primary channel supports up to 8-lane SLVS / HiSPi or 4-lane MIPI input
 - Secondary channel supports up to 2-lane SLVS / HiSPi / MIPI input
 - In SVLS mode, the two input channels may be combined to support a single 10-lane SLVS / HiSPi sensor
 - Support for popular CMOS sensors; Aptina, Sony, OV, Panasonic, Samsung
 - Two clocking options (PLL-generated GCLK_VIN or SLVS bit clock)
 - 16-bit CCIR.601 video input with external sync signals
 - 8-bit, 10-bit, 12-bit or 14-bit BT.656 video input with embedded sync codes including full-data-range support
- Video Output (VOUT) interfaces
 - Two logical channels to drive three video output ports
 - One logical channel drives HDMI or analog
 - One logical channel drives digital
 - Popular LCD panel controllers (RGB mode)
 - Support for RGBA and YUVA OSD
 - Video DAC for 480i/576i composite PAL/NTSC output
 - BT.656 embedded sync YUV output (8-bit or 16-bit mode)
 - HDMI 1.4b output with Consumer Electronics Control (CEC) and on-chip PHY
- AHB Bus DMA controller
 - Memory-to-memory transfers including support for transfers between memory and peripherals
 - Programmable transfer count up to 4 MB
 - DMA scatter/gather via chained descriptor list in memory with DMA control information source

- Dedicated DMA co-processor for graphics and image operations
 - Offers linear copy, 2-D copy, composite, and alpha-blend image operations
 - Supports 4- to 32-bit pixel formats
- I2S digital audio interface (stereo)
 - Audio encoding/decoding
- Ethernet MAC controller
 - IEEE 802.3 compliant with full- and half-duplex (IEEE 802.3x flow-control) and Jumbo frames
 - IEEE 802.1Q VLAN tag detection
 - Checksum off-load for received IP and TCP/UDP packets
 - Dedicated pins for RMII or MII interface
 - FIFO (2 KB / 2 KB) and DMA support
- Two USB 2.0 interfaces
 - One host and one configurable host/device interface, each with built-in PHY
- Flexible Storage Media Input / Output (SMIO) interface
 - NAND Flash controller
 - Up to 8-Gbit device, 512-Byte and 2-KByte page sizes
 - 8-bit flash chip data bus
 - 4-bit and 8-bit SLC with ECC hardware and read-confirm support
 - BCH error correction and increased spare area available
 - Three SD controllers (SD0, SD1, SD2)
 - SD0:
 - SDIO v3.0, SD, SDHC, SDXC, MMC and eMMC operation with boot support and UHS-I speed
 - SD1:
 - SDIO v3.0, SD, SDHC, SDXC, MMC and eMMC operation
 - SD2:
 - SDIO v1.0, SD, SDHC, SDXC, MMC and eMMC operation
 - 32-GByte maximum capacity for SDHC SD Card
 - 2-TByte maximum capacity for SDXC SD Card
 - 1-bit, 4-bit and 8-bit SD modes, CRC7 for command and CRC16 for data integrity
- Multiple boot options
 - SPI-NOR, NAND Flash, USB and eMMC
- Vector interrupt controller including VIC CPU-offload functionality
- SSI / SPI controller interfaces
 - Two SSI / SPI masters with up to eight device enables
 - One dedicated SSI / SPI slave port to connect to an external system master
- Two-wire serial Inter-Integrated Circuit (IDC) interfaces (x3)

- Configurable IDC buses (x2)
- One IDC bus dedicated for use with HDMI
- UART interface (x2)
 - DMA support
 - One interface supports flow control
- Infrared remote receiver interface which supports most popular protocols
- 114 General Purpose Input/Output (GPIO) pins with individual pull-up/down control
- ADC (four channels) with high/low threshold interrupt generation and 12-bit resolution
- Built-in power controller for power-up/down sequencing
- Real Time Clock (RTC)
- Interval timing with eight general-purpose timers configurable as external event counters
- Watchdog timer (one)
- Stepper motor interface (five channels) with four-channel Micro-Stepper interface
- Four Pulse Width Modulators (PWM)
- JTAG In-Circuit Emulator (ICE) interface for debugging (one)
- 404-pin TFBGA package (15 mm x 15 mm)
- 28-nm CMOS Low Power (LP) technology
- Operating temperature from -20 °C to +85 °C

1.5 Introduction: Memory Map

A12 memory allocation information is provided in this section as follows:

- [\(Section 1.5.1\) Memory Map: Cortex-A9 System Address](#)
- [\(Section 1.5.2\) Memory Map: AHB Address](#)
- [\(Section 1.5.3\) Memory Map: APB Address](#)
- [\(Section 1.5.4\) Memory Map: AXI Address](#)

1.5.1 Memory Map: Cortex-A9 System Address

Address	Description
0000.0000 – DFFF.FFFF	External DDR SDRAM
E000.0000 – E7FF.FFFF	AHB Mapped Peripherals
E800.0000 – E8FF.FFFF	APB Mapped Peripherals
F000.0000 – F002.FFFF	AXI Mapped Peripherals
FFFF.0000 – FFFF.1000	Boot Flash

Table 1-1. Cortex-A9 System Address Map.

Notes:

1. Minimum External DDR SDRAM recommended is 512 MB.

1.5.2 Memory Map: AHB Address

Memory Range	Sub-Range	Description	Size
0000.0000 - DFFD.FFFF		Main memory (DRAM access), subject to memory available on board.	Variable
DFFE.0000 - DFFE.FFFF		DRAM and DDR Controller Configuration. DRAM Controller address is DFFE.0800	64K
E000.0000 - E000.0FFF		Flash 4-KB DMA Data FIFO	4 KB
E000.1000 - E000.1FFF		Flash I/O Register Base (CPU Interface)	4 KB
	E000.1000 - E000.107F	Flash I/O Controller registers	128 Bytes
	E000.1080 - E000.10FF	FIFO DMA Controller registers	128 Bytes
	E000.1100 - E000.117F	Flash (NAND) Controller registers	128 Bytes
E000.2000 - E000.2FFF		SD0 Controller	4 KB
E000.3000 - E000.3FFF		VIC 1	4 KB
E000.4000 - E000.4FFF		DRAM Controller Configuration	4 KB
E000.5000 - E000.5FFF		DMA Controller	4 KB
E000.6000 - E000.7FFF		USB Device Controller	8 KB
E000.8000 - E000.AFFF		Reserved	
E000.B000 - E000.BFFF		AHB CPU ID (0xC)	4 KB
E000.C000 - E000.CFFF		SD1 Controller	4 KB
E000.D000 - E000.DFFF		Reserved	
E000.E000 - E000.EFFF		Ethernet Controller (MAC)	4 KB
E000.F000 - E000.FFFF		Ethernet Controller (DMA)	4 KB

Memory Range	Sub-Range	Description	Size
E001.0000 - E001.0FFF		VIC 2	4 KB
E001.1000 - E001.1FFF		Reserved	4 KB
E001.2000 - E001.2FFF		Flash I/O Subsystem (FIOS) DMA (FDMA)	4 KB
E001.3000 - E001.3FFF		HDMI Transmitter	4 KB
E001.4000 - E001.4FFF		Reserved	
E001.5000 - E001.5FFF		GDMA	4 KB
E001.6000 - E001.7FFF		Reserved	
E001.8000 - E001.8FFF		USB Host Controller (EHCI)	4 KB
E001.9000 - E001.9FFF		USB Host Controller (OHCI)	4 KB
E001.A000 - E001.AFFF		I2S Controller (I2S1)	4 KB
E001.B000 - E001.BFFF		Scratchpad	4 KB
E001.C000 - E001.CFFF		VIC 3	4 KB
E001.D000 - E001.DFFF		Secure	4 KB
E001.E000 - E001.EFFF		Reserved	
E001.F000 - E001.FFFF		SD2 Controller	4 KB
E002.0000 - E002.0FFF		SSI / SPI Master Controller (SSI0)	4 KB
E002.1000 - E002.1FFF		SSI / SPI Master Controller (SSI1)	4 KB
E002.2000 - E002.5FFF		Reserved	
E002.6000 - E002.6FFF		SSI / SPI Slave (SSIS0)	4 KB
E002.7000 - E003.0FFF		Reserved	
E003.1000 - E003.1FFF		NOR-SPI	4 KB
E003.2000 - E003.2FFF		UART1 (DMA + Flow Control)	4 KB
E003.3000 - E7FF.FFFF		Reserved	

Table 1-2. AHB Address Ranges.

1.5.3 Memory Map: APB Address

Memory Range	Description	Size
E800.1000 - E800.1FFF	IDC Controller 1 (IDC1) - For HDMI Use	4 KB
E800.2000 - E800.2FFF	Reserved	
E800.3000 - E800.3FFF	IDC Controller 0 (IDC0)	4 KB
E800.4000 - E800.4FFF	Stepper Motor Controller	4 KB
E800.5000 - E800.5FFF	UART 0	4 KB
E800.6000 - E800.6FFF	Infrared Remote	4 KB
E800.7000 - E800.7FFF	IDC Controller 2 (IDC2)	4 KB
E800.8000 - E800.8FFF	PWM	4 KB
E800.9000 - E800.9FFF	GPIO Controller 0	4 KB
E800.A000 - E800.AFFF	GPIO Controller 1	4 KB
E800.B000 - E800.BFFF	Timer	4 KB
E800.C000 - E800.CFFF	WDT	4 KB
E800.D000 - E800.DFFF	Reserved	
E800.E000 - E800.EFFF	GPIO Controller 2	4 KB
E800.F000 - E800.FFFF	Reserved	
E801.0000 - E801.0FFF	GPIO Controller 3	4 KB
E801.1000 - E801.2FFF	Reserved	
E801.3000 - E801.3FFF	Reserved	

Memory Range	Description	Size
E801.4000 - E801.4FFF	Reserved	
E801.5000 - E801.5FFF	Digital GPIO Pin Pull-Up/Down Control, PWC and RTC	4 KB
E801.6000 - E801.6FFF	I / O Multiplexing	4 KB
E801.D000 - E801.DFFF	ADC	
E801.8000 - E8FF.FFFF	Reserved	

Table 1-3. APB Address Ranges.

1.5.4 Memory Map: AXI Address

The Cortex-A9 AXI bus connects several ARM peripherals to the system. For detailed information on these peripherals, please refer to ARM core documentation.

Memory Range	Description	Size
F000.0000 - F000.00FF	Snoop Control Unit	256 Bytes
F000.0100 - F000.01FF	Reserved	256 Bytes
F000.0200 - F000.02FF	Reserved	256 Bytes
F000.0300 - F000.05FF	Reserved	
F000.0600 - F000.06FF	Reserved	256 Bytes
F000.0700 - F000.0FFF	Reserved	
F000.1000 - F000.1FFF	AXI Configuration	4 KB
F000.2000 - F000.2FFF	L2CC	4 KB
F000.3000 - F001.FFFF	Reserved	
F002.0000 - F002.7FFF	Cryptography Engine 0	32 KB
F002.8000 - F002.FFFF	Reserved	

Table 1-4. Cortex Peripheral Addresses.

2. SYSTEM BOOT AND CLOCK CONFIGURATION

2.1 System: Overview

This chapter provides information regarding the A12 system boot-up and configuration process. The chapter is organized as follows:

- [\(Section 2.2\) System: Boot and Clock Configuration](#)
- [\(Section 2.3\) System: Power-on Configuration](#)

2.2 System: Boot and Clock Configuration

The system boot-up and configuration process is managed by Ambarella's boot code. The A12 system software interacts with PLLs, PHYs and several other low-level hardware blocks using APB Reset, Clock and Test (RCT) registers with a system-layer application programming interface (API). This includes the setting of clock frequencies.

Internally, the A12 configures and optimizes system clocks and PLLs to meet operating mode performance requirements while saving as much power as possible. The peripheral clocks can be further optimized with the register programming information provided in this document. Please contact an Ambarella representative for additional information.

2.3 System: Power-on Configuration

This section provides power-on configuration information for the A12 family of SoCs. The section is organized as follows:

- [\(Section 2.3.1\) POC: Pin Assignment Table](#)
- [\(Section 2.3.2\) POC: Clock Configuration](#)
- [\(Section 2.3.3\) POC: Boot Mode Summary](#)
- [\(Section 2.3.4\) POC: Force USB Boot](#)
- [\(Section 2.3.5\) POC: NAND Boot](#)
- [\(Section 2.3.6\) POC: Boot ROM with SSI / SPI](#)
- [\(Section 2.3.7\) POC: eMMC Boot](#)
- [\(Section 2.3.8\) POC: Boot Bypass](#)
- [\(Section 2.3.9\) POC: Ethernet Selection](#)
- [\(Section 2.3.10\) POC: eFUSE ROM](#)

2.3.1 POC: Pin Assignment Table

The A12 chip uses 32-bit power-on configuration (POC) with strapping functions on external I2S and Transport Stream (TS) pins. After deassertion of the **POR_L** power-on reset pin, the chip generally latches the initial hardware settings on the pins. The tables in this section detail the power-on configuration bits POC[0:31] and corresponding strapping pins. Note that if POC[31] is set to 1, some (or all) of the configuration data may be obtained from eFUSE ROM.

Bit	PIN	Function	Setting
0	VDO_OUT[0]	Ethernet Select	Ethernet 0 - Disable 1 - Enable When Ethernet is enabled with power-on configuration bit POC[0] = 1 there are restrictions on programming GPIO service on the GPIO pins that correspond to unused Ethernet functions (Section 2.3.9).
1:3	VDO_OUT[3:1]	IDSP / Core / DDR Clock Configuration	Refer to Section 2.3.2 below.
4:5	VDO_OUT[5:4]	Boot Mode[1:0] Select	Select boot mode (SPI-NOR is the A12 default) 0x0 - SPI-NOR 0x1 - NAND 0x2 - eMMC 0x3 - SPI-EEPROM
6	VDO_OUT[6]	Ethernet PHY Interface Select	0 - MII 1 - RMII
7	VDO_OUT[7]	USB Clock Core 12 MHz	Set to 1 if using 12-MHz reference clock input for USB PHY
8	VDO_OUT[8]	Boot Bypass	0 - Disable Boot Bypass 1 - Enable Boot Bypass
9	VDO_OUT[9]		Reserved (Set to 0)
10	VDO_OUT[10]	Force USB Boot	Setting this pin to 1 will override the current boot mode setting.
11	VDO_OUT[11]	Clock Source Mode[0]	Select clock sources for Core and DDR clocks 0x0 - Normal Sources 0x1 - Reference Clock
12	VDO_OUT[12]	Clock Source Mode[1]	0x2 - Reserved 0x3 - Reserved
13	VDO_OUT[13]	Reference Clock is 48 MHz	Set to 1 if using 48-MHz reference clock input
14	VDO_OUT[14]	Boot Option Select	SPI-NOR Boot Mode: 0 - Use SC_[N] for boot pin 1 - Use SMIO_[N] for boot pin NAND Boot Mode: 0 - Disable Fast Flash Mode 1 - Enable Fast Flash Mode
15	VDO_OUT[15]	Boot Option Select	NAND Boot Mode 0 - Disable Spare Cell 2X Mode 1 - Enable Spare Cell 2X Mode

Bit	PIN	Function	Setting
16	I2S_SO	Boot Option Select	NAND Boot Mode: 0 - Disable BCH error correction (ECC) 1 - Enable BCH error correction (ECC) eMMC Boot Mode: 0 - Disable 8-bit Bus Width Mode 1 - Enable 8-bit Bus Width Mode
17	VDO_HSYNC	Boot Option Select	NAND Boot Mode: 0 - Disable Read Confirmation Mode 1 - Enable Read Confirmation Mode eMMC Boot Mode: 0 - Disable 4-bit Bus Width Mode 1 - Enable 4-bit Bus Width Mode
18	ENET_TXD_0	Boot Option Select - NAND Flash Page Size Mode	NAND Boot Mode: 0 - 512 -bytes Flash Page Size Mode 1 - 2048-bytes (2 KB) Flash Page Size Mode
19	ENET_TXD_1	Boot Option Select - NAND 2x2KB Page Mode	NAND Boot Mode: 0 - Disable 2x2KB Page Mode 1 - Enable 2x2KB Page Mode When active, the Flash controller uses two 2-KB memory blocks to store boot code, rather than one. Note that bit[19] is only valid when bit[18] is set to 1.
20:27			Reserved (NC) No mapping to external pins
28	VDO_CLK	Select Method for USB0 Host/Device Configuration	0 - Use IDDIG0 signal from USB0 PHY to configure it as a host or device 1 - Mask the IDDIG0 signal from USB0 PHY and use bit[29] for USB type selection Note that bit[28] is ignored when bit[10] is set to 1
29	VDO_VSYNC	Configure USB0 as a Host or Device	0 - Host 1 - Device Valid only when bit[28] is set to 1 Note that bit[29] is ignored when bit[10] is set to 1
30			Reserved (NC) No mapping to external pin
31	VDO_HVLD	System Configuration Data Source	0 - Configuration data is set by pins 1 - Configuration data is read from on-chip eFUSE ROM (except bit[31])

Table 2-1. Power-On Configuration Pin Assignments.

2.3.2 POC: Clock Configuration

The POC[3:1] bits (**VDO_OUT[3:1]**) allow the user (with a 24-MHz system clock) to select from the pre-configured PLL settings in the table below. These frequencies are enabled during the power-on process only, and the settings may be overridden by system software for operation.

POC[3:1] (VDO_OUT[3:1])	VDSP (gclk_core)	IDSP (gclk_idsp)	DRAM (gclk_dram)	Cortex (gclk_cortex)
000	312	408	600	816
001	252	348	396	720
010	240	336	528	720
011	144	144	396	672
100	432	408	600	792
101	312	432	336	504
110	348	408	600	1008
111	216	216	216	216

Table 2-2. Power-On Clock Rates (MHz).

2.3.3 POC: Boot Mode Summary

Several bits of the power-on configuration pins are decoded to determine the A12 boot mode. Table 2-3 below shows the available boot-mode settings, including which pins must be controlled (with pull-up/pull-down) and which can be left unconnected (floating). Table 2-4 outlines the A12 shared boot options.

POC Bit	POC descrip-tion	Pin	SPI-NOR boot	NAND boot	MMC boot	SPI boot	USB boot	Boot By-pass
0	enet_sel	VDO_OUT_0	0	0	0	0	0	0
3:1	clk_freq	VDO_OUT_3:1	0/1	0/1	0/1	0/1	0/1	0/1
4	boot_mode[0]	VDO_OUT_4	0	1	0	1	X	X
5	boot_mode[1]	VDO_OUT_5	0	0	1	1	X	X
6	enet_phy_intf_sel	VDO_OUT_6	0	0	0	0	0	0
7	usbp_clkcore_12Mhz	VDO_OUT_7	X	X	X	X	0/1	X
8	boot_bypass	VDO_OUT_8	0	0	0	0	0	1
9	freq_ramp	VDO_OUT_9	0	0	0	0	0	0
10	force_usb_boot	VDO_OUT_10	0	0	0	0	1	0
11	clk_mode[0]	VDO_OUT_11	0/1	0/1	0/1	0/1	0/1	0/1
12	clk_mode[1]	VDO_OUT_12	0/1	0/1	0/1	0/1	0/1	0/1
13	refclk_is_48Mhz	VDO_OUT_13	0/1	0/1	0/1	0/1	0/1	0/1
14	boot_option[0]	VDO_OUT_14	0/1	0/1	X	X	X	X
15	boot_option[1]	VDO_OUT_15	0/1	0/1	X	X	X	X
16	boot_option[2]	I2S_SO	0/1	0/1	0/1	X	X	X
17	boot_option[3]	VDO_HSYNC	0/1	0/1	0/1	0/1	X	X
18	boot_option[4]	ENET_TXD_0	0/1	0/1	X	0/1	X	X
19	boot_option[5]	ENET_TXD_1	X	0/1	X	X	X	X
27:20								

POC Bit	POC description	Pin	SPI-NOR boot	NAND boot	MMC boot	SPI boot	USB boot	Boot By-pass
28	usb0_mask_id-dig0	VDO_CLK	X	X	X	X	0/1	X
29	usb0_type_sel	VDO_VSYNC	X	X	X	X	0/1	X
30								
31	sysconfig_src	VDO_HVLD	0/1	0/1	0/1	0/1	0/1	0/1

Table 2-3. Power-On Boot Mode.

Notes:

1. Switching between NAND / SPI-NOR / eMMC and USB boot mode requires careful selection of POC bits 4, 5, and 10.
2. For MMC boot, use eMMC NAND with a specification higher than or equal to Version 4.4, which supports boot function. A generic SD card or an eMMC 4.1 NAND should not be used.
3. The A12 chip includes three SD controllers. On hardware connection, please use SD0 pins to connect to eMMC (4.4 or later) NAND. SD1 and SD2 pins should not be used.

Boot Option	SPI-NOR	NAND	eMMC	SPI	Force USB
boot_option[0]	Boot pin selection: 0: Use SC_[N] for NOR-SPI 1: Use SMIO_[N] for NOR-SPI	Flash Fast Boot	DDR Mode Boot		
boot_option[1]	LSB First	NAND Spare Cell 2x	High Speed Boot		
boot_option[2]	SPI Mode Selection	NAND ECC BCH Enable	8-bit Bus Width		
boot_option[3]	Use Alternative Clock	NAND Read Confirm	4-bit Bus Width	Byte Address[0]	
boot_option[4]	Use 2-Bytes Address	NAND Page Size		Byte Address[1]	
boot_option[5]		2x2K Page Boot Mode			

Table 2-4. Shared Boot Options Matrix

2.3.4 POC: Force USB Boot

The USB boot option is enabled using POC bit 10. Setting this pin to high will override the current boot mode setting and force the chip into USB boot mode. Use POC[7] for USB PHY clock settings. The USB PHY reference clock can be either 12- or 24-MHz.

2.3.5 POC: NAND Boot

The NAND Flash boot option is enabled and implemented using POC bits 4, 5, 14, 15, 16, 17, 18, and 19 as described above in [Section 2.3.3](#).

- Enable the NAND Flash boot option using POC bits 4 and 5.
- POC[18] enables NAND Flash page size of 512 bytes or 2 KB.
- When the NAND Flash page size is set to 2 KB using POC[18], POC bit 19 can be enabled. When POC[19] is set to 1, the Flash controller uses two 2-KB memory blocks to store boot code, rather than one.
- Use the NAND Read Confirm option (POC[17]) for large-block read operations. Read confirm should be enabled when the NAND Flash size is 1 Gb or greater. Large-block NAND Flash devices simplifies accessing the NAND Flash array for read operations. A page read command consists of a single command (0h) followed by four address cycles for a 1-Gb device. A read confirm command (30h) is issued regardless of the area accessed in the NAND Flash array.
- Enabling Flash Fastboot with POC[14] allows direct modification of the Flash file system by a host (e.g., computer) over a USB connection.
- Use POC[16] to enable the Bose-Chaudhuri-Hocquenghem (BCH) method for low-power error correction (ECC).
- POC[15] provides spare area DMA capability. Use the spare cell 2x feature for 512-byte block NAND with 32-byte spare area or 2048-byte block NAND with 128-byte spare area.

2.3.6 POC: Boot ROM with SSI / SPI

The default boot mode for the A12 chip is the SPI-NOR boot option. There is also an option to boot from EEPROM memory. SSI / SPI boot options are enabled and configured using POC bits 4, 5, and 14 as described above in [Section 2.3.3](#).

2.3.7 POC: eMMC Boot

The SD0 controller provides 1-, 4- and 8-bit eMMC boot with normal, high-speed and DDR-mode operations. eMMC boot is enabled with POC bits 13, 16, 17, 18 and 19 and disabling other boot modes as described in [Section 2.3.3](#).

- Use POC bits 4 and 5 to enable eMMC boot mode.
- Select bus width:
 - POC[17] SD4 boot, POC[16] SD8 boot = 0, 0 (for 1-bit boot)
 - POC[17] SD4 boot, POC[16] SD8 boot = 1, 0 (for 4-bit boot)
 - POC[17] SD4 boot, POC[16] SD8 boot = 0, 1 (for 8-bit boot)
- High-speed mode selection
 - POC[15] HS boot = 0 (Host Controller outputs CMD line and DAT lines at the falling edge of the SD clock (up to 25 MHz))

- POC[15] HS boot = 1 (Host Controller outputs CMD line and DAT lines at the rising edge of the SD clock (up to 50 MHz))
- DDR mode selection
 - POC[14] DDR boot = 0 (data sample on single edges of clock)
 - POC[14] DDR boot = 1 (data sample on both edges of clock)

2.3.8 POC: Boot Bypass

The Boot Bypass option is enabled using POC[8] as described above in [Section 2.3.3](#). Boot bypass is used when the boot process from media (NAND, USB, SPI etc.) will be bypassed. This option may be selected to allow a system debug process to run, for example.

2.3.9 POC: Ethernet Selection

Ethernet function is enabled / disabled with power-on configuration bit POC[0]. POC bit 6 is used to select between the MII or RMII Ethernet standards.

2.3.10 POC: eFUSE ROM

The A12 chip allows users to select which power-on configuration bits are loaded from eFUSE ROM or from strapping pins.

The A12 eFUSE is 64 bits; i.e., the lower 32 bits store system configuration data and the upper 32 bits handle selection.

If strapping pin 31 (**VDO_HVLD**) is pulled down, all configuration values are loaded from the pins. If it is pulled up, all power-on configuration bits are loaded from eFUSE with the exception of POC bit 10, which is always loaded from POC pins ([Table 2-1](#)).

The contents of the eFUSE ROM module are set at the factory and are not user-accessible.

3. DSP SUBSYSTEM

This chapter provides the following information on the A12 DSP Subsystem:

- [\(Section 3.1\) DSP: System Overview](#)
- [\(Section 3.2\) DSP: Image Processing](#)
- [\(Section 3.3\) DSP: Control of Video Input \(VIN\) and Output \(VOUT\)](#)

3.1 DSP: System Overview

The DSP Subsystem interfaces with the Video Input (VIN) module, the Video Output (VOUT) module and the DRAM controller as illustrated in the diagram below. The subsystem runs DSP microcode loaded by the Cortex-A9, and performs tasks per command control from the Cortex-A9 such as video processing, image processing, video post-processing (for output), and video encode / decode.

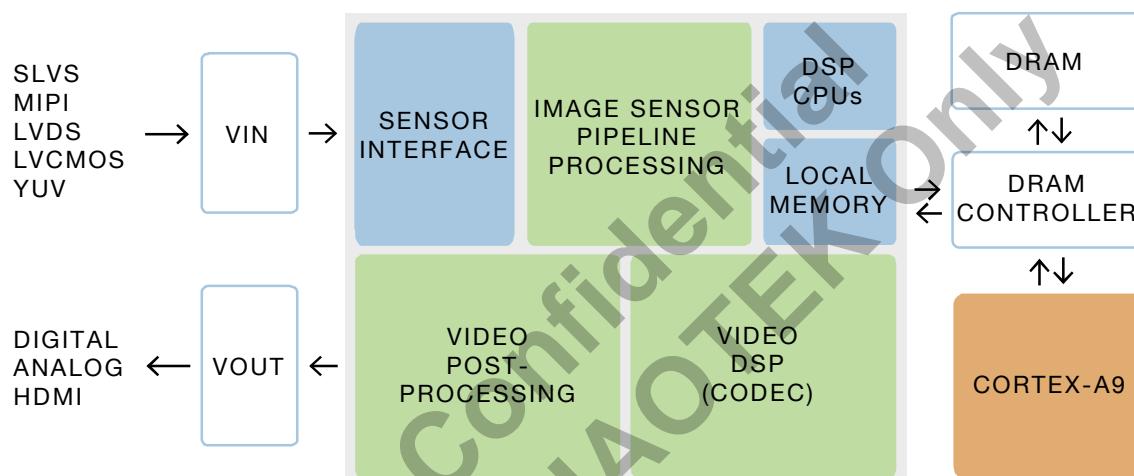


Figure 3-1. The A12 DSP Subsystem and its Relationship with System Modules.

Note that there is no direct hardware access to the DSP Subsystem from memory-mapped registers. The Cortex-A9 uses a dedicated DRAM memory location to send commands to or receive status updates from the DSP Subsystem.

3.2 DSP: Image Processing

The diagram below schematically illustrates DSP image processing on the A12 chip:

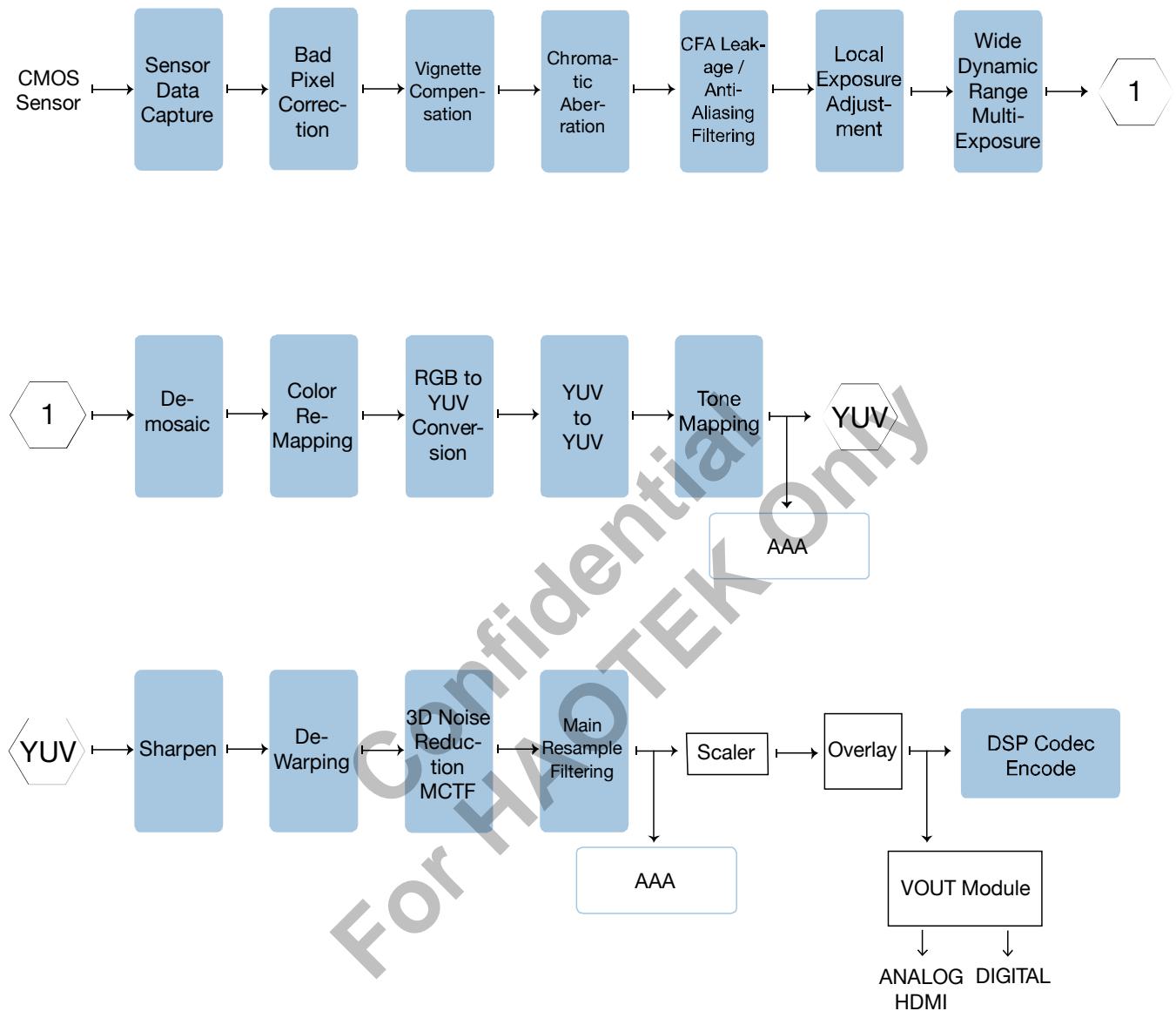


Figure 3-2. A12 DSP Image Processing Pipeline.

The A12 DSP Subsystem accepts data from A12 sensor interfaces as well as YUV data from external sources (such as an HDMI transmitter). Image processing in the CFA domain includes bad pixel and vignetting corrections, chromatic aberration removal, noise and leakage filtering as well as local exposure adjustment. In the RGB domain, demosaicing, color remapping, and color space transformations are performed.

YUV domain processes include sharpening, dewarping, chroma noise filtering, 3D motion compensated temporal filtering (MCTF) and resampling. Auto-Exposure, Auto-Focus, and Auto-White Balance (AAA) statistics are sampled from both the RGB and YUV domains.

3.3 DSP: Control of Video Input (VIN) and Output (VOUT)

The DSP Subsystem handles all register programming for the Video Input and Output modules (VIN and VOUT). The register programming details provided in this manual are intended for reference and debugging. Refer to the chip A12 datasheet for interface pins and settings.

For Confidential
For HAOTEK Only

4. DMA ENGINE SUBSYSTEM

This chapter provides programming information for the A12 Direct Memory Access (DMA) Engine Subsystem. The chapter is organized as follows:

- [\(Section 4.1\) DMA: Features](#)
- [\(Section 4.2\) DMA: Engines and Channel Assignment](#)
- [\(Section 4.3\) DMA: Operation Overview](#)
- [\(Section 4.4\) DMA: Flash I/O Dual Space Mode](#)
- [\(Section 4.5\) DMA: Operation Detail](#)
- [\(Section 4.6\) DMA: Transaction Non-Descriptor Mode](#)
- [\(Section 4.7\) DMA: Transaction Descriptor Mode](#)
- [\(Section 4.8\) DMA: Descriptor Format](#)
- [\(Section 4.9\) DMA: Programming DMA and FDMA Registers](#)
- [\(Section 4.10\) DMA: Registers](#)
- [\(Section 4.11\) DMA: Errors](#)
- [\(Section 4.12\) DMA: Interrupts](#)

For the purposes of this chapter, the terms DMA transaction and DMA operation are used interchangeably.

4.1 DMA: Features

The DMA Engine Subsystem provides Direct Memory Access (DMA) functionality for AMBA peripherals. It moves blocks of data between the main memory and AHB-bus peripheral devices, as well as between areas within the main memory itself. Features of the DMA Engine Subsystem include:

- Eight DMA channels
 - Six channels for SSI/SPI
 - Two channels for the I2S controller
- One FDMA channel
- Memory-to-memory transfers including support for transfers between memory and peripherals with DMA between:
 - AHB bus devices and RAM memory
 - Two memory areas in RAM memory
- Physical DMA
- Programmable address, transaction size (8 to 1024 bytes), and transfer size (1, 2, 4, or 8 bytes)
- Programmable transfer count (up to 4 MB)
- Memory interface transaction size optimized for RAM burst operations

- Direct software writes or descriptor information from memory to program DMA control
- DMA scatter/gather via chained descriptor list in memory with DMA control information source
- Host processor initiates the DMA operation
- Interrupts signal the end of DMA operations
- Recognition and logging of error types that end DMA operations
- Sharing of the AHB bus master interface (and the memory interface) with round-robin arbitration
- Fully pipelined for highest throughput
- AHB bus slave interface for register reads and writes

4.2 DMA: Engines and Channel Assignment

The A12 DMA subsystem provides a single DMA Engine and eight channels. The FDMA engine (one channel) is located at base address 0xE001.2000.

DMA Channel Number	DMA Engine Address	Description
0	0xE001.5000	SSI0 and NOR-SPI Tx Request
1		SSI0 and NOR-SPI Rx Request
2		SSI1 Tx Acknowledge
3		SSI1 Rx Acknowledge
4		SSIS0 and UART Tx Acknowledge
5		SSIS0 and UART Rx Acknowledge
6		Primary I2S Rx
7		Primary I2S Tx

Table 4-1. Current Assignment of DMA Engine Channels.

Note:

- For DMA channel multi-function detail, please refer to [Section 14.3.2.4](#).

4.3 DMA: Operation Overview

The DMA Engine Subsystem supports block transfers of sizes up to 4 MB. The exact transfer count of each DMA operation is set by software. DMA operations are performed as a series of main-memory and AHB-bus transactions to move the data block. The length of each AHB bus transaction is software-programmable so that it can be optimized according to system characteristics. Each individual AHB bus transaction is from 8 bytes to 1024 bytes according to a value programmed into a DMA Channel Control register (**DMA_ch[n].control**). Additionally, the data size of each data transfer on the AHB bus is software programmable to be 1, 2, 4 or 8 bytes.

The source and destination starting addresses are set by software. These addresses are incremented by the DMA Engine Subsystem to form the request addresses at the memory- and AHB-bus interfaces of the DMA Engine Subsystem. All are physical addresses.

The DMA control information determined by software (e.g., source and destination starting addresses, transfer count, bus transaction size, data transfer size, and bus address increment mode) can be set in two ways. The first is by direct software writes to the DMA Engine Subsystem registers with DMA control information. The second is through descriptors in memory that hold the DMA control information and are initialized by software. To support DMA configuration from descriptors, the DMA Engine Subsystem contains logic to read the descriptors from tables in memory and load the appropriate registers with the DMA control information.

A DMA operation begins when a DMA channel is enabled and after its source and destination starting addresses, transfer count, bus transaction size, data transfer size, and bus address increment mode are configured. The DMA Engine Subsystem moves the data block, and the DMA operation ends naturally when the number of bytes specified by the transfer count have been moved. A DMA operation may also end early due to an error.

When a DMA channel is disabled by hardware for any reason, an interrupt informs the host processor. Each DMA channel has a register that identifies the cause of the interrupt. The cause can be an event such as a natural DMA operation end, a natural end of a descriptor chain, or any among several types of errors.

DMA channels share a single AHB bus master at the DMA Engine Subsystem bus interface. The subsystem has a single interface to memory that is shared by all DMA channels. It uses round-robin arbitration to choose the DMA channel whose request is serviced on the bus and memory interfaces.

The DMA Engine Subsystem contains several registers. These registers are accessed using single transactions initiated by host processor reads and writes. There are registers that hold DMA channel control and status information, DMA channel source and destination starting addresses, DMA channel descriptor addresses, and DMA Engine Subsystem control and status information.

For more detailed information, refer to [Section 4.5](#).

4.4 DMA: Flash I/O Dual Space Mode

The DMA Engine Subsystem provides a main area and spare area for DMA operations. During normal operations, the main and spare areas are read and written one-at-a-time. Additionally the A12 FDMA system enables dual-space mode operation for Flash I/O. With the Flash I/O DMA dual-space mode, the data specified by the main control register and spare control register are interleaved with the step-size specified using `DMA_ch[n].dsm_control` register bits `main_jp_size` and `spare_jp_size`.

4.5 DMA: Operation Detail

As described previously, DMA transactions move a block of data from a source data area to a destination data area. The data is transferred via a series of reads from the source data area and writes to the destination data area. When performing these reads and writes, the DMA Engine Subsystem begins by reading data at the lowest source location and writing this data to the lowest destination location. It then increments the source and destination addresses for the next data transfer. Source and destination addresses are continually incremented for all subsequent data transfers until the entire data block has been moved or an error occurs. Either of these events ends the DMA transaction, and the DMA Engine Subsystem informs the host processor that the DMA transaction has ended with an interrupt.

The DMA operation can be summarized by the following steps:

1. The DMA channel is set up.
2. The DMA channel is enabled to transfer data. This starts the DMA transaction.
3. A series of reads and writes are performed to transfer the data from its source to the destination.
4. Data transfers end when the entire block of data has been moved or an error occurs.
5. DMA channel data transfer is automatically disabled by hardware.
6. Completion status for the DMA transaction is logged in the Channel Status register (**DMA_ch[n].status**).
7. Completion status for the spare-area DMA transaction is logged in the Spare Area Channel Status register (**DMA_ch[n].spare_status**).

The DMA channel must be configured prior to the commencement of a DMA operation. Each DMA channel has three types of registers that hold channel configuration information for the DMA operation:

1. Channel Control (**DMA_ch[n].control**)
2. Channel Source Address (**DMA_ch[n].source_addr**)
3. Channel Destination Address (**DMA_ch[n].dest_addr**)

The starting address for the reads from the source data area is written to the Channel Source Address register. The Channel Destination Address register is written with the starting address of the writes to the destination data area. The Channel Control register is written to indicate the number of bytes to be moved, the AHB bus transaction size, the AHB bus data transfer size, the AHB bus address increment mode, and whether the source and destination data areas are in memory or in AHB bus peripheral devices.

Up to 4 MB can be moved by a DMA operation. The AHB bus transaction size can be set from 8 bytes to 1024 bytes. Data transfers on the AHB bus can be one, two, four or eight bytes. DMA operations can be memory-to-memory, AHB-bus-to-memory, or memory-to-AHB-bus. Moves between AHB bus devices are not supported. All of this information is programmed into the Channel Control register. The register map and set details for the DMA Engine Subsystem are provided in [Section 4.9](#).

Follow the steps below to initiate and configure a DMA transaction:

1. Write the **Channel Source Address** and **Channel Destination Address** registers.
2. Write the Spare Area registers **Channel DMA Transaction Count**, **Channel Source Address**, **Channel Destination Address**, and **Channel Dual-Space Mode Control** if dual-space mode should be enabled.
3. Write the **Channel Control** register with the remaining configuration information.
4. Set the data transfer enable flag to 1.

Once enabled, the DMA channel issues read requests to the data source and write requests to the data destination. Upon receiving read data, the DMA channel temporarily buffers it before writing it to the destination. This buffering accumulates enough data to issue requests to a DRAM controller, which will treat these requests as burst requests.

Data reads and writes continue until the number of bytes shown in the count field of the Channel Control register, have been moved or until an error is reported. When either of these events occurs, the DMA transaction ends. [Section 4.11](#) details the DMA error descriptions.

When the DMA transaction ends, DMA channel data transfer is automatically disabled. This occurs when the hardware resets the data transfer enable flag in the Channel Control register to 0 after the last data has been written to the destination. The cause of the DMA transaction completion is logged in the Channel Status register. The Channel Status register also shows the number of bytes that were transferred. Note that when an error ends the DMA transaction, the number of bytes shown may differ from the number of bytes that were intended to be transferred (i.e., the value programmed into the count field of the Channel Control register). If dual-space mode is enabled, the status of spare-area DMA transactions is logged in the Spare Area Channel Status register.

4.6 DMA: Transaction Non-Descriptor Mode

The host processor sets up each DMA transaction in non-descriptor mode. A DMA channel is in non-descriptor mode when the Channel Control register descriptor mode bit is 0.

Before a DMA operation, software running on the host processor selects one of the DMA channels for the DMA operation. For details on Channel Assignment see [Section 4.2](#). After selecting the channel, the software then configures the chosen channel by directly writing to the three registers that hold channel transaction-configuration information:

1. Channel Control register (**DMA_ch[n].control**)
2. Channel Source Address (**DMA_ch[n].source_addr**)
3. Channel Destination Address (**DMA_ch[n].dest_addr**)

Note that the DMA channel must be disabled (Channel Control register-enable bit is 0) before it can be configured. Refer to the DMA registers in [Section 4.9](#) for details.

The DMA transaction begins when the enable bit is written to 1 by the host processor, as the data transfer enable flag is the Channel Control register-enable bit in non-descriptor mode. Typically, the host processor writes the Channel Source Address and Channel Destination Address registers first. Then the host processor writes the Channel Control register with the remainder of the configuration information and sets the enable bit to 1.

When data transfer ends (either naturally, when the number of set bytes in the Channel Control register count field have been moved, or when an error occurs) the DMA channel generates an interrupt. This interrupt informs the host processor that the DMA transaction has completed.

The host processor recognizes this interrupt, and branches to a routine to handle it. The DMA channel signaling the interrupt is identified by reading the DMA Engine Interrupt register. The host processor can also read the Channel Status register to determine how the DMA transaction ended.

4.7 DMA: Transaction Descriptor Mode

In descriptor mode, the host processor sets up several transactions in DMA descriptors located in memory. Each DMA descriptor contains:

- The DMA transaction configuration information described above, for one DMA transaction.
- A pointer to a memory location where the DMA transaction completion status will be written once the DMA transaction ends.
- A pointer to the descriptor for the next DMA transaction.

When the host processor enables the DMA channel by setting the Channel Control register-enable bit to 1, the DMA Engine Subsystem processes the DMA descriptors and DMA transactions occur for all descriptors without additional host processor intervention. A DMA channel is in descriptor mode when the Channel Control register descriptor mode bit is 1.

Prior to the execution of a DMA operation, software running on the host processor selects one of the DMA channels for the DMA operation. Subsequently, it writes descriptors to memory for several DMA transactions, and loads the Channel Descriptor Address (**DMA_ch[n].descriptor**) register with the address of the descriptor for the first transaction.

The DMA channel must be disabled (set the Channel Control register-enable bit to 0) when setting up the descriptors for the first DMA transactions. When extending the DMA descriptor chain in memory, the DMA channel can be either enabled or disabled.

DMA descriptor processing begins when the host processor enables the DMA channel by writing the Channel Control register-enable bit to 1. When the DMA channel is enabled, the descriptor-processing logic fetches the descriptor for the first DMA transaction from memory at the location shown in the DMA Channel Descriptor Address register.

Using the contents of the fetched descriptor, the descriptor-processing logic loads the DMA channel transaction information registers, saves the DMA transaction status report address, and saves the next descriptor address. To do this, the descriptor-processing logic writes the following DMA channel registers:

- Channel Control (**DMA_ch[n].control**)
- Channel Source Address (**DMA_ch[n].source_addr**)
- Channel Destination Address (**DMA_ch[n].dest_addr**)
- Channel Descriptor Address (**DMA_ch[n].desc_addr**)

In descriptor mode, the data transfer enable flag is maintained by the descriptor-processing logic. After the DMA channel transaction information registers are loaded by the descriptor-processing logic, the descriptor-processing logic sets the data transfer enable flag to 1, and the transaction begins.

The data block is moved from its source to the destination. When the data transfer ends (either naturally, when the number of bytes shown in the count field of the DMA descriptor have been moved, or when an error occurs) the data transfer enable flag is reset to 0 by the descriptor-processing logic. The descriptor-processing logic then writes the transaction completion status found in the Channel Status register to memory at the report address location shown in the descriptor.

If the DMA descriptor end-of-chain bit is not set to 1, the descriptor-processing logic then fetches the next descriptor from memory, loads the channel transaction information registers, and starts the next transaction data transfer. This process continues until a DMA descriptor with an end-of-chain bit set to 1 is encountered, or until an error is encountered for cases in which the Channel Control register stop-on-error bit is set to 1.

After the transaction described by a descriptor with its end-of-chain bit set to 1, or when an error is encountered and the Channel Control register stop-on-error bit is 1, the descriptor processing concludes and the DMA channel is automatically disabled. The channel then generates an interrupt, which informs the host processor that the transactions have been completed.

The host processor recognizes this interrupt, and branches to a routine to handle it. The DMA channel which signaled the interrupt is identified by reading the DMA Engine Interrupt register.

4.8 DMA: Descriptor Format

Descriptor fields, which are read-only, are described in detail in [Section 4.5](#) above. If dual-space mode is enabled, additional descriptors for the spare-area DMA transactions are required. The layouts for the traditional and spare descriptors are similar, except Spare Area DMA transactions do not require the Control Information field.

Memory Address	Descriptor	Comments
Descriptor address	Source starting address [31:0]	
Descriptor address + 4	Destination starting address [31:0]	
Descriptor address + 8	Next descriptor address [31:0]	
Descriptor address + 12	Report address [31:0]	
Descriptor address + 16	Transfer Count	Refer to Table 4-3 .
Descriptor address + 20	Control Information	Not used for spare-area DMA transactions Refer to Table 4-1 .

Table 4-2. A DMA Descriptor is Composed of Six 32-bit Words.

Bits	Name	Attr	Reset	Description
31:22				Reserved
21:0	transfer_count	R		Number of bytes to transfer See DMA_ch[n].control description. The size of each descriptor memory block must be a multiple of eight bytes. The transfer count in each descriptor must also be a multiple of eight.

Table 4-3. DMA Descriptor Transfer Count.

Bits	Name	Attr	Reset	Description
31:25				Reserved
24	eoc	R		End of chain flag DMA will stop after this descriptor. Note that if the eoc bit is set, the DMA channel will generate an interrupt when this descriptor's DMA operation ends without error.
23	wm	R		Write memory See DMA_ch[n].control description.
22	rm	R		Read memory See DMA_ch[n].control description.

Bits	Name	Attr	Reset	Description
21	ni	R		No bus address increment flag 1 - A bus starting address written to DMA_ch[n].source or DMA_ch[n].destination will not be incremented during the DMA operation. Therefore, addresses on the AHB bus will only increment up to the amount shown in the bus transaction block size field of this DMA_ch[n].control and then return to the value written to DMA_ch[n].source or DMA_ch[n].destination to begin incrementing again when this bit is 1. 0 - The AHB bus address is incremented continually throughout the DMA operation. (This no-increment mode affects only peripheral addressing, but not main memory addressing, mainly used for peripherals that have a FIFO-type DMA interface such as the I2S unit.)
20:19	ts	R		Transfer size See DMA_ch[n].control description.
18:16	blk	R		Bus transaction block size See DMA_ch[n].control description.
15:3				Reserved
2	id	R		Interrupt on done 1 indicates that when this descriptor's DMA operation ends without an error, the DMA channel will signal an interrupt. No interrupt will be signaled when this bit is 0.
1	ie	R		Interrupt on error When the DMA channel continues running upon an error, 1 indicates that the DMA channel will signal an interrupt when an error occurs. No interrupt will be signaled as long as the DMA channel continues to run when this bit is 0. An interrupt is always signaled when the DMA channel stops. Therefore, when the DMA channel stops on errors in descriptor mode this bit is ignored.
0	st	R		Stop on error 1 indicates that the DMA channel will be disabled when an error occurs. The DMA channel will continue running when an error occurs in descriptor mode if this bit is 0.

Table 4-4. DMA Descriptor Control Information.

4.9 DMA: Programming DMA and FDMA Registers

This section lists the registers used for the DMA and FDMA engines.

- [\(Section 4.9.1\) Programming: DMA Engine Registers](#)
- [\(Section 4.9.2\) Programming: FDMA Engine Registers](#)

4.9.1 Programming: DMA Engine Registers

The DMA Engine Subsystem contains registers that hold DMA addresses, control information and results. Each DMA channel is associated with one register that contains:

- Channel control information (**DMA_ch[n].control**)
- Source address (**DMA_ch[n].source_addr**)
- Destination address (**DMA_ch[n].dest_addr**)
- Descriptor address (**DMA_ch[n].desc_addr**)
- Channel results including interrupts and transfer count (**DMA_ch[n].status**)

These registers are used by each DMA channel to control its DMA transactions, and to report completion status for these transactions. Additionally the DMA subsystem provides a single global register (**DMA_intr_sts**) that identifies DMA channels with outstanding interrupts.

These registers are accessed with single AHB bus transactions that have a data size of 4 bytes. During an AHB bus transaction, AMBA Address[9:0] is the register index that indicates the register being accessed

4.9.2 Programming: FDMA Engine Registers

The FDMA engine utilizes all registers described for the DMA engine ([Section 4.9.1](#)) for main-area operations. It additionally provides the following registers for spare-area operation and controls for use with dual-space mode:

- Spare-area DMA transaction count (**DMA_ch[n].spare_count**)
- Spare-area source address (**DMA_ch[n].spare_source_addr**)
- Spare-area destination address (**DMA_ch[n].spare_dest_addr**)
- Spare-area channel results including interrupts and transfer count (**DMA_ch[n].spare_status**)
- Registers to control dual-space mode operations for each channel (**DMA_ch[n].dsm_control**)

4.10 DMA: Registers

4.10.1 DMA Registers: Map

Register Offset	Register Name	Description
0x200	DMA_ch0_spare_count	Channel 0 Spare Area Count
0x204	DMA_ch0_spare_source_addr	Channel 0 Spare Area Source Address
0x208	DMA_ch0_spare_dest_addr	Channel 0 Spare Area Destination Address
0x20C	DMA_ch0_spare_status	Channel 0 Spare Area Status
0x210 - 0x27C		Reserved
0x280	DMA_ch0_spare_desc_addr	Channel 0 Spare Area Descriptor Address
0x284 - 0x2FC		Reserved
0x300	DMA_ch0_control	Channel 0 Control
0x304	DMA_ch0_source_addr	Channel 0 Main Source Address
0x308	DMA_ch0_dest_addr	Channel 0 Main Destination Address
0x30C	DMA_ch0_status	Channel 0 Main Status
0x310	DMA_ch1_control	Channel 1 Control
0x314	DMA_ch1_source_addr	Channel 1 Main Source Address
0x318	DMA_ch1_dest_addr	Channel 1 Main Destination Address
0x31C	DMA_ch1_status	Channel 1 Main Status
0x320	DMA_ch2_control	Channel 2 Control
0x324	DMA_ch2_source_addr	Channel 2 Main Source Address
0x328	DMA_ch2_dest_addr	Channel 2 Main Destination Address
0x32C	DMA_ch2_status	Channel 2 Main Status
0x330	DMA_ch3_control	Channel 3 Enable and Control
0x334	DMA_ch3_source_addr	Channel 3 Main Source Address
0x338	DMA_ch3_dest_addr	Channel 3 Main Destination Address
0x33C	DMA_ch3_status	Channel 3 Main Status
0x340	DMA_ch4_control	Channel 4 Control
0x344	DMA_ch4_source_addr	Channel 4 Main Source Address
0x348	DMA_ch4_dest_addr	Channel 4 Main Destination Address
0x34C	DMA_ch4_status	Channel 4 Main Status
0x350	DMA_ch5_control	Channel 5 Control
0x354	DMA_ch5_source_addr	Channel 5 Main Source Address
0x358	DMA_ch5_dest_addr	Channel 5 Main Destination Address
0x35C	DMA_ch5_status	Channel 5 Main Status
0x360	DMA_ch6_control	Channel 6 Control
0x364	DMA_ch6_source_addr	Channel 6 Main Source Address
0x368	DMA_ch6_dest_addr	Channel 6 Main Destination Address
0x36C	DMA_ch6_status	Channel 6 Main Status
0x370	DMA_ch7_control	Channel 7 Control
0x374	DMA_ch7_source_addr	Channel 7 Main Source Address
0x378	DMA_ch7_dest_addr	Channel 7 Main Destination Address
0x37C	DMA_ch7_status	Channel 7 Main Status
0x380	DMA_ch0_desc_addr	Channel 0 Main Descriptor Address
0x384	DMA_ch1_desc_addr	Channel 1 Main Descriptor Address

Register Offset	Register Name	Description
0x388	DMA_ch2_desc_addr	Channel 2 Main Descriptor Address
0x38C	DMA_ch3_desc_addr	Channel 3 Main Descriptor Address
0x390	DMA_ch4_desc_addr	Channel 4 Main Descriptor Address
0x394	DMA_ch5_desc_addr	Channel 5 Main Descriptor Address
0x398	DMA_ch6_desc_addr	Channel 6 Main Descriptor Address
0x39C	DMA_ch7_desc_addr	Channel 7 Main Descriptor Address
0x3A0	DMA_ch0_dsm_control	Channel 0 Dual-Space Mode Control
0x3A4 - 0x3EC		Reserved
0x3F0	DMA_intr_sts	Global Interrupt Register

Table 4-5. DMA Engine Subsystem Registers.

4.10.2 DMA Registers: Details

4.10.2.1 DMA_ch[n].spare_count Register

Bits	Name	Attr	Reset	Description
31:22				Reserved
21:0	spare_count	RW	0	Channel [n] DMA transaction byte count for spare area

Table 4-6. DMA Channel[n].spare_count Register.

4.10.2.2 DMA_ch[n].spare_source_addr

Each DMA Channel Spare Area Source Address (**DMA_ch[n].spare_source_addr** register) holds the address of the next read request of the DMA operation. Software writes to this register are used to set up the starting spare-area source address for a channel, prior to enabling that channel. This register is automatically updated during the DMA operation after each read request, to show the location of the next read request. This register can only be written when the DMA channel is idle and the Channel Control register-enable bit is 0.

Bits	Name	Attr	Reset	Description
31:22				Reserved
21:0	spare_source_addr	RW	NR	Channel [n] source address for spare area Byte address of the next read request. Must be double word aligned.

Table 4-7. DMA Channel[n].spare_source_addr Register.

4.10.2.3 DMA_ch[n].spare_dest_addr

Each DMA channel Spare Area Destination Address register (**DMA_ch[n].spare_dest_addr**) holds the address of the next write request of a DMA operation. Software writes to this register are used to set up the starting spare-area destination address for a channel, prior to enabling that channel. This register is automatically updated during the DMA operation after each write request, to show the location of the next write request. This register can

only be written when the DMA channel is idle and the Channel Control register-enable bit is 0.

Bits	Name	Attr	Reset	Description
31:0	spare_dest_addr	RW	NR	Channel [n] destination address for spare area Byte address of the next write request. Must be double word aligned.

Table 4-8. DMA Channel[n] Spare Destination Address.

4.10.2.4 DMA_ch[n].spare_status

Each DMA Channel Spare Area Status register (**DMA_ch[n].spare_status**) captures completion status when a DMA operation ends. This includes the interrupt flags that show the reason for terminating the operation, and the number of bytes transferred by the operation. Software should write this register to 0 prior to starting each DMA operation. Except for the descriptor operation-error bit and the descriptor operation-done bit, this register can only be written when the DMA channel is idle and the Channel Control register-enable bit is 0. The descriptor operation-error bit and the descriptor operation-done bit can be written at any time. DMA Engine logic prevents writes to bits other than the descriptor operation-error bit and the descriptor operation-done bit, when it is unsafe to write these other bits.

Bits	Name	Attr	Reset	Description
31	spare_dm	RW	0	Descriptor memory error This bit is set to 1 when an error occurs on a descriptor memory read.
30	spare_oe	RW	0	Descriptor DMA operation error This bit is set to 1 in descriptor mode when a DMA operation ends due to an error.
29	spare_da	RW	0	Descriptor address error This bit is set to 1 when DMA_ch[n].spare_desc_addr is not double word aligned.
28	spare_dd	RW	0	Descriptor chain done This bit is set to 1 when the DMA operation of the last descriptor of a descriptor chain ends.
27	spare_od	RW	0	Descriptor DMA operation done This bit is set to 1 in descriptor mode when a DMA operation ends without an error.
26	spare_me	RW	0	DMA memory error This bit is set to 1 when an error occurs on a DMA data memory read.
25	spare_be	RW	0	DMA bus error This bit is set to 1 when an error occurs on an AHB bus read.
24				Reserved
23	spare_ae	RW	0	DMA address error This bit is set to 1 when either DMA_ch[n].spare_source_addr or DMA_ch[n].spare_dest_addr is not double word aligned.
22	spare_dn	RW	0	DMA operation done This bit is set to 1 when a DMA operation ends with the number of bytes shown in DMA_ch[n].control count field having been transferred. If a DMA operation ends due to an error before all data has been transferred, this bit is unchanged (0) if the register is cleared before DMA operation starts.

Bits	Name	Attr	Reset	Description
21:0	spare_count	RW	0	DMA transfer byte count Number of bytes moved by the DMA operation. When an error occurs, the DMA operation ends regardless of the number of bytes that have been transferred before the error. This field shows the number of good bytes moved when data is read from memory and when data is read from the AHB bus without an error. When data is read from the AHB bus and an error ends the DMA operation, this field can show a transfer count that is up to eight bytes more than the number of bytes transferred before the error. This field is automatically reset to 0 when a DMA operation starts.

Table 4-9. DMA Channel[n] Spare Status Register.

4.10.2.5 DMA_ch[n].spare_desc_addr

Each DMA Channel Spare Area Descriptor Address (**DMA_ch[n].spare_desc_addr** register) holds the address of the next descriptor that will be used to configure the channel if it is in descriptor mode as indicated by the Channel Control register descriptor mode bit. This register can only be written when the channel is idle and the Channel Control register-enable bit is 0.

Bits	Name	Attr	Reset	Description
31:0	spare_desc_addr	RW	NR	Byte address of the next descriptor that will be used to configure DMA_ch[n].control if it is in descriptor mode. Must be eight-byte aligned.

Table 4-10. DMA Channel[n] Spare Descriptor Address Register.

4.10.2.6 DMA_ch[n].control Register

Each DMA Channel Control register (**DMA_ch[n].control**) holds information that controls DMA operations for that channel. Software writes to this register are used to configure the channel and then to enable the channel to initiate the transaction.

Bits	Name	Attr	Reset	Description
31	en	RW	0	DMA channel enable 0 - DMA channel is idle 1 - DMA operation is in progress
30	d	RW	0	Descriptor mode 0 - DMA channel is configured by software writes to the DMA Engine Subsystem 1 - DMA channel is configured from a descriptor

Bits	Name	Attr	Reset	Description
29	wm	RW	1	Write memory 0 - DMA writes are to the AHB bus 1 - DMA writes are to memory
28	rm	RW	1	Read memory 0 - DMA reads are from the AHB bus 1 - DMA reads are from memory
27	ni	RW	1	No bus address increment flag 0 - The AHB bus address is incremented continually throughout the DMA operation 1 - Bus starting address written to DMA_ch[n].source_addr or DMA_ch[n].dest_addr will not be incremented during the DMA operation. Therefore, addresses on the AHB bus will only increment up to the amount shown in the bus transaction block size field of this DMA_ch[n].control , and then return to the value written to the DMA_ch[n].source_addr or DMA_ch[n].dest_addr to begin incrementing again when this bit is 1. (This no-increment mode affects only peripheral AMBA addressing, but not main memory addressing, mainly used as a work-around for peripherals that have a FIFO-type DMA interface such as the I2S unit.)
26:24	blk	RW	0	Bus transaction block size Each request to the AHB bus master in the DMA Engine Subsystem is a request to transfer the number of bytes indicated by this bus transaction block size field: The AHB bus master uses this information and chooses the optimal AHB bus burst type. Bus characteristics and usage should be considered to set this block size for efficient bus usage without starving other bus masters. This field can only be written when the DMA channel is idle and the Channel Control register-enable bit is 0.
23:22	ts	RW	0	Transfer size Indicates the data size for each data transfer on the AHB bus
21:0	count	RW	0	DMA transaction byte count Number of bytes moved by the DMA operation if the DMA operation ends without an error. When an error occurs, the DMA operation ends regardless of the number of bytes that have been transferred before the error.

Table 4-11. DMA Channel[n] Enable and Control Register.

blk	Bus Request Size	Comments
0	8 bytes	
1	16 bytes	
2	32 bytes	For best performance, use Bus Request Size of 32 bytes or less
3	64 bytes	
4	128 bytes	
5	256 bytes	
6	512 bytes	
7	1024 bytes	

Table 4-12. Definition of **blk** in **DMA_ch[n].control**.

ts[1:0]	Bus Transfer Size
0	Byte
1	Half word - 2 bytes
2	Word - 4 bytes
3	Double word - 8 bytes

Table 4-13. Definition of ts in DMA_ch[n] control.

4.10.2.7 DMA_ch[n].source_addr Register

Each DMA Channel Main Area Source Address register (**DMA_ch[n].source_addr**) holds the address of the next read request of the DMA operation. Software writes to this register are used to set up the starting source address for the channel, prior to enabling the channel. This register is automatically updated during the DMA operation after each read request, to show the location of the next read request. This register can only be written when the DMA channel is idle and the Channel Control register-enable bit is 0.

Bits	Name	Attr	Reset	Description
31:0	source_addr	RW	NR	Byte address of the next read request.

Table 4-14. DMA Channel[n] Main Source Address.

4.10.2.8 DMA_ch[n].dest_addr Register

Each DMA Channel Main Area Destination Address register (**DMA_ch[n].dest_addr**) holds the address of the next write request of the DMA operation. Software writes to this register are used to set up the starting destination address for the channel, prior to enabling the channel. This register is automatically updated during the DMA operation after each write request, to show the location of the next write request. This register can only be written when the DMA channel is idle and the Channel Control register-enable bit is 0.

Bits	Name	Attr	Reset	Description
31:0	dest_addr	RW	NR	Byte address of the next write request.

Table 4-15. DMA Channel[n] Main Destination Address.

4.10.2.9 DMA_ch[n].status Register

Each DMA Channel Main Area Status register (**DMA_ch[n].status**) captures completion status when a DMA operation concludes. This includes the interrupt flags that show the reason for the termination of the operation, and the number of bytes transferred by the operation. Software should write this register to 0 before starting each DMA operation. Except for the descriptor operation-error bit and the descriptor operation-done bit, this register can only be written when the channel is idle and the Channel Control register-enable bit is 0. The descriptor operation-error bit and the descriptor operation-done bit can be written at any time. DMA Engine logic prevents writes to bits other than the descriptor operation-error bit and the descriptor operation-done bit, when it is not safe to write these other bits.

Bits	Name	Attr	Reset	Description
31	dm	RW	0	Descriptor memory error This bit is set to 1 when an error occurs on a descriptor memory read.
30	oe	RW	0	Descriptor DMA operation error This bit is set to 1 in descriptor mode when a DMA operation ends due to an error.
29				Reserved
28	dd	RW	0	Descriptor chain done This bit is set to 1 when the DMA operation of the last descriptor of a descriptor chain ends.
27	od	RW	0	Descriptor DMA operation done This bit is set to 1 in descriptor mode when a DMA operation ends without an error.
26	me	RW	0	DMA memory error This bit is set to 1 when an error occurs on a DMA data memory read.
25	be	RW	0	DMA bus error This bit is set to 1 when an error occurs on an AHB bus read.
24	rwe	RW	0	DMA bus read/write error This bit is set to 1 when a DMA operation between two AHB bus devices is attempted by setting both the Channel Control register (DMA_ch[n].control) bits rm and wm to 0.
23				Reserved
22	dn	RW	0	DMA operation done This bit is set to 1 when a DMA operation ends with the number of bytes shown in DMA_ch[n].control count field having been transferred. If a DMA operation ends due to an error before all data has been transferred, this bit is unchanged (0) if the register is cleared before DMA operation starts.
21:0	count	RW	0	DMA transfer byte count Number of bytes moved by the DMA operation. When an error occurs, the DMA operation ends regardless of the number of bytes that have been transferred before the error. This field shows the number of good bytes moved when data is read from memory and when data is read from the AHB bus without an error. When data is read from the AHB bus and an error ends the DMA operation, this field can show a transfer count that is up to eight bytes more than the number of bytes transferred before the error. This field is automatically reset to 0 when a DMA operation starts.

Table 4-16. DMA Channel[n] Main Status Register.

4.10.2.10 DMA_ch[n].desc_addr Register

Each DMA Channel Main Area Descriptor Address (**DMA_ch[n].desc_addr** register) holds the address of the next descriptor that will be used to configure the channel if it is in descriptor mode as indicated by the Channel Control register descriptor mode bit. This register can only be written when the channel is idle and the Channel Control register-enable bit is 0.

Bits	Name	Attr	Reset	Description
31:0	desc_addr	RW	0	Byte address of the next descriptor that will be used to configure DMA_ch[n].control if it is in descriptor mode.

Table 4-17. DMA Channel[n] Main Descriptor Address Register.

4.10.2.11 DMA_ch[n].dsm_control Register

Bits	Name	Attr	Reset	Description
31	dual_space_mode	W	0	Dual-space mode enable 0 - Disable 1 - Enable
30:8				Reserved
7:4	main_jp_size	W	0x9	Main jump - dual-space mode If dual-space mode is enabled, the DMA engine jumps to transfer spare data when $2^{\text{main_jp_size}}$ Bytes of main data is transferred. This value must be at least 9.
3:0	spare_jp_size	W	0x4	Spare jump - dual-space mode If dual-space mode is enabled, the DMA engine jumps to transfer main data when $2^{\text{spare_jp_size}}$ Bytes of spare data is transferred. This value must be from 3 to 9.

Table 4-18. DMA Channel[n] Dual Space Mode Control Register.

4.10.2.12 DMA_int_sts Register

The DMA Interrupt register (**DMA_interrupt**) shows the channels that have outstanding interrupts.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7	i7	R	0	DMA channel 7 interrupt 1 indicates that DMA channel 7 has an outstanding interrupt. The Channel 7 Status register (DMA_ch[n].status) should be read for the details of this channel 7 interrupt. This bit is cleared by writing the Channel 7 Status register to 0.
6	i6	R	0	DMA channel 6 interrupt 1 indicates that DMA channel 6 has an outstanding interrupt. The Channel 6 Status register (DMA_ch[n].status) should be read for the details of this channel 6 interrupt. This bit is cleared by writing the Channel 6 Status register to 0.

Bits	Name	Attr	Reset	Description
5	i5	R	0	DMA channel 5 interrupt 1 indicates that DMA channel 5 has an outstanding interrupt. The Channel 5 Status register (DMA_ch[n]_status) should be read for the details of this channel 5 interrupt. This bit is cleared by writing the Channel 5 Status register to 0.
4	i4	R	0	DMA channel 4 interrupt 1 indicates that DMA channel 4 has an outstanding interrupt. The Channel 4 Status register (DMA_ch[n]_status) should be read for the details of this channel 4 interrupt. This bit is cleared by writing the Channel 4 Status register to 0.
3	i3	R	0	DMA channel 3 interrupt 1 indicates that DMA channel 3 has an outstanding interrupt. The Channel 3 Status register (DMA_ch[n]_status) should be read for the details of this channel 3 interrupt. This bit is cleared by writing the Channel 3 Status register to 0.
2	i2	R	0	DMA channel 2 interrupt 1 indicates that DMA channel 2 has an outstanding interrupt. DMA_ch2_status should be read for the details of this channel 2 interrupt. This bit is cleared by writing DMA_ch2_status to 0.
1	i1	R	0	DMA channel 1 interrupt 1 indicates that DMA channel 1 has an outstanding interrupt. DMA_ch1_status should be read for the details of this channel 1 interrupt. This bit is cleared by writing DMA_ch1_status to 0.
0	i0	R	0	DMA channel 0 interrupt

Table 4-19. DMA Interrupt Register.

4.11 DMA: Errors

Each DMA channel detects five error conditions:

1. Descriptor memory error
2. Descriptor address error
3. DMA memory error
4. DMA bus error
5. DMA bus read/write error

When an error is detected by a DMA channel, the operation ends, the channel is disabled, and an interrupt is signaled, if any of the following conditions are true:

- The DMA channel is in non-descriptor mode (the **DMA_ch[n].control** descriptor mode bit is 0).
- The DMA channel is in descriptor mode (the **DMA_ch[n].control** descriptor mode bit is 1) and it is configured to stop upon an error (the **DMA_ch[n].control** stop-on-error bit is 1).
- The DMA channel is in descriptor mode (the **DMA_ch[n].control** descriptor mode bit is 1) and a descriptor memory error or descriptor address error occurs.

When a DMA channel detects an error, the operation ends, the channel advances to the next operation and continues to run, and optionally, may signal an interrupt under the following circumstance:

- The DMA channel is in descriptor mode (the **DMA_ch[n].control** descriptor mode bit is 1), it is configured to continue upon an error (the **DMA_ch[n].control** stop-on-error bit is 0), and a DMA memory error, bus error, or bus read/write error occurs.

Refer to [Section 4.12](#) for a detailed description of the DMA Engine Subsystem interrupt mechanism.

4.11.1 DMA Error: Descriptor Address

A descriptor address error occurs when the address in the Channel Descriptor Address register (**DMA_ch[n].descriptor**) is not double word aligned. A descriptor address error has the following effects:

- No data is transferred for the descriptor with the address error.
- The Channel Status register (**DMA_ch[n].status**) address error bit is 1.
- The descriptor processing ends in the channel with the descriptor address error.

4.11.2 DMA Error: Bus

A bus error occurs on an AHB bus-read when the channel's AHB bus master receives an AHB bus ERROR response from the AHB bus slave that it was attempting to read. In the DMA Engine Subsystem, AHB bus-writes fail silently. The cause of a bus error depends on the AHB bus device that reports the error. Note that errors reported by devices may or may not indicate bad data. Therefore, the data associated with a bus error is written to its destination regardless. It is the responsibility of the device and host processor to communicate the data status through other device-specific registers or previously agreed-upon specifications. A bus error has the following effects:

- The DMA Engine Subsystem writes the data associated with the error and all earlier data to its destination.
- The Channel Status register (**DMA_ch[n].status**) DMA bus error bit is set to 1.
- The operation ends in the channel with the bus error.

4.11.3 DMA Error: Bus Read/Write

A bus read/write error occurs when both the source and destination are AHB bus devices. This combination is not permitted. Specifically, this occurs when both the Channel Control register (**DMA_ch[n].control**) read memory and write memory bits are set to 0. A bus read/write error has the following effects:

- No data is transferred.
- The Channel Status register (**DMA_ch[n].status**) DMA bus read/write error bit is set to 1.
- The operation ends in the channel with the bus read/write error.

4.12 DMA: Interrupts

Each DMA channel independently signals an interrupt when one of the following eight conditions occur for the channel. In addition, the channel must be either disabled or in descriptor mode and configured to send an interrupt upon an error or completed DMA operation:

1. Descriptor memory error
2. Descriptor address error
3. Completed descriptor processing (EOC encountered)
4. DMA memory error
5. DMA bus error
6. DMA bus read/write error
7. DMA address error
8. Completed DMA transfer

The DMA Engine Subsystem interrupt is signaled by assertion (1) of the DMA Interrupt Signal DMA_INT which is output to the VIC. Channels with active interrupts have their corresponding bit set to 1 in the DMA Interrupt register (**DMA_interrupt**). Each channel's active interrupts are shown in the Channel Status register (**DMA_ch[n].status**). Refer to [Section 4.9](#) above for more information.

An interrupt remains active until the software has cleared the interrupt by clearing the corresponding bit (write to 0) in one of the Channel Status registers (**DMA_ch[n].status**). The global DMA Engine Subsystem interrupt, signaled at the DMA_INT output port, is deasserted when all interrupts are cleared.

5. GRAPHICS DMA (GDMA)

This chapter provides information for the Graphics DMA (GDMA) co-processor. The chapter is organized as follows:

- [\(Section 5.1\) GDMA: Overview](#)
- [\(Section 5.2\) GDMA: Registers](#)
- [\(Section 5.3\) GDMA: Support for Pending Operations](#)
- [\(Section 5.4\) GDMA: Assumptions](#)
- [\(Section 5.5\) GDMA: Use](#)
- [\(Section 5.6\) GDMA: Operations](#)
- [\(Section 5.7\) GDMA: Pixel Formats](#)

5.1 GDMA: Overview

The Graphics DMA co-processor (GDMA) performs six image operations on one or two source area(s) in DRAM and writes the resulting image to a destination area, also in DRAM. The source and destination areas are each defined by a base address, row pitch (in bytes), width (in pixels), height (in rows), and pixel format (bits per pixel). The six operations ([Section 5.6](#)) are:

1. Linear copy
2. Two-dimensional copy
3. Composite
4. Alpha Blend (where source 2 is pre-multiplied by the alpha value)
5. Alpha Blend (where source 2 is not pre-multiplied by the alpha value)
6. Run-length decode

The source 1, source 2, and destination areas may have different pixel formats, provided that the destination format is equal to or larger than the source format(s). The supported pixel formats ([Section 5.7](#)) from smallest to largest are:

- 4-bit (lookup)
- 8-bit (lookup)
- 16-bit (1555)
- 16-bit (565)
- 24-bit
- 32-bit

The source 1 area may overlap arbitrarily (in DRAM) with the destination area. The source 2 area must not overlap with the destination area unless the source 2 area is identical to the destination area.

There are no byte-alignment restrictions for programming the GDMA registers that define the source and destination areas. Any integer value may be used to program these registers.

Up to eight outstanding operations can be issued to the GDMA by programming the appropriate GDMA registers via stores to the corresponding AHB addresses as described in [Section 5.3](#). When the GDMA completes all pending operations, it sends a completion interrupt to the VIC. The user can also check the number of operations that have not been completed by loading the AHB address and reading the register **GDMA_numpendingops** ([Section 5.2.2.11](#)).

5.2 GDMA: Registers

5.2.1 GDMA Registers: Map

The GDMA is located at base address 0xE001.5000 on the AHB bus. All GDMA registers below are four-byte aligned.

Register Offset	Register Name	Description
0x000	GDMA_source1base	Base Address of Source 1
0x004	GDMA_source1pitch	Row Pitch of Source 1
0x008	GDMA_source2base	Base Address of Source 2
0x00C	GDMA_source2pitch	Row Pitch of Source 2
0x010	GDMA_destinationbase	Base Address of Destination
0x014	GDMA_destinationpitch	Row Pitch of Destination
0x018	GDMA_width	Source / Destination Width
0x01C	GDMA_height	Source / Destination Height
0x020	GDMA_transparent	Transparent Value
0x024	GDMA_opcode	Operation Code
0x028	GDMA_numpendingops	Number of Pending Operations
0x02C	GDMA_pixelformat	Pixel Format
0x030	GDMA_alpha	Alpha Value for Blend
0x400	GDMA_lookuptable	Color Lookup Table

Table 5-1. Graphics DMA Register Map.

5.2.2 GDMA Registers: Details

5.2.2.1 **GDMA_source1base** Register

Bits	Name	Attr	Reset	Description
31:0	source1base	RW	NR	Base address of source 1 area

Table 5-2. Graphics DMA Source 1 Base Register.

5.2.2.2 **GDMA_source1pitch Register**

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:0	source1pitch	RW	NR	Row pitch of source 1 area (bytes)

Table 5-3. *Graphics DMA Source 1 Pitch Register.*

5.2.2.3 **GDMA_source2base Register**

Bits	Name	Attr	Reset	Description
31:0	source2base	RW	NR	Base address of source 2 area

Table 5-4. *Graphics DMA Source 2 Base Register.*

5.2.2.4 **GDMA_source2pitch Register**

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:0	source2pitch	RW	NR	Row pitch of source 2 area (bytes)

Table 5-5. *Graphics DMA Source 2 Pitch Register.*

5.2.2.5 **GDMA_destinationbase Register**

Bits	Name	Attr	Reset	Description
31:0	destinationbase	RW	NR	Base address of destination (area)

Table 5-6. *Graphics DMA Destination Base Register.*

5.2.2.6 **GDMA_destinationpitch Register**

Bits	Name	Attr	Reset	Description
31:13				Reserved
15:0	destinationpitch	RW	NR	Row pitch of destination area (bytes)

Table 5-7. *Graphics DMA Destination Pitch Register.*

5.2.2.7 GDMA_width Register

Bits	Name	Attr	Reset	Description
31:13				Reserved
12:0	width	RW	NR	Number of pixels minus 1 per row in source/destination area

Table 5-8. Graphics DMA Width Register.

5.2.2.8 GDMA_height Register

Bits	Name	Attr	Reset	Description
31:12				Reserved
11:0	height	RW	NR	Number of rows minus 1 in source/destination area

Table 5-9. Graphics DMA Height Register.

5.2.2.9 GDMA_transparent Register

Bits	Name	Attr	Reset	Description
31:0	transparent	RW	NR	Transparent value (for Composite operation) The valid width of this register depends on the source 2 pixel format. 8 bits - Source 2 at 8 bits 4 bits - Source 2 at 4 bits 15 bits - Source 2 at 16 bits (1555). MSB is ignored. 16 bits - Source 2 at 16 bits (565) 24 bits - Source 2 at 24 bits 32 bits - Source 2 at 32 bits

Table 5-10. Graphics DMA Transparent Value Register.

5.2.2.10 GDMA_opcode Register

Bits	Name	Attr	Reset	Description
31:3				Reserved

Bits	Name	Attr	Reset	Description
2:0	op	RW	0	Operation Code A write to this register initiates the corresponding operation. All other registers for this operation should be written before writing to this register. 0 - Linear copy 1 - Two-dimensional copy 2 - Composite 3 - Alpha blend (source 2 colors are pre-multiplied with alpha) 4 - Run length decode 5 - Alpha blend (source 2 colors are not pre-multiplied with alpha) 6 - No operation 7 - No operation

Table 5-11. Graphics DMA Operation Code Register.

5.2.2.11 GDMA_numpendingops Register

Bits	Name	Attr	Reset	Description
31:4				Reserved
3:0	numpendingops	R	0	Number of operations remaining to be completed. Value of zero indicates GDMA is idle.

Table 5-12. Graphics DMA Number of Pending Operations Register.

5.2.2.12 GDMA_pixelformat Register

Bits	Name	Attr	Reset	Description
31:12				Reserved
11	throttledram	RW	NR	ThrottledRAM When throttledram is 1, the maximum number of outstanding DRAM requests issued by the GDMA is 1. Otherwise, the maximum number of outstanding DRAM requests issued is 4.
10:8	source1format	RW	NR	Source 1 Pixel Format 0 - 8 bit 1 - 4 bit 2 - 16 bit (1555) 3 - 16 bit (565) 4 - 24 bit 5 - 32 bit
7				Reserved
6:4	source2format	RW	NR	Source 2 Pixel Format 0 - 8 bit 1 - 4 bit 2 - 16 bit (1555) 3 - 16 bit (565) 4 - 24 bit 5 - 32 bit

Bits	Name	Attr	Reset	Description
3				Reserved
2:0	destinationformat	RW	NR	Destination Pixel Format 0 - 8 bit 1 - 4 bit 2 - 16 bit (1555) 3 - 16 bit (565) 4 - 24 bit 5 - 32 bit

Table 5-13. Graphics DMA Pixel Format Register.

For Confidential
HAOTEX Only

5.2.2.13 GDMA_alpha Registers

Bits	Name	RW	Reset	Description
31:8				Reserved
7:0	alpha	RW	NR	Alpha value for Alpha Blend operations

Table 5-14. Graphics DMA Alpha Value Register.

5.2.2.14 GDMA_lookupable Register

Bits	Name	Attr	Reset	Description
31:0	lookup	W	NR	8b / 4b Lookup Table Color lookup table (256 entries) indexed by source color values when the source format is 8 bit or 4 bit. The valid width of the lookup table output depends on the destination pixel format. 16 bits - Destination format is 16 bits (1555) 16 bits - Destination format is 16 bits (565) 24 bits - Destination format is 24 bits 32 bits - Destination format is 32 bits

Table 5-15. Graphics DMA Color Lookup Table Register.

5.3 GDMA: Support for Pending Operations

There are eight instances (0 through 7) of all registers with the exception of **GDMA_numpendingops** and **GDMA_lookupable**. This facilitates support of up to eight pending operations on the chip.

When writing to these registers (0 through 7), addresses should be exactly as specified in [Section 5.2.1](#). The instance need not be specified in the address because the GDMA manages which instance of the register is written. See below for details on multiple operations.

When reading these registers, the instance (0 through 7) must be specified in bits [8:6] of the address. For example, the address for reading instance six of **GDMA_destinationpitch** is 0x60015194.

The instances sequence as follows. The GDMA contains an internal three-bit wide instance pointer that is incremented by 1 each time **GDMA_opcode** is written (i.e. each time an operation is initiated). It is not incremented when other registers are written and is not reset to 0 when **GDMA_numpendingops** becomes 0.

The following diagram illustrates the sequence in which GDMA register instances are updated.

EVENT	INSTANCE POINTER
Store(s) to register(s) other than GDMA_opcode	0
Store to GDMA_opcode	1
Store(s) to register(s) other than GDMA_opcode	1
Store to GDMA_opcode	2
Wait until TWO operations complete (i.e. GDMA_numpendingops becomes 0)	2
Store(s) to register(s) other than GDMA_opcode	2
Store to GDMA_opcode	3
Store(s) to register(s) other than GDMA_opcode	3
Store to GDMA_opcode	4
Store(s) to register(s) other than GDMA_opcode	4
Store to GDMA_opcode	5
Store(s) to register(s) other than GDMA_opcode	5
Store to GDMA_opcode	6
Store(s) to register(s) other than GDMA_opcode	6
Store to GDMA_opcode	7
Store(s) to register(s) other than GDMA_opcode	7
Store to GDMA_opcode	0
Store(s) to register(s) other than GDMA_opcode	0
Store to GDMA_opcode	1
Store(s) to register(s) other than GDMA_opcode	1
Store to GDMA_opcode	2
Wait until EIGHT operations complete (i.e. GDMA_numpendingops becomes 0)	2

The order of register instances updated for this group of eight operations: 2, 3, 4, 5, 6, 7, 0, 1
(NOT 0, 1, 2, 3, 4, 5, 6, 7).

Figure 5-1. Update Sequence for GDMA Register Instances.

5.4 GDMA: Assumptions

The proper functioning of the GDMA co-processor is contingent on the following:

- When two sources are used (the **GDMA_opcode** bit **op** = 2, 3 or 5), the source 2 area cannot overlap with the destination area unless the two areas exactly match.
 - For the source 2 area to exactly match the destination area, **source2base** must equal **destinationbase**; **source2pitch** must equal **destinationpitch**; and **GDMA_pixelformat**, **source2format** and **destinationformat** must be equal.
- For run-length decoding (**op** = 4), the source 1 area (i.e. the encoded stream) and the destination area cannot overlap.
- When any **GDMA_pixelformat** bit **source1format**, **source2format** or **destinationformat** is 4 bits, the number of pixels per row must be even (i.e. **GDMA_width** must be an odd number).
- When either **GDMA_pixelformat** bits **source1format** or **source2format** is 16 bits (1555 or 565) or 24 bits, and **destinationformat** is 32 bits, **GDMA_alpha** must be programmed.
- Source data that is 16 bits (1555 or 565) or 24 bits have no intrinsic alpha value, whereas destination data of 32 bits contain an alpha value per pixel, and **GDMA_alpha** is used to convert from the source format to the destination format.
- For run-length decoding (**op** = 4), **GDMA_pixelformat** must be programmed as 0x0 (for four outstanding DRAM requests) or 0x800 (for one outstanding DRAM request).

5.5 GDMA: Use

It is recommended to use the GDMA co-processor as follows:

1. Program all necessary GDMA registers for the operation by issuing stores to the corresponding AHB addresses. The table below indicates the registers to program for each operation.

Register	Linear Copy	2D Copy	Composite	Alpha Blend (op = 3 or op = 5)	Run Length Decode
GDMA_source1base	X	X	X	X	X
GDMA_source1pitch		X	X	X	
GDMA_source2base			X	X	
GDMA_source2pitch			X	X	
GDMA_destinationbase	X	X	X	X	X
GDMA_destinationpitch		X	X	X	X
GDMA_width	X	X	X	X	X
GDMA_height		X	X	X	X
GDMA_transparent			X		
GDMA_pixelformat	X	X	X	X	X
GDMA_alpha	X	X	X	X	

Table 5-16. Recommended Use of the Graphics DMA; Where an X Indicates a Register to Program.

2. Initiate the operation by issuing a store command to the **GDMA_opcode** AHB address.
3. Repeat Steps 1 and 2 (in order) up to seven more times (for up to eight total operations issued).
4. Optionally execute other unrelated ARM code.
5. Check the number of uncompleted operations by issuing a load command to the **GDMA_numpendingops** AHB address or wait for the GDMA completion interrupt.
6. Repeat Steps 1 through 5 to perform additional operations as desired.

5.6 GDMA: Operations

5.6.1 GDMA Operations: Linear Copy and 2D Copy

Linear copy (**op** = 0) and two-dimensional copy (**op** = 1) both transfer pixels from the source 1 area to the destination area.

Linear copy operates on only one row (i.e., **GDMA_source1pitch**, **GDMA_destinationpitch** and **GDMA_height** settings are irrelevant) whereas two-dimensional copy operates on one or more rows.

If **source1format** equals **destinationformat**, the source 1 pixels from DRAM are written to one of six source buffers (buffer 0, 1, 2, 3, 4 or 5), read from the same buffer, and then sent to DRAM. Note that when the **source1format** is 16 bits (1555) and the **destinationformat** is 16 bits (565), the GDMA considers **source1format** to equal **destinationformat**.

If **source1format** does not equal **destinationformat**, the source 1 pixels from DRAM are written to one of two source buffers (buffer 0 or 1), read from the source buffer, converted to the destination format and written to one of four destination buffers (buffer 2, 3, 4 or 5), and finally read from the destination buffer and sent to DRAM.

The supported pixel format combinations are:

Destination Format	Source 1 Format
4 bits	4 bits
8 bits	8 bits
16 bits (1555)	4 bits, 8 bits, 16 bits (1555)
16 bits (565)	4 bits, 8 bits, 16 bits (1555), 16 bits (565)
24 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits
32 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits, 32 bits

Table 5-17. Graphics DMA Pixel Format Combinations for Linear Copy and 2D Copy.

5.6.2 GDMA Operations: Composite

Composite operations manage pixels from both source 1 and source 2 areas based on the following equation:

```
destination = (source2 == transparent) ? ConvertToDestFormat(source1.color) :
ConvertToDestFormat(source2.color) ;
```

where **transparent** is in **source2format**.

The source 1 pixels from DRAM are written to one of two source 1 buffers (buffer 0 or 1), while the source 2 pixels from DRAM are written to one of two source 2 buffers (buffer 2 or 3). The source 1 and 2 pixels are read from their respective source buffers, converted to the destination format and operated upon. The result is written to one of two destination buffers (buffer 4 or 5) and finally read from the destination buffer and sent to DRAM.

The supported pixel format combinations are:

Destination Format	Source 1 Format	
4 bits	4 bits	4 bits
8 bits	8 bits	8 bits
16 bits (1555)	4 bits, 8 bits, 16 bits (1555)	4 bits, 8 bits, 16 bits (1555)
16 bits (565)	4 bits, 8 bits, 16 bits (1555), 16 bits (565)	4 bits, 8 bits, 16 bits (1555), 16 bits (565)
24 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits
32 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits, 32 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits, 32 bits

Table 5-18. Graphics DMA Pixel Format Combinations for Composite Operations.

5.6.3 GDMA Operations: Alpha Blend (Source 2 Pre-Multiplied by Alpha)

Alpha blend operations manage pixels from both source 1 and source 2 areas based on the equations below. Source 1 pixels are assumed to be pre-multiplied by their alpha values. Source 2 pixels also are assumed to be pre-multiplied by their alpha values (**op** = 3).

```
adjustedSource2Alpha = (source2.alpha == 0) ? 0 : (source2.alpha + 1) ;
destination.color = clip( source2.color + round( source1.color * (1 – adjustedSource2Alpha)
) );
```

where **source2.alpha** is 8 bits wide, and the color is red, green, blue or alpha;
round(value) = **(value + 128) >> 8**; and
clip(value) = **(value > 255) ? 255 : value**.

When **source2format** is 32 bits, **source2.alpha** is the most significant byte of the pixel; otherwise **source2.alpha** is taken from **GDMA_alpha**. When **destinationformat** is 16 bits (1555 or 565), the **destination.color** is further reduced to 5 bits ((**destination.color[7:0] + 4**) **>> 3**) or 6 bits ((**destination.color[7:0]+ 2**) **>> 2**).

The source 1 pixels from DRAM are written into one of two source 1 buffers (buffer 0 or 1), while the source 2 pixels from DRAM are written into one of two source 2 buffers (buffer 2 or 3). The source 1 and source 2 pixels are read from their respective source buffers, converted to the destination format and operated upon. The result is written to one of two destination buffers (buffer 4 or 5), and finally read from the destination buffer and sent to DRAM.

The supported pixel format combinations are:

Destination Format	Source 1 Format	Source 2 Format
16 bits (1555)	4 bits, 8 bits, 16 bits (1555)	4 bits, 8 bits, 16 bits (1555)
16 bits (565)	4 bits, 8 bits, 16 bits (1555), 16 bits (565)	4 bits, 8 bits, 16 bits (1555), 16 bits (565)
24 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits
32 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits, 32 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits, 32 bits

Table 5-19. Graphics DMA Pixel Format Combinations for Alpha Blend (Source 2 is Pre-Multiplied by Alpha).

5.6.4 GDMA Operations: Alpha Blend (Source 2 Not Pre-Multiplied by Alpha)

Alpha blend operations manage pixels from both source 1 and source 2 areas based on the equations below. Source 1 pixels are assumed to be pre-multiplied by their alpha values (**op** = 5), while source 2 pixels are not.

```
adjustedSource2Alpha = (source2.alpha == 0) ? 0 : (source2.alpha + 1);
destination.color = clip( round( source2.color * adjustedSource2Alpha +
    source1.color * (1 - adjustedSource2Alpha) ) );
```

where **source2.alpha** is 8 bits wide, and the color can be red, green, blue or alpha;
round(value) = **(value + 128) >> 8**; and
clip(value) = **(value > 255) ? 255 : value**.

When **source2format** is 32 bits, **source2.alpha** is the most significant byte of the pixel; otherwise **source2.alpha** is taken from **GDMA_alpha**. When **destinationformat** is 16 bits (1555 or 565), the **destination.color** is further reduced to 5 bits ((**destination.color[7:0] + 4**) **>> 3**) or 6 bits ((**destination.color[7:0] + 2**) **>> 2**).

The source 1 pixels from DRAM are written into one of two source 1 buffers (buffer 0 or 1), while the source 2 pixels from DRAM are written into one of two source 2 buffers (buffer 2 or 3). The source 1 and source 2 pixels are read from their respective source buffers, converted to the destination format and operated upon. The result is written into one of two destination buffers (buffer 4 or 5), and finally read from the destination buffer and sent to DRAM.

The supported pixel format combinations are:

Destination Format	Source 1 Format	Source 2 Format
16 bits (1555)	4 bits, 8 bits, 16 bits (1555)	4 bits, 8 bits, 16 bits (1555)
16 bits (565)	4 bits, 8 bits, 16 bits (1555), 16 bits (565)	4 bits, 8 bits, 16 bits (1555), 16 bits (565)
24 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits
32 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits, 32 bits	4 bits, 8 bits, 16 bits (1555), 16 bits (565), 24 bits, 32 bits

Table 5-20. Graphics DMA Pixel Format Combinations for Alpha Blend (Source 2 Not Pre-Multiplied by Alpha).

5.6.5 GDMA Operations: Run Length Decode

Run-length decode (**op** = 4) operations decode pixels from source 1 (the encoded stream) and write the decoded stream to the destination area. The pixel format is always 8 bits. The encoded stream is assumed to use big endian byte ordering, although it is stored in DRAM with little endian byte ordering.

The following tables demonstrate the concepts:

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

Table 5-21. Four-Byte Data Fetched from DRAM (Little Endian).

Type	Byte 0								Byte 1								Byte 2								Byte 3								
									1	1	1	1	1	1			2	2	2	2	1	1	1	1	3	3	2	2	2	2	2	2	
	7	6	5	4	3	2	1	0	5	4	3	2	1	0	9	8	3	2	1	0	9	8	7	6	1	0	9	8	7	6	5	4	
Pixel Code	V	V	V	V	V	V	V	V																									
End of Line Signal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																	
Run Length Zero (1 - 63)	0	0	0	0	0	0	0	0	0	0	N	N	N	N	N	N																	
Run Length Zero (64 - 16 K)	0	0	0	0	0	0	0	0	0	1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N		
Run Length (3 - 63)	0	0	0	0	0	0	0	0	1	0	N	N	N	N	N	N	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V		
Run Length (64 - 16 K)	0	0	0	0	0	0	0	0	1	1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	V	V	V	V	V		

Table 5-22. Decoding of Encoded Stream (BigEndian).

If the GDMA has decoded (**width** + 1) bytes before encountering an End of Line Signal, it will continue reading and decoding the encoded stream but discard the corresponding decoded data until it either encounters an End of Line Signal or has discarded approximately (**width** + 1)/2 bytes of decoded data, whichever comes first. This mechanism prevents the GDMA from hanging for an encoded stream that never contains an End of Line Signal.

If the GDMA encounters an End of Line Signal before it has decoded (**width** + 1) bytes, it will write 0xFF to the destination until it has written a total of (**width** + 1) bytes to the destination for the current line.

5.7 GDMA: Pixel Formats

The following diagrams provide layouts of the various GDMA pixel formats using source 1 as an example. Each box represents 1 byte of memory.

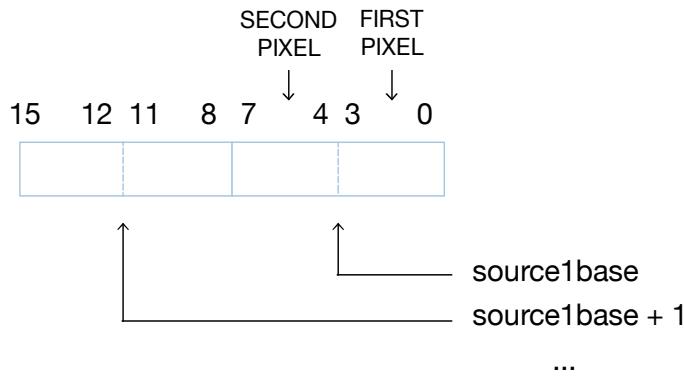


Figure 5-2. GDMA 4-bit Pixel Format.

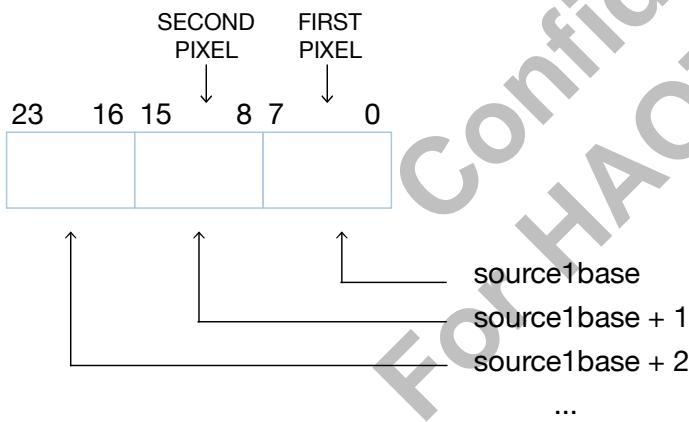


Figure 5-3. GDMA 8-bit Pixel Format.

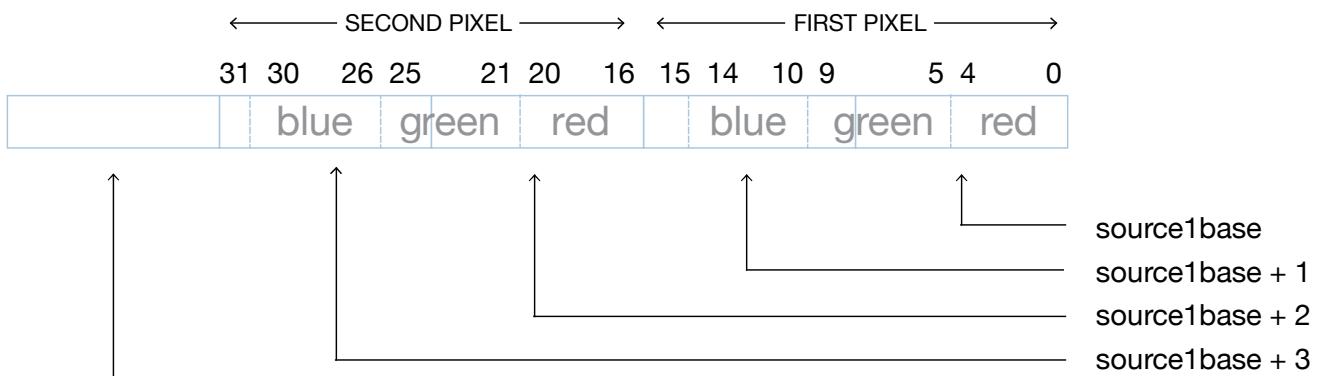


Figure 5-4. GDMA 16-bit Pixel Format (1555).

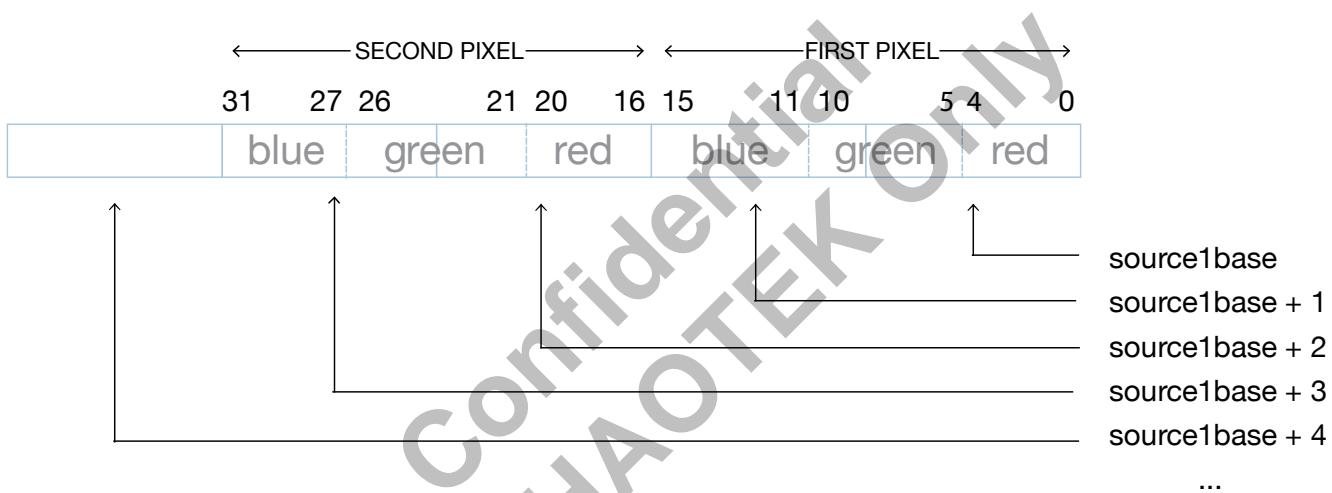


Figure 5-5. GDMA 16-bit Pixel Format (565).

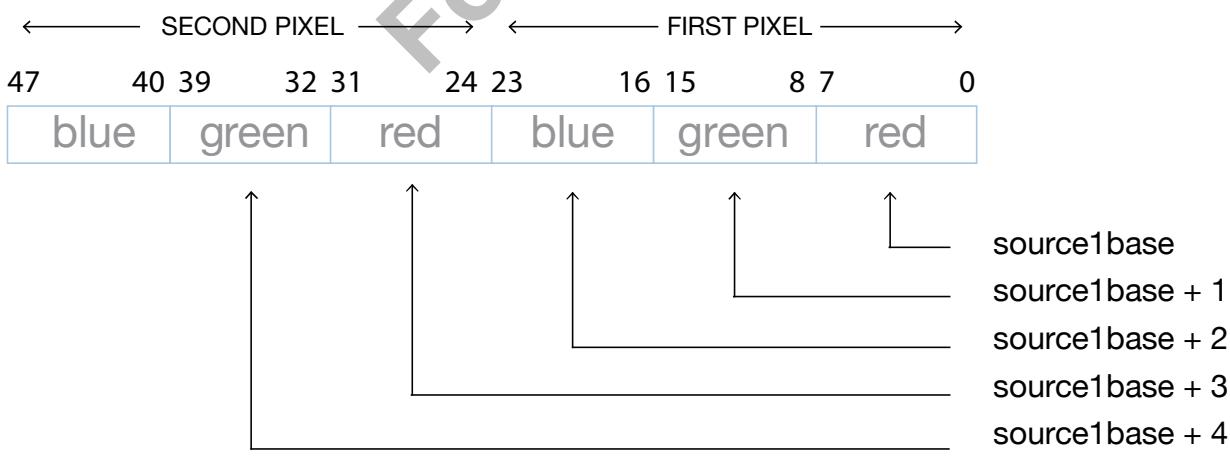


Figure 5-6. GDMA 24-bit Pixel Format.

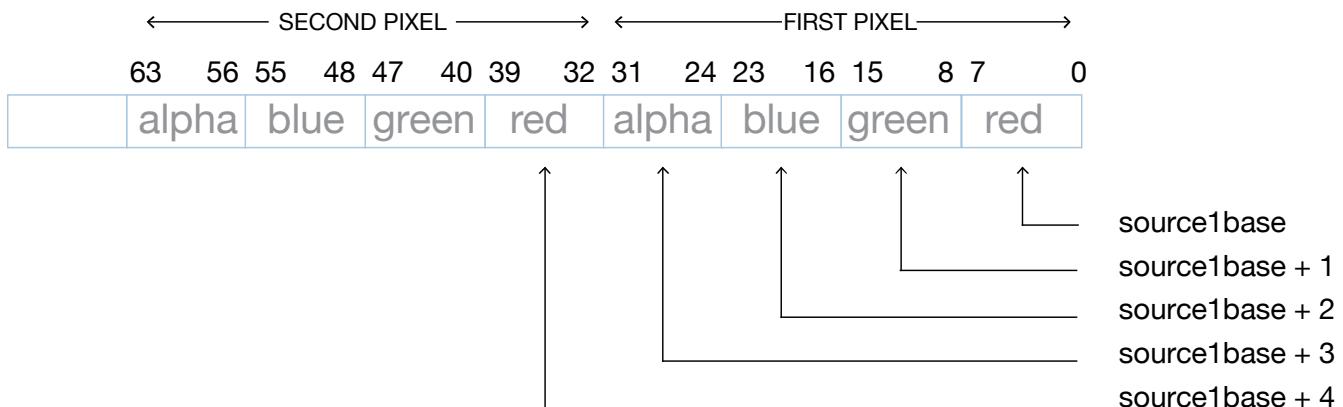


Figure 5-7. GDMA 32-bit Pixel Format.

6. VIDEO / SENSOR INPUT INTERFACE

The A12 Video / Sensor Input (VIN) module supports multiple serial and parallel input modes as shown below.

- Supports up to 8-lane SLVS / HiSPi input
- Supports up to 4-lane MIPI input
- In SVLS mode, two input channels may be combined to support a single 10-lane SLVS / HiSPi sensor
- Support for 14-bit parallel and LVCMOS sensors
- Fast External YUV Source

The Video / Sensor Input module is implemented as part of the DSP subsystem on the A12 chip and is configured using a set of APIs. [Chapter 3](#) of this manual provides an overview of the DSP subsystem and related APIs.

For further information regarding the A12 VIN interface:

- The VIN clock GCLK_SO_VIN is generally configured by A12 system software.
- VIN interface and pin details are provided in the “*A12 Datasheet*” Chapter 2 and Chapter 3, respectively. Electrical characteristics can be found in Chapter 4.
- Refer to “*A12 System Hardware*” for VIN interface connection detail.

7. VIDEO OUTPUT

The A12 Video Output (VOUT) interface supports a total of three output ports using two logical video channels. One VOUT channel is capable of driving digital output to RGB LCD panels, while the second VOUT channel drives either analog composite output or HDMI output to the on-chip HDMI transmitter (Tx) unit. Additional features of the VOUT module include:

- Logical division of video output into distinct sections as shown in the diagram below.

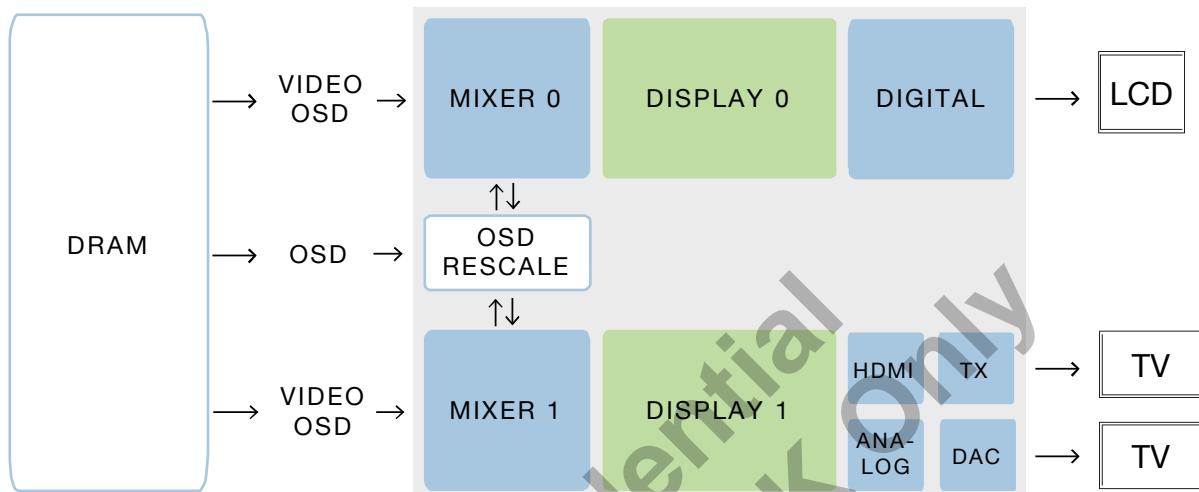


Figure 7-1. Relationship Between the Various A12 Video Output Sections.

- An optional On-Screen Display (OSD) Rescale unit which may be used by Channel 0 or Channel 1 (but not both) at any given time.
- Mixer sections which combine the OSD and video data.
- Display sections which create the output frame timing and convert the data into the relevant display format.
- A video digital-to-analog converter (DAC) and an on-chip HDMI transmitter (Tx).

For additional information on the A12 VOUT module, please refer to the following:

- A12 VOUT is controlled by the DSP subsystem ([Chapter 3](#)). Both analog and digital video outputs can be programmed using API calls.
- [Chapter 8](#) of this manual includes HDMI / CEC programming information.
- The VOUT clock GCLK_VO and LCD display clock GCLK_VO2 are generally configured by A12 system software.
- VOUT interface and pin details are provided in the “*A12 Datasheet*” Chapter 2 and Chapter 3, respectively. Electrical characteristics can be found in Chapter 4.
- Refer to the “*A12 System Hardware*” document for interface connection detail.

8. HDMI AND CEC

This chapter provides information regarding the A12 High Definition Multimedia Interface (HDMI) Controller and Consumer Electronics Control (CEC) interface. The chapter is organized as follows:

- [\(Section 8.1\) HDMI: Controller](#)
- [\(Section 8.2\) HDMI: Consumer Electronics Control](#)

i Note that the information contained in this chapter applies to A12 chips only.

8.1 HDMI: Controller

The A12 chip includes HDMI support and has a built-in HDMI PHY. The A12 HDMI controller conforms to the *HDMI Specification Version 1.4b* (2011) or *HDMI Spec*. The controller is covered in this section as follows:

- [\(Section 8.1.1\) HDMI Controller: Features](#)
- [\(Section 8.1.2\) HDMI Controller: Register Maps](#)
- [\(Section 8.1.3\) HDMI Controller: Register Details](#)
- [\(Section 8.1.4\) HDMI Controller: Operating Procedures](#)

The A12 provides I2S controllers which can be used to configure audio for HDMI output. Refer to [Chapter 9](#) for I2S programming.

The HDMI PHY clock GCLK_HDMI_PHY is an internal PLL reference clock that is configured by A12 system software.

8.1.1 HDMI Controller: Features

HDMI 1.4b-compliant transmitter (Tx):

- Flexible video input timing:
 - Various formats compliant with the *EIA/CEA-861D Standard*
 - Controlled through VOUT configuration
 - Sync timing settings for HDMI period placements
 - Supports 3D video formats
- Various supported audio formats:
 - Four output channels
 - Linear PCM compliant with the *IEC 60958 Standard*
 - Audio data rate up to video format
 - Supports High Bit-Rate Audio
- Master I2C/IDC interface (IDC2) for integrated DDC channel (APB instance)
- Monitor detection with Hot Plug Detect (HPD) signal

- Color space (Gamut) configuration

Configuration interface unit:

- Register banks for separate HDMISE and HDMI PHY
- AHB-to-internal configuration interface wrapper

HDMI signal encoder:

- Packetization of HDMI data for audio samples, infoframes, etc.
- Transition-minimized differential signaling (TMDS) period placement and signal encoding

Note that the A12 HDMI transmitter supports audio speaker placement. Refer to *EIA/CEA-861D* for details.

8.1.2 HDMI Controller: Register Maps

The HDMI Controller register space is divided into seven units.

- [\(Section 8.1.2.1\) HDMI Controller: System Register Map](#)
- [\(Section 8.1.2.2\) HDMI Controller: Data Island Part Register Map](#)
- [\(Section 8.1.2.3\) HDMI Controller: Video Part Register Map](#)
- [\(Section 8.1.2.4\) HDMI Controller: HDMISE Test Mode Register Map](#)
- [\(Section 8.1.2.5\) HDMI Controller: TMDS Asynchronous FIFO Register Map](#)
- [\(Section 8.1.2.6\) HDMI Controller: Debug Mode Register Map](#)
- [\(Section 8.1.2.7\) HDMI Controller: HDMI PHY Configuration Register Map](#)

8.1.2.1 HDMI Controller: System Register Map

Register Address	Register Name	Description
0x000	HDMI_int_enable	Interrupt Enable
0x004	HDMI_int_sts	Interrupt Status
0x008	HDMI_op_mode	Operation Mode
0x00C	HDMI_clock_gated	Clock Gated Control
0x010	HDMI_hdmise_soft_resetn	HDMISE Soft Reset
0x014	HDMI_sts	Real-Time Status Of Interrupt Sources

Table 8-1. HDMI Controller System Registers.

8.1.2.2 HDMI Controller: Data Island Part Register Map

Register Address	Register Name	Description
0x100	DI_aunit_mclk	Defines input audio clock frequency to produce CTS (Cycle Time Stamp) value based on 128 fs (fs = sampling frequency) formula
0x104	DI_aunit_ncts_ctrl	Defines whether CTS and N values are software or hardware controlled
0x108	DI_aunit_n	Current N Value for 128 fs Generation
0x10C	DI_aunit_cts	Current CTS Value for 128 fs Generation
0x110	DI_aunit_src	Audio Input Channel Selection
0x114	DI_aunit_cs0	Audio Channel Status Setting Word0
0x118	DI_aunit_cs1	Audio Channel Status Setting Word1
0x11C	DI_aunit_cs2	Audio Channel Status Setting Word2
0x120	DI_aunit_cs3	Audio Channel Status Setting Word3
0x124	DI_aunit_cs4	Audio Channel Status Setting Word4
0x128	DI_aunit_cs5	Audio Channel Status Setting Word5
0x12C	DI_aunit_layout	Audio Layout Type Indicator
0x130	DI_packet_tx_ctrl	Packet Transmit Control
0x134	DI_packet_general_ctrl	Set Enable or Disable AV Mute
0x138	DI_packet_acp0	ACP Packet Header
0x13C	DI_packet_acp1	ACP Packet Content PB0 - PB3
0x140	DI_packet_acp2	ACP Packet Content PB4 - PB6
0x144	DI_packet_acp3	ACP Packet Content PB7 - PB10
0x148	DI_packet_acp4	ACP Packet Content PB11 - PB13
0x14C	DI_packet_acp5	ACP Packet Content PB14 - PB17
0x150	DI_packet_acp6	ACP Packet Content PB18 - PB20
0x154	DI_packet_acp7	ACP Packet Content PB21 - PB24
0x158	DI_packet_acp8	ACP Packet Content PB25 - PB27
0x15C	DI_packet_isrc1_0	ISRC1 Packet Header
0x160	DI_packet_isrc1_1	ISRC1 Packet Content PB0 - PB3
0x164	DI_packet_isrc1_2	ISRC1 Packet Content PB4 - PB6
0x168	DI_packet_isrc1_3	ISRC1 Packet Content PB7 - PB10
0x16C	DI_packet_isrc1_4	ISRC1 Packet Content PB11 - PB13
0x170	DI_packet_isrc1_5	ISRC1 Packet Content PB14 - PB17
0x174	DI_packet_isrc1_6	ISRC1 Packet Content PB18 - PB20
0x178	DI_packet_isrc1_7	ISRC1 Packet Content PB21 - PB24
0x17C	DI_packet_isrc1_8	ISRC1 Packet Content PB25 - PB27
0x180	DI_packet_isrc2_0	ISRC2 Packet Header
0x184	DI_packet_isrc2_1	ISRC2 Packet Content PB0 - PB3
0x188	DI_packet_isrc2_2	ISRC2 Packet Content PB4 - PB6
0x18C	DI_packet_isrc2_3	ISRC2 Packet Content PB7 - PB10
0x190	DI_packet_isrc2_4	ISRC2 Packet Content PB11 - PB13
0x194	DI_packet_isrc2_5	ISRC2 Packet Content PB14 - PB17
0x198	DI_packet_isrc2_6	ISRC2 Packet Content PB18 - PB20
0x19C	DI_packet_isrc2_7	ISRC2 Packet Content PB21 - PB24
0x1A0	DI_packet_isrc2_8	ISRC2 Packet Content PB25 - PB27
0x1A4	DI_packet_avi0	AVI Packet Header

Register Address	Register Name	Description
0x1A8	DI_packet_avi1	AVI Packet Content PB0 - PB3
0x1AC	DI_packet_avi2	AVI Packet Content PB4 - PB6
0x1B0	DI_packet_avi3	AVI Packet Content PB7 - PB10
0x1B4	DI_packet_avi4	AVI Packet Content PB11 - PB13
0x1B8	DI_packet_avi5	AVI Packet Content PB14 - PB17
0x1BC	DI_packet_avi6	AVI Packet Content PB18 - PB20
0x1C0	DI_packet_avi7	AVI Packet Content PB21 - PB24
0x1C4	DI_packet_avi8	AVI Packet Content PB25 - PB27
0x1C8	DI_packet_spd0	SPD Packet Header
0x1CC	DI_packet_spd1	SPD Packet Content PB0 - PB3
0x1D0	DI_packet_spd2	SPD Packet Content PB4 - PB6
0x1D4	DI_packet_spd3	SPD Packet Content PB7 - PB10
0x1D8	DI_packet_spd4	SPD Packet Content PB11 - PB13
0x1DC	DI_packet_spd5	SPD Packet Content PB14 - PB17
0x1E0	DI_packet_spd6	SPD Packet Content PB18 - PB20
0x1E4	DI_packet_spd7	SPD Packet Content PB21 - PB24
0x1E8	DI_packet_spd8	SPD Packet Content PB25 - PB27
0x1EC	DI_packet_audio0	AUDIO Packet Header
0x1F0	DI_packet_audio1	AUDIO Packet Content PB0 - PB3
0x1F4	DI_packet_audio2	AUDIO Packet Content PB4 - PB6
0x1F8	DI_packet_audio3	AUDIO Packet Content PB7 - PB10
0x1FC	DI_packet_audio4	AUDIO Packet Content PB11 - PB13
0x200	DI_packet_audio5	AUDIO Packet Content PB14 - PB17
0x204	DI_packet_audio6	AUDIO Packet Content PB18 - PB20
0x208	DI_packet_audio7	AUDIO Packet Content PB21 - PB24
0x20C	DI_packet_audio8	AUDIO Packet Content PB25 - PB27
0x210	DI_packet_mpeg0	MPEG Packet Header
0x214	DI_packet_mpeg1	MPEG Packet Content PB0 - PB3
0x218	DI_packet_mpeg2	MPEG Packet Content PB4 - PB6
0x21C	DI_packet_mpeg3	MPEG Packet Content PB7 - PB10
0x220	DI_packet_mpeg4	MPEG Packet Content PB11 - PB13
0x224	DI_packet_mpeg5	MPEG Packet Content PB14 - PB17
0x228	DI_packet_mpeg6	MPEG Packet Content PB18 - PB20
0x22C	DI_packet_mpeg7	MPEG Packet Content PB21 - PB24
0x230	DI_packet_mpeg8	MPEG Packet Content PB25 - PB27
0x234	DI_packet_gamut0	Gamut Packet Header
0x238	DI_packet_gamut1	Gamut Packet Content PB0 - PB3
0x23C	DI_packet_gamut2	Gamut Packet Content PB4 - PB6
0x240	DI_packet_gamut3	Gamut Packet Content PB7 - PB10
0x244	DI_packet_gamut4	Gamut Packet Content PB11 - PB13
0x248	DI_packet_gamut5	Gamut Packet Content PB14 - PB17
0x24C	DI_packet_gamut6	Gamut Packet Content PB18 - PB20
0x250	DI_packet_gamut7	Gamut Packet Content PB21 - PB24
0x254	DI_packet_gamut8	Gamut Packet Content PB25 - PB27
0x258	DI_packet_vendor0	Vendor Specific Infoframe Header
0x25C	DI_packet_vendor1	Vendor Specific Infoframe Content PB0 - PB3

Register Address	Register Name	Description
0x260	DI_packet_vendor2	Vendor Specific Infoframe Content PB4 - PB6
0x264	DI_packet_vendor3	Vendor Specific Infoframe Content PB7 - PB10
0x268	DI_packet_vendor4	Vendor Specific Infoframe Content PB11 - PB13
0x26C	DI_packet_vendor5	Vendor Specific Infoframe Content PB14 - PB17
0x270	DI_packet_vendor6	Vendor Specific Infoframe Content PB18 - PB20
0x274	DI_packet_vendor7	Vendor Specific Infoframe Content PB21 - PB24
0x278	DI_packet_vendor8	Vendor Specific Infoframe Content PB25 - PB27
0x280 - 0x2C0		Reserved
0x2C4	DI_i2s_mode	I2S Operation Mode Selection
0x2C8	DI_i2s_rx_ctrl	I2S Rx Control
0x2CC	DI_i2s_wlen	I2S Word Length
0x2D0	DI_i2s_wpos	I2S Ignored Bits Between Two ws Edges
0x2D4	DI_i2s_slot	I2S Slot Count Register
0x2D8	DI_i2s_rx_fifo_gth	I2S Receiver FIFO Threshold
0x2DC	DI_i2s_clock	I2S Clock and Word select
0x2E0	DI_i2s_init	I2S Enable and Reset
0x2E4	DI_i2s_rx_data0	I2S0 FIFO Rx Data
0x2E8	DI_i2s_rx_data1	I2S1 FIFO Rx Data
0x2EC - 0x2F0		Reserved
0x2F4	DI_i2s_fifo_cntr	I2S FIFO Counter
0x2F8	DI_i2s_gate_off	I2S Gate Off
0x2FC	DI_packet_misc	Control Options of Data Islands

Table 8-2. HDMI Controller Data Island Part Registers.

8.1.2.3 HDMI Controller: Video Part Register Map

Register Address	Register Name	Description
0x360	Vunit_vblank_left	Bits for vertical blanking left porch
0x364	Vunit_vblank_pulse	Bits for vertical blanking pulse width
0x368	Vunit_vblank_right	Bits for vertical blanking right porch
0x36C	Vunit_hblank_left	Bits for horizontal blanking left porch
0x370	Vunit_hblank_pulse	Bits for horizontal blanking pulse porch
0x374	Vunit_hblank_right	Bits for horizontal blanking right porch
0x378	Vunit_vactive	Number of active lines of one frame
0x37C	Vunit_hactive	Number of active pixels of one line
0x380	Vunit_ctrl	VUNIT control
0x384	Vunit_vsync_detect	VSync detect enable

Table 8-3. HDMI Controller Video Part Registers.

8.1.2.4 HDMI Controller: HDMISE Test Mode Register Map

Register Address	Register Name	Description
0x388	HDMISE_tm	HDMISE Test Mode setting register

Table 8-4. HDMI Controller HDMISE Test Mode Registers.

8.1.2.5 HDMI Controller: TMDS Asynchronous FIFO Register Map

Register Address	Register Name	Description
0x38C	P2P_afifo_level	PCLK (GCLK_VO) to PHY_CLK (PHY_CLK_VO) AFIFO usage level indicator register
0x390	P2P_afifo_ctrl	PCLK (GCLK_VO) to PHY_CLK (PHY_CLK_VO) AFIFO control register

Table 8-5. HDMI Controller TMDS Asynchronous FIFO Registers.

8.1.2.6 HDMI Controller: Debug Mode Register Map

Register Address	Register Name	Description
0x394	HDMISE_dbg	Debugging Information

Table 8-6. HDMI Controller Debug Mode Registers.

8.1.2.7 HDMI Controller: HDMI PHY Configuration Register Map

Register Address	Register Name	Description
0x700	HDMI_phy_ctrl	HDMI PHY Control
0x704	HDMI_phy_ctrl1	HDMI PHY Control

Table 8-7. HDMI Controller HDMI PHY Configuration Registers.

8.1.3 HDMI Controller: Register Details

The set details for the full HDMI Controller register space are provided in this section. See [Section 8.1.2](#) for details on the PHY PLL section register, which uses a different base address.

8.1.3.1 HDMI_int_enable Register

Bits	Name	Attr	Reset	Description
31:26				Reserved
25	rx_sense_re-remove_en	RW	0	HDMI PHY Rx sense remove detection enable 0 - Disable 1 - Enable
24	hdmise_idle_en	RW	0	HDMISE in idle state interrupt enable 0 - Disable 1 - Enable
23	p2p_exceed_ub_en	RW	0	PCLK (GCLK_VO) to PHY_CLK (PHY_CLK_VO) AFIFO usage level exceed upper bound interrupt enable 0 - Disable 1 - Enable
22	p2p_below_lb_en	RW	0	PCLK (GCLK_VO) to PHY_CLK (PHY_CLK_VO) AFIFO usage level below lower bound interrupt enable 0 - Disable 1 - Enable
21	p2p_rempy_en	RW	0	PCLK (GCLK_VO) to PHY_CLK (PHY_CLK_VO) AFIFO read empty interrupt enable 0 - Disable 1 - Enable
20	p2p_wfull_en	RW	0	PCLK (GCLK_VO) to PHY_CLK (PHY_CLK_VO) AFIFO write full interrupt enable 0 - Disable 1 - Enable
19	cts_change_en	RW	0	CTS value change interrupt enable 0 - Disable 1 - Enable
18	i2s_rx_idle_en	RW	0	I2S Rx IDLE enable 0 - Disable 1 - Enable
17	i2s_rx_gth_val-id_en	RW	0	I2S Rx Threshold Count Valid enable 0 - Disable 1 - Enable
16	i2s_rx_fifo_over-en	RW	0	I2S Rx FIFO Overrun enable 0 - Disable 1 - Enable
15	i2s_rx_fifo_full_en	RW	0	I2S Rx FIFO Full enable 0 - Disable 1 - Enable
14	i2s_rx_fifo_empty_en	RW	0	I2S Rx FIFO Empty enable 0 - Disable 1 - Enable

Bits	Name	Attr	Reset	Description
13	<code>phy_rx_sense_en</code>	RW	0	HDMI PHY Rx Sense detected enable 0 - Disable 1 - Enable
12:6				Reserved
5	<code>cec_tx_interrupt_ok_en</code>	RW	0	CEC Tx interrupt ok enable 0 - Disable 1 - Enable
4	<code>cec_tx_interrupt_fail_en</code>	RW	0	CEC Tx interrupt fail enable 0 - Disable 1 - Enable
3	<code>cec_rx_interrupt_en</code>	RW	0	CEC Rx interrupt enable 0 - Disable 1 - Enable
2	<code>hot_plug_loss_en</code>	RW	0	Hot Plug loss enable 0 - Disable 1 - Enable
1	<code>hot_plug_detect_en</code>	RW	0	Hot Plug detect enable 0 - Disable 1 - Enable
0	<code>vsync_active_detect_en</code>	RW	0	VSync active edge detect enable 0 - Disable 1 - Enable

Table 8-8. HDMI Interrupt Enable Register.

8.1.3.2 HDMI_int_sts Register

Bits	Name	Attr	Reset	Description
31:26				Reserved
25	<code>phy_rx_sense_remove</code>	RW	0	HDMI PHY Rx sense remove detection 0 - Rx sense stays at high 1 - Rx sense removed
24	<code>hdmise_idle</code>	RW	0	HDMISE in idle state interrupt 0 - In busy state 1 - In idle state
23	<code>p2p_exceed_ub</code>	RW	0	PCLK (GCLK_VO) to PHY_CLK (PHY_CLK_VO) AFIFO usage level exceed upper bound interrupt 0 - No overflow 1 - Overflow
22	<code>p2p_below_lb</code>	RW	0	PCLK (GCLK_VO) to PHY_CLK (PHY_CLK_VO) AFIFO usage level below lower bound interrupt 0 - Not below lower bound 1 - Below lower bound
21	<code>p2p_rempy</code>	RW	0	PCLK (GCLK_VO) to PHY_CLK (PHY_CLK_VO) AFIFO read empty interrupt 0 - AFIFO not empty 1 - AFIFO empty

Bits	Name	Attr	Reset	Description
20	p2p_wfull	RW	0	PCLK (GCLK_VO) to PHY_CLK (PHY_CLK_VO) AFIFO write full interrupt 0 - AFIFO not full 1 - AFIFO full
19	cts_change	RW	0	CTS value change interrupt 0 - CTS value not changed 1 - CTS value changed
18	i2s_rx_idle	RW	1	I2S Rx IDLE. 0 - Not idle 1 - Idle
17	i2s_rx_gth_valid	RW	0	I2S Rx Threshold Count Valid 0 - Threshold count not reached 1 - Threshold count is reached.
16	i2s_rx_fifo_over	RW	0	I2S Rx FIFO Overrun 0 - Not overrun 1 - Overrun
15	i2s_rx_fifo_full	RW	0	I2S Rx FIFO Full 0 - Not full 1 - Full
14	i2s_rx_fifo_empty	RW	1	I2S Rx FIFO Empty 0 - Not empty 1 - Empty
13	phy_rx_sense	RW	0	HDMI PHY Rx Sense detected 0 - No Rx sense detected 1 - Rx sense detected
12:6				Reserved
5	cec_tx_interrupt_ok	RW	0	CEC Tx interrupt ok Interrupt flag for a successful Tx frame 0 - No CEC Tx interrupt ok 1 - CEC Tx interrupt ok
4	cec_tx_interrupt_fail	RW	0	CEC Tx interrupt fail Interrupt flag for a failed Tx frame (after retry fail) 0 - No CEC Tx interrupt fail 1 - CEC Tx interrupt fail
3	cec_rx_interrupt	RW	0	CEC Rx interrupt Interrupt flag for a successful Rx (frame is available) 0 - No CEC Rx interrupt 1 - CEC Rx interrupt
2	hot_plug_loss	RW	0	Hot Plug Loss 0 - Not loss 1 - Loss
1	hot_plug_detect	RW	0	Hot Plug detect 0 - Not detected 1 - Detected
0	vsync_active_detect	RW	0	VSync active edge detect 0 - Not detected 1 - Detected

Table 8-9. HDMI Interrupt Status Register.

8.1.3.3 HDMI_op_mode Register

Bits	Name	Attr	Reset	Description
31:2				Reserved
1	op_en	RW	0	Operation enable Set this bit to start HDMI or DVI operation. This bit must be set after all other registers are set. This bit will self set to 0 when HDMI_hdmise_soft_resetn is low. 0 - Operation disabled 1 - Operation enabled
0	op_mode	RW	0	Operation mode 0 - DVI mode 1 - HDMI mode For HDMI mode, the HDCPCE control register (HDCP_hdcpce_ctl) field use_eess must be set with this bit or the HDMISE will be forced to run in DVI mode)

Table 8-10. Operation Mode Register.

8.1.3.4 HDMI_clock_gated Register

This register is used to turn off clocks propagating to HDMI Tx logic to save power. Internal logic does not see the clock toggle.

Bits	Name	Attr	Reset	Description
31:3				Reserved
2	cec_clock_en	RW	1	CEC clock enable 0 - Disable 1 - Enable
1	hdcp_clock_en	RW	1	HDCP clock enable 0 - Disable 1 - Enable
0	hdmise_clock_en	RW	1	HDMISE clock enable 0 - Disable 1 - Enable

Table 8-11. HDMI Clock Gated Control Register.

8.1.3.5 HDMI_hdmise_soft_resetn Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	hdmise_soft_re-setn	RW	0	HDMISE soft reset 0 - Reset state (reset pixel, PHY and audio clock domains) 1 - Non-reset state This bit will reset all registers from offset 0x100.

Figure 8-1. HDMI HDMISE Soft Reset Register.

8.1.3.6 HDMI_sts Register

Bits	Name	Attr	Reset	Description
31:25				Reserved
24	hdmi_se_idle	R	0	On line HDMI SE Timing State Machine Idle signal
23	p2p_exceed_ub	R	0	On line P2P AFIFO exceed upper bound signal
22	p2p_below_lb	R	0	On line P2P AFIFO below lower bound signal
21	p2p_rempy	R	0	On line P2P AFIFO read empty signal
20	p2p_wfull	R	0	On line P2P AFIFO write full signal
19	cts_change	R	0	On line CTS change signal
18	i2s_idle	R	0	On line I2S idle signal
17	i2s_gth_valid	R	0	On line I2S FIFO threshold valid signal
16	i2s_fifo_over	R	0	On line I2S FIFO overrun signal
15	i2s_fifo_full	R	0	On line I2S FIFO full signal
14	i2s_fifo_empty	R	0	On line I2S FIFO empty signal
13	rx_sense	R	0	On line Rx sense signal
12:6				Reserved
5	cec_tx_interrupt_ok	R	0	On line CEC Tx interrupt ok signal
4	cec_tx_interrupt_fail	R	0	On line CEC Tx interrupt fail signal
3	cec_rx_interrupt	R	0	On line CEC Rx interrupt signal
2				Reserved
1	hpd	R	0	One line hot plug signal
0				Reserved

Figure 8-2. HDMI HDMISE Real Time Status Register.

8.1.3.7 DI_aunit_mclk Register

This register defines the input audio clock frequency to produce Cycle Time Stamp (CTS) values (using 128 fs or sampling frequency). It informs HDMI Tx of the clock ratio between the audio main clock and audio sample frequency for HDMI Rx audio clock recovery.

Bits	Name	Attr	Reset	Description
31:3				Reserved
2:0	mclk_conf	RW	0x1	MCLK (GCLK_AU or GCLK_AU_1CH) Audio main clock input frequency mode: 0x0 - 128*fs 0x1 - 256*fs 0x2 - 384*fs 0x3 - 512*fs 0x4 - 768*fs 0x5 - 192*fs 0x6 - 64*fs 0x7 - 32*fs Others - Reserved

Table 8-12. HDMI Controller Data Island Register for Audio Clock CTS Frequency.

8.1.3.8 DI_aunit_ncts_ctrl Register

This register is used to define whether CTS and N values are software- or hardware-controlled.

Bits	Name	Attr	Reset	Description
31:2				Reserved
1	ncts_en	RW	0	N/CTS Packet Enable 0 - N/CTS packet disabled 1 - N/CTS packet enabled
0	cts_sel	RW	0	CTS Source Select Mode: 0 - Hardware update mode. In this mode, CTS value will be determined by Hardware (recommended). 1 - Software update mode. In this mode, CTS value is from the software setting.

Table 8-13. HDMI Controller Data Island Register for Audio Clock CTS and N Values Control.

It is recommended to determine CTS values by hardware; however, CTS values can be programmed using software mode. Note that the **DI_aunit_ncts_ctrl > cts_sel** high (software mode) is for diagnostic purpose only.

8.1.3.9 DI_aunit_n Register

This register is used to set the current N Value for 128 fs generation.

Bits	Name	Attr	Reset	Description
31:20				Reserved
19:0	aunit_n	RW	0	Current N Value for Audio Clock Regeneration Software must write an integer value larger than 0 to this register after reset.

Table 8-14. HDMI Controller Data Island Register for Audio Clock Regeneration N Values.

8.1.3.10 DI_aunit_cts Register

This register provides the current Cycle Time Stamp (CTS) value for 128 fs generation.

Bits	Name	Attr	Reset	Description
31:20				Reserved
19:0	aunit_cts	RW	0	Current CTS value When DI_aunit_ncts_ctrl > cts_sel = 0, this field is read-only and stands for the current hardware determined CTS value for Audio Clock Regeneration When DI_aunit_ncts_ctrl > cts_sel = 1, this field is readable and writable and stands for current CTS setting value of cts_sel in software mode for Audio Clock Regeneration.

Table 8-15. HDMI Controller Data Island Register for Audio Clock Regeneration CTS Values.

8.1.3.11 DI_aunit_src Register

Bits	Name	Attr	Reset	Description
31:6				Reserved
5	flat_line1	RW	0	I2S Channel 1 is flat line sample 0- Flat line 1- Not flat line
4	flat_line0	RW	0	I2S Channel 0 is flat line sample 0- Flat line 1- Not flat line
3:2				Reserved
1	i2s1_en	RW	1	I2S Channel 1 Enable 0 - Disable 1 - Enable
0	i2s0_en	RW	1	I2S Channel 0 Enable 0 - Disable 1 - Enable

Table 8-16. HDMI Controller Data Island Register for Audio Input Channel Selection.

One I2S unit supports two audio channels (Left and Right Channels).

8.1.3.12 DI_aunit_cs0 Register

Bits	Name	Attr	Reset	Description
31:0	aunit_cs0	RW	0	Channel Status Bits 31 - 0

Table 8-17. HDMI Controller Data Island Register for Audio Channel Status Setting Word0.

8.1.3.13 DI_aunit_cs1 Register

Bits	Name	Attr	Reset	Description
31:0	aunit_cs1	RW	0	Channel Status Bits 63 - 32

Table 8-18. HDMI Controller Data Island Register for Audio Channel Status Setting Word1.

8.1.3.14 DI_aunit_cs2 Register

Bits	Name	Attr	Reset	Description
31:0	aunit_cs2	RW	0	Channel Status Bits 95 - 64

Table 8-19. HDMI Controller Data Island Register for Audio Channel Status Setting Word2.

8.1.3.15 DI_aunit_cs3 Register

Bits	Name	Attr	Reset	Description
31:0	aunit_cs3	RW	0	Channel Status Bits 127 - 96

Table 8-20. HDMI Controller Data Island Register for Audio Channel Status Setting Word3.

8.1.3.16 DI_aunit_cs4 Register

Bits	Name	Attr	Reset	Description
31:0	aunit_cs4	RW	0	Channel Status Bits 159 - 128

Table 8-21. HDMI Controller Data Island Register for Audio Channel Status Setting Word4.

8.1.3.17 DI_aunit_cs5 Register

Bits	Name	Attr	Reset	Description
31:0	aunit_cs5	RW	0	Channel Status Bits 191 - 160

Table 8-22. HDMI Controller Data Island Register for Audio Channel Status Setting Word5.

8.1.3.18 DI_aunit_layout Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	layout	RW	0	Audio Packet Layout Type 0 - Layout 0 (supports 2 channels) 1 - Layout 1 (supports 3 to 8 channels)

Table 8-23. HDMI Controller Data Island Audio Layout Type Indicator Register.

8.1.3.19 DI_packet_tx_ctrl Register

Bits	Name	Attr	Reset	Description
31	buf_switch_en	RW	0	Double Buffer Switch Enable 0 - Disable 1 - Enable This bit is for switching double buffer. When this bit is set, the contents of all packets will take effect when encountering the next VSync. This bit will be cleared automatically when the next VSync is detected.
30:18				Reserved
17	vendor_rpt	RW	0	Vendor Specific Infoframe Transmission 0 - Disable 1 - Enable

Bits	Name	Attr	Reset	Description
16	vendor_en	RW	0	Vendor Specific Infoframe Transmission Enable 0 - Disable 1 - Enable
15	gamut_rpt	RW	0	Repeat Gamut Metadata Infoframe Transmission Only support phase 0 transmission in the current design 0 - Disable (send once after gamut_en is set) 1 - Enable (send in VBLANK period of every frame)
14	gamut_en	RW	0	Gamut Metadata Infoframe Transmission Enable Only phase 0 transmission is supported in the current design. 0 - Disable 1 - Enable
13	mpeg_rpt	RW	0	Repeat MPEG Infoframe Transmission 0 - Disable (send once after mpeg_en is set) 1 - Enable (send in VBLANK period of every frame)
12	mpeg_en	RW	0	MPEG Infoframe Transmission Enable 0 - Disable 1 - Enable
11	aud_rpt	RW	0	Repeat AUD Infoframe Transmission 0 - Disable (send once after aud_en is set) 1 - Enable (send in VBLANK period of every frame)
10	aud_en	RW	0	AUD Infoframe Transmission Enable 0 - Disable 1 - Enable
9	spd_rpt	RW	0	Repeat SPD Infoframe Transmission 0 - Disable (send once after spd_en is set) 1 - Enable (send in VBLANK period of every frame)
8	spd_en	RW	0	SPD Infoframe Transmission Enable 0 - Disable 1 - Enable
7	avi_rpt	RW	0	Repeat AVI Infoframe Transmission 0 - Disable (send once after avi_en is set) 1 - Enable (send in VBLANK period of every frame)
6	avi_en	RW	0	AVI Infoframe Transmission Enable 0 - Disable 1 - Enable
5	isrc_rpt	RW	0	Repeat ISRC Packet Transmission 0 - Disable (send once after isrc1_en is set) 1 - Enable (send in VBLANK period of every frame)
4	isrc_en	RW	0	ISRC Packet Transmission Enable 0 - Disable 1 - Enable
3	acp_rpt	RW	0	Repeat ACP Packet Transmission 0 - Disable (send once after acp_en is set) 1 - Enable (send in VBLANK period of every frame)
2	acp_en	RW	0	ACP Packet Transmission Enable 0 - Disable 1 - Enable
1	gen_rpt	RW	0	Repeat General Control Packet Transmission 0 - Disable (send once after acp_en is set) 1 - Enable (send in VBLANK period of every frame)

Bits	Name	Attr	Reset	Description
0	gen_en	RW	0	General Control Packet Transmission Enable 0 - Disable 1 - Enable

Table 8-24. HDMI Controller Data Island Packet Transmit Control Register.

If a data packet is transmitted one time only, the enable bit will be cleared by hardware following transmission. There are two buffers (BUF_A and BUF_B) designed for infoframe and packet content storage. After reset, BUF_A faces the AHB interface and the software can program preferred infoframe and packet content through address offset 0x138 - 0x278 to BUF_A. Meanwhile BUF_B faces the HDMI internal logic and state machines. The software then sets register **DI_packet_tx_ctrl** to determine which infoframes and packets are enabled and whether they will be transmitted once or continuously. Note that setting **DI_packet_tx_ctrl** bit 31 (**buf_switch_en**) to 1 enables the double buffer switch.

When there is a new VSync signal, BUF_A will switch to face internal logic, and contents from 0x138 - 0x278 will be sent depending on how they are enabled. Meanwhile, BUF_B will switch to face AHB. If another setting of infoframes and packets is preferred, the software can program new content and set **DI_packet_tx_ctrl** again.

8.1.3.20 DI_packet_general_ctrl Register

Bits	Name	Attr	Reset	Description
31:17				Reserved
16	def_phase	RW	0	Default Phase Current design supports 24 bits video only. 0 - The PP bits may vary and the first pixel of each Video Data Period may vary. 1 - The first pixel of each Video Data period shall always have a pixel packing phase of 0 (10P0, 12P0, 16P0). The first pixel following each Video Data period shall always have a pixel packing phase of 0 (10C0, 12C0, 16C0). The first pixel following every transition of HSync or VSync shall always have a pixel packing phase of 0 (10C0, 12C0, 16C0).
15:12	pp	RW	0	Pixel Packing Phase Current design only supports 24 bits video. 0x0 - Phase 4 (10P4) 0x1 - Phase 1 (10P1, 12P1, 16P1) 0x2 - Phase 2 (10P2, 12P2) 0x3 - Phase 3 (10P3) 0x4 - Reserved Others - Reserved
11:8	cd	RW	0	Color Depth Current design only supports 24 bits video. 0x0 - Color Depth not indicated 0x4 - 24 bits per pixel 0x5 - 30 bits pixel (not supported) 0x6 - 36 bits per pixel (not supported) 0x7 - 48 bits per pixel (not supported) Others - Reserved
7:5				Reserved

Bits	Name	Attr	Reset	Description
4	clr_avmute	RW	0	Clear AV Mute Flag 0 - Disable 1 - Enable
3:1				Reserved
0	set_avmute	RW	0	Set AV Mute Flag 0 - Disable 1 - Enable

Table 8-25. HDMI Controller Data Island Packet General Control Register.

Note that if the illegal combination of {CLR_AVMUTE, SET_AVMUTE}{(1,1)} is set, then {0,0} will be sent instead.

8.1.3.21 DI_packet_acp0 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	acp_hb2	RW	0	ACP packet header byte 2
15:8	acp_hb1	RW	0	ACP packet header byte 1
7:0	acp_hb0	RW	0	ACP packet header byte 0

Table 8-26. HDMI Controller Data Island ACP Packet Header.

8.1.3.22 DI_packet_acp1 Register

Bits	Name	Attr	Reset	Description
31:24	acp_pb3	RW	0	ACP packet PB3
23:16	acp_pb2	RW	0	ACP packet PB2
15:8	acp_pb1	RW	0	ACP packet PB1
7:0	acp_pb0	RW	0	ACP packet PB0

Table 8-27. Register for ACP Packet Content PB0 - PB3.

8.1.3.23 DI_packet_acp2 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	acp_pb6	RW	0	ACP packet PB6
15:8	acp_pb5	RW	0	ACP packet PB5
7:0	acp_pb4	RW	0	ACP packet PB4

Table 8-28. HDMI Controller Data Island ACP Packet Content PB4 - PB6.

8.1.3.24 DI_packet_acp3 Register

Bits	Name	Attr	Reset	Description
31:24	acp_pb10	RW	0	ACP packet PB10
23:16	acp_pb9	RW	0	ACP packet PB9
15:8	acp_pb8	RW	0	ACP packet PB8
7:0	acp_pb7	RW	0	ACP packet PB7

Table 8-29. HDMI Controller Data Island ACP Packet Content PB7 - PB10.

8.1.3.25 DI_packet_acp4 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	acp_pb13	RW	0	ACP packet PB13
15:8	acp_pb12	RW	0	ACP packet PB12
7:0	acp_pb11	RW	0	ACP packet PB11

Table 8-30. HDMI Controller Data Island ACP Packet Content PB11 - PB13.

8.1.3.26 DI_packet_acp5 Register

Bits	Name	Attr	Reset	Description
31:24	acp_pb17	RW	0	ACP packet PB17
23:16	acp_pb16	RW	0	ACP packet PB16
15:8	acp_pb15	RW	0	ACP packet PB15
7:0	acp_pb14	RW	0	ACP packet PB14

Table 8-31. HDMI Controller Data Island ACP Packet Content PB14 - PB17.

8.1.3.27 DI_packet_acp6 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	acp_pb20	RW	0	ACP packet PB20
15:8	acp_pb19	RW	0	ACP packet PB19
7:0	acp_pb18	RW	0	ACP packet PB18

Table 8-32. HDMI Controller Data Island ACP Packet Content PB18 - PB20.

8.1.3.28 DI_packet_acp7 Register

Bits	Name	Attr	Reset	Description
31:24	acp_pb24	RW	0	ACP packet PB24

Bits	Name	Attr	Reset	Description
23:16	acp_pb23	RW	0	ACP packet PB23
15:8	acp_pb22	RW	0	ACP packet PB22
7:0	acp_pb21	RW	0	ACP packet PB21

Table 8-33. HDMI Controller Data Island ACP Packet Content PB21 - PB24.

8.1.3.29 DI_packet_acp8 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	acp_pb27	RW	0	ACP packet PB27
15:8	acp_pb26	RW	0	ACP packet PB26
7:0	acp_pb25	RW	0	ACP packet PB25

Table 8-34. HDMI Controller Data Island ACP Packet Content PB25 - PB27.

8.1.3.30 DI_packet_isrc1_0 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	isrc1_hb2	RW	0	IRSC1 packet header byte2
15:8	isrc1_hb1	RW	0	ISRC1 packet header byte1
7:0	isrc1_hb0	RW	0	ISRC1 packet header byte0

Table 8-35. HDMI Controller Data Island ISRC1 Packet Header.

8.1.3.31 DI_packet_isrc1_1 Register

Bits	Name	Attr	Reset	Description
31:24	isrc1_pb3	RW	0	IRSC1 packet PB3
23:16	isrc1_pb2	RW	0	IRSC1 packet PB2
15:8	isrc1_pb1	RW	0	ISRC1 packet PB1
7:0	isrc1_pb0	RW	0	ISRC1 packet PB0

Table 8-36. HDMI Controller Data Island ISRC1 Packet Content PB0 - PB3.

8.1.3.32 DI_packet_isrc1_2 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	isrc1_pb6	RW	0	IRSC1 packet PB6
15:8	isrc1_pb5	RW	0	ISRC1 packet PB5

Bits	Name	Attr	Reset	Description
7:0	isrc1_pb4	RW	0	ISRC1 packet PB4

Table 8-37. HDMI Controller Data Island ISRC1 Packet Content PB4 - PB6.

8.1.3.33 DI_packet_isrc1_3 Register

Bits	Name	Attr	Reset	Description
31:24	isrc1_pb10	RW	0	IRSC1 packet PB10
23:16	isrc1_pb9	RW	0	IRSC1 packet PB9
15:8	isrc1_pb8	RW	0	ISRC1 packet PB8
7:0	isrc1_pb7	RW	0	ISRC1 packet PB7

Table 8-38. HDMI Controller Data Island ISRC1 Packet Content PB7 - PB10.

8.1.3.34 DI_packet_isrc1_4 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	isrc1_pb13	RW	0	IRSC1 packet PB13
15:8	isrc1_pb12	RW	0	ISRC1 packet PB12
7:0	isrc1_pb11	RW	0	ISRC1 packet PB11

Table 8-39. HDMI Controller Data Island ISRC1 Packet Content PB11 - PB13.

8.1.3.35 DI_packet_isrc1_5 Register

Bits	Name	Attr	Reset	Description
31:24	isrc1_pb17	RW	0	IRSC1 packet PB17
23:16	isrc1_pb16	RW	0	IRSC1 packet PB16
15:8	isrc1_pb15	RW	0	ISRC1 packet PB15
7:0	isrc1_pb14	RW	0	ISRC1 packet PB14

Table 8-40. HDMI Controller Data Island ISRC1 Packet Content PB14 - PB17.

8.1.3.36 DI_packet_isrc1_6 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	isrc1_pb20	RW	0	IRSC1 packet PB20
15:8	isrc1_pb19	RW	0	ISRC1 packet PB19
7:0	isrc1_pb18	RW	0	ISRC1 packet PB18

Table 8-41. HDMI Controller Data Island ISRC1 Packet Content PB18 - PB20.

8.1.3.37 DI_packet_isrc1_7 Register

Bits	Name	Attr	Reset	Description
31:24	isrc1_pb24	RW	0	IRSC1 packet PB24
23:16	isrc1_pb23	RW	0	IRSC1 packet PB23
15:8	isrc1_pb22	RW	0	ISRC1 packet PB22
7:0	isrc1_pb21	RW	0	ISRC1 packet PB21

Table 8-42. HDMI Controller Data Island ISRC1 Packet Content PB21 - PB24.

8.1.3.38 DI_packet_isrc1_8 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	isrc1_pb27	RW	0	IRSC1 packet PB27
15:8	isrc1_pb26	RW	0	ISRC1 packet PB26
7:0	isrc1_pb25	RW	0	ISRC1 packet PB25

Table 8-43. HDMI Controller Data Island ISRC1 Packet Content PB25 - PB27.

8.1.3.39 DI_packet_isrc2_0 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	isrc2_hb2	RW	0	IRSC2 packet header byte2
15:8	isrc2_hb1	RW	0	ISRC2 packet header byte1
7:0	isrc2_hb0	RW	0	ISRC2 packet header byte0

Table 8-44. HDMI Controller Data Island ISRC2 Packet Header.

8.1.3.40 DI_packet_isrc2_1 Register

Bits	Name	Attr	Reset	Description
31:24	isrc2_pb3	RW	0	IRSC2 packet PB3
23:16	isrc2_pb2	RW	0	IRSC2 packet PB2
15:8	isrc2_pb1	RW	0	ISRC2 packet PB1
7:0	isrc2_pb0	RW	0	ISRC2 packet PB0

Table 8-45. HDMI Controller Data Island ISRC2 Packet Content PB0 - PB3.

8.1.3.41 DI_packet_isrc2_2 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	isrc2_pb6	RW	0	IRSC2 packet PB6
15:8	isrc2_pb5	RW	0	ISRC2 packet PB5
7:0	isrc2_pb4	RW	0	ISRC2 packet PB4

Table 8-46. HDMI Controller Data Island ISRC2 Packet Content PB4 - PB6.

8.1.3.42 DI_packet_isrc2_3 Register

Bits	Name	Attr	Reset	Description
31:24	isrc2_pb10	RW	0	IRSC2 packet PB10
23:16	isrc2_pb9	RW	0	IRSC2 packet PB9
15:8	isrc2_pb8	RW	0	ISRC2 packet PB8
7:0	isrc2_pb7	RW	0	ISRC2 packet PB7

Table 8-47. HDMI Controller Data Island ISRC2 Packet Content PB7 - PB10.

8.1.3.43 DI_packet_isrc2_4 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	isrc2_pb13	RW	0	IRSC2 packet PB13
15:8	isrc2_pb12	RW	0	ISRC2 packet PB12
7:0	isrc2_pb11	RW	0	ISRC2 packet PB11

Table 8-48. HDMI Controller Data Island ISRC2 Packet Content PB11 - PB13.

8.1.3.44 DI_packet_isrc2_5 Register

Bits	Name	Attr	Reset	Description
31:24	isrc2_pb17	RW	0	IRSC2 packet PB17
23:16	isrc2_pb16	RW	0	IRSC2 packet PB16
15:8	isrc2_pb15	RW	0	ISRC2 packet PB15
7:0	isrc2_pb14	RW	0	ISRC2 packet PB14

Table 8-49. HDMI Controller Data Island ISRC2 Packet Content PB14 - PB17.

8.1.3.45 DI_packet_isrc2_6 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved

Bits	Name	Attr	Reset	Description
23:16	isrc2_pb20	RW	0	IRSC2 packet PB20
15:8	isrc2_pb19	RW	0	ISRC2 packet PB19
7:0	isrc2_pb18	RW	0	ISRC2 packet PB18

Table 8-50. HDMI Controller Data Island ISRC2 Packet Content PB18 - PB20.

8.1.3.46 DI_packet_isrc2_7 Register

Bits	Name	Attr	Reset	Description
31:24	isrc2_pb24	RW	0	IRSC2 packet PB24
23:16	isrc2_pb23	RW	0	IRSC2 packet PB23
15:8	isrc2_pb22	RW	0	ISRC2 packet PB22
7:0	isrc2_pb21	RW	0	ISRC2 packet PB21

Table 8-51. HDMI Controller Data Island ISRC2 Packet Content PB21 - PB24.

8.1.3.47 DI_packet_isrc2_8 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	isrc2_pb27	RW	0	IRSC2 packet PB27
15:8	isrc2_pb26	RW	0	ISRC2 packet PB26
7:0	isrc2_pb25	RW	0	ISRC2 packet PB25

Table 8-52. HDMI Controller Data Island ISRC2 Packet Content PB25 - PB27.

8.1.3.48 DI_packet_avi0 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	avi_hb2	RW	0	AVI packet header byte2
15:8	avi_hb1	RW	0	AVI packet header byte1
7:0	avi_hb0	RW	0	AVI packet header byte0

Table 8-53. HDMI Controller Data Island AVI0 Packet Header.

8.1.3.49 DI_packet_avi1 Register

Bits	Name	Attr	Reset	Description
31:24	avi_pb3	RW	0	AVI packet PB3
23:16	avi_pb2	RW	0	AVI packet PB2
15:8	avi_pb1	RW	0	AVI packet PB1

Bits	Name	Attr	Reset	Description
7:0	avi_pb0	RW	0	AVI packet PB0

Table 8-54. HDMI Controller Data Island AVI Packet Content PB0 - PB3.

8.1.3.50 DI_packet_avi2 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	avi_pb6	RW	0	AVI packet PB6
15:8	avi_pb5	RW	0	AVI packet PB5
7:0	avi_pb4	RW	0	AVI packet PB4

Table 8-55. HDMI Controller Data Island AVI Packet Content PB4 - PB6.

8.1.3.51 DI_packet_avi3 Register

Bits	Name	Attr	Reset	Description
31:24	avi_pb10	RW	0	AVI packet PB10
23:16	avi_pb9	RW	0	AVI packet PB9
15:8	avi_pb8	RW	0	AVI packet PB8
7:0	avi_pb7	RW	0	AVI packet PB7

Table 8-56. HDMI Controller Data Island AVI Packet Content PB7 - PB10.

8.1.3.52 DI_packet_avi4 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	avi_pb13	RW	0	AVI packet PB13
15:8	avi_pb12	RW	0	AVI packet PB12
7:0	avi_pb11	RW	0	AVI packet PB11

Table 8-57. HDMI Controller Data Island AVI Packet Content PB11 - PB13.

8.1.3.53 DI_packet_avi5 Register

Bits	Name	Attr	Reset	Description
31:24	avi_pb17	RW	0	AVI packet PB17
23:16	avi_pb16	RW	0	AVI packet PB16
15:8	avi_pb15	RW	0	AVI packet PB15
7:0	avi_pb14	RW	0	AVI packet PB14

Table 8-58. HDMI Controller Data Island AVI Packet Content PB14 - PB17.

8.1.3.54 DI_packet_avi6 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	avi_pb20	RW	0	AVI packet PB20
15:8	avi_pb19	RW	0	AVI packet PB19
7:0	avi_pb18	RW	0	AVI packet PB18

Table 8-59. HDMI Controller Data Island AVI Packet Content PB18 - PB20.

8.1.3.55 DI_packet_avi7 Register

Bits	Name	Attr	Reset	Description
31:24	avi_pb24	RW	0	AVI packet PB24
23:16	avi_pb23	RW	0	AVI packet PB23
15:8	avi_pb22	RW	0	AVI packet PB22
7:0	avi_pb21	RW	0	AVI packet PB21

Table 8-60. HDMI Controller Data Island AVI Packet Content PB21 - PB24.

8.1.3.56 DI_packet_avi8 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	avi_pb27	RW	0	AVI packet PB27
15:8	avi_pb26	RW	0	AVI packet PB26
7:0	avi_pb25	RW	0	AVI packet PB25

Table 8-61. HDMI Controller Data Island AVI Packet Content PB25 - PB27.

8.1.3.57 DI_packet_spd0 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	spd_hb2	RW	0	SPD packet header byte2
15:8	spd_hb1	RW	0	SPD packet header byte1
7:0	spd_hb0	RW	0	SPD packet header byte0

Table 8-62. SPD Packet Header.

8.1.3.58 DI_packet_spd1 Register

Bits	Name	Attr	Reset	Description
31:24	spd_pb3	RW	0	SPD packet PB3

Bits	Name	Attr	Reset	Description
23:16	spd_pb2	RW	0	SPD packet PB2
15:8	spd_pb1	RW	0	SPD packet PB1
7:0	spd_pb0	RW	0	SPD packet PB0

Table 8-63. HDMI Controller Data Island SPD Packet Content PB0 - PB3.

8.1.3.59 DI_packet_spd2 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	spd_pb6	RW	0	SPD packet PB6
15:8	spd_pb5	RW	0	SPD packet PB5
7:0	spd_pb4	RW	0	SPD packet PB4

Table 8-64. HDMI Controller Data Island SPD Packet Content PB4 - PB6.

8.1.3.60 DI_packet_spd3 Register

Bits	Name	Attr	Reset	Description
31:24	spd_pb10	RW	0	SPD packet PB10
23:16	spd_pb9	RW	0	SPD packet PB9
15:8	spd_pb8	RW	0	SPD packet PB8
7:0	spd_pb7	RW	0	SPD packet PB7

Table 8-65. HDMI Controller Data Island SPD Packet Content PB7 - PB10.

8.1.3.61 DI_packet_spd4 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	spd_pb13	RW	0	SPD packet PB13
15:8	spd_pb12	RW	0	SPD packet PB12
7:0	spd_pb11	RW	0	SPD packet PB11

Table 8-66. HDMI Controller Data Island SPD Packet Content PB11 - PB13.

8.1.3.62 DI_packet_spd5 Register

Bits	Name	Attr	Reset	Description
31:24	spd_pb17	RW	0	SPD packet PB17
23:16	spd_pb16	RW	0	SPD packet PB16
15:8	spd_pb15	RW	0	SPD packet PB15

Bits	Name	Attr	Reset	Description
7:0	spd_pb14	RW	0	SPD packet PB14

Table 8-67. HDMI Controller Data Island SPD Packet Content PB14 - PB17.

8.1.3.63 DI_packet_spd6 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	spd_pb20	RW	0	SPD packet PB20
15:8	spd_pb19	RW	0	SPD packet PB19
7:0	spd_pb18	RW	0	SPD packet PB18

Table 8-68. HDMI Controller Data Island SPD Packet Content PB18 - PB20.

8.1.3.64 DI_packet_spd7 Register

Bits	Name	Attr	Reset	Description
31:24	spd_pb24	RW	0	SPD packet PB24
23:16	spd_pb23	RW	0	SPD packet PB23
15:8	spd_pb22	RW	0	SPD packet PB22
7:0	spd_pb21	RW	0	SPD packet PB21

Table 8-69. HDMI Controller Data Island SPD Packet Content PB21 - PB24.

8.1.3.65 DI_packet_spd8 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	spd_pb27	RW	0	SPD packet PB27
15:8	spd_pb26	RW	0	SPD packet PB26
7:0	spd_pb25	RW	0	SPD packet PB25

Table 8-70. HDMI Controller Data Island SPD Packet Content PB25 - PB27.

8.1.3.66 DI_packet_audio0 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	audio_hb2	RW	0	AUDIO packet header byte2
15:8	audio_hb1	RW	0	AUDIO packet header byte1
7:0	audio_hb0	RW	0	AUDIO packet header byte0

Table 8-71. Audio Packet Header Register.

8.1.3.67 DI_packet_audio1 Register

Bits	Name	Attr	Reset	Description
31:24	audio_pb3	RW	0	AUDIO packet PB3
23:16	audio_pb2	RW	0	AUDIO packet PB2
15:8	audio_pb1	RW	0	AUDIO packet PB1
7:0	audio_pb0	RW	0	AUDIO packet PB0

Table 8-72. HDMI Controller Data Island Audio Packet Content PB0 - PB3.

8.1.3.68 DI_packet_audio2 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	audio_pb6	RW	0	AUDIO packet PB6
15:8	audio_pb5	RW	0	AUDIO packet PB5
7:0	audio_pb4	RW	0	AUDIO packet PB4

Table 8-73. HDMI Controller Data Island Audio Packet Content PB4 - PB6.

8.1.3.69 DI_packet_audio3 Register

Bits	Name	Attr	Reset	Description
31:24	audio_pb10	RW	0	AUDIO packet PB10
23:16	audio_pb9	RW	0	AUDIO packet PB9
15:8	audio_pb8	RW	0	AUDIO packet PB8
7:0	audio_pb7	RW	0	AUDIO packet PB7

Table 8-74. HDMI Controller Data Island Audio Packet Content PB7 - PB10.

8.1.3.70 DI_packet_audio4 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	audio_pb13	RW	0	AUDIO packet PB13
15:8	audio_pb12	RW	0	AUDIO packet PB12
7:0	audio_pb11	RW	0	AUDIO packet PB11

Table 8-75. HDMI Controller Data Island Audio Packet Content PB11 - PB13.

8.1.3.71 DI_packet_audio5 Register

Bits	Name	Attr	Reset	Description
31:24	audio_pb17	RW	0	AUDIO packet PB17

Bits	Name	Attr	Reset	Description
23:16	audio_pb16	RW	0	AUDIO packet PB16
15:8	audio_pb15	RW	0	AUDIO packet PB15
7:0	audio_pb14	RW	0	AUDIO packet PB14

Table 8-76. HDMI Controller Data Island Audio Packet Content PB14 - PB17.

8.1.3.72 DI_packet_audio6 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	audio_pb20	RW	0	AUDIO packet PB20
15:8	audio_pb19	RW	0	AUDIO packet PB19
7:0	audio_pb18	RW	0	AUDIO packet PB18

Table 8-77. HDMI Controller Data Island Audio Packet Content PB18 - PB20.

8.1.3.73 DI_packet_audio7 Register

Bits	Name	Attr	Reset	Description
31:24	audio_pb24	RW	0	AUDIO packet PB24
23:16	audio_pb23	RW	0	AUDIO packet PB23
15:8	audio_pb22	RW	0	AUDIO packet PB22
7:0	audio_pb21	RW	0	AUDIO packet PB21

Table 8-78. HDMI Controller Data Island Audio Packet Content PB21 - PB24.

8.1.3.74 DI_packet_audio8 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	audio_pb27	RW	0	AUDIO packet PB27
15:8	audio_pb26	RW	0	AUDIO packet PB26
7:0	audio_pb25	RW	0	AUDIO packet PB25

Table 8-79. HDMI Controller Data Island Audio Packet Content PB25 - PB27.

8.1.3.75 DI_packet_mpeg0 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	mpeg_hb2	RW	0	MPEG packet header byte2
15:8	mpeg_hb1	RW	0	MPEG packet header byte1

Bits	Name	Attr	Reset	Description
7:0	mpeg_hb0	RW	0	MPEG packet header byte0

Table 8-80. MPEG Packet Header.

8.1.3.76 DI_packet_mpeg1 Register

Bits	Name	Attr	Reset	Description
31:24	mpeg_pb3	RW	0	MPEG packet PB3
23:16	mpeg_pb2	RW	0	MPEG packet PB2
15:8	mpeg_pb1	RW	0	MPEG packet PB1
7:0	mpeg_pb0	RW	0	MPEG packet PB0

Table 8-81. HDMI Controller Data Island MPEG Packet Content PB0 - PB3.

8.1.3.77 DI_packet_mpeg2 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	mpeg_pb6	RW	0	MPEG packet PB6
15:8	mpeg_pb5	RW	0	MPEG packet PB5
7:0	mpeg_pb4	RW	0	MPEG packet PB4

Table 8-82. HDMI Controller Data Island MPEG Packet Content PB4 - PB6.

8.1.3.78 DI_packet_mpeg3 Register

Bits	Name	Attr	Reset	Description
31:24	mpeg_pb10	RW	0	MPEG packet PB10
23:16	mpeg_pb9	RW	0	MPEG packet PB9
15:8	mpeg_pb8	RW	0	MPEG packet PB8
7:0	mpeg_pb7	RW	0	MPEG packet PB7

Table 8-83. HDMI Controller Data Island MPEG Packet Content PB7 - PB10.

8.1.3.79 DI_packet_mpeg4 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	mpeg_pb13	RW	0	MPEG packet PB13
15:8	mpeg_pb12	RW	0	MPEG packet PB12
7:0	mpeg_pb11	RW	0	MPEG packet PB11

Table 8-84. HDMI Controller Data Island MPEG Packet Content PB11 - PB13.

8.1.3.80 DI_packet_mpeg5 Register

Bits	Name	Attr	Reset	Description
31:24	mpeg_pb17	RW	0	MPEG packet PB17
23:16	mpeg_pb16	RW	0	MPEG packet PB16
15:8	mpeg_pb15	RW	0	MPEG packet PB15
7:0	mpeg_pb14	RW	0	MPEG packet PB14

Table 8-85. HDMI Controller Data Island MPEG Packet Content PB14 - PB17.

8.1.3.81 DI_packet_mpeg6 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	mpeg_pb20	RW	0	MPEG packet PB20
15:8	mpeg_pb19	RW	0	MPEG packet PB19
7:0	mpeg_pb18	RW	0	MPEG packet PB18

Table 8-86. HDMI Controller Data Island MPEG Packet Content PB18 - PB20.

8.1.3.82 DI_packet_mpeg7 Register

Bits	Name	Attr	Reset	Description
31:24	mpeg_pb24	RW	0	MPEG packet PB24
23:16	mpeg_pb23	RW	0	MPEG packet PB23
15:8	mpeg_pb22	RW	0	MPEG packet PB22
7:0	mpeg_pb21	RW	0	MPEG packet PB21

Table 8-87. HDMI Controller Data Island MPEG Packet Content PB21 - PB24.

8.1.3.83 DI_packet_mpeg8 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	mpeg_pb27	RW	0	MPEG packet PB27
15:8	mpeg_pb26	RW	0	MPEG packet PB26
7:0	mpeg_pb25	RW	0	MPEG packet PB25

Table 8-88. HDMI Controller Data Island MPEG Packet Content PB25 - PB27.

8.1.3.84 DI_packet_gamut0 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved

Bits	Name	Attr	Reset	Description
23:16	gamut_hb2	RW	0	Gamut packet header byte2
15:8	gamut_hb1	RW	0	Gamut packet header byte1
7:0	gamut_hb0	RW	0	Gamut packet header byte0

Table 8-89. Gamut Packet Header.

8.1.3.85 DI_packet_gamut1 Register

Bits	Name	Attr	Reset	Description
31:24	gamut_pb3	RW	0	Gamut packet PB3
23:16	gamut_pb2	RW	0	Gamut packet PB2
15:8	gamut_pb1	RW	0	Gamut packet PB1
7:0	gamut_pb0	RW	0	Gamut packet PB0

Table 8-90. HDMI Controller Data Island Gamut Packet Content PB0 - PB3.

8.1.3.86 DI_packet_gamut2 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	gamut_pb6	RW	0	Gamut packet PB6
15:8	gamut_pb5	RW	0	Gamut packet PB5
7:0	gamut_pb4	RW	0	Gamut packet PB4

Table 8-91. HDMI Controller Data Island Gamut Packet Content PB4 - PB6.

8.1.3.87 DI_packet_gamut3 Register

Bits	Name	Attr	Reset	Description
31:24	gamut_pb10	RW	0	Gamut packet PB10
23:16	gamut_pb9	RW	0	Gamut packet PB9
15:8	gamut_pb8	RW	0	Gamut packet PB8
7:0	gamut_pb7	RW	0	Gamut packet PB7

Table 8-92. HDMI Controller Data Island Gamut Packet Content PB7 - PB10.

8.1.3.88 DI_packet_gamut4 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	gamut_pb13	RW	0	Gamut packet PB13
15:8	gamut_pb12	RW	0	Gamut packet PB12

Bits	Name	Attr	Reset	Description
7:0	gamut_pb11	RW	0	Gamut packet PB11

Table 8-93. HDMI Controller Data Island Gamut Packet Content PB11 - PB13.

8.1.3.89 DI_packet_gamut5 Register

Bits	Name	Attr	Reset	Description
31:24	gamut_pb17	RW	0	Gamut packet PB17
23:16	gamut_pb16	RW	0	Gamut packet PB16
15:8	gamut_pb15	RW	0	Gamut packet PB15
7:0	gamut_pb14	RW	0	Gamut packet PB14

Table 8-94. HDMI Controller Data Island Gamut Packet Content PB14 - PB17.

8.1.3.90 DI_packet_gamut6 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	gamut_pb20	RW	0	Gamut packet PB20
15:8	gamut_pb19	RW	0	Gamut packet PB19
7:0	gamut_pb18	RW	0	Gamut packet PB18

Table 8-95. HDMI Controller Data Island Gamut Packet Content PB18 - PB20.

8.1.3.91 DI_packet_gamut7 Register

Bits	Name	Attr	Reset	Description
31:24	gamut_pb24	RW	0	Gamut packet PB24
23:16	gamut_pb23	RW	0	Gamut packet PB23
15:8	gamut_pb22	RW	0	Gamut packet PB22
7:0	gamut_pb21	RW	0	Gamut packet PB21

Table 8-96. HDMI Controller Data Island Gamut Packet Content PB21 - PB24.

8.1.3.92 DI_packet_gamut8 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	gamut_pb27	RW	0	Gamut packet PB27
15:8	gamut_pb26	RW	0	Gamut packet PB26
7:0	gamut_pb25	RW	0	Gamut packet PB25

Table 8-97. HDMI Controller Data Island Gamut Packet Content PB25 - PB27.

8.1.3.93 DI_packet_vendor0 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	vendor_hb2	RW	0	Vendor packet header byte2
15:8	vendor_hb1	RW	0	Vendor packet header byte1
7:0	vendor_hb0	RW	0	Vendor packet header byte0

Table 8-98. Vendor Packet Header.

8.1.3.94 DI_packet_vendor1 Register

Bits	Name	Attr	Reset	Description
31:24	vendor_pb3	RW	0	Vendor packet PB3
23:16	vendor_pb2	RW	0	Vendor packet PB2
15:8	vendor_pb1	RW	0	Vendor packet PB1
7:0	vendor_pb0	RW	0	Vendor packet PB0

Table 8-99. HDMI Controller Data Island Vendor Packet Content PB0 - PB3.

8.1.3.95 DI_packet_vendor2 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	vendor_pb6	RW	0	Vendor packet PB6
15:8	vendor_pb5	RW	0	Vendor packet PB5
7:0	vendor_pb4	RW	0	Vendor packet PB4

Table 8-100. HDMI Controller Data Island Vendor Packet Content PB4 - PB6.

8.1.3.96 DI_packet_vendor3 Register

Bits	Name	Attr	Reset	Description
31:24	vendor_pb10	RW	0	Vendor packet PB10
23:16	vendor_pb9	RW	0	Vendor packet PB9
15:8	vendor_pb8	RW	0	Vendor packet PB8
7:0	vendor_pb7	RW	0	Vendor packet PB7

Table 8-101. HDMI Controller Data Island Vendor Packet Content PB7 - PB10.

8.1.3.97 DI_packet_vendor4 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved

Bits	Name	Attr	Reset	Description
23:16	vendor_pb13	RW	0	Vendor packet PB13
15:8	vendor_pb12	RW	0	Vendor packet PB12
7:0	vendor_pb11	RW	0	Vendor packet PB11

Table 8-102. HDMI Controller Data Island Vendor Packet Content PB11 - PB13.

8.1.3.98 DI_packet_vendor5 Register

Bits	Name	Attr	Reset	Description
31:24	vendor_pb17	RW	0	Vendor packet PB17
23:16	vendor_pb16	RW	0	Vendor packet PB16
15:8	vendor_pb15	RW	0	Vendor packet PB15
7:0	vendor_pb14	RW	0	Vendor packet PB14

Table 8-103. HDMI Controller Data Island Vendor Packet Content PB14 - PB17.

8.1.3.99 DI_packet_vendor6 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	vendor_pb20	RW	0	Vendor packet PB20
15:8	vendor_pb19	RW	0	Vendor packet PB19
7:0	vendor_pb18	RW	0	Vendor packet PB18

Table 8-104. HDMI Controller Data Island Vendor Packet Content PB18 - PB20.

8.1.3.100 DI_packet_vendor7 Register

Bits	Name	Attr	Reset	Description
31:24	vendor_pb24	RW	0	Vendor packet PB24
23:16	vendor_pb23	RW	0	Vendor packet PB23
15:8	vendor_pb22	RW	0	Vendor packet PB22
7:0	vendor_pb21	RW	0	Vendor packet PB21

Table 8-105. HDMI Controller Data Island Vendor Packet Content PB21 - PB24.

8.1.3.101 DI_packet_vendor8 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	vendor_pb27	RW	0	Vendor packet PB27
15:8	vendor_pb26	RW	0	Vendor packet PB26

Bits	Name	Attr	Reset	Description
7:0	vendor_pb25	RW	0	Vendor packet PB25

Table 8-106. HDMI Controller Data Island Vendor Packet Content PB25 - PB27.

8.1.3.102 DI_i2s_mode Register

Bits	Name	Attr	Reset	Description
31:3				Reserved
2:0	dai_mode	RW	0x4	0x0 - Left-justified mode 0x1 - Right-justified mode 0x2 - MSB extended mode 0x4 - I2S mode 0x6 - Reserved

Table 8-107. HDMI Controller Data Island I2S Operation Mode Selection Register.

8.1.3.103 DI_i2s_rx_ctrl Register

Bits	Name	Attr	Reset	Description
31:3				Reserved
2	rx_ord	RW	0	Receiver bit order 0 - MSB 1 - LSB
1	rx_ws_mst	RW	0	Receiver mode 0 - Slave 1 - Master
0	rx_ws_inv	RW	0	Receiver word select invert. 0 - Receive first data on I2S word select signal I2S_WS = 0 1 - Receive first data on I2S word select signal I2S_WS = 1

Table 8-108. HDMI Controller Data Island I2S Rx Control Register.

8.1.3.104 DI_i2s_wlen Register

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	dai_wlen	RW	0x17	Word precision - 1

Table 8-109. HDMI Controller Data Island I2S Word Length Register.

8.1.3.105 DI_i2s_wpos Register

Bits	Name	Attr	Reset	Description
31:5				Reserved

Bits	Name	Attr	Reset	Description
4:0	dai_wpos	RW	0	Number of ignored bits between ws rising and falling edges

Table 8-110. HDMI Controller Data Island I2S Word Position Register.

8.1.3.106 DI_i2s_slot Register

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	dai_slot	RW	0	Slot count register

Table 8-111. HDMI Controller Data Island I2S Slot Count Register.

8.1.3.107 DI_i2s_rx_fifo_gth Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	rx_fifo_gth	RW	0x3	Received FIFO threshold

Table 8-112. HDMI Controller Data Island I2S Receiver FIFO Threshold Register.

8.1.3.108 DI_i2s_clock Register

This register informs the HDMI Tx that incoming audio data is valid on the rising edge or negative edge of the I2S bit clock.

Bits	Name	Attr	Reset	Description
31:6				Reserved
5	rx_scp	RW	0	Receiver I2S_CLK polarity 0 - Sample I2S data on rising edge 1 - Sample I2S data on falling edge
4:0				Reserved

Table 8-113. HDMI Controller Data Island I2S Clock and Word Select Register.

8.1.3.109 DI_i2s_init Register

Bits	Name	Attr	Reset	Description
31:2				Reserved
1	rx_enable	RW	0	Receiver enable
0	dai_reset	RW	0	FIFO reset; write 1 to reset FIFO.

Table 8-114. HDMI Controller Data Island I2S Enable and Reset Register

8.1.3.110 DI_i2s_rx_data0 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:0	rx_fifo_dout0	R	0	Rx FIFO data 0

Table 8-115. HDMI Controller Data Island I2S0 FIFO Rx Data Register.

8.1.3.111 DI_i2s_rx_data1 Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:0	rx_fifo_dout1	R	0	Rx FIFO data 1

Table 8-116. HDMI Controller Data Island I2S1 FIFO Rx Data Register.

8.1.3.112 DI_rx_i2s_fifo_cntr Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	rx_fifo_cntr	R	0	Receiver FIFO data count

Table 8-117. HDMI Controller Data Island I2S receiver FIFO Data Counter Register.

8.1.3.113 DI_i2s_gate_off Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	gate_off_en	RW	0	I2S Rx gate off enable 0 - Disable 1 - Enable

Table 8-118. HDMI Controller Data Island I2S Gate Off Register.

8.1.3.114 DI_packet_misc Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7	aud_deliver	RW	0	Audio Packet Deliver Mode 0 - Receive audio packets during one line then send received audio packets during next line 1 - Send audio packets as soon as possible
6	hbr	RW	0	HBR (High Bit-Rate) Audio Mode 0 - L-PCM mode 1 - HBR mode

Bits	Name	Attr	Reset	Description
5	i2s_rx_mode	RW	0	I2S Internal Enable Setting 0 - I2S rx_enable is internally enabled combining rx_enable bits setting and HDMI internal state machine starts to work 1 - I2S rx_enable is on or off as programmed
4	ncts_priority	RW	0	NCTS Priority Compared to Audio Packet 0 - NCTS has lower priority 1 - NCTS has higher priority
3	cts_sw_mode	RW	0	CTS Software Mode 0 - When cts_sel = 1, CTS value will keep as programmed 1 - When cts_sel = 1, programmed CTS value will be sent once then in the following NCTS packets, CTS = 0 will be sent.
2	spd_send_ctrl	RW	0	SPD Infoframe Send Control 0 - Send once within one field of interlaced video mode. 1 - Send once within one frame of interlaced video mode.
1	right_valid_bit	RW	0	Right Channel Valid Bit of Audio Packet
0	left_valid_bit	RW	0	Left Channel Valid Bit of Audio Packet

Table 8-119. HDMI Controller Data Island Control Options Register.

Registers from VUNIT_vblank_left to VUNIT_hactive are listed below and their functions are illustrated in the following figure.

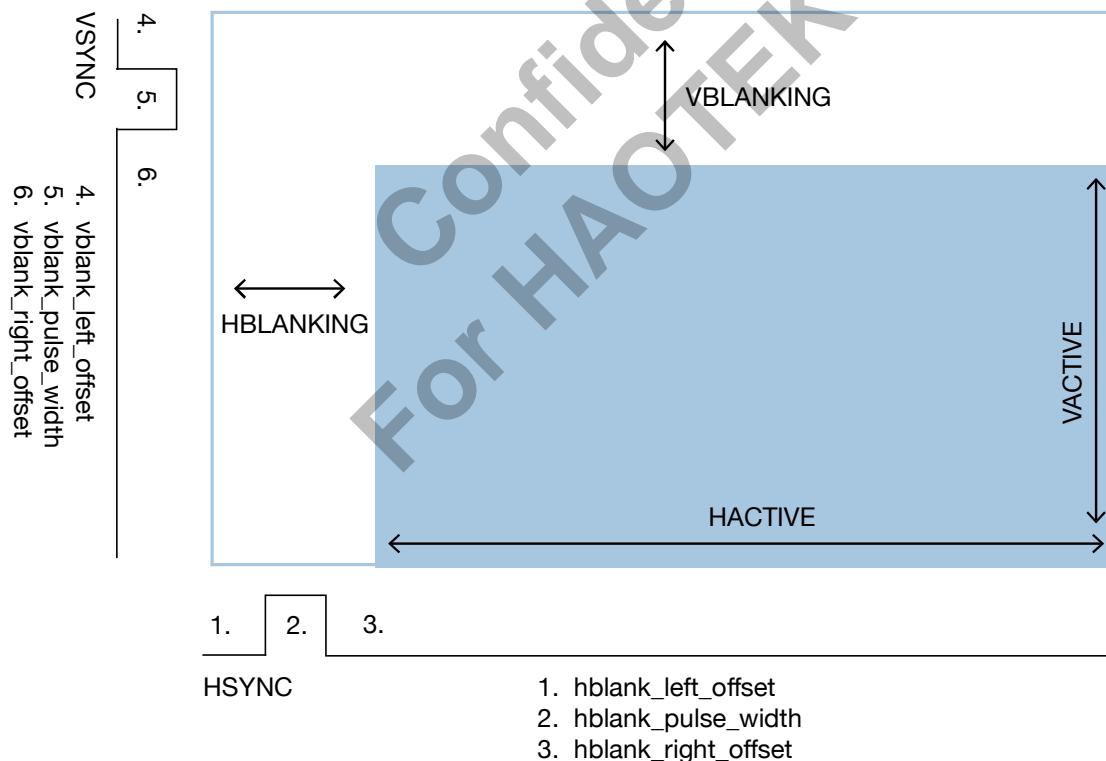


Figure 8-3. HDMI Controller VUNIT Video Frame Format.

8.1.3.115 VUNIT_vblank_left Register

Bits	Name	Attr	Reset	Description
31:14				Reserved
13:0	vblank_left_offset	RW	0	Bits before VSync is active

Table 8-120. HDMI Controller VUNIT Register for the Front Porch Size of the Vertical Sync Signal.

For interlaced video format, fill the parameters of field 0.

8.1.3.116 VUNIT_vblank_pulse Register

Bits	Name	Attr	Reset	Description
31:14				Reserved
13:0	vblank_pulse_width	RW	0	Bits while VSync is active

Table 8-121. HDMI Controller VUNIT Register for the Pulse Width of the Vertical Sync Signal.

For interlaced video format, fill the parameters of field 0.

8.1.3.117 VUNIT_vblank_right Register

Bits	Name	Attr	Reset	Description
31:14				Reserved
13:0	vblank_right_offset	RW	0	Bits after VSync is active

Table 8-122. HDMI Controller VUNIT Register for the Back Porch Size of the Vertical Sync Signal.

For interlaced video format, fill the parameters of field 0.

8.1.3.118 VUNIT_hblank_left Register

Bits	Name	Attr	Reset	Description
31:14				Reserved
13:0	hblank_left_offset	RW	0	Bits before HSync is active

Table 8-123. HDMI Controller VUNIT Register for the Front Porch Size of the Horizontal Sync Signal.

For interlaced video format, fill the parameters of field 0.

8.1.3.119 VUNIT_hblank_pulse Register

Bits	Name	Attr	Reset	Description
31:14				Reserved
13:0	hblank_pulse_width	RW	0	Bits f while HSync is active

Table 8-124. HDMI Controller VUNIT Register for the Pulse Width of the Horizontal Sync Signal.

For interlaced video format, fill the parameters of field 0.

8.1.3.120 VUNIT_hblank_right Register

Bits	Name	Attr	Reset	Description
31:14				Reserved
13:0	hblank_right_offset	RW	0	Bits after HSync is active

Table 8-125. HDMI Controller VUNIT Register for the Back Porch of the Vertical Sync Signal.

For interlaced video format, fill the parameters of field 0.

8.1.3.121 VUNIT_vactive Register

Bits	Name	Attr	Reset	Description
31:14				Reserved
13:0	vunit_vactive	RW	0	Active lines of one frame

Table 8-126. HDMI Controller VUNIT Register for the Number of Active Lines in One Frame.

For interlaced video format, fill the parameters of field 0.

8.1.3.122 VUNIT_hactive Register

Bits	Name	Attr	Reset	Description
31:15				Reserved
14:0	vunit_hactive	RW	0	Active pixels of one line

Table 8-127. HDMI Controller VUNIT Register for the Number of Active Pixels in One Line.

For interlaced video format, fill the parameters of field 0.

8.1.3.123 VUNIT_ctrl Register

Bits	Name	Attr	Reset	Description
31:4				Reserved
3:2	video_mode	RW	0	Input Video Mode 0x0 - Progressive 3D frame packing 3D line alternative 3D side-by-side (full) 3D side-by-side (half) 0x1 - Interlace 0x2 - 3D field alternative mode 0x3 - Reserved
1	hsync_pol	RW	0	HSync Polarity 0 - Positive Polarity 1 - Negative Polarity

Bits	Name	Attr	Reset	Description
0	vsync_pol	RW	0	VSync Polarity 0 - Positive Polarity 1 - Negative Polarity

Table 8-128. HDMI Controller VUNIT Control Register.

8.1.3.124 VUNIT_vsync_detect Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	vsync_detect_en	RW	1	VSync detect enable 0 - Disable VSync detection 1 - Enable VSync detection

Table 8-129. HDMI Controller VUNIT VSync Detect Enable Register.

8.1.3.125 HDMISE_tm Register

Bits	Name	Attr	Reset	Description
31:24	bg_r	RW	0	Red constituent of background color This field setting will take effect only when vdata_src_mode and video_pattern_mode are high.
23:16	bg_g	RW	0	Green constituent of background color This field setting will take effect only when vdata_src_mode and video_pattern_mode are high.
15:8	bg_b	RW	0	Blue constituent of background color This field setting will take effect only when vdata_src_mode and video_pattern_mode are high.
7:4				Reserved
3	adata_src_mode	RW	0	Audio data source selection mode 0 - Audio data is from external source 1 - Audio data is internally generated tone
2	video_pattern_mode	RW	0	Video pattern selection mode. This bit is only valid when vdata_src_mode is high. 0 - Output pattern grades from black to white, 256 pixel clocks repeat once 1 - Output pattern is programmable background color
1	vdata_src_mode	RW	0	Video data source selection mode 0 - Video data from VOUT 1 - Video data is generated by HDMISE (debug purpose)
0	i2s_dout_mode	RW	0	I2S receive data read out mode. 0 - FIFO data will be read out from HDMISE 1 - FIFO data will be read out from AHB bus (debug purpose)

Table 8-130. HDMI Controller HDMISE Test Mode Register.

8.1.3.126 P2P_afifo_level Register

Bits	Name	Attr	Reset	Description
31:28	p2p_afifo_ub	RW	0xC	P2P AFIFO level upper bound
27:24	p2p_afifo_lb	RW	0x4	P2P AFIFO level lower bound
23:21				Reserved
20:16	p2p_afifo_max_level	R	0	P2P AFIFO max usage level
15:13				Reserved
12:8	p2p_afifo_min_level	R	0	P2P AFIFO min usage level
7:5				Reserved
4:0	p2p_afifo_level	R	0	P2P AFIFO current usage level

Table 8-131. HDMI Controller TMDS Asynchronous FIFO PCLK to PHY_CLK AFIFO Usage Level Indicator.

8.1.3.127 P2P_afifo_ctrl Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	p2p_afifo_en	RW	0	P2P AFIFO enable. HDMITX will drive data to PHY only when this bit is set. 0 - Disable 1 - Enable

Table 8-132. HDMI Controller TMDS Asynchronous FIFO PCLK to PHY_CLK AFIFO Control Register.

8.1.3.128 HDMISE_dbg Register

Bits	Name	Attr	Reset	Description
31:13				Reserved
12	dbg_invd_aud	RW	0	Send invalid audio packet when no audio sample to send in blanking region of a line. 0 - Disable 1 - Enable
11				Reserved
10:8	dbg_ch_swp	RW	0	Debug RGB channels swap 0x0 - R'G'B' = RGB 0x1 - R'G'B' = RBG 0x2 - R'G'B' = GRB 0x3 - R'G'B' = GBR 0x4 - R'G'B' = BRG 0x5 - R'G'B' = BGR 0x6 - R'G'B' = Reserved 0x7 - R'G'B' = Reserved Where RBG stands for channels before swap, R'G'B' stands for channels after swap.
7				Reserved

Bits	Name	Attr	Reset	Description
6	dbg_ch_r_rev	RW	0	Debug channel red bits reserved 0 - Bits of R channel not reserved 1 - Bits of R channel reserved
5	dbg_ch_g_rev	RW	0	Debug channel green bits reserved 0 - Bits of G channel not reserved 1 - Bits of G channel reserved
4	dbg_ch_b_rev	RW	0	Debug channel blue bits reserved 0 - Bits of B channel not reserved 1 - Bits of B channel reserved
3				Reserved
2:1	dbg_vdata_src_mode	RW	0	Debug Video Data Source Selection Mode 0x0 - Normal mode. Using VOUT sync signals and video data. 0x1 - Debug mode using internal generated sync signals and video data. 0x2 - Debug mode using VOUT sync signals and internal video data. 0x3 - Reserved.
0	dbg_p2p_afifo_by-pass	RW	0	Debug P2P AFIFO Bypass mode. When this bit is set to 1, data from HDMI TX to HDMI PHY will not go through AFIFO. Only valid when pixel clock and PHY clock have the same clock source. 0 - Disable bypass 1 - Enable bypass

Table 8-133. HDMI Controller HDMISE Debug Register.

8.1.3.129 HDMI_phy_ctrl Register

Bits	Name	Attr	Reset	Description
31:19				Reserved
18:16	pes	RW	2	Adjust Bias Current. Default: 3'b010 000: Iref*(0.50) 001: Iref*(0.75) 010: Iref*(1.00) 011: Iref*(1.25) 100: Iref*(1.50) 101: Iref*(1.75) 110: Iref*(2.00) 111: Iref*(2.25) Where Iref is the internal bias current.
15:11				Reserved
10:8	pib	RW	0	Set the Open Drain Output Current 0x0 - 0 mA added 0x1 - 0.675 mA added 0x2 - 1.35 mA added 0x3 - 2.025 mA added 0x4 - 2.7 mA added 0x5 - 3.375 mA added 0x6 - 4.05 mA added 0x7 - 4.725 mA added

Bits	Name	Attr	Reset	Description
7	pd_bg	RW	0	Band Gap Voltage Generator Power Down Mode 0 - Active mode 1 - Power down mode
6	pdb_hdmi	RW	1	HDMI Power Down Mode 0 - Power down mode 1 - Active mode
5:1				Reserved
0	rstnd_hdmi	RW	1	HDMI PHY Reset 0 - Reset HDMI 1 - Non-reset HDMI

Table 8-134. HDMI Controller PHY Control Registers.

For Confidential
For HAOTEX Only

8.1.3.130 HDMI_phy_ctrl1 Register

Bits	Name	Attr	Reset	Description
31:16				Reserved
15				Double the output current when termination is active. Default to 0.
14				Reserved
13:12				Set termination resistance. Default to 2'b00.
11				Disable third-tap pre-emphasis. Default to 0.
10:8				Set third-tap pre-emphasis current. Default to 3'b000. 000 0% pre-emphasis 001 1.67% pre-emphasis 010 3.33% pre-emphasis 011 5.00% pre-emphasis 100 6.67% pre-emphasis 101 8.33% pre-emphasis 110 10.0% pre-emphasis 111 11.7% pre-emphasis
7				Disable second-tap pre-emphasis. Default to 0.
6:4				Set second-tap pre-emphasis current. Default to 3'b000 000 0% pre-emphasis 001 3.33% pre-emphasis 010 6.67% pre-emphasis 011 10.0% pre-emphasis 100 13.3% pre-emphasis 101 16.7% pre-emphasis 110 20.0% pre-emphasis 111 23.3% pre-emphasis
3				Reserved to 0
2:0				Set first-tap pre-emphasis current. Default to 3'b100. 000 0% pre-emphasis 001 6.67% pre-emphasis 010 10.0% pre-emphasis 011 16.7% pre-emphasis 100 20.0% pre-emphasis 101 26.7% pre-emphasis 110 30.0% pre-emphasis 111 36.7% pre-emphasis

Table 8-135. HDMI Controller PHY Control 1 Registers.

8.1.4 HDMI Controller: Operating Procedures

Once HCLK (GCLK_AHB), the pixel clock, the audio clock (GCLK_AU) and the HDMITX PHY clock (PHY_CLK_VO) are available, deassert the HDMISE software reset to enable the HDMITX. Then enable the asynchronous FIFO to drive encoded data to the HDMITX PHY.

- Configure interrupt enable register **HDMI_int_enable**.

- Set the input video format according to the input source. Refer to the example below.

The following is an example showing the steps required to configure the 1920x1080i video format.

- Set **VUNIT_vblank_left**, **VUNIT_vblank_pulse** and **VUNIT_vblank_right** registers:

- VUNIT_vblank_right** determines how many lines from the VSYNC pulse will end before the Data Enable goes high. In 1920x1080i format, **VUNIT_vblank_right** proceeds from line 6 to line 20 or 15 lines. Use field 0 parameters if the video format is interlaced. Refer to [Figure 8-4](#) and [Figure 8-5](#).
- The **VUNIT_vblank_right**, **VUNIT_vblank_pulse** and **VUNIT_vblank_left** registers must be set as follows (values in decimal):
 - **VUNIT_vblank_right:** 15
 - **VUNIT_vblank_pulse:** 5
 - **VUNIT_vblank_left:** 2

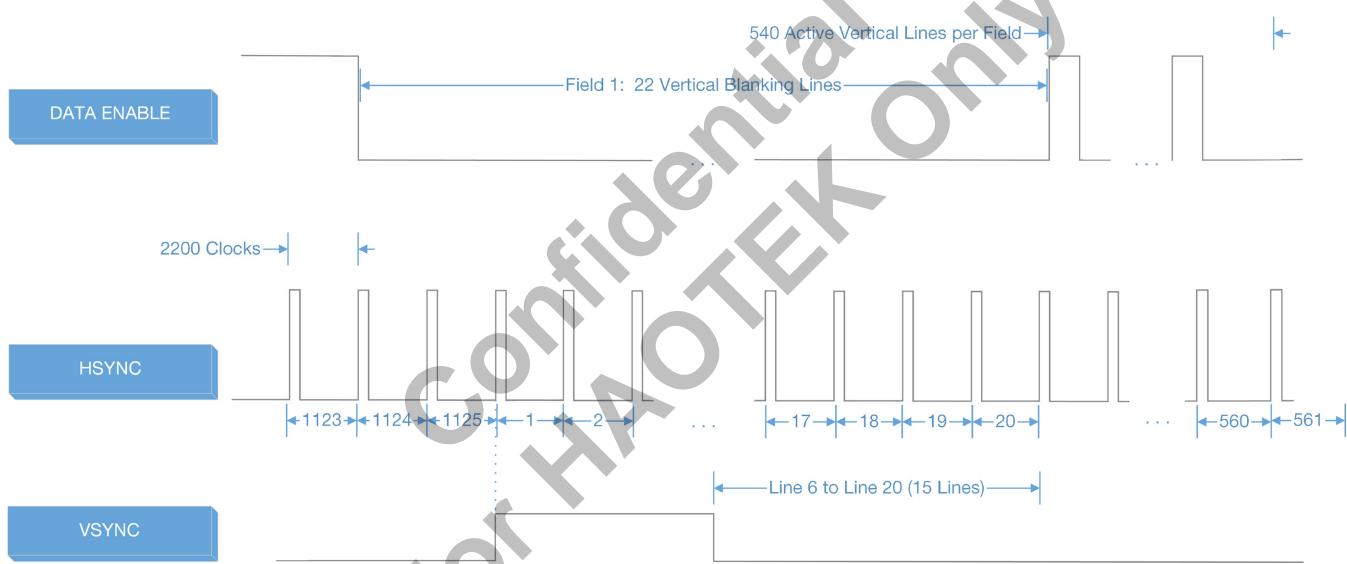


Figure 8-4. Example of HDMI Register Programming: Setting the Input Video Format (1920x1080i) (1 of 3).

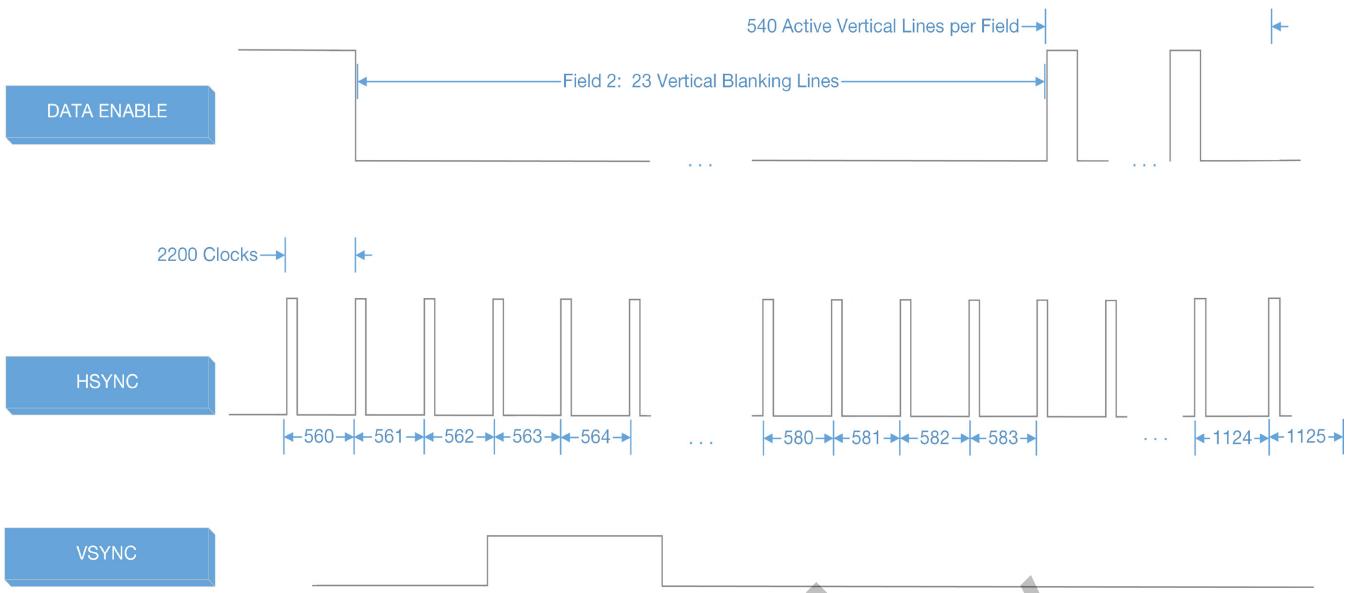


Figure 8-5. Example of HDMI Register Programming: Setting the Input Video Format (1920x1080i) (2 of 3).

2. Set the **VUNIT_hblank_left**, **VUNIT_hblank_pulse** and **VUNIT_hblank_right** registers. Refer to [Figure 8-6](#).
 - The definition for vertical blanking is as follows (values in decimal):
 - **VUNIT_hblank_right**: 148
 - **VUNIT_hblank_pulse**: 44
 - **VUNIT_hblank_left**: 88



Figure 8-6. Example of HDMI Register Programming: Setting the Input Video Format (1920x1080i) (3 of 3).

3. Set the **VUNIT_vactive** register:

- For progressive video, **active lines** is the number of active lines of a frame. For interlaced video, **active lines** is the number of a field. For 1920x1080i, this register is set as 540 (value in decimal).

4. Set the **VUNIT_hactive** register:

- There are 1920 (value in decimal) active pixels to a line.

5. Set the **VUNIT_ctrl** register to define the VSYNC/HSYNC polarity and choose progressive or interlaced video input.

6. Set the audio registers (HDMI mode only):

- Set the **DI_aunit_src** register to define how many I2S channels are used. The A12 family of chips supports one I2S channel.
- Set the I2S registers for capturing input audio data.
- Configure the **DI_aunit_mclk** register to define the ratio of the audio clock to the sampling clock.
- Set the **DI_aunit_n** register to define the N value of NCTS packet.
- Set the **DI_aunit_cts** register if using software mode.
- Set the **DI_aunit_n cts_ctrl** register to enable NCTS packet transmission and select the CTS value mode.

7. Set packet-related registers (HDMI mode only):

- Write contents of all infoframes to be sent into the corresponding registers.
- Once the above step is completed, configure the **DI_packet_tx_ctrl** register to define which infoframe will be sent and to choose to send once or periodically. The **buf_switch_en** bit must be set when the infoframe contents are overwritten. For a specific infoframe, if periodic transmissions are preferred, the **xxx_en** and **xxx_rpt** bits must be set simultaneously. For one time transmission, enable **xxx_en**, which will be cleared automatically when the infoframe is sent. When the new contents are loaded, the **buf_switch_en** will be cleared as well.

8. Configure the operating mode and enable:

- Once all other registers are set, set the **HDMI_op_mode** register to choose DVI or HDMI mode and then enable the HDMITX. The first bit of the **HDMI_op_mode** register defines the current mode as DVI or HDMI. If HDMI is set, the EESS encryption in the HDCPCE must be set or DVI mode will be forced.

9. VIDEO test mode:

- Set bit 1 of the **HDMISE_tm** register to put the HDMITX into video test mode. The output video pattern is a gradient from dark to bright repeated once per 256 cycles.

10. Audio test mode:

- Set bit 0 of the **HDMISE_tm** register to put the HDMITX into audio test mode. The received audio raw data can be read via register **DI_i2s_rx_data0**.

For Confidential
For HAOTEK Only

8.2 HDMI: Consumer Electronics Control

Consumer Electronics Control (CEC) is a protocol that enables high-level command and control of connected audiovisual devices. It consists of two major components: (1) Low-level protocol and (2) High-level protocol. The low-level protocol defines the bit timing and CEC frames. The high-level protocol defines features and message transmission functions. The CEC controller portion of the HDMI controller implements the low-level protocol, leaving the high-level protocol to be implemented via software.

The A12 CEC interface is covered in this section as follows:

- ([Section 8.2.1](#)) HDMI CEC: Register Map
- ([Section 8.2.2](#)) HDMI CEC: Register Details
- ([Section 8.2.3](#)) HDMI CEC: Operating Procedures

8.2.1 HDMI CEC: Register Map

CEC configuration registers are listed by address in the following table.

Register Address	Register Name	Description
0x600	<code>CEC_ctrl</code>	CEC Control
0x604	<code>CEC_status</code>	CEC Tx and Rx Status
0x608	<code>CEC_rx_start_ptrn</code>	Rx Pattern Start Bit
0x60C	<code>CEC_rx_zero_ptrn</code>	Rx Pattern Zero Bit
0x610	<code>CEC_rx_one_ptrn</code>	Rx Pattern for One Bit
0x614	<code>CEC_clk_ptrn</code>	Divided Clock Pattern
0x618	<code>CEC_tx_data0</code>	Tx Data Block 0 - 3
0x61C	<code>CEC_tx_data1</code>	Tx Data Block 4 - 7
0x620	<code>CEC_tx_data2</code>	Tx Data Block 8 - 11
0x624	<code>CEC_tx_data3</code>	Tx Data Block 12 - 15
0x628	<code>CEC_rx_data0</code>	Rx Data Block 0
0x62C	<code>CEC_rx_data1</code>	Rx Data Block 1
0x630	<code>CEC_rx_data2</code>	Rx Data Block 2
0x634	<code>CEC_rx_data3</code>	Rx Data Block 3
0x638	<code>CEC_ctrl2</code>	CEC Control Register 2
0x63C	<code>CEC_tole_ptrn_lower</code>	Lower Boundary Validation Range Pattern
0x640	<code>CEC_low_ptrn</code>	Pattern for 'LOW' bit
0x644	<code>CEC_tole_ptrn_upper</code>	Upper Boundary Validation Range Pattern
0x648	<code>CEC_misc_ptrn</code>	Miscellaneous Patterns (e.g., Nominal Sample Time and Glitch)
0x64C	<code>CEC_tx_start_ptrn</code>	Tx Pattern Start Bit
0x650	<code>CEC_tx_zero_ptrn</code>	Tx Pattern for Zero Bit
0x654	<code>CEC_tx_one_ptrn</code>	Tx Pattern for One Bit

Table 8-136. CEC Engine Registers.

8.2.2 HDMI CEC: Register Details

Details for each CEC register are outlined in this section.

8.2.2.1 CEC_ctrl Register

Bits	Name	Attr	Reset	Description
31	reset	RW		Local CEC reset
30	tx_enable	RW	0	Flag to enable a Tx frame transmission 0 - Disable 1 - Enable This flag will be auto-cleared after tx_interrupt_ok or tx_interrupt_fail is signaled.
29:26	tx_block_no	RW	0	Number of blocks in a Tx frame (0 - 15 means 1 - 16)
25:22	rx_block_no	R	0	Number of blocks in a Rx frame (0 - 15 means 1 - 16)
21	ack_reject	RW	0	Controls whether it is rejected when receiving data.
20	init_addr_retry_enb	RW	0	If init_addr_retry_enb equals to 1, the CTRL_STATE goes to resend state when the initiator address is illegal. Otherwise, the tx_interrupt_fail is asserted rather than resending state.
19:17	free_start_bit	RW	0	The free_start_bit is defined when the next Tx frame starts. 0 - Default free time: Retry: 3 - 5 bits New initiator issue: 5 - 7 bits Present initiator issue: 7 - 10 bits 1 - Prohibited 2 - Next frame starts at 3 - 5 bits 3 - Next frame starts at 5 - 7 bits 4 - Next frame starts at 7 - 10 bits 5 - Next frame starts at 10 - bits
16	cec_wait_for_ack	RW	1	Only used in Rx mode 0 - The CEC sends an acknowledge bit 2.4 ms after the EOM bit arrives 1 - The CEC sends an acknowledge bit depending on the ACK signal from Rx
15	rx_int_direct (function selection)	RW	1	Only used in Rx mode 0 - rx_interrupt is asserted no matter if the logical address of the current frame equals the logical address of the device 1 - rx_interrupt is asserted only when the end of this frame and the logical address of the frame equals the logical address of the device
14	nominal_sample_dis	RW	0	Only used in Tx mode 0 - The output signal of CEC is different from CEC line in HDMI jack, the CEC turns into the retry state 1 - The CEC enters the retry routine only when the CEC output signal is different from the CEC line in HDMI jack at the nominal sample point
13	cec_arbitration	RW	0	Asserted by the CEC module when the Tx arbitration occurs and must be cleared by software
12:0				Reserved

Table 8-137. CEC Control Register.

8.2.2.2 CEC_status Register

Bits	Name	Attr	Reset	Description
31:15				Reserved
14:12	tx_status	R		Tx status 1 - Idle 2 - Sending normal message 3 - Sending acknowledge message 4 - Sending lowbit message (when detecting illegal bit duration) 5 - Protected when Rx is receiving data
11				Reserved
10:8	normal_status	R		NORMAL state (when tx_status == 2) 1 - Sending start bit 2 - Sending block bit 3 - Waiting for ACK bit 4 - Idle
7	rx_fail	R		If rx_fail equals 1, receive mode is fail.
6	cec_reset_ok	R		If cec_reset_ok is 1, the reset signals of CEC registers are valid.
5:3				Reserved
2:0	rx_status	R		Rx status 0 - Idle3 (idle for 7 - 10 bit periods) 1 - Idle1 (idle for 0 - 3 bit periods) 2 - Idle2 (idle for 3 - 5 bit periods) 3 - Idle3 (idle for 5 - 7 bit periods) 6 - Idle3 (idle for >10 bit periods) 4 - Rx is busy 5 - Protected when Tx is sending data

Table 8-138. CEC Tx and Rx Status Register.

8.2.2.3 CEC_rx_start_ptrn Register

Bits	Name	Attr	Reset	Description
31:26				Reserved
25:16	period_all	RW	0x3B5	Number of CEC base clock cycles for the entire period in symbol start. Note the CEC base clock is specified in 0x514. (See CEC Figure 3 Start bit pulse format showing minimum and maximum tolerances in the <i>HDMI Spec.</i>) $4500 \text{ us} / (1/216 \text{ MHz} * 1000) = 0x3CE$
15:10				Reserved
9:0	period_0	RW	0x30C	Number of CEC base clock cycles for 0 period in symbol start $(3700 \text{ us} / (1/216 \text{ MHz} * 1000)) = 0x321$

Table 8-139. CEC Rx Pattern for the Start Bit Register.

8.2.2.4 CEC_rx_zero_ptrn Register

Bits	Name	Attr	Reset	Description
31:26				Reserved
25:16	period_all	RW	0x1FA	Number of CEC base clock cycles for the entire period in symbol zero. Note that the CEC base clock is specified in 0x514. (See CEC Figure 4 Timing diagrams for both bit states in the <i>HDMI Spec.</i>) $(2400 \text{ us} / (1/216 \text{ MHz} * 1000)) = 0x207$
15:10				Reserved
9:0	period_0	RW	0x13C	Number of CEC base clock cycles for 0 period in symbol zero. $1500 \text{ us} / (1/216 \text{ MHz} * 1000) = 325 = 0x145$

Table 8-140. CEC Rx Pattern for the Zero Bit Register.

8.2.2.5 CEC_rx_one_ptrn Register

Bits	Name	Attr	Reset	Description
31:26				Reserved
25:16	period_all	RW	0x1FA	Number of CEC base clock cycles for the entire period in symbol one. Note that the CEC base clock is specified in 0x514 (See CEC Figure 5 Timing Diagrams for Follower Asserted Bits (Logical 0) in the <i>HDMI Spec.</i>) $(2400 \text{ us} / (1/216 \text{ MHz} * 1000)) = 0x207$
15:10				Reserved
9:0	period_0	RW	0x7E	Number of CEC base clock cycles for 0 period in symbol one. $600 \text{ us} / (1/216 \text{ MHz} * 1000) = 130 = 0x82$

Table 8-141. CEC Rx Pattern for the One Bit Register.

8.2.2.6 CEC_clk_ptrn Register

Bits	Name	Attr	Reset	Description
31:13				Reserved
12:0	div_ratio	RW	0x400	CEC base clock freq = GCLK_AHB clock freq / div_ratio

Table 8-142. CEC Divided Clock Pattern Register.

The A12 chip sets GCK_AHB using a set of APIs, as described in [Chapter 2](#).

8.2.2.7 CEC_tx_data0 Register

Bits	Name	Attr	Reset	Description
31:24	block0	RW	0	1st Tx block data
23:16	block1	RW	0	2nd Tx block data
15:8	block2	RW	0	3rd Tx block data

Bits	Name	Attr	Reset	Description
7:0	block3	RW	0	4th Tx block data

Table 8-143. CEC Tx Data Block 0 - 3 Register.

8.2.2.8 CEC_tx_data1 Register

Bits	Name	Attr	Reset	Description
31:24	block4	RW	0	5th Tx block data
23:16	block5	RW	0	6th Tx block data
15:8	block6	RW	0	7th Tx block data
7:0	block7	RW	0	8th Tx block data

Table 8-144. CEC Tx Data Block 4 - 7 Register.

8.2.2.9 CEC_tx_data2 Register

Bits	Name	Attr	Reset	Description
31:24	block8	RW	0	9th Tx block data
23:16	block9	RW	0	10th Tx block data
15:8	block10	RW	0	11th Tx block data
7:0	block11	RW	0	12th Tx block data

Table 8-145. CEC Tx Data Block 8 - 11 Register

8.2.2.10 CEC_tx_data3 Register

Bits	Name	Attr	Reset	Description
31:24	block12	RW	0	13th Tx block data
23:16	block13	RW	0	14th Tx block data
15:8	block14	RW	0	15th Tx block data
7:0	block15	RW	0	16th Tx block data

Table 8-146. CEC Tx Data Block 12 - 15 Register.

8.2.2.11 CEC_rx_data0 Register

Bits	Name	Attr	Reset	Description
31:24	block0	R		1st block data in the Rx frame (header)
23:16	block1	R		2nd block data in the Rx frame
15:8	block2	R		3rd block data in the Rx frame
7:0	block3	R		4th block data in the Rx frame

Table 8-147. CEC Rx Data Block 0 - 3 Register.

8.2.2.12 CEC_rx_data1 Register

Bits	Name	Attr	Reset	Description
31:24	block4	R		5th block data in the Rx frame
23:16	block5	R		6th block data in the Rx frame
15:8	block6	R		7th block data in the Rx frame
7:0	block7	R		8th block data in the Rx frame

Table 8-148. CEC Rx Data Block 4 - 7 Register.

8.2.2.13 CEC_rx_data2 Register

Bits	Name	Attr	Reset	Description
31:24	block8	R		9th block data in the Rx frame
23:16	block9	R		10th block data in the Rx frame
15:8	block10	R		11th block data in the Rx frame
7:0	block11	R		12th block data in the Rx frame

Table 8-149. CEC Rx Data Block 8 - 11 Register.

8.2.2.14 CEC_rx_data3 Register

Bits	Name	Attr	Reset	Description
31:24	block12	R		13th block data in the Rx frame
23:16	block13	R		14th block data in the Rx frame
15:8	block14	R		15th block data in the Rx frame
7:0	block15	R		16th block data in the Rx frame

Table 8-150. CEC Rx Data Block 12 - 15 Register.

8.2.2.15 CEC_ctrl2 Register

Bits	Name	Attr	Reset	Description
31:28				Reserved
27:16	cec_free_init	RW	0x4F1	The 1st free time node for the 3-bit period
15:13				Reserved
12:8	cec_max_retry_no	RW	0x5	The maximum retry number
7:4				Reserved
3:0	dev_logic_addr	RW	0x1	The current device logical address

Table 8-151. CEC Control Register 2.

8.2.2.16 CEC_tole_ptrn_lower Register

This register sets the CEC pattern for the validation range of the lower boundary.

Bits	Name	Attr	Reset	Description
31:26				Reserved
25:16	cif_cec_bit_tole_200us_lower	RW	0x2F	Tolerance timing of lower boundary for 0.2 ms
15:10				Reserved
9:0	cif_cec_bit_tole_350us_lower	RW	0x4F	Tolerance timing of lower boundary for 0.35 ms

Table 8-152. CEC Tolerance Pattern Lower Boundary Register.

8.2.2.17 CEC_low_ptrn Register

Bits	Name	Attr	Reset	Description
31:26				Reserved
25:16	cif_cec_max_low_dur	RW	0x329	1.6*bit pattern period
15:10				Reserved
9:0	cif_cec_min_low_dur	RW	0x2C4	1.4*bit pattern period

Table 8-153. CEC LOW Bit Pattern Register.

8.2.2.18 CEC_tole_ptrn_upper Register

This register sets the pattern for the validation range of the upper boundary.

Bits	Name	Attr	Reset	Description
31:26				Reserved
25:16	cif_cec_bit_tole_200us_upper	RW	0x2F	Tolerance timing of upper boundary for 0.2 ms
15:10				Reserved
9:0	cif_cec_bit_tole_350us_upper	RW	0x4F	Tolerance timing of upper boundary for 0.35 ms

Table 8-154. CEC Tolerance Pattern Upper Boundary Register.

8.2.2.19 CEC_misc_ptrn Register

Bits	Name	Attr	Reset	Description
31:26				Reserved
25:16	max_glitch_count	RW	0x4F	Check to determine if the signal is a glitch. Used for Rx mode.
15:9				Reserved

Bits	Name	Attr	Reset	Description
8:0	nominal_sample_time	RW	0xDD	The cycle count definition of the nominal sample point when nominal_sample_dis = 0.

Table 8-155. CEC Miscellaneous Pattern Register.

8.2.2.20 CEC_tx_start_ptrn Register

Bits	Name	Attr	Reset	Description
31:26				Reserved
25:16	period_all	RW	0x3B5	Number of CEC base clock cycles for the entire period in symbol start. Note the CEC base clock is specified in 0x514. (See CEC Figure3 in the <i>HDMI Spec.</i>) $4500 \text{ us} / (1/216 \text{ MHz} * 1000) = 0x3CE$
15:10				Reserved
9:0	period_0	RW	0x30C	Number of CEC base clock cycles for 0 period in symbol start $(3700 \text{ us} / (1/216 \text{ MHz} * 1000)) = 0x321$

Table 8-156. CEC Tx Start Bit Pattern Register.

8.2.2.21 CEC_tx_zero_ptrn Register

Bits	Name	Attr	Reset	Description
31:26				Reserved
25:16	period_all	RW	0x1FA	Number of CEC base clock cycles for the entire period in symbol zero. Note the CEC base clock is specified in 0x514. (See CEC Figure 4 in the <i>HDMI Spec.</i>) $((2400 \text{ us} / (1/216 \text{ MHz} * 1000)) = 0x207)$
15:10				Reserved
9:0	period_0	RW	0x13C	Number of CEC base clock cycles for 0 period in symbol zero. $1500 \text{ us} / (1/216 \text{ MHz} * 1000) = 325 = 0x145$

Table 8-157. CEC Tx Zero Bit Pattern Register.

8.2.2.22 CEC_tx_one_ptrn Register

Bits	Name	Attr	Reset	Description
31:26				Reserved
25:16	period_all	RW	0x1FA	Number of CEC base clock cycles for the entire period in symbol one. Note the CEC base clock is specified in 0x514 (See CEC Figure 5 in the <i>HDMI Spec.</i>) $(2400 \text{ us} / (1/216 \text{ MHz} * 1000)) = 0x207$
15:10				Reserved

Bits	Name	Attr	Reset	Description
9:0	period_0	RW	0x7E	Number of CEC base clock cycles for 0 period in symbol one. 600 us / (1/216 MHz * 1000) = 130 = 0x82

Table 8-158. CEC Tx One Bit Pattern Register.

8.2.3 HDMI CEC: Operating Procedures

To transmit a CEC packet:

1. Program registers **CEC_tx_data0/1/2/3** with CEC frame data and fill the **CEC_ctrl** bit field **tx_block_no** with the number of CEC frames to be transmitted.
2. Set the **CEC_ctrl** bit **tx_enable** to 1 to enable CEC frame transmission.
3. Check the **HDMI_int_sts** bits **tx_interrupt_ok** or **tx_interrupt_fail** for device acknowledgment.
4. Clear **tx_interrupt_ok** or **tx_interrupt_fail** before the next frame transmission.

To receive a CEC packet:

1. Note that the system retrieves the interrupt from the **HDMI_int_sts** bit **rx_interrupt** when a CEC packet is received successfully by the controller.
2. Check the **CEC_ctrl** bit **rx_block_no** for the number of frames in the received packet.
3. Read the **CEC_rx_data0/1/2/3** frame data from registers.
4. Clear the **HDMI_int_sts** bit **rx_interrupt** for the next CEC packet.

9. I2S INTERFACE

This chapter covers register programming for the I2S interface as follows:

- [\(Section 9.1\) I2S: Overview](#)
- [\(Section 9.2\) I2S: Operation and Clocking](#)
- [\(Section 9.3\) I2S: Selecting the Data Format](#)
- [\(Section 9.4\) I2S: Setting the Transmitter Control Register](#)
- [\(Section 9.5\) I2S: Resetting the Transceiver](#)
- [\(Section 9.6\) I2S: Set Receiver Control Register](#)
- [\(Section 9.7\) I2S: Registers](#)

9.1 I2S: Overview

The A12 I2S interface supports full-duplex transmit/receive for digital audio. In addition to HDMI audio, the A12 I2S interface enables connections with external Audio AD/DA devices and supports audio recording/playback without HDMI. The A12 I2S ports can be used for two-channel audio. The I2S controller registers are located at AHB base address 0xE001.A000.

9.2 I2S: Operation and Clocking

- The A12 I2S interface provides various data formats, each of which can be configured using a mode register through the port AHB interface.
- The I2S interface supports two data transfer paths: one for transmit and the other for receive. Each data transfer path contains a FIFO, a FIFO threshold, FIFO flags, an interrupt signal, a DMA request and a DMA acknowledge signal. Interrupts to the ARM processor core are routed through the Vector Interrupt Controller (VIC) on the AHB bus. DMA transfers are handled through the AHB DMA Engine. The I2S can generate two interrupt signals to the Cortex-A9 (i.e., one for transmit and the other for receive). For related DMA information, please refer to [Section 14.3.2.4 “AHBSP_ctl Register”](#).
- The I2S clock GCLK_AU is typically sourced and configured by A12 system software using a set of APIs, and the register programming defined here is used for adjustment and refinement.
- The I2S bit clock **I2S_CLK** can serve as an input/output in I2S mode. In all other modes the transceiver acts as a master only, and **I2S_CLK** serves as an output.
- The I2S transmitter control registers and receiver control registers default to slave mode after reset. Programming must initialize these registers to master mode operation.
- When the I2S interface is in master mode, **I2S_CLK** serves as an output, and the bit clock is generated by dividing the audio master clock signal **CLK_AU** with the **I2S_clock** register **clk_div** field.
- Audio sampling rates of 8 kHz, 11.025 kHz, 12 kHz, 16 kHz, 22.05 kHz, 24 kHz, 32 kHz, 44.1 kHz, 48 kHz and 96 kHz are supported.

9.3 I2S: Selecting the Data Format

The I2S transceiver provides the following registers to determine the audio data format.

- Mode register **I2S_mode** which selects between five audio data formats.
- Word-length register **I2S_wlen** which selects the audio data precision.
- Word-position register **I2S_wpos** which selects the audio data location.
- Slot-counter register **I2S_slot** which selects the data time slot for DSP mode.

The sections below cover:

- ([Section 9.3.1](#)) I2S Format: I2S Mode
- ([Section 9.3.2](#)) I2S Format: Left-Justified Mode
- ([Section 9.3.3](#)) I2S Format: Right-Justified Mode
- ([Section 9.3.4](#)) I2S Format: MSB Extend Mode

9.3.1 I2S Format: I2S Mode

I2S mode is triggered by setting the **I2S_mode** register **mode** bit to 0x4. In I2S mode, the MSBs of **I2S_SO** and **I2S_SI** are valid on the second **I2S_CLK** cycle period following an **I2S_WS** transition. When the data format is set to I2S mode, the **tx_ws_inv** bit of the Transmitter Control register (**I2S_tx_ctrl**) and the **rx_ws_inv** bit of the Receiver Control register (**I2S_rx_ctrl**) must both be set to 0. If the corresponding **tx_ws_mst** (**I2S_tx_ctrl**) or **rx_ws_mst** (**I2S_rx_ctrl**) is a master the configuration table below applies.

Word Precision	Word Length	Word Position	WS Channel Width
16	0xF	0	16 bits
16	0xF	>1	32 bits
18	0x11	Ignored	32 bits
20	0x15	Ignored	32 bits
24	0x17	Ignored	32 bits

Table 9-1. Configuration for I2S Mode.

9.3.2 I2S Format: Left-Justified Mode

In left-justified mode (**I2S_mode** bit **mode** set to 0x0), the MSB of **I2S_SI** and **I2S_SO** is valid on the first **I2S_CLK** cycle period following an **I2S_WS** transition. When the data format is set to left-justified mode, the **tx_ws_inv** bit of the Transmitter Control register (**I2S_tx_ctrl**) and the **rx_ws_inv** bit of the Receiver Control register (**I2S_rx_ctrl**) must both be set to 1. If **tx_ws_mst** (**I2S_tx_ctrl**) or **rx_ws_mst** (**I2S_rx_ctrl**) is a master the configuration table below applies.

Word Precision	wlen	wpos	WS Channel Width
16	0xF	0	16 bits
16	0xF	>1	32 bits
18	0x11	Ignored	32 bits

20	0x15	Ignored	32 bits
24	0x17	Ignored	32 bits

Table 9-2. Configuration for Left-Justified Mode.

9.3.3 I2S Format: Right-Justified Mode

In right-justified mode (**I2S_mode** bit **mode** set to 0x1), the LSB of **I2S_SO** and **I2S_SI** is valid on the **I2S_CLK** cycle period preceding an **I2S_WS** transition. When the data format is set to right-justified mode, the **tx_ws_inv** bit of the Transmitter Control register (**I2S_tx_ctrl**) and the **rx_ws_inv** bit of the Receiver Control register (**I2S_rx_ctrl**) must be set to 1. When the **dai_wpos** register is more than zero, the transmitter will assert the SD output LOW and the receiver will ignore the SD input on the wpos period. If the **tx_ws_mst** (**I2S_tx_ctrl**) or **rx_ws_mst** (**I2S_rx_ctrl**) is a master use the configuration table below.

Word Precision	dai_wlen	dai_wpos	WS Channel Width
16	0xF	0x0	16 bits
16	0xF	0xF	32 bits
18	0x11	0xD	32 bits
20	0x15	0xB	32 bits
24	0x17	0x7	32 bits

Table 9-3. Configuration for Right-Justified Mode.

9.3.4 I2S Format: MSB Extend Mode

In MSB mode (**I2S_mode** bit **mode** set to 0x2), the MSB of **I2S_SO** and **I2S_SI** is extended from the first **I2S_CLK** cycle period following the current **I2S_WS** transition. The LSB of **I2S_SO** and **I2S_SI** is valid on the **I2S_CLK** cycle period preceding the next **I2S_WS** transition.

When the data format is set to MSB extend mode, **tx_ws_inv** (**I2S_tx_ctrl**) and **rx_ws_inv** (**I2S_rx_ctrl**) must be set to 1. When the **dai_wpos** register is more than zero, the transmitter will maintain the MSB of current channel data to output and the receiver will ignore the **I2S_SO** and **I2S_SI** input on the wpos period. If the **tx_ws_mst** (**I2S_tx_ctrl**) or **rx_ws_mst** (**I2S_rx_ctrl**) is a master the configuration table below applies.

Word precision	dai_wlen	dai_wpos	WS Channel Width
16	0xF	0	16 bits
16	0xF	0xF	32 bits
18	0x11	0xD	32 bits
20	0x15	0xB	32 bits
24	0x17	0x7	32 bits

Table 9-4. Configuration for MSB Extend Mode.

9.4 I2S: Setting the Transmitter Control Register

The sections below cover I2S transmitter settings:

- [\(Section 9.4.1\) I2S Transmitter: Loop Back Test](#)
- [\(Section 9.4.2\) I2S Transmitter: I2S_SO Bit Order](#)
- [\(Section 9.4.3\) I2S Transmitter: ws Invert](#)
- [\(Section 9.4.4\) I2S Transmitter: Unison](#)
- [\(Section 9.4.5\) I2S Transmitter: Transmit Mute](#)
- [\(Section 9.4.6\) I2S Transmitter: Transmit Mono](#)

For Confidential
HAOTEK Only

9.4.1 I2S Transmitter: Loop Back Test

When the Transmitter Control register (**I2S_tx_ctrl**) Loop Back Test bit (**loopback**) is set to 1, the transmitter will read the received data and the **I2S_SO** will delay one **I2S_WS** cycle before sending the received data. When the Loop Back Test bit of the Transmitter Control register is set to 1, the receiver writes the data into the receive FIFO normally and the transmitter reads from the transmit FIFO normally, but the data read from the FIFO will be dropped.

9.4.2 I2S Transmitter: I2S_SO Bit Order

The Transmitter Control register **I2S_tx_ctrl** **order** bit is used to configure the send order of **I2S_SO**. If this bit is 0, the first bit of **I2S_SO** following the edge of **I2S_WS** is the MSB of the FIFO data output bus. Otherwise, the first bit of **I2S_SO** is the LSB of the FIFO data output bus. For example, if the word length is 15 and the bit order is 0, the sequence of the **I2S_SO** bit data is bit 15 to bit 0. Otherwise, the sequence of the **I2S_SO** bit data is bit 0 to bit 15.

9.4.3 I2S Transmitter: ws Invert

When the transmitter is enabled, the left-channel data will be sent first. The Transmitter Control register **I2S_tx_ctrl** bit **ws_inv** is used to control whether the first left-channel data is sent on **I2S_WS = 0** or **I2S_WS = 1**. When **ws_inv** is set to 1, the first left-channel data will be sent on **I2S_WS = 0**. When **ws_inv** is set to 0, the first left-channel data will be sent on **I2S_WS = 1**. When the I2S Transceiver is in I2S mode, the **ws_inv** bit should be set to 0. When one of the left-justified, right-justified, or MSB extend modes is selected, the **ws_inv** bit should be set to 1.

9.4.4 I2S Transmitter: Unison

The Transmitter Control register (**I2S_tx_ctrl**) **unison** bit determines how data is written into the FIFOs by the APB interface. If this bit is set to 0, the left-channel FIFO will be written when the APB bus applies writes to offset 0x2C, while the right channel FIFO will be written at offset 0x14. If this bit is set to 1, writing to offset 0x2C will write the left- and right-channel FIFOs simultaneously. The left-channel value is determined by bits 31:16 and the right channel is determined by bits 15:0. Note that writing to offset 0x30 will write the right-channel FIFO but not the left-channel FIFO.

9.4.5 I2S Transmitter: Transmit Mute

When the Transmitter Control register (**I2S_tx_ctrl**) **mute** bit is written to 1, the mute sequence will be sent after any ongoing transfer is completed. When the mute sequence begins, the transmitter read FIFO operation, FIFO-related interrupts, and DMA request signals all function normally.

9.4.6 I2S Transmitter: Transmit Mono

The Transmitter Control register (**I2S_tx_ctrl**) **mono** bits control the transfer channel. If the **mono[1]** bit is 0, the transmitter is in dual-channel data transfer mode. Otherwise only one-channel data will be transmitted. If the **mono[0]** bit is 1, the I2S transmitter is left-monophonic (one-channel) and **I2S_SO** sends the left-channel data at the right-channel period. The right-channel FIFO will be read normally and the right-channel data will be dropped.

Otherwise, the I2S transmitter functions in right-monophonic (one-channel) transfer mode and **I2S_SO** sends right-channel data at the left-channel period. The left-channel FIFO will be read normally and the left-channel data will be dropped.

9.5 I2S: Resetting the Transceiver

There are two types of resets enabled in the I2S transceiver: a power-on reset and a FIFO reset. During a power-on reset, all the registers of the I2S transceiver are reset. A FIFO reset can result in one of several outcomes:

1. If the FIFO reset bit of the Initial Control register is set to 1, only the transmitter and receiver FIFOs are reset and the remaining configurable registers retain their original values.
2. If the transmitter FIFO reset bit is set to 1, only the transmitter FIFOs are reset and the remaining configurable registers, including the receiver FIFO, retain their values.
3. If the receiver FIFO reset bit is set to 1, only the receiver FIFO is reset and all other configurable registers retain their values.

9.6 I2S: Set Receiver Control Register

The sections below cover I2S receiver settings:

- ([Section 9.6.1](#)) I2S Receiver: Loop Back Test
- ([Section 9.6.2](#)) I2S Receiver: I2S_SI Bit Order
- ([Section 9.6.3](#)) I2S Receiver: ws Invert

9.6.1 I2S Receiver: Loop Back Test

The Receiver Control register (**I2S_rx_ctrl**) **loopback** bit controls whether the serial data source is **I2S_SI** or **I2S_SO**. If **loopback** is 1, the serial data source is **I2S_SO**. If the bit is 0, the serial data source is **I2S_SI**.

9.6.2 I2S Receiver: I2S_SI Bit Order

The Receiver Control register (**I2S_rx_ctrl**) **bit order** is used to control the sampling order of **I2S_SI**. If **order** is 0, the first sampled bit of **I2S_SO** following the edge of the **I2S_WS** source will be written in the MSB of the FIFO data. Otherwise, the first sampled bit of **I2S_SI** will be written in the LSB of the FIFO data.

9.6.3 I2S Receiver: ws Invert

The Receiver Control register (**I2S_rx_ctrl**) bit **rx_ws_inv** indicates the word-select **I2S_WS** signal polarity. In I2S mode, driving the **I2S_WS** signal low or high corresponds to the left channel or right channel, respectively. This represents the default mode. If the **rx_ws_inv** bit is set to 1 the **I2S_WS** signal takes on the opposite polarity.

9.7 I2S: Registers

This section provides a map of the I2S registers ([Section 9.7.1](#)) and details on register settings ([Section 9.7.2](#)). The remainder of the chapter provides notes on I2S programming specifics.

9.7.1 I2S Registers: Map

Register Offset	Register Name	Description
0x00	I2S_mode	Mode register
0x04	I2S_rx_ctrl	Receiver control
0x08	I2S_tx_ctrl	Transmitter control
0x0C	I2S_wlen	Word length
0x10	I2S_wpos	Ignored bits between two ws edges
0x14	I2S_slot	Number of slots for DSP mode
0x18	I2S_tx_fifo_lth	Transmitter FIFO threshold
0x1C	I2S_rx_fifo_gth	Receiver FIFO threshold
0x20	I2S_clock	Clock and word select control
0x24	I2S_init	Enable and reset register
0x28	I2S_tx_fifo_flag	Transmitter status register
0x2C	I2S_tx_left_data	Transmitter left channel data (for non-DMA transfers only)
0x30	I2S_tx_right_data	Transmitter right channel data (for non-DMA transfers only)
0x34	I2S_rx_fifo_flag	Receiver status
0x38	I2S_rx_data	Receiver FIFO threshold (for non-DMA transfers only)
0x3C	I2S_tx_fifo_cntr	Transmitter FIFO reside data count
0x40	I2S_rx_fifo_cntr	Receiver FIFO reside data count
0x44	I2S_tx_interrupt_enable	Transmitter interrupt enable
0x48	I2S_rx_interrupt_enable	Receiver interrupt enable
0x4C	I2S_rx_echo	Receiver echo enable
0x50	I2S_24bitmux_mode	DMA data L/R interleave (Only Left address is valid; Right address should not be used.)
0x54	I2S_shift	Tx/Rx shift
0x58	I2S_channel_select	Select channel
0x5C - 0x7C		Reserved
0x80	I2S_rx_data_dma	Required address for DMA transfers
0x88		Reserved
0xC0	I2S_tx_left_data_dma	Required address for DMA transfers (This precludes the use of non L/R interleaved mode)

Table 9-5. I2S Digital Audio interface (DAI) Registers.

9.7.2 I2S Registers: Detail

9.7.2.1 I2S_mode Register

Bits	Name	Attr	Reset	Description
31:3				Reserved
2:0	mode	RW	0	0x0 - Left-justified mode 0x1 - Right-justified mode 0x2 - MSB extend mode 0x4 - I2S mode 0x6 - DSP Mode

Table 9-6. I2S Mode Register.

9.7.2.2 I2S_rx_ctrl Register

Bits	Name	Attr	Reset	Description
31:4				Reserved
3	loopback	RW	0	Receiver loop back test 0 - The receiver will receive serial data from I2S_SI 1 - The receiver will receive serial data from I2S_SI
2	order	RW	0	Receiver bit order 0 - MSB is first 1 - LSB is first
1	rx_ws_mst	R	0	Receiver mode 0 - Receiver is in slave mode 1 - Receiver is in master mode; i.e., I2S_WS is output Note: The I2S interface supports Slave mode only in I2S mode. Software must program this field to 1 for all other modes.
0	rx_ws_inv	RW	0	Receiver ws invert 0 - Receive first data on I2S_WS = 0 1 - Receive first data on I2S_WS = 1

Table 9-7. I2S Receiver Control Register.

9.7.2.3 I2S_tx_ctrl Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7	loopback	RW	0	Transmitter loop back test 0 - The transmitter operates normally 1 - The transmitter will transmit the data that is received by the receiver
6	order	RW	0	Transmitter bit order 0 - MSB is first 1 - LSB is first

Bits	Name	Attr	Reset	Description
5	tx_ws_mst	RW	0	Transmitter mode 0 - Transmitter is in slave mode 1 - Transmitter is in master mode; i.e., I2S_WS is output Note: The I2S Interface supports slave mode in I2S mode only. Software must program this field to 1 for all other modes except I2S.
4	tx_ws_inv	RW	0	Transmitter ws invert 0 - Left channel under I2S_WS = 0 1 - Left channel under I2S_WS = 1
3	unison	RW	0	Transmitter unison 0 - tx_left_data[24:0] = left_data[24:0] 1 - tx_left_data[31:0] = {right_data[15:0], left_data[15:0]}
2	mute	RW	0	Transmitter mute 0 - Normal
1:0	mono	RW	0	Transmitter mono 0x0 - Stereo 0x1 - Reserved 0x2 - Mono right channel 0x3 - Mono left channel

Table 9-8. I2S Transmitter Control Register.

9.7.2.4 I2S_wlen Register

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	wlen	RW	0	Word precision - 1. For example, 16 bit = 0xF and 24 bit = 0x17.

Table 9-9. I2S Word Length Register.

9.7.2.5 I2S_wpos Register

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	ignored_bits	RW	0	The content of this register represents the number of ignored bits between ws rising and falling edge.

Table 9-10. I2S Word Position Register.

9.7.2.6 I2S_slot Register

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	slot count	RW	0	Data Slot count after each ws strobe (only in DSP mode)

Table 9-11. I2S Slot Register.

9.7.2.7 I2S_tx_fifo_lth Register

Bits	Name	Attr	Reset	Description
31:7				Reserved
6:0	ft	RW	0	Transmitter FIFO Threshold Register When the FIFO contains fewer words than this threshold, the DMA request and interrupt signal will be asserted.

Table 9-12. I2S Transmitter FIFO Threshold Register.

9.7.2.8 I2S_rx_fifo_gth Register

Bits	Name	Attr	Reset	Description
31:7				Reserved
6:0	ft	RW	0	Transmitter FIFO Threshold Register When the FIFO contains more words than this threshold, the DMA request and interrupt signal will be asserted.

Table 9-13. I2S Receiver FIFO Threshold Register.

9.7.2.9 I2S_clock Register

Bits	Name	Attr	Reset	Description
31:10				Reserved
9	woe	RW	0	I2S_WS signal output enable 0 - Disable output to the I2S_WS pin 1 - Enable output to the I2S_WS pin Note: The I2S Interface supports Slave mode for I2S mode only. Software must program this field to 1 for all other modes except I2S.
8	soe	RW	0	I2S_CLK signal output enable 0 - Disable output to the I2S_CLK pin 1 - Enable output to the I2S_CLK pin Note: The I2S Interface supports Slave mode for I2S mode only. Software must program this field to 1 for all other modes except I2S.
7	ss	RW	1	I2S_CLK source selection 0 - I2S_CLK is input (slave mode) 1 - I2S_CLK is output (master mode) Note: The I2S Interface supports Slave mode for I2S mode only. Software must program this field to 1 for all other modes except I2S.
6	tsp	RW	0	Transmitter I2S_CLK polarity 0 - I2S_SO data change at rising edge 1 - I2S_SO data change at falling edge
5	rsp	RW	0	Receiver I2S_CLK polarity 0 - Sample I2S_SI data at rising edge 1 - Sample I2S_SI data at falling edge
4:0	clk_div	RW	0	Clock divider: I2S_CLK frequency = CLK_AU frequency / [2 (clk_div+1)]

Table 9-14. I2S Clock Control Register.

9.7.2.10 I2S_init Register

Bits	Name	Attr	Reset	Description
31:5				Reserved
4	txfrst	W	0	Transmitter FIFO Reset. Write 1 to reset Transmitter FIFO
3	rxfrst	W	0	Receiver FIFO Reset. Write 1 to reset Receiver FIFO
2	te	RW	0	Transmitter enable
1	re	RW	0	Receiver enable
0	frst	W	0	FIFO reset Write 1 to reset both Transmitter & Receiver FIFOs. Automatically cleared by hardware after FIFO reset. Returns 0 on read by software.

Table 9-15. I2S Initial Control Register.

9.7.2.11 I2S_tx_fifo_flag Register

Bits	Name	Attr	Reset	Description
31:5				Reserved
4	ti	R	1	Transmitter idle flag
3	ftv	R	0	FIFO threshold is valid
2				Reserved
1	ff	R	0	FIFO full flag
0	fe	R	1	FIFO empty flag

Table 9-16. I2S Transmitter Status Register.

9.7.2.12 I2S_tx_left_data Register

When the **I2S_tx_ctrl** bit **unison** is 1 the data width of this address is 32 bits and the following register set detail applies.

Bits	Name	Attr	Reset	Description
31:0	data	W		The data width of this address is 32 bits when the I2S_tx_ctrl bit unison is 1. The high 16-bits data is written into right-channel data FIFO and the low 16-bits data is written into left-channel FIFO data.

Table 9-17. I2S Transmitter Left Channel Data Register When the Transmit Unison Bit of **I2S_tx_ctrl** is 1.

When the **I2S_tx_ctrl** bit **unison** is 0 the following register set detail applies.

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:0	data	W		When unison is 0 and the I2S is not in L/R interleaved mode (as described in Section 9.3 “I2S: Selecting the Data Format”), the data width of this address is 24 bits, which is only for left-channel FIFO data. When in L/R interleaved mode, this register is the write port of left/right interleaved data.

*Table 9-18. I2S Transmitter Left Channel Data Register When the Transmit Unison Bit of **I2S_tx_ctrl** is 0.*

9.7.2.13 I2S_tx_right_data Register

Bits	Name	Attr	Reset	Description
32:24				Reserved
23:0	data	W		The data width of this address is 24 bits for the right-channel FIFO data.

Table 9-19. I2S Transmitter Right Channel Data Register.

9.7.2.14 I2S_rx_fifo_flag Register

Bits	Name	Attr	Reset	Description
31:5				Reserved
4	ri	R	1	Receiver idle flag
3	ftv	R	0	FIFO threshold is valid
2	fo	R	0	FIFO overflow flag
1	ff	R	0	FIFO full flag
0	fe	R	1	FIFO empty flag

Table 9-20. I2S Receiver FIFO Flag Register.

9.7.2.15 I2S_rx_data Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:0	data	R	0	When the APB bus reads data from this address, the FIFO will be pop one entry data. When FIFO is empty, the APB read data is always zero.

Table 9-21. I2S Receiver Data Register.

9.7.2.16 I2S_tx_fifo_cntr Register

Bits	Name	Attr	Reset	Description
31:7				Reserved
6:0	c	R	0	Transmitter FIFO data count

Table 9-22. I2S Transmitter FIFO Counter Register.

9.7.2.17 I2S_rx_fifo_cntr Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	c	R	0	Receiver FIFO data counter

Table 9-23. I2S Receiver FIFO Counter register.

9.7.2.18 I2S_tx_interrupt_enable Register

Bits	Name	Attr	Reset	Description
31:5				Reserved
4	ti	RW	0	Transmitter idle interrupt enable
3	ftv	RW	1	FIFO valid threshold interrupt enable
2				Reserved
1	ff	RW	0	FIFO full interrupt enable
0	fe	RW	0	FIFO empty interrupt enable

Table 9-24. I2S Transmitter Interrupt Enable Register.

9.7.2.19 I2S_rx_interrupt_enable Register

Bits	Name	Attr	Reset	Description
31:5				Reserved
4	ri	RW	0	Receiver idle interrupt enable
3	ftv	RW	1	FIFO valid threshold interrupt enable
2	fo	RW	0	FIFO overflow interrupt enable
1	ff	RW	0	FIFO full interrupt enable
0	fe	RW	0	FIFO empty interrupt enable

Table 9-25. I2S Receiver Interrupt Enable Register.

9.7.2.20 I2S_rx_echo Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	echo	RW	0	0 - No echo 1 - Echo Rx data to Tx as well as forwarding to main memory

Table 9-26. I2S Receiver Rx Echo Register.

9.7.2.21 I2S_24bitmux_mode Register

When the 24-bit multiplexed mode is enabled, the Transmit Left register (**I2S_tx_left_data**) contains interleaved left- and right-channel data. The first data in the FIFO after reset is left-channel, the second is right-channel and so on.

Bits	Name	Attr	Reset	Description
31:4				Reserved
3	dma_boot_sel	RW	0	DMA/Boot Select 1 - Default value used during boot
2	reset_0	RW	0	1 - Reset Channel 0
1	fdma_burst	RW	0	0 - Enable FDMA Burst 1 - Disable FDMA Burst
0	multi_24_en	RW	0	0 - Disable 24-bit multiplexed mode 1 - Enable 24-bit multiplexed mode Word length register I2S_wlen must be set to 24-bit operation

Table 9-27. I2S 24-Bit Multiplexed Mode Register.

9.7.2.22 I2S_shift Register

Bits	Name	Attr	Reset	Description
31:2				Reserved
1	rx_shift_en	RW	0	0 - Disable Rx shift 1 - Enable Rx shift
0	tx_shift_en	RW	0	0 - Disable Tx shift 1 - Enable Tx shift

Table 9-28. I2S Shift Register.

9.7.2.23 I2S_channel_select Register

Bits	Name	Attr	Reset	Description
31:2				Reserved
1:0	channel_select	RW	0	0x0 - 2 channels enabled 0x1 - 4 channels enabled 0x2 - Reserved 0x3 - Reserved

Table 9-29. I2S Channel Select Register.

9.7.2.24 I2S_rx_data_dma Register

This register must be programmed for receiver data when using the AHB DMA Engine for transfer.

Software must use the no-increment mode of the DMA channel control registers and set the **src/dest blk** increment to 54 bytes or lower on the peripheral side (peripheral side refers to I2S).

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:0	data	R	0	When the AHB bus reads data from this address the FIFO will pop one entry. When FIFO is empty, the AHB read data is always zero.

Table 9-30. I2S Receive DMA Data Register.

9.7.2.25 I2S_tx_left_data_dma Register

This register must be programmed for transmitter data when using the AHB DMA Engine for transfer. If the sample word length is 16 bits/sample, unison mode must be enabled. If the sample word length is 24 bits/sample, the 24-bit L/R multiplexed mode must be enabled ([Section 9.3](#)). This effectively precludes the use of non L/R (left/right) interleaved data when using the AHB DMA engine for transfer.

Software must also use the no-increment mode of the DMA channel control register and set the source/destination block increment to 54 bytes or lower on the peripheral side (peripheral side refers to I2S).

Bits	Name	Attr	Reset	Description
31:0	data	W		This address is used to transfer data using the AHB DMA Engine When sample word length is 16 bits/sample, unison mode applies.

Table 9-31. I2S Transmit DMA Left Data Register When Sample Word Length is 16 Bits/Sample.

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:0	data	W		This address is used to transfer data using the AHB DMA Engine When sample word length is 24 bits/sample, the 24-bit L/R multiplexed mode applies.

Table 9-32. I2S Transmit DMA Left Data Register When Sample Word Length is 24 Bits/Sample.

For Confidential
For HAOTEK Only

10. ETHERNET CONTROLLER

This chapter provides information on the A12 Ethernet controller. The chapter is organized as follows:

- [\(Section 10.1\) Ethernet: Overview](#)
- [\(Section 10.2\) Ethernet: Register Maps](#)
- [\(Section 10.3\) Ethernet: Register Details](#)
- [\(Section 10.4\) Ethernet: Descriptors](#)

10.1 Ethernet: Overview

The Ethernet Controller enables a host to transmit and receive data over Ethernet in compliance with the “*IEEE 802.3-2002 Standard for Ethernet Based LANs*”. Features include:

- Supports 10/100-Mbps data transfer rates with IEEE 802.3-compliant RMII and MII interfaces to communicate with an external Fast Ethernet PHY
- Supports both full-duplex and half-duplex operation
- Preamble and start-of-frame data (SFD) insertion in Transmit (Tx), and deletion in Receive (Rx) paths
- Options for Automatic Pad generation and CRC Stripping on receive frames
- Received data FIFO and transmit data FIFO at 2 KB each
- Programmable InterFrameGap (40-96 bit-times in steps of 8)
- Supports a variety of flexible address filtering modes:
 - Three additional 48-bit perfect (DA) address filters with masks for each byte
 - Three 48-bit SA address comparison check with masks for each byte
 - 64-bit hash filter (optional) for multicast and uni-cast (DA) addresses
 - Option to pass all multicast addressed frames
 - Promiscuous mode support to pass all frames without any filtering for network monitoring
 - Passes all incoming packets (as per filter) with a status report
- Separate 32-bit status returned for transmission and reception packets
- Supports IEEE 802.1Q VLAN tag detection for reception frames
- MDIO Master interface (optional) for PHY device configuration and management
- Module for detection of LAN wake-up frames and AMD Magic Packet frames
- Receive module for checksum off-load for received IPv4 and TCP packets encapsulated by the Ethernet frame
- A second 25-MHz clock pin for Ethernet PHY

10.1.1 Ethernet Overview: Address Space

The Ethernet controller is mapped to the AHB base addresses 0xE000.E000 (MAC) and 0xE000.F000 (DMA).

10.1.2 Ethernet Overview: Interrupt Vector

Ethernet controller interrupts are mapped to the vector interrupt controller (VIC). The MAC Subsystem SBD_INTR_O interrupt maps to VIC 1 line 27. MAC core interrupt PMT_INTR_O maps to VIC 3 line 6.

10.1.3 Ethernet Overview: Clocking

The Ethernet clock GCLK_GTX is generally configured by A12 system software.

10.1.4 Ethernet Overview: System Level Block Diagram

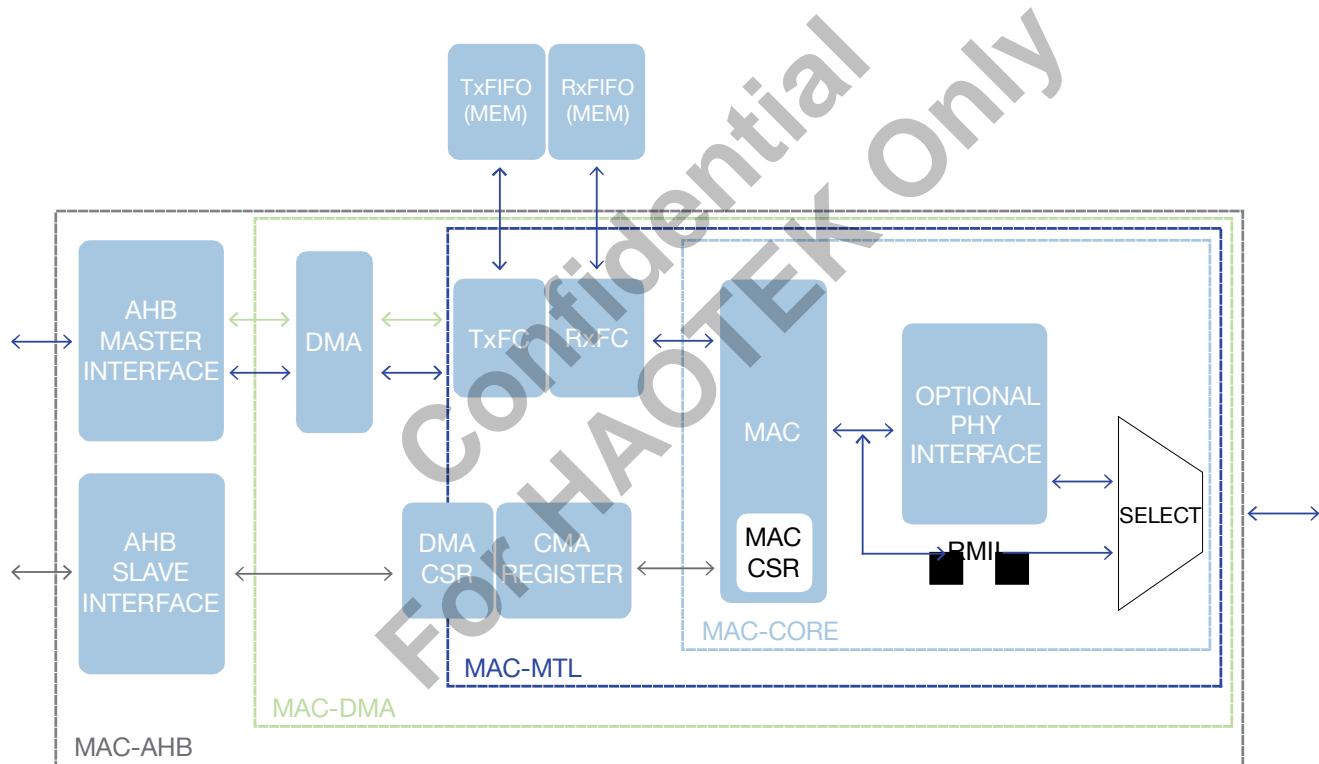


Figure 10-1. A12 Ethernet Controller System-Level Block Diagram.

10.2 Ethernet: Register Maps

The register maps in this section provide high-level summaries of each register or group of registers.

10.2.1 Ethernet: DMA Register Map

Register Offset	Register Name	Description
0x0000	DMA_R0_bus_mode	Register 0 - Bus Mode Controls the Host Interface Mode
0x0004	DMA_R1_tx_poll	Register 1 - Transmit Poll Demand Used by the host to instruct the DMA to poll the Transmit Descriptor List
0x0008	DMA_R2_rx_poll	Register 2 - Receive Poll Demand Used by the Host to instruct the DMA to poll the Receive Descriptor List
0x000C	DMA_R3_rx_descriptor	Register 3 - Receive Descriptor List Address Points DMA to the start of the Receive Descriptor List
0x0010	DMA_R4_tx_descriptor	Register 4 - Transmit Descriptor List Address Points DMA to the start of the Transmit Descriptor List
0x0014	DMA_R5_status	Register 5 - Status Read by software driver (application) during interrupt service routine or polling to determine the DMA status
0x0018	DMA_R6_operation_mode	Register 6 - Operation Mode Establishes Rx and Tx operating modes and command.
0x001C	DMA_R7_interrupt_enable	Register 7 - Interrupt Enable Enables the interrupts reported by the Status Register.
0x0020	DMA_R8_fmbuff_count	Register 8 - Missed Frame and Buffer Overflow Count Contains counters for discarded frames because host Receive Descriptor unavailable and Receive FIFO Overflow
0x0024		Reserved
0x0028	DMA_R10_axi_bus_mode	Register 10 - Controls the behavior of the AXI master to control burst splitting and the number of outstanding requests
0x0032 - 0x0044		Reserved
0x0048	DMA_R18_host_tx_descript	Register 18 - Current Host Transmit Descriptor Points to start of current Transmit Descriptor read by the DMA
0x004C	DMA_R19_host_rx_descript	Register 19 - Current Host Receive Descriptor Points to the start of current Receive Descriptor read by the DMA
0x0050	DMA_R20_host_tx_buffer	Register 20 - Current Host Transmit Buffer Address Points to the current Transmit Buffer address read by the DMA.
0x0054	DMA_R21_host_rx_buffer	Register 21 - Current Host Receive Buffer Address Points to the current Receive Buffer address read by the DMA

Table 10-1. Map of the Ethernet Controller DMA Registers.

10.2.2 Ethernet: MAC Register Map

Register Offset	Register Name	Description
0x0000	MAC_R0_MAC_config	Register 0 - MAC Configuration The operation mode register for the MAC
0x0004	MAC_R1_MAC_fm_filter	Register 1 - MAC Frame Filter Contains the frame filtering controls
0x0008	MAC_R2_hash_table_high	Register 2 - Hash Table High Contains the higher 32 bits of the Multicast Hash table
0x000C	MAC_R3_hash_table_low	Register 3 - Hash Table Low Contains the lower 32 bits of the Multicast Hash table
0x0010	MAC_R4_RMII_address	Register 4 - RMII Address Controls the management cycles to an external PHY
0x0014	MAC_R5_RMII_data	Register 5 - RMII Data register Contains data to write to or read from the PHY register
0x0018	MAC_R6_flow_control	Register 6 - Flow Control register Controls the generation of control frames
0x001C	MAC_R7_VLAN_tag	Register 7 - VLAN Tag register Identifies IEEE 802.1Q VLAN type frames
0x0020	MAC_R8_version	Register 8 - Version Identifies the version of the Core
0x0024 – 0x0034		Reserved
0x0038	MAC_R14_interrupt_status	Interrupt register
0x003C	MAC_R15_interrupt_mask	Interrupt Mask register
0x0040	MAC_R16_MAC_add0_high	Register 16 - MAC Address0 High Contains the higher 16 bits of the 1st MAC address
0x0044	MAC_R17_MAC_add0_low	Register 17 - MAC Address0 Low Contains the lower 32 bits of the 1st MAC address
0x0048	MAC_R18_MAC_add1_high	Register 18 - MAC Address1 High Contains the higher 16 bits of the 2nd MAC address
0x004C	MAC_R19_MAC_add1_low	Register 19 - MAC Address1 Low Contains the lower 32 bits of the 2nd MAC address
0x0050	MAC_R20_MAC_add2_high	Register 20 - MAC Address2 High Contains the lower 32 bits of the 3rd MAC address
0x0054	MAC_R20_MAC_add2_low	Register 21 - MAC Address2 Low Contains the lower 32 bits of the 3rd MAC address

Table 10-2. Map of the Ethernet Controller MAC Registers.

10.3 Ethernet: Register Details

The register descriptions that follow specify how the application and the core can access the register fields of the Control and Status Registers (CSRs). The register access conventions (Read, Write, Set, Clear, and so on) are provided in [Chapter 28](#).

Please note the following with regard to the Ethernet Control registers:

- All CSR registers are implemented as 32-bit registers and can be accessed in one Read/Write cycle with a 32-bit AHB slave port. If the user initiates a non-32-bit register access, the byte-enable signals are used to qualify the corresponding register bytes. Only Write operations are qualified with byte-enables while read-operations are always 32-bit.
- Note that register bits[7:0] correspond to address[1:0] = 00 in Little Endian mode, while register bits[31:24] correspond to address[1:0] = 00 in Big Endian mode.
- The transfer of particular register contents (e.g., MAC DA address register) to the Transmit or Receive clock domains are initiated when a particular byte is written. This is indicated in the register descriptions, where applicable.
- When any register content is transferred to a different clock domain after a write operation, there should be no further writes to the same location until the first write is updated. Otherwise, the second write operation will not update to the destination clock domain. Thus, the delay between two writes to the same register location should be at least four cycles of the destination clock.

10.3.1 Ethernet Register Details: DMA

This section defines the bits for each DMA register. For AHB slave interfaces, byte, half-word or word accesses are possible.

10.3.1.1 DMA_R0_bus_mode Register

The Bus Mode register (**DMA_R0_bus_mode**) establishes the bus operating modes for the DMA.

Bits	Name	Attr	Reset	Description
31:26		R	0	Reserved
25	aal	RW	0	Address-Aligned Beats When this bit is set high and the fb bit equals 1, the AHB interface generates all bursts aligned to the start address LS bits. If the fb bit equals 0, the first burst (accessing the data buffer's start address) is not aligned, but subsequent bursts are aligned to the address.
24	4xpbl	RW	0	4xPBL Mode When set high, this bit multiplies the programmable burst length (PBL) value (bits [22:17] or rpbl and bits [13:8] or pbl) four times. Thus the DMA will transfer data in to a maximum of 4, 8, 16, 32, 64 and 128 beats depending on the PBL value.
23	usp	RW	0	Use Separate PBL When set high, it configures the RxDMA to use the value configured in bits [22:17] or rpbl as PBL while the PBL value in bits [13:8] or pbl is applicable to TxDMA operations only. When reset to low, the PBL value in pbl or bits [13:8] is applicable for both DMA engines.

Bits	Name	Attr	Reset	Description
22:17	rpbl	RW	0x01	RxDMA programmable burst length (RPBL) These bits indicate the maximum number of beats to be transferred in one RxDMA transaction. This will be the maximum value that is used in a single block Read/Write. The RxDMA will always attempt to burst as specified in RPBL each time it starts a Burst transfer on the host bus. RPBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value will result in undefined behavior. These bits are valid and applicable only when USP is set high.
16	fb	RW	0	Fixed Burst This bit controls whether the AHB Master interface performs fixed burst transfers or not. When set, the AHB will use only SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers. When reset, the AHB will use SINGLE and INCR burst transfer operations.
15:14	pr	RW	0	Rx:Tx priority ratio RxDMA requests are given priority over TxDMA requests in the following ratio. This is valid only when the da bit is reset. 0x0 - 1:1 0x1 - 2:1 0x2 - 3:1 0x3 - 4:1
13:8	pbl	RW	0x01	Programmable Burst Length These bits indicate the maximum number of beats to be transferred in one DMA transaction. This will be the maximum value that is used in a single block Read/Write. The DMA will always attempt to burst as specified in pbl each time it starts a Burst transfer on the host bus. The bit pbl can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value will result in undefined behavior. When USP is set high, this pbl value is applicable for TxDMA transactions only. The pbl values have the following limitations. The maximum number of beats (pbl) possible is limited by the size of the Tx FIFO and Rx FIFO in the MTL layer and the data bus width on the DMA. The FIFO has a constraint that the maximum beat supported is half the depth of the FIFO. For different data bus widths and FIFO sizes, the valid pbl range (including x4 mode) is provided in the following table (Table 10-4). If pbl is common for both transmit and receive DMA, the minimum Rx FIFO and Tx FIFO depths must be considered. Do not program out-of-range pbl values, as this may result in system instability.
7				Reserved
6:2	dsl	RW	0	Descriptor Skip Length This bit specifies the number of Word/DWord/LWord (depending on 32/64/128-bit bus) to skip between two unchained descriptors. The address skipping begins from the end of the current descriptor to the start of the next descriptor. When dsl value equals zero, then the descriptor table is taken as contiguous by the DMA, in Ring mode.
1	da	RW	0	DMA Arbitration scheme 0 - Round-robin with Rx:Tx priority given in bits [15:14] 1 - Rx has priority over Tx

Bits	Name	Attr	Reset	Description
0	swr	R_WS_SC	0	Software Reset When this bit is set, the MAC DMA Controller resets all MAC Subsystem internal registers and logic. It is cleared automatically after the the reset operation has completed in all core clock-domains.

Table 10-3. Ethernet Controller DMA Register 0 (Bus Mode) Which Controls the Host Interface Mode.

Data Bus Width	FIFO Depth	Valid PBL Range in Full-Duplex Mode	Valid PBL Range in Half-Duplex Mode
64	2KB	All	All

Table 10-4. Ethernet PBL Values.

10.3.1.2 DMA_R1_tx_poll Register

The Transmit Poll Demand register (**DMA_R1_tx_poll**) enables the Transmit DMA to verify whether or not the current descriptor is owned by DMA.

Bits	Name	Attr	Reset	Description
31:0	tpd	RO_WT	0	Transmit Poll Demand When these bits are written with any value, the DMA reads the current descriptor pointed to by the Current Host Transmit Descriptor register or DMA_R18_host_tx_descript . If that descriptor is not available (owned by Host), transmission returns to the Suspend state and the DMA Status register (DMA_R5_status[2]) is asserted. If the descriptor is available, transmission resumes.

Table 10-5. Ethernet Controller DMA Register 1 is Used to Instruct the DMA to Poll the Transmit Descriptor List.

10.3.1.3 DMA_R2_rx_poll Register

The Receive Poll Demand register (**DMA_R2_rx_poll**) enables the receive DMA to check for new descriptors.

Bits	Name	Attr	Reset	Description
31:0	rpd	RO_WT	0	Receive Poll Demand When these bits are written with any value, the DMA reads the current descriptor pointed to by DMA_R19_host_rx_descript . If that descriptor is not available (owned by Host), reception returns to the Suspended state and the DMA status register (DMA_R5_status[7]) is not asserted. If the descriptor is available, reception resumes.

Table 10-6. Ethernet Controller DMA Register 2 is Used to Instruct the DMA to Poll the Receive Descriptor List.

10.3.1.4 DMA_R3_rx_descriptor Register

The Receive Descriptor List Address register (**DMA_R3_rx_descriptor**) points to the start of the Receive Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word/DWord/ LWord-aligned (for 32/64/128-bit data bus). The DMA internally converts the lists to bus-width aligned addresses by making the corresponding LS bits low. Writing to **DMA_R3_rx_descriptor** is permitted only when reception is stopped. When stopped, **DMA_R3_rx_descriptor** must be written to before the receive Start command is given.

Bits	Name	Attr	Reset	Description
31:0	srl	RW	0	Start of Receive List This field contains the base address of the First Descriptor in the Receive Descriptor list. The LSB bits [1/2/3:0] for 32/64/128-bit bus-width) will be ignored and taken as all-zero by the DMA internally. Hence these LSB bits are Read-Only.

Table 10-7. Ethernet Controller DMA Register 3 Points the DMA to the Start of the Receive Descriptor List.

10.3.1.5 DMA_R4_tx_descriptor Register

The Transmit Descriptor List Address register (**DMA_R4_tx_descriptor**) points to the start of the Transmit Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word/DWord/LWord-aligned (for 32/64/128-bit data bus). The DMA internally converts the lists to bus-width aligned addresses by making the corresponding LSB bits low. Writing to **DMA_R4_tx_descriptor** is permitted only when transmission has stopped. When stopped, **DMA_R4_tx_descriptor** can be written before the transmission Start command is given.

Bits	Name	Attr	Reset	Description
31:0	start_tx	RW	0	Start of Transmit List This field contains the base address of the First Descriptor in the Transmit Descriptor list. The LSB bits [1/2/3:0] for 32/64/128-bit bus-width) will be ignored and taken as all-zero by the DMA internally. Hence these LSB bits are Read Only.

Table 10-8. Ethernet Controller DMA Register 4 Points the DMA to the Start of the Transmit Descriptor List.

10.3.1.6 DMA_R5_status Register

The Status register (**DMA_R5_status**) contains all the status bits that the DMA reports to the host. This Register 5 (**DMA_R5_status**) is typically read by the software driver during an interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted. **DMA_R5_status** bits are not cleared when read. Writing 1 to (unreserved) bits in this **DMA_R5_status[16:0]** clears them and writing 0 has no effect. Each field (bits[16:0]) can be masked by masking the appropriate bit in **DMA_R7_interrupt_enable**.

Bits	Name	Attr	Reset	Description
31:29				Reserved
28	gpi	R	0	<p>MAC PMT Interrupt</p> <p>This bit reflects the PMT_INTR_O signal output of the MAC core. The software must read the corresponding registers in the MAC core to retrieve the exact cause of interrupt and clear the source of interrupt to make this bit 0. The interrupt signal from the MAC subsystem (SBD_INTR_O) is high when this bit is high.</p>
27	gmi	R	0	<p>MAC MMC Interrupt</p> <p>This bit reflects an interrupt event in the MMC module of the MAC core. The software must read the corresponding registers in the MAC core to retrieve the exact cause of interrupt and clear the source of interrupt to make this bit 0. The interrupt signal from the MAC subsystem (SBD_INTR_O) is high when this bit is high.</p>
26	gli	R	0	<p>MAC Line interface Interrupt</p> <p>This bit reflects an interrupt event in the PCS interface block of the MAC Core. The software must read the corresponding registers in the MAC core to retrieve the exact cause of interrupt and clear the source of interrupt to make this bit 0. The interrupt signal from the MAC subsystem (SBD_INTR_O) is high when this bit is high.</p>
25:23	eb	R	0	<p>Error Bits</p> <p>These bits indicate the type of error that triggered a Bus Error (error response on the AHB interface). Valid only with Fatal Bus Error bit (DMA_R5_status[13]) set. This field does not generate an interrupt.</p> <p>Bit 23</p> <p>0 - Error during data transfer by RxDMA 1 - Error during data transfer by TxDMA</p> <p>Bit 24</p> <p>0 - Error during write transfer 1 - Error during read transfer</p> <p>Bit 25</p> <p>0 - Error during data-buffer access 1 - Error during descriptor access</p>
22:20	ts	R	0	<p>Transmit Process State</p> <p>These bits indicate the Transmit DMA FSM state. This field does not generate an interrupt.</p> <p>0x0 - Stopped; Reset or Stop Transmit Command issued. 0x1 - Running; Fetching Transmit Transfer Descriptor. 0x2 - Running; Waiting for status. 0x3 - Running; Reading Data from host memory buffer and queuing it to transmit buffer (Tx FIFO). 0x4, 0x5 - Reserved. 0x6 - Suspended; Transmit Descriptor Unavailable or Transmit Buffer Underflow. 0x7 - Running; Closing Transmit Descriptor.</p>

Bits	Name	Attr	Reset	Description
19:17	rs	R	0	<p>Receive Process State</p> <p>These bits indicate the Receive DMA FSM state. This field does not generate an interrupt.</p> <p>0x0 - Stopped: Reset or Stop Receive Command issued. 0x1 - Running: Fetching Receive Transfer Descriptor. 0x2 - Reserved. 0x3 - Running: Waiting for receive packet. 0x4 - Suspended: Receive Descriptor Unavailable. 0x5 - Running: Closing Receive Descriptor. 0x6 - Reserved. 0x7 - Running: Transferring the receive packet data from receive buffer to host memory.</p>
16	nis	R_WC	0	<p>Normal Interrupt Summary</p> <p>Normal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in DMA_R7_interrupt_enable:</p> <ul style="list-style-type: none"> DMA_R5_status[0] - Transmit Interrupt DMA_R5_status[2] - Transmit Buffer Unavailable DMA_R5_status[6] - Receive Interrupt DMA_R5_status[14] - Early Receive Interrupt <p>Only unmasked bits affect the Normal Interrupt Summary bit. This is a sticky bit and must be cleared (by writing 1 to this bit) each time a corresponding bit that causes nis to be set is cleared.</p>
15	ais	R_WC	0	<p>Abnormal Interrupt Summary</p> <p>Bit value is the logical OR of the following when the corresponding interrupt bits are enabled in DMA_R7_interrupt_enable:</p> <ul style="list-style-type: none"> DMA_R5_status[1] - Transmit Process Stopped DMA_R5_status[3] - Transmit Jabber Timeout DMA_R5_status[4] - Receive FIFO Overflow DMA_R5_status[5] - Transmit Underflow DMA_R5_status[7] - Receive Buffer Unavailable DMA_R5_status[8] - Receive Process Stopped DMA_R5_status[9] - Receive Watchdog Timeout DMA_R5_status[10] - Early Transmit Interrupt DMA_R5_status[13] - Fatal Bus Error <p>Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit and must be cleared each time a corresponding bit that causes ais to be set is cleared.</p>
14	eri	R_SS_SC_WC	0	<p>Early Receive Interrupt</p> <p>This bit indicates that the DMA has filled the first data buffer of the packet. Receive Interrupt DMA_R5_status[6] automatically clears this bit.</p>
13	fbi	R_WC	0	<p>Fatal Bus Error Interrupt</p> <p>This bit indicates that a Bus Error occurred (DMA_R5_status[25:23]). When this bit is set, the DMA disables all its bus accesses.</p>
12:11				Reserved
10	eti	R_WC	0	<p>Early Transmit Interrupt</p> <p>This bit indicates that the frame to be transmitted was fully transferred to the MTL Transmit FIFO.</p>

Bits	Name	Attr	Reset	Description
9	rwt	R_WC	0	Receive Watchdog Timeout This bit is asserted when a frame with a length greater than 2048 bytes is received.
8	rps	R_WC	0	Receive Process Stopped This bit is asserted when the Receive Process enters the Stopped state.
7	ru	R_WC	0	Receive Buffer Unavailable This bit indicates that the Next Descriptor in the Receive List is owned by the host and cannot be acquired by the DMA. The Receive Process is suspended. To resume processing Receive descriptors, the host should change the ownership of the descriptor and issue a Receive Poll Demand command. If no Receive Poll Demand is issued, Receive Process resumes when the next recognized incoming frame is received. DMA_R5_status[7] is set only when the previous Receive Descriptor was owned by the DMA.
6	ri	R_WC	0	Receive Interrupt This bit indicates the completion of frame reception. Specific frame status information has been posted in the descriptor. Reception remains in the Running state.
5	unf	R_WC	0	Transmit Underflow This bit indicates that the Transmit Buffer experienced an Underflow during frame transmission. Transmission is suspended and an Transmit Descriptor Underflow Error is set using Transmit Descriptor 0 bit 1.
4	ovf	R_WC	0	Receive Overflow This bit indicates that the Receive Buffer experienced an Overflow during frame reception. If the partial frame is transferred to the application, the Receive Descriptor overflow status is set using Receive Descriptor 0 bit 11.
3	tjt	R_WC	0	Transmit Jabber Timeout This bit indicates that the Transmit Jabber Timer expired, meaning that the transmitter had been excessively active. The transmission process is aborted and placed in the Stopped state. This causes the Transmit Jabber Timeout flag to assert (i.e., Transmit Descriptor 0 bit 14).
2	tu	R_WC	0	Transmit Buffer Unavailable This bit indicates that the Next Descriptor in the Transmit List is owned by the host and cannot be acquired by the DMA. Transmission is suspended. Bits[22:20] explain the Transmit Process state transitions. To resume processing transmit descriptors, the host should change the ownership of the bit of the descriptor and then issue a Transmit Poll Demand command.
1	tps	R_WC	0	Transmit Process Stopped This bit is set when the transmission is stopped.
0	ti	R_WC	0	Transmit Interrupt This bit indicates that frame transmission is complete and Transmit Descriptor 1 bit 31 is set.

Table 10-9. Ethernet Controller DMA Register 5 is Read to Determine the Status of the DMA.

10.3.1.7 DMA_R6_operation_mode Register

The Operation Mode register (**DMA_R6_operation_mode**) establishes the Transmit and Receive operating modes and commands. This register should be the last CSR to be written as part of DMA initialization.

Bits	Name	Attr	Reset	Description
31:27				Reserved
26	dt	RW	0	Disable Dropping of TCP/IP Checksum Error Frames When this bit is set, the core does not drop frames that only have errors detected by the Receive Checksum Offload engine. Such frames do not include errors (including FCS errors) in the Ethernet frame received by the MAC, but have errors in the encapsulated payload only. When this bit is reset, all error frames are dropped if the fef bit is reset.
25	rsf	RW	0	Receive Store and Forward When this bit is set, the MTL only reads a frame from the Rx FIFO after the complete frame has been written to it, ignoring RTC bits. When this bit is reset, the Rx FIFO operates in Cut-Through mode, subject to the threshold specified by the RTC bits.
24	dff	RW	0	Disable Flushing of Received Frames When this bit is set, the RxDMA does not flush any frames due to the unavailability of receive descriptors/buffers as it does normally when this bit is reset.
23:22				Reserved
21	tsf	RW	0	Transmit Store and Forward When this bit is set, transmission begins when a full frame resides in the MTL Transmit FIFO. When this bit is set, the ttc values specified in DMA_R6_operation_mode [16:14] are ignored. This bit should be changed only when transmission is stopped.
20	ftf	R_WS_SC	0	Flush Transmit FIFO When this bit is set, the transmit FIFO controller logic is reset to its default values and thus all data in the TxFIFO is lost/flushed. This bit is cleared internally when the flushing operation is completed. DMA_R6_operation_mode should not be written to until this bit is cleared.
19:17				Reserved
16:14	ttc	RW	0	Transmit Threshold Control These three bits control the threshold level of the MTL Transmit FIFO. Transmission begins when the frame size within the MTL Transmit FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are used only when the corresponding store-forward bit is reset. 0x0 - 64 0x1 - 128 0x2 - 192 0x3 - 256 0x4 - 40 0x5 - 32 0x6 - 24 0x7 - 16

Bits	Name	Attr	Reset	Description
13	st	RW	0	<p>Start/Stop Transmission Command</p> <p>When this bit is set, transmission is placed in the Running state, and the DMA checks the Transmit List at the current position for a frame to be transmitted. Descriptor acquisition is attempted either from the current position in the list, which is the Transmit List Base Address set by DMA_R4_tx_descriptor, or from the position retained when transmission was stopped previously. If the current descriptor is not owned by the DMA, transmission enters the Suspended state and the Transmit Buffer Unavailable (DMA_R5_status[2]) bit is set. The Start Transmission command is effective only when transmission is stopped. If the command is issued before setting DMA_R4_tx_descriptor, this will result in unpredictable DMA behavior.</p> <p>When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and becomes the current position when transmission resumes. The stop transmission command is effective only when the transmission of the current frame is complete or when the transmission is in the Suspended state.</p>
12:11	rfd	RW	0	<p>Threshold for de-activating flow control (in both HD and FD)</p> <p>These bits control the threshold (fill-level of RxFIFO) at which flow-control is deasserted after activation.</p> <p>0x0 - Full 1 KB 0x1 - Full 2 KB 0x2 - Full 3 KB 0x3 - Full 4 KB</p> <p>Note that the deassertion is effective only after the flow-control is asserted.</p>
10:8				Reserved
7	fef	RW	0	<p>Forward Error frames</p> <p>When reset, the RxFIFO will drop frames with error status (CRC error, collision error, RMII_ER, giant frame, watchdog timeout, overflow). However, if the frame's start byte (write) pointer is already transferred to the Read Controller-side, then the frames will not be dropped. When fef is set, all frames, except runt-error frames, will be forwarded to the DMA.</p>
6	fuf	RW	0	<p>Forward Undersized good frames</p> <p>When set, the RxFIFO will forward Undersized frames (runt frames containing no errors and with lengths less than 64 bytes, including pad-bytes and CRC). When reset, the Rx-FIFO will drop all frames of size less than 64 bytes, unless they have already been transferred due to a lower value of Receive Threshold (e.g., RTC = 01).</p>
5				Reserved

Bits	Name	Attr	Reset	Description
4:3	rtc	RW	0	<p>Receive Threshold control</p> <p>These two bits control the threshold level of the MTL Receive FIFO. Transfer (request) to DMA starts when the frame size within the MTL Receive FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are transferred automatically.</p> <p>0x0 - 64 0x1 - 32 0x2 - 96 0x3 - 128</p>
2	osf	RW	0	<p>Operate on Second Frame</p> <p>When this bit is set, the DMA is instructed to process a second frame of Transmit data even before the status of the first frame is obtained.</p>
1	sr	RW	0	<p>Start/Stop Receive</p> <p>When this bit is set, the Receive process is placed in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes incoming frames. Descriptor acquisition is attempted from the current position in the list, which is the address set in DMA_R3_rx_descriptor or the position retained when the Receive process was previously stopped. If no descriptor is owned by the DMA, reception is suspended and the Receive Buffer Unavailable (DMA_R5_status[7]) bit is set. The Start Receive command is effective only when reception has stopped. If the command was issued before setting the register DMA_R3_rx_descriptor, unpredictable DMA behavior will result.</p> <p>When this bit is cleared, RxDMA operation is terminated following the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process resumes. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state.</p>
0				Reserved

Table 10-10. Ethernet Controller DMA Register 6 for Receive and Transmit Operating Modes and Commands.

10.3.1.8 DMA_R7_interrupt_enable Register

The Interrupt Enable register (**DMA_R7_interrupt_enable**) enables the interrupts reported by **DMA_R5_status**. Setting a bit to 0x1 enables a corresponding interrupt. After a hardware or software reset, all interrupts are disabled.

Bits	Name	Attr	Reset	Description
31:17				Reserved

Bits	Name	Attr	Reset	Description
16	nie	RW	0	<p>Normal Interrupt Summary Enable</p> <p>When this bit is set, a normal interrupt is enabled. When this bit is reset, a normal interrupt is disabled. This bit enables the following bits:</p> <ul style="list-style-type: none"> DMA_R5_status[0] - Transmit Interrupt DMA_R5_status[2] - Transmit Buffer Unavailable DMA_R5_status[6] - Receive Interrupt DMA_R5_status[14] - Early Receive Interrupt
15	aie	RW	0	<p>Abnormal Interrupt Summary Enable</p> <p>When this bit is set, an abnormal interrupt is enabled. When this bit is reset, an abnormal interrupt is disabled. This bit enables the following bits.</p> <ul style="list-style-type: none"> DMA_R5_status[1] - Transmit Process Stopped DMA_R5_status[3] - Transmit Jabber Timeout DMA_R5_status[4] - Receive Overflow DMA_R5_status[5] - Transmit Underflow DMA_R5_status[7] - Receive Buffer Unavailable DMA_R5_status[8] - Receive Process Stopped DMA_R5_status[9] - Receive Watchdog Timeout DMA_R5_status[10] - Early Transmit Interrupt DMA_R5_status[13] - Fatal Bus Error
14	ere	RW	0	<p>Early Receive Interrupt Enable</p> <p>When this bit is set with the Normal Interrupt Summary Enable (DMA_R7_interrupt_enable[16]) bit, Early Receive Interrupt is enabled. When this bit is reset, Early Receive Interrupt is disabled.</p>
13	fbe	RW	0	<p>Fatal Bus Error Enable</p> <p>When this bit is set with the Abnormal Interrupt Summary Enable (DMA_R7_interrupt_enable[15]) bit, the Fatal Bus Error Interrupt is enabled. When this bit is reset, Fatal Bus Error Enable Interrupt is disabled.</p>
12:11				Reserved
10	ete	RW	0	<p>Early Transmit Interrupt Enable</p> <p>When this bit is set with an Abnormal Interrupt Summary Enable (DMA_R7_interrupt_enable[15]), Early Transmit Interrupt is enabled. When this bit is reset, Early Transmit Interrupt is disabled.</p>
9	rwe	RW	0	<p>Receive Watchdog Timeout Enable</p> <p>When this bit is set with an Abnormal Interrupt Summary Enable (DMA_R7_interrupt_enable[15]), the Receive Watchdog Timeout Interrupt is enabled. When this bit is reset, Receive Watchdog Timeout Interrupt is disabled.</p>
8	rse	RW	0	<p>Receive Stopped Enable</p> <p>When this bit is set with an Abnormal Interrupt Summary Enable (DMA_R7_interrupt_enable[15]), Receive Stopped Interrupt is enabled. When this bit is reset, Receive Stopped Interrupt is disabled.</p>
7	rue	RW	0	<p>Receive Buffer Unavailable Enable</p> <p>When this bit is set with an Abnormal Interrupt Summary Enable (DMA_R7_interrupt_enable[15]), Receive Buffer Unavailable Interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable Interrupt is disabled.</p>

Bits	Name	Attr	Reset	Description
6	rie	RW	0	Receive Interrupt Enable When this bit is set with the Normal Interrupt Summary Enable (DMA_R7_interrupt_enable[16]) bit, Receive Interrupt is enabled. When this bit is reset, Receive Interrupt is disabled.
5	une	RW	0	Underflow Interrupt Enable When this bit is set with the Abnormal Interrupt Summary Enable (DMA_R7_interrupt_enable[15]) bit, Transmit Underflow Interrupt is enabled. When this bit is reset, Underflow Interrupt is disabled.
4	ove	RW	0	Overflow Interrupt Enable When this bit is set with the Abnormal Interrupt Summary Enable (DMA_R7_interrupt_enable[15]) bit, Receive Overflow Interrupt is enabled. When this bit is reset, Overflow Interrupt is disabled.
3	tje	RW	0	Transmit Jabber Timeout Enable When this bit is set with the Abnormal Interrupt Summary Enable (DMA_R7_interrupt_enable[15]) bit, Transmit Jabber Timeout Interrupt is enabled. When this bit is reset, Transmit Jabber Timeout Interrupt is disabled.
2	tue	RW	0	Transmit Buffer Unavailable Enable When this bit is set with the Normal Interrupt Summary Enable (DMA_R7_interrupt_enable[16]) bit, Transmit Buffer Unavailable Interrupt is enabled. When this bit is reset, Transmit Buffer Unavailable Interrupt is disabled.
1	tse	RW	0	Transmit Stopped Enable When this bit is set with the Abnormal Interrupt Summary Enable (DMA_R7_interrupt_enable[15]) bit, Transmission Stopped Interrupt is enabled. When this bit is reset, Transmission Stopped Interrupt is disabled.
0	tie	RW	0	Transmit Interrupt Enable When this bit is set with the Normal Interrupt Summary Enable (DMA_R7_interrupt_enable[16]) bit, Transmit Interrupt is enabled. When this bit is reset, Transmit Interrupt is disabled.

Table 10-11. Ethernet Controller DMA Register 7 Enables the Interrupts Reported by the Status Register.

10.3.1.9 DMA_R8_fmbuff_count Register

The DMA maintains two counters to track the number of missed frames during reception. This register (**DMA_R8_fmbuff_count**) reports the current value of the counter. The counter is used for diagnostic purposes. Bits[15:0] indicate missed frames due to the unavailability of the host buffer. Bits[27:17] indicate missed frames due to buffer overflow conditions (MTL and MAC) and runt frames (good frames of size less than 64 bytes) dropped by the MTL.

Bits	Name	Attr	Reset	Description
31:29				Reserved
28	over_fifo	R_WC	0	Overflow bit for FIFO Overflow Counter

Bits	Name	Attr	Reset	Description
27:17	fms_count	R_WC	0	Number of frames missed by the application This counter is incremented each time the MTL asserts the sideband signal MTL_RXOVERFLOW_O. The counter is cleared when this register is written with any value with MCI_BE_I[2] as 1.
16	over_fms_count	R_WC	0	Overflow bit for Missed Frame Counter
15:0	fms_ctrl	R_WC	0	Number of frames missed by the controller Indicates the number of frames missed by the controller due to the unavailability of the Host Receive Buffer. This counter is incremented each time the DMA discards an incoming frame. The counter is cleared when this register is written with any value with MCI_BE_I[0] as 1.

Table 10-12. Ethernet Controller DMA Register 8 for the Missed Frame and Buffer Overflow Counter.

10.3.1.10 DMA_R10_axi_bus_mode Register

The AXI Bus Mode Register controls the behavior of the AXI master. It is mainly used to control the burst splitting and the number of outstanding requests. This register is present and valid only in the GMAC-AXI configuration. In addition, this register is valid only in the Channel 0 DMA when multiple channels are present in the AV mode.

Bits	Name	Attr	Reset	Description
31:24		RO	00H	Reserved
23:20	WR_OSR_LMT	R_W	'h1	AXI Maximum Write Outstanding Request Limit This value limits the maximum outstanding request on the AXI write interface. Maximum outstanding requests - WR_OSR_LMT+1 Note: <ul style="list-style-type: none">• Bit 22 is reserved if AXI_GM_MAX_WR_REQUESTS = 4• Bit 23 is reserved if AXI_GM_MAX_WR_REQUESTS != 16
19:16	RD_OSR_LMT	R_W	'h1	AXI Maximum Read Outstanding Request Limit This value limits the maximum outstanding requests on the AXI read Interface. Maximum outstanding requests = RD_OSR_LMT+1 Note: <ul style="list-style-type: none">• Bit 18 is reserved if AXI_GM_RD_REQUESTS = 4• Bit 19 is reserved if AXI_GM_MAX_RD_REQUESTS !=16
15:14		RO	00	Reserved
13	ONEKBBE	R_W	0	1 KB Boundary Crossing Enable for the GMAC-AXI Master When set, the GMAC-AXI master performs burst transfers that do not cross 1KB boundary. When reset, the GMAC-AXI master performs burst transfers that do not cross 4KB boundary.
12	AXI_AAL	RO	0	Address-Aligned Beats This bit is read-only bit and reflects Bit 25 (AAL) of the Register 0 (Bus Mode Register). When this bit is set to 1, the GMAC-AXI performs address-aligned burst transfers on both read and write channels.
11:8		RO	OH	Reserved

Bits	Name	Attr	Reset	Description
7	BLEN256	R_W	0	AXI Burst Length 256 When this bit is set to 1, the GMAC-AXI is allowed to select a burst length of 256 on the AXI master interface. This bit is present only when the configuration parameter AXI_BL is set to 256. Otherwise, this bit is reserved and is read-only (RO).
6	BLEN128	R_W	0	AXI Burst Length 128 When this bit is set to 1, the GMAC-AXI is allowed to select a burst length of 128 on the AXI master interface. This bit is present only when the configuration parameter AXI_BL is set to 128 or more. Otherwise, this bit is reserved and is read-only (RO)
5	BLEN64	R_W	0	AXI Burst Length 64 When this bit is set to 1, the GMAC-AXI is allowed to select a burst length of 64 on the AXI master interface. This bit is present only when the configuration parameter AXI_BL is set to 64 or more. Otherwise, this bit is reserved and is read-only (RO)
4	BLEN32	R_W	0	AXI Burst Length 32 When this bit is set to 1, the GMAC-AXI is allowed to select a burst length of 32 on the AXI master interface. This bit is present only when the configuration parameter AXI_BL is set to 32 or more. Otherwise, this bit is reserved and is read-only (RO).
3	BLEN16	R_W	0	AXI Burst Length 16 When this bit is set to 1 or UNDEF is set to 1, the GMAC-AXI is allowed to select a burst length of 16 on the AXI master interface.
2	BLEN8	R_W	0	AXI Burst Length 8 When this bit is set to 1, the GMAC-AXI is allowed to select a burst length of 8 on the AXI master interface. Setting this bit has no effect when UNDEF is set to 1.
1	BLEN4	R_W	0	AXI Burst Length 4 When this bit is set to 1, the GMAC-AXI is allowed to select a burst length of 4 on the AXI master interface. Setting this bit has no effect when UNDEF is set to 1.
0	UNDEF	RO	1	AXI Undefined Burst Length This bit is read-only bit and indicates the complement (invert) value of Bit 16 (FB) in Register 0 (Bus Mode Register). <ul style="list-style-type: none"> When this bit is set to 1, the GMAC-AXI is allowed to perform any burst length equal to or below the maximum allowed burst length programmed in Bits[7:3]. When this bit is set to 0, the GMAC-AXI is allowed to perform only fixed burst lengths as indicated by BLEN256, BLEN128, BLEN64, BLEN32, BLEN16, BLEN8, or BLEN4, or a burst length of 1. If UNDEF is set and none of the BLEN bits is set, then GMAC-AXI is allowed to perform a burst length of 16.

Table 10-13. Ethernet Controller DMA Register 10 for the AXI Bus Mode.

10.3.1.11 DMA_R18_host_tx_descript Register

The Current Host Transmit Descriptor register (**DMA_R18_host_tx_descript**) points to the start address of the current Transmit Descriptor read by the DMA.

Bits	Name	Attr	Reset	Description
31:0	clr_txptr	R	0	Host Transmit Descriptor Address Pointer Cleared on Reset Pointer updated by DMA during operation.

Table 10-14. Ethernet Controller DMA Register 18 Points to the Start of the Host Transmit Descriptor.

10.3.1.12 DMA_R19_host_rx_descript Register

The Current Host Receive Descriptor register (**DMA_R19_host_rx_descript**) points to the start address of the current Receive Descriptor read by the DMA.

Bits	Name	Attr	Reset	Description
31:0	clr_rxptr	R	0	Host Receive Descriptor Address Pointer Cleared on Reset Pointer updated by DMA during operation.

Table 10-15. Ethernet Controller DMA Register 19 Points to the Start of the Receive Descriptor.

10.3.1.13 DMA_R20_host_tx_buffer Register

The Current Host Transmit Buffer Address register (**DMA_R20_host_tx_buffer**) points to the current Transmit Buffer Address being read by the DMA.

Bits	Name	Attr	Reset	Description
31:0	clr_txbuff	R	0	Host Transmit Buffer Address Pointer Cleared on Reset Pointer updated by DMA during operation.

Table 10-16. Ethernet Controller DMA Register 20 Points to the Transmit Buffer Address.

10.3.1.14 DMA_R21_host_rx_buffer Register

The Current Host Receive Buffer Address register (**DMA_R21_host_rx_buffer**) points to the current Receive Buffer address being read by the DMA.

Bits	Name	Attr	Reset	Description
31:0	clr_rxbuff	R	0	Host Receive Buffer Address Pointer Cleared on Reset Pointer updated by DMA during operation.

Table 10-17. Ethernet Controller DMA Register 21 Points to the Receive Buffer Address.

10.3.2 Ethernet Register Details: MAC

10.3.2.1 Register 0 (MAC Configuration Register)

MAC Register 0 for MAC Configuration (**MAC_R0_MAC_config**) establishes receive and transmit operating modes.

Bits	Name	Attr	Reset	Description
31:24				Reserved
23	wd	RW	0	Watchdog Disable When this bit is set, the MAC disables the watchdog timer on the receiver. When this bit is reset, the MAC allows no more than 2048 (10240 if je is set high) bytes of the receiving frame and cuts off any bytes subsequently received.
22	jd	RW	0	Jabber Disable When this bit is set, the MAC disables the jabber timer on the transmitter. When this bit is reset, the MAC cuts-off the transmitter if the Application sends out more than 2048 (10240 if je is set high) bytes of data during transmission.
21	be	RW	0	Frame Burst Enable When this bit is set, the MAC allows frame-bursting during transmission in RMII Half-Duplex mode. This bit is reserved in 10/100-Mbps-only or Full-Duplex-only configurations
20	je	RW	0	Jumbo Frame Enable When this bit is set, MAC allows Jumbo frames of size 9018 bytes (9022 bytes for VLAN tagged frames) without reporting a giant frame error in the receive frame status. Jabber Disable bit should be 1 to transmit jumbo frames.
19:17	ifg	RW	0	Inter Frame Gap These bits control the minimum ifg between frames during transmission. 0x0 - 96 bit times 0x1 - 88 bit times 0x2 - 80 bit times 0x7 - 40 bit times Note that, in Half-Duplex mode, the minimum ifg can be configured up to 64 bit-times (ifg = 100) only.
16	dcrs	RW	0	Disable Carrier Sense (CRS) During Transmission When set high, this bit causes the MAC transmitter to ignore the (R)MII CRS signal during frame transmission in Half-Duplex mode. This request results in no errors generated due to Loss of Carrier or No Carrier during such transmission. When this bit is low, the MAC transmitter generates such errors due to Carrier Sense and will even abort these transmissions.
15	ps	RW	0	Port Select This bit selects the Ethernet line speed 0: For 1000 Mbps operations 1: For 10 or 100 Mbps operations For 10 or 100 Mbps operations, this bit, along with fes bit, selects the exact line speed.

Bits	Name	Attr	Reset	Description
14	fes	RW	0	<p>Speed</p> <p>This bit selects the speed of the MII, RMII, SMII, RGMII, SGMII, or RevMII interface.</p> <p>0: 10 Mbps 1: 100 Mbps</p> <p>This bit is enabled only when the RMII, SMII, RGMII, SGMII, or RevMII interface is enabled during core configuration. This bit generates link speed encoding when TC (bit 24) is set in RMII, SMII, or SGMII mode.</p> <p>This bit is always driven for RevMII. If the MAC is configured with an RMII, SMII, SGMII, or RGMII PHY interface, this bit can optionally be driven as an output signal (MAC_SPEED_O[0]) to reflect the value of this bit in the MAC_SPEED_O signal. Moreover, this bit is reserved when RMII is enabled without enabling the Output Port for Speed Selection option during core configuration.</p>
13	do	RW	0	<p>Disable Receive Own</p> <p>When this bit is set, the MAC disables the reception of frames when the GMII_TXEN_O is asserted in Half-Duplex mode.</p> <p>When this bit is reset, the MAC receives all packets that are provided by the PHY while transmitting. This bit is not applicable (R with default value) if the MAC is operating in Full-Duplex mode.</p>
12	lm	RW	0	<p>Loop-back Mode</p> <p>When this bit is set, the MAC operates in loop-back mode at RMII/MII. The (R)MII Receive clock input (CLK_RX_I) is required for the loopback to function properly, as the Transmit clock is not looped-back internally.</p>
11	dm	RW	0	<p>Duplex mode</p> <p>When this bit is set, the MAC operates in a Full-Duplex mode, allowing it to transmit and receive simultaneously. This bit is R with default value of 1 in Full-Duplex-only configuration.</p>
10	ipc	RW	0	<p>Checksum Offload</p> <p>When this bit is set, the MAC calculates the 16-bit 1's complement of the 1's complement sum of the payload data (16-bit) and sends the result to the application at the end of frame.</p> <p>It also verifies whether the IPv4 Header checksum (assumed to be bytes 25-26 or 29-30 (VLAN-tagged) of Received Ethernet frame) is correct for the received frame and provides the status in the Receive Status Word. In addition, the MAC Core appends the 16-bit Checksum calculated for the payload of the IP datagram (bytes after the header) to the Ethernet frame transferred to the application.</p> <p>When this bit is reset, then this function is disabled.</p>
9	dr	RW	0	<p>Disable Retry</p> <p>When this bit is set, the MAC will attempt only 1 transmission. When a collision occurs on the RMII/MII, the MAC will ignore the current frame transmission and report a Frame Abort with excessive collision error in the transmit frame status.</p> <p>When this bit is reset, the MAC will attempt retries based on the settings of bl. This bit is applicable only in Half-Duplex mode and is reserved in Full-Duplex-only configuration.</p>

Bits	Name	Attr	Reset	Description
8		R	0	Reserved
7	acs	RW	0	<p>Automatic Pad/CRC Stripping</p> <p>When this bit is set, the MAC will strip the Pad/FCS field on incoming frames only if the length's field-value is less than or equal to 1500 bytes. All received frames with a length-field greater than or equal to 1501 bytes will be passed to the application without the Pad/FCS field being stripped.</p> <p>When this bit is reset, the MAC will pass all incoming frames to the Host unmodified.</p>
6:5	bl	RW	0	<p>Back-off Limit</p> <p>The Back-off limit determines the random integer number (r) of slot time delays (512 bit times for 10/100 Mbps) the MAC waits prior to rescheduling a transmission attempt during retries after a collision. This bit is applicable only in Half-Duplex mode and is reserved (R) in Full-Duplex-only configuration.</p> <p>0x0 - k = min (n, 10) 0x1 - k = min (n, 8) 0x2 - k = min (n, 4) 0x3 - k = min (n, 1)</p> <p>where n = retransmission attempt. The random integer r takes the value in the range $0 \leq r < 2^k$</p>
4	dc	RW	0	<p>Deferral Check</p> <p>When this bit is set, the deferral check function is enabled in the MAC. The MAC will issue a Frame Abort status, along with the excessive deferral error bit set in the transmit frame status when the transmit state machine is deferred for more than 24,288 bit-times in 10/100 Mbps mode. If Jumbo frame mode is enabled in 10/100-Mbps mode, the threshold for deferral is 155,680 bit-times. Deferral begins when the transmitter is ready to transmit, but is prevented because of an active carrier-sense (CRS) signal on the RMII/MII. Defer time is not cumulative. If the transmitter defers for 10,000 bit-times, then transmits, collides, backs off, and must defer again after completion of back-off, the deferral timer resets to 0 and restarts.</p> <p>When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. This bit is applicable only in Half-Duplex mode and is reserved (R) in Full-Duplex-only configuration.</p>
3	te	RW	0	<p>Transmitter Enable</p> <p>When this bit is set, the transmit state machine of the MAC is enabled for transmission on the RMII/MII. When this bit is reset, the MAC transmit state machine is disabled after the completion of the transmission of the current frame, and will not transmit any further frames.</p>
2	re	RW	0	<p>Receiver Enable</p> <p>When this bit is set, the receiver state machine of the MAC is enabled for receiving frames from the RMII/MII. When this bit is reset, the MAC receive state machine is disabled after the completion of reception of the current frame, and will not receive further frames.</p>
1:0				Reserved

Table 10-18. Ethernet Controller MAC Register 0 (MAC Configuration) Sets the Operation Mode.

10.3.2.2 MAC_R1_MAC_fm_filter Register

MAC Register 1 is the MAC Frame Filter register (**MAC_R1_MAC_fm_filter**) which contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering. The second level of filtering is performed on the incoming frame, based on other controls such as Pass Bad Frames and Pass Control Frames.

Bits	Name	Attr	Reset	Description
31	ra	RW	0	Receive All When this bit is set, the MAC Receiver module passes to the application all frames received irrespective of whether they pass the address filter or not. The result of the SA/DA Filtering is updated (pass or fail) in the corresponding bits in the Receive Status Word. When this bit is reset, the Receiver module passes to the Application only those frames that pass the SA/DA address filter.
30:11				Reserved
10	hpf	RW	0	Hash or Perfect Filter When set, this bit configures the address filter to pass a frame if it matches either the perfect filtering or the hash filtering requirements as set by hmc or huc bits. When set low and if the huc/hmc bit is set, the frame is passed only if it matches the Hash filter.
9	saf	RW	0	Source Address Filter Enable The MAC core compares the SA field of the received frames with the values programmed in the enabled SA registers (See MAC_R16_MAC_add0_high and MAC_R17_MAC_add0_low). If the comparison matches, then the SAMatch bit of RxStatus Word is set high. When this bit is set high and the SA filter fails, the MAC drops the frame. When this bit is reset, the MAC Core forwards the received frame to the application along with the updated SA Match bit of the RxStatus depending on the SA address comparison.
8	saif	RW	0	SA Inverse Filtering When this bit is set, the Address Check block operates in inverse filtering mode for the SA address comparison. The frames whose SA matches the SA registers will be marked as failing the SA Address filter. (See MAC_R16_MAC_add0_high and MAC_R17_MAC_add0_low). When this bit is reset, frames whose SA does not match the SA registers will be marked as failing the SA Address filter.

Bits	Name	Attr	Reset	Description
7:6	pcf	RW	0	<p>Pass Control Frames</p> <p>These bits control the forwarding of all control frames (including unicast and multicast PAUSE frames). Note that the processing of PAUSE control frames depends only on rfe of MAC_R6_flow_control[2].</p> <p>0x0, 0x1 - MAC prevents all control frames from reaching the application</p> <p>0x2 - MAC forwards all control frames to the application even if they fail the Address Filter</p> <p>0x3 - MAC only forwards control frames that pass the Address Filter.</p>
5	dbf	RW	0	<p>Disable Broadcast Frames</p> <p>When this bit is set, the AFM module filters all incoming broadcast frames.</p> <p>When this bit is reset, the AFM module passes all received broadcast frames.</p>
4	pm	RW	0	<p>Pass All Multicast</p> <p>When set, this bit indicates that all received frames with a multicast destination address (first bit in the destination address field is 1) are passed.</p> <p>When reset, filtering of multicast frame depends on hmc bit.</p>
3	daif	RW	0	<p>DA Inverse Filtering</p> <p>When this bit is set, the Address Check block operates in inverse filtering mode for DA address comparison of both unicast and multicast frames.</p> <p>When reset, normal filtering of frames is performed.</p>
2	hmc	RW	0	<p>Hash MultiCast</p> <p>When set, the MAC performs destination address filtering of received multicast frames according to the hash table.</p> <p>When reset, the MAC performs perfect destination address filtering for multicast frames (i.e, compares the DA field with the values programmed in DA registers.) See MAC_R16_MAC_add0_high and MAC_R17_MAC_add0_low.</p>
1	huc	RW	0	<p>Hash UniCast</p> <p>When set, the MAC performs destination address filtering of unicast frames according to the hash table.</p> <p>When reset, the MAC performs perfect destination address filtering for unicast frames; i.e, compares the DA field with the values programmed in DA registers. (See MAC_R16_MAC_add0_high and MAC_R17_MAC_add0_low).</p>
0	pr	RW	0	<p>Promiscuous Mode</p> <p>When this bit is set, the Address Filter module passes all incoming frames regardless of their destination or source address. The SA/DA Filter Fails status bits of the Receive Status Word will always be cleared when pr is set.</p>

Table 10-19. Ethernet Controller MAC Register 1 (MAC Frame Filter) Contains Frame Filtering Controls.

10.3.2.3 MAC_R2_hash_high Register

The 64-bit Hash Table is used for group address filtering. For hash filtering, the contents of the destination ad-

dress in the incoming frame is passed through the CRC logic, and the upper 6 bits of the CRC register are used to index the contents of the Hash table. The most significant bit determines the register to be used (Hash Table High or Hash Table Low), and the other 5 bits determine the bit within the register.

A hash value of 0 results in the selection of bit 0 of the register, and a value of 0x3F results in the selection of bit 31. For example, if the DA of the incoming frame is received as 0x1F52419CB6AF (0x1F is the first byte received on RMII interface), then the internally calculated 6-bit hash value is 0x2C and the **MAC_R2_hash_table_high** register bit[12] is checked for filtering. If the DA of the incoming frame is received as 0xA00A98000045, then the calculated 6-bit hash value is 0x07 and **MAC_R3_hash_table_low** register bit[7] is checked for filtering. If the corresponding bit value of the register is 1, the frame is accepted. Otherwise, it is rejected.

If the **pm** (Pass All Multicast) bit is set in **MAC_R1_MAC_fm_filter**, then all multicast frames are accepted regardless of the multicast hash values. If the Hash Table register is configured to be double-synchronized to the (R) MII clock domain, the synchronization is triggered only when Bits[31:24] (in Little Endian mode) or Bits[7:0] (in Big Endian mode) of the Hash Table High/Low Registers are written to. Please note that consecutive writes to these registers should be performed only after at least four clock cycles in the destination clock domain when double-synchronization is enabled.

The table below provides details for the MAC Hash Table High Register (**MAC_R2_hash_table_high**). This register contains the higher 32 bits of the Hash table.

Bits	Name	Attr	Reset	Description
31:0	hth	RW	0	Hash Table High This field contains the upper 32 bits of Hash table.

Table 10-20. Ethernet Controller MAC Register 2 Contains the High 32 Bits of the Multicast Hash Table.

10.3.2.4 **MAC_R3_hash_table_low** Register

The Hash Table Low register (**MAC_R3_hash_table_low**) contains the lower 32 bits of the Hash table.

Bits	Name	Attr	Reset	Description
31:0	htl	RW	0	Hash Table Low This field contains the lower 32 bits of Hash table.

Table 10-21. Ethernet Controller MAC Register 3 Contains the Low 32 bits of the Multicast Hash Table.

10.3.2.5 **MAC_R4_RMII_address** Register

The RMII Address register (**MAC_R4_RMII_address**) controls the management cycles to the external PHY through the management interface.

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:11	pa	RW	0	Physical Layer Address This field indicates which of the 32 possible PHY devices are being accessed.
10:6	gr	RW	0	RMII Register These bits select the desired RMII register in the selected PHY device.

Bits	Name	Attr	Reset	Description
5				Reserved
4:2	cr	RW	0	<p>CSR Clock Range The CSR Clock Range selection determines the Core Clock frequency and is used to select the frequency of the MDC clock: Selection Core Clock / MDC Clock</p> <p>0x0 - 60 MHz to 100 MHz Core Clock/42 0x1 - 100 MHz to 150 MHz Core Clock/62 0x2 - 20 MHz to 35 MHz Core Clock/16 0x3 - 35 MHz to 60 MHz Core Clock/26 0x4 - 150 MHz to 250 MHz Core Clock/102 0x5 - 250 MHz to 300 MHz Core Clock/122 0x6, 0x7 - Reserved</p>
1	gw	RW	0	<p>RMII Write When set, this bit indicates to the PHY that this will be a Write operation using the RMII Data register (MAC_R5_RMII_data). If this bit is not set, this will be a Read operation, resulting in the placement of data in the RMII Data register.</p>
0	gb	R_WS_SC	0	<p>RMII Busy This bit should read a logic 0 before writing to MAC_R4_RMII_address and the RMII Data register (MAC_R5_RMII_data). This bit must also be set to 0 during a Write to MAC_R4_RMII_address. During PHY register access, this bit will be set to 1 by the application to indicate that a Read or Write access is in progress. MAC_R5_RMII_data (RMII Data) should be kept valid until this bit is cleared by the MAC during a PHY Write operation. The MAC_R5_RMII_data bit is invalid until cleared by the MAC during a PHY Read operation. The MAC_R4_RMII_address should not be written to until this bit is cleared.</p>

Table 10-22. Ethernet Controller MAC Register 4 (RMII Address) for Management Cycles to External PHY.

10.3.2.6 **MAC_R5_RMII_data** Register

The RMII Data register (**MAC_R5_RMII_data**) stores Write data to be written to the PHY register located at the address specified in **MAC_R4_RMII_address**. **MAC_R5_RMII_data** also stores Read data from the PHY register located at the address specified by **MAC_R4_RMII_address**.

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:0	gd	RW	0	<p>RMII Data This contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY prior to a Management Write operation.</p>

Table 10-23. Ethernet Controller MAC Register 5 (RMII Data) Stores Read or Write Data.

10.3.2.7 MAC_R6_flow_control Register

The Flow Control register (**MAC_R6_flow_control**) controls the generation and receipt of the Control (Pause Command) frames by the MAC's flow-control module. A Write made to a register with the Busy bit (**fcb**) set to 1 triggers the Flow Control block to generate a Pause Control frame. The fields of the control frame are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field **pt** of the control frame. The Busy bit remains set until the control frame is transferred onto the cable. The Host must ensure that the Busy bit is cleared before writing to the register.

Bits	Name	Attr	Reset	Description
31:16	pt	RW	0	<p>Pause Time This field holds the value to be used in the Pause Time field in the transmit control frame. If the Pause Time bit is configured to be double-synchronised to the (R)MII clock domain, then consecutive writes to this register should be performed only after at least 4 clock cycles in the destination clock domain.</p>
15:8				Reserved
7	dzpq	RW	0	<p>Disable Zero-Quanta Pause When set, this bit disables the automatic generation of Zero-Quanta Pause Control frames on the deassertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal SBD_FLOWCTRL_I/MTI_FLOWCTRL_I). When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled.</p>
6				Reserved
5:4	plt	RW	0	<p>Pause Low threshold This field configures the threshold of the PAUSE timer at which the input flow control signal MTI_FLOWCTRL_I (or SBD_FLOWCTRL_I) is checked for automatic re-transmission of the PAUSE Frame. The threshold values should always be greater than the Pause Time pt configured in Bits[31:16]. For example, if pt = 0x100 (256 slot-times), and plt = 01, then a second PAUSE frame is automatically transmitted if the MTI_FLOWCTRL_I signal is asserted at 228 (256-28) slot-times after the first Pause-frame is transmitted. Selection Threshold: 0x0 - Pause Time at 4 Slot time 0x1 - Pause Time at 28 Slot time 0x2 - Pause Time at 144 Slot time 0x3 - Pause Time at 256 Slot time Slot time is defined as the time required to transmit 512 bits (64 bytes) on the RMII/MII interface.</p>
3	up	RW	0	<p>Unicast Pause Frame detect When this bit is set, the MAC will detect the Pause frames with the station's unicast address specified in MAC Address0 High Register (MAC_R18_MAC_add1_high) and MAC Address0 Low Register (MAC_R17_MAC_add0_low), in addition to detecting Pause frames with the unique multicast address. When this bit is reset, the MAC will detect only a Pause frame with the unique multicast address specified in the 802.3x standard.</p>

Bits	Name	Attr	Reset	Description
2	rfe	RW	0	Receive Flow Control Enable When this bit is set, the MAC will decode the received Pause frame and disable its transmitter for a specified (Pause Time pt) period. When this bit is reset, the decode function of the Pause frame is disabled.
1	tfe	RW	0	Transmit Flow Control Enable In Full-Duplex mode, when this bit is set, the MAC enables the flow-control operation to transmit Pause frames. When this bit is reset, the flow-control operation in the MAC is disabled, and the MAC will not transmit any Pause frames. In Half-Duplex mode, when this bit is set, the MAC enables the back-pressure operation. When this bit is reset, the backpressure feature is disabled.
0	fcb/bpa	RW	0	Flow Control Busy/Back-Pressure Activate This bit initiates a Pause Control frame in Full-duplex mode and activates the back-pressure function in Half-duplex mode if the tfe bit is set. In full-duplex mode, this bit should be read as 0 before writing to the Flow Control register MAC_R6_flow_control . To initiate a Pause control frame, the Application must set this bit to 1. During a transfer of the Control Frame, this bit will continue to be set to signify that a frame transmission is in progress. After the completion of Pause control frame transmission, the MAC will reset this bit to 0. MAC_R6_flow_control should not be written to until this bit is cleared. In Half-duplex mode, when this bit is set (and tfe is set), then back-pressure is asserted by the MAC Core. With back-pressure enabled, when the MAC receives a new frame, the transmitter begins sending a JAM pattern resulting in a collision. This control register bit is logically OR'ed with the MTI_FLOWCTRL_I input signal for the back-pressure function. When the MAC is configured to full-duplex mode, the bpa is automatically disabled.

Table 10-24. Ethernet Controller MAC Register 6 (Flow Control) Controls Generation of Control Frames.

10.3.2.8 **MAC_R7_VLAN_tag** Register

The VLAN Tag register (**MAC_R7_VLAN_tag**) contains the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the receiving frame (Length/Type) with 16'h8100, and the following two bytes are compared with the VLAN tag; if a match occurs, it sets the received VLAN bit in the receive frame status. The legal length of the frame is increased from 1518 bytes to 1522 bytes. If the VLAN Tag register is configured to be double-synchronized to the (R)MII clock domain, then consecutive writes made to these registers should be performed only after at least four clock cycles in the destination clock domain.

Bits	Name	Attr	Reset	Description
31:17				Reserved

Bits	Name	Attr	Reset	Description
16	etv	RW	0	Enable 12-Bit VLAN Tag Comparison When this bit is set, a 12-bit VLAN identifier, rather than the complete 16-bit VLAN tag, is used for comparison and filtering. Bits[11:0] of the VLAN tag are compared with the corresponding field in the received VLAN-tagged frame. When this bit is reset, all 16 bits of the received VLAN frame's fifteenth and sixteenth bytes are used for comparison.
15:0	vl	RW	0	VLAN Tag Identifier This contains the 802.1Q VLAN Tag to identify VLAN frames, and is used as a comparison with the 15th and 16th bytes of the received VLAN frames. If vl is all-zeros, then the MAC does not check the 15th and 16th bytes for VLAN tag comparison.

Table 10-25. Ethernet Controller MAC Register 7 (VLAN Tag) Identifies IEEE 802.1Q VLAN Frames .

10.3.2.9 MAC_R8_version Register

The MAC Register 8 is the Version register (**MAC_R8_version**) which identifies the Ethernet Controller version.

Bits	Name	Attr	Reset	Description
31:16				Reserved
7150	version	R	0x2034	Version

Table 10-26. Ethernet Controller MAC Register 8 Identifies the Version of the Core.

10.3.2.10 MAC_R14_interrupt_status Register

The MAC Interrupt Status register (**MAC_R14_interrupt_status**) identifies the events in the MAC-CORE that can generate interrupts.

Bits	Name	Attr	Reset	Description
31:4				Reserved
3	mgc_wake	R	0	This bit is set whenever a Magic packet or Wake-on-LAN frame is received in Power-Down mode. This bit is cleared when both bits[6:5] are cleared due to a read operation made to the PMT Control and Status Register as described in Section 3.8.1.1 of the <i>Ethernet MAC Universal Databook</i> from Synopses.
2:0				Reserved

Table 10-27. Ethernet Controller MAC Register 14 Provides Interrupt Status.

10.3.2.11 MAC_R15_interrupt_mask Register

The bits of the MAC Interrupt Mask register (**MAC_R15_interrupt_mask**) enable the user to mask the interrupt signal due to the corresponding event in the Interrupt Status register **MAC_R14_interrupt_status**.

Bits	Name	Attr	Reset	Description
31:4				Reserved
3	int_off	RW	0	Disables interrupt generation when set to 1
2:0				Reserved

Table 10-28. Ethernet Controller MAC Register 15 is the Interrupt Mask Register.

10.3.2.12 MAC_R16_MAC_add0_high Register

The MAC Address0 High register (**MAC_R16_MAC_add0_high**) holds the upper 16 bits of the first 6-byte MAC address of the station. Note that the first DA byte that is received on the (R)MII interface corresponds to the least significant byte (Bits [7:0]) of the MAC Address Low register (**MAC_R17_MAC_add0_low**). For example, if 0x112233445566 is received (0x11 is the first byte) on the (R)MII as the destination address, then the MAC Address0 Register [47:0] is compared with 0x665544332211.

If the MAC address registers are configured to be double-synchronized to the (R)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in Little Endian mode) or Bits[7:0] (in Big Endian mode) of the MAC Address Low Register (**MAC_R17_MAC_add0_low**) are written to. Please note that consecutive writes made to the Address Low Register should be performed only after at least four clock cycles in the destination clock domain for proper synchronization updates.

Bits	Name	Attr	Reset	Description
31	mo	R	1	Always 1
30:16				Reserved
15:0	A[47:32]	RW	0xFFFF	MAC Address0 [47:32] This field contains the upper 16 bits (47:32) of the first 6-byte MAC address. This is used by the MAC for filtering of received frames and for inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.

Table 10-29. Ethernet Controller MAC Register 16 is the MAC Address0 High Register.

10.3.2.13 MAC_R17_MAC_add0_low Register

The MAC Address0 Low register (**MAC_R17_MAC_add0_low**) holds the lower 32 bits of the first 6-byte MAC address of the station.

Bits	Name	Attr	Reset	Description
31:0	A[31:0]	RW	0xFFFF_FFFF	MAC Address0 [31:0] This field contains the lower 32 bits of the first 6-byte MAC address. This is used by the MAC for filtering of received frames and for inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.

Table 10-30. Ethernet Controller MAC Register 17 is the MAC Address0 Low Register.

10.3.2.14 MAC_R18_MAC_add1_high Register

The MAC Address1 High register (**MAC_R18_MAC_add1_high**) holds the upper 16 bits of the second 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (R)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in Little Endian mode) or Bits[7:0] (in Big Endian mode) of the MAC Address Low Register (**MAC_R19_MAC_add1_low**) are written to. Please note that consecutive writes made to the Address Low Register should be performed only after at least 4 clock cycles in the destination clock domain for proper synchronization updates.

Bits	Name	Attr	Reset	Description
31	ae	RW	0	Address Enable When this bit is set, the Address filter module uses the second MAC address for perfect filtering. When reset, the address filter module will ignore the address for filtering.
30	sa	RW	0	Source Address When this bit is set, the MAC Address1[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address1[47:0] is used to compare with the DA fields of the received frame.
29:24	mbc	RW	0	Mask Byte Control These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC core does not compare the corresponding byte of received DA/SA with the contents of MAC Address1 registers. Each bit controls the masking of bytes as follows: Bit 24 - MAC_R19_MAC_add1_low[7:0] Bit 27 - MAC_R19_MAC_add1_low[31:24] Bit 28 - MAC_R18_MAC_add1_high[7:0] Bit 29 - MAC_R18_MAC_add1_high[15:8]
23:16				Reserved
15:0	a	RW	0xFFFF	MAC Address1 [47:32] This field contains the upper 16 bits (47:32) of the second 6-byte MAC address.

Table 10-31. Ethernet Controller MAC Register 18 is the MAC Address1 High Register.

Note:

- The descriptions for Register 20 (**MAC_R20_MAC_add2_high**) are as shown in the table above for **MAC_R18_MAC_add1_high**.

10.3.2.15 MAC_R19_MAC_add1_low Register

The MAC Register19 or MAC Address1 Low register (**MAC_R19_MAC_add1_low**) holds the lower 32 bits of the second 6-byte MAC address of the station.

Bits	Name	Attr	Reset	Description
31:0	A	RW	0xFFFF_FFFF	MAC Address1 [31:0] This field contains the lower 32 bits of the second 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

Table 10-32. Ethernet Controller MAC Register 19 is the MAC Address1 Low Register.

Note:

- The descriptions for Register 21 (**MAC_R20_MAC_add2_low**) are as shown in Table 21-31 for **MAC_R19_MAC_add1_low**.

10.4 Ethernet: Descriptors

Receive Descriptors and Transmit Descriptors are detailed in [Section 10.4.1](#) and [Section 10.4.2](#), respectively. Please be advised that the register descriptions are intended for Little Endian mode.

10.4.1 Ethernet Descriptors: Receive

The MAC Subsystem requires at least two descriptors when receiving a frame. The Receive state machine of the DMA (in the MAC Subsystem) always attempts to acquire an extra descriptor in anticipation of an incoming frame (The size of the incoming frame is unknown). Before the RxDMA closes a descriptor, it will attempt to acquire the next descriptor even if no frames are received. In a single-descriptor (receive) system, the subsystem will generate a descriptor error if the receive buffer is unable to accommodate the incoming frame and the next descriptor is not owned by the DMA. Thus, the Host is forced to increase either its descriptor pool or the buffer size. Otherwise, the subsystem begins dropping all incoming frames.

The figure below illustrates the receive descriptor format in Little Endian mode with a 32-bit data bus. Each descriptor is provided with two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory management schemes. Descriptor addresses must be aligned to 32 bits.

	31	0
RDES0	OWN Bit	Status
RDES1	Control Bits	Byte Count Buffer 2
RDES2	Buffer 1 Address	
RDES3	Buffer 2 Address / Next Descriptor Address	

Figure 10-2. Ethernet Receive Descriptor Format in Little Endian Mode with a 32-Bit Data Bus.

10.4.1.1 Ethernet Receive Descriptor 0

Receive Descriptor 0 (RDES0) contains the received frame status, frame length, and descriptor ownership information. If the DUT is configured for Big Endian mode with a 32-bit data bus width, then byte lanes 3 and 0 are swapped while byte lanes 1 and 2 are swapped; that is, bits 31:24 will be available on data bus[7:0], and vice-versa.

Bits	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA of the MAC Subsystem. When this bit is reset, it indicates that the descriptor is owned by the Host. The DMA clears this bit when it either completes the frame reception or when the buffers that are associated with this descriptor are full.
30	AFM: Destination Address Filter Fail When set, this bit indicates that a frame failed in the DA Filter in the MAC Core.
29:16	FL: Frame Length These bits indicate the byte length of the received frame that was transferred to Host memory (including CRC). This field is valid only when the Last Descriptor (RDES0[8]) is set and the Descriptor Error (RDES0[14]) is reset. The Frame Length also includes the 2 bytes appended to the Ethernet Frame when IP Checksum calculation is enabled and the received frame is not a MAC Control frame. This field is valid when Last Descriptor (RDES0[8]) is set. When Last Descriptor and Error Summary bits are not set, this field will indicate the current number of bytes transferred for that frame.
15	ES: Error Summary Indicates the logical OR of the following bits: RDES0[1] - CRC Error RDES0[3] - Receive Error RDES0[4] - Watchdog Timeout RDES0[6] - Late Collision RDES0[10] - Giant Frame RDES0[11] - Overflow Error RDES0[14] - Descriptor Error. This field is valid only when the Last Descriptor (RDES0[8]) is set.
14	DE: Descriptor Error When set, this bit indicates a frame truncation occurred on a frame that exceeded the current descriptor buffers, and that the DMA does not own the Next Descriptor. This field is valid only when the Last Descriptor (RDES0[8]) is set.
13	SAF: Source Address Filter Fail When set, this bit indicates that the SA field of the frame failed the SA Filter in the MAC Core.
12	LE: Length Error When set, this bit indicates that the actual length of the frame received does not match the information in the Length/ Type field. This bit is valid only when the Frame Type (RDES0[5]) bit is reset.
11	OE: Overflow Error When set, this bit indicates that the received frame was damaged due to buffer overflow in MTL.
10	VLAN : VLAN tag When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the MAC Core.
9	FS: First Descriptor When set, this bit indicates that the descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the following Descriptor contains the beginning of the frame.
8	LS: Last Descriptor When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame

Bits	Description
7	<p>Timestamp Available, IP Checksum Error (Type1), or Giant Frame When Advanced Timestamp feature is present, when set, this bit indicates that a snapshot of the Timestamp is written in descriptor words 6 (RDES6) and 7 (RDES7). This is valid only when the Last Descriptor bit (RDES0[8]) is set.</p> <p>When IP Checksum Engine (Type 1) is selected, this bit (when set) indicates one of the following:</p> <ul style="list-style-type: none"> - The 16-bit IPv4 header checksum calculated by the core did not match the received checksum bytes. - The header checksum checking is bypassed for non-IPv4 frames. <p>Otherwise, this bit (when set) indicates the Giant Frame Status. Giant frames are larger than 1,518-byte (or 1,522-byte for VLAN or 2,000-byte when Bit 27 of the MAC Configuration register is set) normal frames and larger than 9,018-byte (9,022-byte for VLAN) frame when Jumbo Frame processing is enabled.</p>
6	<p>LC: Late Collision When set, this bit indicates that late collision has occurred while receiving the frame in Half-duplex mode.</p>
5	<p>FT: Frame Type When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 16'h0600). When this bit is reset, it indicates that the received frame is an IEEE 802.3x standard frame. This bit is not valid for runt frames less than 14 bytes.</p>
4	<p>RWT: Receive Watchdog Timeout When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout.</p>
3	<p>RE: Receive Error When set, this bit indicates that the RMII_PHY_RXER_I signal is asserted while RMII_PHY_RXDV_I signal is asserted during frame reception. This error also includes carrier extension errors in RMII and Half-duplex mode. Errors can occur due to lack of an extension, or an error (e.g., rxd != 0f) during extension.</p>
2	<p>DE: Dribble Bit Error When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in MII Mode.</p>
1	<p>CE: CRC Error When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8]) is set.</p>
0	<p>Extended Status Available/Rx MAC Address When either Advanced Timestamp or IP Checksum Offload (Type 2) is present, this bit (when set) indicates that the extended status is available in descriptor word 4 (RDES4). This is valid only when the Last Descriptor bit (RDES0[8]) is set. This bit is invalid when Bit 30 is set.</p> <p>When IP Checksum Offload (Type 2) is present, this bit is set even when the IP Checksum Offload engine bypasses the processing of a received frame. The bypassing may be because of non-IP frame or IP frame with a non-TCP/UDP/ICMP payload.</p> <p>When Advance Timestamp Feature or IPC Full Offload is not selected, this bit indicates Rx MAC Address status. When set, this bit indicates that the Rx MAC Address registers value (1 to 15) matched the frame's DA field. When reset, this bit indicates that the Rx MAC Address Register 0 value matched the DA field.</p>

Table 10-33. Ethernet Receive Descriptor 0 Register.

10.4.1.2 Ethernet Receive Descriptor 1

Receive Descriptor 1 (RDES1) contains the buffer sizes and other bits which control the descriptor chain/ring.

Bits	Description
31	Disable Interrupt on Completion When set, this bit will prevent the setting of the receive interrupt (RI) (CSR5[6]) bit of the Status Register for the received frame that ends in the buffer pointed to by this descriptor. This, in turn, will disable the assertion of the interrupt to Host due to an RI for that frame.
30:26	Reserved
25	RER: Receive End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a Descriptor Ring.
24	RCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When RDES1[24] is set, RBS2 (RDES1[21-11]) should be all zeros. RDES1[25] takes precedence over RDES1[24].
23:22	Reserved
21:11	RBS2: Receive Buffer 2 Size These bits indicate the second data buffer byte size. The buffer size must be a multiple of 4/8/16 depending upon the bus-widths (32/64/128), even if the value of RDES2 (buffer1 address pointer) is not aligned to the bus-width. In the case where the buffer size is not a multiple of 4/8/16, the resulting behavior is undefined. This field is not valid if RDES1[24] is set.
10:0	RBS1: Receive Buffer 1 Size Size indicates the first data buffer's byte size. The buffer size must be a multiple of 4/8/16 depending upon bus-width (32/64/128), even if the value of RDES2 (buffer1 address pointer) is not aligned. In the case where the buffer size is not a multiple of 4/8/16, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor depending on the value of RCH (bit 24).

Table 10-34. Ethernet Receive Descriptor 1 Register.

10.4.1.3 Ethernet Receive Descriptor 2

Receive Descriptor 2 (RDES2) contains the address pointer to the first data buffer in the descriptor.

Bits	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There are no limitations on buffer address alignment with the exception of the following condition: The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. Note that the DMA performs a write operation with the RDES2[3/2/1:0] bits as "zero" during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[3/2/1:0] (corresponding to bus-widths of 128/64/32) if the address pointer is to a buffer where the middle or last part of the frame is stored.

Table 10-35. Ethernet Receive Descriptor 2 Register.

10.4.1.4 Ethernet Receive Descriptor 3

Receive Descriptor 3 (RDES3) contains the address pointer either to the second data buffer in the descriptor or the next descriptor.

Bits	Description
31:0	<p>Buffer 2 Address Pointer (Next Descriptor Address)</p> <p>These bits indicate the physical address of Buffer 2 when descriptor chaining is used. If the Second Address Chained (RDES1[24]) bit is set, then this address contains the pointer to the physical memory where the Next Descriptor is present. If RDES1[24] is set, then the buffer (Next Descriptor) address pointer must be bus-width aligned ($RDES3[3/21:0] = 0$ corresponding to bus-width of 128/64/32. Internally the LSBs are ignored). However, when RDES1[24] is reset, there are no limitations on the RDES3 value with the exception of the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores $RDES3[3/2/1:0]$ (corresponding to bus-widths of 128/64/32) if the address pointer is to a buffer where the middle or last part of the frame is stored.</p>

Table 10-36. Ethernet Receive Descriptor 3 Register.

10.4.2 Ethernet Descriptors: Transmit

The figure below illustrates the transmit descriptor format in Little Endian mode with a 32-bit data bus. Each descriptor has two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory management schemes. Descriptor addresses must be aligned to 32 bits.

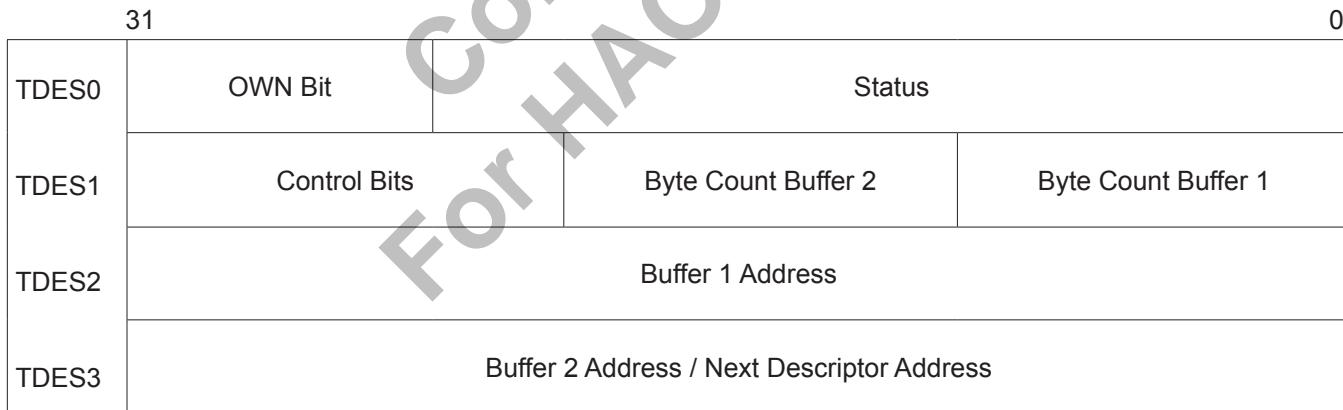


Figure 10-3. Ethernet Transmit Descriptor Format in Little Endian Mode with a 32-Bit Data Bus.

10.4.2.1 Ethernet Transmit Descriptor 0

Transmit Descriptor 0 (TDES0) contains the transmitted frame status and the descriptor ownership information.

Bits	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, it indicates that the descriptor is owned by the Host. The DMA clears this bit when it either completes frame transmission or when the buffers allocated in the descriptor are empty. The ownership bit of the First Descriptor of the frame should be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.
30:16	Reserved
15	ES: Error Summary Indicates the logical OR of the following bits: TDES0[14] - Jabber Timeout TDES0[13] - Frame Flush TDES0[11] - Loss of Carrier TDES0[10] - No Carrier TDES0[9] - Late Collision TDES0[8] - Excessive Collision TDES0[2] - Excessive Deferral TDES0[1] - Underflow Error
14	JT: Jabber Timeout When set, this bit indicates the MAC transmitter has experienced a Jabber Timeout. This bit will be set only when the JB bit of the MAC configuration register is not disabled (deasserted).
13	FF: Frame Flushed When set, this bit indicates that the DMA/MTL flushed the frame due to a software flush command issued by the CPU.
12	Reserved
11	LC: Loss of Carrier When set, this bit indicates that a Loss of Carrier occurred during frame transmission (that is, the RMII_PHY_CRS_I signal was inactive for one or more transmit clock periods during frame transmission). This bit is valid only for the frames transmitted without collision and when the MAC is operating in Half-Duplex Mode.
10	NC: No Carrier When set, this bit indicates that the carrier sense signal from the PHY was not asserted during transmission.
9	LC: Late Collision When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte-times including Preamble in MII Mode and 512 byte-times including Preamble and Carrier Extension in RMII Mode). Not valid if Underflow Error is set.
8	EC: Excessive Collision When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the DR (Disable Retry) bit in the MAC Configuration Register is set, this bit is set after the first collision and the transmission of the frame is aborted.
7	VF: VLAN Frame When set, this bit indicates that the transmitted frame was a VLAN-type frame.
6:3	CC: Collision Count This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is invalid when the Excessive Collisions bit (TDES0[8]) is set.

Bits	Description
2	ED: Excessive Deferral When set, this bit indicates that the transmission has ended due to excessive deferral of over 24,288 bit-times (155,680 bits times in Jumbo Frame-enabled mode) if the Deferral Check (DC) bit is set high in the MAC Control Register.
1	UF: Underflow Error When set, this bit indicates that the MAC aborted the frame because data arrived late from the Host memory, and the DMA encountered an empty Transmit Buffer while transmitting the frame. The transmission process enters a suspended state and sets both Transmit Underflow (MAC_R5_RMII_data[5]) and Transmit Interrupt (MAC_R5_RMII_data[0]).
0	DB: Deferred Bit When set, this bit indicates that the MAC will defer before transmission due to the presence of a carrier signal. This bit is valid only in Half-Duplex Mode.

Table 10-37. Ethernet Transmit Descriptor 0 Register.

10.4.2.2 Ethernet Transmit Descriptor 1

Transmit Descriptor 1 (TDES1) contains the buffer sizes and other bits to control the descriptor chain/ring and frame being transferred.

Bits	Description
31	IC: Interrupt on Completion When set, this bit sets Transmit Interrupt (MAC_R5_RMII_data[0]) after the present frame has been transmitted.
30	LS: Last Segment When set, this bit indicates that the buffer contains the last segment of the frame.
29	FS: First Segment When set, this bit indicates that the buffer contains the first segment of a frame.
28:27	Reserved
26	DC: Disable CRC When set, the MAC does not append the Cyclic Redundancy Check (CRC) to the end of the transmitted frame. This bit is valid only when the First Segment (TDES1[29]) bit is set.
25	TER: Transmit End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a Descriptor Ring.
24	TCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When RDES1[24] is set, RBS2 (RDES1[21-11]) should be all zeros. RDES1[25] takes precedence over RDES1[24].
23	DP: Disable Padding When set, the MAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes and the CRC field is added despite the state of the DC (TDES1[26]) bit. This bit is valid only when the First Segment (TDES1[29]) bit is set.
22	Reserved
21:11	TBS2: Transmit Buffer 2 Size These bits indicate the Second Data Buffer byte size. This field is invalid if TDES1[24] is set.

Bits	Description
10:0	TBS1: Transmit Buffer 1 Size These bits indicate the First Data Buffer byte size. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor depending on the value of TCH (bit 24).

Table 10-38. Ethernet Transmit Descriptor 1 Register.

10.4.2.3 Ethernet Transmit Descriptor 2

Transmit Descriptor 2 (TDES2) contains the address pointer to the first buffer of the descriptor.

Bits	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There is no limitation on buffer address alignment.

Table 10-39. Ethernet Transmit Descriptor 2 Register.

10.4.2.4 Ethernet Transmit Descriptor 3

Transmit Descriptor 3 (TDES3) contains the address pointer either to the second buffer of the descriptor or the next descriptor.

Bits	Description
31:0	Buffer 2 Address Pointer (Next Descriptor Address) Indicates the physical address of Buffer 2 when descriptor chaining is used. If the Second Address Chained (TDES1[24]) bit is set, then this address contains the pointer to the physical memory where the Next Descriptor is present. The buffer address pointer must be aligned to the bus-width only when TDES1[24] is set. (Internally, LSBs are ignored.)

Table 10-40. Ethernet Transmit Descriptor 3 Register.

11. USB DEVICE CONTROLLER

This chapter describes the A12 USB Device Controller as follows:

- [\(Section 11.1\) USB Device: Overview](#)
- [\(Section 11.2\) USB Device: Clocking](#)
- [\(Section 11.3\) USB Device: Host / Device Select](#)
- [\(Section 11.4\) USB Device: Force USB Boot Mode](#)
- [\(Section 11.5\) USB Device: Device Controller Operations](#)
- [\(Section 11.6\) USB Device: Controller / CSR Registers](#)
- [\(Section 11.7\) USB Device: Initialization](#)

11.1 USB Device: Overview

The A12 chip provides an interface for a USB Device Controller (UDC). The features of the USB Device include:

- Compliance with USB2.0 and USB1.1.
- Support for High-Speed, Full-Speed and Low-Speed operations.
- Built-in DMA Controller with Scatter-Gather support.
- A total of three configurations (i.e., configuration 0, plus configurations 1/2).
- Support for up to six interfaces (and seven alternate interfaces) for configurations 1/2.
- Support for five in/out endpoints as well as endpoint 0 (the mandatory default control endpoint EP0 per USB standards).
- 256 * 4 bytes of Rx FIFO (RAM) and 576 * 4 bytes of Tx FIFO (RAM).

The USB Device Controller is situated on the AHB bus, and can function both as a master (i.e., in DMA mode), and as a slave (i.e., in Slave mode).

In DMA mode, the subsystem acts as a master on the AHB to read and write the memory structures in the system memory. In Slave mode, the subsystem responds to AHB transactions from the application (the AHB master).

In both modes, the subsystem acts as a slave for Control and Status Register (CSR) access. In Slave mode, the internal FIFOs are memory-mapped.

DMA mode should be used with applications that require software controllability and throughput. Slave mode should be used with applications that are software-independent and can be functionally implemented via hardware.

Both of these modes are explained in detail in the following sections.

11.2 USB Device: Clocking

A12 system software generally controls the USB PHY clock GCLK_USB_PHY.

11.3 USB Device: Host / Device Select

The A12 USB Device Controller can be configured to function in Master (DMA) or Slave mode. Use power-on configuration (POC) bits 28 and 29 to configure the USB interface accordingly ([Section 2.3](#)).

11.4 USB Device: Force USB Boot Mode

The A12 chip provides a Force USB power-on boot mode. Refer to [Section 2.3](#) for further detail.

11.5 USB Device: Device Controller Operations

The USB Controller Subsystem supports two modes of operation. The first is a DMA-based implementation in which the subsystem can master the AHB bus for data transfers ([Section 11.5.1](#)). The second is a slave implementation in which the subsystem is slaved to the application and the application bus master(s) reads data from, or writes data to, the memory-mapped subsystem FIFOs ([Section 11.5.2](#)). In both DMA and Slave modes all data transfers are interrupt-driven.

In the USB Controller Subsystem ([Section 11.5.3](#)), there are two hosts: the USB host, which initiates USB traffic, and the application, which responds to USB host commands from the device side. The USB Controller Subsystem is used for device-type applications only. USB host functionality is beyond the scope of this document.

11.5.1 Operations: DMA Mode

In DMA mode, prior to the start of an action, the application must initialize the buffer descriptor chains for all active endpoints, and must program the necessary CSRs in the UDC and subsystem during the USB reset. The setup and data descriptor pointers must be 16-byte aligned and the data buffer pointer in the descriptor information field must be 8-byte aligned.

11.5.2 Operations: Slave Mode

In Slave mode, the subsystem operates as a slave and the application bus master(s) reads data from or writes data to the memory-mapped subsystem FIFOs. All data transfers are interrupt-driven, except ISO-IN and interrupt-IN transfers, which are periodic. The USB host initiates USB traffic and the application responds to all USB host commands. In this mode, the USB Controller Subsystem can be used only in device-type applications. Prior to the start of an operation, the application must completely program the necessary CSRs in the subsystem.

11.5.3 Operations: USB Control and Status Registers (CSRs)

The subsystem Control and Status Registers (CSRs) provide a high degree of control, resulting in a subsystem that is both configurable and scalable. The CSRs are divided into two basic categories: global CSRs, which are specific to the USB device, and endpoint CSRs, which are specific to a particular endpoint. Each endpoint has a set of these endpoint-specific CSRs. Interrupt bits in the interrupt registers are cleared on a write of 1 to that bit (R_WC).

11.6 USB Device: Controller / CSR Registers

11.6.1 USB Device Controller / CSR Registers: Map

The endpoint-specific CSRs occupy the first 512 bytes of the map. Each endpoint consumes 32 bytes of memory space in both the IN and OUT directions. The global CSRs, which serve the device, consume 32 bytes of memory space. The following table shows the address allocation for the CSRs.

Register Offset	Register Name	Description
IN Endpoint-Specific registers		
0x000	USB_end0_ctrl_in	Endpoint 0 Control
0x004	USB_end0_status_in	Endpoint 0 Status
0x008	USB_end0_buffsize_in	Endpoint 0 Buffer Size
0x00C	USB_end0_packetsize_in	Endpoint 0 Maximum Packet Size
0x010		Reserved
0x014	USB_end0_desptr_in	Endpoint 0 Data Descriptor Pointer
0x018		Reserved
0x01C	USB_end0_write_confirm	Endpoint 0 Write Confirmation (for Slave mode)
0x020 - 0x03C	USB_end1 registers	Endpoint 1 registers
0x040 - 0x0SC	USB_end2 registers	Endpoint 2 registers
0x060 - 0x07C	USB_end3 registers	Endpoint 3 registers
0x080 - 0x09C	USB_end4 registers	Endpoint 4 registers
0x0A0 - 0x0BC	USB_end5 registers	Endpoint 5 registers
0x0C0 - 0x1FC		Reserved
OUT Endpoint-Specific registers		
0x200	USB_end0_ctrl_out	Endpoint 0 Control
0x204	USB_end0_status_out	Endpoint 0 Status
0x208	USB_end0_packet_fm_out	Endpoint 0 Packet Frame Number
0x20C	USB_end0_buffpacket_out	Endpoint 0 Buffer Size OUT/Maximum Packet Size
0x210	USB_end0_setup_buffptr	Endpoint 0 SETUP Buffer Pointer
0x214	USB_end0_desptr_out	Endpoint 0 Data Descriptor Pointer
0x218		Reserved
0x21C	USB_end0_read_confirm	Endpoint 0 Read Confirmation register for zero-length OUT data (for Slave mode)
0x220 - 0x23C	USB_end1 registers	Endpoint 1 registers
0x240 - 0x25C	USB_end2 registers	Endpoint 2 registers
0x260 - 0x27C	USB_end3 registers	Endpoint 3 registers

Register Offset	Register Name	Description
0x280 - 0x29C	USB_end4 registers	Endpoint 4 registers
0x2A0 - 0x20C	USB_end5 registers	Endpoint 5 registers
0x2C0 - 0x3FC		Reserved
Global registers		
0x400	USB_dev_config	Device Configuration
0x404	USB_dev_ctrl	Device Control
0x408	USB_dev_status	Device Status
0x40C	USB_dev_int	Device Interrupt
0x410	USB_dev_int_mask	Device Interrupt Mask
0x414	USB_end_int	Endpoint Interrupt
0x418	USB_end_int_mask	Endpoint Interrupt Mask
0x41C	USB_test	Test Mode
0x420 - 0x4FC		Reserved
0x500 - 0x7FC	USB_udc	UDC registers

Table 11-1. USB CSR Memory Map.

Apart from the subsystem CSRs, the UDC contains CSRs that require 768 bytes of memory space. These CSRs are mapped to the 0x500 - 0x7FC address space. The SETUP command address pointer register (offset address 500H) is no longer writable and returns 0 when read, due to the fact that the SETUP command address pointer is already hard-coded to 0xFFFF in both the subsystem and the UDC core. The application must use offset addresses starting from 0x504, then 0x508, and so on to program the UDC core endpoint buffers. The subsystem maps these offset addresses in turn with the UDC CSR addresses, starting from 0x4, then 0x8, and so on.

Writing to the offset address (offset 0x01C + [endpoint number x 0x020]) confirms the IN data into the TxFIFO. This is applicable only in the Slave mode of operation. The data in the Rx FIFO is mapped from address 0x800 up to an address of the (0x800 + 256 * 4 bytes of Rx FIFO (RAM)), which is followed by the address space of the Tx FIFO.

11.6.2 USB Device Controller / CSR Registers: Details

The USB Device global CSR registers are detailed below.

11.6.2.1 USB_dev_config Register

Bits	Name	Attr	Reset	Description
31:19				Reserved
18	set_desc	RW	0	Indicates that the device supports set descriptor requests. Values for this field are: 0 - The USB controller subsystem returns a STALL handshake to the USB host. 1 - The SETUP packet for the set descriptor request passes to the application.

Bits	Name	Attr	Reset	Description
17	csr_prg	RW	0	Indicates the device supports dynamic UDC register programming. The application can program the USB_udc registers dynamically whenever it has received an interrupt for either a Set Configuration or a Set Interface request. If this bit is enabled, the USB Controller Subsystem returns a non-acknowledge (NAK) handshake during the status IN stage of both the Set Configuration and Set Interface requests until the application has written 1 to the csr_done bit 13 of the USB_dev_ctrl register.
16	halt_status	RW	0	Indicates whether the UDC-AHR Subsystem responds with a STALL or an acknowledge (ACK) handshake when the USB host has issued a Clear_Feature (ENDPOINT_HALTED) request for Endpoint 0. Values for this field are: 0 - ACK 1 - STALL
15:13	hs_timeout_calib	RW	0	These three bits indicate the number of PHY clocks to the UDC20-AHB Subsystem timeout counter. The application uses these bits to increase the timeout value (736 to 848 bit-times in high-speed operation), which depends on the PHY's delay in generating a line-state condition. The default timeout value is 736 bit-times. These bits are reserved for the UDC-AHB Subsystem.
12:10	fs_timeout_calib	RW	0	These three bits indicate the number of PHY clocks to the UDC20-AHB Subsystem timeout counter. The application uses these bits to increase the timeout value (16 to 18 bit-times in full-speed operation), which depends on the PHY's delay in generating a line-state condition. The default timeout value is 16 bit-times. These bits are reserved for the UDC-AHB Subsystem.
9	phy_error_detect	RW	0	If the application sets this bit, the device detects the PHY_RXVAL-ID or PHY_RXACTIVE input signal to be continuously asserted for 2 ms, indicating PHY error. This bit is reserved for the UDC-AHB Subsystem.
8	status_1	RW		This bit, together with status (Bit 7), provides the option for the UDC-AH9 Subsystem to respond to the USS host with a STALL or ACK handshake if the USB host has issued a non-zero-length data packet during the STATUS-OUT stage of a CONTROL transfer.
7	status	RW		This bit, together with status_1 (Bit 8), provides the option for the USB Controller Subsystem to respond to the USB host with a STALL or ACK handshake if the USB host has issued a non-zero-length data packet during the STATUS-OUT stage of a CONTROL transfer.
6				Reserved
5	p1	RW	1	PHY interface Indicates if the UTMI PHY must support an 8-bit or 16-bit interface. Values for this field are: 0 - 16-bit 1 - 8-bit Note: This bit is only used for the USB Controller Subsystem.
4	ss	RW	0	Indicates that the device supports SyncFrame
3	sp	RW	0	Indicates that the device is self-powered

Bits	Name	Attr	Reset	Description
2	rwkp	RW	0	Indicates that the device is remote wake-up capable
1:0	spd	RW	0	<p>Device speed This is the expected speed the application programs to the sub-system. The actual speed the subsystem operates depends on the enumeration speed (enum_spd) in the USB_dev_status register. Possible values for this field are: 0x0 - HS (PHY clock = 30 or 60 MHz) 0x1 - FS (PHY clock = 30 or 60 MHz) 0x2 - LS (PHY clock = 6 MHz) 0x3 - FS (PHY clock = 48 MHz)</p> <p>Note: This bit is only used for the UDC-AHR Subsystem.</p>

Table 11-2. USB Device Configuration CSR Register (Global).

11.6.2.2 USB_dev_ctrl Register

This register is set at run-time and controls the device after device configuration is completed.

Bits	Name	Attr	Reset	Description
31:24	thlen	RW	0	<p>Threshold Length Indicates the number (thlen + 1) of 32-bit entries in the RxFIFO before the DMA can initiate data transfer.</p>
23:16	brlen	RW	0	<p>Burst Length Indicates the length, in 32-bit transfers, of a single burst on the AHB. The subsystem sends a number of 32-bit transfers equal to (brlen + 1).</p>
15				Reserved
14	srx_flush	RW	0	<p>Receive FIFO Flush for Single Receive FIFO. The core must be configured with the UDC20AHB_SNAK_ENH_CC enhancement to use this bit, as this will enable the application to set the nak bit (USB_end[n].ctrl_in/out) when the receive FIFO is not empty. When the application wants to flush the receive FIFO, it must first set the devnak bit in this USB_dev_ctrl register. If the receive DMA is in progress, then the core will finish the current descriptor, terminate the DMA, and flush the data in the receive FIFO and clear this bit. The application must clear this bit after the receive FIFO is empty, by checking the rxfifo_empty bit in the USB_dev_status register.</p>
13	csr_done	W	0	The application uses this bit to notify the USB Controller Subsystem that the application has completed programming all required UDC registers, and the Subsystem can acknowledge the current Set Configuration or Set Interface command.
12	devnak	RW	0	When the application sets this bit, the Subsystem core returns a NAK handshake to all OUT endpoints. By writing 1 to this bit, the application does not need to write 1 to the snak (bit 7) of USB_end[n].ctrl_in/out .

Bits	Name	Attr	Reset	Description
11	scale	RW	0	Scale Down This bit reduces the timer values inside the USB Controller subsystem when running gate-level simulation only. When this bit is set to 1, timer values are scaled down to reduce simulation time. Reset this bit to 0 for normal operation.
10	sd	RW	0	Soft Disconnect The application software uses this bit to signal the UDC2O to soft-disconnect. When set to 1, this bit causes the device to enter the disconnected state.
9	mode	RW	0	Enables the application to dictate subsystem operation in either DMA mode (1) or Slave mode (0) operation.
8	bren	RW	0	Burst Enable When this bit is set, transfers on the AHB are split into bursts.
7	the	RW	0	Threshold Enable When this bit is set, a number of quadlets equivalent to the threshold value is transferred from the RxFIFO to the memory.
6	bf	RW	0	The DMA is in Buffer Fill mode and transfers data into contiguous locations pointed to by the buffer address.
5	be	RW	0	System Endianness A value of 1 indicates a big endian system.
4	du	RW	0	Descriptor Update When this bit is set, the DMA updates the descriptor at the end of each packet processed.
3	tde	RW	0	Transmit DMA is enabled
2	rde	RW	0	Receive DMA is enabled
1				Reserved
0	res	RW	0	Resuming Signaling on the USB To perform a remote wake-up resume, the application sets this bit to 1. The USB Controller Subsystem signals the USB host to resume the USB bus; however, the application must first set rwkp (bit 2) in tsb_dev_config (indicating that the Subsystem supports the Remote Wakeup feature)

Table 11-3. USB Device Control CSR Registers (Global).

11.6.2.3 USB_dev_status Register

This register reflects status information required to service various interrupts. This is a read-only register.

Bits	Name	Attr	Reset	Description
31:18	ts	R	0	Frame number of the received SOF For high-speed operation: [31:21] - Millisecond frame number [20:18] - Microframe number For full-speed operation: [31:29] - Reserved [28:18] - Millisecond frame number

Bits	Name	Attr	Reset	Description
17	rmtwkp_state	R	0	The state of remote wakeup feature as a result of a Set/Clear Feature (Remotewakeup) command from the host. A value of 1 indicates a Set Feature (Remotewakeup) has been received. A value of 0 indicates Clear Feature (Remotewakeup) has been received. Any change to this bit sets an interrupt for rmtwkp_state_int (bit 7 of USB_dev_int) if not masked.
16	phy_error	R	0	Either the PHY_RXVALID or PHY_RXACTIVE input signal is detected to be continuously asserted for 2 ms, indicating PHY error. The UDC20-AHB Subsystem goes to the Suspend state as a result. When the application serves the early suspend interrupt es (bit 2 of USB_dev_int) it also must check this bit to determine if the early suspend interrupt was generated due to PHY error detection.
15	rxfifo_empty	R	1	RxFIFO empty status: 1 - RxFIFO is empty. 0 - RxFIFO is not empty.
14:13	enum_spd	R	0	<p>Enumerated Speed These bits hold the speed at which the subsystem comes up after the speed enumeration.</p> <p>If the expected speed (the spd bit in USB_dev_config) is high-speed and the subsystem connects to a 1.1 host controller, then after Speed Enumeration, these bits indicate that the subsystem is operating in full-speed mode (0x1 for spd = 0x0).</p> <p>If the spd is high-speed and the subsystem connects to a 2.0 host controller, then after Speed Enumeration, these bits indicate that the subsystem is operating in high-speed mode (0x0 for spd = 0x0).</p> <p>If the speed is low-speed or full-speed and the subsystem connects to either a 1.1 or a 2.0 host controller, then after Speed Enumeration, these bits indicate that the subsystem is operating in low-speed mode (0x2 for spd = 0x2) or full-speed mode (0x1 for spd = 0x1, and 0x3 for spd = 0x3).</p> <p>Possible Values for this field are: 0x0 - High speed 0x1 - Full speed 0x2 - Low speed 0x3 - Full speed</p>
12	susp	R	0	Suspend status This bit is set as long as a Suspend condition is detected on the USB.
11:8	alt	R	0	This 4-bit field represents the alternate setting, to which the above interface is switched.
7:4	intf	R	0	This 4-bit field reflects the interface set by the SetInterface command.
3:0	cfg	R	0	This 4-bit field reflects the configuration set by the SetConfiguration command.

Table 11-4. USB Device Status CSR Register (Global).

11.6.2.4 USB_dev_int Register

Device interrupts are set when system-level events occur, and these interrupts are used by the application to make system-level changes. After checking the register, the application must clear the interrupt by writing 1 to the correct bit.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7	rmtwkp_state_int	R_WC	0	A Set/Clear Feature (Remote Wakeup) is received by the core. This bit is set by the core when rmtwkp_state (bit 17 of USB_dev_status) changes: HIGH to LOW or LOW to HIGH.
6	enum	R_WC	0	Speed enumeration is complete
5	sof	R_WC	0	An SOF token is detected on the USB
				A suspend is detected on the USB
4	us	R_WC	0	Note: For the UDC20-AHB subsystem, there is no Suspend interrupt to the application if the PHY clock is suspended via the SUSPENDM signal
3	ur	R_WC	0	A reset is detected on the USB
2	es	R_WC	0	An idle state has been detected on the USB bus for 3 milliseconds
				The device has received a Set_Interface command
1	si	R_WC	0	Note: If the application has not served this interrupt, the subsystem returns a NAK handshake to all transactions except the 8 SETUP packet bytes coming from the USB host.
				The device has received a Set_Configuration command
0	sc	R_WC	0	Note: If the application has not served this interrupt, the subsystem returns a NAK handshake to all transactions except the 8 SETUP packet bytes coming from the USB host.

Table 11-5. USB Device Interrupt CSR Register (Global).

11.6.2.5 USB_dev_int_mask Register

The device interrupt mask can be set for system-level interrupts using this register. Programming 1 in the appropriate bit position in the Interrupt Mask register masks the designated interrupt. Once masked, an interrupt signal will not reach the application, nor will its interrupt bit be set.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	mask	RW	1	Mask equivalent device interrupt bit

Table 11-6. USB Device Interrupt Mask CSR Register (Global).

11.6.2.6 USB_end_int Register

The Endpoint Interrupt or **USB_end_int** register is used to set endpoint-level interrupts. Since all 6 endpoints can be bidirectional, each endpoint has two interrupt bits (one for each direction). The application is required to clear the interrupt by writing 1 to the correct bit after checking the register.

Bits	Name	Attr	Reset	Description
31:22				Reserved
21:16	out_ep	R_WC	0	One bit per OUT endpoint, set when an event occurs on that endpoint
15:6				Reserved
5:0	in_ep	R_WC	0	One bit per IN endpoint, set when an event occurs on that endpoint

Table 11-7. USB Endpoint Interrupt CSR Register (Global).

11.6.2.7 USB_end_int_mask Register

This register is used to mask endpoint interrupts. A write of 1 to any bit in this register masks the corresponding endpoint for possible interrupts. Once masked, an interrupt signal does not reach the application, nor will its interrupt bit be set.

Bits	Name	Attr	Reset	Description
31:22				Reserved
21:16	out_ep_mask	RW	1	Masks interrupts to the OUT endpoint equivalent to this value
15:6				Reserved
5:0	in_ep_mask	RW	1	Masks interrupts to the IN endpoint equivalent to this value

Table 11-8. USB Endpoint Interrupt Mask CSR Register (Global).

The USB Device endpoint-specific CSR registers are detailed below.

11.6.2.8 **USB_end[n].ctrl_in/out Registers**

This register is used to program each endpoint (number N) as required by the application. If the endpoint is bidirectional, there will be two such endpoint registers.

Bits	Name	Attr	Reset	Description
31:12				Reserved
11	close_desc	RW	0	<p>Close descriptor channel for this endpoint This bit applies only to OUT endpoints. The application sets this bit to close the descriptor channel, and the UDC Subsystem clears this bit after the channel is closed. This bit provides the application with a mechanism to close the descriptors in cases where the USB host does not indicate an end-of-transfer (by issuing a short packet to the USB device). To close the descriptor channel for a particular endpoint, the application sets the close_desc bit in the Endpoint Control register (USB_end[n].ctrl_in/out). When the channel is closed, the UDC Subsystem clears this bit and generates an interrupt. This bit must be used only for bulk and interrupt OUT endpoints. In addition, before closing the descriptor, software must ensure that the current descriptor reachable by the DMA is active (buffer status is Host Ready). When closed, the descriptor is marked with the Last bit set, and is assigned one of the following descriptor statuses:</p> <ul style="list-style-type: none"> Buffer Fill mode: Accumulated byte count is available in the Rx Bytes field. Packet-Per-Buffer With Descriptor Update mode: Current reachable descriptor is marked with the Last descriptor, and the Rx Bytes field is set to 0. Packet-Per-Buffer Without Descriptor Update mode: Current reachable descriptor is marked with the Last descriptor, and the accumulated byte count is available in the Rx Bytes field.
10	send_null	RW	0	<p>Send NULL packet This bit provides the application with a mechanism to instruct the UDC Subsystem to send a NULL (zero-length) packet when no data is available in the particular endpoint's TxFIFO. If this bit is set, when no data is available in the endpoint's TxFIFO, the UDC Subsystem sends a NULL packet.</p>
9	rrdy	RW	0	<p>Receive Ready If this bit is set by the application, on receiving an OUT packet, the DMA sends the packet to system memory. This bit is deasserted at the end of packet if the Descriptor Update bit du is set in the Device Control register USB_dev_ctrl. This bit is deasserted at the end of payload if the Descriptor Update bit is deasserted. This bit can be set by the application at any time. The application cannot clear this bit if the DMA is busy transferring data.</p>

Bits	Name	Attr	Reset	Description
8	cnak	W	0	<p>Clear Non-Acknowledge (NAK)</p> <p>Used by the application to clear nak (bit 6, below). After the subsystem sets nak, the application must clear it with a write of 1 to cnak. (For example, after the application has decoded the SETUP packet and determined it is a valid command, the application must set the cnak bit of the control endpoint to 1 to clear the nak bit.) The application must also clear the nak bit whenever the subsystem sets it (The subsystem sets it when the application sets a Stall bit.) The application can write cnak without waiting for a RxFIFO empty condition.</p> <p>The nak bit is cleared immediately upon write to cnak bit. No polling is necessary.</p>
7	snak	W	0	<p>Set NAK</p> <p>Used by the application to set nak (bit 6). The application must not set the nak bit for an IN endpoint until it has received an IN token interrupt indicating that the TxFIFO is empty.</p>
6	nak	R	0	<p>After a SETUP packet, which is decoded by the application, is received by the core, the core sets the nak bit for all control IN/OUT endpoints. nak is also set after a STALL response for the endpoint (the STALL bit s is bit 0 in this Endpoint Control register USB_end[n].ctrl_in/out).</p> <p>1 - The endpoint responds to the USB host with a NAK handshake. 0 - The endpoint responds normally.</p> <p>Note 1: A SETUP packet is sent to the application regardless of whether the nak bit is set. Note 2: If the NAK for ISOC OUT endpoint is set, then the data out from the host is accepted, provided there is space in the FIFO.</p>
5:4	et	RW	0	<p>Endpoint Type</p> <p>The possible values for this field are:</p> <ul style="list-style-type: none"> 0x0 - Control endpoint 0x1 - Isochronous endpoint 0x2 - Bulk endpoint 0x3 - Interrupt endpoint
3	p	RW	0	<p>Poll Demand</p> <p>Poll demand from the application. The application can set this bit after an IN token is received from the endpoint. The application can also set this bit before an IN token is received for the endpoint, provided it has received the IN transfer data in advance.</p> <p>Note: After sending a zero-length or short packet, the application must wait for the next xferdone_txempty interrupt (USB_end[n].status_in/out), before setting up the p bit for the next transfer.</p>
2	sn	RW	0	<p>Configures the endpoint for Snoop mode. In this mode, the subsystem does not check the correctness of OUT packets before transferring them to application memory. Reserved for IN endpoints.</p>
1	f	RW	0	<p>Flush the TxFIFO</p> <p>Reserved for OUT endpoints. The application firmware sets this bit to 1 after it has detected a disconnect/connect on the USB cable, then waits for the IN token endpoint interrupt before resetting this bit to 0. This flushes the stale data out of the TxFIFO. This bit is cleared by the core when Transmit DMA Complete occurs on this endpoint (tdc bit in USB_end[n].status_in/out).</p>

Bits	Name	Attr	Reset	Description
0	s	RW	0	<p>STALL Handshake On successful reception of a SETUP packet (decoded by the application), the subsystem clears both IN and OUT stall bits (s), and sets both the IN and OUT nak bits. The application must verify that the RxFIFO is empty prior to setting the IN and OUT STALL bit s. For non-SETUP packets, the subsystem clears either IN or out s bits only if a STALL handshake is returned to the USB host, then sets the corresponding nak bit. The subsystem returns a STALL handshake for the subsequent transactions of the stalled endpoint until the USB host issues a CLEAR_FEATURE command to clear it. Once this bit is set, if the subsystem has already returned a STALL handshake to the USB host, the application firmware cannot clear the s bit to stop the subsystem from sending the STALL handshake on the endpoint. The host must instead send one of the following commands to clear the endpoint Halt status: Clear Feature (Halt) SetConfiguration SetInterface</p> <p>Note: The application can set the STALL bit without waiting for a RxFIFO empty condition. No polling is necessary.</p>

Table 11-9. USB Endpoint Control CSR Register (Endpoint-Specific).

11.6.2.9 **USB_end[n].status_in/out Registers**

Bits	Name	Attr	Reset	Description
31:29				Reserved
28	cdc_clear	R_WC	0	This bit is valid only for OUT endpoints when close_desc is enabled in USB_end[n].ctrl_in/out . This bit will be set by the core hardware when close_desc is cleared, generating an interrupt. After servicing the interrupt the application software must clear this bit.
27	xferdone_txempty	R_WC	0	This bit indicates that the TxFIFO is empty after the DMA transfer has been completed. The application can use this bit to set the poll bit for the next transfer. The application must first clear this bit after servicing the interrupt.
26	rss	R_WC	0	Received Set Stall This bit indicates that the Set Feature (ENDPOINT_HALTI) command is received for this endpoint. To stall this endpoint, the application can set the stall bit s in USB_end[n].ctrl_in/out . Subsequently, the application clears this rss bit to acknowledge the reception of a Set Feature stall command. Once this rss bit is cleared, the core sends the zero-length packet for the Status IN phase of the Set Feature command. The Received Set Stall rss indication is applicable only for Bulk and Interrupt transactions.

Bits	Name	Attr	Reset	Description
25	rcs	R_WC	0	Received Clear Stall Indication This bit indicates that the Clear Feature (ENDPOINT_HALTI) command is received for this endpoint. To continue to stall the endpoint, the application must set the stall bit s in USB_end[n].ctrl_in/out . Subsequently, the application clears this rcs bit to acknowledge the reception of a clear stall command. Once this bit is cleared, the core sends the zero-length packet for the Status IN phase of the clear stall command. The Received Clear Stall indication is applicable only for Bulk and Interrupt transactions.
24				Reserved
23	iso_in_done	RW	0	Isochronous IN transaction for the current microframe is complete. This bit indicates that the isochronous IN transaction for this endpoint is complete. The application can use this information to program the isochronous IN data for the next microframe. This bit is used only in Slave-Only mode.
22:11	rx_pkt_size	RW	0	Receive Packet Size Indicates the number of bytes in the current receive packet. Because the USB host always sends 8 bytes of SETUP data, these bits do not indicate the receipt of 8 bytes of SETUP data for a SETUP packet. Rather, these bits indicate the configuration status (Configuration number [22:19], Interface number [18:15], and Alternate Setting number [14:11]). This field is used in Slave mode only. In DMA mode, the application must check the status from the endpoint data descriptor.
10	tdc	R_WC	0	Transmit DMA Completion Indicates that the transmit DMA has completed transferring a descriptor chain's data to the TxFIFO. After servicing the interrupt, the application must clear this bit.
9	he	R_WC	0	Error response on the host bus (AHB) when performing a data transfer, descriptor fetch, or descriptor update for this particular endpoint. After servicing the interrupt, the application must clear this bit.
8				Reserved
7	bna	R_WC	0	Buffer Not Available The subsystem sets this bit when the descriptor status is either Host Busy or DMA Done to indicate that the descriptor was not ready at the time the DMA attempted access. After servicing the interrupt, the application must clear this bit.
6	in	R_WC	0	An IN token has been received by this endpoint. After servicing the interrupt, the application must clear this bit. (Reserved for OUT endpoints.)

Bits	Name	Attr	Reset	Description
5:4	out	R_WC	0	<p>An OUT packet has been received by this endpoint. The encoding of these two bits indicates the type of data received. The possible values for this field are:</p> <ul style="list-style-type: none"> 0x0 - None 0x1 - Received data 0x2 - Received SETUP data (8 bytes) 0x3 - Reserved (The application must write the same values to clear these bits.) <p>Note: In Slave mode the application must clear these bits only after clearing the FIFO (in other words, only when the RxFIFO is empty). Clearing these bits before the FIFO is empty causes these bits to remain set until the FIFO is cleared. If by that time the Set/Clear Feature command referring to this endpoint is received, then this Endpoint Status register USB_end[n].status_in/out shows both out and rcs/rss bits, which may cause instability in the application.</p>
3:0				Reserved

Table 11-10. USB Endpoint Status CSR Register (Endpoint-Specific).

11.6.2.10 **USB_end[n].buffsize_in / USB_end[n].packet_fm_out Registers**

This dual-function register holds the endpoint buffer size when the endpoint is an IN endpoint. When the endpoint is an OUT endpoint, the register contains the frame number in which the packet is received, updated in bits 13:0. This frame number information is useful when handling isochronous traffic.

Bits	Name	Attr	Reset	Description
31:18				Reserved
17:16 (In)	iso_in_pid	RW	0	<p>Initial data PID to be sent for a high-bandwidth isochronous IN transaction. This field is used only in Slave mode.</p> <ul style="list-style-type: none"> 0x0 - DATA0 PID is sent 0x1 - DATA0 PID is sent 0x2 - DATA1 PID is sent 0x3 - DATA2 PID is sent
17:16 (Out)	iso_out_pid	R	0	<p>Data PID received for a high-bandwidth isochronous OUT transaction. This field indicates that the data PID for the current packet is available in the Receive FIFO. This field is used only in Slave mode.</p> <ul style="list-style-type: none"> 0x0 - DATA0 PID is received 0x1 - DATA1 PID is received 0x2 - DATA2 PID is received 0x3 - MDATA PID is received
15:0	buff_size	RW	0	<p>Buffer size required for this endpoint</p> <p>The application can program this field to make each endpoint's buffers adaptive, providing flexibility in buffer size when the interface or configuration is changed. This value is in 32-bit words, and indicates the number of 32-bit word entries in the Transmit FIFO. (IN only)</p>

Bits	Name	Attr	Reset	Description
15:0	frame_number	R	0	Frame number in which the packet is received For high-speed operation: [15:14] - Reserved [13:3] - Millisecond frame number [2:0] - Microframe number For full-speed operation: [15:11] - Reserved [10:0] - Millisecond frame number

Table 11-11. USB Buffer Size/Frame Number Register (Endpoint-Specific).

11.6.2.11 **USB_end[n].buffpacket_out** Register

This register holds the endpoint buffer size when the endpoint is an OUT endpoint. This register also specifies the maximum packet size an endpoint should support. This maximum size is used to calculate whether the Receive FIFO has sufficient space to accept a packet. When changing the maximum packet size for a specific endpoint, the user must also program the UDC register space.

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:0	max_pkt_size	RW	0	Maximum packet size for the endpoint (value in bytes)

Table 11-12. USB Endpoint Buffer Size OUT/Maximum Packet Size Register (Endpoint-Specific).

11.6.2.12 **USB_end[n].setup_buffptr** Register

Endpoint SETUP buffer pointers are used for SETUP commands. This Endpoint Setup Buffer Pointer register **USB_end[n].SETUP_buffptr** tracks control endpoint buffer registers. Endpoint buffer registers for all other endpoint types are reserved. The **USB_end[n].SETUP_buffptr** is used only in DMA mode. This is applicable only to control endpoints. For all other endpoints the register is reserved.

Bits	Name	Attr	Reset	Description
31:0	subptr	RW	0	SETUP Buffer Pointer

Table 11-13. USB Endpoint SETUP Buffer Pointer Register (Endpoint-Specific).

11.6.2.13 **USB_end[n].desptr_out** Register

This register contains data descriptor pointers. Both IN and OUT endpoints have one data descriptor pointer each.

Bits	Name	Attr	Reset	Description
31:0	desptr	RW	0	Descriptor Pointer

Table 11-14. USB Endpoint Data Descriptor Pointer Register (Endpoint-Specific).

11.6.2.14 USB_end[n].read/write_confirm Registers

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	confirm	W	0	Read or Write confirm bit

Table 11-15. USB Endpoint [n] Read-Confirm and Write-Confirm registers.

11.6.2.15 USB_test Register

The Test Mode register (**USB_test**) sets the USB Controller Subsystem to Test mode. In Test mode, the application can use the AHB to read from a TxFIFO or write to an RxFIFO. Test mode is supported only in the Slave operational configuration. In Test mode, only single DWORD transactions are supported: byte and word transactions on the AHB are not supported. To synchronize the write/read pulses between AHB and VCI clocks, wait states are introduced on the AHB for every data transfer.

While in Test mode, all transactions from the USB host, except the SETUP packet, receive a NAK response. When the SETUP packet is received, it is accepted, but the data is not written into the Receive FIFO. After writing the TxFIFO, the data must be confirmed by writing to the offset address (offset 0x01C + [endpoint number x 0x020]), which is similar to the normal mode of operation. In non-Test modes, reading from a TxFIFO or writing to an RxFIFO results in an AHB error response. The application must never read an empty TxFIFO or write a full RxFIFO even though an AHB error response is not provided. Writing to the RxFIFO in Test mode and enabling the DMA to transfer data are not supported.

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	tstmode	RW	0	Test Mode indicator

Table 11-16. USB Test Mode Register.

11.6.2.16 USB_udc2O Endpoint Register

Bits	Name	Attr	Reset	Description
31:30				Reserved
29:19	max_packet_size	RW	0	Maximum packet size
18:15	alt_setting_end	RW	0	Alternate setting to which this endpoint belongs
14:11	interface_no_end	RW	0	Interface number to which this endpoint belongs
10:7	config_no_end	RW	0	Configuration number to which this endpoint belongs
6:5	end_type	RW	0	Endpoint type The possible values for this field are: 0x0 - Control 0x1 - Isochronous 0x2 - Bulk 0x3 - Interrupt
4	end_direction	RW	0	Endpoint direction The possible values for this field are: 0 - Out 1 - In

Bits	Name	Attr	Reset	Description
3:0	logical_end_no	RW	0	Logical Endpoint Number

Table 11-17. USB UDC2O Endpoint Register

11.7 USB Device: Initialization

To initialize the UDC20-AHB Subsystem, application software must program the following registers:

- Device Configuration register ([USB_dev_config](#) in [Section 11.6.2.1](#)).
- Device Control register ([USB_dev_ctrl](#) in [Section 11.6.2.2](#)).
- Endpoint Control registers for Logical Endpoint 0, 1, 2, and so on ([USB_end\[n\].ctrl_in/out](#) in [Section 11.6.2.8](#)).
- Endpoint Buffer Size registers for Logical Endpoint 0, 1, 2, and so on ([USB_end\[n\].buffsize_in](#) in [Section 11.6.2.10](#)).
- Endpoint Max Packet Size registers for Logical Endpoints 0, 1, 2, and so on ([USB_end\[n\].buffpacket_out](#) in [Section 11.6.2.11](#)).
- Endpoint 0 Setup Buffer Pointer register ([USB_end\[n\].setup_buffptr](#) in [Section 11.6.2.12](#)).
- UDC20 registers for Physical Endpoint 1, 2, and so on ([USB_udc2O_Endpoint](#) in [Section 11.6.2.16](#)).

12. USB HOST CONTROLLER

This chapter provides information regarding the A12 USB Host Controller. The chapter is organized as follows:

- [\(Section 12.1\) USB Host: Overview](#)
- [\(Section 12.2\) USB Host: Host / Device Select](#)
- [\(Section 12.3\) USB Host: Force USB Boot Mode](#)
- [\(Section 12.4\) USB Host: Registers](#)

12.1 USB Host: Overview

The USB Host Controller is an Open Host Controller Interface (OHCI) and Enhanced Host Controller Interface (EHC) compliant USB 2.0/1.1 Host. The EHCI controller implements data and descriptor pre-fetching for higher performance. There is a 4-KB data buffer and a 272-B descriptor buffer.

The sections below document the OHCI, EHCI and implementation-specific registers of the USB Host Controller. For more information on OHCI, please reference *OpenHCI: Open Host Controller Interface Specification for USB*. For more information on EHCI, please reference *Enhanced Host Controller Interface Specification for Universal Serial Bus*. Both of these documents are publicly available online.

Note that system software generally sets and configures the USB PHY clock sources using a set of APIs with RCT registers. The register programming defined in this chapter is used for adjustment and refinement.

12.2 USB Host: Host / Device Select

The A12 chip provides one USB high-speed interface that can be configured to function in master or slave mode. For information on this interface, please refer to the previous chapter.

12.3 USB Host: Force USB Boot Mode

The A12 chip provides a Force USB power-on boot mode. Refer to [Section 2.3.4](#) for more information.

12.4 USB Host: Registers

This section provides information on the registers of the USB Host Controller as follows:

- ([Section 12.4.1](#)) USB Host Registers: OpenHCI Map
- ([Section 12.4.2](#)) USB Host Registers: OpenHCI Details
- ([Section 12.4.3](#)) USB Host Registers: EnhancedHCI Map
- ([Section 12.4.4](#)) USB Host Registers: EnhancedHCI Details.

12.4.1 USB Host Registers: OpenHCI Map

Register Offset	Register Name	Description
0x00	Hc_revision	HcRevision Provides the HCI version implemented by this HC
0x04	Hc_control	HcControl Register Defines the operating modes for the Host Controller
0x08	Hc_commandstatus	HcCommandStatus Reflects the current status of the Host Controller and is used by the Host Controller to receive commands issued by the Host Controller Driver
0x0C	Hc_interruptstatus	HcInterruptStatus Provides status on various events that cause hardware interrupts
0x10	Hc_interruptenable	HcInterruptEnable Provides enable bits that control which events generate a hardware interrupt and correspond to an associated interrupt bit in Hc_interruptstatus
0x14	Hc_interruptdisable	HcInterruptDisable Provides disable bits that correspond to an associated interrupt bit in Hc_interruptstatus . Writing 1 to a bit in this register clears the corresponding bit in the Hc_interruptenable
0x18	Hc_hcca	HcHCCA Contains the physical address of the Host Controller Communication Area (HCCA)
0x1C	Hc_periodcurrented	HcPeriodCurrentED Contains the physical address of the current Isochronous or Interrupt Endpoint Descriptor
0x20	Hc_controlheaded	HcControlHeadED Contains the physical address of the first Endpoint Descriptor of the Control list
0x24	Hc_controlcurrented	HcControlCurrentED Contains the physical address of the current Endpoint Descriptor of the Control list
0x28	Hc_bulkheaded	HcBulkHeadED Contains the physical address of the first Endpoint Descriptor of the Bulk list

Register Offset	Register Name	Description
0x2C	Hc_bulkcurrented	HcBulkCurrentED Contains the physical address of the current endpoint of the Bulk list
0x30	Hc_donehead	HcDoneHead Contains the physical address of the last completed Transfer Descriptor that was added to the Done queue
0x34	Hc_fmininterval	HcFmInterval Allows the definition of frame time intervals in order to synchronize with an external clocking resource and to adjust an unknown local clock offset
0x38	Hc_fmremaining	HcFmRemaining This 14-bit down counter shows the bit-time remaining in the current frame
0x3C	Hc_fnumber	HcFmNumber This 16-bit counter provides a timing reference among events occurring in the Host Controller (HC) and the Host Controller Driver (HCD)
0x40	Hc_periodicstart	HcPeriodicStart Has a 14-bit programmable value which determines when is the earliest time HC should begin processing the periodic list
0x44	Hc_lthreshold	HcLSThreshold Contains an 11-bit value used by the Host Controller to determine whether to commit to the transfer of a maximum of 8-byte LS packet before EOF
0x48	Hc_rhdescriptor_a	HcRhDescriptorA The first of two registers that describe the Root Hub
0x4C	Hc_rhdescriptor_b	HcRhDescriptorB The second of two registers that describe the Root Hub
0x50	Hc_rhstatus	HcRhStatus Two-part register that represents the Hub Status field and the Hub Status Change field
0x54	Hc_rhportstatus_1	HcRhPortStatus 1 Controls and reports Port 1 events
0x58	Hc_rhportstatus_2	HcRhPortStatus 2 Controls and reports Port 2 events

Table 12-1. OpenHCI Registers.

12.4.2 USB Host Registers: OpenHCI Details

12.4.2.1 Hc_revision Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	rev	R	0x10	<p>Revision</p> <p>Contains the BCD representation of the version of the HCI specification that is implemented by this HC.</p> <p>For example, a value of 0x11 corresponds to version 1.1. All of the HC implementations that are compliant with this specification will have a value of 0x10</p>

Table 12-2. OpenHCI Revision Register.

12.4.2.2 Hc_control Register

Bits	Name	Attr	Reset	Description
31:11				Reserved
10	rwe	RW	0	<p>RemoteWakeUpEnable</p> <p>This bit is used by the HCD to enable or disable the remote wakeup feature upon the detection of upstream resume signaling. When this bit is set and the ResumeDetected bit (rd) in Hc_interruptstatus is set, a remote wakeup is signaled to the host system. Setting this bit has no impact on the generation of hardware interrupts.</p>
9	rwc	RW	0	<p>RemoteWakeUpConnected</p> <p>This bit indicates whether the HC supports remote wakeup signaling. If remote wakeup is supported and used by the system it is the responsibility of system firmware to set this bit during POST. The HC clears the bit in the event of a hardware reset but does not alter it in the event of a software reset.</p>
8	ir	RW	0	<p>InterruptRouting</p> <p>This bit determines the routing of interrupts generated by events registered in Hc_interruptstatus. If cleared, all interrupts are routed to the normal host bus interrupt mechanism. If set, interrupts are routed to the System Management Interrupt. The HCD clears this bit in the event of a hardware reset, but it does not alter this bit in the event of a software reset. The HCD uses this bit as a tag to indicate ownership of the HC.</p>

Bits	Name	Attr	Reset	Description
7:6	hcfs	RW	0	<p>HostControllerFunctionalState for USB</p> <p>0x0 - USBRESET</p> <p>0x1 - USBRESUME</p> <p>0x2 - USBOPERATIONAL</p> <p>0x3 - USBSUSPEND</p> <p>A transition to USBOPERATIONAL from another state causes SOF generation to begin 1 ms later. The HCD may determine whether the HC has begun sending SOFs by reading the StartofFrame field (sf) of Hc_interruptstatus.</p> <p>This field may be changed by the HC only when in the USBSUSPEND state. The HC may move from the USBSUSPEND state to the USBRESUME state after detecting the resume signaling from a downstream port.</p> <p>The HC enters USBSUSPEND following a software reset, whereas it enters USBRESET following a hardware reset. The latter also resets the Root Hub and asserts subsequent reset signaling to downstream ports.</p>
5	ble	RW	0	<p>BulkListEnable</p> <p>This bit is set to enable the processing of the Bulk list in the next frame. If cleared by the HCD, processing of the Bulk list does not occur after the next SOF. The HC checks this bit when it determines whether or not to process the list. When disabled, the HCD may modify the list. If Hc_bulkcurrented is pointing to an ED to be removed, the HCD must advance the pointer by updating Hc_bulkcurrented before re-enabling processing of the list.</p>
4	cle	RW	0	<p>ControlListEnable</p> <p>This bit is set to enable the processing of the Control list in the next frame. If cleared by the HCD, processing of the Control list does not occur after the next SOF. The HC must check this bit when it determines to process the list. When disabled, the HCD may modify the list. If Hc_controlcurrented is pointing to an ED to be removed, the HCD must advance the pointer by updating Hc_controlcurrented before re-enabling processing of the list.</p>
3	ie	RW	0	<p>IsochronousEnable</p> <p>This bit is used by the HCD to enable/disable processing of isochronous EDs. While processing the periodic list in a frame, the HC checks the status of this bit when it finds an Isochronous ED (F = 1). If set (enabled), the HC continues processing the EDs. If cleared (disabled), the HC halts processing of the periodic list (which now contains only isochronous EDs) and begins processing the Bulk/Control lists. Setting this bit is guaranteed to take effect in the next frame (not the current frame).</p>
2	ple	RW	0	<p>PeriodicListEnable</p> <p>This bit is set to enable the processing of the periodic list in the next frame. If cleared by the HCD, processing of the periodic list does not occur after the next SOF. The HC must check this bit before it begins processing the list.</p>

Bits	Name	Attr	Reset	Description
1:0	cbsr	RW	0	<p>ControlBulkServiceRatio</p> <p>This specifies the service ratio between Control and Bulk EDs.</p> <p>Before processing nonperiodic lists, the HC must compare the ratio specified with its internal count on how many nonempty Control EDs have been processed, in determining whether to continue serving another Control ED or switching to Bulk EDs.</p> <p>The internal count will be retained when crossing the frame boundary. In case of reset, the HCD is responsible for restoring this value.</p>

Table 12-3. OpenHCI Control Register.

12.4.2.3 Hc_commandstatus Register

Bits	Name	Attr	Reset	Description
31:18				Reserved
17:16	soc	RW	0	<p>SchedulingOverrunCount</p> <p>These bits are incremented on each scheduling overrun error. It is initialized to 0x0 and wraps around at 0x3. This will be incremented when a scheduling overrun is detected even if SchedulingOverrun bit (so) in Hc_interruptstatus has already been set.</p> <p>This is used by the HCD to monitor persistent scheduling problems.</p>
15:4				Reserved
3	ocr	RW	0	<p>OwnershipChangeRequest</p> <p>This bit is set by an OS HCD to request a change of control of the HC. When set the HC will set the OwnershipChange field (oc) in Hc_interruptstatus. After the changeover, this bit is cleared and remains so until the next request from the OS HCD.</p>
2	bif	RW	0	<p>BulkListFilled</p> <p>This bit is used to indicate whether there are TDs on the Bulk list. It is set by the HCD when it adds a TD to an ED in the Bulk list. When the HC begins to process the head of the Bulk list, it checks BF. As long as BulkListFilled bif is 0, the HC will not start processing the Bulk list. If bif is 1, HC will start processing the Bulk list and will set BF to 0. If the HC finds a TD on the list, then the HC will set BulkListFilled to 1 causing the Bulk list processing to continue. If no TD is found on the Bulk list, and if the HCD does not set BulkListFilled, then bif will still be 0 when the HC completes processing the Bulk list and Bulk list processing will stop.</p>
1	clf	RW	0	<p>ControlListFilled</p> <p>This bit is used to indicate whether there are TDs on the Control list. It is set by the HCD when it adds a TD to an ED in the Control list. When the HC begins to process the head of the Control list, it checks clf. As long as ControlListFilled is 0, the HC will not start processing the Control list. If clf of EHCI_configflag is 1, HC will start processing the Control list and will set ControlListFilled to 0. If the HC finds a TD on the list, then the HC will set ControlListFilled to 1 causing the Control list processing to continue. If no TD is found on the Control list, and if the HCD does not set ControlListFilled, then ControlListFilled will still be 0 when the HC completes processing the Control list and Control list processing will stop.</p>

Bits	Name	Attr	Reset	Description
0	hcr	RW	0	<p>HostControllerReset This bit is set by the HCD to initiate a software reset of the HC. Regardless of the functional state of the HC, it moves to the US-BSUSPEND state in which most of the operational registers are reset except those stated otherwise; e.g., the InterruptRouting field ir of Hc_control, and no Host bus accesses are allowed. This bit is cleared by the HC upon the completion of the reset operation. The reset operation must be completed within 10 ms. This bit, when set, should not cause a reset to the Root Hub and no subsequent reset signaling should be asserted to its downstream ports.</p>

Table 12-4. OpenHCI Command and Status Register.

12.4.2.4 Hc_interruptstatus Register

Bits	Name	Attr	Reset	Description
31				Reserved
30	oc	RW	0	<p>OwnershipChange This bit is set by the HC when the HCD sets the OwnershipChangeRequest field (ocr) in Hc_commandstatus. This event, when unmasked, will always generate an immediate System Management Interrupt (SMI). This bit is tied to 0 when the SMI is not implemented.</p>
29:7				Reserved
6	rhsc	RW	0	<p>RootHubStatusChange This bit is set when the content of Hc_rhstatus or the content of Hc_rhportstatus_1/2 has changed.</p>
5	fno	RW	0	<p>FrameNumberOverflow This bit is set when the MSB of Hc_fnumber (bit 15) changes value, from 0 to 1 or from 1 to 0, and after HccaFrameNumber has been updated.</p>
4	ue	RW	0	<p>UnrecoverableError This bit is set when the HC detects a system error unrelated to USB. The HC should not proceed with processing or signaling before the system error has been corrected. The HCD clears this bit after the HC has been reset.</p>
3	rd	RW	0	<p>ResumeDetected This bit is set when the HC detects that a device on the USB is asserting resume signaling. It is the transition from no resume signaling to resume signaling that causes this bit to be set. This bit is not set when the HCD sets the USBRESUME state.</p>
2	sf	RW	0	<p>StartofFrame This bit is set by the HC at each start of a frame and after the update of HccaFrameNumber. The HC also generates a SOF token at the same time.</p>
1	wdh	RW	0	<p>WritebackDoneHead This bit is set immediately after the HC has written Hc_donehead to HccaDoneHead. Further updates of HccaDoneHead will not occur until this bit has been cleared. The HCD should only clear this bit after it has saved the content of HccaDoneHead.</p>

Bits	Name	Attr	Reset	Description
0	so	RW	0	SchedulingOverrun This bit is set when the USB schedule for the current frame overruns and after the update of HccaFrameNumber. A scheduling overrun will also cause the SchedulingOverrunCount bit (soc) of Hc_commandstatus to be incremented.

Table 12-5. OpenHCI Interrupt Status Register.

12.4.2.5 **Hc_interruptenable** Register

Writing 1 to a bit in this register sets the corresponding bit, whereas writing 0 to a bit in this register leaves the corresponding bit unchanged. On read, the current value of this register is returned.

Bits	Name	Attr	Reset	Description
31	mie	RW	0	0 - Ignore 1 - Enables interrupt generation due to events specified in the other bits of this register (used by the HCD as a Master Interrupt Enable)
30	oc	RW	0	0 - Ignore 1 - Enable interrupt generation due to ownership change
29:7				Reserved
6	rhsc	RW	0	0 - Ignore 1 - Enable interrupt generation due to root hub status change
5	fno	RW	0	0 - Ignore 1 - Enable interrupt generation due to frame number overflow
4	ue	RW	0	0 - Ignore 1 - Enable interrupt generation due to unrecoverable error
3	rd	RW	0	0 - Ignore 1 - Enable interrupt generation due to resume detect
2	sf	RW	0	0 - Ignore 1 - Enable interrupt generation due to start of frame
1	wdh	RW	0	0 - Ignore 1 - Enable interrupt generation due to Hc_donehead writeback
0	so	RW	0	0 - Ignore 1 - Enable interrupt generation due to scheduling overrun

Table 12-6. OpenHCI Interrupt Enable Register.

12.4.2.6 **Hc_interruptdisable** Register

Each disable bit in the **Hc_interruptdisable** register corresponds to an associated interrupt bit in the **Hc_interruptstatus** Register. The **Hc_interruptdisable** register is coupled with the **Hc_interruptenable** Register. Thus, writing 1 to a bit in this register clears the corresponding bit in **Hc_interruptenable**, whereas writing 0 to a bit in this register leaves the corresponding bit in **Hc_interruptenable** unchanged. On read, the current value of the **Hc_interruptenable** register is returned.

Bits	Name	Attr	Reset	Description
31	mie	RW	1	0 - Ignore 1 - Disables interrupt generation due to events specified in the other bits of this register

Bits	Name	Attr	Reset	Description
30	oc	RW	0	0 - Ignore 1 - Disable interrupt generation due to ownership change
29:7				Reserved
6	rhsc	RW	0	0 - Ignore 1 - Disable interrupt generation due to root hub status change
5	fno	RW	0	0 - Ignore 1 - Disable interrupt generation due to frame number overflow
4	ue	RW	0	0 - Ignore 1 - Disable interrupt generation due to unrecoverable error
3	rd	RW	0	0 - Ignore 1 - Disable interrupt generation due to resume detect
2	sf	RW	0	0 - Ignore 1 - Disable interrupt generation due to start of frame
1	wdh	RW	0	0 - Ignore 1 - Disable interrupt generation due to Hc_donehead Writeback
0	so	RW	0	0 - Ignore 1 - Disable interrupt generation due to scheduling overrun

Table 12-7. OpenHCI Interrupt Disable Register.

12.4.2.7 Hc_hcca Register

Bits	Name	Attr	Reset	Description
31:8	hcca	RW	0	Base address of the Host Controller Communication Area
7:0				Reserved

Table 12-8. OpenHCI Host Controller Communication Area Register.

12.4.2.8 Hc_periodcurrented Register

This register contains the physical address of the current Isochronous or Interrupt Endpoint Descriptor.

Bits	Name	Attr	Reset	Description
31:4	pced	RW	0	PeriodCurrentED This is used by the HC to point to the head of one of the Periodic lists which will be processed in the current frame. The content of this register is updated by the HC after a periodic ED has been processed. The HCD may read the content in determining which ED is currently being processed at the time of reading.
3:0				Reserved

Table 12-9. OpenHCI Periodic Current ED Register.

12.4.2.9 Hc_controlheaded Register

This register contains the physical address of the current Isochronous or Interrupt Endpoint Descriptor

Bits	Name	Attr	Reset	Description
31:4	ched	RW	0	ControlHeadED The HC traverses the Control list starting with the Hc_controlheaded pointer. The content is loaded from Hc_hcca during the initialization of the HC.
3:0				Reserved

Table 12-10. OpenHCI Control Head ED Register.

12.4.2.10 Hc_controlcurrented Register

This register contains the physical address of the current Endpoint Descriptor of the Control list.

Bits	Name	Attr	Reset	Description
31:4	cced	RW	0	ControlCurrentED This pointer is advanced to the next ED after serving the present one. The HC will continue processing the list from where it left off in the last frame. When it reaches the end of the Control list, HC checks the ControlListFilled bit (clf) in Hc_commandstatus . If set, it copies the content of Hc_controlheaded to Hc_controlcurrented and clears the bit. If not set, it does nothing. The HCD may modify this register only when the ControlListEnable or cle bit of Hc_control is cleared. When set, the HCD only reads the instantaneous value of this Register. Initially, this is set to 0 to indicate the end of the Control list.
3:0				Reserved

Table 12-11. OpenHCI Control Current ED Register.

12.4.2.11 Hc_bulkheaded Register

This register contains the physical address of the first Endpoint Descriptor of the Bulk list.

Bits	Name	Attr	Reset	Description
31:4	bhed	RW	0	BulkHeadED The HC traverses the Bulk list starting with the Hc_bulkheaded pointer. The content is loaded from Hc_hcca during the initialization of the HC.
3:0				Reserved

Table 12-12. OpenHCI Bulk Head ED Register.

12.4.2.12 Hc_bulkcurrenited Register

This register is for the physical address of the current endpoint of the Bulk list.

Bits	Name	Attr	Reset	Description
31:4	bced	RW	0	BulkCurrentED This is advanced to the next ED after the HC has served the present one. The HC continues processing the list from where it left off in the last frame. When it reaches the end of the Bulk list, the HC checks the ControlListFilled bit (clf) of Hc_control . If set, it copies the content of Hc_bulkheaded to Hc_bulkcurrented and clears the bit. If it is not set, it does nothing. The HCD is only allowed to modify this register when the BulkListEnable bit (ble) of Hc_control is cleared. When set, the HCD only reads the instantaneous value of this Register. This is initially set to 0 to indicate the end of the Bulk list.
3:0				Reserved

Table 12-13. OpenHCI Bulk Current ED Register.

12.4.2.13 **Hc_donehead** Register

This register contains the physical address of the last completed Transfer Descriptor that was added to the Done queue.

Bits	Name	Attr	Reset	Description
31:4	dh	RW	0	DoneHead When a TD is completed, the HC writes the content of Hc_donehead to the NextTD field of the TD. The HC then overwrites the content of Hc_donehead with the address of this TD. This is set to 0 when the HC writes the content of this register to Hc_hcca . It also sets the WritebackDoneHead bit (wdh) of Hc_interruptstatus .
3:0				Reserved

Table 12-14. OpenHCI Done Head Register.

12.4.2.14 Hc_fminterval Register

This register allows the definition of frame time intervals in order to synchronize with an external clocking resource and to adjust unknown local clock offset.

Bits	Name	Attr	Reset	Description
31	fit	RW	0	FrameIntervalToggle The HCD toggles this bit when it loads a new value to the FrameInterval bit (fi).
30:16	fsmpls	RW	0	FSLargestDataPacket This field specifies a value which is loaded into the Largest Data Packet Counter at the beginning of each frame. The counter value represents the largest amount of data in bits which can be sent or received by the HC in a single transaction at any given time without causing a scheduling overrun. The field value is calculated by the HCD.
15:14				Reserved
13:0	fi	RW	0x2EDF	FrameInterval This specifies the interval between two consecutive SOFs in bit-times. The nominal value is set to be 11,999. The HCD should store the current value of this field before resetting the HC by setting the HostControllerReset field (hcr) of Hc_commandstatus , as this will cause the HC to return this field to its nominal value. The HCD may choose to restore the stored value upon the completion of the Reset sequence.

Table 12-15. OpenHCI Frame Interval Register.

12.4.2.15 Hc_fmremaining Register

This register uses a 14-bit down counter to show the bit time remaining in the current frame.

Bits	Name	Attr	Reset	Description
31	frt	RW	0	FrameRemainingToggle This bit is loaded from the FrameIntervalToggle field (fit) of Hc_fminterval when FrameRemaining reaches 0. This bit is used by the HCD for the synchronization between FrameInterval (fi) bit in Hc_fminterval and FrameRemaining (fr).
30:14				Reserved
13:0	fr	RW	0	FrameRemaining This counter is decremented at each bit-time. When it reaches 0, it is reset by loading the FrameInterval (fi) value specified in Hc_fminterval at the next bit time boundary. When entering the USBOPERATIONAL state, the HC re-loads the content with the FrameInterval (fi) value and uses the updated value from the next SOF.

Table 12-16. OpenHCI Frame Remaining Register.

12.4.2.16 Hc_fmnumber Register

This register uses a 16-bit counter as a timing reference for events occurring in the Host Controller (HC) and the Host Controller Driver (HCD).

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:0	fn	RW	0	<p>FrameNumber</p> <p>This is incremented when Hc_fmremaining is re-loaded. It will be rolled over to 0x0 after 0xFFFF. When entering the USB OPERATIONAL state, this will be incremented automatically. The content will be written to Hc_hcca after the HC has incremented the Frame-Number fn at each frame boundary and sent a SOF but before the HC reads the first ED in that frame. After writing to Hc_hcca, HC will set the StartofFrame bit (sf) in Hc_interruptstatus.</p>

Table 12-17. OpenHCI Frame Number Register.

12.4.2.17 Hc_periodicstart Register

This register determines the earliest time the HC should begin processing the periodic list.

Bits	Name	Attr	Reset	Description
31:14				Reserved
13:0	ps	RW	0	<p>PeriodicStart</p> <p>After a hardware reset, this field is cleared. The field is then set by the HCD during the HC initialization. The value is calculated roughly as 10% off from Hc_fminterval. A typical value is 0x3E67. When Hc_fmremaining reaches the value specified, processing of the periodic lists will assume priority over Control/Bulk processing. The HC will therefore begin processing the Interrupt list after completing the current Control or Bulk transaction that is in progress.</p>

Table 12-18. OpenHCI Periodic Start Register.

12.4.2.18 Hc_lsthreshold Register

This register contains an 11-bit value used by the Host Controller to determine whether to commit to transfer of a maximum 8-byte LS packet before EOF.

Bits	Name	Attr	Reset	Description
31:12				Reserved
11:0	lst	RW	0x0628	<p>LSThreshold</p> <p>This field contains a value which is compared to the FrameRemaining field (fr) prior to initiating a Low-Speed transaction. The transaction is initiated only if FrameRemaining (fr) \geq this field. The value is calculated by the HCD with the consideration of transmission and setup overhead.</p>

Table 12-19. OpenHCI Low Speed Threshold Register.

12.4.2.19 Hc_rhdescriptor_a Register

Bits	Name	Attr	Reset	Description
31:24	potpgt	RW	2	<p>PowerOnToPowerGoodTime</p> <p>This byte specifies the duration that the HCD waits before accessing a powered-on port of the Root Hub. The unit of time is 2 ms. The duration is calculated as potpgt * 2 ms.</p>
23:13				Reserved
12	nocp	RW	0	<p>NoOverCurrentProtection</p> <p>This bit describes how the over current status for the Root Hub ports is reported. When this bit is cleared, the OverCurrentProtectionMode field (ocpm) specifies global or per-port reporting.</p> <p>0 - Over current status is reported collectively for all downstream ports</p> <p>1 - No over current status protection supported</p>
11	ocpm	RW	1	<p>OverCurrentProtectionMode</p> <p>This bit describes how the over current status for the Root Hub ports is reported. This field is valid only if the NoOverCurrentProtection field (nocp) is cleared.</p> <p>0 - Over current status is reported collectively for all downstream ports</p> <p>1 - Over current status is reported on a per-port basis</p>
10	dt	R	0	<p>DeviceType</p> <p>This bit specifies that the Root Hub is not a compound device.</p>
9	nps	RW	0	<p>NoPowerSwitching</p> <p>These bits are used to specify whether power switching is supported or ports are always powered. When this bit is cleared, the Power-SwitchingMode (psm) specifies global or per-port switching.</p> <p>0 - Ports are power switched</p> <p>1 - Ports are always powered-on when the HC is powered on</p>
8	psm	RW	1	<p>PowerSwitchingMode</p> <p>This bit is used to specify how the power switching of the Root Hub ports is controlled. This field is only valid if the NoPowerSwitching field (nps) is cleared.</p> <p>0 - All ports are powered at the same time</p> <p>1 - Each port is powered individually. This mode allows port power to be controlled by either the global switch or per-port switching. If the PortPowerControlMask (ppcm) is set, the port responds only to port power commands (Set/ClearPortPower). If the port mask is cleared, then the port is controlled only by the global power switch (Set/ClearGlobalPower).</p>
7:0	ndp	R	2	<p>NumberDownstreamPorts</p> <p>These bits specify the number of downstream ports supported by the Root Hub.</p>

Table 12-20. OpenHCI Root Hub Descriptor A Register.

12.4.2.20 Hc_rhdescriptor_b Register

Bits	Name	Attr	Reset	Description
31:16	ppcm	RW	6	<p>PortPowerControlMask</p> <p>Each bit indicates whether a port is affected by a global power control command when PowerSwitchingMode (psm) is set. When set, the port's power state is only affected by per-port power control (Set/ClearPortPower). When cleared, the port is controlled by the global power switch (Set/ClearGlobalPower). If the device is configured to global switching mode (psm = 0), this field is not valid.</p> <p>Bit [0] - Reserved</p> <p>Bit [1] - Ganged-power mask on Port 1</p> <p>Bit [2:15] - Reserved</p>
15:0	dr	RW	0	<p>DeviceRemovable</p> <p>Each bit is dedicated to a port of the Root Hub. When cleared, the attached device is removable. When set, the attached device is not removable.</p> <p>Bit [0] - Reserved</p> <p>Bit [1] - Device attached to Port 1</p> <p>Bit [2:15] - Reserved</p>

Table 12-21. OpenHCI Root Hub Descriptor B Register.

12.4.2.21 Hc_rhstatus Register

This is a two-part register that represents the Hub Status field and the Hub Status Change field

Bits	Name	Attr	Reset	Description
31	crwe	W	0	<p>ClearRemoteWakeUpEnable</p> <p>1 - Clears DeviceRemoteWakeUpEnable (drwe)</p> <p>0 - No effect</p>
30:18				Reserved
17	ocic	RW	0	<p>OverCurrentIndicatorChange</p> <p>This bit is set by hardware when a change has occurred to the OCI field of this Register. The HCD clears this bit by writing 1.</p> <p>Writing 0 has no effect.</p>
16	lpsc	RW	0	<p>(Read) LocalPowerStatusChange</p> <p>The Root Hub does not support the local power status feature; thus, this bit is always read as 0.</p> <p>(Write) SetGlobalPower</p> <p>In global power mode (psm = 0). This bit is written to 1 to activate power to all ports; i.e., clears the PortPowerStatus bit (pps). In per-port power mode, it sets pps only on ports whose PortPowerControl-Mask bit (ppcm) is not set. Writing 0 has no effect.</p>

Bits	Name	Attr	Reset	Description
15	drwe	RW	0	(Read) DeviceRemoteWakeupEnable This bit enables the ConnectStatusChange (see the csc bit of Hc_rhportstatus_1/2) as a resume event, causing a USBSUSPEND to US-BRESUME state transition and setting the ResumeDetected interrupt bit (rd). 0 - ConnectStatusChange is not a remote wakeup event 1 - ConnectStatusChange is a remote wakeup event (Write) SetRemoteWakeupEnable 0 - No effect 1 - Sets drwe
14:2				Reserved
1	oci	R	0	OverCurrentIndicator This bit reports over current status conditions when the global reporting is implemented. When set, an over current status condition exists. When cleared, all power operations are normal. If per-port over current status protection is implemented this bit is always 0
0	lps	RW	0	(Read) LocalPowerStatus The Root Hub does not support the local power status feature; thus, this bit is always read as 0. (Write) ClearGlobalPower In global power mode (PowerSwitchingMode psm = 0), 1 to turn off power to all ports; i.e., clears the PortPowerStatus bit (pps). In per-port power mode, it clears pps only on ports whose PortPowerControlMask bit (ppcm) is not set. Writing 0 has no effect.

Table 12-22. OpenHCI Root Hub Status Register.

12.4.2.22 Hc_rhportstatus_1/2 Registers

Bits	Name	Attr	Reset	Description
31:21				Reserved
20	prsc	RW	0	PortResetStatusChange This bit is set at the end of the 10-ms port reset signal. The HCD writes 1 to clear this bit. Writing 0 has no effect. 0 - Port reset is not complete 1 - Port reset is complete
19	ocic	RW	0	PortOverCurrentIndicatorChange This bit is valid only if over current conditions are reported on a per-port basis. This bit is set when Root Hub changes the PortOverCurrentIndicator bit (poci). The HCD writes 1 to clear this bit. Writing 0 has no effect. 0 - No change in poci 1 - The poci bit has changed

Bits	Name	Attr	Reset	Description
18	pssc	RW	0	<p>PortSuspendStatusChange This bit is set when the full resume sequence has been completed. This sequence includes the 20-s resume pulse, LS EOP, and 3-ms resynchronization delay. The HCD writes 1 to clear this bit. Writing 0 has no effect. This bit is also cleared when ResetStatusChange is set.</p> <p>0 - Resume is not completed 1 - Resume completed</p>
17	pesc	RW	0	<p>PortEnableStatusChange This bit is set when hardware events cause the PortEnableStatus bit (pes) to be cleared. Changes from the HCD writes do not set this bit. The HCD writes 1 to clear this bit. Writing 0 has no effect.</p> <p>0 - No change in pes 1 - Change in pes</p>
16	csc	RW	0	<p>ConnectStatusChange This bit is set when a connect or disconnect event occurs. The HCD writes 1 to clear this bit. Writing 0 has no effect. If CurrentConnectStatus bit (ccs) is cleared when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, this bit is set to force the driver to re-evaluate the connection status since these writes should not occur if the port is disconnected.</p> <p>0 - No change in ccs 1 - Change in ccs</p> <p>Note: If the DeviceRemovable[NDP] bit (dr) is set, this bit is set only after a Root Hub reset to inform the system that the device is attached.</p>
15:10		RW	0	Reserved
9	lsda	RW	0	<p>(Read) LowSpeedDeviceAttached This bit indicates the speed of the device attached to this port. When set, a Low-Speed device is attached to this port. When cleared, a Full-Speed device is attached to this port. This field is valid only when the CurrentConnectStatus bit (ccs) is set.</p> <p>0 - Full-speed device attached 1 - Low-speed device attached</p> <p>(Write) ClearPortPower The HCD clears the PortPowerStatus bit (pps) by writing 1 to this bit. Writing 0 has no effect.</p>

Bits	Name	Attr	Reset	Description
8	pps	RW		<p>(Read) PortPowerStatus This bit reflects the port's power status, regardless of the type of power switching implemented. This bit is cleared if an over current condition is detected. The HCD sets this bit by writing SetPortPower or SetGlobalPower. The HCD clears this bit by writing ClearPortPower or ClearGlobalPower. Which power control switches are enabled is determined by PowerSwitchingMode bit (psm) and PortPowerControlMask[NDP] bit (ppcm). In global switching mode (psm = 0), only Set/ClearGlobalPower controls this bit. In per-port power switching (psm = 1), if ppcm for the port is set, only Set/ClearPortPower commands are enabled. If the mask is not set, only Set/ClearGlobalPower commands are enabled. When port power is disabled, CurrentConnectStatus (ccs), PortEnableStatus (pes), PortSuspendStatus (pss), and PortResetStatus (prs) should be reset.</p> <p>0 - Port power is off 1 - Port power is on</p> <p>(Write) SetPortPower The HCD writes 1 to set the PortPowerStatus bit (pps). Writing 0 has no effect.</p>
7:5				Reserved
4	prs	RW	0	<p>(Read) PortResetStatus When this bit is set by a write to SetPortReset, port reset signaling is asserted. When reset is completed, this bit is cleared when PortResetStatusChange (prsc) is set. This bit cannot be set if CurrentConnectStatus (ccs) is cleared.</p> <p>0 - Port reset signal is not active 1 - Port reset signal is active</p> <p>(Write) SetPortReset The HCD sets the port reset signaling by writing 1 to this bit. Writing 0 has no effect. If CurrentConnectStatus (ccs) is cleared, this write does not set PortResetStatus (prs), but instead sets ConnectStatusChange (csc). This informs the driver that it attempted to reset a disconnected port.</p>
3	poci	RW	0	<p>(Read) PortOverCurrentIndicator This bit is only valid when the Root Hub is configured in such a way that over current conditions are reported on a per-port basis. If per-port over current reporting is not supported, this bit is set to 0. If cleared, all power operations are normal for this port. If set, an over current condition exists on this port. This bit always reflects the over current input signal</p> <p>0 - No over current condition. 1 - Over current condition detected.</p> <p>(Write) ClearSuspendStatus The HCD writes 1 to initiate a resume. Writing 0 has no effect. A resume is initiated only if PortSuspendStatus (pss) is set.</p>

Bits	Name	Attr	Reset	Description
2	pss	RW	0	<p>(Read) PortSuspendStatus This bit indicates the port is suspended or engaged in the resume sequence. It is set by a SetSuspendState write and cleared when PortSuspendStatusChange (pssc) is set at the end of the resume interval. This bit cannot be set if CurrentConnectStatus (ccs) is cleared. This bit is also cleared when PortResetStatusChange (prsc) is set at the end of the port reset or when the HC is placed in the USBRESUME state. If an upstream resume is in progress, it should propagate to the HC.</p> <p>0 - Port is not suspended 1 - Port is suspended</p> <p>(Write) SetPortSuspend The HCD sets the PortSuspendStatus bit (pss) by writing 1 to this bit. Writing 0 has no effect. If CurrentConnectStatus (ccs) is cleared, this write does not set PortSuspendStatus (pss); instead it sets the ConnectStatusChange bit (csc). This informs the driver that it attempted to suspend a disconnected port.</p>
1	pes	RW	0	<p>(Read) PortEnableStatus This bit indicates whether the port is enabled or disabled. The Root Hub may clear this bit when an over current condition, disconnect event, power switch-off, or operational bus error such as babble is detected. This change also causes the PortEnableStatusChange bit (pesc) to be set. The HCD sets this bit by writing SetPortEnable and clears it by writing ClearPortEnable. This bit cannot be set when the CurrentConnectStatus bit (ccs) is cleared. This bit is also set, if not already, at the completion of a port reset when ResetStatusChange is set or a port suspend when SuspendStatusChange is set.</p> <p>0 - Port is disabled 1 - Port is enabled</p> <p>(Write) SetPortEnable The HCD sets PortEnableStatus by writing 1 to the bit pes. Writing 0 has no effect. If the CurrentConnectStatus bit (ccs) is cleared, this write does not set PortEnableStatus (pes), but instead sets ConnectStatusChange (csc). This informs the driver that it attempted to enable a disconnected port.</p>
0	ccs	RW	0	<p>(Read) CurrentConnectStatus This bit reflects the current state of the downstream port.</p> <p>0 - No device connected 1 - Device connected</p> <p>(Write) ClearPortEnable The HCD writes 1 to this bit to clear the PortEnableStatus bit (pes). Writing 0 has no effect. The CurrentConnectStatus is not affected by a write.</p> <p>Note: This bit is always read 1 when the attached device is nonremovable (DeviceRemovable[NDP]).</p>

Table 12-23. OpenHCI Root Hub Port Status Change Register to Control and Report Port 1/2 Events.

12.4.3 USB Host Registers: EnhancedHCI Map

Register Offset	Register Name	Description
0x00	EHCI_caplength and EHCI_hciversion	Capability and Version
0x04	EHCI_hcsparams	Structural Parameter
0x08	EHCI_hccparams	Capability Parameter
0x0C		Reserved
0x10	EHCI_usbcm	USB Command
0x14	EHCI_usbsts	USB Status
0x18	EHCI_usbintr	USB Interrupt Enable
0x1C	EHCI_frindex	USB Frame Index
0x20		Reserved
0x24	EHCI_periodiclistbase	Periodic Frame List Base Address
0x28	EHCI_periodiclistbase	Asynchronous List Address
0x2C – 0x4F		Reserved
0x50	EHCI_configflag	Configured Flag
0x54	EHCI_portsc_1	Port 1 Status/Control
0x58	EHCI_portsc_2	Port 2 Status/Control

Table 12-24. EnhancedHCI Register Map.

12.4.4 USB Host Registers: EnhancedHCI Details

12.4.4.1 EHCI_caplength and EHCI_hciversion Register

Bits	Name	Attr	Reset	Description
31:16	hciversion	R	0x0100	EHCI Version supported
15:8				Reserved
7:0	caplength	R	0x10	Offset to start of operational registers

Table 12-25. EHCI Capability and Version Registers.

12.4.4.2 EHCI_hcsparams Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:20	dpn	R	0	Debug Port Number This register identifies which of the host controller ports is designated as the debug port. The value is the port number (one-based) of the debug port. A nonzero value indicates a debug port is present.
19:17				Reserved

Bits	Name	Attr	Reset	Description
16	p_indicator	R	0	Port Indicators This bit indicates whether the ports support Port Indicator Control. When this bit is 1, the Port Status and Control registers (EHCI_portsc_1/2) include a readable/writeable field for controlling the state of the port indicator.
15:12	n_cc	R	1	Number of Companion Controller This field indicates the number of companion controllers associated with the USB 2.0 host controller. A value of 0 indicates there are no companion host controllers. Port-ownership hand-off is not supported. Only high-speed devices are supported on the host controller root ports. A value larger than 0 indicates there are companion USB 1.1 host controller(s). Port-ownership hand-offs are supported. High-, Full- and Low-speed devices are supported on the host controller root ports.
11:8	n_pcc	R	0x2	Number of Ports per Companion Controller This field indicates the number of ports supported per companion host controller. It is used to communicate the port routing configuration to system software.
7	prr	R	0	Port Routing Rules This field specifies the method by which all ports are mapped to companion controllers. The first N_PCC ports are routed to the lowest-numbered function companion host controller, the next N_PCC ports are routed to the next lowest function companion controller, and so on.
6:5				Reserved
4	ppc	R	1	Port Power Control This field indicates whether the host controller implementation includes port power control. 1 - Ports have port power switches 0 - Ports do not have port power switches
3:0	n_ports	R	0x2	Number of Ports This field specifies the number of physical downstream ports implemented on the host controller. The value of this field determines how many port registers are addressable in the Operational Register Space.

Table 12-26. EHCI Structural Parameter Register.

12.4.4.3 EHCI_hccparams Register

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:8	eeco	R	0xA0	EHCI Extended Capabilities Pointer (EECP) This optional field indicates the existence of a capabilities list. A value of 0 indicates that 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.

Bits	Name	Attr	Reset	Description
7:4	ist	R	2	Isochronous Scheduling Threshold This field specifies, relative to the current position of the executing host controller, where the 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) prior to flushing the state. When bit [7] is 1, then host software assumes the host controller may cache an isochronous data structure for an entire frame.
3				Reserved
2	aspC	R	1	Asynchronous Schedule Park Capability If this bit is set to 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 EHCI_usbcmcmd Register.
1	pflf	R	1	Programmable Frame List Flag
0				Reserved

Table 12-27. EHCI Capability Parameter Register.

12.4.4.4 EHCI_usbcmcmd Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	itc	RW	0x8	Interrupt Threshold Control This field is used by the system software to select the maximum rate at which the host controller will issue interrupts. The only valid values are defined below. If the software writes an invalid value to this register, the results are undefined. Value: Maximum Interrupt Interval 0x00 - Reserved 0x01 - 1 micro-frame 0x02 - 2 micro-frames 0x04 - 4 micro-frames 0x08 - 8 micro-frames (default, equates to 1 ms) 0x10 - 16 micro-frames (2 ms) 0x20 - 32 micro-frames (4 ms) 0x40 - 64 micro-frames (8 ms) Software modifications to this bit when hhalted equals 0 will produce undefined behavior.
15:12				Reserved
11	aspme	RW	1	Asynchronous Schedule Park Mode Enable The software uses this bit to enable or disable Park mode. When this bit is 1, Park mode is enabled. When this bit is 0, Park mode is disabled.
10				Reserved

Bits	Name	Attr	Reset	Description
9:8	aspmc	RW	0x3	Asynchronous Schedule Park Mode Count Contains a count of the number of successive transactions the host controller may execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 0x1 to 0x3. Do not write 0 to these bits when aspme is 1 as this results in undefined behavior.
7	lhcr	RW	0	Light Host Controller Reset Allows the driver to reset the EHCI controller without affecting the state of the ports or the relationship to the companion host controllers. For example, the EHCI_portsc registers should not be reset to their default values and the cf bit of EHCI_configflag setting should not go to 0 (retaining port ownership relationships). If the host software reads this bit as 0, it indicates that the Light Host Controller Reset has completed and it is safe for the host software to re-initialize the host controller. When the host software reads this bit as 1, it indicates that the Light Host Controller Reset has not yet completed.
6	iaad	RW	0	Interrupt On Async Advance Doorbell This bit is used as a doorbell by the software to tell the host controller to issue an interrupt the next time it advances the asynchronous schedule. The software must write 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the EHCI_usbsts Interrupt On Async Advance status bit (iaa). If the EHCI_usbintr Interrupt On Async Advance Enable bit (iaae) 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 iaa to 1. The software should not write 1 to this bit when the asynchronous schedule is disabled. Doing so will yield undefined results.
5	ase	RW	0	Asynchronous Schedule Enable This bit controls whether the host controller skips processing the Asynchronous Schedule as follows: 0 - Do not process the Asynchronous Schedule 1 - Use EHCI_periodiclistbase to access the Asynchronous Schedule.
4	pse	RW	0	Periodic Schedule Enable This bit controls whether the host controller skips processing the Periodic Schedule as follows: 0 - Do not process the Periodic Schedule 1 - Use EHCI_periodiclistbase to access the Periodic Schedule.
3:2	fls	RW	0	Frame List Size This field specifies the size of the frame list. The size the frame list controls which bits in EHCI_frlindex should be used for the Frame List Current index. Values mean: 0x0 - 1024 elements (4096 bytes) Default value 0x1 - 512 elements (2048 bytes) 0x2 - 256 elements (1024 bytes) – for resource-constrained environments 0x3 - Reserved

Bits	Name	Attr	Reset	Description
1	hreset	RW	0	<p>Host Controller Reset</p> <p>The software uses this control bit to reset the host controller. The effect of this event on Root Hub registers is similar to the effect of a Chip Hardware Reset. When the software writes 1 to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines, etc. to their initial values. Transactions currently in progress on USB are immediately terminated. A USB reset is not driven on downstream ports. All operational registers, including port registers and port state machines are set to their initial values, and port ownership reverts to the companion host controller. The software must reinitialize the host controller in order to return the host controller to an operational state.</p> <p>This bit is set to 0 by the Host Controller when the reset process is complete. The software cannot terminate the reset process early by writing 0 to this Register. The software should not set this bit to 1 when the EHCI_usbsts HC Halted bit (hchalted) is 0. Attempting to reset an actively running host controller will result in undefined behavior.</p>
0	rs	RW	0	<p>Run/Stop</p> <p>1 - Run</p> <p>0 - Stop</p> <p>When set to 1, the Host Controller proceeds with execution of the schedule. The Host Controller continues execution as long as this bit is set to 1. When this bit is set to 0, the Host Controller completes the current and actively pipelined transactions on the USB and then halts. The Host Controller must halt within 16 micro-frames after the software clears the run bit. The EHCI_usbsts HC Halted bit (hchalted) indicates when the Host Controller has finished its pending pipelined transactions and has entered the stopped state. The software must not write 1 to this field unless the host controller is in the Halted state (hchalted = 1). Otherwise, undefined results will occur.</p>

Table 12-28. EHCI USB Command Register.

12.4.4.5 EHCI_usbsts Register

Bits	Name	Attr	Reset	Description
31:16				Reserved
15	ass	R	0	<p>Asynchronous Schedule Status</p> <p>The bit reports the current real status of the Asynchronous Schedule. If this bit is 0 then the status of the Asynchronous Schedule is disabled. If this bit is 1 then the status of the Asynchronous Schedule is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when the software transitions the EHCI_usbcm Asynchronous Schedule Enable bit (ase). When this bit and ase are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0).</p>

Bits	Name	Attr	Reset	Description
14	pss	R	0	<p>Periodic Schedule Status</p> <p>The bit reports the current real status of the Periodic Schedule. If this bit is 0 then the status of the Periodic Schedule is disabled. If this bit is 1 then the status of the Periodic Schedule is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when the software transitions the EHCI_usbcm Periodic Schedule Enable bit (pse). When this bit and pse are the same value, the Periodic Schedule is either enabled (1) or disabled (0).</p>
13	rec	R	0	<p>Reclamation</p> <p>This is a read-only status bit, which is used to detect an empty asynchronous schedule.</p>
12	hchalted	R	1	<p>HCHalted</p> <p>This bit is 0 when the EHCI_usbcm Run/Stop bit (rs) is 1. The Host Controller sets this bit to 1 after it has stopped executing as a result of the Run/Stop bit being set to 0, either by the software or by the Host Controller hardware (e.g., due to an internal error).</p>
11:6				Reserved
5	iaa	R_WC	0	<p>Interrupt on Async Advance</p> <p>System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing 1 to the Interrupt on EHCI_usbcm Async Advance Doorbell bit (iaad). This status bit indicates the assertion of that interrupt source.</p>
4	hse	R_WC	0	<p>Host System Error</p> <p>The Host Controller sets this bit to 1 when a serious error occurs during a host system access attempt involving the Host Controller module. When this error occurs, the Host Controller clears the EHCI_usbcm Run/Stop bit (rs) to prevent further execution of the scheduled TDs.</p>
3	flr	R_WC	0	<p>Frame List Rollover</p> <p>The Host Controller sets this bit to 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 EHCI_usbcm register) is 1024, EHCI_findex rolls over when EHCI_findex[13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to 1 when EHCI_findex[12] toggles.</p>
2	pcd	R_WC	0	<p>Port Change Detect</p> <p>The Host Controller sets this bit to 1 when a port for which the Port Owner bit is set to 0 has a change-bit transition from 0 to 1 or a Force Port Resume fpr bit-transition from 0 to 1 as a result of a J-K transition detected on a suspended port. This bit will also be set when the Connect Status Change bit (csc) is set to 1; i.e., after system software has relinquished ownership of a connected port by writing 1 to a port's Port Owner bit (po). This bit can be maintained in the Auxiliary power well. Alternatively, on a D3-to-D0 transition of the EHCI HC device, this bit can also be loaded with the OR of all of the EHCI_portsc change-bits (e.g., force port resume, over-current change, enable/disable change, and connect status change).</p>

Bits	Name	Attr	Reset	Description
1	usberrint	R_WC	0	USB Error Interrupt The Host Controller sets this bit to 1 when completion of a USB transaction results in an error condition (e.g., error counter underflow). If the Transfer Descriptor on which the error interrupt occurred also had its Interrupt on Complete bit (IOC) set, then both this bit and usbint are set.
0	usbint	R_WC	0	USB Interrupt The Host Controller sets this bit to 1 on the completion of a USB transaction, which results in the retirement of a Transfer Descriptor that had its Interrupt on Complete bit (IOC) set. The Host Controller also sets this bit to 1 when a short packet is detected (actual number of bytes received is less than the expected number of bytes).

Table 12-29. EHCI USB Status Register.

12.4.4.6 EHCI_usbintr Register

Bits	Name	Attr	Reset	Description
31:6				Reserved
5	iaae	RW	0	When this bit is 1, and the EHCI_usbsts Interrupt on Async Advance bit (iaa) is 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged when the software clears iaa .
4	hsee	RW	0	When this bit is 1, and the EHCI_usbsts Host System Error Status bit (hse) is 1, the host controller will issue an interrupt. The interrupt is acknowledged by the software clearing hse .
3	fle	RW	0	When this bit is 1, and the EHCI_usbsts Frame List Rollover bit (fle) is 1, the host controller will issue an interrupt. The interrupt is acknowledged by the software clearing fle .
2	pcie	RW	0	When this bit is 1, and the EHCI_usbsts Port Change Detect bit (pcd) is 1, the host controller will issue an interrupt. The interrupt is acknowledged by the software clearing pcd .
1	usbeie	RW	0	When this bit is 1, and the EHCI_usbsts USB Interrupt bit (usberrint) is 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by the software clearing usberrint .
0	usbie	RW	0	When this bit is 1, and the EHCI_usbsts USB Interrupt bit (usberrint) is 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by the software clearing usbint .

Table 12-30. EHCI USB Interrupt Enable Register.

12.4.4.7 EHCI_frindex Register

Bits	Name	Attr	Reset	Description
31:14				Reserved
13:0	frindex	RW	0	<p>Frame Index</p> <p>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 fsls in the EHCI_usbcmdu register:</p> <ul style="list-style-type: none"> fsls = 0x0 - 1024 elements and N = 12 fsls = 0x1 - 512 elements and N = 11 fsls = 0x2 - 256 elements and N = 10 fsls = 0x3 - Reserved

Table 12-31. EHCI USB Frame Index Register.

12.4.4.8 EHCI_periodiclistbase Register

Bits	Name	Attr	Reset	Description
31:12	ba	RW	0	<p>Base Address</p> <p>These bits correspond to memory address signals [31:12], respectively.</p>
11:0				Reserved and must be 0x000

Table 12-32. EHCI Periodic Frame List Base Address Register.

12.4.4.9 EHCI_asynclistaddr Register

Bits	Name	Attr	Reset	Description
31:5	lpl	RW	NR	<p>Link Pointer Low</p> <p>These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH).</p>
4:0				Reserved

Table 12-33. EHCI Asynchronous List Address Register.

12.4.4.10 EHCI_configflag Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	cf	RW	0	<p>Configure Flag</p> <p>Host software sets this bit as the last action in the process of configuring the Host Controller. This bit controls the default port-routing control logic. Bit values and side-effects are listed below.</p> <p>0 - Port routing control logic default-routes each port to an implementation-dependent classic host controller.</p> <p>1 - Port routing control logic default-routes all ports to the host controller.</p>

Table 12-34. EHCI Configured Flag Register.
12.4.4.11 EHCI_portsc_1/2 Registers

Bits	Name	Attr	Reset	Description
31:23				Reserved
22	wkoc_e	RW	0	<p>Wake on Over Current Enable</p> <p>Writing this bit to 1 configures the port to recognize over current conditions as wake-up events.</p> <p>This field is 0 if Port Power bit (pp) is 0.</p>
21	wkdscnnt_e	RW	0	<p>Wake on Disconnect Enable</p> <p>Writing this bit to 1 configures the port to recognize device disconnects as wake-up events.</p> <p>This field is 0 if pp is 0.</p>
20	wkcne_e	RW	0	<p>Wake on Connect Enable</p> <p>Writing this bit to 1 configures the port to recognize device connects as wake-up events.</p> <p>This field is 0 if pp is 0.</p>
19:16	ptc	RW	0	<p>Port Test Control</p> <p>When this field is 0, the port is NOT operating in a test mode.</p> <p>Non-zero values indicate test mode operation and specific test modes. The encoding of the test mode bits are:</p> <p>0x0 - Test mode not enabled 0x1 - Test J_STATE 0x2 - Test K_STATE 0x3 - Test SE0_NAK 0x4 - Test Packet 0x5 - Test FORCE_ENABLE 0x6 through 0xF - Reserved</p>
15:14	pic	RW	0	<p>Port Indicator Control</p> <p>Writing to these bits has no effect if the EHCI_hcsparams register p_indicator bit is 0. If p_indicator is 1, then the bit encodings are:</p> <p>0x0 - Port indicators are off 0x1 - Amber 0x2 - Green 0x3 - Undefined</p>

Bits	Name	Attr	Reset	Description
13	po	RW	1	<p>Port Owner</p> <p>This bit unconditionally reverts to 0 when the EHCI_configflag configured bit (cf) transitions from 0 to 1. This bit unconditionally reverts to 1 when cf is 0.</p> <p>The 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). The software writes 1 to this bit when the attached device is not a high-speed device. When this bit is 1 the companion host controller owns and controls the port.</p>
12	pp	RW	0	<p>Port Power</p> <p>This bit represents the current setting of the port power switch (0 = off, 1 = on). When power is unavailable on a port (pp = 0), the port is nonfunctional and will not report attaches, detaches, etc. When an over current condition is detected on a powered port and the EHCI_hcsparams bit ppc is 1, the pp bit in each affected port may be transitioned by the host controller from 1 to 0 (removing power from the port).</p>
11:10	ls	R	0	<p>Line Status</p> <p>These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. These bits are used for detection of low-speed USB devices prior to the port reset and enable sequence. This field is valid only when the Port Enable bit (ped) is 0 and the current connect status bit is set to 1.</p> <p>0x0 - USB state of SE0. This is not a low-speed device, EHCI reset should be performed.</p> <p>0x1 - USB K_STATE. This is a low-speed device, and ownership of port should be released.</p> <p>0x2 - USB J_STATE. This is not a low-speed device, and EHCI reset should be performed.</p> <p>0x3 - USB State is Undefined. This is not a low-speed device, and EHCI reset should be performed.</p> <p>This field is undefined if Port Power (pp) is 0.</p>
9				Reserved

Bits	Name	Attr	Reset	Description
8	pr	RW	0	<p>Port Reset 1 - Port is in Reset 0 - Port is not in Reset When the software writes 1 to this bit (from 0), the bus reset sequence is initiated per the <i>Universal Serial Bus Revision 2.0</i> specification. The software writes 0 to this bit to terminate the bus reset sequence and must maintain this bit at 1 long enough to ensure the reset sequence completes. When the software writes this bit to 1, it must also write 0 to the Port Enable bit (ped). When the software writes this bit to 0 there may be a delay before the bit status changes to 0. The bit status will not read 0 until after the reset completes. If the port is in high-speed mode after reset is complete, the host controller will automatically enable this port (e.g., set the Port Enable bit to 1). A host controller must terminate the reset and stabilize the state of the port within 2 ms of transitioning this bit from 1 to 0. For example, if the port detects that the attached device is in high-speed mode during reset, then the host controller must bring the port to the enabled state within 2 ms of writing this bit to 0. The EHCI_usbsts bit hchalted should be 0 before the software attempts to use this bit. The host controller may hold Port Reset pr asserted to 1 when hchalted is 1. This field is 0 if Port Power (pp) is 0.</p>
7	spd	RW	0	<p>Suspend 1 - Port in suspend state 0 - Port not in suspend state Port Enabled (ped) and Suspend (spd) bits of this register define the port states: Disable - ped = 0 and spd = X Enable - ped = 1 and spd = 0 Suspend - ped = 0 and spd = 1 In suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction, if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and there may be a delay in suspending a port if there is a transaction currently in progress on the USB. A write of 0 to this bit is ignored by the host controller. The host controller will unconditionally set this bit to 0 when: The software sets the Force Port Resume bit (fpr) to 0 (from 1). The software sets the Port Reset bit (pr) to 1 (from 0). If the host software sets this bit to 1 when the port is not enabled (ped = 0) the results are undefined. This field is 0 if Port Power (pp) is 0.</p>

Bits	Name	Attr	Reset	Description
6	fpr	RW	0	<p>Force Port Resume 1 - Resume detected/driven on port 0 - No resume (K_STATE) detected/driven on port Functionality defined for manipulating this bit depends on the value of the Suspend bit (spd). For example, if the port is not suspended (spd = ped = 1) and the software transitions this bit to 1, then the effects on the bus are undefined. The 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 1 because a J-to-K transition is detected, the Port Change Detect bit (pcd) also is set to 1. If the software sets this bit to 1, the host controller must not set pcd.</p> <p>Note that when the EHCI controller maintains ownership of the port, the resume sequence adheres to the standard defined in the <i>Universal Serial Bus Revision 2.0</i> specification. Resume signaling (Full-speed 'K') is driven on the port while this bit remains 1. The software must appropriately time the Resume and set this bit to 0 when the time has elapsed. Writing 0 (from 1) causes the port to return to high-speed mode (forcing the bus below the port into a high-speed idle). This bit will remain 1 until the port has switched to the high-speed idle. The host controller must complete this transition within 2 ms after the software has set this bit to 0. This field is 0 if Port Power (pp) is 0.</p>
5	occ	R_WC	0	<p>Over-Current Change 1 - This bit is set to 1 when there is a change in status to over current active. This bit is cleared by a write of 1.</p>
4	oca	R	0	<p>Over-Current Active 1 - This port currently has an over-current condition 0 - This port does not have an over-current condition This bit will automatically transition from 1 to 0 when the over current condition is removed.</p>
3	pedc	R_WC	0	<p>Port Enable/Disable Change 1 - Port enabled/disabled status has changed 0 - No change For the root hub, this bit is set to 1 only when a port is disabled due to the appropriate conditions existing at the EOF2 point. This bit is cleared by writing it to 1. This field is 0 if Port Power (pp) is 0.</p>

Bits	Name	Attr	Reset	Description
2	ped	RW	0	<p>Port Enabled/Disabled 1 - Enable 0 - Disable</p> <p>Ports can only be enabled by the host controller as a part of the reset and enable process. The software cannot enable a port by writing 1 to this field. The host controller will set this bit to 1 only when the reset sequence determines that the attached device is a high-speed device. 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 various host controller and bus events. When the port is disabled (0) downstream propagation of data is blocked on this port, with the exception of reset.</p> <p>This field is 0 if Port Power (pp) is 0.</p>
1	csc	R_WC	0	<p>Connect Status Change 1 - Change in Current Connect Status 0 - No change</p> <p>Indicates a change has occurred in the port's Current Connect Status (ccs). The host controller sets this bit for all changes to the port device connect status, even if the system software has not cleared an existing connect status change. The software sets this bit to 0 by writing 1 to it.</p> <p>This field is 0 if Port Power (pp) is 0.</p>
0	ccs	R	0	<p>Current Connect Status 1 - Device is present on port 0 - No device is present</p> <p>This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (csc) to be set.</p> <p>This field is 0 if Port Power (pp) is 0.</p>

Table 12-35. EHCI Status/Control Registers for Ports 1/2.

13. FLASH AND SD CONTROLLERS

The A12 provides a Flash I/O subsystem, a NAND Flash Controller and up to three SD Controllers. For the purposes of this manual, these will be collectively referred to as the Flash/SD Interface. The A12 Flash/SD Interface is covered in this chapter as follows:

- ([Section 13.1](#)) Flash/SD: Overview
- ([Section 13.2](#)) Flash/SD: Flash I/O Subsystem
- ([Section 13.3](#)) Flash/SD: NAND Flash Controller
- ([Section 13.4](#)) Flash/SD: SD PHY
- ([Section 13.5](#)) Flash/SD: SD Controllers

Refer to [Chapter 2](#) for NAND Flash and SD eMMC boot mode details. For interface pins and descriptions, refer to the A12 chip datasheet.

13.1 Flash/SD: Overview

The table below summarizes the A12 Flash and SD controllers. This section continues with a summary of features ([Section 13.1.1](#)) and a block diagram of the subsystem and its controllers ([Section 13.1.2](#)).

Controllers	Description
Flash I/O Controller Section 13.2	Controls multiplexing of external pins between NAND Flash and SD controllers. Provides a DMA interface for the NAND Flash controller.
NAND Flash Controller Section 13.3	Provides an interface to external NAND Flash memory.
SD Controllers Section 13.5	Provide interfaces to external SD devices. Includes a built-in DMA controller.

Table 13-1. Flash and SD Controller Interfaces.

Controllers are activated by programming the Flash I/O Control register ([Section 13.2.9.1](#)).

13.1.1 Overview: Features

The Flash I/O Subsystem, NAND Flash Controller and the SD Controllers enable interaction with external NAND Flash and SD / SDIO / SDHC / SDXC / eMMC devices. Features of the Flash/SD Interface include:

- Flash I/O Subsystem
 - Enables both NAND Flash and SD Controllers and controls multiplexing of external pins
 - Programming of DMA / FIFO transactions for NAND Flash
 - Enables Flash DMA dual-space mode for the NAND Flash Controller
 - Programming of BCH error correction for the NAND Flash Controller
- NAND Flash controller
 - Up to 4-GB device, 512-B and 2-KB page sizes
 - 8-bit flash chip data bus
 - SLC with ECC hardware and read confirm support
 - BCH error correction and increased spare area available
 - NAND Flash Boot option
- SD controllers
 - For information regarding specific SD controller features, please refer to the relevant A12 chip datasheet.

13.1.2 Overview: Block Diagram

The Flash I/O Subsystem utilizes a CPU interface and an interface to the AHB FDMA Engine ([Chapter 4 “DMA Engine Subsystem”](#)). The NAND Flash uses the AHB FDMA Engine for DMA transfers to/from main memory. The FDMA engine offers normal and dual-space mode operation for fast DMA transactions. There is a DMA FIFO Controller for the 2KB Data FIFO that buffers data from/to the FDMA engine for data write/read requests to/from the NAND Flash controller. The VIC includes interrupts for the subsystem, including the NAND Flash controller CPU Command Done and DMA Command Done Interrupts.

The SD Controller uses its own bus mastering DMA engine and does not rely on the AHB DMA Engine for DMA transfers.

[Figure 13-1](#) on the following page shows the functional blocks of the Flash I/O Subsystem.

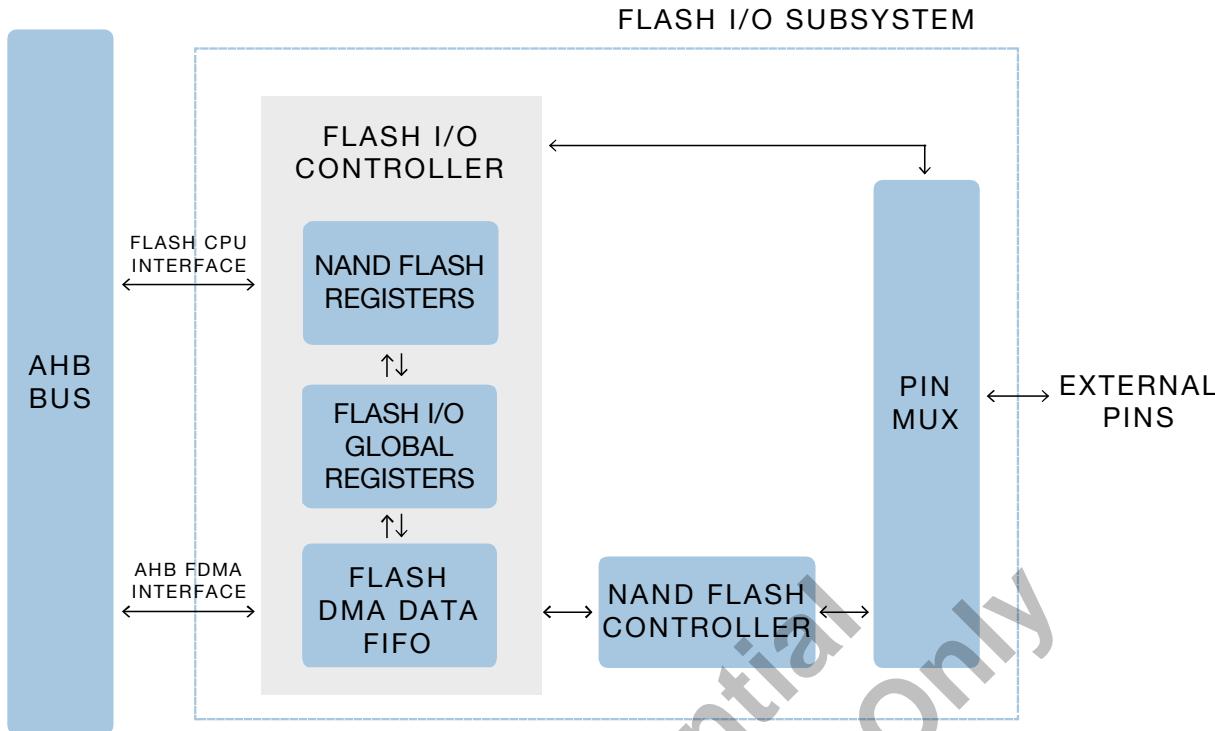


Figure 13-1. Functional Relationship Between the Flash I/O Subsystem.

13.2 Flash/SD: Flash I/O Subsystem

This section discusses address space, programming and registers for the overall Flash I/O Subsystem as follows:

- (Section 13.2.1) Flash I/O: Address Space
- (Section 13.2.2) Flash I/O: Operation
- (Section 13.2.3) Flash I/O: BCH Error Correction and Dual-Space Mode
- (Section 13.2.4) Flash I/O: BCH Codec Protected Range
- (Section 13.2.5) Flash I/O: Enabling BCH and Dual-space Mode
- (Section 13.2.6) Flash I/O: Subsystem Initialization
- (Section 13.2.7) Flash I/O: Activating Controllers
- (Section 13.2.8) Flash I/O: Register Map
- (Section 13.2.9) Flash I/O: Register Details
- For NAND Flash Controller registers and programming see Section 13.3.
- For SD Controller registers and programming see Section 13.5.

13.2.1 Flash I/O: Address Space

The Flash I/O Subsystem occupies two 4-KB address blocks on the AHB. One block is for the CPU interface of the Flash I/O subsystem ([Section 13.2.1.1](#)). The other block is for its DMA interface ([Section 13.2.1.2](#)).

13.2.1.1 Flash I/O: CPU Interface Address Space

The base address space starting at 0xE000.1000 is designated for the Flash I/O CPU interface. This range is further divided into sub-ranges for the control and status registers (CSRs) of the Flash I/O Subsystem. The table below shows the AHB bus transaction information when the CPU address space is accessed.

Access Targets	AHB Offset	AHB Bus Access	AHB Bus Data Size	AHB Bus Burst Types
Flash I/O Global Registers	0x1000 – 0x107F	RW	4 bytes	SINGLE
FIFO DMA Controller Registers	0x1080 – 0x10FF	RW	4 bytes	SINGLE
Flash NAND Controller Registers	0x1100 – 0x117F	RW	4 bytes	SINGLE

Table 13-2. Flash I/O Subsystem CPU Interface Address Space.

13.2.1.2 Flash I/O: DMA Address Space

The address space beginning at 0xE000.0000 is for the Flash DMA Data FIFO interface. This address range is used by the AHB DMA Engine for DMA transfers between the main memory and the Flash Controller.

Note that the SD Controllers maintain their own register address space and DMA engine, and therefore do not use the Flash I/O DMA interface to perform DMA transfers to or from main memory.

The DMA address space is for accessing the Data FIFO using the AHB DMA Engine and the Cortex-A9 host processor. The table below provides the AHB bus transaction information during DMA address space access.

Master	Random Read Mode	Address Offset From CPU Slave Base Address	AHB Bus Access	AHB Bus Data Size	AHB Bus Burst Types
DMA Engine	No	Ignored - FIFO access	RW	4 bytes	SINGLE, INCR, INCR4, INCR8, INCR16
ARM Core	No	Ignored - FIFO access	RW	4 bytes	SINGLE
ARM Core	Yes	Location to read	R	4 bytes	SINGLE

Table 13-3. Flash I/O Subsystem DMA Interface Address Space.

13.2.2 Flash I/O: Operation

Once the Flash I/O Subsystem is initialized and configured, operations may be initiated in the memory targets (Flash chips). The two basic types of operations are:

1. Data transfer operations such as read/write of the memory targets or the DMA transfers of data between main memory and the Flash I/O Subsystem.

2. Non-data transfer operations such as the configuration of memory targets, the retrieval of read memory status information, and commands that do not require block data transfers, such as erase and reset.

13.2.3 Flash I/O: BCH Error Correction and Dual-Space Mode

The A12 chip supports 1-bit error correcting code (ECC) operations and Bose-Chaudhuri-Hocquenghem (6- or 8-bit) error correction. The DMA FIFO Controller must be in dual-space mode to enable the BCH codec controller and BCH encoding / decoding.

Enabling dual-space mode ([Section 13.2.5](#)) improves the efficiency of the BCH codec controller by interleaving the read/write sequence between memory and the data FIFO. Note that programming is required to prepare main-area and spare-area space in advance. Dual-space mode is also used for Flash chips with only one partial programming cycle; i.e., the operational mode that can simultaneously access Flash main and spare areas.

Operations performed in dual-space mode proceed according to the following general steps:

1. The BCH codec controller generates handshake signals with the BCH encoder/decoder and enables transfers between memory and the BCH encoder/decoder.
2. Initially, the FDMA engine transfers one 2-KB page of data with the interleave sequence of 4 x 512 B of main-area data and 32 B of spare-area data. The DMA FIFO Controller stores the main- and spare-area data in the Data FIFO buffer and Spare RAM, respectively. Note that the lowest latency is achieved if the DMA FIFO Controller uses the same interleave sequence as the FDMA engine.
3. When the first pair of data is ready, the BCH Codec Controller begins fetching data and feeding it uncoded to the BCH encoder. After BCH parity is reached, the BCH Codec Controller writes the coded data back to Spare RAM. Meanwhile, the DMA FIFO Controller begins fetching the main-area data (whole) from the Data FIFO buffer and sending it to the NAND Flash Controller. The DMA FIFO Controller then begins reading spare-area data from the Spare RAM when the parity data are ready.

13.2.4 Flash I/O: BCH Codec Protected Range

The following diagrams illustrate the Protected Ranges for the BCH Codec:

- Section 13.2.4.1 “Flash BCH Category 1 - Small Page (512 + 16 Bytes), x8 NAND”
- Section 13.2.4.2 “Flash BCH Category 2 - Small Page (256 + 8 Words), x16 NAND”
- Section 13.2.4.3 “Flash BCH Category 3 - Large Page (2K + 64 Bytes), x8 NAND”
- Section 13.2.4.4 “Flash BCH Category 4 - Large Page (1K + 32 Words), x16 NAND”
- Section 13.2.4.5 “Flash BCH Category 5 - Large Spare Page (2K + 128 Bytes), x8 NAND”
- Section 13.2.4.6 “Flash BCH Category 6 - Large Spare Page (1K + 64 Words), x16 NAND”

13.2.4.1 Flash BCH Category 1 - Small Page (512 + 16 Bytes), x8 NAND

Protected Range Category 1 is not supported by the A12 6-bit / 8-bit BCH codec.

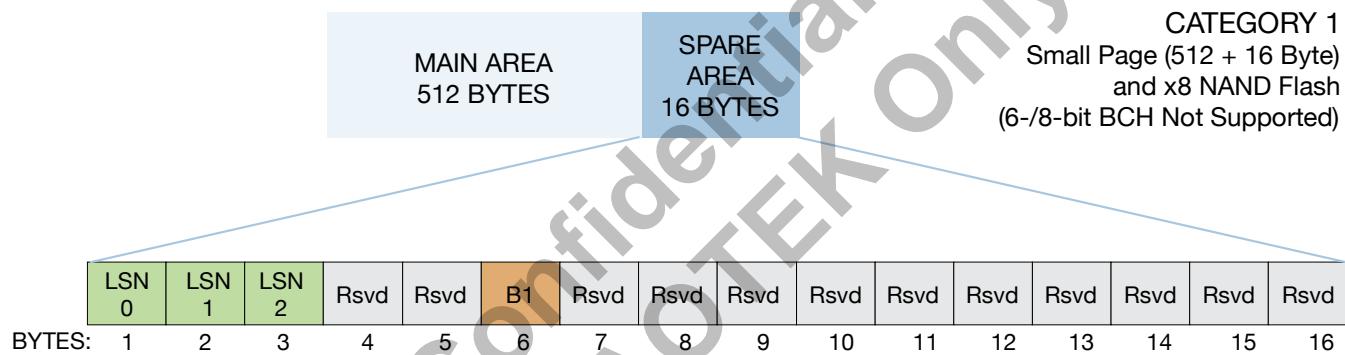


Figure 13-2. Flash BCH Codec Protected Range Category 1: Small Page (512 +16 Bytes) and x8 NAND.

13.2.4.2 Flash BCH Category 2 - Small Page (256 + 8 Words), x16 NAND

Protected Range Category 2 is not supported by the A12 6-bit / 8-bit BCH codec.

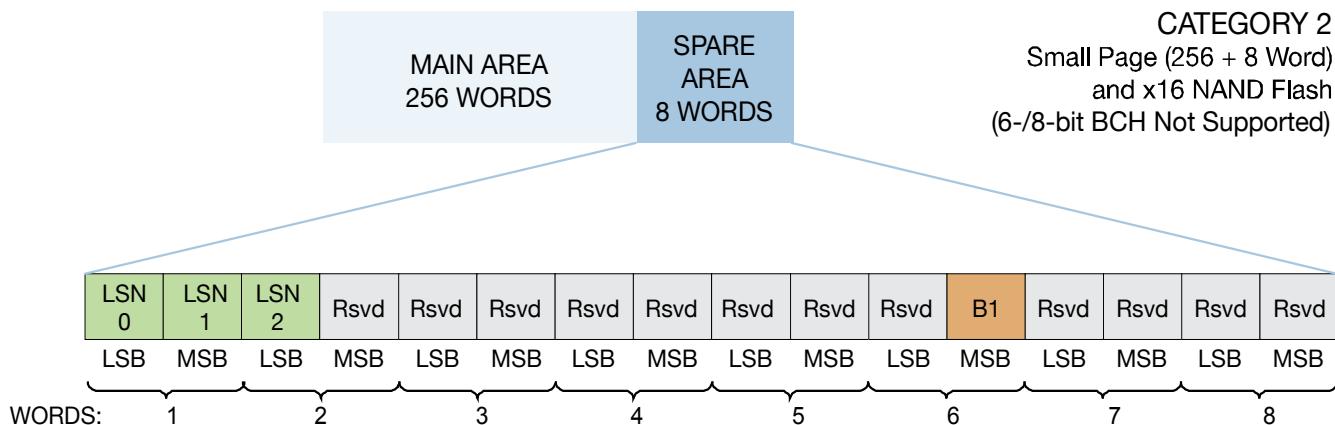


Figure 13-3. Flash BCH Codec Protected Range Category 2: Small Page (256 + 8 Words) and x16 NAND.

13.2.4.3 Flash BCH Category 3 - Large Page (2K + 64 Bytes), x8 NAND

The A12 chip supports Protected Range Category 3 for the 6-bit BCH codec; i.e., `Flash_IO_control` (0x000) bit 5 set to 0 ($t = 6$). The protected message includes all data in the main area and the first six bytes of each 16-byte subset in the spare area. To function correctly, `Flash_IO_dsm_control` register bits `main_jp_size` must be 0x9 and `spare_jp_size` must be 0x4. The DMA transaction byte count at `Flash_IO_dma_control` (0x080) bits `count` must be a multiple of 2112 bytes.

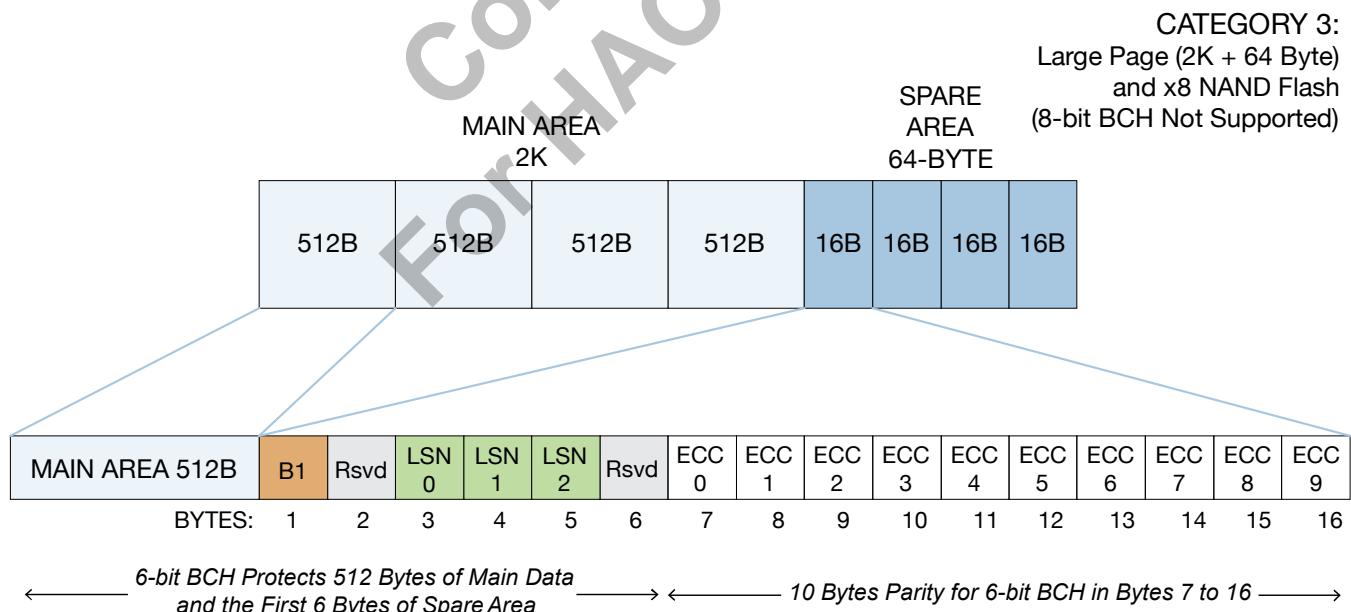


Figure 13-4. Flash BCH Codec Protected Range Category 3: Large Page (2K + 64 Bytes) and x8 NAND.

13.2.4.4 Flash BCH Category 4 - Large Page (1K + 32 Words), x16 NAND

The A12 chip supports Protected Range Category 4 for the 6-bit BCH codec; i.e., **Flash_IO_control** (0x000) bit 5 set to 0 ($t = 6$). The protected message includes all data in the main area and the first three words of each eight-word subset in spare area. **Flash_IO_dsm_control** register bits **main_jp_size** is 0x9 and **spare_jp_size** must be 0x4. The DMA transaction byte count at **Flash_IO_dma_control** (0x080) bits **count** must be a multiple of 2112 bytes.

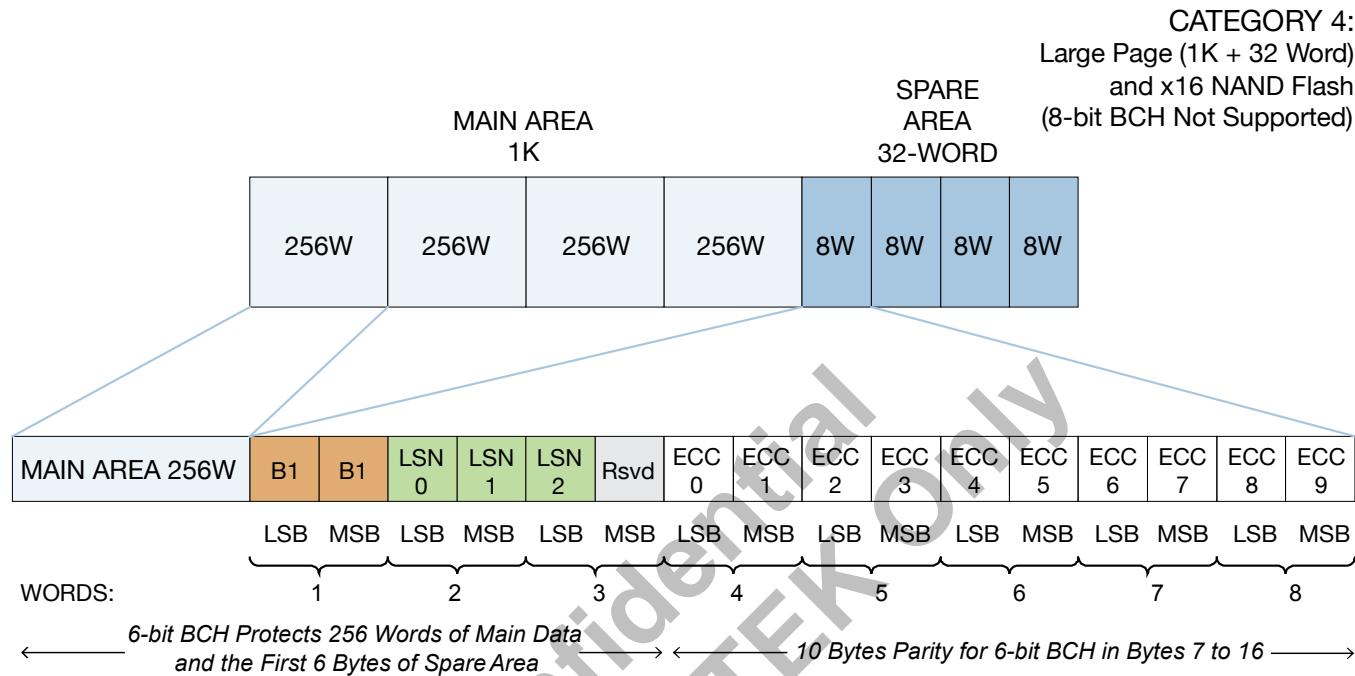


Figure 13-5. Flash BCH Codec Protected Range Category 4: Large Page (1K + 32 Words) and x16 NAND.

13.2.4.5 Flash BCH Category 5 - Large Spare Page (2K + 128 Bytes), x8 NAND

The A12 chip supports Protected Range Category 5 for both 6-bit and 8-bit BCH codecs; i.e., **Flash_IO_control** (0x000) bit 5 set to 0 ($t = 6$) or 1 ($t = 8$). The 6-bit BCH layout is the same for BCH Categories 3 and 5, with the unused spare area being reserved.

For 8-bit BCH, the protected message range includes all data in the main area and the first 19 bytes of each 32-byte subset in the spare area. To function correctly with 8-bit BCH, **Flash_IO_dsm_control** register bits **main_jp_size** must be 0x9 and **spare_jp_size** must be 0x5. The DMA transaction byte count at **Flash_IO_dma_control** (0x080) bits **count** must be a multiple of 2176 bytes.

Refer to [Figure 13-6](#) on the following page.

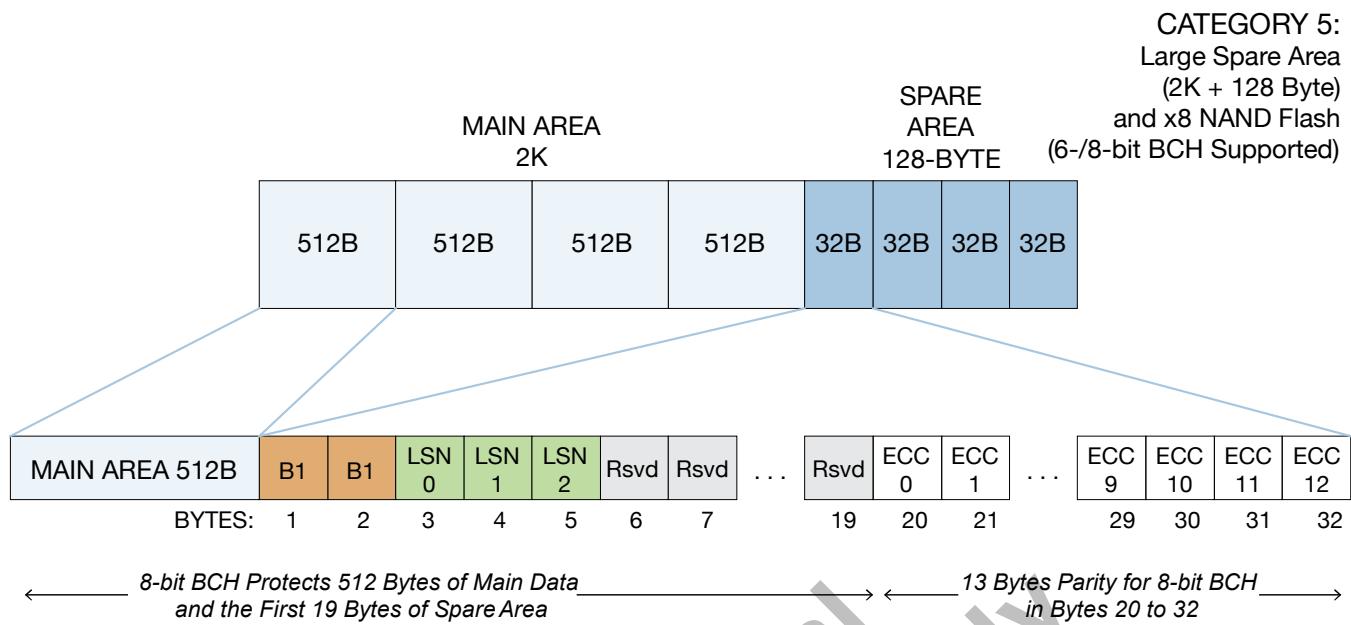


Figure 13-6. Flash BCH Codec Protected Range Category 5: Large Spare Page (2K + 128 Bytes), x8 NAND.

13.2.4.6 Flash BCH Category 6 - Large Spare Page (1K + 64 Words), x16 NAND

The A12 chip supports Protected Range Category 6 for both 6-bit and 8-bit BCH codecs; i.e., `Flash_IO_control` (0x000) bit 5 set to 0 ($t = 6$) or 1 ($t = 8$). The 6-bit BCH layout is the same for BCH Categories 4 and 6, with the unused spare area being reserved.

For 8-bit BCH, the protected message includes all data in the main area and the first 19 bytes of each 16-word subset in the spare area. To function correctly with 8-bit BCH, `Flash_IO_dsm_control` register bits `main_jp_size` must be 0x9 and `spare_jp_size` must be 0x5. The DMA transaction byte count at `Flash_IO_dma_control` (0x080) bits `count` must be a multiple of 2176 bytes.

Refer to [Figure 13-7](#) on the following page.

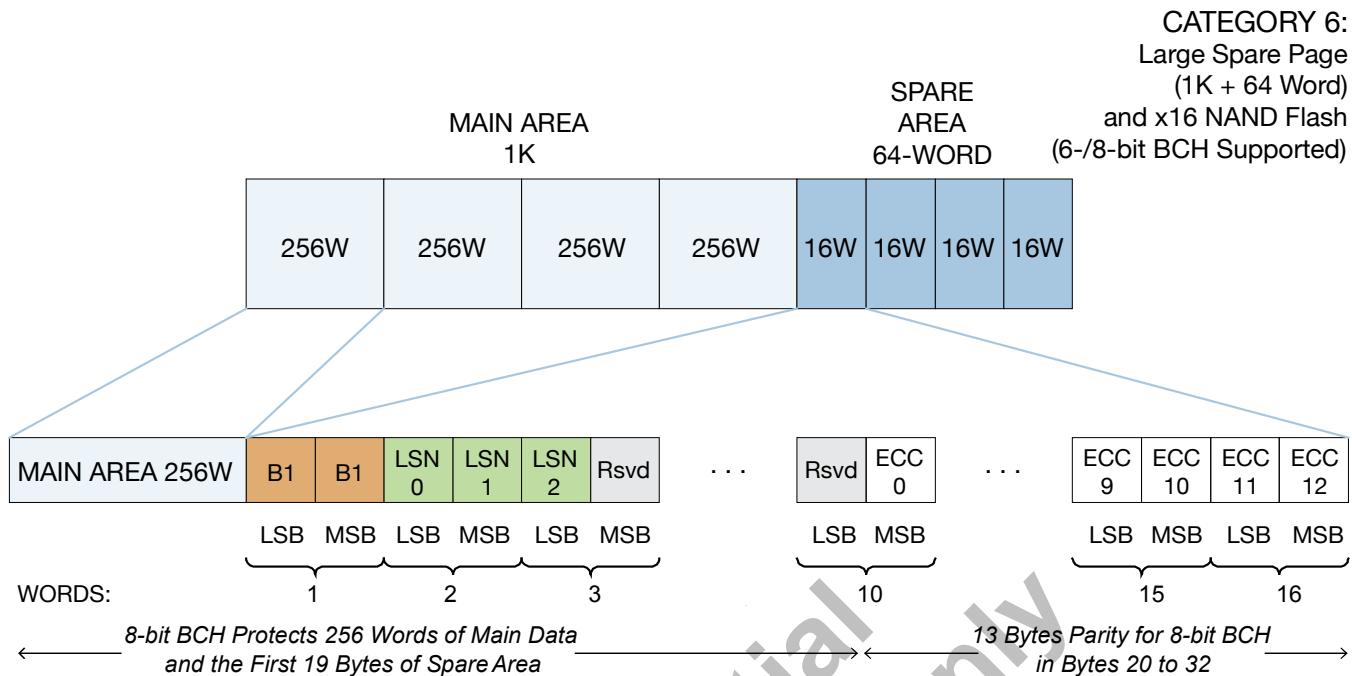


Figure 13-7. Flash BCH Codec Protected Range Category 6: Large Spare Page (1K + 64 Words), x16 NAND.

13.2.5 Flash I/O: Enabling BCH and Dual-space Mode

The BCH codec should be enabled prior to using `Flash_IO_dma_control` to enable the DMA engine (Section 13.2.9.3). Refer to Section 13.2.6 below for the Flash I/O initialization steps.

1. Use the NAND Control register (`NAND_control`) to disable ECC-check and ECC-generation (Section 13.3.3.1).
2. Switch to dual-space mode by following the steps below.
 - A. Setup the main/spare jump size using the `Flash_IO_dsm_control` register (0x0A0) bits `main_jp_size` (bits 7:4) and `spare_jp_size` (bits 3:0). The Flash main-area byte count should be 512, and the spare-area byte count should be 16 or 32 depending on the BCH codec protected-range category (see Section 13.2.4).
 - B. Enable dual-space mode using the DMA data FIFO register `Flash_IO_dsm_control` (0x0A0) field `dsm_en` (bit 31).
 - C. Set the transaction count in the `Flash_IO_dma_control` register (0x080) as the sum of the main-area byte count and the spare-area byte count to be transferred.
3. Use the `Flash_IO_control` (0x000) register to setup the `ecc_bch_8t` and `ecc_bch_en` options and co-bits as appropriate.

The DMA FIFO Controller uses the ECC error-detected interrupt for read-disturb issues that occur in Flash chips using 6/8-bit ECC. When the BCH decoder detects an error, the NAND Flash block number is recorded and the status is set in the `Flash_IO_ecc_rpt_status` register before the interrupt is asserted. If there are errors detected in the same block on different pages, the interrupt will be asserted only once. Please note that the `Flash_IO_ecc_rpt_`

status register records only the last block number with a detected error; i.e., register content may be overwritten before software can read it.

If dual-space mode is enabled for the DMA FIFO Controller, it also must be enabled with the same settings for the FDMA engine.

13.2.6 Flash I/O: Subsystem Initialization

Before a Flash chip or an SD card can be accessed, the Flash I/O Subsystem registers must be initialized after reset. Initialization steps are as follows:

1. Exit random-read mode by writing to the Flash I/O Control register ([Flash_IO_control](#) in [Section 13.2.9.1](#))
2. Read the Flash I/O DMA Status register to retrieve the completion status of the automatic Data FIFO fill ([Flash_IO_dma_status](#) in [Section 13.2.9.5](#))
3. Clear the Flash I/O DMA Status register ([Flash_IO_dma_status](#))
4. Initialize the NAND Flash Controller ([Section 13.3.5](#))
5. Initialize the SD Controller ([Section 13.5](#))

13.2.7 Flash I/O: Activating Controllers

Controllers are activated by programming the Flash I/O Control register ([Section 13.2.9.1](#)). The software must ensure that there are no pending or ongoing transactions when switching between controllers.

13.2.8 Flash I/O: Register Map

The Flash I/O Subsystem contains registers that hold configuration information, control information, timing parameters, status, IDs, and commands. These registers communicate with the major blocks of the Flash I/O Subsystem including:

- Flash I/O Subsystem Global Registers
- 4-KB Data FIFO
- Flash Controller

The registers listed in the table below are memory-mapped into the AHB bus CPU slave address space of the Flash I/O Subsystem. They are accessed with single 4-byte AHB bus transactions. During an AHB bus transaction, AHB address[31:10] is the base address[31:10] of the Flash I/O Subsystem CPU slave address space, and AHB address[9:0] is the register index that indicates the register being accessed. The tables below list the Flash I/O Subsystem registers, their register indexes, access permissions, and brief descriptions.

Register Offset	Register Name	Description
0x000	Flash_IO_control	Flash I/O Control Flash I/O Subsystem global control information
0x004	Flash_IO_status	Flash I/O Status Flash I/O Subsystem command completion status
0x008 - 0x07C		Reserved
0x080	Flash_IO_dma_control	Flash I/O DMA Control Data FIFO DMA enable and control information
0x084	Flash_IO_dma_address	Flash I/O DMA Address Data FIFO DMA start address
0x08C	Flash_IO_dma_status	Flash I/O DMA Status Data FIFO DMA completion status
0x0A0	Flash_IO_dsm_control	Flash Dual-Space Mode Control
0x0A4	Flash_IO_ecc_rpt_status	Flash 6/8-bit ECC Error Detected Status

Table 13-4. Flash I/O Subsystem Global and DMA Registers.

Notes:

- For NAND Flash Controller registers see [Section 13.3.2 “NAND Flash: Register Map”](#)
- For SD Controller registers see [Section 13.5.6 “SD Controllers: Register Map”](#)

13.2.9 Flash I/O: Register Details

13.2.9.1 Flash_IO_control Register

The Flash I/O Control register (**Flash_IO_control**) holds Flash I/O global control information.

Bits	Field	Attr	Reset	Description
31:18				Reserved
17	da	RW	0	DMA acknowledge 1 - The previous Flash I/O Subsystem DMA request has been serviced, and the Flash I/O Subsystem can make another DMA request. This bit is intended to serve as a software DMA request/acknowledge mechanism, as an alternative to the hardware DMA request/acknowledge protocol using IO_DMA_REQ and IO_DMA_ACK hardware signals (IO_DMA_REQ and IO_DMA_ACK signals are not directly visible to the software).

Bits	Field	Attr	Reset	Description
16	dr	RW	0	DMA request 1 - The hardware DMA request signal, IO_DMA_REQ, is asserted. This bit is intended to serve as a software DMA request/acknowledge mechanism, as an alternative to the hardware DMA request/acknowledge protocol - IO_DMA_REQ and IO_DMA_ACK ports (IO_DMA_REQ and IO_DMA_ACK signals are not directly visible to the software).
15:9				Reserved
8	sx	R	0	SD present 0 - There is no SD card in the system 1 - An SD card is connected at the pins
7				Reserved
6	ecc_bch_en	RW	0	BCH encoding / decoding enable 0 - Disable 1 - Enable
5	ecc_bch_8t	RW	0	Select t parameter of BCH codec 0 - Set t = 6 1 - Set t = 8
4	rs	RW	0	Spare-area read enable 0 - The spare area will not be read. When performing a DMA transaction involving NAND Flash chips, this bit must be 1 if ECC correction is enabled, if DMA should stop upon encountering a read error, or if read errors are to be reported. The table below summarizes the settings for this bit. 1 - The spare area of NAND Flash chips will be read when performing a DMA transaction.
3	se	RW	0	Stop on error 0 - DMA will continue upon encountering a read error, and the OKAY response will always be returned on the AHB bus. Read errors are only detected for NAND Flash reads using their ECC. This bit must be 0 when the random read-mode bit is 1. The table below summarizes the settings for this bit. 1 - DMA will end early and an ERROR response will be returned on the AHB bus if a read error is encountered during DMA.
2	co	RW	0	ECC enable 0 - No correction will be performed. Read errors are only detected for NAND Flash reads using their ECC. This bit must be 0 when the random read-mode bit is 1. 1 - Read data will be corrected when a correctable read error occurs The table below summarizes the settings for this bit. Note that multiple-bit errors cannot be corrected.
1	rr	RW	1	Random read mode 0 - No Data FIFO random reads can be performed. When this bit is changed from 1 to 0, the Data FIFO is emptied. The stop-on-error bit and ECC correction-enable bit must be 0 when this bit is 1. 1 - Flash I/O Subsystem is in random read mode, and a location in the Data FIFO can be read by the host processor.
0				Reserved

Table 13-5. Flash I/O Control Register for Subsystem and Global Control Information.

Memory Target tg[1:0] Field Flash I/O DMA Control Register	ECC correction co Field Flash I/O Control Register	Stop On Error se Field Flash I/O Control Register	Spare-Area Read Enable rs Field Flash I/O Control Register
0 - NAND Flash	0 - No	0 - No	0 - Not enabled The NAND_control spare-area enable bit (se) must be 0 if the spare-area address bit (sa) is 0
0 - NAND Flash	0 - No	0 - No	1 - Enabled The NAND_control spare-area enable bit (se) must be 1
0 - NAND Flash	1 - Yes	0 - No	1 - Enabled The NAND_control spare-area enable bit (se) must be 1
0 - NAND Flash	0 - No	1 - Yes	1 - Enabled The NAND_control spare-area enable bit (se) must be 1
0 - NAND Flash	1 - Yes	1 - Yes	1 - Enabled The NAND_control spare-area enable bit (se) must be 1
1:3 - Reserved			

Table 13-6. The *co* and *se* Fields of *Flash_IO_control*.

13.2.9.2 Flash_IO_status Register

The Flash I/O Status register (**Flash_IO_status**) holds command-completion interrupt flags from the Flash and SD Controllers. If one of these interrupt flags is 1, the command-done interrupt is signaled to the VIC. Each of these interrupt flags is cleared by writing the appropriate register.

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	fi	R	0	Flash command-done interrupt 0 - No flag 1 - The Flash Controller has finished a Flash operation To clear this interrupt, write the NAND Flash Interrupt register (NAND_interrupt) to 0.

Table 13-7. Flash IO Subsystem Command Completion Status Register.

13.2.9.3 Flash_IO_dma_control Register

The Flash I/O DMA Control register (**Flash_IO_dma_control**) holds information that controls DMA operation in the Flash I/O Subsystem. The software writes to this register to set up the DMA operation and then enable the Flash I/O Subsystem to begin the DMA transaction.

Bits	Name	Attr	Reset	Description
31	en	RW	0	DMA enable 0 - There is no outstanding DMA operation. The software sets this bit to 1 to start the DMA operation. When the DMA operation is completed due to the successful transfer of the number of bytes shown in the count field or due to an error (if the stop-on-error bit se in the Flash I/O Control register Flash_IO_control is 1), this bit is automatically reset to 0. 1 - A DMA operation is in progress This bit can be written only when all DMA interrupts shown in the Flash I/O DMA Status register (Flash_IO_dma_status) have been cleared by writing the interrupt flags in Flash_IO_dma_status to 0.
30	rm	RW	0	Read memory 0 - DMA reads are from the memory target shown in the memory target field 1 - DMA reads are from main memory This bit is automatically set to 0 when the Flash I/O Subsystem is in random read mode.
29:28	tg	RW	1	Memory target DMA transactions are performed between the main memory and the memory target shown in this field according to the table below.
27:24	blk	RW	6	AHB bus transaction block size This field should be set to the number of bytes that will be in each AHB bus burst. See table below.
23:22	ts	RW	0	Transfer size Indicates the data size for each data transfer on the AHB bus. See table below.
21:0	count	RW	0	DMA transaction byte count Number of bytes transferred by the DMA operation if the DMA operation ends without an error. When an error occurs and the stop-on-error bit se in the Flash I/O Control register (Flash_IO_control) is 1, the DMA operation ends regardless of the number of bytes that have been transferred prior to the error.

Table 13-8. FLASH I/O Data FIFO DMA Enable and Control Register.

tg[1:0]	Memory Target	Comments
0	NAND Flash	
1-2		Reserved
3	SD Card	Will cause the Flash I/O subsystem to mux the internal FIFO for use by SD controller. Used for eMMC boot time only.

Table 13-9. The **tg** Field of **Flash_IO_dma_control**.

blk[3:0]	AHB bus Burst Size	blk[3:0]	AHB bus Burst Size	blk[3:0]	AHB bus Burst Size
0	8 bytes	5	256 bytes	9	4096 bytes

blk[3:0]	AHB bus Burst Size	blk[3:0]	AHB bus Burst Size	blk[3:0]	AHB bus Burst Size
1	16 bytes	6	512 bytes	10	8192 bytes
2	32 bytes	7	1024 bytes	11	16384 bytes
3	64 bytes	8	2048 bytes	12	32768 bytes
4	128 bytes				

Table 13-10. The *blk* Field of *Flash_IO_dma_control*.

ts[1:0]	Bus Transfer Size
0	Byte
1	Half word - 2 bytes
2	Word - 4 bytes
3	Double word - 8 bytes

Table 13-11. The *ts* Field of *Flash_IO_dma_control*.

13.2.9.4 Flash_IO_dma_address Register

The Flash I/O DMA Address register (*Flash_IO_dma_address*) holds the memory target start address for the DMA operation. The software writes to this register to set this start address before enabling DMA. This register is automatically updated during the DMA operation. This register can only be written when DMA is disabled; i.e., when the *Flash_IO_dma_control* enable bit (*en*) is 0.

Bits	Name	Attr	Reset	Description
31:0	address	RW	NR	Memory target start address for a DMA operation (must be double-word (8 byte) aligned)

Table 13-12. Flash IO DMA Address Register for Data FIFO Start Addresses.

13.2.9.5 Flash_IO_dma_status Register

The Flash I/O DMA Status register (*Flash_IO_dma_status*) captures completion status when a DMA operation ends. This includes the interrupt flags that communicate the reason the DMA operation ended, and the number of bytes transferred by the DMA operation. The software should write this register to 0 before starting each DMA operation. This register can only be written when DMA is disabled; i.e., the *Flash_IO_dma_control* enable bit (*en*) is 0.

Bits	Name	Attr	Reset	Description
31:27				Reserved
26	re	RW	0	Read error This bit is set to 1 when an error occurs during a NAND Flash read.
25	ae	RW	0	Address error This bit is set to 1 when the Flash IO DMA Address register (<i>Flash_IO_dma_address</i>) is not written with a double-word aligned address.

Bits	Name	Attr	Reset	Description
24	dn	RW	0	DMA operation done This bit is set to 1 when a DMA operation ends with the number of bytes shown in the Flash_IO_dma_control field count having been transferred. If a DMA operation ends due to an error before all data has been transferred, this bit is unchanged; i.e., 0 if this register is cleared before the DMA operation starts.
23:22				Reserved
21:0	count	RW	0	DMA transfer byte count Number of bytes transferred by the DMA operation. This field is automatically reset to 0 when a DMA operation begins.

Table 13-13. Flash I/O DMA Status Register for Data FIFO Completion Status.

13.2.9.6 Flash_IO_dsm_control Register

To minimize latency, the Flash I/O Dual-Space Mode Control register (**Flash_IO_dsm_control**) should be set equal to the register equivalent for the FDMA engine.

Bits	Name	Attr	Reset	Description
31	dsm_en	RW	0	Dual-Space Mode Enable 0 - Not enabled 1 - Enabled
30:8				Reserved
7:4	main_jp_size	RW	0x9	If dual-space mode is enabled, the DMA FIFO Controller will jump to transfer spare data when $2^{\text{main_jp_size}}$ bytes of main data is transferred. This value must be at least 9.
3:0	spare_jp_size	RW	0x4	If dual-space mode is enabled, the FDMA engine will jump to transfer main data when $2^{\text{spare_jp_size}}$ bytes of spare data is transferred. The legal value is from 3 to 9.

Table 13-14. Flash I/O Dual-Space Mode Control Register.

13.2.9.7 Flash_IO_ecc_rpt_status Register

The Flash I/O ECC Report Status register (**Flash_IO_ecc_rpt_status**).

Bits	Name	Attr	Reset	Description
31	ecc_rpt_err_det	RW	0	Flag indicates there is data error detected in Flash block number #err_rpt_blk by BCH decoder. 0 - No flag 1 - Error detected
30	ecc_rpt_corr_fail	RW	0	Flag indicates that the data error detected in Flash block number #err_rpt_blk is not correctable 0 - No flag 1 - Error not correctable
29:15				Reserved

Bits	Name	Attr	Reset	Description
14:0	ecc_rpt_blk	R	0	Flash block number that contains error data. Valid only when ecc_rpt_err_det is 1.

Table 13-15. Flash I/O ECC RPT Status Register.

13.3 Flash/SD: NAND Flash Controller

This section includes information regarding address space, programming and registers for the NAND Flash Controller as follows:

- [\(Section 13.3.1\) NAND Flash: Overview](#)
- [\(Section 13.3.2\) NAND Flash: Register Map](#)
- [\(Section 13.3.3\) NAND Flash: Register Details](#)
- [\(Section 13.3.4\) NAND Flash: Interrupts](#)
- [\(Section 13.3.5\) NAND Flash: Initialization](#)

- For Flash I/O Subsystem registers and programming see [Section 13.2](#).
- For SD Controller registers and programming see [Section 13.5](#).

13.3.1 NAND Flash: Overview

The A12 NAND Flash Controller is compatible with NAND Flash chips of various types and from a wide range of manufacturers. Features of the A12 Flash Controller include:

- Support for NAND Flash memory systems between 8 MB and 4 GB in size
- Read / Program / Erase, Read Status, Read ID, Copy Back, and Reset commands
- 8-bit Flash chip data bus
- 512-byte or 2048-byte page size
- ECC hardware support (1-bit errors only)
- Bose-Chaudhuri-Hocquenghem (BCH) error correction (6-bit or 8-bit)
- Configurable spare-area usage
 - Option for 2x spare area size
 - 2KB-page type with 128-byte spare area
 - 512KB-page type with 32-byte spare area
- Command confirmation
- 2-, 4- or 5-cycle ID register read (configurable)
- Interrupt or ARM core polling for Flash command completion

- Optional burst mode
- Software-programmable timing parameters to accommodate a wide range of Flash speed grades and system clock frequencies.

The ARM core initiates Flash chip operations by writing commands to the Flash Controller. When a Flash chip operation completes, the Flash Controller optionally signals a maskable interrupt to the ARM core. The ARM core may also poll Flash Controller registers to determine when a Flash chip operation has completed. Refer to [Section 13.3.4](#) for further information on Flash interrupts. Refer to [Section 13.3.5](#) for details on NAND Flash Controller initialization.

The NAND Flash Controller provides registers to configure the main and spare area used by the DMA FIFO Controller. Included is an additional option to double the spare area available to FDMA. The 2x spare-area feature is enabled using the **NAND_extended_control** register **flash_ctl_sp_2x** bit ([Section 13.3.3.17](#)). Refer to [Section 13.2.3](#) in this chapter for details on Flash I/O dual-space mode and its use with BCH error correction. Refer to [Section 13.3.1](#) for related programming of the Flash DMA engine (FDMA).

The A12 chip supports 2-, 4-, or 5-cycle ID register reads. The 2- and 4-cycle reads are selected using the **NAND_control** register **i4** bit. Use the **NAND_extended_control** register **i5** bit to implement a fifth data byte read and the **NAND_extended_id** register field **flash_ext_id** to store it ([Section 13.3.3.18](#)).

The NAND Flash controller includes an optional burst mode setting. Burst mode can be enabled for the spare area by configuring **NAND_control** register spare-area address (**sa**) and spare-area enable (**se**) bits ([Section 13.3.3.1](#))

13.3.2 NAND Flash: Register Map

- The Flash Controller contains registers that hold NAND Flash system configuration information, NAND Flash control information, NAND Flash timing parameters, NAND Flash chip status, NAND Flash chip ID, NAND Flash controller commands, and the NAND Flash interrupt status.
- These registers are active when the Flash Controller is in NAND Flash mode.
- These registers must be initialized each time the Flash Controller enters NAND Flash mode.
- These registers are mapped to the AHB bus and are accessible only when the Flash Controller is configured to NAND Flash mode.

Register Offset	Register Name	Description
0x100 - 0x11C		Reserved
0x120	NAND_control	Flash Controller control information NAND Flash mode
0x124	NAND_command	Flash Controller command and associated address NAND Flash mode
0x128	NAND_timing_int0	NAND Flash chip timing parameters NAND Flash mode
0x12C	NAND_timing_int1	NAND Flash chip timing parameters NAND Flash mode

Register Offset	Register Name	Description
0x130	NAND_timing_int2	NAND Flash chip timing parameters NAND Flash mode
0x134	NAND_timing_int3	NAND Flash chip timing parameters NAND Flash mode
0x138	NAND_timing_int4	NAND Flash chip timing parameters NAND Flash mode
0x13C	NAND_timing_int5	NAND Flash chip timing parameters NAND Flash mode
0x140	NAND_status	NAND Flash chip status NAND Flash mode
0x144	NAND_id	NAND Flash device ID information NAND Flash mode
0x148	NAND_copy_address	2nd address for NAND Flash copy operations NAND Flash mode
0x14C	NAND_length	NAND Flash command and associated length NAND Flash mode
0x150	NAND_interrupt	NAND Flash command-done interrupt NAND Flash mode
0x154	NAND_read_command_word	NAND Flash command word for read command NAND Flash mode
0x158	NAND_program_command_word	NAND Flash command word for program command NAND Flash mode
0x15C	NAND_extended_control	NAND Flash controller extended control information
0x160	NAND_extended_id	NAND Flash device extended ID information NAND Flash Mode

Table 13-16. NAND Flash Controller Registers.

13.3.3 NAND Flash: Register Details

13.3.3.1 NAND_control Register

The NAND Flash Control register (**NAND_control**) contains Flash chip control and configuration information.

Bits	Name	Attr	Reset	Description
31:30				Reserved
29:28	a	RW	0	Address[33:32] These address bits are concatenated with the 32-bit address of each command to form the 34-bit address that is used in systems with more than 4 GB of Flash memory. Bits[1:0] correspond with bits[33:32] of the 34-bit address.
27	sa	RW	0	Spare area address 0 - The address is a main-area address and the column address is a location in the main area. 1 - The address is a spare-area address and the column portion of the address refers to a location within the spare area
26	se	RW	0	Spare area enable 0 - The spare area cannot be accessed 1 - Enables access to the spare area. This bit is set to 1 to access the spare area after the main area is accessed when performing page reads. This bit must also be set to 1 when the spare-area address bit is 1.
25	c2	RW	NR	Column address in two cycles 0 - Only one column address cycle is required (512-byte pages) 1 - Two column address cycles are required (2048-byte pages)
24	p3	RW	1	Page address in three cycles 0 - Only two page address cycles are required 1 - Three page address cycles are required
23	i4	RW	0	ID register read in four cycles 0 - Two read cycles are required to read the Flash chip ID register 1 - Four read cycles are required to read the Flash chip ID register
22	rc	RW	1	Read confirm 0 - No read-confirm command is required 1 - Flash chips require a read-confirm command after the address input of the read command
21	cc	RW	0	Copy confirm Writing this bit to 1 indicates that the Flash chips require a copy-confirm command after the address input of the copy command. If no copy-confirm command is required, this bit should be set to 0.
20	ce	RW	0	Copy enable 0 - No copy-back command will be sent to the Flash chips 1 - Enables use of the Flash chip-copy back command
19:18	ec	RW	0	ECC check enable This field indicates if ECC checking-on-reads is enabled for the main-area and spare-area Logical Sector Number (LSN).
17:16	eg	RW	0	ECC generation enable This field indicates if ECC generation on programs is enabled for the main-area and spare-area Logical Sector Number (LSN).
15:11			0	Reserved

Bits	Name	Attr	Reset	Description
10	was	RW	0	Write with automatic status update When enabled, the status register will contain an OR'ed result of all NAND Flash transactions. This is used to detect errors that occur during a long transaction sequence with numerous write commands.
9	wp	RW	1	Write protect pin 0 - Flash can be erased or programmed 1 - Write protection is enabled
8	ie	RW	1	Interrupt enable 0 - FLASH_INT will not be asserted when a command completes 1 - Enables assertion of the FLASH_INT output port when a Flash command-done interrupt occurs.
7	xs	RW	0	16-bit Flash chip data bus 0 - Flash chips have an eight-bit data bus 1 - Flash chips have a sixteen-bit data bus
6:4	sz	RW	0	Flash chip size This field indicates the size of the Flash chips.
3:2	eb	RW	0	External banks This field indicates the number of Flash banks in the system.
1:0	wd	RW	0	Data bus width This field indicates the data bus width of the Flash controller.

Table 13-17. NAND Flash Control Register.

ec[1:0]	Main Area ECC Check Enabled	Spare Area LSN ECC Check Enabled
0	No	No
1	No	Yes
2	Yes	No
3	Yes	Yes

Table 13-18. NAND Flash Control Register *ec* Field.

eg[1:0]	Main Area ECC Generate Enabled	Spare Area LSN ECC Generate Enabled
0	No	No
1	No	Yes
2	Yes	No
3	Yes	Yes

Table 13-19. NAND Flash Control Register *eg* Field.

sz[2:0]	Flash Chip Size (Not Including Spare Area)	Page Size
0	64 Mbits	2 KByte / 512 Bytes
1	128 Mbits	2 KByte / 512 Bytes
2	256 Mbits	2 KByte / 512 Bytes
3	512 Mbits	2 KByte / 512 Bytes
4	1 Gbits	2 KByte / 512 Bytes

sz[2:0]	Flash Chip Size (Not Including Spare Area)	Page Size
5	2 Gbits	2 KByte / 512 Bytes
6	4 Gbits	2 KByte / 512 Bytes
7	8 Gbits	2 KByte / 512 Bytes

Table 13-20. NAND Flash Control Register, sz[2:0] Field.

eb[1:0]	Number of Flash Banks
0	1
1	2
2	4
3	Reserved

Table 13-21. NAND Flash Control Register eb Field.

wd[1:0]	Data Bus Width	xs	Number of Flash Chips/Bank
0	8 bits	Must be 0	1
1	16 bits	0	2
1	16 bits	1	1
2	32 bits	0	4
2	32 bits	1	2
3	64 bits	Must be 1	4

Table 13-22. NAND Flash Control Register wd Field.

13.3.3.2 NAND_command Register

Bits	Name	Attr	Reset	Description
31:4	address	RW	NR	Address[31:4] For the following active commands: Erase, Read, Read Status, Read ID, Program, and Copy Back. This field is not used by other commands. The table below shows the command-specific address information in this field.
3:0	cmd	RW	2	Command Indicates the Flash chip operation in progress. The table below shows the commands and encodings for each command, and identifies the commands that the ARM core can write to this field.

Table 13-23. NAND Flash Command Register.

The NAND Flash Command register (**NAND_command**) holds the active command being executed by the Flash chips and its associated address. The ARM core writes this register to specify the next Flash chip operation. The ARM core reads this register to determine the command in progress.

cmd[3:0]	Can be Written by ARM Core	Command	Address[27:0] Written by ARM Core	Address[27:0] Read by Host Processor
0x0	Yes	NOP	0	Undefined
0x1	Yes	DMA	0	Undefined
0x2	Yes	Reset	0	Undefined
0x3	No	NOP2	Not Applicable	Undefined
0x4	No	NOP3	Not Applicable	Undefined
0x5	No	NOP4	Not Applicable	Undefined
0x6	No	NOP5	Not Applicable	Undefined
0x7	Yes if NAND_control copy enable (ce) = 1	Copy Back	Start address[31:4] of the data to be copied	Before Copy Back command is sent to Flash chips using start address[31:4] of the data to be copied After Copy Back command is sent to Flash chips using start address[31:4] of the copy of the data
0x8	No	NOP6	Not Applicable	Undefined
0x9	Yes	Erase	Address[31:4] of the block to be erased	Address[31:4] of the block being erased
0xA	Yes	Read ID	Address[31:4] of the bank to be read	Address[31:4] of the bank being read
0xB	No	NOP7	Not Applicable	Undefined
0xC	Yes	Read Status	Address[31:4] of the bank to be read	Address[31:4] of the bank being read
0xD	No	NOP8	Not Applicable	Undefined
0xE	No - Read will become active after the processor writes the DMA command and a subsequent read request arrives at the Flash Controller requester interface	Read	Not Applicable	Address[31:4] of the location currently being read. The read start address is indicated by the mem_flash_addr[31:0] input port.
0xF	No - Program will become active after the processor writes the DMA command and a subsequent write request arrives at the Flash Controller requester interface	Program	Not Applicable	Address[31:4] of the location currently being written. The write start address is indicated by the mem_flash_addr[31:0] input port.

Table 13-24. NAND Flash Command Encodings.

13.3.3.3 NAND_timing_int0 Register

The NAND Flash Timing Interval 0 register (**NAND_timing_int0**) holds cycle counts for Flash timing parameters **tcls**, **tals**, **tcs**, and **tds**.

Bits	Name	Attr	Reset	Description
31:24	tcls	RW	0x20	tCLS count tcls (command setup to write-pulse rising edge time) expressed as a number of clock cycles. For example, if tcls is 0 ns and the clock rate is 100 MHz, the value for this field should be set to 1. The value written to this field should be rounded up for fractional cycle counts.
23:16	tals	RW	0x20	tALS count tals (address setup to write-pulse rising edge time) expressed as a number of clock cycles. For example, if tals is 0 ns and the clock rate is 100 MHz, the value for this field should be set to 1. The value written to this field should be rounded up for fractional cycle counts.
15:8	tcs	RW	0x20	tCS count tcs (chip enable setup to write-pulse rising edge time) expressed as a number of clock cycles. For example, if tcs is 0 ns and the clock rate is 100 MHz, the value for this field should be set to 1. The value written to this field should be rounded up for fractional cycle counts.
7:0	tds	RW	0x20	tDS count tds (data setup to write-pulse rising edge time) expressed as a number of clock cycles. For example, if tds is 20 ns and the clock rate is 100 MHz, the value for this field should be set to 2. The value written to this field should be rounded up for fractional cycle counts.

Table 13-25. NAND Flash Timing Interval 0 Register.

13.3.3.4 NAND_timing_int1 Register

The NAND Flash Timing Interval 1 register (**NAND_timing_int1**) holds cycle counts for Flash timing parameters **tclh**, **talh**, **tch**, and **tdh**.

Bits	Name	Attr	Reset	Description
31:24	tclh	RW	0x20	tCLH count tclh (command hold from write-pulse rising edge time) expressed as a number of clock cycles. For example, if tclh is 10 ns and the clock rate is 100 MHz, the value for this field should be set to 1. The value written to this field should be rounded up for fractional cycle counts.
23:16	talh	RW	0x20	tALH count talh (address hold from write-pulse rising edge time) expressed as a number of clock cycles. For example, if talh is 10 ns and the clock rate is 100 MHz, the value for this field should be set to 1. The value written to this field should be rounded up for fractional cycle counts.
15:8	tch	RW	0x20	tCH count tch (chip-enable hold from write-pulse rising edge time) expressed as a number of clock cycles. For example, if tch is 10 ns and the clock rate is 100 MHz, the value for this field should be set to 1. The value written to this field should be rounded up for fractional cycle counts.

Bits	Name	Attr	Reset	Description
7:0	tdh	RW	0x20	tDH count tdh (data hold from write-pulse rising edge time) expressed as a number of clock cycles. For example, if tdh is 10 ns and the clock rate is 100 MHz, the value for this field should be set to 1. The value written to this field should be rounded up for fractional cycle counts.

Table 13-26. NAND Flash Timing Interval 1 Register.

13.3.3.5 NAND_timing_int2 Register

The NAND Flash Timing Interval 2 register (**NAND_timing_int2**) holds cycle counts for Flash timing parameters **twp**, **twh**, **twb**, and **trr**.

Bits	Name	Attr	Reset	Description
31:24	twp	RW	0x20	tWP count twp (write-pulse width) expressed as a number of clock cycles. For example, if twp is 25 ns and the clock rate is 100 MHz, the value for this field should be set to 3. The value written to this field should be rounded up for fractional cycle counts.
23:16	twh	RW	0x20	tWH count twh (write high hold time) expressed as a number of clock cycles. For example, if twh is 15 ns and the clock rate is 100 MHz, the value for this field should be set to 2. The value written to this field should be rounded up for fractional cycle counts.
15:8	twb	RW	0x20	tWB count twb (write-pulse rising edge to Busy time) expressed as a number of clock cycles. For example, if twb is 100 ns and the clock rate is 100 MHz, the value for this field should be set to 10. The value written to this field should be rounded up for fractional cycle counts.
7:0	trr	RW	0x20	tRR count trr (ready setup to read-pulse falling edge time) expressed as a number of clock cycles. For example, if trr is 20 ns and the clock rate is 100 MHz, the value for this field should be set to 2. The value written to this field should be rounded up for fractional cycle counts.

Table 13-27. NAND Flash Timing Interval 2 Register.

13.3.3.6 NAND_timing_int3 Register

The NAND Flash Timing Interval 3 register (**NAND_timing_int3**) holds cycle counts for Flash timing parameters **trp**, **treh**, **trb**, and **tceh**.

Bits	Name	Attr	Reset	Description
31:24	trp	RW	0x20	tRP count trp (read-pulse width) expressed as a number of clock cycles. For example, if trp is 25 ns and the clock rate is 100 MHz, the value for this field should be set to 3. The value written to this field should be rounded up for fractional cycle counts.

Bits	Name	Attr	Reset	Description
23:16	treh	RW	0x20	tREH count treh (read high hold time) expressed as a number of clock cycles. For example, if treh is 15 ns and the clock rate is 100 MHz, the value for this field should be set to 2. The value written to this field should be rounded up for fractional cycle counts.
15:8	trb	RW	0x20	tRB count trb (read-pulse rising edge to Busy time) expressed as a number of clock cycles. For example, if trb is 100 ns and the clock rate is 100 MHz, the value for this field should be set to 10. The value written to this field should be rounded up for fractional cycle counts.
7:0	tceh	RW	0x20	tCEH count tceh (chip enable hold from read-pulse rising edge time) expressed as a number of clock cycles. For example, if tceh is 100 ns and the clock rate is 100 MHz, the value for this field should be set to 10. The value written to this field should be rounded up for fractional cycle counts.

Table 13-28. NAND Flash Timing Interval 3 Register.

13.3.3.7 NAND_timing_int4 Register

The NAND Flash Timing Interval 4 register (**NAND_timing_int4**) holds cycle counts for Flash timing parameters **trdelay**, **tclr**, **twhr**, and **tir**.

Bits	Name	Attr	Reset	Description
31:24	trdelay	RW	0x20	tRDELAY count trdelay (read-pulse falling edge to data valid time including all C to Q delays, transit time delays, buffer delays, etc.) expressed as a number of clock cycles. For example, if trdelay is 35 ns and the clock rate is 100 MHz, the value for this field should be set to 4. The value written to this field should be rounded up for fractional cycle counts. trdelay must be the value between trp and min((trp + treh +1), trhz + trp)
23:16	tclr	RW	0x20	tCLR count tclr (command setup to read-pulse falling edge time) expressed as a number of clock cycles. For example, if tclr is 10 ns and the clock rate is 100 MHz, the value for this field should be set to 1. The value written to this field should be rounded up for fractional cycle counts.
15:8	twhr	RW	0x20	tWHR count twhr (write-pulse rising edge to read-pulse falling edge time) expressed as a number of clock cycles. For example, if twhr is 60 ns and the clock rate is 100 MHz, the value for this field should be set to 6. The value written to this field should be rounded up for fractional cycle counts.

Bits	Name	Attr	Reset	Description
7:0	tir	RW	0x20	tIR count tir (Flash chip output high Z to read-pulse falling edge time) expressed as a number of clock cycles. For example, if tir is 0 ns and the clock rate is 100 MHz, the value for this field should be set to 1. The value written to this field should be rounded up for fractional cycle counts.

Table 13-29. NAND Flash Timing Interval 4 Register.

13.3.3.8 NAND_Timing_int5 Register

The NAND Flash Timing Interval 5 register (**NAND_timing_int5**) holds cycle counts for Flash timing parameters **tww**, **trhz**, and **tar**.

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	tww	RW	0x20	tWW count tww (ready to write-pulse falling edge time) expressed as a number of clock cycles. For example, if tww is 25 ns and the clock rate is 100 MHz, the value for this field should be set to 3. The value written to this field should be rounded up for fractional cycle counts.
15:8	trhz	RW	0x20	trHZ count trhz (read-pulse rising edge to Flash chip output high Z time) expressed as a number of clock cycles. For example, if trhz is 30 ns and the clock rate is 100 MHz, the value for this field should be set to 3. The value written to this field should be rounded up for fractional cycle counts.
7:0	tar	RW	0x20	TAR count tar (address setup to read-pulse falling edge time) expressed as a number of clock cycles. For example, if tar is 10 ns and the clock rate is 100 MHz, the value for this field should be set to 1. The value written to this field should be rounded up for fractional cycle counts.

Table 13-30. NAND Flash Timing Interval 5 Register.

13.3.3.9 NAND_status Register

The NAND Flash Status register (**NAND_status**) holds the status information retrieved from the Flash chips by the last Read Status command. When there are multiple Flash chips per bank, the Read Status command is issued simultaneously to all Flash chips in the bank. The status information received from each Flash chip has a designated field in **NAND_status**.

Bits	Name	Attr	Reset	Description
31:24	stat3	R	NR	Status 3 Status from the fourth least significant Flash chip on the Flash Controller data bus

Bits	Name	Attr	Reset	Description
23:16	stat2	R	NR	Status 2 Status from the third least significant Flash chip on the Flash Controller data bus
15:8	stat1	R	NR	Status 1 Status from the second least significant Flash chip on the Flash Controller data bus
7:0	stat0	R	NR	Status 0 Status from the Flash chip connected to the least significant Flash Controller data bus wires

Table 13-31. NAND Flash Status Register.

13.3.3.10 NAND_id Register

The NAND Flash ID register (**NAND_id**) holds the ID information retrieved from the Flash chips by the last Read ID command. When there are multiple Flash chips per bank, it is assumed that these chips are of the same type. The ID information retrieved from the least significant Flash chip of the bank is saved in **NAND_id**.

Bits	Name	Attr	Reset	Description
31:24	mkr	R	0	Maker Flash chip manufacturer code
23:16	dev	R	0	Device Flash chip device code
15:8	id3	R	0	ID3 Information from 3rd ID read cycle
7:0	id4	R	0	ID4 Information from 4th ID read cycle

Table 13-32. NAND Flash ID Register.

13.3.3.11 NAND_copy_address Register

The NAND Flash Copy Address register (**NAND_copy_address**) holds the destination start address for the Copy Back command.

Bits	Name	Attr	Reset	Description
31:0	address	RW	0	Address Start address where the Copy Back command data is to be copied

Table 13-33. NAND Flash Copy Address Register.

13.3.3.12 NAND_length Register

The NAND Flash Length register (**NAND_length**) holds the active command being executed by the Flash chips and a byte count of data remaining to be transferred.

Bits	Name	Attr	Reset	Description
31				Reserved
30:16	length	R	0x7FFF	Length[14:0] Remaining byte count minus one of the data to be transferred for the active commands Read and Program. This field is not used by other commands.
15:4				Reserved
3:0	cmd	R	0x2	Command[3:0] Indicates the Flash chip operation in progress. Refer to the NAND Flash Command register (NAND_command) descriptions for encodings.

Table 13-34. NAND Flash Length Register.

13.3.3.13 NAND_interrupt Register

The NAND Flash Interrupt register (**NAND_interrupt**) holds the command-done interrupt flag. Refer to [Section 13.3.4](#) for related detail.

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	di	R	0	Command-done interrupt This bit is automatically set to 1 when a Flash operation completes. If the interrupt-enable (ie) bit in the NAND Flash Control register (NAND_control) is 1, the FLASH_INT output is asserted when the command-done interrupt bit is 1. To deassert the Flash interrupt, the software writes this bit to 0.

Table 13-35. NAND Flash Interrupt Register.

13.3.3.14 NAND_read_command_word Register

The NAND Flash Read Command Word register (**NAND_read_command_word**) sets first and second command words for read commands in NAND Flash mode.

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	flash_read_cmdwd_1	R/W	0x00	First command word of read operation When executing a read operation, the command word of the first CLE signal can be programmed by this Register.
15:8				Reserved
7:0	flash_read_cmdwd_2	R	0x30	Second command word of read operation When executing a read operation, the command word of the second CLE signal can be programmed by this Register.

Table 13-36. NAND Flash Read Command Word Register.

13.3.3.15 NAND_program_command_word Register

The NAND Flash Program Command Word register (**NAND_program_command_word**) sets first and second command words to program commands in NAND Flash mode.

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	flash_read_cmdwd_1	R/W	0x00	First command word of program operation When executing a program operation, the command word of the first CLE signal can be programmed by this Register.
15:8				Reserved
7:0	flash_read_cmdwd_2	R	0x30	Second command word of program operation When executing a program operation, the command word of the second CLE signal can be programmed by this Register.

Table 13-37. NAND Flash Porgram Command Word Register.

13.3.3.16 NAND_program_command_word Register

The NAND Flash Program Command Word register (**NAND_program_command_word**) sets first and second command words to program commands in NAND Flash mode.

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	flash_read_cmdwd_1	R/W	0x80	First command word of program operation When executing a program operation, the command word of the first CLE signal can be programmed by this Register.
15:8				Reserved
7:0	flash_read_cmdwd_2	R	0x10	Second command word of program operation When executing a program operation, the command word of the second CLE signal can be programmed by this Register.

Table 13-38. NAND Flash Program Command Word Register.

13.3.3.17 NAND_extended_control Register

The NAND Flash Extended Control register (**NAND_extended_control**) sets timing for ID register reads and enables the spare area to facilitate memory transfer. Refer to [Section 4.4](#) for further information regarding the use of the spare area with Flash I/O DMA (FDMA) transactions.

Bits	Name	Attr	Reset	Description
31:24				Reserved
23	i5	RW	0x00	ID register read in five cycles 0 - When this bit is 0, two or four read cycles are required to read the Flash chip ID Register. 1 - Writing this bit to 1 indicates that five read cycles are required to read the Flash chip ID Register.
22:1				Reserved

Bits	Name	Attr	Reset	Description
0	flash_ctl_sp_2x	RW	0x00	2x spare area enable 0 - Default spare area is 16 bytes for 512-byte page size and 64 bytes for 2-Kbyte page size. 1 - Writing this bit to 1 increases the spare area to 32 bytes for 512-byte page size and 128 bytes for 2-Kbyte page size.

Table 13-39. NAND Flash Extended Control Register.

13.3.3.18 NAND_extended_id Register

The NAND Flash Extended ID register (**NAND_extended_id**) stores the fifth-byte data read when **NAND_extended_control** bit 23 is set to 1.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	flash_ext_id	RW	0x00	Byte 4 of the ID register

Table 13-40. NAND Extended ID Register.

13.3.4 NAND Flash: Interrupts

The Flash Controller sets the command-done interrupt bit in a NAND Interrupt register to 1 when a Flash operation completes. Software may also write this interrupt bit to 1. The Flash Controller signals a command-done interrupt if the interrupt enable bit is 1. A command-done interrupt remains active until the software has cleared it by writing 0 to the command-done interrupt bit.

Note that the NAND Flash controller maintains interrupt lines directly to the vector interrupt controller (VIC).

13.3.5 NAND Flash: Initialization

The NAND Flash Controller registers must be initialized before Flash chips can be accessed. The Controller registers must be initialized:

- After reset
- Each time the NAND Flash mode changes

The initialization process involves entering the appropriate values in the Flash Controller configuration registers. Initialization depends on the mode; however, an example initialization process for NAND is provided as follows:

- Before accessing the Flash chip, initialize:
 - NAND Flash Control register (**NAND_control**)
 - NAND Flash Timer registers (**NAND_timing_int[n]**)
 - NAND Flash Interrupt register (**NAND_interrupt**)

These registers can be initialized in two ways:

1. Hard-wired reset values can be loaded into these registers
2. The software can write to these registers when reset is deasserted

Notes:

- Hard-wired reset values represent worst-case values which will allow the controller to communicate with the external Flash memory in which the system boot-code resides.
- After boot-up, the system software must program the timing values particular to the specific memory devices in use to ensure optimal I/O performance.
- To determine configuration values for these registers, refer to the relevant Flash chip datasheet provided by the chip manufacturer

For Confidential
For HAOTEX Only

13.4 Flash/SD: SD PHY

13.4.1 SD PHY: Register Map

The SD PHY block is controlled by the Reset, Clock and Test (RCT) debug unit.

SD PHY register descriptions are provided below.

Register Offset	Field	Bits	Type	Reset	Description
0x4C0	Reserved	31:27	-	-	Reserved
	sd_clkout_bypass	26	RW	1	Bypass clk_sdcard_out 1: Bypass clk_sdcard_out 0: Delay the phase of clk_sdcard_out
	sd_RST	25	RW	0	Active-high reset of SD PHY
	Reserved	24:20	-	-	Reserved
	sd_rx_clk_pol	19	RW	0	Change SD clock polarity
	sd_data_cmd_bypass	18	RW	1	Bypass input SD command and data
	sd_DLL_bypass	17	RW	1	Bypass DLL
	Reserved	16	-	-	Reserved
	sd_sbc	15	RW	0	Power-down shift register when locked
		14:3	-	-	Reserved
		2:1	RW	0	Adjust coarse delay as shown below 00 0 ns 01 1 ns 10 2 ns 11 3 ns
		0	RW	0	Enable DLL
0x4C4	sd_SEL3	31:24	-	-	Reserved
	sd_SEL2	23:16	RW	0	Fine tune DLL delay
	sd_SEL1	15:8	RW	0	Fine tune DLL delay
	sd_SEL0	7:0	RW	0	Fine tune DLL delay
0x4F0	Reserved	31:8	-	-	Reserved
	sd_RCT_vshift	7:0	R		Vshift value

Table 13-41. SD PHY Registers.

13.4.2 SD PHY: Programming

13.4.2.1 SD PHY Programming: Bypass SD PHY Block

To bypass the entire SD PHY block, set the register fields as shown below.

- `sd_clkout_bypass` = 1
- `sd_dll_bypass` = 1
- `sd_data_cmd_bypass` = 1

13.4.2.2 SD PHY Programming: DLL

To use the DLL, follow the steps below.

- Enable the SD clock by setting bit 2 of the Clock Control Register (offset 0x2C) in the SD (host) controller.
- Set `sbc[0]` = 1 to enable the DLL.
- Assert `sd_rst` high longer than 1 us for normal operation.
- When the DLL is locked, `Vshift[7:0]` will be a stable value.
- When `sel[5]` is 0, incrementing `sel[4:0]` by 1 will increase delay by one step.
- When `sel[5]` is 1, incrementing `sel[4:0]` by 1 will decrease delay by one step.

13.4.2.3 SD PHY Programming: Additional Notes

- Set `sd_data_cmd_bypass`=1 when the SD clock is off. This allows some PME signals to be bypassed.
- Set `sbc[15]` to 1 such that `Vshift` will be stable once the DLL is in a locked state.
- The SD PHY allows users to invert the clock polarity by setting `rx_clk_pol` = 1. Note that when this function is enabled with `sd_clkout_bypass`=0, the polarity of the outgoing clock will also be inverted. e.g., If the idle state of `clk_in` is 0, then the idle state of `clk_sdcard_out` will become 1.

13.5 Flash/SD: SD Controllers

This section provides information regarding the A12 SD controllers including:

- [\(Section 13.5.1\) SD Controllers: Overview](#)
 - [\(Section 13.5.2\) SD Controllers: Base Address Information](#)
 - [\(Section 13.5.3\) SD Controllers: eMMC Boot](#)
 - [\(Section 13.5.4\) SD Controllers: Clocking](#)
 - [\(Section 13.5.5\) SD Controllers: SDMA and ADMA Operation](#)
 - [\(Section 13.5.6\) SD Controllers: Register Map](#)
 - [\(Section 13.5.7\) SD Controllers: Register Details](#)
 - [\(Section 13.5.8\) SD Controllers: ADMA Block Diagram and Descriptor Programming](#)
-
- The reader is referred to the A12 chip datasheet for interface pin and power-rail details, including multiplexing information.
 - See [Section 2.3](#) of this manual for eMMC power-on configuration information.
- i** Note that both the total number and applicable specifications of A12(m) SD controllers are chip-dependent.
Please consult the relevant A12 or A12m datasheet for chip-specific detail.

13.5.1 SD Controllers: Overview

The A12 chip includes up to three SD Controllers for SD / SDIO / SDHC / SDXC / MMC / eMMC operations. These controllers conform to the specifications outlined in “*SD Specifications Part 1 Physical Layer Specification Version 3.00*” (SD Spec).

For information regarding specific SD controller features, please refer to the relevant A12 chip datasheet.

13.5.2 SD Controllers: Base Address Information

The SD controllers are located at the AHB base addresses shown below.

- SD0: 0xE000.2000
- SD1: 0xE000.C000
- SD2: 0xE001.F000

13.5.3 SD Controllers: eMMC Boot

Refer to [Section 2.3.7](#) for eMMC boot mode details.

13.5.4 SD Controllers: Clocking

The A12 chip configures the SD0 controller clock (gclk_sd48), the SD1 controller clock (gclk_sdio), and the SD2 controller clock (gclk_sdxc) using a set of APIs. The register programming described below is used for adjustment and refinement. Contact an Ambarella representative for additional detail.

13.5.5 SD Controllers: SDMA and ADMA Operation

The A12 SD controllers maintain an independent bus mastering DMA engine and do not rely on the AHB DMA Engine for DMA transfers. The SD controllers support the single DMA (SDMA) transfers used with previous Ambarella chips and also include an advanced DMA (ADMA) algorithm for high transfer speeds.

For Confidential
For HAOTEX Only

13.5.5.1 SD Controllers: SDMA Operation

SDMA transfer operations involve the use of DMA Interrupts at every page boundary. These interrupts trigger the CPU to reprogram a new system address to the page. While this process is generally effective, it places a limitation on transfer speed.

The command flow for SDMA transfer operations is illustrated in the figure below.

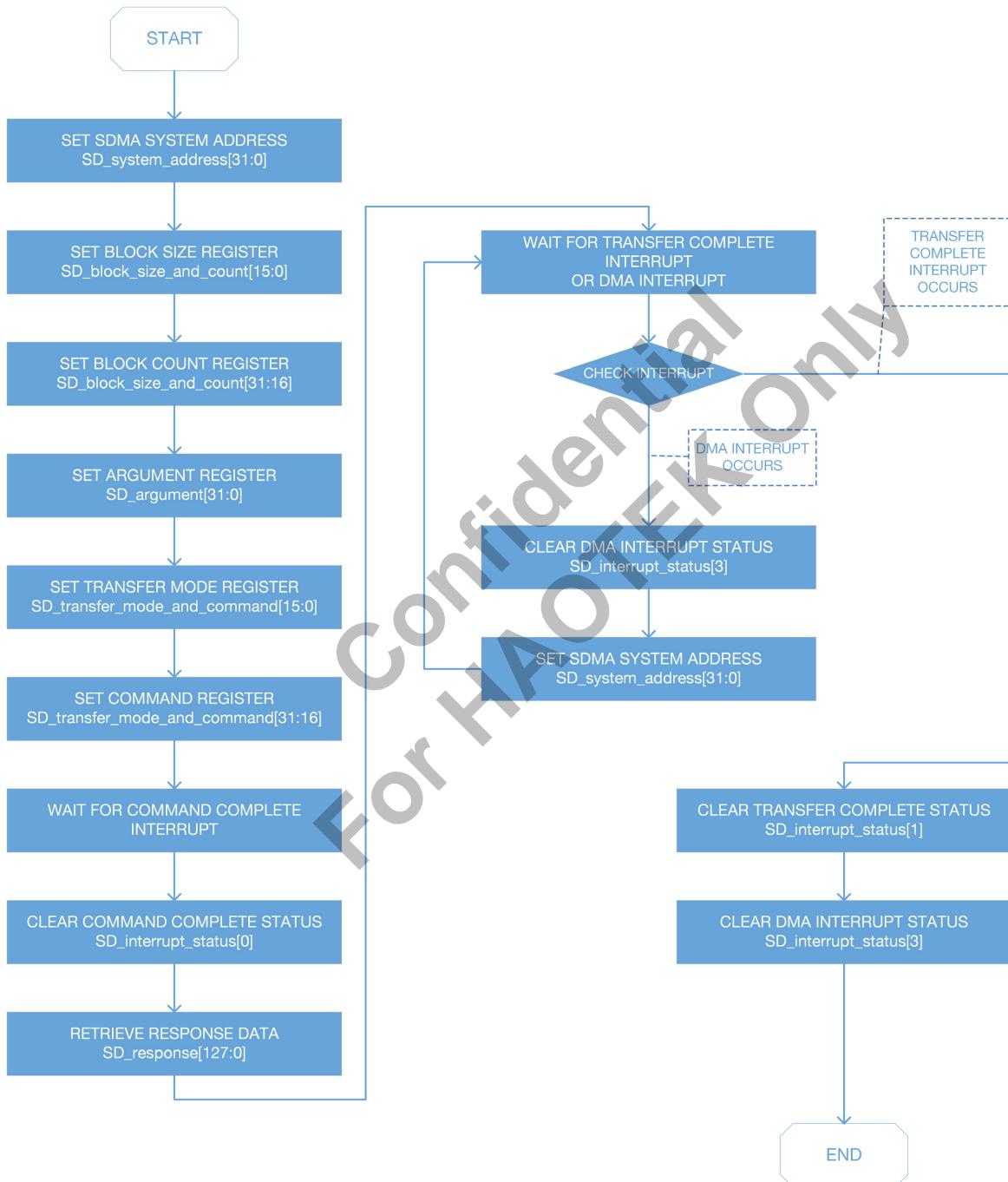


Figure 13-8. SD Controllers: SDMA Command Flow.

13.5.5.2 SD Controllers: ADMA Operation

The ADMA engine enables higher data-transfer speeds by using a scatter-gather DMA algorithm. Using this algorithm, the Host Driver programs a Descriptor Table with a list of pending data transfers between system memory and the SD card. This enables the ADMA engine to operate without interrupting the Host Driver. Refer to [Section 13.5.8](#) for an ADMA block diagram and Descriptor Table programming detail.

The command flow for ADMA transfer operations is illustrated in the figure below.

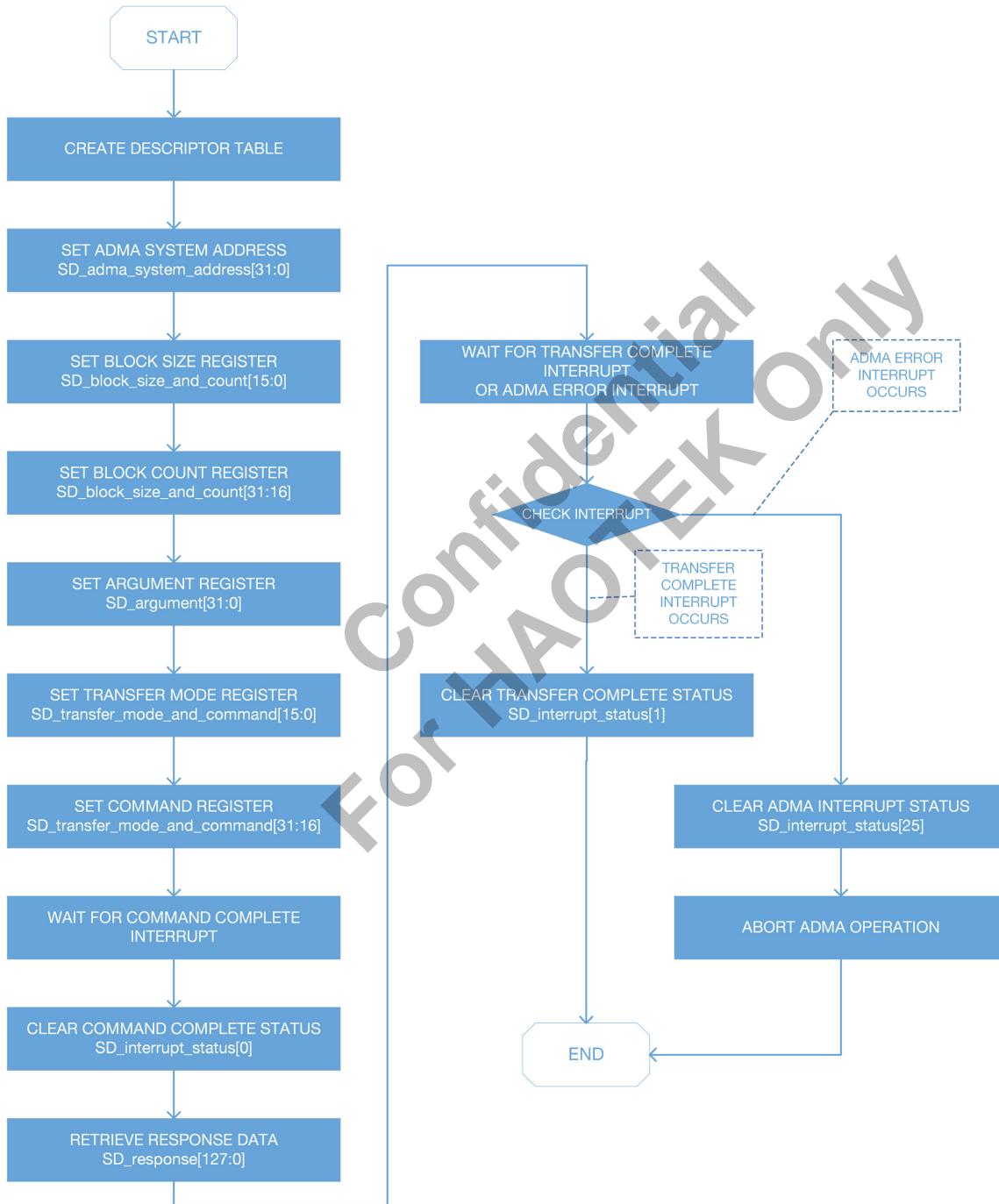


Figure 13-9. SD Controllers: ADMA Command Flow.

The following is a list of steps in the command flow for ADMA transfer operations.

1. Create the Descriptor table for ADMA in the system memory .
2. Set the Descriptor address for ADMA in the ADMA System Address register (**SD_adma_system_address[31:0]**).
3. Set the value corresponding to the executed data byte length of one block in the Block Size register (**SD_block_size_and_count[15:0]**). This step can be executed simultaneously with Step 4.
4. Set the value corresponding to the executed data block count in the Block Count register (**SD_block_size_and_count[31:16]**) in accordance with the Determination of Transfer Type as described in [Table 13-48](#). This step can be executed simultaneously with Step 3.
5. Set the Argument register value (**SD_argument[31:0]**).
6. Set the Transfer Mode register value (**SD_transfer_mode_and_command[15:0]**). The host driver determines Multi / Single Block Select (**sin_multi**), Block Count Enable (**blk_cnt_en**), Data Transfer Direction (**data_trans_dir_select**), Auto CMD12 Enable (**auto_cmd12_enable**) and DMA Enable (**dma_en**) bits. Multi / Single Block Select and Block Count Enable are determined according to the Determination of Transfer Type as described in [Table 13-48](#). This step can be executed simultaneously with Step 7.
7. Set the Command register value (**SD_transfer_mode_and_command[31:16]**). When writing the upper byte, the SD command is issued and DMA is initialized. This step can be executed simultaneously with Step 6.

8. Wait for the Command Complete interrupt.
9. Write 1 to Normal Interrupt Status register Command Complete bit (**SD_interrupt_status[0]**) to clear the interrupt.
10. Read the Response register (**SD_response[127:0]**) for necessary information about the issued command.
11. Wait for the Transfer Complete interrupt or the ADMA Error interrupt.
12. If the Transfer Complete interrupt is 1, proceed to Step 13. If the ADMA Error Interrupt is 1, proceed to Step 14.
13. Clear the Normal Interrupt Status register Transfer Complete Status (**SD_interrupt_status[1]**) by writing the bit to 1. The transfer operation is complete. Proceed to Step 16.
14. Clear the Normal Interrupt Status register ADMA Error Interrupt Status (**SD_interrupt_status[25]**) by writing the bit to 1.
15. Abort the ADMA operation. The SD card must be stopped by issuing an abort command. If necessary, the host driver checks the ADMA Error Status register to determine the reason for the ADMA error.
16. End of transfer operation.

13.5.6 SD Controllers: Register Map

The following table contains a map of the SD Controller registers. Note that several of the registers include fields with reset values that depend on SD / MMC boot modes.

Register Offset	Register Name	Description
0x000	SD_system_address	Single DMA (SDMA) System Address High/Low
0x004	SD_block_size_and_count	Block Size / Block Count (SD/MMC boot)
0x008	SD_argument	Argument
0x00C	SD_transfer_mode_and_command	Transfer Mode (SD/MMC boot) / Command (SD/MMC boot)
0x010 - 0x01E	SD_response	Response
0x020	SD_buffer_data_port	Buffer Data Port
0x024	SD_present_state	Present State
0x028	SD_hpbw_control	Host Control (SD/MMC boot) / Power Control / Block Gap / Wakeup
0x02C	SD_clockcntl_timeoutcntl_swreset	Clock Control (SD/MMC boot) / Timeout Control / Software Reset
0x030	SD_interrupt_status	Normal Interrupt Status / Error Interrupt Status
0x034	SD_interrupt_status_enable	Normal Interrupt Status Enable (SD/MMC boot) / Error Interrupt Status Enable
0x038	SD_interrupt_signal_enable	Normal Interrupt Signal Enable / Error Interrupt Signal Enable
0x03C	SD_auto_cmd12_error_status	Auto CMD12 Error Status register Host Control 2 register

Register Offset	Register Name	Description
0x040 - 0x044	SD_capabilities	Capabilities Register provides Host Driver with Information Specific to Host Controller Implementation
0x048 - 0x04C	SD_max_current_capabilities	Maximum Current Capabilities
0x050		Reserved
0x054	SD_adma_error_status	Advanced DMA (ADMA) Error Status
0x058	SD_adma_system_address	Advanced DMA (ADMA) System Address
0x060 - 0x06E		Reserved for Preset Value
0x070	SD_boot_control	Boot Control (SD/MMC boot)
0x074	SD_boot_status	Boot Status (SD/MMC boot)
0x078	SD_block_count_cmd23	Block Count for CMD23
0x07C	SD_voltage_switch	Voltage Switch
0x080 - 0x0E0		Reserved
0x0F8	SD_read_latency_control	Read Latency Control
0x0FC	SD_slot_interrupt_and_version	Slot Interrupt / Version ID

Table 13-42. SD Controller AHB Registers.

13.5.7 SD Controllers: Register Details

13.5.7.1 SD_system_address Register

This register contains the system memory address for a Single DMA (SDMA) transfer. When the Host Controller (HC) stops an SDMA transfer, this register points to the system address of the next contiguous data position. It can be accessed only if the execution of all transfers has ended. Read operations performed during a transfer will return an invalid value. The Host Driver (HD) initializes this register prior to initiating an SDMA transaction.

After the SDMA transfer has concluded, the system address of the next contiguous data position can be read from this register. The SDMA engine pauses at every boundary specified by the Host DMA Buffer Size in the Block Size register (**SD_block_size_and_count**) in order for the register to be updated. The Host Controller generates a DMA Interrupt to request that this update occur, and then the Host Driver sets the system address of the next data position. When the most significant upper byte of this register (003h) is written, the Host Controller restarts the SDMA transfer.

When restarting SDMA with the Resume command or by setting the Block Gap Control register (**SD_hpbw_control**) field **continue_request**, the Host Controller starts at the next contiguous address stored in the System Address register **SD_system_address**.

Bits	Name	Attr	Reset	Description
31:0	dma_system_address	RW	0	System Address for SDMA transfers (Not used for ADMA transfers)

Table 13-43. SD Controller SDMA System Address Register.

13.5.7.2 SD_block_size_and_count Register

Bits	Name	Attr	Reset	Description
31:16	block_count	RW	0 #	<p>Blocks Count for Current Transfer (Not used for CMD23)</p> <p>This register is enabled when the SD_transfer_mode_and_command field block_count_enable is set to 1 and is valid only for multiple block transfers. The Host Controller decrements the block count after each block transfer and stops when the count reaches zero. This register can be accessed only if no transaction is currently executing (i.e., after a transaction has stopped). Read operations performed during transfer will return an invalid value and write operations are ignored. When saving transfer context as a result of a Suspend command, the number of blocks yet to be transferred can be determined by reading this Register. When restoring transfer context prior to issuing a Resume command, the Host Driver restores the previously saved block count.</p> <p>In SDXC mode, when executing CMD23 before CMB18/25, the effective block count is the programmed SD_block_count_cmd23 register value (offset 0x78) and this register (offset 0x06) is ignored.</p> <p>0x0000 - Stop Count 0x0001 - 1 block 0x0002 - 2 blocks --- 0xFFFF - 65535 blocks</p> <p>Note that this bit is used for eMMC boot operation and the reset value depends on the MMC_BOOT configuration (see Section 2.3): For MMC_BOOT = 0, the reset value is 0x0000 (No data transfer) For MMC_BOOT = 1, the reset value is 0x0004 (4 blocks)</p>
15	block_size	RW	0	<p>Transfer Block Size 12th bit This bit is for 4-KB data block transfers</p> <p>(Continued Below)</p>

Bits	Name	Attr	Reset	Description
14:12	dma_buffer_size	RW	0	<p>SDMA Buffer Size (Used for SDMA only. ADMA does not use this Register.)</p> <p>To perform lengthy SDMA transfers, the System Address register (SD_system_address) must be updated at every system boundary during transfer, as large contiguous memory space may not be available in the virtual memory system. The host_dma_buffer_size bits specify the size of the contiguous buffer in the system memory. The SDMA transfer process pauses at every boundary specified by these fields and the Host Controller generates the DMA Interrupt to request a SD_system_address update.</p> <p>For this function to be active:</p> <ul style="list-style-type: none"> - The SD_capabilities field dma_support must be set to 1 - The SD_transfer_mode_and_command field dma_enable must be set to 1. <p>0x0 - 4 KB (Detects A11 Carry out) 0x1 - 8 KB (Detects A12 Carry out) 0x2 - 16 KB (Detects A13 Carry out) 0x3 - 32 KB (Detects A14 Carry out) 0x4 - 64 KB (Detects A15 Carry out) 0x5 - 128 KB (Detects A16 Carry out) 0x6 - 256 KB (Detects A17 Carry out) 0x7 - 512 KB (Detects A18 Carry out)</p> <p>(Continued Below)</p>

Bits	Name	Attr	Reset	Description
11:0	transfer_block_size	RW	0 #	<p>Transfer Block Size This register specifies the block size for block data transfers for SD commands CMD17, CMD18, CMD24, CMD25, and CMD53.</p> <p>The valid values range from 0 to 0x400 (1KB), which is less-than or equal to half of the SD FIFO size. This register can be accessed only if no transaction is currently executing (i.e., after a transaction has stopped). Read operations performed during transfer will return an invalid value and write operations are ignored.</p> <p>0x000 - No Data Transfer 0x001 - 1 bytes 0x002 - 2 bytes 0x003 - 3 bytes 0x004 - 4 bytes ---- 0x1FF - 511 bytes 0x200 - 512 bytes ---- 0x800 - 2048 bytes</p> <p>Note that during eMMC boot operation (MMC_BOOT = 1), transfer_block_size should be 512 bytes to match the MMC device block size. When the received data reach the block size, the received data are sent to the FIOS buffer.</p> <p>Reset values for this register depend on MMC_BOOT boot configuration inputs: For MMC_BOOT = 0, the reset value is 0x000 (No data transfer) For MMC_BOOT = 1, the reset value is 0x200 (512 B)</p>

Table 13-44. SD Controller Block Size / Block Count Register.

13.5.7.3 SD_argument Register

Bits	Name	Attr	Reset	Description
31:0	command_argument	RW	0	<p>Command Argument The SD Command Argument is specified as bits 39:8 of Command-Format.</p>

Table 13-45. SD Controller Argument Register.

13.5.7.4 SD_transfer_mode_and_command Register

Bits	Name	Attr	Reset	Description
31:30				Reserved
29:24	command_index	RW	0	<p>Command Index This bit is set to the command number (CMD0-63, ACMD0-63).</p>

Bits	Name	Attr	Reset	Description
23:22	command_type	RW	0	<p>Command Type This register involves three types of commands: Suspend, Resume and Abort. These bits must be set to 0x0 for all other commands.</p> <p>Suspend Command If the Suspend command succeeds, the Host Controller assumes the SD Bus has been released and it is therefore available to receive the next command that uses the DAT line. The Host Controller deasserts Read Wait for read transactions and stops checking for write transactions. The Interrupt cycle starts in 4-bit mode. If the Suspend command fails, the Host Controller maintains its current state, and the Host Driver restarts the transfer by setting the Block Gap Control register (SD_hpbw_control) field continue_request.</p> <p>Resume Command The Host Driver restarts the data transfer by restoring the registers in the range of 0x000-0x00D. The Host Controller checks for Busy before starting write transfers.</p> <p>Abort Command If this command is set when executing a read transfer, the Host Controller stops all reads to the buffer. If this command is set when executing a write transfer, the Host Controller stops driving the DAT line. After issuing the Abort command, the Host Driver must issue a software reset 0x0 - Normal 0x1 - Suspend 0x2 - Resume 0x3 - Abort</p>
21	data_present_select	RW	0 #	<p>Data Present Select This bit is set to 1 to indicate that data is present and will be transferred using the DAT line. It is set to 0 when the following three commands are issued: (1) Commands using only the CMD line (e.g., CMD52); (2) Commands with no data transfer instructions but which use a Busy signal on the DAT[0] line; (3) Resume commands.</p> <p>0 - No Data Present 1 - Data Present</p> <p>Note that this bit is used for eMMC boot operation and the reset value depends on the MMC_BOOT configuration (see Section 2.3): For MMC_BOOT = 0, the reset value is 0 For MMC_BOOT = 1, the reset value is 1</p>

Bits	Name	Attr	Reset	Description
20	command_index_check_enable	RW	0	<p>Command Index Check Enable</p> <p>If this bit is set to 1, the Host Controller checks the index field in the response to confirm whether it has the same value as the command index. If the values differ, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked.</p> <p>0 - Disable 1 - Enable</p>
19	command_crc_check_enable	RW	0	<p>Command CRC Check Enable</p> <p>If this bit is set to 1, the Host Controller checks the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked.</p> <p>0 - Disable 1 - Enable</p>
18				Reserved
17:16	response_type_select	RW	0	<p>Response Type Select</p> <p>0x0 - No Response 0x1 - Response length 136 0x2 - Response length 48 0x3 - Response length 48 check Busy after response</p>
15:7				Reserved
6	cmd_comp_int_en_ata	RW	0 #	<p>Command completion signal enable for CE-ATA Devices</p> <p>0 - Device will not send a command completion signal 1 - Device will send a command completion signal</p> <p>Note that this bit is used for eMMC boot operation and the reset value is independent of the MMC_BOOT configuration for the A12 chip (see Section 2.3).</p>
5	sin_multi	RW	0 #	<p>Multi / Single Block Select</p> <p>This bit enables multiple block DAT line data transfers.</p> <p>0 - Single Block 1 - Multiple Block</p> <p>Note that this bit is used for eMMC boot operation and the reset value depends on the MMC_BOOT configuration (see Section 2.3):</p> <p>For MMC_BOOT = 0, the reset value is 0 For MMC_BOOT = 1, the reset value is 1</p>
4	data_trans_dir_select	RW	0 #	<p>Data Transfer Direction Select</p> <p>This bit defines the direction of DAT line data transfers.</p> <p>0 - Write (Host to Card) 1 - Read (Card to Host)</p> <p>Note that this bit is used for eMMC boot operation and the reset value depends on the MMC_BOOT configuration (see Section 2.3):</p> <p>For MMC_BOOT = 0, the reset value is 0 For MMC_BOOT = 1, the reset value is 1</p>
3				Reserved

Bits	Name	Attr	Reset	Description
2	auto_cmd12_enable	RW	0 #	<p>Auto CMD12 Enable Multiple block transfers from memory require CMD12 to stop the transaction. When this bit is set to 1, the Host Controller issues CMD12 automatically when the last block transfer is completed. The Host Driver does not set this bit to issue commands that do not require CMD12 to stop data transfer.</p> <p>0 - Disable 1 - Enable</p> <p>Note that this bit is used for eMMC boot operation and the reset value is independent of the MMC_BOOT configuration for the A12 chip (see Section 2.3).</p>
1	blk_cnt_en	RW	0 #	<p>Block Count Enable This bit is used to enable the Block count register (SD_block_size_and_count), which is only relevant for multiple block transfers. When this bit is 0, the Block Count register is disabled, which is useful in executing an infinite transfer.</p> <p>This bit should be set to 1 for ADMA mode</p> <p>0 - Disable 1 - Enable</p> <p>Note that this bit is used for eMMC boot operation and the reset value depends on the MMC_BOOT configuration (see Section 2.3):</p> <p>For MMC_BOOT = 0, the reset value is 0 For MMC_BOOT = 1, the reset value is 1</p>
0	dma_en	RW	0 #	<p>DMA Enable DMA can be enabled only if the SD_capabilities field dma_support is set. If this bit is set to 1, a DMA operation will begin when the Host Driver writes to the upper byte of Command register SD_transfer_mode_and_command (0x00F).</p> <p>0 - Disable 1 - Enable</p> <p>Note that this bit is used for eMMC boot operation and the reset value is independent of the MMC_BOOT configuration for the A12 chip (see Section 2.3).</p>

Table 13-46. SD Controller Transfer Mode Register.

Response Type	Index Check Enable	CRC Check Enable	Name of Response Type
00	0	0	No Response
01	0	1	R2
10	0	0	R3, R4
10	1	1	R1, R6, R5
11	1	1	R1b, R5b

Table 13-47. Relation Between Parameters and the Name of Response Type.

Multi / Single Block Select sin_multi	Block Count Enable blk_cnt_en	Block Count blk_cnt	Function
0			Single Transfer
1	0		Infinite Transfer
1	1	Not Zero	Multiple Transfer
1	1	Zero	Stop Multiple Transfer

Table 13-48. SD Controller Determination of Transfer Type.

13.5.7.5 SD_response Register

Bits	Name	Attr	Reset	Description
127:0	command_response	R	0	<p>Command Response</p> <p>The following table describes the mapping of command responses from the SD Bus to this register. In the table, R[] refers to a bit range within the response data as transmitted on the SD Bus. REP[] refers to a bit range within the Response register (SD_response).</p>

Table 13-49. SD Controller Response Register.

Type of Response	Meaning of Response	Response Field	Response Register
R1, R1b (normal response)	Card Status	R[39:8]	REP[31:0]
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R[39:8]	REP[127:96]
R2 (CID, CSD Register)	CID or CSD reg. incl.	R[127:8]	REP[119:0]
R3 (OCR Register)	OCR Register for memory	R[39:8]	REP[31:0]
R4 (OCR Register)	OCR Register for I/O etc.	R[39:8]	REP[31:0]
R5, R5b	SDIO Response	R[39:8]	REP[31:0]
R6 (Published RCA response)	New published RCA[31:16] etc.	R[39:8]	REP[31:0]

Table 13-50. Response Bit Definition for Each Response Type.

13.5.7.6 SD_buffer_data_port Register

Bits	Name	Attr	Reset	Description
31:0	buffer_data	RW	NR	<p>Buffer Data</p> <p>The Host Controller Buffer can be accessed through this 32-bit Data Port Register.</p>

Table 13-51. SD Controller Buffer Data Port Register.

13.5.7.7 SD_present_state Register

This bit indicates whether one of the DAT lines on SD bus is in use.

Bits	Name	Attr	Reset	Description
31:29				Reserved
28:25	dat_line_signal_level	R	0	DAT[7:4] Line Signal Level This register is used to check the DAT line level for error recovery, as well as for debugging. D25 - DAT[4] D26 - DAT[5] D27 - DAT[6] D28 - DAT[7]
24	cmd_line_sig- nal_level	R	0	CMD Line Signal Level This register is used to check the CMD line level for error recovery, and for debugging.
23:20	dat_line_signal_level	R	0	DAT[3:0] Line Signal Level This register is used to check the DAT line level for error recovery, and for debugging. This is especially useful in detecting the Busy signal level from DAT[0]. D20 - DAT[0] D21 - DAT[1] D22 - DAT[2] D23 - DAT[3]
19	write_protect_ switch_pin_level	R	0	Write Protect Switch Pin Level The Write Protect Switch is supported for memory and combination cards. This bit reflects the SD[N].WP pin. 0 - Write protected (SD[N].WP = 1) 1 - Write enabled (SD[N].WP = 0)
18	card_detect_pin_ level	R	0	Card Detect Pin Level This bit reflects the inverse value of the SD[N].CD pin. 0 - No Card present (SD[N].CD = 1) 1 - Card present (SD[N].CD = 0)
17	card_state_stable	R	0	Card State Stable This bit is used for testing. If it is 0, the Card Detect Pin Level card_detect_pin_level is unstable. If this bit is set to 1, it means the Card Detect Pin Level is stable. The Software Reset register (SD_clockctl_timeoutctrl_swreset) field software_reset_for_all will not affect this bit. 0 - Reset or Debouncing 1 - No Card or Incorrectly Inserted

Bits	Name	Attr	Reset	Description
16	card_inserted	R	0	<p>Card Inserted</p> <p>This bit indicates whether a card has been inserted. Changing from 0 to 1 generates a Card Insertion Interrupt in the Normal Interrupt Status register (SD_interrupt_status) and changing from 1 to 0 generates a Card Removal Interrupt in SD_interrupt_status. The Software Reset register (SD_clockctl_timeoutctrl_swreset) field software_reset_for_all does not affect this bit.</p> <p>If a card is removed while powered and while its clock is oscillating, the Host Controller clears the Power Control register (SD_hpbw_control) bit sd_bus_power and the Clock control register (SD_clockctl_timeoutctrl_swreset) bit sd_clock_enable. In addition, the Host Driver will clear the Host Controller by software_reset_for_all. The card detect is active regardless of sd_bus_power.</p> <p>0 - Reset or Debouncing or No Card 1 - Card Inserted</p>
15:12				Reserved
11	buffer_read_enable	R	0	<p>Buffer Read Enable</p> <p>This register is used for non-DMA read transfers. This read-only flag indicates that valid data exists in the host-side buffer. If this bit is 1, readable data exists in the buffer. A change of this bit from 1 to 0 occurs when all the block data is read from the buffer. A change of this bit from 0 to 1 occurs when all the block data is ready in the buffer and generates the Buffer Read Ready Interrupt.</p> <p>0 - Read Disable 1 - Read Enable</p>
10	buffer_write_enable	R	0	<p>Buffer Write Enable</p> <p>This register is used for non-DMA write transfers. This read only flag indicates if space is available for write data. If this bit is 1, data can be written to the buffer. A change of this bit from 1 to 0 occurs when all the block data is written to the buffer. A change of this bit from 0 to 1 occurs when top of block data can be written to the buffer and generates the Buffer Write Ready Interrupt.</p> <p>0 - Write Disable 1 - Write Enable</p>

Bits	Name	Attr	Reset	Description
9	read_transfer_active	R	0	<p>Read Transfer Active This register is used for detecting completion of a read transfer.</p> <p>This bit is set to 1 under either of the following two conditions: (1) After the end-bit of the read command; (2) When writing a 1 to the Block Gap Control register (SD_hpbw_control) field continue_request to restart a read transfer.</p> <p>This bit is cleared to 0 under any of the following three conditions: (1) When the last data block (as specified by block length) is transferred to the system; (2) In the case of ADMA, when the end-of-read operation is designated by Descriptor Table; (3) When all valid data blocks have been transferred to the system and no current block transfers are being sent as a result of the Block Gap Control register (SD_hpbw_control) field stop_at_block_gap_request being set to 1. A transfer complete interrupt is generated when this bit goes to 0.</p> <p>0 - No valid data 1 - Transferring data</p>
8	write_transfer_active	R	0	<p>Write Transfer Active This register indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the Host Controller.</p> <p>This bit is set under either of the following two conditions: (1) After the end-bit of the write command. (2) When writing a 1 to the Block Gap Control register (SD_hpbw_control) field continue_request to restart a write transfer.</p> <p>This bit is cleared under either of the following conditions: (1) After retrieving the CRC status of the last data block as specified by the transfer count (Single or Multiple). In the case of ADMA, the transfer count is designated by the Descriptor Table; (2) After retrieving the CRC status of a block where data transmission is about to be stopped by a Block Gap Control register (SD_hpbw_control) field stop_at_block_gap_request.</p> <p>During a write transaction, a Block Gap Event Interrupt is generated when this bit is changed to 0, as a result of the stop_at_block_gap_request being set. This register is useful for the Host Driver in determining when to issue commands during write Busy.</p> <p>0 - No valid data 1 - Transferring data</p>
7:3				Reserved

Bits	Name	Attr	Reset	Description
2	dat_line_active	R	0	<p>This bit indicates whether one of the DAT lines on the SD bus is in use.</p> <p>(a) In the case of read transactions: This bit indicates whether a read transfer is currently executing on the SD Bus. Changing this value from 1 to 0 generates a Block Gap Event interrupt in the Normal Interrupt Status register (SD_interrupt_status) as the result of the Stop At Block Gap Request being set.</p> <p>This bit will be set under either of the following conditions: (1) After the end-bit of the read command. (2) When writing a 1 to Continue Request in the Block Gap Control register (SD_hpbw_control) to restart a read transfer.</p> <p>This bit will be cleared under either of the following conditions: (1) When the end-bit of the last data block is sent from the SD Bus to the Host Controller. In the case of ADMA, the last block is designated by the last transfer in the Descriptor Table. (2) When a read transfer is stopped at the block gap due to a Stop At Block Gap Request. The Host Controller will stop the read operation at the start of the interrupt cycle of the next block gap by driving Read Wait or stopping the SD clock.</p> <p>(Continued Below)</p>

Bits	Name	Attr	Reset	Description
				<p>(b) In the case of write transactions: This bit indicates that a write transfer is currently executing on the SD Bus. Changing this value from 1 to 0 generates a Transfer Complete interrupt in the Normal Interrupt Status register (SD_interrupt_status).</p> <p>This bit will be set under either of the following conditions: (1) After the end-bit of the write command. (2) When writing a 1 to Continue Request in the Block Gap Control register (SD_hpbw_control) to continue a write transfer.</p> <p>This bit will be cleared under either of the following conditions: (1) When the SD card releases Write Busy for the last data block. If the SD card does not drive Busy signals, the Host Controller will consider the card drive Not Busy. In the case of ADMA, the last block is designated by the last transfer in the Descriptor Table. (2) When the SD card releases Write Busy as a result of a Stop At Block Gap Request.</p> <p>(c) Command with Busy: This bit indicates whether a command that shows a Busy status (e.g., erase command) is executing on the SD Bus. This bit is set after the end-bit of the command and cleared when Busy is deasserted.</p> <p>Changing this bit from 1 to 0 generates a Transfer Complete interrupt in the Normal Interrupt Status register (SD_interrupt_status).</p> <p>0 - DAT line inactive 1 - DAT line active</p>
1	command_inhibit_dat	R	0	<p>Command Inhibit (DAT) This register bit is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this bit is 0, it indicates that the Host Controller can issue the next SD command. Changing this bit from 1 to 0 generates a Transfer Complete Interrupt in the Normal Interrupt Status register (SD_interrupt_status).</p> <p>Note: The SD Host Driver can save registers in the range of 0x000-0x00D for a suspend transaction after this bit has changed from 1 to 0.</p> <p>0 - Can issue command which uses the DAT line 1 - Cannot issue command which uses the DAT line</p>

Bits	Name	Attr	Reset	Description
0	command_inhibit_cmd	R	0	<p>Command Inhibit (CMD)</p> <p>If this bit is 0, it indicates the CMD line is not in use and the Host Controller can issue a SD command using the CMD line. This bit is set immediately after the Command register SD_transfer_mode_and_command (0x00F) is written. This bit is cleared when the command response is received. Even if the Command Inhibit (DAT) is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing this bit from 1 to 0 generates a Command Complete Interrupt in the Normal Interrupt Status register (SD_interrupt_status). If the Host Controller cannot issue the command due to a command conflict error or a Command Not Issued By Auto CMD12 Error, this bit will remain 1 and the Command Complete will not be set. Status issuing Auto CMD12 is not read from this bit.</p>

Table 13-52. SD Controller Present State Register.

Note that the Host Driver can issue CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DAT lines are Busy during data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands can only be issued when Command Inhibit (DAT) is set to zero.

13.5.7.8 SD_hpbw_control Register

Bits	Name	Attr	Reset	Description
31:27				Reserved
26	wakeup_event_enable_on_sd_card_removal	RW	0	<p>Wakeup Event Enable On SD Card Removal</p> <p>This bit enables a wakeup event via Card Removal assertion in the Normal Interrupt Status register (SD_interrupt_status). FN_WUS (Wake up Support) in CIS does not affect this bit.</p> <p>0 - Disable 1 - Enable</p>
25	wakeup_event_enable_on_sd_card_insertion	RW	0	<p>Wakeup Event Enable On SD Card Insertion</p> <p>This bit enables a wakeup event via Card Insertion assertion in the Normal Interrupt Status register (SD_interrupt_status). FN_WUS (Wake up Support) in CIS does not affect this bit.</p> <p>0 - Disable 1 - Enable</p>
24	wakeup_event_enable_on_card_interrupt	RW	0	<p>Wakeup Event Enable On Card Interrupt</p> <p>This bit enables a wakeup event via Card Interrupt assertion in the Normal Interrupt Status register (SD_interrupt_status). This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1.</p> <p>0 - Disable 1 - Enable</p>
23:20				Reserved
19	interrupt_at_block_gap	RW	0	<p>Interrupt At Block Gap</p> <p>Setting this bit to 1 enables interrupt detection at the block gap for a multiple block transfer. If the SD card cannot signal an interrupt during a multiple block transfer, this bit must be set to 0. When the Host Driver detects an SD card insertion, it will set this bit according to the CCCR of the SDIO card.</p>

Bits	Name	Attr	Reset	Description
18	read_wait_control	RW	0	<p>Read Wait Control The read-wait function is optional for SDIO cards. If the card supports read-wait, set this bit to enable use of the read-wait protocol to stop reading data using the DAT[2] line. Otherwise, the Host Controller is required to stop the SD clock to hold read data, which restricts commands generation. When the Host Driver detects an SD card insertion, it will set this bit according to the CCCR of the SDIO card. If the card does not support read-wait, this bit must not be set to 1; otherwise, DAT line conflict may occur. If this bit is set to 0, Suspend / Resume is not supported.</p> <p>0 - Disable Read Wait Control 1 - Enable Read Wait Control</p>
17	continue_request	RW_SC	0	<p>Continue Request This bit is used to restart a transaction which was stopped using the stop_at_block_gap_request. To cancel a Stop At Block Gap Request, set stop_at_block_gap_request to 0 and set this bit to restart the transfer. The Host Controller automatically clears this bit under either of the following conditions: (1) In the case of a read transaction, the DAT Line Active changes from 0 to 1 as a read transaction restarts; (2) In the case of a write transaction, the Write transfer active changes from 0 to 1 as the write transaction restarts. It is not necessary for the Host driver to set this bit to 0. If stop_at_block_gap_request is set to 1, a write to this bit is ignored.</p> <p>0 - Ignored 1 - Restart</p>
16	stop_at_block_gap_request	RW	0	<p>Stop At Block Gap Request This bit is used to stop executing a transaction at the next block gap for non-DMA transfers. Until the transfer complete bit is set to 1, indicating a transfer completion, the Host Driver must leave this bit set to 1. Clearing both the stop_at_block_gap_request and continue_request does not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The Host Controller honors the stop_at_block_gap_request for write transfers, but for read transfers it requires that the SD card support Read Wait. Therefore, the Host Driver does not set this bit during read transfers unless the SD card supports Read Wait and has set Read Wait Control to 1. In the case of write transfers in which the Host Driver writes data to the Buffer Data Port register (SD_buffer_data_port), the Host Driver will set this bit after all block data is written. If this bit is set to 1, the Host Driver will not write data to the Buffer data port Register. This bit affects Read Transfer Active, Write Transfer Active, DAT line active and Command Inhibit (DAT) in the Present State register (SD_present_state). 0 - Transfer 1 - Stop</p>
15:12				Reserved

Bits	Name	Attr	Reset	Description
11:9	sd_bus_voltage_select	RW	0	<p>SD Bus Voltage Select</p> <p>By setting these bits, the Host Driver selects the voltage level for the SD card. Before setting this register, the Host Driver should check the Capabilities register (SD_capabilities) voltage support bits. If an unsupported voltage is selected, the Host System will not supply the SD bus voltage</p> <p>0x0 through 0x4 - Reserved</p> <p>0x5 - 1.8 V (Typ.)</p> <p>0x6 - 3.0 V (Typ.)</p> <p>0x7 - 3.3 V (Flat-top.)</p>
8	sd_bus_power	RW	0	<p>SD Bus Power</p> <p>Before setting this bit, the SD host driver will set sd_bus_voltage_select. If the Host Controller detects a No Card State, this bit will be cleared.</p> <p>0 - Power off</p> <p>1 - Power on</p>
7:5				Reserved
4	dma_select	RW	0	<p>DMA Select</p> <p>One of the supported DMA modes can be selected using this bit. The host driver will verify support of DMA modes by referring the Capabilities register (SD_capabilities). Use of the selected DMA is determined by the DMA Enable fields of the Transfer Mode register (SD_transfer_mode_and_command).</p> <p>0 - SDMA is selected</p> <p>1 - 32-bit address ADMA is selected</p>
3	sd8_bit_mode	RW	0 #	<p>Data Transfer Width - SD8 Bit Mode</p> <p>This bit selects the data width of the Host Controller. The Host Driver will set it to match the data width of the SD card.</p> <p>0 - 8-bit mode is not selected</p> <p>1 - 8-bit mode is selected</p> <p>Note that this bit is used for eMMC boot operation and the reset value depends on the MMC_BOOT configuration (see Section 2.3):</p> <p>For MMC_BOOT = 0, the reset value is 0</p> <p>For MMC_BOOT = 1, the reset value depends on the POC SD8_BOOT bit</p>
2	high_speed_enable	RW	0 #	<p>High Speed Enable</p> <p>If this bit is set to 0 (default), the Host Controller outputs CMD line and DAT lines at the falling edge of the SD clock (up to 25 MHz). If this bit is set to 1, the Host Controller outputs CMD line and DAT lines at the rising edge of the SD clock (up to 50 MHz)</p> <p>0 - Normal Speed Mode</p> <p>1 - High Speed Mode</p> <p>Note that this bit is used for eMMC boot operation and the reset value depends on the MMC_BOOT configuration (see Section 2.3):</p> <p>For MMC_BOOT = 0, the reset value is 0</p> <p>For MMC_BOOT = 1, the reset value depends on the POC HS_BOOT bit</p>

Bits	Name	Attr	Reset	Description
1	data_transfer_width_sd1_sd4	RW	0 #	<p>Data Transfer Width - SD1 bit mode / SD4 bit mode This bit selects the data width of the Host Controller. The Host Driver will select it to match the data width of the SD card. 0 - 1-bit mode 1 - 4-bit mode Note that SD1 mode is selected by setting this bit and sd8_bit_mode to 0.</p> <p>Note that this bit is used for eMMC boot operation and the reset value depends on the MMC_BOOT configuration (see Section 2.3): For MMC_BOOT = 0, the reset value is 0 For MMC_BOOT = 1, the reset value depends on the POC SD4_BOOT bit</p>
0				Reserved

Table 13-53. SD Controller Host Control/Power Control/Block Gap/Wakeup Control Register.

13.5.7.9 SD_clockctl_timeoutctl_swreset Register

During Host Controller initialization, the Host Driver will set the **SD[N]_CLK** Frequency Select and the Data Timeout Counter Value according to the Capabilities register (**SD_capabilities**).

Bits	Name	Attr	Reset	Description
31:27				Reserved
26	software_reset_for_dat_line	RW_SC	0	<p>Software Reset for DAT line Only a portion of the data circuit is reset. The following registers and bits are cleared by this bit: Buffer Data Port register (SD_buffer_data_port) Buffer is cleared and initialized Present State register (SD_present_state) Buffer Read Enable Buffer Write Enable Read Transfer Active Write Transfer Active DAT Line Active Command Inhibit (DAT) Block Gap Control register (SD_hpbw_control) Continue Request Stop At Block Gap Request Normal Interrupt Status register (SD_interrupt_status) Buffer Read Ready Buffer Write Ready Block Gap Event Transfer Complete 0 - Work 1 - Reset</p>

Bits	Name	Attr	Reset	Description
25	software_reset_for_cmd_line	RW_SC	0	<p>Software Reset for CMD Line Only part of command circuit is reset. The following registers and bits are cleared by this bit: Present State register (SD_present_state) Command Inhibit (CMD) Normal Interrupt Status register (SD_interrupt_status) Command Complete 0 - Work 1 - Reset</p>
24	software_reset_for_all	RW_SC	0	<p>Software Reset for All This reset affects the entire Host Controller with the exception of the card detection circuit. Register bits of type R, RW, RW_SC are cleared to 0. During its initialization, the Host Driver will set this bit to 1 to reset the Host Controller. The Host Controller will reset this bit to 0 when Capabilities registers (SD_capabilities) are valid and the Host Driver can read them. Additional use of Software Reset For All may not affect the value of SD_capabilities. If this bit is set to 1, the SD card will reset itself and must be re-initialized by the Host Driver. 0 - Work 1 - Reset</p>
23:20				Reserved
19:16	data_timeout_counter_value	RW	0	<p>Data Timeout Counter Value This value determines the interval by which DAT line timeouts are detected. Refer to the Data Timeout Error in the Error Interrupt Status register (SD_interrupt_status) for information on factors that dictate timeout generation. Timeout clock frequency will be generated by dividing the base clock TMCLK by this value. When setting this register, prevent inadvertent timeout events by clearing the Data Timeout Error Status Enable (SD_interrupt_status_enable) 0x0 - TMCLK * 2^13 0x1 - TMCLK * 2^14 ----- ----- 0xE - TMCLK * 2^27 0xF - Reserved</p>

Bits	Name	Attr	Reset	Description
15:8	sdclk_frequency_select	RW	0	<p>SD[N].CLK Frequency Select This register is used to select the frequency of the SD[N].CLK pin. The frequency is not programmed directly; rather this register holds the divisor of the Base Clock Frequency for the SD clock in the Capabilities register (SD_capabilities). Only the following settings are permitted.</p> <ul style="list-style-type: none"> 0x00 - base clock 0x01 - base clock divided by 2 0x02 - base clock divided by 4 0x04 - base clock divided by 8 0x08 - base clock divided by 16 0x10 - base clock divided by 32 0x20 - base clock divided by 64 0x40 - base clock divided by 128 0x80 - base clock divided by 256 <p>Setting 0x00 specifies the highest frequency of the SD Clock. Multiple bits should not be set.</p> <p>The two default divider values can be calculated by the frequency that is defined by the Capabilities register (SD_capabilities) bit base_clock_frequency_for_sd_clock: (1) 25 MHz divider value, and (2) 400 KHz divider value</p> <p>The frequency of the SD[N].CLK is set by the following formula: Clock Frequency = (Base clock) / divisor.</p> <p>Ambarella recommends choosing the smallest possible divisor which results in a clock frequency that is less-than or equal to the target frequency. Maximum Frequency = 50 MHz (base clock) Minimum Frequency = 195.3125 KHz (50 MHz / 256)</p>
7:3	Reserved			
2	sd_clock_enable	RW	1 #	<p>SD Clock Enable The Host Controller stops SD[N].CLK when writing this bit to 0. SD[N].CLK Frequency Select can be changed when this bit is 0. The Host Controller maintains the same clock frequency until SD[N].CLK is stopped (Stop at SD[N].CLK = 0). If the Host Controller detects a No Card state, this bit will be cleared.</p> <ul style="list-style-type: none"> 0 - Disable 1 - Enable <p>Note that this bit is used for eMMC boot operation and the reset value is independent of the MMC_BOOT configuration for the A12 chip (see Section 2.3).</p>
1	internal_clock_stable	R	0	<p>Internal Clock Stable This bit is set to 1 when the SD clock is stable after writing the internal_clock_enable field to 1. The SD Host Driver will wait to set sd_clock_enable until this bit is set to 1. Note that this bit is useful when using a PLL that requires the setup time.</p> <ul style="list-style-type: none"> 0 - Not Ready 1 - Ready

Bits	Name	Attr	Reset	Description
0	internal_clock_enable	RW	1 #	<p>Internal Clock Enable This bit is set to 0 when the Host Driver is not using the Host Controller or when the Host Controller is awaiting a wakeup event. The Host Controller should stop its internal clock for a very low power state. Registers can still be read and written. The clock begins to oscillate when this bit is set to 1. When clock oscillation is stable, the Host Controller will set the internal_clock_stable field to 1. This bit will not affect card detection.</p> <p>0 - Stop 1 - Oscillate</p> <p>Note that this bit is used for eMMC boot operation and the reset value is independent of the MMC_BOOT configuration for the A12 chip (see Section 2.3).</p>

Table 13-54. SD Controller Clock Control / Timeout Control / Software Reset Register.

13.5.7.10 SD_interrupt_status Register

The Interrupt Status register (**SD_interrupt_status**) combines Normal Interrupt Status and Error Interrupt Status functions. The Normal Interrupt Enable Status functions affect the read value of this register, while Normal Interrupt Signal register functions do not. An Interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits except Card Interrupt and Error Interrupt, writing 1 to a bit clears it. The Card Interrupt is cleared when the Card Driver services the interrupt condition.

The Error Interrupt Status values defined in this register can be enabled via the Error Interrupt Status Enable register, but not by the Error Interrupt Signal Enable Register. The Interrupt is generated when the Error Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. Writing to 1 clears the bit and writing to 0 keeps the bit unchanged. More than one status can be cleared with one register write.

Bits	Name	Attr	Reset	Description
31:28	vendor_specific_error_status	RW	0	Vendor-Specific Error Status Additional status bits defined in this register by the vendor
27:26				Reserved
25	adma_error	RW	0	ADMA Error This bit is set when the Host Controller detects errors during an ADMA-based data transfer. The state of the ADMA at an error occurrence is saved in the ADMA Error Status register (SD_adma_error_status). In addition, the Host Controller generates this Interrupt when it detects invalid descriptor data (Valid=0) at the ST_FDS state. The ADMA Error State (adma_error_state) field in the ADMA Error Status register indicates whether an error occurs in ST_FDS state. The Host Driver may find that the Valid bit is not set at the error descriptor.

Bits	Name	Attr	Reset	Description
24	auto_cmd12_error	RW	0	<p>Auto CMD12 Error.</p> <p>This error occurs when one of the bits in Auto CMD12 Error Status register (SD_auto_cmd12_error_status) has changed from 0 to 1. In addition, this bit is set to 1 when Auto CMD12 is not executed due to the previous command error.</p> <p>0 - No Error 1 - Error</p>
23	current_limit_error	RW	0	<p>Current Limit Error</p> <p>By setting the Power Control register (SD_hpbw_control) bit sd_bus_power, the Host Controller is requested to supply power for the SD Bus. If the Host Controller supports the Current Limit Function, it can be protected from an illegal card by terminating the power supply, in which case this bit indicates a failure status. Reading 1 indicates the Host Controller is not supplying power to the SD card. Reading 0 indicates that the Host Controller is supplying power and that no error has occurred. This bit will always be set to 0 if the Host Controller does not support this function.</p> <p>0 - No Error 1 - Power Fail</p>
22	data_end_bit_error	RW	0	<p>Data End-Bit Error</p> <p>This error occurs when a 0 is detected at the end-bit position of read data which uses the DAT line or the end-bit position of the CRC status.</p> <p>0 - No Error 1 - Error</p>
21	data_crc_error	RW	0	<p>Data CRC Error</p> <p>This error occurs when a CRC error is detected during a transfer of read data which uses the DAT line, or when the Write CRC Status returns a value other than 010.</p> <p>0 - No Error 1 - Error</p>
20	data_timeout_error	RW	0	<p>Data Timeout Error</p> <p>This error occurs when one of following timeout conditions is detected: (1) Busy Timeout - R1b, R5b type; (2) Busy Timeout after Write CRC stat; (3) Write CRC status Timeout; (4) Read Data Timeout</p> <p>0 - No Error 1 - Timeout</p>
19	command_index_error	RW	0	<p>Command Index Error</p> <p>This error occurs if a Command Index error is indicated by the Command Response.</p> <p>0 - No Error 1 - Error</p>
18	command_end_bit_error	RW	0	<p>Command End-Bit Error</p> <p>This error occurs when the end-bit of a Command Response is 0.</p> <p>0 - No Error 1 - End-Bit Error Generated</p>

Bits	Name	Attr	Reset	Description
17	command_crc_error	RW	0	<p>Command CRC Error</p> <p>Command CRC Error is generated in two cases: (1) If a response is returned and the Command Timeout Error (command_timeout_error) is set to 0, this bit is set to 1 when detecting a CRT error in the command response. (2) The Host Controller detects a CMD-line conflict by monitoring the CMD line when a command is issued.</p> <p>If the Host Controller drives the CMD line to 1 level, but detects 0 level on the CMD line at the next SD[N].CLK edge, then the Host Controller will abort the command (i.e., stop driving the CMD line) and set this bit to 1.</p> <p>The Command Timeout Error will also be set to 1 to distinguish CMD line conflict.</p> <p>0 - No Error 1 - CRC Error Generated</p>
16	command_timeout_error	RW	0	<p>Command Timeout Error</p> <p>This error occurs if no response is returned within 64 SD[N].CLK cycles from the end-bit of the command. If the Host Controller detects a CMD-line conflict, this bit will be set without waiting for 64 SD[N].CLK cycles, as the command will be aborted.</p> <p>0 - No Error 1 - Timeout</p>
15	error_interrupt	R	0	<p>Error Interrupt</p> <p>Setting any bit in the Error Interrupt Status register (SD_interrupt_status) sets this bit. Therefore, the Host Driver can test for an error by checking this bit first.</p> <p>0 - No Error 1 - Error</p>
14:9				Reserved
8	card_interrupt	R	0	<p>Card Interrupt</p> <p>Writing this bit to 1 does not clear this bit. It is cleared by resetting the SD card interrupt factor. In 1-bit mode, the Host Controller will detect the card interrupt without wakeup support. In 4-bit mode, the card-interrupt signal is sampled during the interrupt cycle.</p> <p>When this register has been set, Card Interrupt Status Enable in the Normal Interrupt Status register (SD_interrupt_status) will be set to 0 in order to clear the card-interrupt statuses latched in the Host Controller and stop driving the Host System. After completion of the card-interrupt service (the reset factor in the SD card and the interrupt signal may not be asserted), set Card Interrupt Status Enable to 1 and begin sampling the interrupt signal again.</p> <p>0 - No Card Interrupt 1 - Generate Card Interrupt</p>
7	card_removal	RW	0	<p>Card Removal</p> <p>This register is set if the Card Inserted bit in the Present State register (SD_present_state) changes from 1 to 0. When the Host Driver writes this bit to 1, the status of the Card Inserted bit should be confirmed.</p> <p>0 - Card State Stable or Debouncing 1 - Card Removed</p>

Bits	Name	Attr	Reset	Description
6	card_insertion	RW	0	<p>Card Insertion</p> <p>This register is set if the Card Inserted bit in the Present State register (SD_present_state) changes from 0 to 1. When the Host Driver writes this bit to 1, the status of the Card Inserted bit should be confirmed..</p> <p>0 - Card State Stable or Debouncing 1 - Card Inserted</p>
5	buffer_read_ready	RW	0	<p>Buffer Read Ready</p> <p>This register is set if the Buffer Read Enable bit changes from 0 to 1.</p> <p>0 - Not Ready to read Buffer 1 - Ready to read Buffer</p>
4	buffer_write_ready	RW	0	<p>Buffer Write Ready</p> <p>This register is set if the Buffer Write Enable bit changes from 0 to 1.</p> <p>0 - Not Ready to Write Buffer 1 - Ready to Write Buffer</p>
3	dma_interrupt	RW	0	<p>DMA Interrupt Status</p> <p>This register is set if the Host Controller detects the Host DMA Buffer Boundary in the Block Size register (SD_block_size_and_count).</p> <p>In the case of ADMA, the Host Controller generates this interrupt by setting the Int field in the Descriptor Table.</p> <p>0 - No DMA Interrupt 1 - DMA Interrupt is Generated</p>
2	block_gap_event	RW	0	<p>Block Gap Event</p> <p>If the Block Gap Control register (SD_hpbw_control) field stop_at_block_gap_request is set, this bit is set.</p> <p>Read Transaction:</p> <p>This bit is set at the falling edge of the DAT Line Active Status (when the transaction is stopped). Read Wait access must be supported to use this function.</p> <p>Write Transaction:</p> <p>This bit is set at the falling edge of Write Transfer Active Status (after retrieving CRC status).</p> <p>0 - No Block Gap Event 1 - Transaction stopped at Block Gap</p>

Bits	Name	Attr	Reset	Description
1	transfer_complete	RW	0	<p>Transfer Complete This bit is set when a read / write transaction is completed.</p> <p>Read Transaction: This bit is set at the falling edge of Read Transfer Active Status. There are two conditions under which the Interrupt is generated. The first is when a data transfer is completed as specified by data length; i.e., after the last data has been read to the Host System. The second is when data has stopped at the block gap and data transfer is completed by setting the Block Gap Control register (SD_hpbw_control) field stop_at_block_gap_request; i.e., after valid data has been read to the Host System.</p> <p>Write Transaction: This bit is set at the falling edge of the DAT Line Active Status. There are two conditions under which the Interrupt is generated. The first is when the last data has been written to the card (specified by data length) and the Busy signal is released. The second is when data transfers have been stopped at the block gap by setting the Block Gap Control register (SD_hpbw_control) field stop_at_block_gap_request and data transfers complete; i.e., after valid data is written to the SD card and the Busy signal is released.</p> <p>Command with Busy: This bit is set when Busy is de-asserted. Refer to DAT Line Active (dat_line_active) and Command Inhibit (DAT) (command_inhibit_dat) fields in the Present State register (SD_present_state).</p> <p>As shown in the table below, Transfer Complete has a higher priority than the SD_interrupt_status Data Timeout Error (data_timeout_error). If both bits are set to 1, execution of a command can be considered to be completed.</p> <p>0 - No Data Transfer Complete 1 - Data Transfer Complete</p>
0	command_complete	RW	0	<p>Command Complete This bit is set when the end-bit of the command response is retrieved (Except Auto CMD12). Refer to Command Inhibit (CMD) (command_inhibit_cmd) in the Present State register (SD_present_state).</p> <p>Note: The SD_interrupt_status register Command Timeout Error (command_timeout_error) has a higher priority than command_complete. If both are set to 1, it can be assumed that the response was not received correctly.</p> <p>0 - No Command Complete 1 - Command Complete</p>

Table 13-55. SD Controller Normal Interrupt Status / Error Interrupt Status Register.

Transfer Complete <code>transfer_complete</code>	Data Timeout Error <code>data_timeout_error</code>	Meaning of the Status
0	0	Interrupted by Another Factor
0	1	Timeout occurred during transfer
1		Data Transfer Complete

Table 13-56. Normal Interrupt Status: Relation Between Transfer Complete and Data Timeout Error.

Command Complete <code>command_complete</code>	Command Timeout Error <code>command_timeout_error</code>	Meaning of the Status
0	0	Interrupted by Another Factor
	1	Response not received within 64 <code>SD[N].CLK</code> cycles
1	0	Response Received

Table 13-57. Normal Interrupt Status: Relation Between Command Complete and Command Timeout Error.

Command CRC Error <code>command_crc_error</code>	Command Timeout Error <code>command_timeout_error</code>	Kind of Error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD Line Conflict

Table 13-58. Error Interrupt Status: Relation Between Command CRC Error and Command Timeout Error.

13.5.7.11 SD_interrupt_status_enable Register

For the following, setting a bit to 1 enables the interrupt status.

Bits	Name	Attr	Reset	Description
31:28	<code>vendor_specific_error_status_enable</code>	RW	0	Vendor-Specific Error Status Enable 0 - Masked 1 - Enabled
27:26				Reserved
25	<code>adma_error_status_enable</code>	RW	0	ADMA Error Status Enable 0 - Masked 1 - Enabled
24	<code>auto_cmd12_error_status_enable</code>	RW	0	Auto CMD12 Error Status Enable 0 - Masked 1 - Enabled
23	<code>current_limit_error_status_enable</code>	RW	0	Current Limit Error Status Enable 0 - Masked 1 - Enabled
22	<code>data_end_bit_error_status_enable</code>	RW	0	Data End-Bit Error Status Enable 0 - Masked 1 - Enabled
21	<code>data_crc_error_status_enable</code>	RW	0	Data CRC Error Status Enable 0 - Masked 1 - Enabled

Bits	Name	Attr	Reset	Description
20	data_timeout_error_status_enable	RW	0	Data Timeout Error Status Enable 0 - Masked 1 - Enabled
19	command_index_error_status_enable	RW	0	Command Index Error Status Enable 0 - Masked 1 - Enabled
18	command_end_bit_error_status_enable	RW	0	Command End-Bit Error Status Enable 0 - Masked 1 - Enabled
17	command_crc_error_status_enable	RW	0	Command CRC Error Status Enable 0 - Masked 1 - Enabled
16	command_timeout_error_status_enable	RW	0	Command Timeout Error Status Enable 0 - Masked 1 - Enabled
15	fixed_to_0	R	0	Fixed to 0 The Host Controller will control error Interrupts using the Error Interrupt Status Enable register (SD_interrupt_status_enable).
14:9				Reserved
8	card_interrupt_status_enable	RW	0	Card Interrupt Status Enable If this bit is set to 0, the Host Controller will clear the interrupt request to the System. The Card Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The Host Driver should clear card_interrupt_status_enable before servicing the Card Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. 0 - Masked 1 - Enabled
7	card_removal_status_enable	RW	0	Card Removal Status Enable 0 - Masked 1 - Enabled
6	card_insertion_status_enable	RW	#	Card Insertion Status Enable 0 - Masked 1 - Enabled Note that this bit is used for eMMC boot operation, and the reset value depends on the MMC_BOOT configuration (see Section 2.3): For MMC_BOOT = 0, the reset value is 0 For MMC_BOOT = 1, the reset value is 1
5	buffer_read_ready_status_enable	RW	0	Buffer Read Ready Status Enable 0 - Masked 1 - Enabled
4	buffer_write_ready_status_enable	RW	0	Buffer Write Ready Status Enable 0 - Masked 1 - Enabled
3	dma_interrupt_status_enable	RW	0	DMA Interrupt Status Enable 0 - Masked 1 - Enabled

Bits	Name	Attr	Reset	Description
2	<code>block_gap_event_status_enable</code>	RW	0	Block Gap Event Status Enable 0 - Masked 1 - Enabled
1	<code>transfer_complete_status_enable</code>	RW	0	Transfer Complete Status Enable 0 - Masked 1 - Enabled
0	<code>command_complete_status_enable</code>	RW	0	Command Complete Status Enable 0 - Masked 1 - Enabled

Table 13-59. Normal Interrupt Status Enable / Error Interrupt Status Enable Register.

Notes:

- With Normal Interrupt Status Enable, the Host Controller may sample the card Interrupt signal during the interrupt period and may hold its value in the flip-flop. If Card Interrupt Status Enable is set to 0, the Host Controller clears all internal signals regarding Card Interrupt.
- For Error Interrupt Status Enable be aware that to Detect a CMD Line conflict, the Host Driver must set both the Command Timeout Error Status Enable and the Command CRC Error Status Enable bits to 1.

13.5.7.12 SD_normal_interrupt_signal_enable Register

This register is used to select which status is indicated to the Host System for a given interrupt. These status bits all share the same 1-bit interrupt line. Setting one of these bits to 1 enables interrupt generation.

Bits	Name	Attr	Reset	Description
31:28	<code>vendor_specific_error_signal_enable</code>	RW	0	Vendor-Specific Error Signal Enable 0 - Masked 1 - Enabled
27:26				Reserved
25	<code>adma_error_signal_enable</code>	RW	0	ADMA Error Signal Enable 0 - Masked 1 - Enabled
24	<code>auto_cmd12_error_signal_enable</code>	RW	0	Auto CMD12 Error Signal Enable 0 - Masked 1 - Enabled
23	<code>current_limit_error_signal_enable</code>	RW	0	Current Limit Error Signal Enable 0 - Masked 1 - Enabled
22	<code>data_end_bit_error_signal_enable</code>	RW	0	Data End-Bit Error Signal Enable 0 - Masked 1 - Enabled
21	<code>data_crc_error_signal_enable</code>	RW	0	Data CRC Error Signal Enable 0 - Masked 1 - Enabled
20	<code>data_timeout_error_signal_enable</code>	RW	0	Data Timeout Error Signal Enable 0 - Masked 1 - Enabled

Bits	Name	Attr	Reset	Description
19	command_index_error_signal_enable	RW	0	Command Index Error Signal Enable 0 - Masked 1 - Enabled
18	command_end_bit_error_signal_enable	RW	0	Command End-Bit Error Signal Enable 0 - Masked 1 - Enabled
17	command_crc_error_signal_enable	RW	0	Command CRC Error Signal Enable 0 - Masked 1 - Enabled
16	command_timeout_error_signal_enable	RW	0	Command Timeout Error Signal Enable 0 - Masked 1 - Enabled
15	fixed_to_0	R	0	Fixed to 0 The Host Driver will control error Interrupts using the Error Interrupt Signal Enable register (SD_interrupt_signal_enable).
14:9				Reserved
8	card_interrupt_signal_enable	RW	0	Card Interrupt Signal Enable 0 - Masked 1 - Enabled
7	card_removal_signal_enable	RW	0	Card Removal Signal Enable 0 - Masked 1 - Enabled
6	card_insertion_signal_enable	RW	0	Card Insertion Signal Enable 0 - Masked 1 - Enabled
5	buffer_read_ready_signal_enable	RW	0	Buffer Read Ready Signal Enable 0 - Masked 1 - Enabled
4	buffer_write_ready_signal_enable	RW	0	Buffer Write Ready Signal Enable 0 - Masked 1 - Enabled
3	dma_interrupt_signal_enable	RW	0	DMA Interrupt Signal Enable 0 - Masked 1 - Enabled
2	block_gap_event_signal_enable	RW	0	Block Gap Event Signal Enable 0 - Masked 1 - Enabled
1	transfer_complete_signal_enable	RW	0	Transfer Complete Signal Enable 0 - Masked 1 - Enabled
0	command_complete_signal_enable	RW	0	Command Complete Signal Enable 0 - Masked 1 - Enabled

Table 13-60. SD Controller Normal Interrupt Signal Enable / Error Interrupt Signal Enable Register.

13.5.7.13 SD_auto_cmd12_error_status Register

When Auto CMD12 Error Status is set, the Host Driver will check this register to identify the error type. This register is valid only when the Auto CMD12 Error is set.

Bits	Name	Attr	Reset	Description
31				Reserved for Preset Value Enabled
30	asynchronous_interrupt_enable	RW	1	<p>This bit can be set to 1 if a card supports asynchronous interrupts and Asynchronous Interrupt Support is set to 1 in the Capabilities register. Asynchronous interrupt is effective when DAT[1] interrupt is used in 4-bit SD mode (and the Interrupt Pin Select is set to 0 in the Shared Bus Control register).</p> <p>If this bit is set to 1, the Host Driver is able to stop the SDCLK during an asynchronous interrupt period to save power. During this period, the Host Controller continues to deliver the Card Interrupt to the host when it is asserted by the Card.</p> <p>1 - Enable 0 - Disable</p>
29	cmd23_disable	RW	0	1: Disable SDXC CMD23 (for correct ACMD23 sequences; [CMD55+CMD23] when operating bus-muxed cards) 0: Enable SDXC CMD23
28-24				Reserved
23				Reserved for Sampling Clock Select
22				Reserved for Execute Tuning
21:20				Reserved for Driver Strength Select
19	1.8v_signaling_enable	RW	0	<p>This bit controls the voltage regulator for the I/O cell. Note that 3.3 V is supplied to the card regardless of signaling voltage. Setting this bit from 0 to 1 initiates a change in signal voltage from 3.3 V to 1.8 V. The 1.8-V regulator output will be stable within 5 ms. The Host Controller clears this bit if switching to 1.8-V signaling fails.</p> <p>Clearing this bit from 1 to 0 initiates a change in signal voltage from 1.8 V to 3.3 V. The 3.3-V regulator output will be stable within 5 ms.</p> <p>The Host Driver can set this bit to 1 when the Host Controller supports 1.8-V signaling (One of the support bits is set to 1: SDR50, SDR104, or DDR50 in the Capabilities register) and the card or device supports UHS-1 (S18A-1. Refer to Bus Signal Voltage Switch Sequence in the <i>Physical Layer Specification Version 3.0x</i>).</p> <p>1 - 1.8-V Signaling 0 - 3.3-V Signaling</p>
18	ddr_mode	RW	ddr_boot_input_pin	1 - ddr_mode 0 - sdr_mode
17:16				Reserved for UHS Mode Select
15:8				Reserved

Bits	Name	Attr	Reset	Description
7	command_not_issued_by_auto_cmds12_error	R	0	Command Not Issued By Auto CMD12 Error Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 error (D04 - D01) in this register. 0 - No Error 1- Not Issued
6:5				Reserved
4	auto_cmd12_index_error	R	0	Auto CMD12 Index Error Occurs if the Command Index occurs in response to a command. 0 - No Error 1 Error
3	auto_cmd12_end_bit_error	R	0	Auto CMD12 End Bit Error Occurs when detecting that the end bit of command response is 0. 0 - No Error 1 - End Bit Error Generated
2	auto_cmd12_crc_error	R	0	Auto CMD12 CRC Error Occurs when detecting a CRC error in the command response. 0 - No Error 1 - CRC Error Generated
1	auto_cmd12_timeout_error	R	0	Auto CMD12 Timeout Error Occurs if no response is returned within 64 SD[N]_CLK cycles from the end bit of the command. If this bit is set to 1, the other error status bits (D04 - D02) are meaningless. 0 - No Error 1 - Timeout
0	auto_cmd12_not_executed	R	0	Auto CMD12 not Executed If memory multiple block data transfer does not start due to command error, this bit is not set because it is not necessary to issue Auto CMD12. Setting this bit to 1 means the Host Controller cannot issue Auto CMD12 to stop memory multiple block transfer due to some error. If this bit is set to 1, other error status bits (D04 - D01) are meaningless. 0 - Executed 1 - Not Executed

Table 13-61. SD Controller Auto CMD12 Error Status and Host Control 2 Register.

Auto CMD12 CRC Error	Auto CMD12 Timeout Error	Error Type
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD Line Conflict

Table 13-62. Relationship Between Auto CMD12 CRC Error and Auto CMD12 Timeout Error.

13.5.7.14 SD_capabilities Register

This register provides the Host Driver with information specific to the Host Controller implementation. The Host Controller may implement these values as fixed or loaded from Flash memory during power-on initialization.

Bits	Name	Attr	Reset	Description
63:32				Reserved
31:30	slot_type	RW	0	<p>This field indicates usage of a slot by a specific Host System. (A Host Controller register set is defined per-slot.). Embedded Slot for One Device (01) indicates that only one non-removable device is connected to a SD bus slot.</p> <p>00 - Removable Card Slot 01 - Embedded Slot for One Device 11 - Reserved</p> <p>Note: The register type is HWI in [5], but is RW here.</p>
29	asynchronous_interrupt_support	HWI	1	<p>1 - Asynchronous Interrupt Supported 0 - Asynchronous Interrupt Not Supported</p>
28				Reserved
27	interrupt_mode	HWI	1	<p>Interrupt mode 0 - Not Supported 1 - Supported</p>
26	voltage_support_1.8v	HWI	1	<p>Voltage Support 1.8 V 0 - 1.8 V Not Supported 1 - 1.8 V Supported</p> <p>Note that the Host Driver sets the Power Control register (SD_hpbw_control) field sd_bus_voltage_select according to support bits[26:24]. If multiple voltages are supported, select the lowest usable voltage by comparing the OCR value from the card.</p>
25	voltage_support_3.0v	HWI	0	<p>Voltage Support 3.0 V 0 - 3.0 V Not Supported 1 - 3.0 V Supported</p> <p>Note that the Host Driver sets the Power Control register (SD_hpbw_control) field sd_bus_voltage_select according to support bits[26:24]. If multiple voltages are supported, select the lowest usable voltage by comparing the OCR value from the card.</p>
24	voltage_support_3.3v	HWI	1	<p>Voltage Support 3.3 V 0 - 3.3 V Not Supported 1 - 3.3 V Supported</p> <p>Note that the Host Driver sets the Power Control register (SD_hpbw_control) field sd_bus_voltage_select according to support bits[26:24]. If multiple voltages are supported, select the lowest usable voltage by comparing the OCR value from the card.</p>
23	suspend_resume_support	HWI	1	<p>Suspend / Resume Support This bit indicates whether the Host Controller supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanisms are not supported and the Host Driver will not issue either Suspend or Resume commands. 0 - Not Supported 1 - Supported</p>
22	dma_support	HWI	1	<p>DMA Support This bit indicates whether the Host Controller is capable of using DMA to transfer data between system memory and the Host Controller directly. 0 - DMA Not Supported 1 - DMA Supported</p>

Bits	Name	Attr	Reset	Description
21	high_speed_support	HWI	1	High Speed Support This bit indicates whether the Host Controller and the Host System support High-Speed mode. 0 - High Speed Not Supported 1 - High Speed Supported
20		R	0	Reserved
19	adma_support	HWI	1	ADMA Support This bit indicates whether the Host Controller is ADMA-capable. 0 - ADMA Not Supported 1 - ADMA Supported
18		R	0	Reserved
17:16	max_block_length	HWI	1	Max Block Length This value specifies the maximum block size that the Host Driver can read and write to the buffer in the Host Controller. The buffer will transfer this block size without wait cycles. Three sizes can be defined as indicated below. 0x0 - 512 bytes 0x1 - 1024 bytes 0x2 - 2048 bytes 0x3 - Reserved
15:0				Reserved

Table 13-63. SD Controller Capabilities Register.

13.5.7.15 SD_max_current_capabilities Register

Bits	Name	Attr	Reset	Description
63:24				Reserved
23:16	maximum_current_for_1.8v	R	0	Maximum Current for 1.8 V
15:8	maximum_current_for_3.0v	R	0	Maximum Current for 3.0 V
7:0	maximum_current_for_3.3v	R	0x01	Maximum Current for 3.3 V

Table 13-64. SD Controller Maximum Current Capabilities Register.

The maximum current values are defined as follows:

Register Value	Current Value
0	Retrieve information via another method
1	4 mA
2	8 mA
3	12 mA
255	1020 mA

Table 13-65. SD Controller Definitions for Maximum Current Value.
13.5.7.16 SD_adma_error_status Register

When an ADMA Error Interrupt occurs, the ADMA Error State field (**adma_error_state**) in this register holds the ADMA state, and the ADMA System Address Register (**SD_adma_system_address**) holds the address around the error descriptor.

Bits	Name	Attr	Reset	Description
31:6				Reserved
5	adma_byte_error	R	0	ADMA Byte Error Total data length cannot be divided by the block length. 0 – No error 1 – Error This bit is reserved in the SDA specification.
4	adma_block_error	R	0	ADMA Block Error The total data length specified by the Descriptor table differs from that specified by Block Count and Block Length. This bit is reserved in the SDA specification.
3	invalid_adma_descriptor	R	0	Invalid ADMA Descriptor This bit is set after fetching an ADMA descriptor to Valid = 0. 0 – No error 1 – Error This bit is reserved in the SDA specification.

Bits	Name	Attr	Reset	Description
2	adma_length_mismatch_error	R	0	<p>ADMA Length Mismatch Error This error occurs under two conditions: (1) The total data length specified by the Descriptor table differs from that specified by the Block Count and Block Length (bit[4]=1); (2) Total data length cannot be divided by the block length (bit[5]=1)</p> <p>bit[2] = bit[4] bit[5].</p> <p>0 – No error 1 – Error</p>
1:0	adma_error_state	R	0	<p>ADMA Error State This field indicates the state of ADMA when the error occurred. This field never takes on value 0x2, since the ADMA never stops in this state.</p> <p>0x0 - Error state is ST_STOP (Stop DMA) and SYS_SDR register points to the next error descriptor 0x1 - Error state is ST_FDS (Fetch Descriptor) and SYS_SDR points to the error descriptor 0x2 - Not used 0x3 - Error state is ST_TFR (Transfer Data) and SYS_SDR points to the next error descriptor</p>

Table 13-66. SD Controller ADMA Error Status Register.

13.5.7.17 SD_adma_system_address Register

This register contains the physical descriptor address used for ADMA data transfer.

Bits	Name	Attr	Reset	Description
31:0	adma_system_address	RW	0	<p>This register holds the byte address of the executing command from the Descriptor table. At the start of an ADMA transaction, the Host Driver should set the start address of the Descriptor table. The ADMA increments the register address, which points to the next line, when fetching every Descriptor line. When the ADMA Error Interrupt is generated, this register holds the valid Descriptor address depending on the ADMA state. The Host Driver should program the Descriptor Table on a 32-bit boundary and set a 32-bit boundary address to this Register. The ADMA ignores the lower two bits of this register and assumes it to be 0x0.</p>

Table 13-67. SD Controller ADMA System Address Register.

13.5.7.18 SD_boot_control Register

Bits	Name	Attr	Reset	Description
31:17				Reserved
16	rst_fio_boot_en	RW	0	FIO Boot Enable on SD Software Reset This bit is used for eMMC boot operation (Section 2.3). If this bit is 1 during an SD software reset, an RST_FIO_BOOT is asserted and an IO_RESET_MISC of the original FIOS also will be asserted to enter boot mode.
15:12				Reserved
11:0	boot_size	RW	0x800	Boot Code Length in Bytes This bit is used for eMMC boot operation (Section 2.3). When the size of received boot codes reaches this value, eMMC boot logic will terminate the boot operation. As FIOS FIFO is 64 bits in width, the size of this field should be a multiple of 8 bytes:.

Table 13-68. SD Controller Boot Control Register.
13.5.7.19 SD_boot_status Register

Bits	Name	Attr	Reset	Description
31:25				Reserved
24	boot_end_alt	R	0	Boot End Alt This bit is used for eMMC boot operation (Section 2.3). When the fetched boot code reaches the value of the SD_boot_control field boot_size , this bit will be asserted. Both hardware and software resets clear this bit.
23:17				Reserved
16	boot_end	R_SC	0	Boot End This bit is used for eMMC boot operation (Section 2.3). When the fetched boot code reaches boot_size , this bit will be asserted. Software reset does not clear this bit.
15:1				Reserved
0	rdy_to_boot	R	0	Ready to Boot This bit is used for eMMC boot operation (Section 2.3). This bit is deasserted after a reset operation and is asserted 74 cycles after a power-on or a reset operation (including the SD command CMD0 of value 0xF0F0F0F0).

Table 13-69. SD Controller Boot Status Register.

13.5.7.20 SD_block_count_cmd23 Register

Bits	Name	Attr	Reset	Description
31:0	block_cnt	R	0	<p>Block Count for CMD23</p> <p>This register is updated by the Argument register (SD_argument) when the command index is written as 23. This register is independent from the Block Count register SD_block_size_and_count (offset 0x000).</p> <p>If no CMD23 is executed before CMD18/25, the effective block count is the programmed value at SD_block_size_and_count (offset 0x006), and this register (offset 0x078) is ignored.</p>

Table 13-70. SD Controller Block Count (CMD23) Register.

13.5.7.21 SD_voltage_switch Register

This register is used to detect activity on the bus in preparation for switching the voltage.

Bits	Name	Attr	Reset	Description
31:17				Reserved
16	cmd_stat	RW	0	<p>CMD Status</p> <p>Asserted when the CMD signal driven by the card is high for 1 cycle (this bit has a value of 1). Cleared when written to 0 by the software.</p> <p>This field can be used in operating the voltage switch. Contact an Ambarella representative for details.</p>
15:4				Reserved
3:0	dat_stat	RW	0	<p>DAT Status</p> <p>Asserted when DAT[3:0] signals driven by the card are high for 1 cycle (this bit has a value of 1). Cleared when written to 0 by the software.</p> <p>This field can be used in operating the voltage switch. Contact an Ambarella representative for details.</p>

Table 13-71. SD Controller Voltage Switch Register.

13.5.7.22 SD_read_latency_control Register

Bits	Name	Attr	Reset	Description
31:16				Reserved
15	disable_hs_cmd_conf	RW	0	0x0: Enable command conflict check in high-speed mode 0x1: Disable command conflict check in high-speed mode
14:12	d_cmd_out_hs_sel	RW	0	Selects which tap should be used to check for command conflict in high-speed mode.
11	disable_ds_cmd_conf	RW	0	0x0: Enable command conflict check in default-speed mode 0x1: Disable command conflict check in default-speed mode
10:8	d_cmd_out_ds_sel	RW	0	Selects which tap should be used to check for command conflict in default-speed mode.

Bits	Name	Attr	Reset	Description
7:4	trn_to_crc_status	RW	0	<p>Determines the cycle count between write data and CRC status. The SD controller will begin to check the start bit of CRC status (trn_to_crc_status + 1) cycles after the end bit of the write data.</p> <p>0x0 - 1 cycle 0x1 - 2 cycles ----- 0xF - 16 cycles</p>
3:0	ncr_reg	RW	0	<p>Determines the cycle count between command and response. The SD controller will begin to check the start bit of the received response (ncr_reg + 1) cycles after the end bit of the transmitted command.</p> <p>0x0 - 1 cycle 0x1 - 2 cycles ----- 0xF - 16 cycles</p>

Table 13-72. Read Latency Control Register.

13.5.7.23 SD_slot_interrupt_and_version Register

Bits	Name	Attr	Reset	Description
31:24	vendor_version_number	R	0	<p>Vendor Version Number</p> <p>This register is reserved for the vendor version number. The Host Driver should not use this register.</p>
23:16	specification_version_number	R	0	<p>Specification Version Number</p> <p>This register indicates the Host Controller Specification Version. The Upper and Lower 4 bits specify the version.</p> <p>0 - SD Host Specification version 1.0 Not 0 - Reserved</p>
15:8				Reserved
7:0	interrupt_signal_for_each_slot	R	0	<p>Interrupt Signal for Each Slot</p> <p>These status bits indicate the logical OR of the Interrupt signal and Wakeup signal for each slot. A maximum of eight slots can be defined. If one interrupt signal is associated with multiple slots, the Host Driver can identify which interrupt is generated by reading these status bits. A power-on reset or a Software Reset For All, will clear these bits and deassert the interrupt signal (zero).</p> <p>Bit 00 - Slot 1 Bit 01 - Slot 2 Bit 02 - Slot 3 ----- Bit 07 - Slot 8</p>

Table 13-73. SD Controller Slot Interrupt Status and Controller Version Register.

13.5.8 SD Controllers: ADMA Block Diagram and Descriptor Programming

13.5.8.1 ADMA: Descriptor Overview

The Descriptor Table is created by the Host Driver in system memory. Each descriptor line (one executable unit) consists of an address, length and attribute field:

- The attribute field specifies the operation of the descriptor line and is used to control the descriptor.
- There are three action symbols associated with the Attribute field: (1) **Nop** (No operation) skips the current descriptor line and fetches the next one; (2) **Tran** transfers data designated by an address and length field; and (3) **Link** connects two separated descriptors. For further information on these symbols, see [Table 13-76](#) below. For Attribute[3:0] information, refer to [Table 13-77](#).
- The address field for the link points to the next Descriptor Table.
- The address field is set on the 32-bit boundary (the lower 2 bits always are set to 0).
- One descriptor line consumes 64-bit (8-byte) memory space.
- For data length information see [Table 13-75](#) below.

Address			Length			Reserved			Attribute					
63	32	31	16	15	0	06	05	04	03	02	01	00		
32-bit Address			16-bit Length			10'b0			Act2	Act1	Word	Int	End	Valid

Table 13-74. SD Controllers: ADMA Descriptor Table.

Length Field	Data Length	
	Word = 0	Word = 1
0x0000	65536 bytes	65536 words = 262144 bytes
0x0001	1 byte	1 word = 4 bytes
0x0002	2 bytes	2 words = 8 bytes
0xFFFF	65535 bytes	65535 words = 262140 bytes

Table 13-75. SD Controllers: ADMA Descriptor Data Length Fields.

Attribute		Symbol	Comment	Operation	
05	04			0	1
Act2	Act1				
0	0	Nop	No operation	Do not execute current line and go to next line	
0	1	Rsv	Reserved	Do not execute current line and go to next line	
1	0	Tran	Transfer Data	Transfer data of one descriptor line	
1	1	Link	Link Descriptor	Link to another descriptor	

Table 13-76. SD Controllers: ADMA Descriptor Attribute[5:4] fields Act2 and Act1.

Attribute		Description
Bit	Name	
03	Word	0 - Indicates bytes are 16 bits in length, and the maximum data length that can be transferred in one descriptor is 65536 bytes = 64 KB 1 - Indicates words are 16 bits in length, the maximum data length that can be transferred in one descriptor is 65536 words = 64 K words = 256 KB.
02	Int	0 - Not triggered 1 - Generates DMA Interrupt when the operation of the descriptor line is completed
01	End	0 - Not defined 1 - Indicates end of the descriptor. The Transfer Complete Interrupt is generated when the operation of the descriptor line is completed.
00	Valid	0 - Generate ADMA Error Interrupt and stop ADMA to prevent runaways 1 - Indicates this line of descriptor is effective

Table 13-77. SD Controllers: ADMA Descriptor Attribute[3:0] fields Word, Int, End and Valid.

Below is a block diagram of the A12 ADMA engine, including descriptor table detail.

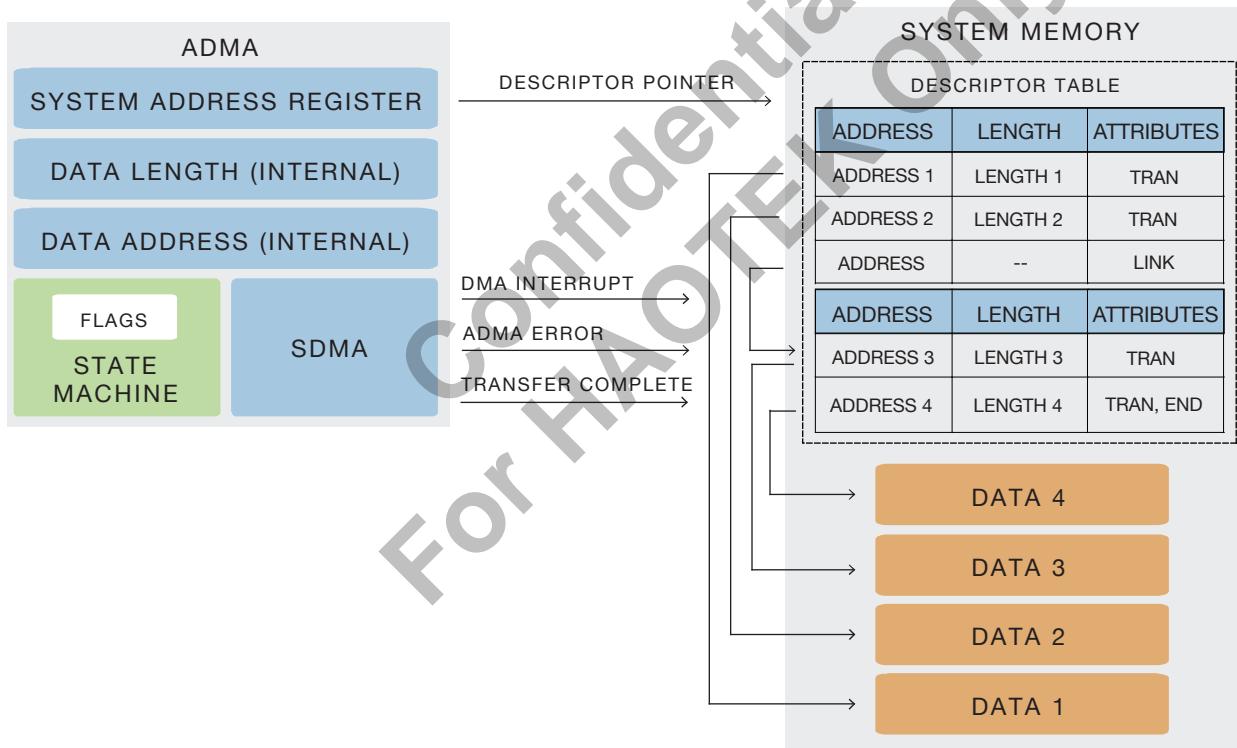


Figure 13-10. SD Controllers: Block Diagram Showing ADMA Transfer and Descriptor Table.

The ADMA engine includes the SDMA engine, a State Machine and Register circuits. The ADMA engine uses the Advanced DMA System Address register (offset 0x58) for the descriptor pointer rather than the SDMA System Address Register (offset 0x00).

Writing to the Command register triggers an ADMA transfer. The ADMA engine fetches one descriptor line and executes it. This procedure is repeated until the end-of-descriptor is located (End=1 in the attribute field).

13.5.8.2 ADMA: Descriptor Programming

The figure below illustrates a typical ADMA descriptor program. The data area is sliced into various lengths, and each slice exists somewhere in system memory. The Descriptor Table consists of the address, length and attributes described by the Host Driver. The data is transferred slice-by-slice as programmed in the Descriptor Table.

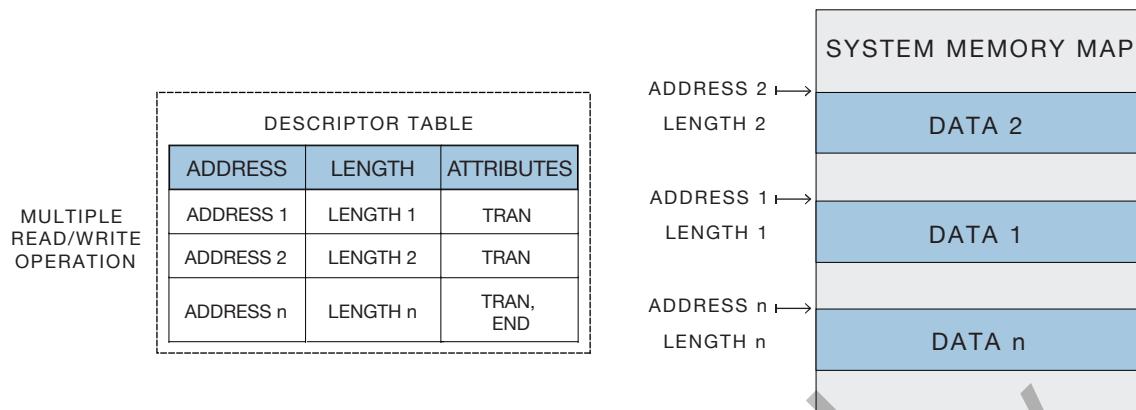


Figure 13-11. SD Controllers: ADMA Descriptor Table and Data Transfer Programming.

Notes:

1. The minimum data length (unit) in each descriptor line is 4 bytes when the attribute Word=0.
2. The minimum address (unit) is 4 bytes.
3. The maximum data length per descriptor line is 64 KB or 256 KB depending on the Word attribute.
4. Total Length = Length 1 + Length 2 + Length 3 + ... + Length n = a multiple of Block Size. Note that the ADMA transfer may fail to terminate if the total length of a descriptor is not a multiple of block size.

When operating the ADMA engine, set the Transfer Mode register (**SD_transfer_mode_and_command**) Block Count Enable bit (**block_count_enable**) to 1. The Block Count field (**block_count**) of the same register is a 16/32-bit register for non-CMD23/CMD23, and it limits blocks transferred to a maximum of 65535 / 0xFFFFFFFF.

The ADMA limits block transfer to 65535 / 0xFFFFFFFF for non-CMD23/CMD23 operation. Block Count registers (**SD_block_size_and_count**/**SD_block_count_cmd23**) specify the entire block count of an ADMA operation, which can include multiple descriptor lines. In this case of multiple descriptor lines, the total length of the Descriptor Table is related to block size and block count.

14. INTERRUPT CONTROLLER (VIC)

This chapter provides register programming information for the A12 Vector Interrupt Controller (VIC). The chapter is organized as follows:

- [\(Section 14.1\) Interrupts: Overview](#)
- [\(Section 14.2\) Interrupts: VIC Interrupt Vectors](#)
- [\(Section 14.3\) Interrupts: VIC Detail](#)
- [\(Section 14.4\) Interrupts: VIC Registers](#)

14.1 Interrupts: Overview

The diagram below illustrates the Vector Interrupt Controller (VIC) and its relationship to the Cortex-A9 processor.

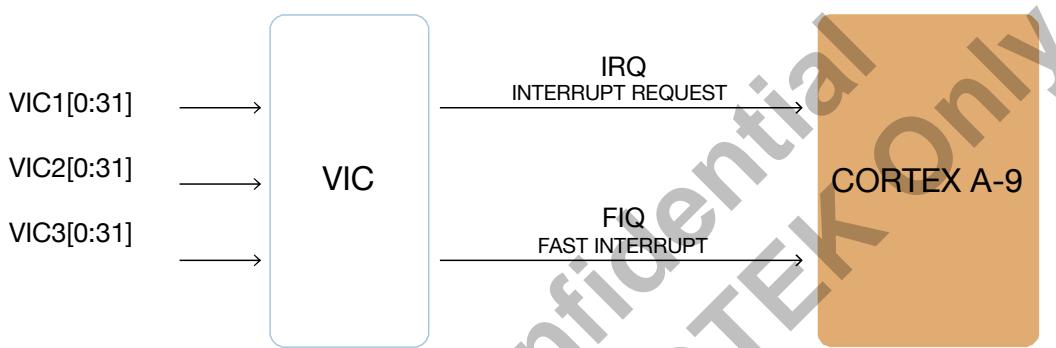


Figure 14-1. Diagram of the A12 Vector Interrupt Controller (VIC).

i Note:

A number of VIC-related registers and signals may not be applicable to specific A12 or A12m SoCs. Please consult the relevant chip datasheet for a comprehensive list of supported features and specifications.

The VIC is a slave unit and provides a software interface to the A12 interrupt system. It supports the two interrupt modes present on the Cortex-A9 system:

1. Fast Interrupt Request (FIQ) for fast, low-latency interrupt handling, and
2. Interrupt Request (IRQ) for normal-priority, general interrupts.

The VIC collects interrupt signals from various modules and transmits IRQs and/or FIQs to the ARM core. The properties of all interrupt signals can be configured by the VIC registers which are mapped to the AHB. A VIC instance supports up to 32 interrupt sources, listed in [Section 14.2](#) below. Software must initialize all relevant registers appropriately, including edge- or level-sensitivity as shown.

14.2 Interrupts: VIC Interrupt Vectors

There are three instances of the VIC on the A12 SoC. The first instance (VIC 1) is located at AHB base address 0xE000.3000. The second instance (VIC 2) is located at AHB base address 0xE001.0000. The third instance (VIC 3) is located at AHB base address 0xE001.C000.

14.2.1 VIC Instance 1

Bit Position	Source	Sensitivity
31	Reserved	
30	GPIO 2	Level
29	GPIO 3	Level
28	USB PHY0 VBUS detection status change	Level
27	Ethernet GMAC Subsystem SBD_INTR_O1	Level
26	Motor Interrupt	Level
25	UART 1	Level
24	SD0 Controller Card Detect (RCT)	Both edges
23	SD1 Controller Card Detect (RCT)	Both edges
22	InfraRed Remote	Level
21	Watchdog Timer (WDT)	Edge
20	SD2 Controller	Level
19	I2C/IDC0 Controller	Level
18	SD0 Controller	Level
17	Flash DMA Done	Level
16	Flash Command Done	Level
15	DMA	Level
14	Timer 3	Edge
13	Timer 2	Edge
12	Timer 1	Edge
11	GPIO 1	Level
10	GPIO 0	Level
9	UART 0	Level
8	Primary I2S RX	Level
7	Primary I2S TX	Level
6	SD2 Controller Card Detect (RCT)	Both edges

Bit Position	Source	Sensitivity
5	USB Charge Detect	Both edges
4	USB	Level
3	VDSP Coding 0	Edge
2	VDSP Coding VIN	Edge
1	VDSP Coding VOUT A	Edge
0	USB VBus Connect	Level

Table 14-1. Interrupt Sources for VIC Instance 1.

14.2.2 VIC Instance 2

Bit Position	Source	Sensitivity
31	Timer 8	Edge
30	Timer 7	Edge
29	Timer 6	Edge
28	Timer 5	Edge
27	Timer 4	Edge
26	VDSP PiP Coding	Edge
25	IDSP PiP Delayed VSync	Edge
24	IDSP PiP Last Pixel	Edge
23	IDSP PiP Master VSync	Edge
22	IDSP PiP Start of Frame	Edge
21	IDSP PiP VSync	Edge
20	SD1 Controller	Level
19	IDC1 Controller	Edge
18	GDMA	Edge
17	Reserved	
16	Reserved	
15	Reserved	
14	VDSP Coding VOUT B	Edge
13	NOR SPI	Edge
12	USB OHCI Interrupt	Level
11	VOUT A ARM	Edge
10	VOUT B ARM	Edge
9	FIOS ECC	Level
8	HDMI Tx	Level
7	USB EHCI Interrupt	Level
6	SSI/SPI Slave	Level
5	SSI/SPI1 Master	Level
4	IDC2 Controller	Level
3	SSI/SPI0 Master	Level
2	ADC Level Change	Level
1	DMA FIOS Channel 0	Level
0	Ethernet GMAC Core PMT_INTR_O1	Level

Table 14-2. Interrupt Sources for VIC Instance 2.

14.2.3 VIC Instance 3

Bit Position	Source	Sensitivity
31	L2CC ECNTRINTR	Level
30	L2CC SLVERRINTR	Level
29	L2CC DECERRINTR	Level
28	PMUIRQ	Level
27	USB Digital ID Change	Level
26	Reserved	
25	SHA1_INTR	Level
24	AES_INTR	Level
23	DES_INTR	Level
22	MD5_INTR	Level
21	L2CCINTR	Level
20	AXI software IRQ[1]	Level
19	AXI software IRQ[0]	Level
18:5	Software IRQ[13:0]	Level
4	IDSP VIN Last Pixel	Edge
3	IDSP VIN Delayed VSync	Edge
2	IDSP VIN Start of Frame	Edge
1	IDSP VIN VSync	Edge
0	IDSP VIN Master VSync	Edge

Table 14-3. Interrupt Sources for VIC Instance 3.

14.3 Interrupts: VIC Detail

14.3.1 VIC CPU Offload Functions and Priorities

The Interrupt Enable register (**VIC_int_enable**) is a 32-bit vector where each bit represents an interrupt source. Software is forced to post-process the register value prior to enabling a particular interrupt source. To reduce the processing required, the A12 chip includes registers such as the Interrupt Enable Integer register (**VIC_int_en_int**). Writing an integer value (0-31) to the **VIC_int_en_int** register sets the corresponding bit of **VIC_int_enable**. Similarly, there are separate registers for setting the interrupt to IRQ or FIQ, a software interrupt, and so on.

Additionally, the software must post-process the IRQ Status (**VIC_irq_status**) and FIQ Status (**VIC_fiq_status**) registers to determine which interrupt is pending. To minimize this type of post-processing effort, the A12 chip includes the Interrupt Pending register (**VIC_int_pending**). This register returns the lowest-priority interrupt of all outstanding interrupts. It is also possible to reprioritize the interrupt sources by enabling the Interrupt Reprioritize register (**VIC_int_re_prioritize**). The priority of each interrupt source is set by programming Interrupt Priority 0 - 5 registers (**VIC_int_priority[0-5]**).

Notes:

- No two interrupt sources should be assigned the same priority.

- Setting the priority register will affect the output of the Interrupt Pending register (**VIC_int_pending**).
- The priorities are for a particular VIC instance and can not be combined with other VIC instances.

14.3.2 VIC Detail: SD Card Detect Interrupt

Bits 24, 23, and 6 of the VIC Instance 1 registers pertain to SD card detects. The SD0 controller uses **SMIO_4** for card detection. VIC interrupt 24 is triggered when there is a state-change (either a rising edge or a falling edge) on the SD Card Detect pin **SMIO_4**. Similarly, the SD1 controller uses **SMIO_32** for SD Card Detect, and the SD2 controller uses **SC_A1**.

Note that when an SD device is connected, these interrupts are redundant as the SD controller is capable of detecting card insertion and removal and also generating a VIC interrupt when not masked. Software may therefore mask the VIC interrupts 24, 23, or 6 and instead use the interrupt provided by the SD Controller.

14.4 Interrupts: VIC Registers

14.4.1 VIC Registers: Map

Register Offset	Register Name	Description
0x00	VIC_irq_status	IRQ status
0x04	VIC_fiq_status	FIQ status
0x08	VIC_raw_intr	Raw Interrupt Status
0x0C	VIC_int_select	Interrupt Select
0x10	VIC_int_enable	Interrupt Enable
0x14	VIC_int_en_clear	Interrupt Enable Clear
0x18	VIC_soft_int	Software Interrupt
0x1C	VIC_soft_int_clear	Software Interrupt Clear
0x20	VIC_protection	Protection Enable
0x24	VIC_sense	Interrupt Sense
0x28	VIC_bothedge	Interrupt Both Edges
0x2C	VIC_event	Interrupt Event
0x30	VIC_int_ptr0_reg	Controls whether a specific VIC bit can be pointed to core #0. Must be set to all bits 1 when enabling VIC.
0x34		Reserved
0x38	VIC_edge_clr	Edge-Triggered Interrupt Clear
0x3C	VIC_int_select_int	Bit to Set In the Interrupt Select
0x40	VIC_int_select_clr_int	Bit to Clear In the Interrupt Select
0x44	VIC_int_en_int	Bit to Set In the Interrupt Enable
0x48	VIC_int_en_clr_int	Bit to Clear In the Interrupt Enable
0x4C	VIC_soft_int_int	Bit to Set In Software Interrupt
0x50	VIC_soft_int_clr_int	Bit to Clear In Software Interrupt
0x54	VIC_int_sense_int	Bit to Set In Sensitivity
0x58	VIC_int_sense_clr_int	Bit to Clear On the Sensitivity
0x5C	VIC_int_bothedge_int	Bit to Set In Bothedge
0x60	VIC_int_bothedge_clr_int	Bit to Clear In Bothedge

Register Offset	Register Name	Description
0x64	VIC_int_evt_int	Bit to Set In Event
0x68	VIC_int_evt_clr_int	Bit to Clear Event
0x6C	VIC_int_pending	Interrupt Pending
0x70	VIC_int_re_prioritize_en	Interrupt Reprioritize Enable
0x74	VIC_int_priority_0	Priority of Interrupt at Bit 0, 1, 2, 3, 4, 5
0x78	VIC_int_priority_1	Priority of Interrupt at Bit 6, 7, 8, 9, 10, 11
0x7C	VIC_int_priority_2	Priority of Interrupt at Bit 12, 13, 14, 15, 16, 17
0x80	VIC_int_priority_3	Priority of Interrupt at Bit 18, 19, 20, 21, 22, 23
0x84	VIC_int_priority_4	Priority of Interrupt at Bit 24, 25, 26, 27, 28, 29
0x88	VIC_int_priority_5	Priority of Interrupt at Bit 30, 31
0x8C	VIC_int_delay_en	Enable/Disable Delay on the Interrupt
0x90	VIC_int_delay	Amount of Delay Before Asserting the Interrupt

Table 14-4. APB Registers of the Vector Interrupt Controller (VIC).

14.4.2 VIC Registers: Detail

14.4.2.1 VIC_irq_status Register

Bits	Name	Attr	Reset	Description
31:0	irqsts	R	0	IRQ status Shows the status of the interrupt after masking by the VIC_int_enable and VIC_int_select registers. Value of 1 (high) indicates the interrupt is active and generates an interrupt to the processor.

Table 14-5. VIC IRQ Status Register.

14.4.2.2 VIC_fiq_status Register

Bits	Name	Attr	Reset	Description
31:0	fiqsts	R	0	FIQ status Shows the status of the interrupt after masking by the VIC_int_enable and VIC_int_select registers. Value of 1 (high) indicates the interrupt is active and generates an interrupt to the processor.

Table 14-6. VIC FIQ Status Register.

14.4.2.3 VIC_raw_intr Register

Bits	Name	Attr	Reset	Description
31:0	rintrsts	R	0	Raw interrupt status Shows the status of the interrupt before masking by the VIC_int_enable Register. Value of 1 (high) indicates the appropriate interrupt request is active before masking.

Table 14-7. VIC Raw Interrupt Status Register.

14.4.2.4 VIC_int_select Register

Bits	Name	Attr	Reset	Description
31:0	select	RW	0	Interrupt selection Select the type of interrupt for the interrupt request: 0 - IRQ interrupt 1 - FIQ interrupt

Table 14-8. VIC Interrupt Selection Register.

14.4.2.5 VIC_int_enable Register

Bits	Name	Attr	Reset	Description
31:0	enable	RW	0	Enables the request lines 0 - Interrupt disabled 1 - Interrupt enabled. Allow interrupt request to processor

Table 14-9. VIC Interrupt Enable Register.

14.4.2.6 VIC_int_en_clear Register

Bits	Name	Attr	Reset	Description
31:0	enclr	W	0	Clears bits in the VIC_int_enable register 0 - Has no effect 1 - Clears bit in the VIC_int_enable register

Table 14-10. VIC Interrupt Enable Clear Register.

14.4.2.7 VIC_soft_int Register

Bits	Name	Attr	Reset	Description
31:0	softint	RW	0	Setting a bit generates a software interrupt for the specific source interrupt before interrupt masking: 0 - Has no effect 1 - Sets the corresponding bit

Table 14-11. VIC Software Interrupt Register.

14.4.2.8 VIC_soft_int_clear Register

Bits	Name	Attr	Reset	Description
31:0	softintrclr	W	0	Clears bits in the VIC_soft_int register 0 - Has no effect 1 - Clears the bit in the VIC_soft_int register

Table 14-12. VIC Software Interrupt Clear Register.

14.4.2.9 VIC_protection Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	protect	RW	0	Enables or disables protected register access When enabled, only privileged-mode accesses can access the interrupt controller registers. When disabled, both user and privileged modes can access the registers. This register can only be accessed in privileged mode.

Table 14-13. VIC Protection Enable Register.

14.4.2.10 VIC_sense Register

Bits	Name	Attr	Reset	Description
31:0	sense	RW	0	Selects the sensitivity type of the interrupt 0 - Edge-triggered 1 - Level-sensitive

Table 14-14. VIC Interrupt Sensitivity Register.

14.4.2.11 VIC_bothedge Register

Bits	Name	Attr	Reset	Description
31:0	bothedge	RW	0	<p>Selects between single- and both-edge sensitivity types (for edge-triggered interrupt requests)</p> <p>0 - Single edge 1 - Both edges</p> <p>Note that when the sensitivity type is level-sensitive, this register has no effect.</p>

Table 14-15. VIC Interrupt Both Edge Register.
14.4.2.12 VIC_event Register

Bits	Name	Attr	Reset	Description
31:0	event	RW	0	<p>Selects between low-level/falling-edge sensitivity and high-level/rising-edge sensitivity</p> <p>0 - Low-level or falling edge triggered 1 - High-level or rising edge triggered</p> <p>Note that when the sensitivity type is both-edge-triggered, this register has no effect.</p>

Table 14-16. VIC Interrupt Event Register.
14.4.2.13 VIC_int_ptr0_reg Register

Bits	Name	Attr	Reset	Description
31:0	int_ptr0_reg	RW	0	Used to control whether a specific VIC bit can be pointed to core #0. Because A12 is a single core chip, this register must be set to all bits 1 when enabling the VIC. This register defaults to all bits 0.

Table 14-17. VIC Interrupt Pointer Core #0 Register.
14.4.2.14 VIC_edge_clr Register

Bits	Name	Attr	Reset	Description
31:0	edgeclr	W	0	<p>Clears bits in the edge detection register</p> <p>0 - No effect 1 - Clears the bit in the edge detection register</p>

Table 14-18. VIC Edge-Triggered Interrupt Clear Register.

14.4.2.15 VIC_int_select_int Register

This register sets the corresponding bit of the Interrupt Selection register (**VIC_int_select**).

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	int_select_int	W	0	Interrupt Selection

Table 14-19. VIC Interrupt Selection Integer Register.

14.4.2.16 VIC_int_select_clr_int Register

This register clears the corresponding bit of the Interrupt Selection register (**VIC_int_select**).

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	int_select_clr_int	W	0	Interrupt Selection Clear

Table 14-20. VIC Interrupt Selection Clear Integer Register.

14.4.2.17 VIC_int_en_int Register

This register sets the corresponding bit of the Interrupt Enable register (**VIC_int_enable**).

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	int_en_int	W	0	Interrupt Enable

Table 14-21. VIC Interrupt Enable Integer Register.

14.4.2.18 VIC_int_en_clr_int Register

This register clears the corresponding bits of the Interrupt Enable register (**VIC_int_enable**).

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	int_en_clr_int	W	0	Interrupt Enable Clear

Table 14-22. VIC Interrupt Enable Clear Integer Register.

14.4.2.19 VIC_soft_int_int Register

This register sets the corresponding bits of the Software Interrupt register (**VIC_soft_int**).

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	soft_int_int	W	0	Software Interrupt

Table 14-23. VIC Software Interrupt Integer Register.

14.4.2.20 VIC_soft_int_clr_int Register

This register clears the corresponding bits in the Software Interrupt register (**VIC_soft_int**).

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	soft_int_clr_int	W	0	Software Interrupt Clear

Table 14-24. VIC Software Interrupt Clear Integer Register.

14.4.2.21 VIC_int_sense_int Register

This register sets the corresponding bits of the Sensitivity register (**VIC_sense**).

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	int_sense_int	W	0	Interrupt Sensitivity type

Table 14-25. VIC Interrupt Sensitivity Integer Register.

14.4.2.22 VIC_int_sense_clr_int Register

This register clears the corresponding bits of the Sensitivity register (**VIC_sense**).

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	int_sense_clear_int	W	0	Interrupt Sensitivity clear

Table 14-26. VIC Interrupt Sensitivity Clear Integer Register.

14.4.2.23 VIC_int_bothedge_int Register

This register sets the corresponding bits of Interrupt Both Edge register (**VIC_bothedge**).

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	int_bothedge_int	W	0	Interrupt Edge type

Table 14-27. VIC Interrupt Both Edge Integer Register.

14.4.2.24 VIC_int_bothedge_clr_int Register

This register clears the corresponding bits of Interrupt Both Edge register (**VIC_bothedge**).

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	int_bothedge_clear_int	W	0	Interrupt Edge clear

Table 14-28. VIC Interrupt Both Edge Clear Integer Register.

14.4.2.25 VIC_int_evt_int Register

This register sets the corresponding bits of the Interrupt Event register (**VIC_event**).

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	int_evt_int	W	0	Interrupt Event type

Table 14-29. VIC Interrupt Event Integer Register.

14.4.2.26 VIC_int_evt_clr_int Register

This register clears the corresponding bits of the Interrupt Event register (**VIC_event**).

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	int_evt_clear_int	W	0	Interrupt Event clear

Table 14-30. VIC Interrupt Event Clear Integer Register.

14.4.2.27 VIC_int_pending Register

This register returns the pending interrupt; i.e., it returns the lowest-numbered of all outstanding interrupts. For example, if interrupts 31, 3, and 7 are outstanding then this register returns 3. This register is affected by interrupt priority settings. See [Section 14.3](#) for details.

Bits	Name	Attr	Reset	Description
31:5				Reserved
4:0	int_pending	R	0	Interrupt Pending register

Table 14-31. VIC Interrupt Pending Register.

14.4.2.28 VIC_int_re_prioritize_en Register

This register enables the re-prioritization of interrupts (write 1 to enable; write 0 to disable). When this register is disabled, the lowest-numbered interrupt is returned as a pending interrupt.

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	int_re_prioritize	RW	0	Interrupt Re-Prioritization Enable register

Table 14-32. Interrupt Re-Prioritize Enable Register.

14.4.2.29 VIC_int_priority_0 Register

This register sets the priority of bits 0 to 5. See [Section 14.3](#) for details on interrupt priority settings.

Bits	Name	Attr	Reset	Description
31:30				Reserved
29:25	pr_b5	RW	5'd5	Priority of Bit 5
24:20	pr_b4	RW	5'd4	Priority of Bit 4
19:15	pr_b3	RW	5'd3	Priority of Bit 3
14:10	pr_b2	RW	5'd2	Priority of Bit 2
9:5	pr_b1	RW	5'd1	Priority of Bit 1
4:0	pr_b0	RW	5'd0	Priority of Bit 0

Table 14-33. VIC Interrupt Priority 0 Register.

14.4.2.30 VIC_int_priority_1 Register

This register sets the priority of bits 6 to 11. See [Section 14.3](#) for details on interrupt priority settings.

Bits	Name	Attr	Reset	Description
31:30				Reserved
29:25	pr_b11	RW	5'd11	Priority of Bit 11
24:20	pr_b10	RW	5'd10	Priority of Bit 10

Bits	Name	Attr	Reset	Description
19:15	pr_b9	RW	5'd9	Priority of Bit 9
14:10	pr_b8	RW	5'd8	Priority of Bit 8
9:5	pr_b7	RW	5'd7	Priority of Bit 7
4:0	pr_b6	RW	5'd6	Priority of Bit 6

Table 14-34. VIC Interrupt Priority 1 Register.

14.4.2.31 VIC_int_priority_2 Register

This register sets the priority of bits 12 to 17. See [Section 14.3](#) for details on interrupt priority settings.

Bits	Name	Attr	Reset	Description
31:30				Reserved
29:25	pr_b17	RW	5'd17	Priority of Bit 17
24:20	pr_b16	RW	5'd16	Priority of Bit 16
19:15	pr_b15	RW	5'd15	Priority of Bit 15
14:10	pr_b14	RW	5'd14	Priority of Bit 14
9:5	pr_b13	RW	5'd13	Priority of Bit 13
4:0	pr_b12	RW	5'd12	Priority of Bit 12

Table 14-35. VIC Interrupt Priority 2 Register.

14.4.2.32 VIC_int_priority_3 Register

This register sets the priority of bits 23 to 18. See [Section 14.3](#) for details on interrupt priority settings.

Bits	Name	Attr	Reset	Description
31:30				Reserved
29:25	pr_b23	RW	5'd23	Priority of Bit 23
24:20	pr_b22	RW	5'd22	Priority of Bit 22
19:15	pr_b21	RW	5'd21	Priority of Bit 21
14:10	pr_b20	RW	5'd20	Priority of Bit 20
9:5	pr_b19	RW	5'd19	Priority of Bit 19
4:0	pr_b18	RW	5'd18	Priority of Bit 18

Table 14-36. VIC Interrupt Priority 3 Register.

14.4.2.33 VIC_int_priority_4 Register

This register sets the priority of bits 24 to 29. See [Section 14.3](#) for details on interrupt priority settings.

Bits	Name	Attr	Reset	Description
31:30				Reserved
29:25	pr_b29	RW	5'd29	Priority of Bit 29
24:20	pr_b28	RW	5'd28	Priority of Bit 28

Bits	Name	Attr	Reset	Description
19:15	pr_b27	RW	5'd27	Priority of Bit 27
14:10	pr_b26	RW	5'd26	Priority of Bit 26
9:5	pr_b25	RW	5'd25	Priority of Bit 25
4:0	pr_b24	RW	5'd24	Priority of Bit 24

Table 14-37. VIC Interrupt Priority 4 Register.

14.4.2.34 VIC_int_priority_5 Register

This register sets the priority of bits 30 to 31. See [Section 14.3](#) for details on interrupt priority settings.

Bits	Name	Attr	Reset	Description
31:10				Reserved
9:5	pr_b31	RW	5'd31	Priority of Bit 31
4:0	pr_b30	RW	5'd30	Priority of Bit 30

Table 14-38. VIC Interrupt Priority 5 Register.

14.4.2.35 VIC_int_delay_en Register

This register enables the delaying of interrupts. Writing 1 enables delaying the raw interrupts and writing 0 disables delay.

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	int_delay_en	RW	0	Interrupt Delay Enable register

Table 14-39. VIC Interrupt Delay Enable Register.

14.4.2.36 VIC_int_delay Register

This register specifies the delay interval for the interrupts. The interrupt is asserted after the delay specified by this delay register if the Delay Enabled register (**VIC_int_delay_en**) is set.

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:0	int_delay	RW	0	Interrupt Delay register (uses units of the AHB clock cycle)

Table 14-40. VIC Interrupt Delay Register.

15. GPIO

Register programming for the A12 General Purpose Input / Output (GPIO) interface is covered in this chapter as follows:

- [\(Section 15.1\) GPIO: Overview](#)
- [\(Section 15.2\) GPIO: Registers](#)
- [\(Section 15.3\) GPIO: Interrupt Control](#)
- [\(Section 15.4\) GPIO: Function Selection](#)

15.1 GPIO: Overview

The A12 chip includes up to 114 configurable multi-purpose GPIO pins, many of which can be configured as I/O pins for various hardware modules. Otherwise, the GPIO pins are controlled by the GPIO register spaces located on the APB bus. The **GPIO0_x** registers control GPIO[0:31], the **GPIO1_x** registers control GPIO[32:63], and so on. Each set of GPIO registers occupies its own range of addresses on the APB bus.

All GPIO pins are interrupt capable. Refer to [Chapter 14](#) for more information regarding the Vector Interrupt Controller (VIC).

15.2 GPIO: Registers

15.2.1 GPIO Registers: Map

APB Base Address	Register Offset	Register Name	Description
E800.9000	0x00	GPIO0_data	Data Register
	0x04	GPIO0_dir	Data Direction
	0x08	GPIO0_is	Edge/Level Interrupt Sensitivity Control
	0x0C	GPIO0_ibc	Edge Interrupt Sensitivity Control
	0x10	GPIO0_iev	Interrupt Event
	0x14	GPIO0_ie	Interrupt Mask
	0x18	GPIO0_afsel	Mode Select
	0x1C	GPIO0_ris	Raw Interrupt Status
	0x20	GPIO0_mis	Masked Interrupt Status
	0x24	GPIO0_ic	Interrupt Clear
	0x28	GPIO0_mask	GPIO Mask
	0x2C	GPIO0_enable	GPIO Enable
E800.A000	0x00	GPIO1_data	Data Register
	0x04	GPIO1_dir	Data Direction
	0x08	GPIO1_is	Edge/Level Interrupt Sensitivity Control
	0x0C	GPIO1_ibc	Edge Interrupt Sensitivity Control
	0x10	GPIO1_iev	Interrupt Event
	0x14	GPIO1_ie	Interrupt Mask
	0x18	GPIO1_afsel	Mode Select

APB Base Address	Register Offset	Register Name	Description
	0x1C	GPIO1_ris	Raw Interrupt Status
	0x20	GPIO1_mis	Masked Interrupt Status
	0x24	GPIO1_ic	Interrupt Clear
	0x28	GPIO1_mask	GPIO Mask
	0x2C	GPIO1_enable	GPIO Enable
E800.E000	0x00	GPIO2_data	Data Register
	0x04	GPIO2_dir	Data Direction
	0x08	GPIO2_is	Edge/Level Interrupt Sensitivity Control
	0x0C	GPIO2_ibc	Edge Interrupt Sensitivity Control
	0x10	GPIO2_iev	Interrupt Event
	0x14	GPIO2_ie	Interrupt Mask
	0x18	GPIO2_afsel	Mode Select
	0x1C	GPIO2_ris	Raw Interrupt Status
	0x20	GPIO2_mis	Masked Interrupt Status
	0x24	GPIO2_ic	Interrupt Clear
	0x28	GPIO2_mask	GPIO Mask
	0x2C	GPIO2_enable	GPIO Enable
E801.0000	0x00	GPIO3_data	Data Register
	0x04	GPIO3_dir	Data Direction
	0x08	GPIO3_is	Edge/Level Interrupt Sensitivity Control
	0x0C	GPIO3_ibc	Edge Interrupt Sensitivity Control
	0x10	GPIO3_iev	Interrupt Event
	0x14	GPIO3_ie	Interrupt Mask
	0x18	GPIO3_afsel	Mode Select
	0x1C	GPIO3_ris	Raw Interrupt Status
	0x20	GPIO3_mis	Masked Interrupt Status
	0x24	GPIO3_ic	Interrupt Clear
	0x28	GPIO3_mask	GPIO Mask
	0x2C	GPIO3_enable	GPIO Enable

Table 15-1. Address Offsets for Each Set of GPIO Registers.

Within each GPIO register, each bit corresponds to one GPIO pin. For example, bit 0 of the **GPIO0_data** register ([Section 15.2.2.1](#)) is the data bit for GPIO pin 0 (**GPIO_0**); bit 31 of the **GPIO0_data** register is the data bit for GPIO pin 31 (**IDC3CLK**); bit 0 of the **GPIO1_data** register is the data bit for GPIO pin 32 (**IDC3DATA**) and so on.

The **GPIO[n].afsel** register is used to select whether the corresponding GPIO is configured to function as a GPIO pin (primary function) or to take on a secondary function.

15.2.2 GPIO Registers: Details

15.2.2.1 GPIO[n]_data Register

Bits	Name	Attr	Reset	Description
31:0	bits	RW	0	Data register of the GPIO controller used to drive and read the PAD

Table 15-2. GPIO Data Register.

15.2.2.2 GPIO[n]_dir Register

Bits	Name	Attr	Reset	Description
31:0	io	RW	0	Selects between input/output control of the GPIO lines 0 - Input 1 - Output

Table 15-3. GPIO Data Direction Register.

15.2.2.3 GPIO[n]_is Register

Bits	Name	Attr	Reset	Description
31:0	is	RW	0	Interrupt sensitivity control 0 - Edge is detected 1 - Level is detected

Table 15-4. GPIO Edge/Level Interrupt Sensitivity Control Register.

15.2.2.4 GPIO[n]_ibe Register

Bits	Name	Attr	Reset	Description
31:0	ibe	RW	0	Interrupt on both edges 0 - GPIO_iev register controls interrupt generation event 1 - Both edges on corresponding pin trigger interrupt

Table 15-5. GPIO Edge Interrupt Sensitivity Control Register.

15.2.2.5 GPIO[n].iev Register

Bits	Name	Attr	Reset	Description
31:0	iev	RW	0	Interrupt event control 0 - Falling edges, or low levels trigger interrupts 1 - Rising edges, or high levels trigger interrupts

Table 15-6. GPIO Interrupt Event Register.

15.2.2.6 GPIO[n].ie Register

Bits	Name	Attr	Reset	Description
31:0	ie	RW	0	Interrupt enable control 0 - Corresponding pin interrupt is masked 1 - Corresponding pin is not masked

Table 15-7. GPIO Interrupt Mask Register.

15.2.2.7 GPIO[n].afsel Register

Bits	Name	Attr	Reset	Description
31:0	afsel	RW	NR	Mode control selection 0 - Corresponding pin is in software control mode 1 - Corresponding pin is in hardware control mode

Table 15-8. GPIO Mode Select Register.

15.2.2.8 GPIO[n].ris Register

Bits	Name	Attr	Reset	Description
31:0	ris	R	0	Raw interrupt status Reflect the status of interrupt trigger condition-detection on pins. 0 - Requirement not met 1 - Requirement met

Table 15-9. GPIO Raw Interrupt Status Register.

15.2.2.9 GPIO[n].mis Register

Bits	Name	Attr	Reset	Description
31:0	mis	R	0	Masked interrupt status 0 - GPIO line interrupt not active 1 - GPIO line asserting interrupt

Table 15-10. GPIO Masked Interrupt Status Register.

15.2.2.10 **GPIO[n].ic** Register

Bits	Name	Attr	Reset	Description
31:0	clr	W		Interrupt clear 0 - Has no effect 1 - Clears edge-detection logic

Table 15-11. GPIO Interrupt Clear Register.

15.2.2.11 **GPIO[n].mask** Register

The application of a mask makes it simpler for the software to modify (or observe) only the bits in the **GPIO[n].data** register of concern.

For example, consider the case that GPIO pin 0 (**GPIO_0**) is connected to an LED (e.g., LED_0) and GPIO pin 1 (**GPIO_1**) is connected to a separate LED (e.g., LED_1). Assume LED_1 is currently on, (i.e., **GPIO0_data[1]** is set to 1). If the goal is for LED_0 to blink without affecting LED_1, the software can be set to **GPIO[n].mask** = 0x1 followed by a sequence of writes to the **GPIO[n].data** register with values 0,1,0,1,0,1, and so on. In this case, only bit 0 is affected, and all other GPIO bits will not be affected by subsequent writes. Due to the mask, LED_1 will stay on even when the software writes 0x00000000 to the **GPIO[n].data** register.

Bits	Name	Attr	Reset	Description
31:0	mask	RW	0	Mask of data read/write. 0 - Corresponding bit of GPIO[n].data is masked 1 - Corresponding bit of GPIO[n].data is accessible

Table 15-12. GPIO Mask Register.

15.2.2.12 **GPIO[n].enable** Register

GPIO[n].enable registers are used to enable all GPIO pins. After reset, the bits in this register revert to zero and all GPIO pins are, in turn, disabled. When the software enables the GPIO function by writing 1's to the **GPIO[n].enable** register, it also must write 0's to the corresponding bits in **GPIO[n].afsel** registers to enable software control of the GPIO pins.

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	enb	RW	0	GPIO enable 0 - All GPIO is disabled 1 - All GPIO is enabled

Table 15-13. GPIO Enable Register.

15.3 GPIO: Interrupt Control

Each GPIO pin can be configured to generate an interrupt on a rising edge, a falling edge, both edges, a logic level high or a logic level low. The interrupt sensitivity can be configured using the **GPIO[n].is**, **GPIO[n].ibe**, **GPIO[n].iev** and **GPIO[n].ie** registers.

When an interrupt condition is detected on a GPIO line, the corresponding bit in the **GPIO[n].ris** register will be set to 1. The **GPIO[n].ris** register contains the “raw” interrupt status for each GPIO pin.

The **GPIO[n].mis** register is the result of the bit-wise AND of the **GPIO[n].ris** register and the **GPIO[n].ie** register. When the **GPIO[n].mis** register is non-zero, the interrupt signal to the VIC will be asserted. In servicing the VIC-**GPIO[n]** interrupt, the software should read the **GPIO[n].mis** register to determine which GPIO pin(s) caused the interrupt.

15.4 GPIO: Function Selection

This section lists the A12 GPIO pins, including function selection information. Refer to the A12 datasheet for GPIO pin descriptions.

15.4.1 GPIO Function Selection: (IOMUX) Registers

There are 12 configurable word-sized (32 bit) registers within the **IOMUX** registry that can be sub-divided into 4 register sets, where each set consists of 3 registers.

Note that these registers will not take effect unless the **CTRL_SET** register is set to 1. All registers, including **CTRL_SET** registers, can be accessed through the APB bus at base address 0xE801.6000. The following table contains a list of IOMUX registers, including address offset.

Address Offset	Register Name	Description
0x00	iomux_reg0_0	Register Set 0
0x04	iomux_reg0_1	
0x08	iomux_reg0_2	
0x0C	iomux_reg1_0	Register Set 1
0x10	iomux_reg1_1	
0x14	iomux_reg1_2	
0x18	iomux_reg2_0	Register Set 2
0x1C	iomux_reg2_1	
0x20	iomux_reg2_2	
0x24	iomux_reg3_0	Register Set 3
0x28	iomux_reg3_1	
0x2C	iomux_reg3_2	
0xF0	iomux_ctrl_set	Register Control

Table 15-14. GPIO Function Selection (IOMUX) Registers.

15.4.2 GPIO Function Selection: Programming Procedures

GPIO functions are selected using the registers referenced above in [Section 15.4.1](#). The 3-bit information, combined from the same bit positions of registers within each set, jointly determines an IO pad signal's source/destination mapping. Refer to the figure below.

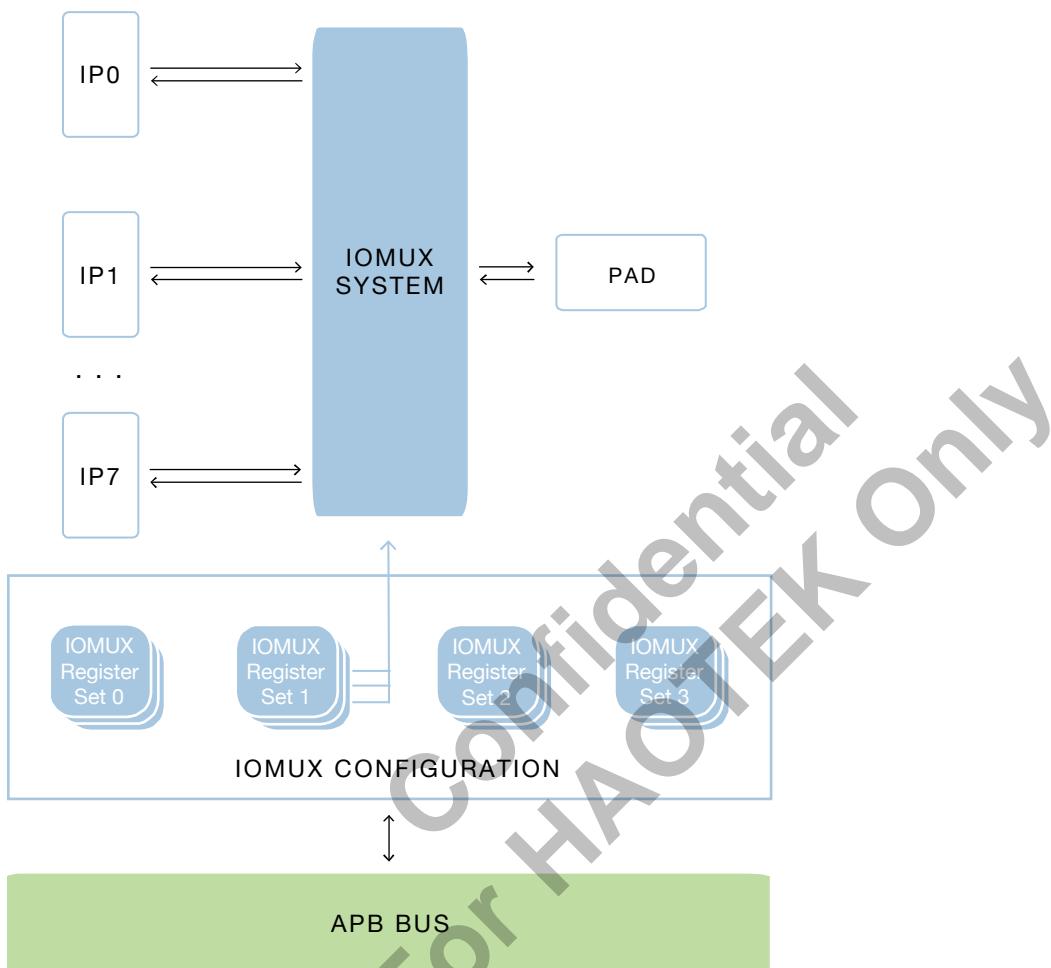


Figure 15-1. GPIO Function Selection: IOMUX Block Diagram.

Note:

- IP0, IP1, etc., represent the A12 modules/subsystems connected to the IOMUX System, which differ depending on the pad under examination. Using **GPIO_0** as an example, only the GPIO and SD Controller Subsystem modules (IPs) would be connected.

Table 15-15 provides a list of register bit locations to be used during GPIO function selection. The first entry has been excerpted below in order to serve as a guide for interpreting the table.

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
0	GPIO_0 ⁽¹⁾	1: GPIO (IO) ⁽²⁾	1: sd_hs_sel (O) ⁽⁴⁾				
		2: {0x8[0], 0x4[0], 0x0[0]} = 4'h000 ⁽³⁾	2: {0x8[0], 0x4[0], 0x0[0]} = 4'h001				

Referring to the excerpted entry above:

⁽¹⁾ Pin Name

⁽²⁾ Alternative Function 0 Name

⁽³⁾ Selection Bit Value for the Corresponding Register Set

In this example, bit 0 in **iomux_reg0_2** (0x8), **iomux_reg0_1** (0x4), and **iomux_reg0_0** (0x0) must be configured as {0, 0, 0 (4'h000)} in order to enable alternative function 0.

⁽⁴⁾ Alternative Function 1 Name

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
0	GPIO_0	1, GPIO (IO) 2, {0x8[0], 0x4[0], 0x0[0]} = 4'h000	1, sd_hs_sel (O) 2, {0x8[0], 0x4[0], 0x0[0]} = 4'h001				
1	GPIO_1	1, GPIO (IO) 2, {0x8[1], 0x4[1], 0x0[1]} = 4'h000	1, ehci_app_prt_ovcurr0 (I) 2, {0x8[1], 0x4[1], 0x0[1]} = 4'h001	1, uart_ahb_rx (I) 2, {0x8[1], 0x4[1], 0x0[1]} = 4'h010	1, ssis_sclk (I) 2, {0x8[1], 0x4[1], 0x0[1]} = 4'h011	1, sc_c0 (O) 2, {0x8[1], 0x4[1], 0x0[1]} = 4'h100	
2	GPIO_2	1, GPIO (IO) 2, {0x8[2], 0x4[2], 0x0[2]} = 4'h000	1, ehci_app_prt_ovcurr1 (I) 2, {0x8[2], 0x4[2], 0x0[2]} = 4'h001	1, uart_ahb_tx (O) 2, {0x8[2], 0x4[2], 0x0[2]} = 4'h010	1, ssis_rxd (I) 2, {0x8[2], 0x4[2], 0x0[2]} = 4'h011	1, sc_c1 (O) 2, {0x8[2], 0x4[2], 0x0[2]} = 4'h100	
3	GPIO_3	1, GPIO (IO) 2, {0x8[3], 0x4[3], 0x0[3]} = 4'h000	1, ehci_prt_pwr_0 (O) 2, {0x8[3], 0x4[3], 0x0[3]} = 4'h001	1, uart_ahb_cts_n (I) 2, {0x8[3], 0x4[3], 0x0[3]} = 4'h010	1, ssis_txd (O+) 2, {0x8[3], 0x4[3], 0x0[3]} = 4'h011	1, sc_c2 (O) 2, {0x8[3], 0x4[3], 0x0[3]} = 4'h100	
4	GPIO_4	1, GPIO (IO) 2, {0x8[4], 0x4[4], 0x0[4]} = 4'h000	1, ehci_prt_pwr_1 (O) 2, {0x8[4], 0x4[4], 0x0[4]} = 4'h001	1, uart_ahb_rts_n (O) 2, {0x8[4], 0x4[4], 0x0[4]} = 4'h010	1, ssis_en (I) 2, {0x8[4], 0x4[4], 0x0[4]} = 4'h011	1, sc_c3 (O) 2, {0x8[4], 0x4[4], 0x0[4]} = 4'h100	

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
5	GPIO_5	1, GPIO (IO) 2, {0x8[5], 0x4[5], 0x0[5]} = 4'h000	1, pwm_1 (O) 2, {0x8[5], 0x4[5], 0x0[5]} = 4'h001	1, idsp_pip_ipad_master_hsync (O) 2, {0x8[5], 0x4[5], 0x0[5]} = 4'h010	1, vin_strig0 (O+) 2, {0x8[5], 0x4[5], 0x0[5]} = 4'h011	1, sc_d0 (O) 2, {0x8[5], 0x4[5], 0x0[5]} = 4'h100	1, uart_ahb_cts_n (I) 2, {0x8[5], 0x4[5], 0x0[5]} = 4'h101
6	GPIO_6	1, GPIO (IO) 2, {0x8[6], 0x4[6], 0x0[6]} = 4'h000	1, pwm_2 (O) 2, {0x8[6], 0x4[6], 0x0[6]} = 4'h001	1, idsp_pip_ipad_master_vsync (O) 2, {0x8[6], 0x4[6], 0x0[6]} = 4'h010	1, vin_strig1 (O+) 2, {0x8[6], 0x4[6], 0x0[6]} = 4'h011	1, sc_d1 (O) 2, {0x8[6], 0x4[6], 0x0[6]} = 4'h100	1, uart_ahb_rts_n (O) 2, {0x8[6], 0x4[6], 0x0[6]} = 4'h101
7	SC_A0	1, GPIO (IO) 2, {0x8[7], 0x4[7], 0x0[7]} = 4'h000	1, sc_a0 (O) 2, {0x8[7], 0x4[7], 0x0[7]} = 4'h001	1, ssi1_sclk (O) 2, {0x8[7], 0x4[7], 0x0[7]} = 4'h010	1, norspi_clk (O) 2, {0x8[7], 0x4[7], 0x0[7]} = 4'h011 3, rct_ahb_nor.spi_boot_0 =((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 0))	1, pwm_0 (O) 2, {0x8[7], 0x4[7], 0x0[7]} = 4'h100	1, sdxc_cmd (IO) 2, {0x8[7], 0x4[7], 0x0[7]} = 4'h101
8	SC_A1	1, GPIO (IO) 2, {0x8[8], 0x4[8], 0x0[8]} = 4'h000	1, sc_a1 (O) 2, {0x8[8], 0x4[8], 0x0[8]} = 4'h001	1, ssi1_txd (O+) 2, {0x8[8], 0x4[8], 0x0[8]} = 4'h010	1, norspi_dq[0] (IO) 2, {0x8[8], 0x4[8], 0x0[8]} = 4'h011 3, rct_ahb_nor.spi_boot_0 =((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 0))	1, pwm_1 (O) 2, {0x8[8], 0x4[8], 0x0[8]} = 4'h100	1, sdxc_cd (I) 2, {0x8[8], 0x4[8], 0x0[8]} = 4'h101
9	SC_A2	1, GPIO (IO) 2, {0x8[9], 0x4[9], 0x0[9]} = 4'h000	1, sc_a2 (O) 2, {0x8[9], 0x4[9], 0x0[9]} = 4'h001	1, ssi1_rxd (I) 2, {0x8[9], 0x4[9], 0x0[9]} = 4'h010	1, norspi_dq[1] (IO) 2, {0x8[9], 0x4[9], 0x0[9]} = 4'h011 3, rct_ahb_nor.spi_boot_0 =((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 0))	1, pwm_2 (O) 2, {0x8[9], 0x4[9], 0x0[9]} = 4'h100	1, sdxc_wp (I) 2, {0x8[9], 0x4[9], 0x0[9]} = 4'h101

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
10	SC_A3	1, GPIO (IO) 2, {0x8[10], 0x4[10], 0x0[10]} = 4'h000	1, sc_a3 (O) 2, {0x8[10], 0x4[10], 0x0[10]} = 4'h001	1, ssi1_en0 (O) 2, {0x8[10], 0x4[10], 0x0[10]} = 4'h010	1, norspi_dq[2] (IO) 2, {0x8[10], 0x4[10], 0x0[10]} = 4'h011 3, rct_ahb_nor- spi_boot_0 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 0))	1, pwm_3 (O) 2, {0x8[10], 0x4[10], 0x0[10]} = 4'h100	1, sdxc_d[0] (IO) 2, {0x8[10], 0x4[10], 0x0[10]} = 4'h101
11	SC_B0	1, GPIO (IO) 2, {0x8[11], 0x4[11], 0x0[11]} = 4'h000	1, sc_b0 (O) 2, {0x8[11], 0x4[11], 0x0[11]} = 4'h001	1, ssi1_en1 (O) 2, {0x8[11], 0x4[11], 0x0[11]} = 4'h010	1, norspi_dq[3] (IO) 2, {0x8[11], 0x4[11], 0x0[11]} = 4'h011 3, rct_ahb_nor- spi_boot_0 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 0))		1, sdxc_d[1] (IO) 2, {0x8[11], 0x4[11], 0x0[11]} = 4'h101
12	SC_B1	1, GPIO (IO) 2, {0x8[12], 0x4[12], 0x0[12]} = 4'h000	1, sc_b1 (O) 2, {0x8[12], 0x4[12], 0x0[12]} = 4'h001	1, ssi1_en2 (O) 2, {0x8[12], 0x4[12], 0x0[12]} = 4'h010	1, norspi_en[0] (O) 2, {0x8[12], 0x4[12], 0x0[12]} = 4'h011 3, rct_ahb_nor- spi_boot_0 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 0))	1, norspi_dq[2] (IO) 2, {0x8[12], 0x4[12], 0x0[12]} = 4'h100	1, sdxc_d[2] (IO) 2, {0x8[12], 0x4[12], 0x0[12]} = 4'h101
13	SC_B2	1, GPIO (IO) 2, {0x8[13], 0x4[13], 0x0[13]} = 4'h000	1, sc_b2 (O) 2, {0x8[13], 0x4[13], 0x0[13]} = 4'h001	1, ssi1_en3 (O) 2, {0x8[13], 0x4[13], 0x0[13]} = 4'h010	1, norspi_en[1] (O) 2, {0x8[13], 0x4[13], 0x0[13]} = 4'h011 3, rct_ahb_nor- spi_boot_0 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 0))	1, norspi_dq[3] (IO) 2, {0x8[13], 0x4[13], 0x0[13]} = 4'h100	1, sdxc_d[3] (IO) 2, {0x8[13], 0x4[13], 0x0[13]} = 4'h101

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
14	SC_B3	1, GPIO (IO) 2, {0x8[14], 0x4[14], 0x0[14]} = 4'h000	1, sc_b3 (O) 2, {0x8[14], 0x4[14], 0x0[14]} = 4'h001	1, pwm_3 (O) 2, {0x8[14], 0x4[14], 0x0[14]} = 4'h010	1, norspi_en[2] (O) 2, {0x8[14], 0x4[14], 0x0[14]} = 4'h011		1, sdxc_d[4] (IO) 2, {0x8[14], 0x4[14], 0x0[14]} = 4'h101
15	SC_CO	1, GPIO (IO) 2, {0x8[15], 0x4[15], 0x0[15]} = 4'h000	1, sc_c0 (O) 2, {0x8[15], 0x4[15], 0x0[15]} = 4'h001	1, uart_ahb_rx (I) 2, {0x8[15], 0x4[15], 0x0[15]} = 4'h010	1, ssis_sclk (I) 2, {0x8[15], 0x4[15], 0x0[15]} = 4'h011		1, sdxc_d[5] (IO) 2, {0x8[15], 0x4[15], 0x0[15]} = 4'h101
16	SC_C1	1, GPIO (IO) 2, {0x8[16], 0x4[16], 0x0[16]} = 4'h000	1, sc_c1 (O) 2, {0x8[16], 0x4[16], 0x0[16]} = 4'h001	1, uart_ahb_tx (O) 2, {0x8[16], 0x4[16], 0x0[16]} = 4'h010	1, ssis_rxd (I) 2, {0x8[16], 0x4[16], 0x0[16]} = 4'h011	1, enet_crs (I) 2, {0x8[16], 0x4[16], 0x0[16]} = 4'h100 3, rct_ahb_mii_sel = ((POC[0] == 1) && (POC[6] == 0))	1, sdxc_d[6] (IO) 2, {0x8[16], 0x4[16], 0x0[16]} = 4'h101
17	SC_C2	1, GPIO (IO) 2, {0x8[17], 0x4[17], 0x0[17]} = 4'h000	1, sc_c2 (O) 2, {0x8[17], 0x4[17], 0x0[17]} = 4'h001	1, uart_ahb_cts_n (I) 2, {0x8[17], 0x4[17], 0x0[17]} = 4'h010	1, ssis_txd (O+) 2, {0x8[17], 0x4[17], 0x0[17]} = 4'h011	1, enet_rxd_2 (I) 2, {0x8[17], 0x4[17], 0x0[17]} = 4'h100 3, rct_ahb_mii_sel = ((POC[0] == 1) && (POC[6] == 0))	1, sdxc_d[7] (IO) 2, {0x8[17], 0x4[17], 0x0[17]} = 4'h101
18	SC_C3	1, GPIO (IO) 2, {0x8[18], 0x4[18], 0x0[18]} = 4'h000	1, sc_c3 (O) 2, {0x8[18], 0x4[18], 0x0[18]} = 4'h001	1, uart_ahb_rts_n (O) 2, {0x8[18], 0x4[18], 0x0[18]} = 4'h010	1, ssis_en (I) 2, {0x8[18], 0x4[18], 0x0[18]} = 4'h011	1, enet_rxd_3 (I) 2, {0x8[18], 0x4[18], 0x0[18]} = 4'h100 3, rct_ahb_mii_sel = ((POC[0] == 1) && (POC[6] == 0))	1, sdxc_clk (O) 2, {0x8[18], 0x4[18], 0x0[18]} = 4'h101
19	SC_D0	1, GPIO (IO) 2, {0x8[19], 0x4[19], 0x0[19]} = 4'h000	1, sc_d0 (O) 2, {0x8[19], 0x4[19], 0x0[19]} = 4'h001	1, uart_ahb_rx (I) 2, {0x8[19], 0x4[19], 0x0[19]} = 4'h010	1, ssis_sclk (I) 2, {0x8[19], 0x4[19], 0x0[19]} = 4'h011	1, enet_col (I) 2, {0x8[19], 0x4[19], 0x0[19]} = 4'h100 3, rct_ahb_mii_sel = ((POC[0] == 1) && (POC[6] == 0))	1, pwm_0 (O) 2, {0x8[19], 0x4[19], 0x0[19]} = 4'h101

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
20	SC_D1	1, GPIO (IO) 2, {0x8[20], 0x4[20], 0x0[20]} = 4'h000	1, sc_d1 (O) 2, {0x8[20], 0x4[20], 0x0[20]} = 4'h001	1, uart_ahb_tx (O) 2, {0x8[20], 0x4[20], 0x0[20]} = 4'h010	1, ssis_rxd (I) 2, {0x8[20], 0x4[20], 0x0[20]} = 4'h011	1, enet_tx_clk (I) 2, {0x8[20], 0x4[20], 0x0[20]} = 4'h100 3, rct_ahb_mii_- sel = ((POC[0] == 1) && (POC[6] == 0))	1, pwm_1 (O) 2, {0x8[20], 0x4[20], 0x0[20]} = 4'h101
21	SC_D2	1, GPIO (IO) 2, {0x8[21], 0x4[21], 0x0[21]} = 4'h000	1, sc_d2 (O) 2, {0x8[21], 0x4[21], 0x0[21]} = 4'h001	1, uart_ahb_- cts_n (I) 2, {0x8[21], 0x4[21], 0x0[21]} = 4'h010	1, ssis_txd (O+) 2, {0x8[21], 0x4[21], 0x0[21]} = 4'h011	1, enet_tx_er (O) 2, {0x8[21], 0x4[21], 0x0[21]} = 4'h100 3, rct_ahb_mii_- sel = ((POC[0] == 1) && (POC[6] == 0))	1, pwm_2 (O) 2, {0x8[21], 0x4[21], 0x0[21]} = 4'h101
22	SC_D3	1, GPIO (IO) 2, {0x8[22], 0x4[22], 0x0[22]} = 4'h000	1, sc_d3 (O) 2, {0x8[22], 0x4[22], 0x0[22]} = 4'h001	1, uart_ahb_- rts_n (O) 2, {0x8[22], 0x4[22], 0x0[22]} = 4'h010	1, ssis_en (I) 2, {0x8[22], 0x4[22], 0x0[22]} = 4'h011	1, enet_txd_2 (O) 2, {0x8[22], 0x4[22], 0x0[22]} = 4'h100 3, rct_ahb_mii_- sel = ((POC[0] == 1) && (POC[6] == 0))	1, pwm_3 (O) 2, {0x8[22], 0x4[22], 0x0[22]} = 4'h101
23	SC_E0	1, GPIO (IO) 2, {0x8[23], 0x4[23], 0x0[23]} = 4'h000	1, sc_e0 (O) 2, {0x8[23], 0x4[23], 0x0[23]} = 4'h001	1, ssi0_en2 (O) 2, {0x8[23], 0x4[23], 0x0[23]} = 4'h010	1, norspi_en[3] (O) 2, {0x8[23], 0x4[23], 0x0[23]} = 4'h011	1, enet_txd_3 (O) 2, {0x8[23], 0x4[23], 0x0[23]} = 4'h100 3, rct_ahb_mii_- sel = ((POC[0] == 1) && (POC[6] == 0))	1, pwm_1 (O) 2, {0x8[23], 0x4[23], 0x0[23]} = 4'h101
24	TIMER0	1, GPIO (IO) 2, {0x8[24], 0x4[24], 0x0[24]} = 4'h000	1, tm11_clk (I) 2, {0x8[24], 0x4[24], 0x0[24]} = 4'h001			1, enet_2nd_- ref_clk (O) 2, {0x8[24], 0x4[24], 0x0[24]} = 4'h100 3, rct_ahb_- enet_sel = (POC[0] == 1)	

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
25	TIMER1	1, GPIO (IO) 2, {0x8[25], 0x4[25], 0x0[25]} = 4'h000	1, tm12_clk (I) 2, {0x8[25], 0x4[25], 0x0[25]} = 4'h001		1, idsp_pip_iopad_master_hsync (O) 2, {0x8[25], 0x4[25], 0x0[25]} = 4'h100	1, enet_mdc (O) 2, {0x8[25], 0x4[25], 0x0[25]} = 4'h011	
26	TIMER2	1, GPIO (IO) 2, {0x8[26], 0x4[26], 0x0[26]} = 4'h000	1, tm13_clk (I) 2, {0x8[26], 0x4[26], 0x0[26]} = 4'h001	1, ssi0_en3 (O) 2, {0x8[26], 0x4[26], 0x0[26]} = 4'h010	1, idsp_pip_iopad_master_vsync (O) 2, {0x8[26], 0x4[26], 0x0[26]} = 4'h011	1, enet_mdio (IO) 2, {0x8[26], 0x4[26], 0x0[26]} = 4'h100	
27	IDCCLK	1, GPIO (IO) 2, {0x8[27], 0x4[27], 0x0[27]} = 4'h000	1, idc0clk (IO) 2, {0x8[27], 0x4[27], 0x0[27]} = 4'h001				
28	IDCDATA	1, GPIO (IO) 2, {0x8[28], 0x4[28], 0x0[28]} = 4'h000	1, idc0data (IO) 2, {0x8[28], 0x4[28], 0x0[28]} = 4'h001				
29	IDC2CLK	1, GPIO (IO) 2, {0x8[29], 0x4[29], 0x0[29]} = 4'h000	1, idc1clk (IO) 2, {0x8[29], 0x4[29], 0x0[29]} = 4'h001		1, norspi_dq[2] (IO) 2, {0x8[29], 0x4[29], 0x0[29]} = 4'h011	1, norspi_en[2] (O) 2, {0x8[29], 0x4[29], 0x0[29]} = 4'h100	
30	IDC2DATA	1, GPIO (IO) 2, {0x8[30], 0x4[30], 0x0[30]} = 4'h000	1, idc1data (IO) 2, {0x8[30], 0x4[30], 0x0[30]} = 4'h001		1, norspi_dq[3] (IO) 2, {0x8[30], 0x4[30], 0x0[30]} = 4'h011	1, norspi_en[3] (O) 2, {0x8[30], 0x4[30], 0x0[30]} = 4'h100	
31	IDC3CLK	1, GPIO (IO) 2, {0x8[31], 0x4[31], 0x0[31]} = 4'h000	1, idc2clk (IO) 2, {0x8[31], 0x4[31], 0x0[31]} = 4'h001	1, vin_strig0 (O+) 2, {0x8[31], 0x4[31], 0x0[31]} = 4'h010			
32	IDC3DATA	1, GPIO (IO) 2, {0x14[0], 0x10[0], 0xc[0]} = 4'h000	1, idc2data (IO) 2, {0x14[0], 0x10[0], 0xc[0]} = 4'h001	1, vin_strig1 (O+) 2, {0x14[0], 0x10[0], 0xc[0]} = 4'h010			
33	IR_IN	1, GPIO (IO) 2, {0x14[1], 0x10[1], 0xc[1]} = 4'h000	1, ir_in (I) 2, {0x14[1], 0x10[1], 0xc[1]} = 4'h001				

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
34	SSIOCLK	1, GPIO (IO) 2, {0x14[2], 0x10[2], 0xc[2]} = 4'h000	1, ssi0_sclk (O) 2, {0x14[2], 0x10[2], 0xc[2]} = 4'h001 3, rct_ahb_ spi_boot = ((POC[10] == 0) && (POC[5:4] == 3))	1, norspi_clk (O) 2, {0x14[2], 0x10[2], 0xc[2]} = 4'h010	1, uart_ahb_rx (I) 2, {0x14[2], 0x10[2], 0xc[2]} = 4'h011	1, ssis_sclk (I) 2, {0x14[2], 0x10[2], 0xc[2]} = 4'h100	
35	SSIOMOSI	1, GPIO (IO) 2, {0x14[3], 0x10[3], 0xc[3]} = 4'h000	1, ssi0_txd (O+) 2, {0x14[3], 0x10[3], 0xc[3]} = 4'h001 3, rct_ahb_ spi_boot = ((POC[10] == 0) && (POC[5:4] == 3))	1, norspi_dq[0] (IO) 2, {0x14[3], 0x10[3], 0xc[3]} = 4'h010	1, uart_ahb_tx (O) 2, {0x14[3], 0x10[3], 0xc[3]} = 4'h011	1, ssis_rxd (I) 2, {0x14[3], 0x10[3], 0xc[3]} = 4'h100	
36	SSIOMISO	1, GPIO (IO) 2, {0x14[4], 0x10[4], 0xc[4]} = 4'h000	1, ssi0_rxd (I) 2, {0x14[4], 0x10[4], 0xc[4]} = 4'h001 3, rct_ahb_ spi_boot = ((POC[10] == 0) && (POC[5:4] == 3))	1, norspi_dq[1] (IO) 2, {0x14[4], 0x10[4], 0xc[4]} = 4'h010	1, uart_ahb_ cts_n (I) 2, {0x14[4], 0x10[4], 0xc[4]} = 4'h011	1, ssis_txd (O+) 2, {0x14[4], 0x10[4], 0xc[4]} = 4'h100	
37	SSIOENO	1, GPIO (IO) 2, {0x14[5], 0x10[5], 0xc[5]} = 4'h000	1, ssi0_en0 (O) 2, {0x14[5], 0x10[5], 0xc[5]} = 4'h001 3, rct_ahb_ spi_boot = ((POC[10] == 0) && (POC[5:4] == 3))	1, norspi_en[0] (O) 2, {0x14[5], 0x10[5], 0xc[5]} = 4'h010	1, uart_ahb_ rts_n (O) 2, {0x14[5], 0x10[5], 0xc[5]} = 4'h011	1, ssis_en (I) 2, {0x14[5], 0x10[5], 0xc[5]} = 4'h100	
38	SSIOEN1	1, GPIO (IO) 2, {0x14[6], 0x10[6], 0xc[6]} = 4'h000	1, ssi0_en1 (O) 2, {0x14[6], 0x10[6], 0xc[6]} = 4'h001 3, rct_ahb_ spi_boot = ((POC[10] == 0) && (POC[5:4] == 3))	1, norspi_en[1] (O) 2, {0x14[6], 0x10[6], 0xc[6]} = 4'h010			

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
39	UART0RX	1, GPIO (IO) 2, {0x14[7], 0x10[7], 0xc[7]} = 4'h000	1, uart0rx (I) 2, {0x14[7], 0x10[7], 0xc[7]} = 4'h001	1, uart_ahb_rx (I) 2, {0x14[7], 0x10[7], 0xc[7]} = 4'h010			
40	UART0TX	1, GPIO (IO) 2, {0x14[8], 0x10[8], 0xc[8]} = 4'h000	1, uart0tx (O) 2, {0x14[8], 0x10[8], 0xc[8]} = 4'h001	1, uart_ahb_tx (O) 2, {0x14[8], 0x10[8], 0xc[8]} = 4'h010			
41	I2S_CLK	1, GPIO (IO) 2, {0x14[9], 0x10[9], 0xc[9]} = 4'h000	1, i2s_clk (IO) 2, {0x14[9], 0x10[9], 0xc[9]} = 4'h001				
42	I2S_SI	1, GPIO (IO) 2, {0x14[10], 0x10[10], 0xc[10]} = 4'h000	1, i2s_si (I) 2, {0x14[10], 0x10[10], 0xc[10]} = 4'h001				
43	I2S_SO	1, GPIO (IO) 2, {0x14[11], 0x10[11], 0xc[11]} = 4'h000	1, i2s_so (O) 2, {0x14[11], 0x10[11], 0xc[11]} = 4'h001				
44	I2S_WS	1, GPIO (IO) 2, {0x14[12], 0x10[12], 0xc[12]} = 4'h000	1, i2s_ws (IO) 2, {0x14[12], 0x10[12], 0xc[12]} = 4'h001				
46	ENET_TXEN	1, GPIO (IO) 2, {0x14[14], 0x10[14], 0xc[14]} = 4'h000 1, enet_txen (O) 2, {0x14[14], 0x10[14], 0xc[14]} = 4'h001 3, rct_ahb_rmii_sel =((POC[0] == 1) && (POC[6] == 1))	1, sc_a0 (O) 2, {0x14[14], 0x10[14], 0xc[14]} = 4'h001 3, rct_ahb_rmii_sel =((POC[0] == 1) && (POC[6] == 1))	1, enet_txen (O) 2, {0x14[14], 0x10[14], 0xc[14]} = 4'h011 3, rct_ahb_mii_sel =((POC[0] == 1) && (POC[6] == 0))	1, ssi1_sclk (O) 2, {0x14[14], 0x10[14], 0xc[14]} = 4'h100	1, norspi_clk (O) 2, {0x14[14], 0x10[14], 0xc[14]} = 4'h101	
47	ENET_TXD_O	1, GPIO (IO) 2, {0x14[15], 0x10[15], 0xc[15]} = 4'h000 1, enet_txd_0 (O) 2, {0x14[15], 0x10[15], 0xc[15]} = 4'h001 3, rct_ahb_rmii_sel =((POC[0] == 1) && (POC[6] == 1))	1, sc_a1 (O) 2, {0x14[15], 0x10[15], 0xc[15]} = 4'h001 3, rct_ahb_mii_sel =((POC[0] == 1) && (POC[6] == 0))	1, enet_txd_0 (O) 2, {0x14[15], 0x10[15], 0xc[15]} = 4'h011 3, rct_ahb_mii_sel =((POC[0] == 1) && (POC[6] == 0))	1, ssi1_txd (O+) 2, {0x14[15], 0x10[15], 0xc[15]} = 4'h100	1, norspi_dq[0] (IO) 2, {0x14[15], 0x10[15], 0xc[15]} = 4'h101	

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
48	ENET_TXD_1	1, GPIO (IO) 2, {0x14[16], 0x10[16], 0xc[16]} = 4'h000	1, enet_txd_1 (O) 2, {0x14[16], 0x10[16], 0xc[16]} = 4'h001 3, rct_ahb_ rmii_sel = ((POC[0] == 1) && (POC[6] == 1))	1, sc_a2 (O) 2, {0x14[16], 0x10[16], 0xc[16]} = 4'h010	1, enet_txd_1 (O) 2, {0x14[16], 0x10[16], 0xc[16]} = 4'h011 3, rct_ahb_mii_ sel = ((POC[0] == 1) & (POC[6] == 0))	1, ssi1_rxd (I) 2, {0x14[16], 0x10[16], 0xc[16]} = 4'h100	1, norspi_dq[1] (IO) 2, {0x14[16], 0x10[16], 0xc[16]} = 4'h101
49	ENET_RXD_0	1, GPIO (IO) 2, {0x14[17], 0x10[17], 0xc[17]} = 4'h000	1, enet_rxd_0 (I) 2, {0x14[17], 0x10[17], 0xc[17]} = 4'h001 3, rct_ahb_ rmii_sel = ((POC[0] == 1) && (POC[6] == 1))	1, sc_a3 (O) 2, {0x14[17], 0x10[17], 0xc[17]} = 4'h010	1, enet_rxd_0 (I) 2, {0x14[17], 0x10[17], 0xc[17]} = 4'h011 3, rct_ahb_mii_ sel = ((POC[0] == 1) & (POC[6] == 0))	1, ssi1_en0 (O) 2, {0x14[17], 0x10[17], 0xc[17]} = 4'h100	1, norspi_en[0] (O) 2, {0x14[17], 0x10[17], 0xc[17]} = 4'h101
50	ENET_RXD_1	1, GPIO (IO) 2, {0x14[18], 0x10[18], 0xc[18]} = 4'h000	1, enet_rxd_1 (I) 2, {0x14[18], 0x10[18], 0xc[18]} = 4'h001 3, rct_ahb_ rmii_sel = ((POC[0] == 1) && (POC[6] == 1))	1, sc_b0 (O) 2, {0x14[18], 0x10[18], 0xc[18]} = 4'h010	1, enet_rxd_1 (I) 2, {0x14[18], 0x10[18], 0xc[18]} = 4'h011 3, rct_ahb_mii_ sel = ((POC[0] == 1) & (POC[6] == 0))	1, ssi1_en1 (O) 2, {0x14[18], 0x10[18], 0xc[18]} = 4'h100	1, norspi_en[1] (O) 2, {0x14[18], 0x10[18], 0xc[18]} = 4'h101
51	ENET_RX_ER	1, GPIO (IO) 2, {0x14[19], 0x10[19], 0xc[19]} = 4'h000	1, enet_rxer (I) 2, {0x14[19], 0x10[19], 0xc[19]} = 4'h001 3, rct_ahb_ rmii_sel = ((POC[0] == 1) && (POC[6] == 1))	1, sc_b1 (O) 2, {0x14[19], 0x10[19], 0xc[19]} = 4'h010	1, enet_rxer (I) 2, {0x14[19], 0x10[19], 0xc[19]} = 4'h011 3, rct_ahb_mii_ sel = ((POC[0] == 1) & (POC[6] == 0))	1, ssi1_en2 (O) 2, {0x14[19], 0x10[19], 0xc[19]} = 4'h100	1, norspi_en[2] (O) 2, {0x14[19], 0x10[19], 0xc[19]} = 4'h101
52	ENET_CRS_DV	1, GPIO (IO) 2, {0x14[20], 0x10[20], 0xc[20]} = 4'h000	1, enet_crs_dv (I) 2, {0x14[20], 0x10[20], 0xc[20]} = 4'h001 3, rct_ahb_ rmii_sel = ((POC[0] == 1) && (POC[6] == 1))	1, sc_b2 (O) 2, {0x14[20], 0x10[20], 0xc[20]} = 4'h010	1, enet_crs_dv (I) 2, {0x14[20], 0x10[20], 0xc[20]} = 4'h011 3, rct_ahb_mii_ sel = ((POC[0] == 1) & (POC[6] == 0))	1, ssi1_en3 (O) 2, {0x14[20], 0x10[20], 0xc[20]} = 4'h100	1, norspi_dq[2] (IO) 2, {0x14[20], 0x10[20], 0xc[20]} = 4'h101

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
53	ENET_REF_CLK	1, GPIO (IO) 2, {0x14[21], 0x10[21], 0xc[21]} = 4'h000	1, enet_ref_clk (O) 2, {0x14[21], 0x10[21], 0xc[21]} = 4'h001 3, rct_ahb_rmii_sel = ((POC[0] == 1) && (POC[6] == 1))	1, sc_b3 (O) 2, {0x14[21], 0x10[21], 0xc[21]} = 4'h010	1, enet_rx_clk (I) 2, {0x14[21], 0x10[21], 0xc[21]} = 4'h011 3, rct_ahb_mii_sel = ((POC[0] == 1) && (POC[6] == 0))		1, norspi_dq[3] (IO) 2, {0x14[21], 0x10[21], 0xc[21]} = 4'h101
54	WP	1, GPIO (IO) 2, {0x14[22], 0x10[22], 0xc[22]} = 4'h000		1, nand_wp (O) 2, {0x14[22], 0x10[22], 0xc[22]} = 4'h010 3, rct_ahb_nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))			
55	SMIO_0	1, GPIO (IO) 2, {0x14[23], 0x10[23], 0xc[23]} = 4'h000		1, nand_ce (O) 2, {0x14[23], 0x10[23], 0xc[23]} = 4'h010 3, rct_ahb_nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_clk (O) 2, {0x14[23], 0x10[23], 0xc[23]} = 4'h011 3, rct_ahb_norspi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		
56	SMIO_1	1, GPIO (IO) 2, {0x14[24], 0x10[24], 0xc[24]} = 4'h000		1, nand_rb (I) 2, {0x14[24], 0x10[24], 0xc[24]} = 4'h010 3, rct_ahb_nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_dq[4] (IO) 2, {0x14[24], 0x10[24], 0xc[24]} = 4'h011 3, rct_ahb_norspi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
57	SMIO_2	1, GPIO (IO) 2, {0x14[25], 0x10[25], 0xc[25]} = 4'h000		1, sd_clk (O) 2, {0x14[25], 0x10[25], 0xc[25]} = 4'h010 3, rct_ahb_ mmc_boot = ((POC[10] == 0) && (POC[5:4] == 2))			
58	SMIO_3	1, GPIO (IO) 2, {0x14[26], 0x10[26], 0xc[26]} = 4'h000		1, sd_cmd (IO) 2, {0x14[26], 0x10[26], 0xc[26]} = 4'h010 3, rct_ahb_ mmc_boot = ((POC[10] == 0) && (POC[5:4] == 2))			
59	SMIO_4	1, GPIO (IO) 2, {0x14[27], 0x10[27], 0xc[27]} = 4'h000		1, sd_cd (I) 2, {0x14[27], 0x10[27], 0xc[27]} = 4'h010 3, rct_ahb_ mmc_boot = ((POC[10] == 0) && (POC[5:4] == 2))			
60	SMIO_5	1, GPIO (IO) 2, {0x14[28], 0x10[28], 0xc[28]} = 4'h000		1, sd_wp (I) 2, {0x14[28], 0x10[28], 0xc[28]} = 4'h010 3, rct_ahb_ mmc_boot = ((POC[10] == 0) && (POC[5:4] == 2))			

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
61	SMIO_6	1, GPIO (IO) 2, {0x14[29], 0x10[29], 0xc[29]} = 4'h000		1, nand_re (O) 2, {0x14[29], 0x10[29], 0xc[29]} = 4'h010 3, rct_ahb_nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_dq[5] (IO) 2, {0x14[29], 0x10[29], 0xc[29]} = 4'h011 3, rct_ahb_norspi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		
62	SMIO_7	1, GPIO (IO) 2, {0x14[30], 0x10[30], 0xc[30]} = 4'h000		1, nand_we (O) 2, {0x14[30], 0x10[30], 0xc[30]} = 4'h010 3, rct_ahb_nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_dq[6] (IO) 2, {0x14[30], 0x10[30], 0xc[30]} = 4'h011 3, rct_ahb_norspi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		
63	SMIO_8	1, GPIO (IO) 2, {0x14[31], 0x10[31], 0xc[31]} = 4'h000		1, nand_ale (O) 2, {0x14[31], 0x10[31], 0xc[31]} = 4'h010 3, rct_ahb_nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_dq[7] (IO) 2, {0x14[31], 0x10[31], 0xc[31]} = 4'h011 3, rct_ahb_norspi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		
64	SMIO_9	1, GPIO (IO) 2, {0x20[0], 0x1c[0], 0x18[0]} = 4'h000		1, nand_d[0] (O) 2, {0x20[0], 0x1c[0], 0x18[0]} = 4'h010 3, rct_ahb_nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_en[0] (O) 2, {0x20[0], 0x1c[0], 0x18[0]} = 4'h011 3, rct_ahb_norspi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
65	SMIO_10	1, GPIO (IO) 2, {0x20[1], 0x1c[1], 0x18[1]} = 4'h000		1, nand_d[1] (IO) 2, {0x20[1], 0x1c[1], 0x18[1]} = 4'h010 3, rct_ahb_ nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_en[1] (O) 2, {0x20[1], 0x1c[1], 0x18[1]} = 4'h011 3, rct_ahb_nor- spi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		
66	SMIO_11	1, GPIO (IO) 2, {0x20[2], 0x1c[2], 0x18[2]} = 4'h000		1, nand_d[2] (IO) 2, {0x20[2], 0x1c[2], 0x18[2]} = 4'h010 3, rct_ahb_ nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_en[2] (O) 2, {0x20[2], 0x1c[2], 0x18[2]} = 4'h011 3, rct_ahb_nor- spi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		
67	SMIO_12	1, GPIO (IO) 2, {0x20[3], 0x1c[3], 0x18[3]} = 4'h000		1, nand_d[3] (IO) 2, {0x20[3], 0x1c[3], 0x18[3]} = 4'h010 3, rct_ahb_ nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_en[3] (O) 2, {0x20[3], 0x1c[3], 0x18[3]} = 4'h011 3, rct_ahb_nor- spi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		
68	SMIO_13	1, GPIO (IO) 2, {0x20[4], 0x1c[4], 0x18[4]} = 4'h000		1, nand_d[4] (IO) 2, {0x20[4], 0x1c[4], 0x18[4]} = 4'h010 3, rct_ahb_ nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_dq[0] (IO) 2, {0x20[4], 0x1c[4], 0x18[4]} = 4'h011 3, rct_ahb_nor- spi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
69	SMIO_14	1, GPIO (IO) 2, {0x20[5], 0x1c[5], 0x18[5]} = 4'h000		1, nand_d[5] (IO) 2, {0x20[5], 0x1c[5], 0x18[5]} = 4'h010 3, rct_ahb_ nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_dq[1] (IO) 2, {0x20[5], 0x1c[5], 0x18[5]} = 4'h011 3, rct_ahb_nor- spi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		
70	SMIO_15	1, GPIO (IO) 2, {0x20[6], 0x1c[6], 0x18[6]} = 4'h000		1, nand_d[6] (IO) 2, {0x20[6], 0x1c[6], 0x18[6]} = 4'h010 3, rct_ahb_ nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_dq[2] (IO) 2, {0x20[6], 0x1c[6], 0x18[6]} = 4'h011 3, rct_ahb_nor- spi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		
71	SMIO_16	1, GPIO (IO) 2, {0x20[7], 0x1c[7], 0x18[7]} = 4'h000		1, nand_d[7] (IO) 2, {0x20[7], 0x1c[7], 0x18[7]} = 4'h010 3, rct_ahb_ nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))	1, norspi_dq[3] (IO) 2, {0x20[7], 0x1c[7], 0x18[7]} = 4'h011 3, rct_ahb_nor- spi_boot_1 = ((POC[10] == 0) && (POC[5:4] == 0) && (POC[14] == 1))		
72	SMIO_17	1, GPIO (IO) 2, {0x20[8], 0x1c[8], 0x18[8]} = 4'h000		1, nand_cle (O) 2, {0x20[8], 0x1c[8], 0x18[8]} = 4'h010 3, rct_ahb_ nand_boot = ((POC[10] == 0) && (POC[5:4] == 1))			

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
73	SMIO_18	1, GPIO (IO) 2, {0x20[9], 0x1c[9], 0x18[9]} = 4'h000		1, sd_d[0] (IO) 2, {0x20[9], 0x1c[9], 0x18[9]} = 4'h010 3, rct_ahb_ mmc_boot = ((POC[10] == 0) && (POC[5:4] == 2))			
74	SMIO_19	1, GPIO (IO) 2, {0x20[10], 0x1c[10], 0x18[10]} = 4'h000		1, sd_d[1] (IO) 2, {0x20[10], 0x1c[10], 0x18[10]} = 4'h010 3, rct_ahb_ mmc_boot = ((POC[10] == 0) && (POC[5:4] == 2))			
75	SMIO_20	1, GPIO (IO) 2, {0x20[11], 0x1c[11], 0x18[11]} = 4'h000		1, sd_d[2] (IO) 2, {0x20[11], 0x1c[11], 0x18[11]} = 4'h010 3, rct_ahb_ mmc_boot = ((POC[10] == 0) && (POC[5:4] == 2))			
76	SMIO_21	1, GPIO (IO) 2, {0x20[12], 0x1c[12], 0x18[12]} = 4'h000		1, sd_d[3] (IO) 2, {0x20[12], 0x1c[12], 0x18[12]} = 4'h010 3, rct_ahb_ mmc_boot = ((POC[10] == 0) && (POC[5:4] == 2))			
77	SMIO_22	1, GPIO (IO) 2, {0x20[13], 0x1c[13], 0x18[13]} = 4'h000		1, sd_d[4] (IO) 2, {0x20[13], 0x1c[13], 0x18[13]} = 4'h010 3, rct_ahb_ mmc_boot = ((POC[10] == 0) && (POC[5:4] == 2))		1, sc_c0 (O) 2, {0x20[13], 0x1c[13], 0x18[13]} = 4'h100	1, ssis_sclk (I) 2, {0x20[13], 0x1c[13], 0x18[13]} = 4'h101

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
78	SMIO_23	1, GPIO (IO) 2, {0x20[14], 0x1c[14], 0x18[14]} = 4'h000		1, sd_d[5] (IO) 2, {0x20[14], 0x1c[14], 0x18[14]} = 4'h010 3, rct_ahb_ mmc_boot = ((POC[10] == 0) && (POC[5:4] == 2))		1, sc_c1 (O) 2, {0x20[14], 0x1c[14], 0x18[14]} = 4'h100	1, ssis_rxd (I) 2, {0x20[14], 0x1c[14], 0x18[14]} = 4'h101
79	SMIO_24	1, GPIO (IO) 2, {0x20[15], 0x1c[15], 0x18[15]} = 4'h000		1, sd_d[6] (IO) 2, {0x20[15], 0x1c[15], 0x18[15]} = 4'h010 3, rct_ahb_ mmc_boot = ((POC[10] == 0) && (POC[5:4] == 2))		1, sc_c2 (O) 2, {0x20[15], 0x1c[15], 0x18[15]} = 4'h100	1, ssis_txd (O+) 2, {0x20[15], 0x1c[15], 0x18[15]} = 4'h101
80	SMIO_25	1, GPIO (IO) 2, {0x20[16], 0x1c[16], 0x18[16]} = 4'h000		1, sd_d[7] (IO) 2, {0x20[16], 0x1c[16], 0x18[16]} = 4'h010 3, rct_ahb_ mmc_boot = ((POC[10] == 0) && (POC[5:4] == 2))		1, sc_c3 (O) 2, {0x20[16], 0x1c[16], 0x18[16]} = 4'h100	1, ssis_en (I) 2, {0x20[16], 0x1c[16], 0x18[16]} = 4'h101
81	SMIO_26	1, GPIO (IO) 2, {0x20[17], 0x1c[17], 0x18[17]} = 4'h000		1, sdio_clk (O) 2, {0x20[17], 0x1c[17], 0x18[17]} = 4'h010			
82	SMIO_27	1, GPIO (IO) 2, {0x20[18], 0x1c[18], 0x18[18]} = 4'h000		1, sdio_cmd (IO) 2, {0x20[18], 0x1c[18], 0x18[18]} = 4'h010			
83	SMIO_28	1, GPIO (IO) 2, {0x20[19], 0x1c[19], 0x18[19]} = 4'h000		1, sdio_d[0] (IO) 2, {0x20[19], 0x1c[19], 0x18[19]} = 4'h010		1, sc_d0 (O) 2, {0x20[19], 0x1c[19], 0x18[19]} = 4'h100	1, ssis_sclk (I) 2, {0x20[19], 0x1c[19], 0x18[19]} = 4'h101
84	SMIO_29	1, GPIO (IO) 2, {0x20[20], 0x1c[20], 0x18[20]} = 4'h000		1, sdio_d[1] (IO) 2, {0x20[20], 0x1c[20], 0x18[20]} = 4'h010		1, sc_d1 (O) 2, {0x20[20], 0x1c[20], 0x18[20]} = 4'h100	1, ssis_rxd (I) 2, {0x20[20], 0x1c[20], 0x18[20]} = 4'h101

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
85	SMIO_30	1, GPIO (IO) 2, {0x20[21], 0x1c[21], 0x18[21]} = 4'h000		1, sdio_d[2] (IO) 2, {0x20[21], 0x1c[21], 0x18[21]} = 4'h010		1, sc_d2 (O) 2, {0x20[21], 0x1c[21], 0x18[21]} = 4'h100	1, ssis_txd (O+) 2, {0x20[21], 0x1c[21], 0x18[21]} = 4'h101
86	SMIO_31	1, GPIO (IO) 2, {0x20[22], 0x1c[22], 0x18[22]} = 4'h000		1, sdio_d[3] (IO) 2, {0x20[22], 0x1c[22], 0x18[22]} = 4'h010		1, sc_d3 (O) 2, {0x20[22], 0x1c[22], 0x18[22]} = 4'h100	1, ssis_en (I) 2, {0x20[22], 0x1c[22], 0x18[22]} = 4'h101
87	SMIO_32	1, GPIO (IO) 2, {0x20[23], 0x1c[23], 0x18[23]} = 4'h000		1, sdio_cd (I) 2, {0x20[23], 0x1c[23], 0x18[23]} = 4'h010			
88	SMIO_33	1, GPIO (IO) 2, {0x20[24], 0x1c[24], 0x18[24]} = 4'h000		1, sdio_wp (I) 2, {0x20[24], 0x1c[24], 0x18[24]} = 4'h010			
89	HPD	1, GPIO (IO) 2, {0x20[25], 0x1c[25], 0x18[25]} = 4'h000	1, hdmitx_hpd (I) 2, {0x20[25], 0x1c[25], 0x18[25]} = 4'h001 3, gpio_de- fault_for_hd- mitx_hpd_cec = tie 1				
90	CEC	1, GPIO (IO) 2, {0x20[26], 0x1c[26], 0x18[26]} = 4'h000	1, hdmitx_cec (IO) 2, {0x20[26], 0x1c[26], 0x18[26]} = 4'h001 3, gpio_de- fault_for_hd- mitx_hpd_cec = tie 1	1, enet_2nd_- ref_clk (O) 2, {0x20[26], 0x1c[26], 0x18[26]} = 4'h010			
91	SVSYNC	1, GPIO (IO) 2, {0x20[27], 0x1c[27], 0x18[27]} = 4'h000	1, vin_svsync (O) 2, {0x20[27], 0x1c[27], 0x18[27]} = 4'h001 3, rct_idsp_ vin_sel = tie 0	1, idsp_pip_io- pad_master_- hsync (O) 2, {0x20[27], 0x1c[27], 0x18[27]} = 4'h010			

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
92	SHSYNC	1, GPIO (IO) 2, {0x20[28], 0x1c[28], 0x18[28]} = 4'h000	1, vin_shsync (O) 2, {0x20[28], 0x1c[28], 0x18[28]} = 4'h001 3, rct_idspl vin_sel = tie 0	1, idsp_pip_io- pad_master_ vsync (O) 2, {0x20[28], 0x1c[28], 0x18[28]} = 4'h010			
93	VDO_OUT_0	1, GPIO (IO) 2, {0x20[29], 0x1c[29], 0x18[29]} = 4'h000	1, vd0_out[0] (O+) 2, {0x20[29], 0x1c[29], 0x18[29]} = 4'h001				
94	VDO_OUT_1	1, GPIO (IO) 2, {0x20[30], 0x1c[30], 0x18[30]} = 4'h000	1, vd0_out[1] (O+) 2, {0x20[30], 0x1c[30], 0x18[30]} = 4'h001				
95	VDO_OUT_2	1, GPIO (IO) 2, {0x20[31], 0x1c[31], 0x18[31]} = 4'h000	1, vd0_out[2] (O+) 2, {0x20[31], 0x1c[31], 0x18[31]} = 4'h001				
96	VDO_OUT_3	1, GPIO (IO) 2, {0x2c[0], 0x28[0], 0x24[0]} = 4'h000	1, vd0_out[3] (O+) 2, {0x2c[0], 0x28[0], 0x24[0]} = 4'h001				
97	VDO_OUT_4	1, GPIO (IO) 2, {0x2c[1], 0x28[1], 0x24[1]} = 4'h000	1, vd0_out[4] (O+) 2, {0x2c[1], 0x28[1], 0x24[1]} = 4'h001				
98	VDO_OUT_5	1, GPIO (IO) 2, {0x2c[2], 0x28[2], 0x24[2]} = 4'h000	1, vd0_out[5] (O+) 2, {0x2c[2], 0x28[2], 0x24[2]} = 4'h001				
99	VDO_OUT_6	1, GPIO (IO) 2, {0x2c[3], 0x28[3], 0x24[3]} = 4'h000	1, vd0_out[6] (O+) 2, {0x2c[3], 0x28[3], 0x24[3]} = 4'h001				
100	VDO_OUT_7	1, GPIO (IO) 2, {0x2c[4], 0x28[4], 0x24[4]} = 4'h000	1, vd0_out[7] (O+) 2, {0x2c[4], 0x28[4], 0x24[4]} = 4'h001				

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
101	VDO_OUT_8	1, GPIO (IO) 2, {0x2c[5], 0x28[5], 0x24[5]} = 4'h000	1, vd0_out[8] (O+) 2, {0x2c[5], 0x28[5], 0x24[5]} = 4'h001				
102	VDO_OUT_9	1, GPIO (IO) 2, {0x2c[6], 0x28[6], 0x24[6]} = 4'h000	1, vd0_out[9] (O+) 2, {0x2c[6], 0x28[6], 0x24[6]} = 4'h001				
103	VDO_OUT_10	1, GPIO (IO) 2, {0x2c[7], 0x28[7], 0x24[7]} = 4'h000	1, vd0_out[10] (O+) 2, {0x2c[7], 0x28[7], 0x24[7]} = 4'h001				
104	VDO_OUT_11	1, GPIO (IO) 2, {0x2c[8], 0x28[8], 0x24[8]} = 4'h000	1, vd0_out[11] (O+) 2, {0x2c[8], 0x28[8], 0x24[8]} = 4'h001				
105	VDO_OUT_12	1, GPIO (IO) 2, {0x2c[9], 0x28[9], 0x24[9]} = 4'h000	1, vd0_out[12] (O+) 2, {0x2c[9], 0x28[9], 0x24[9]} = 4'h001				
106	VDO_OUT_13	1, GPIO (IO) 2, {0x2c[10], 0x28[10], 0x24[10]} = 4'h000	1, vd0_out[13] (O+) 2, {0x2c[10], 0x28[10], 0x24[10]} = 4'h001				
107	VDO_OUT_14	1, GPIO (IO) 2, {0x2c[11], 0x28[11], 0x24[11]} = 4'h000	1, vd0_out[14] (O+) 2, {0x2c[11], 0x28[11], 0x24[11]} = 4'h001				
108	VDO_OUT_15	1, GPIO (IO) 2, {0x2c[12], 0x28[12], 0x24[12]} = 4'h000	1, vd0_out[15] (O+) 2, {0x2c[12], 0x28[12], 0x24[12]} = 4'h001				
109	VDO_CLK	1, GPIO (IO) 2, {0x2c[13], 0x28[13], 0x24[13]} = 4'h000	1, vd0_clk (O+) 2, {0x2c[13], 0x28[13], 0x24[13]} = 4'h001				

GPIO	Pin Name	Multiplexed Function					
		Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
110	VDO_VSYNC	1, GPIO (IO) 2, {0x2c[14], 0x28[14], 0x24[14]} = 4'h000	1, vd0_vsync (O+) 2, {0x2c[14], 0x28[14], 0x24[14]} = 4'h001				
111	VDO_HSYNC	1, GPIO (IO) 2, {0x2c[15], 0x28[15], 0x24[15]} = 4'h000	1, vd0_hsync (O+) 2, {0x2c[15], 0x28[15], 0x24[15]} = 4'h001				
112	VDO_HVLD	1, GPIO (IO) 2, {0x2c[16], 0x28[16], 0x24[16]} = 4'h000	1, vd0_hvld (O+) 2, {0x2c[16], 0x28[16], 0x24[16]} = 4'h001				
113	VD_PWM	1, GPIO (IO) 2, {0x2c[17], 0x28[17], 0x24[17]} = 4'h000	1, pwm_0 (O) 2, {0x2c[17], 0x28[17], 0x24[17]} = 4'h001				

Table 15-15. GPIO Function Selection: Register Bit Locations.

The bit locations in each register set must be programmed as a whole; i.e., treat the 3 bits in the same location on each of the three registers as a 3-bit storage. Refer to the following example.

Assume that the goal is to utilize function 0x1 on pin 0. In order to accomplish this, the first bit location of register set 0 must be programmed to 0x1 as follows:

1. Load `iomux_reg0_0`.
2. OR `iomux_reg0_0` with 0x1.
3. Store value back to `iomux_reg0_0`.
4. Load `iomux_reg0_1`.
5. AND `iomux_reg0_1` with 0xFFFF.
6. Store value back to `iomux_reg0_1`.
7. Load `iomux_reg0_2`.
8. AND `iomux_reg0_2` with 0xFFFF.
9. Store value back to `iomux_reg0_2`.
10. Store 1 to `iomux_ctrl_set`.

11. Store 0 to `iomux_ctrl_set`.

In summary, bit 0 of **IOMUX** register set 0 can be jointly seen as {0, 0, 1}. Note that the configuration will not take effect without the completion of steps 10 and 11.

For Confidential
For HAOTEX Only

16. SSI / SPI

This chapter contains register programming information for the A12 Serial Synchronous / Serial Peripheral Interfaces (SSI / SPI). The chapter is organized as follows:

- [\(Section 16.1\) SSI / SPI: Overview](#)
- [\(Section 16.2\) SSI / SPI: Programming Notes](#)
- [\(Section 16.3\) SSI / SPI: Registers](#)
- [\(Section 16.4\) SSI / SPI: NOR-SPI Programming and Register Information](#)

16.1 SSI / SPI: Overview

The A12 chip provides two SSI / SPI masters, providing up to eight device enables in total, and one dedicated slave. The SSI / SPI interface provides interrupt capability, various transfer modes, and serial master operation. The clock speed and device-enable polarity can be changed with register programming. The A12 provides Motorola interface capability.

In addition, the A12 chip includes a 4-bit NOR-SPI Flash controller, which features asynchronous FIFO, synchronous FIFO, a clock divider, and a two-input AHB input arbiter.

16.1.1 Overview: Master / Slave Relationship

The figure below shows the relationship between a serial master (left) and two slave devices (right) located on the serial bus.

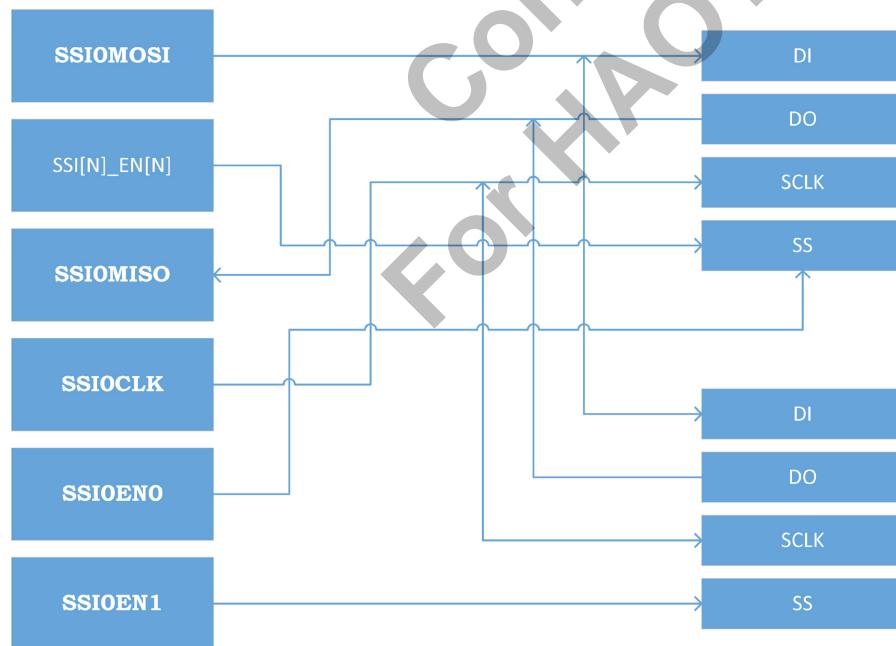


Figure 16-1. SPI Interface with the SSI / SPI as the Serial Master and Two Slave Devices on the Serial Bus.

Note that the pins used to configure the SSI / SPI interfaces are muxed with general purpose input / output (GPIO) functions.

16.1.2 Overview: Interfaces and Device Enables

The A12 SSI / SPI master and slave interfaces are located on the AHB bus and maintain their own register spaces. The following table provides general information on the SSI / SPI interfaces.

Port	Base Address	DMA	FIFO	Baud Rate	Enables	Device Enable Pins	SSI/SPI Function
SSI0	0xE002.0000	Yes	32 16-bit entries	20 MHz	4	SSIOEN0	ssi0_en0 Device Enable
						SSIOEN1	ssi0_en1 Device Enable
						SC_E0	ssi0_en2 Device Enable
						TIMER2	ssi0_en3 Device Enable
SSI1	0xE002.1000	Yes	32 16-bit entries	20 MHz	4	ENET_RXD_0 or SC_A3	ssi1_en0 Device Enable
						ENET_RXD_1 or SC_B0	ssi1_en1 Device Enable
						ENET_RX_ER or SC_B1	ssi1_en2 Device Enable
						ENET_CRS_DV or SC_B2	ssi1_en3 Device Enable
SSIS0 (Slave)	0xE002.6000	Yes	32 16-bit entries	20 MHz		Not Applicable	

Table 16-1. SSI / SPI Interface Overview Information.

Pins related to the NOR-SPI controller are listed along with their associated functions in the table below.

NOR-SPI Pins	Function
SC_A0 SSIOCLK ENET_TXEN SMIO_0	NOR-SPI Clock
SC_A1 SSIOMOSI ENET_RXD_0 SMIO_13	NOR-SPI Data Lane 0
SC_A2 SSIOMISO ENET_RXD_1 SMIO_14	NOR-SPI Data Lane 1
SC_A3 SC_B1 IDC2CLK ENET_CRS_DV SMIO_15	NOR-SPI Data Lane 2
SC_B0 SC_B2 IDC2DATA ENET_REF_CLK SMIO_16	NOR-SPI Data Lane 3
SMIO_1	NOR-SPI Data Lane 4
SMIO_6	NOR-SPI Data Lane 5
SMIO_7	NOR-SPI Data Lane 6

NOR-SPI Pins	Function
SMIO_8	NOR-SPI Data Lane 7
SC_B1 SSIOEN0 ENET_RXD_0 SMIO_9	norspi_en[0] Device Enable
SC_B2 SSIOEN1 ENET_RXD_1 SMIO_10	norspi_en[1] Device Enable
SC_B3 IDC2CLK ENET_RX_ER SMIO_11	norspi_en[2] Device Enable
SC_E0 IDC2DATA SMIO_12	norspi_en[3] Device Enable

Table 16-2. NOR-SPI Pins and Functions.

For Confidential
HAOTEX Only

16.2 SSI / SPI: Programming Notes

This section provides programming details for the A12 SSI / SPI interface as follows:

- ([Section 16.2.1](#)) SSI / SPI Clocking and Timing Diagrams
- ([Section 16.2.2](#)) SSI / SPI Transmit and Receive FIFO Buffers
- ([Section 16.2.3](#)) SSI / SPI Interrupts
- ([Section 16.2.4](#)) SSI / SPI Transfer Modes
- ([Section 16.2.5](#)) SSI / SPI Operational Modes

16.2.1 SSI / SPI Clocking and Timing Diagrams

The SSI / SPI serial bit clock is determined using the clock source (GCLK_SSI for the SSI master, GCLK_SSI2 for the SSI slave, or GCLK_SSI3 for NOR-SPI) along with the clock divider register at register **SSSP_baudr** (offset 0x260). The maximum frequency of the SSI / SPI bit-rate clock is restricted to allow the shift-control logic to capture data on one clock edge and propagate data on the opposite edge.

The frequency of an SSI / SPI bit clock can be derived from the following equation:

$$\text{Freq} = \text{Freq}(\text{CLK_IN}) / \text{sckdv}$$

where **sckdv** is the **SSSP_baudr** bit field that can hold any even value in the range 0 to 65,534. If **sckdv** is 0, then **SSI[N].CLK** is disabled. See [Section 16.3.2.5 “SSSP_baudr Register”](#) for details.

The clock polarity (**SSSP_ctrlr0** field **scpol**) configuration parameter determines whether the inactive state of the serial clock is high or low. To transmit data, both SSI / SPI peripherals must maintain the identical serial clock phase (**SSSP_ctrlr0** field **scph**) and clock polarity (**SSSP_ctrlr0** field **scpol**) values. The data frame can be 4 to 16 bits in length.

When the configuration parameter **scph** = 0, indicating that the serial clock toggles in the middle of the first data bit, the data transmission begins on the falling edge of the slave select signal. The first data bit is captured by the master and slave peripherals on the first edge of the serial clock; therefore, valid data must be present on the TXD and RXD lines prior to the first serial clock edge. [Figure 16-2](#) shows a timing diagram for a single-frame SSI / SPI data transfer with **scph** = 0. The serial clock is shown for configuration parameters **scpol** = 0, indicating a low inactive state, and **scpol** = 1, indicating a high inactive state.

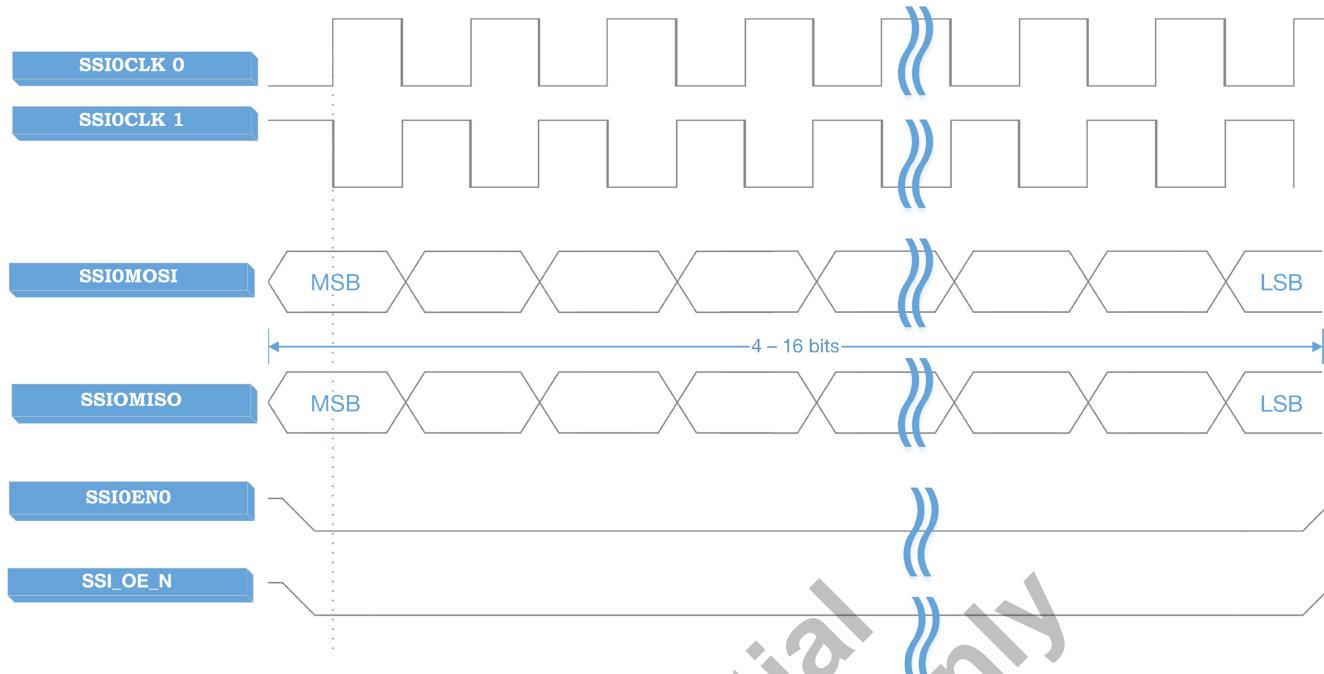


Figure 16-2. SSI/SPI Single-Frame Transfer: Serial Clock Toggles in the Middle of First Data Bit ($scph = 0$).

For continuous data transfers, the slave select signal is required to toggle before the start of the next data frame. This is illustrated in [Figure 16-3](#).

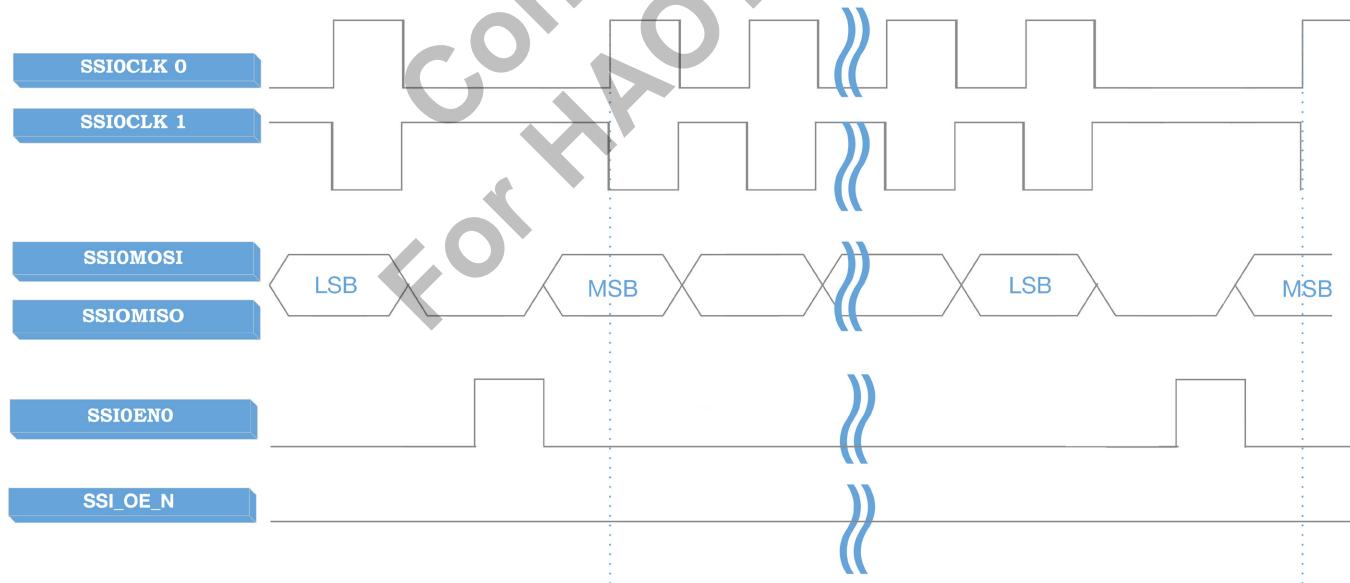


Figure 16-3. SSI/SPI Continuous Transfer: Serial Clock Toggles in the Middle of First Data Bit ($scph = 0$).

When the configuration parameter $scph = 1$, indicating that the serial clock toggles at the start of the first data bit, both master and slave peripherals begin transmitting data on the first serial clock edge after the slave select line is activated. The first data bit is captured on the second (trailing) serial clock edge. Data is propagated by the

master and slave peripherals on the leading edge of the serial clock. [Figure 16-4](#) shows a single-frame SSI / SPI data transfer with $scph = 1$.

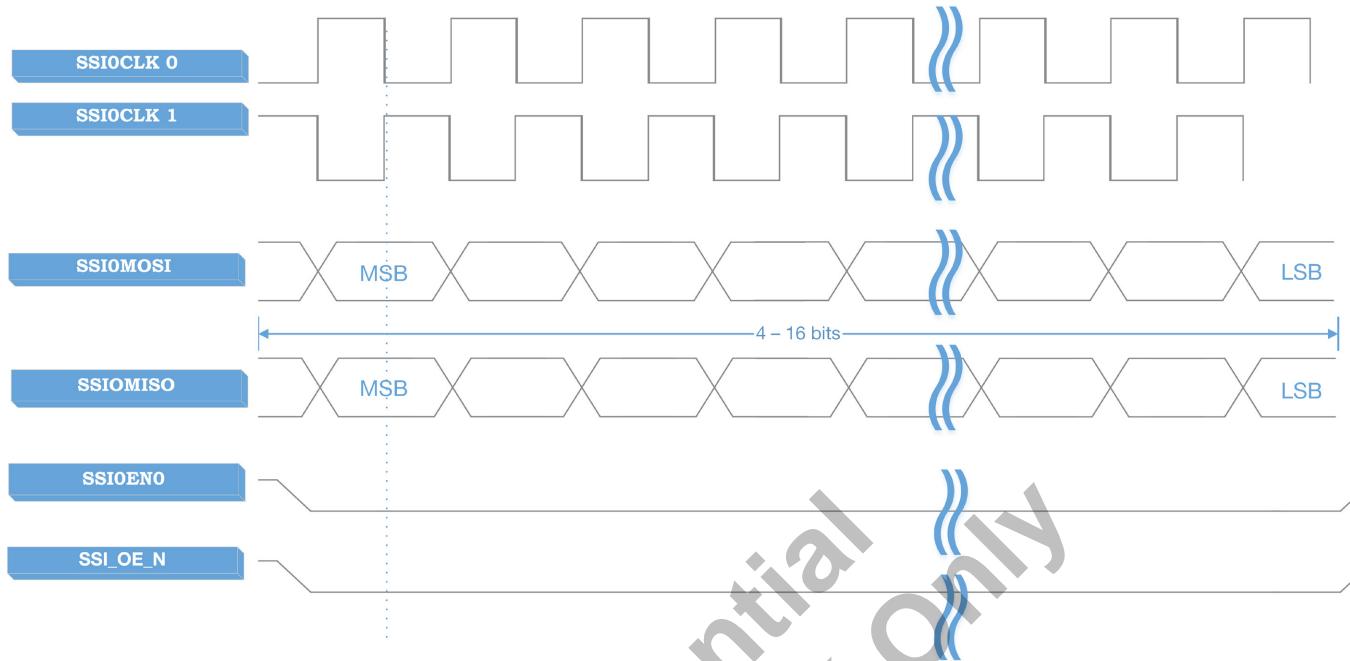


Figure 16-4. SSI/SPI Single-Frame Transfer: Serial Clock Toggles at the Start of First Data Bit ($scph = 1$).

Continuous data frames are transferred in the same way as single frames, with the most significant bit (MSB) of the next frame following directly after the least significant bit (LSB) of the current frame. The slave select signal is held active for the duration of the transfer. [Figure 16-5](#) shows the timing diagram for continuous SSI / SPI transfers when $scph = 1$.

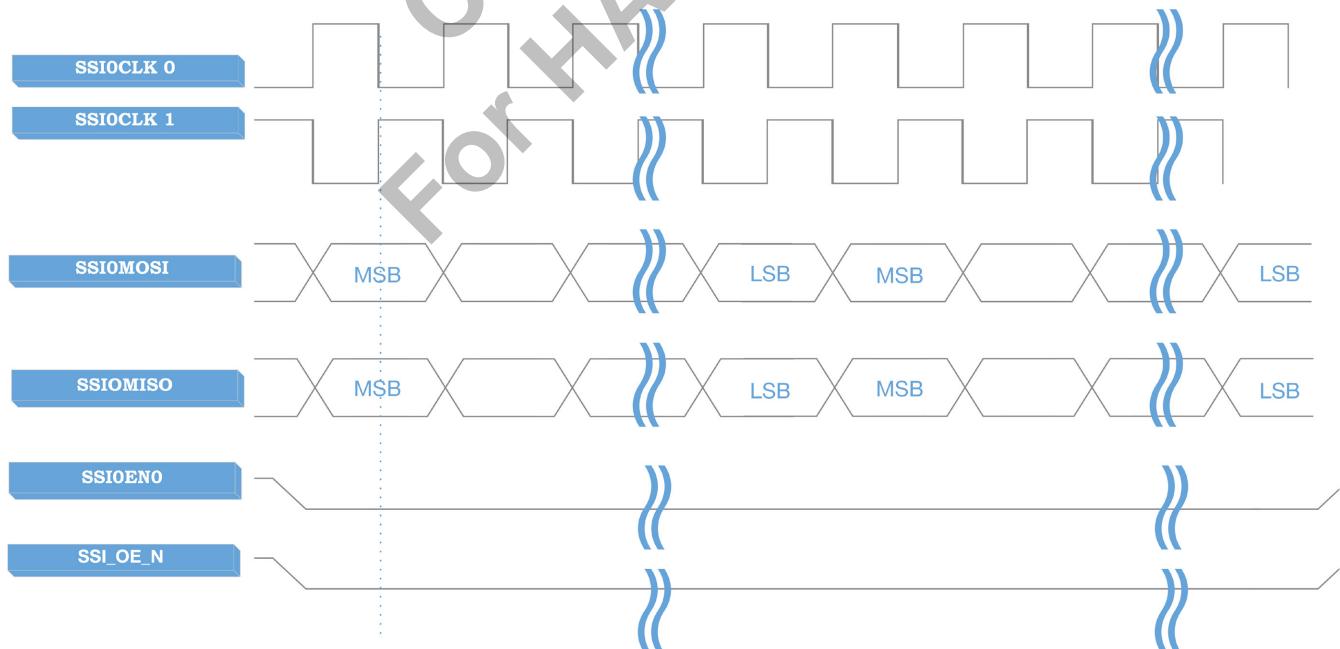


Figure 16-5. SSI/SPI Continuous Transfer: Serial Clock Toggles at the Start of First Data Bit ($scph = 1$).

The signals illustrated in the timing diagrams above include:

- **SSIOCLK** - serial clock from SSI / SPI master:
 - **SSIOCLK 0** and **SSIOCLK 1** represent **scpol = 0** and **scpol = 1**, respectively, with **scpol = 0** indicating a low inactive state, and **scpol = 1** indicating a high inactive state.
- **SSIOENO** - slave select signal from SSI / SPI master
- **SSI_OE_N** - output enable for the SSI / SPI master (internal hardware signal)
- **SSIOMOSI** - transmit data line for the SSI / SPI master
- **SSIOMISO** - receive data line for the SSI / SPI master

For Confidential
For HAOTEX Only

16.2.2 SSI / SPI Transmit and Receive FIFO Buffers

The FIFO buffers used by the SSI / SPI controllers are port register files with a depth of 128 entries (SSI0 and SSI2) and 128 entries (Dedicated Slave).

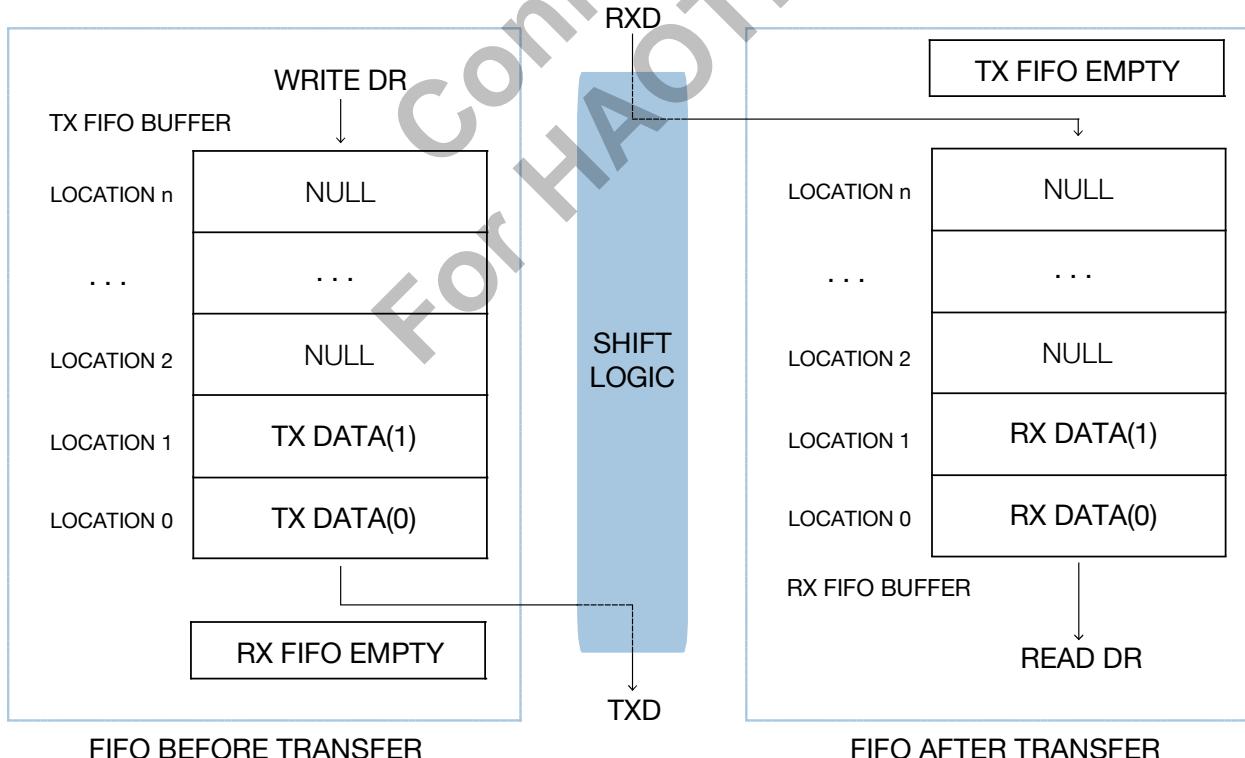
The width of the transmit and receive FIFO buffers is fixed at 32 bits. Data frames less than 32 bits must be right-justified when written into the transmit FIFO buffer. The shift-control logic automatically right-justifies receive data in the receive FIFO buffer. Each data entry in the FIFO buffers contains a single data frame. It is not possible to store multiple data frames in a single FIFO location.

Note that there are three possible transfer modes on the SSI / SPI interface for performing serial transactions (see [Section 16.2.4 "SSI / SPI Transfer Modes"](#)):

1. Transmit and Receive transfer mode
2. Transmit Only transfer mode
3. Receive Only transfer mode

In Transmit and Receive transfer mode (**SSSP_ctrlr0** field **tmod[9:8]** = 0x0), data transmitted from the SSI / SPI to the external serial device is written into the transmit FIFO. Data received from the external serial device into the SSI / SPI is pushed into the receive FIFO.

[Figure 16-6](#) shows the FIFO levels prior to a serial transfer and the FIFO levels on completion of the transfer. In this example, two data words are transmitted from the SSI / SPI to the external serial device in a continuous transfer. The external serial device is responding with two data words.



[Figure 16-6. FIFO Status \(Before and After Transfer\) for Transmit and Receive Mode.](#)

In Transmit Only transfer mode (**SSSP_ctrlr0** field **tmod[9:8]** = 0x1), data transmitted from the SSI / SPI to the external serial device is written into the transmit FIFO. As the data received from the external serial device is considered invalid, it is not stored in the SSI / SPI receive FIFO.

Figure 16-7 shows the FIFO levels prior to the beginning of a serial transfer and the FIFO levels on completion of the transfer. In this example, two data words are transmitted from the SSI / SPI to the external serial device in a continuous transfer.

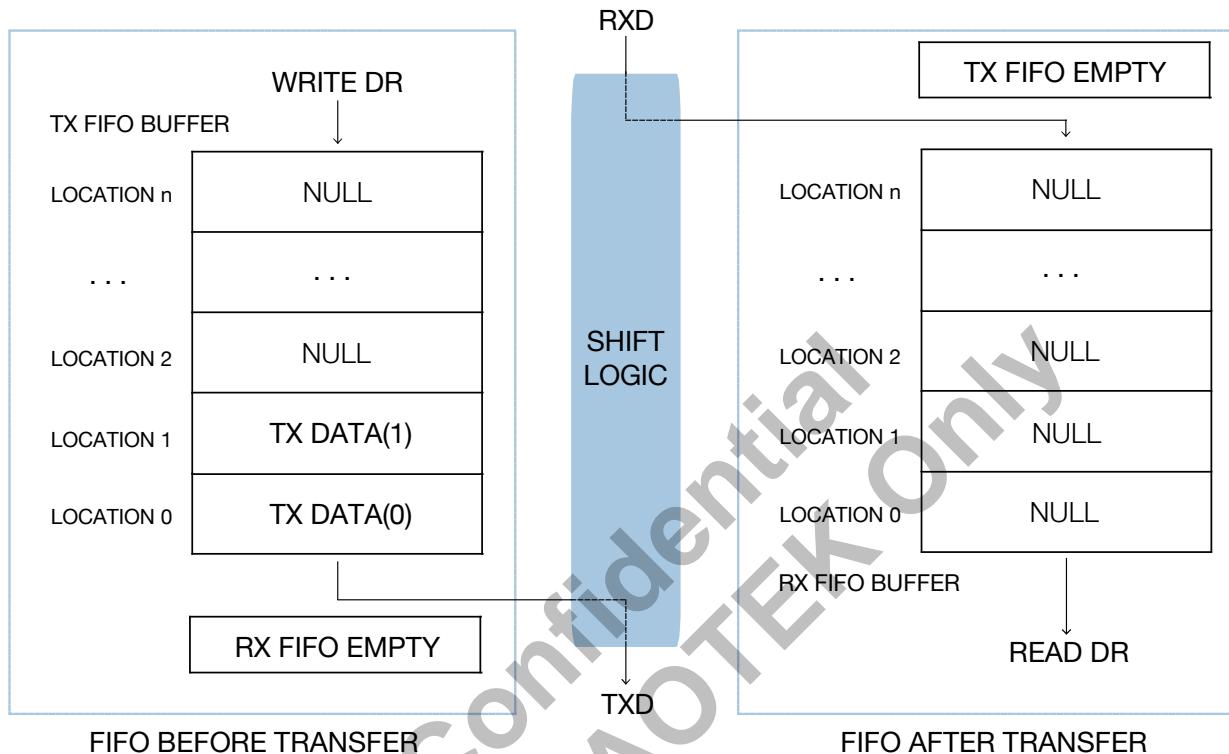


Figure 16-7. FIFO Status (Before and After Transfer) for Transmit Only Mode.

In Receive Only transfer mode (**SSSP_ctrlr0** field **tmod[9:8]** = 0x2), data transmitted from the SSI / SPI to the external serial device is invalid, so a single dummy word is written into the transmit FIFO to begin the serial transfer. This dummy word is re-transmitted by the SSI / SPI for the duration of the transfer. Data received from the external serial device into the SSI / SPI is pushed into the receive FIFO. **Figure 16-8** shows the FIFO levels prior to starting a serial transfer and the FIFO levels on completion of the transfer. In this example, two data words are received by the SSI / SPI from the external serial device in a continuous transfer.

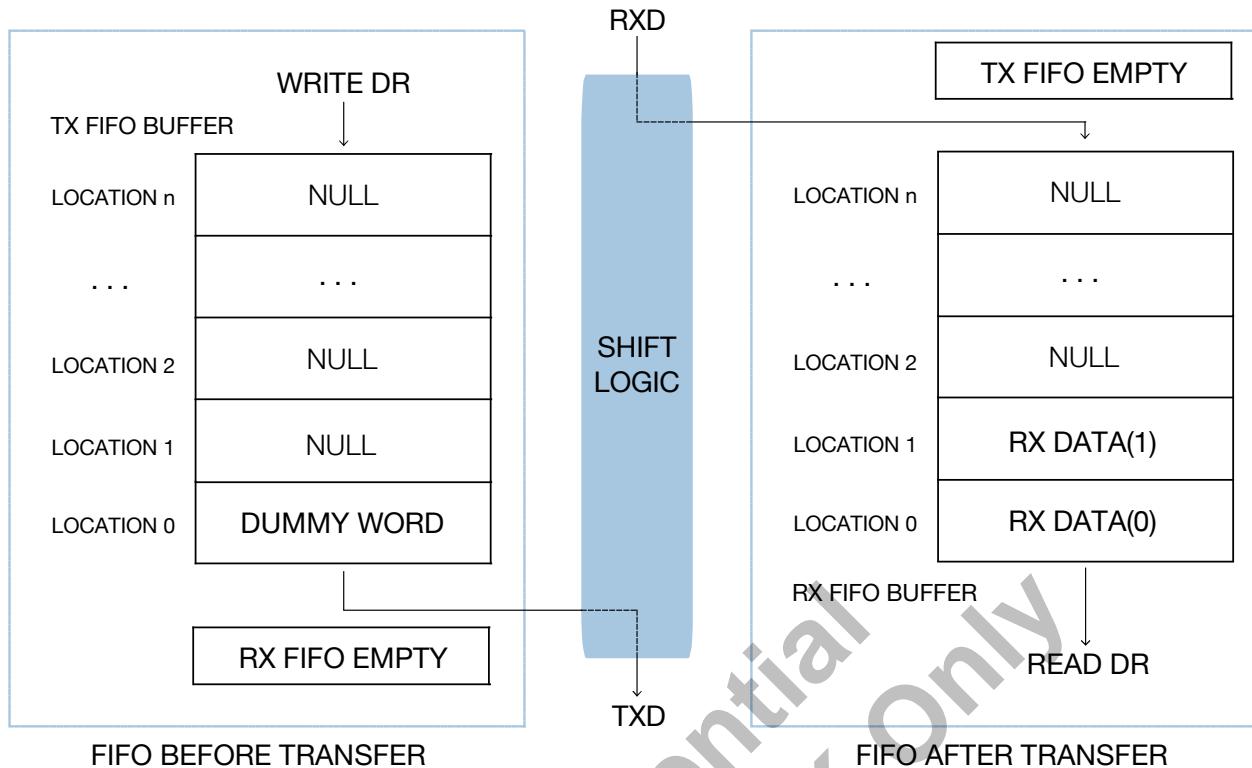


Figure 16-8. FIFO Status (Before and After Transfer) for Receive Only Mode.

Section 16.3 details the SSI / SPI registers. Here, the transmit FIFO is loaded by AHB write commands to the SSI / SPI data register (**SSSP_dr**). Data is popped (removed) from the transmit FIFO by shift-control logic into the transmit shift register. The transmit FIFO generates a FIFO empty interrupt request (SSI_TXE_INTR) when the number of entries in the FIFO is less-than or equal to the FIFO threshold value. The threshold value, set through programmable Transmit FIFO Threshold Level register (**SSSP_txftlr**), determines the number of FIFO entries at which an interrupt is generated. The threshold value allows for sending an early indication to the processor that the transmit FIFO is nearly empty. A transmit FIFO overflow interrupt (SSI_TXO_INTR) is generated if there is an attempt to write data into an already full transmit FIFO.

Data is popped from the receive FIFO by AHB read commands to the SSI / SPI data register (**SSSP_dr**). Transmit data traveling from the system bus to the SSI transmit FIFO must be word aligned. The receive FIFO is loaded from the receive shift register by shift-control logic. The receive FIFO generates a FIFO-full interrupt request (SSI_RXF_INTR) when the number of entries in the FIFO is greater-than or equal to the FIFO threshold value plus 1.

The threshold value, set through the programmable Receive FIFO Threshold Level register (**SSSP_rxftlr**), determines the number of FIFO entries at which an interrupt is generated. A receive FIFO overrun interrupt (SSI_RXO_INTR) is generated when the receive shift-logic attempts to load data into a completely full receive FIFO. However, this newly received data is lost. A receive FIFO underflow interrupt (SSI_RXU_INTR) is generated if there is an attempt to read from an empty receive FIFO. This alerts the processor that the read data is invalid.

16.2.3 SSI / SPI Interrupts

A number of events can trigger an SSI / SPI interrupt, and these individual interrupts are combined into one interrupt request to the Vector Interrupt Controller (VIC). Each individual interrupt request can be masked. The combined interrupt request is the OR'ed result of all other SPI interrupts after masking. The SSI / SPI interrupts are described as follows:

- Transmit FIFO Empty Interrupt (SSI_TXE_INTR): Set when the transmit FIFO is equal to or below its threshold value and requires service to prevent an under-run. The threshold value, set through a software-programmable register, determines the number of transmit FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data is written into the transmit FIFO buffer, bringing it over the threshold level.
- Transmit FIFO Overflow Interrupt (SSI_TXO_INTR): Set when an AHB access attempts to write into the transmit FIFO after it has been completely filled. When set, data written from the AHB is discarded. This interrupt remains set until the software reads the Transmit FIFO Overflow Interrupt Clear register (**SSSP_txoicr**).
- Receive FIFO Full Interrupt (SSI_RXF_INTR): Set when the receive FIFO is equal-to or above its threshold value plus 1 and requires service to prevent an overflow. The threshold value, set through a software-programmable register, determines the number of receive FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data is read from the receive FIFO buffer, bringing it below the threshold level.
- Receive FIFO Overflow Interrupt (SSI_RXO_INTR): Set when the receive logic attempts to place data into the receive FIFO after it has been completely filled. When set, newly received data is discarded. This interrupt remains set until the software reads the Receive FIFO Overflow Interrupt Clear register (**SSSP_rxoicr**).
- Receive FIFO Underflow Interrupt (SSI_RXU_INTR): Set when an AHB access attempts to read from the receive FIFO when it is empty. When set, zeros are read back from the receive FIFO. This interrupt remains set until the software reads the Receive FIFO Underflow Interrupt Clear register (**SSSP_rxuicr**).
- Multi-Master Contention Interrupt (SSI_MST_INTR): This interrupt is only present when the SSI / SPI component is configured as a serial-master device. The interrupt is set when another serial master on the serial bus selects the SSI / SPI master as a serial-slave device and is actively transferring data. This informs the processor of a possible conflict on the serial bus. This interrupt remains set until the software reads the Multi-Master Interrupt Clear register (**SSSP_msticr**).
- Combined Interrupt Request (SSI_INTR): This interrupt is the OR'ed result of all the above interrupt requests after masking. To mask this interrupt signal, the software must mask all other SPI interrupt requests.

16.2.4 SSI / SPI Transfer Modes

The SSI / SPI operates in the following three modes when transferring data on the serial bus:

- ([Section 16.2.4.1](#)) SSI / SPI Transmit and Receive Transfer Mode
- ([Section 16.2.4.2](#)) SSI / SPI Transmit Only Transfer Mode
- ([Section 16.2.4.3](#)) SSI / SPI Receive Only Transfer Mode

The transfer mode is set by writing to the Control register 0 (**SSSP_ctrlr0**) transfer mode field (**tmod**), as described in [Section 16.3.2.1](#).

16.2.4.1 SSI / SPI Transmit and Receive Transfer Mode

When the **SSSP_ctrlr0** transfer mode field (**tmod**) = 0x0 ([Section 16.3.2.1](#)), the SSI / SPI interface transfers data in Transmit and Receive mode. In this mode, both transmit and receive logic are valid, and the data transfer occurs as normal according to the selected frame format. Transmit data is popped (removed) from the transmit FIFO and sent through the TXD line to the target device, which replies with data on the RXD line. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame.

16.2.4.2 SSI / SPI Transmit Only Transfer Mode

When the **SSSP_ctrlr0** transfer mode field (**tmod**) = 0x1 ([Section 16.3.2.1](#)), the SSI / SPI interface transfers data in Transmit Only mode. In this mode, all receive data is invalid and should not be stored in the receive FIFO. Otherwise, the data transfer occurs as normal, according to the selected frame format (serial protocol). Transmit data is popped from the transmit FIFO and sent through the TXD line to the target device, which replies with data on the RXD line. At the end of the data frame, the receive shift register does not load its newly received data into the receive FIFO. Instead, the data in the receive shift register is overwritten by the next transfer. It is required to mask interrupts originating from the receive logic when this mode is used.

16.2.4.3 SSI / SPI Receive Only Transfer Mode

When the **SSSP_ctrlr0** transfer mode field (**tmod**) = 0x2 ([Section 16.3.2.1](#)), the SSI / SPI interface transfers data in Receive Only mode. In this mode, all transmit data is invalid. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame. It is required to mask interrupts originating from the transmit logic when this mode is used.

16.2.5 SSI / SPI Operational Modes

An SSI / SPI bus is capable of maintaining both serial masters and serial slaves, each of which can be leveraged differently under various operational modes. This section details the operational modes of the SSI / SPI interface.

- ([Section 16.2.5.1](#)) Operational Modes: SSI / SPI Serial-Master
- ([Section 16.2.5.2](#)) Operational Modes: Master SPI and SSP Serial Transfers
- ([Section 16.2.5.3](#)) Operational Modes: Slave SPI and SSP Serial Transfers

16.2.5.1 Operational Modes: SSI / SPI Serial-Master

This mode enables serial communication with serial-slave peripheral devices. As a serial-master device, the SSI / SPI module initiates and controls all serial transfers under this mode. Refer to [Figure 16-1](#) for an illustration of this relationship.

The serial bit-rate clock, generated and controlled by the SSI / SPI interface, is driven out on the **SSIOCLK** pin. When the SSI / SPI interface is disabled (**SSSP_ssienr** bit **ssi_en** = 0), no serial transfers can occur and **SSIOCLK** is held in an inactive state.

Data transfers are initiated by the serial-master device, assuming there is at least one valid data entry in the transmit FIFO and a serial-slave device is selected. When actively transferring data, the Status register (**SSSP_sr**) busy flag (**busy**) is set. Software must wait until the busy flag is cleared before attempting a new serial transfer.

Note that the **busy** status (**SSSP_sr**) is not set when data is first written into the transmit FIFO. This bit is set only when the target slave has been selected and the transfer is underway. After writing data into the transmit FIFO, the shift-control logic does not initiate the serial transfer until a positive edge of the **SSIOCLK** signal is present. The delay in waiting for this positive edge depends on the baud rate of the serial transfer. Prior to polling the **busy** status, the software first should poll the Status register (**SSSP_sr**) status flag **txe** (wait for logic 1).

16.2.5.2 Operational Modes: Master SPI and SSP Serial Transfers

When the **SSSP_ctrlr0** register field transfer mode is set to Transmit and Receive or Transmit Only (**tmod** = 0x0 or **tmod** = 0x1, respectively), transfers are terminated by the shift-control logic when the transmit FIFO is empty ([Section 16.3.2.1](#)). For continuous data transfers, it is necessary to ensure that the transmit FIFO buffer does not become empty before all data has been transmitted. The transmit FIFO threshold level (**SSSP_txftlr**) can be used to prompt an early interrupt (SSI_TXE_INTR) to the processor indicating that the transmit FIFO buffer is nearly empty.

When the transfer mode is Receive Only (**tmod** = 0x2), a serial transfer is initiated by writing one “dummy” data word into the transmit FIFO when a serial slave is selected. If the serial transfer is continuous, this same data word is retransmitted until the serial transfer is completed. The transmit FIFO is popped only once at the beginning and may remain empty for the duration of the serial transfer. The end of the serial transfer is managed by using the Control register 1 (**SSSP_ctrlr1**), Number of Frames (**ndf**) field. The receive-logic will terminate the serial transfer when the number of frames received is equal to the **ndf** value + 1. For example, to receive 14 data frames from a serial-slave peripheral, the value 13 would be entered into the **ndf** field. This transfer mode increases the bandwidth of the AHB bus as the transmit FIFO never needs to be serviced during the transfer process. However, the receive FIFO buffer must be read each time the receive FIFO generates a FIFO-full interrupt request to prevent an overflow. The receive FIFO threshold level (**SSSP_rxftlr**) register can be used to provide early indication that the receive FIFO is nearly full.

A typical programming flow for completing a serial transfer from the SSI / SPI serial master is outlined as follows:

1. If the SSI / SPI interface is enabled, disable it by writing 0 to the SSI Enable register (**SSSP_ssienr**) bit **ssi_en**.
2. Set up the SSI / SPI control registers for the transfer. These registers can be set in any order.
 - Write Control register 0 (**SSSP_ctrlr0**). For SSI / SPI transfers, the serial clock polarity and serial clock phase parameters must be identical to those of the target slave device.
 - If the transfer mode is Receive Only, write Control register 1 (**SSSP_ctrlr1**) with the number of frames in the transfer minus 1, as described above.
 - Write the Baud Rate Select register (**SSSP_baudr**) to set the baud rate for the transfer.
 - Write the Transmit and Receive FIFO Threshold Level registers (**SSSP_txftlr** and **SSSP_rxftlr**, respectively) to set FIFO threshold levels.
 - Write the **SSSP_imr** register to set up interrupt masks.
 - The Slave Enable register (**SSSP_ser**) can be written to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to writing to the Data register (**SSSP_dr**), the transfer does not begin until a slave is enabled.
3. Enable the SSI / SPI interface by writing 1 to **ssi_en**.
4. Write the data to be transmitted into the transmit FIFO (write **SSSP_dr**). If no slaves have been enabled in the **SSSP_ser** register, they must be enabled at this stage to initiate the transfer.
5. In the Status Register (**SSSP_sr**) poll the **busy** status to wait for completion of the transfer. The **busy** status cannot be polled immediately. If a transmit FIFO-empty interrupt request is made, write the transmit FIFO (write **SSSP_dr**) register. If a receive FIFO-full interrupt request is made, read the receive FIFO (read **SSSP_dr**) register.
6. The transfer is terminated by the shift-control logic when the transmit FIFO is empty. If the transfer mode is Receive Only (**SSSP_ctrlr0** bit **tmod** = 0x2), the transfer is terminated by the shift-control logic when the specified number of frames have been received. When the transfer is completed, the **SSSP_sr busy** status is reset to 0.
7. If the transfer mode is not Transmit Only, read the receive FIFO until it is empty.
8. Disable the SSI / SPI interface by writing 0 to **ssi_en**.

16.2.5.3 Operational Modes: Slave SPI and SSP Serial Transfers

When an SSI / SPI slave transmits data, it is necessary to ensure that data exists in the transmit FIFO before a transfer is initiated by the serial-master device. If the master initiates a transfer from a slave when no data exists in the transmit FIFO, an error flag (TXE) is set in **SSSP_sr** and the previously transmitted data frame is resent. The transmit FIFO threshold level register (**SSSP_txftlr**) can be used to prompt an early interrupt (SSI_TXE_INTR) to the processor, indicating that the transmit FIFO buffer is nearly empty. When a DMA controller is used, an early request (DMA_TX_REQ) may be sent to the DMA controller to indicate that the transmit FIFO is nearly empty. The FIFO can then be refilled with data to continue the serial transfer. Note that when the **SSSP_ctrlr0** transfer mode is set to Receive Only (**tmod** = 0x2), the transmit FIFO need not contain valid data as the transmit shift register resets each time the slave device is selected.

The receive FIFO buffer should be read each time the receive FIFO generates a FIFO-full interrupt request to prevent an overflow. The receive FIFO threshold level register (**SSSP_rxftlr**) can be used to provide an early indication that the receive FIFO is nearly full. When a DMA controller is used, an early request (DMA_RX_REQ) may be sent to the DMA controller to indicate that the receive FIFO is nearly full.

A typical programming flow for completing a continuous serial transfer from a serial slave is described as follows:

1. If the SSI / SPI interface is enabled, disable it by writing 0 to the SSI Enable register (**SSSP_ssienr**) bit **ssi_en**.
2. Set up the control registers for the transfer. These registers can be set in any order.
 - Write **SSSP_ctrlr0** (set **scph** and **scpol** identical to the master device).
 - Write **SSSP_txftlr** and **SSSP_rxftlr** to set FIFO threshold levels.
 - Write the **SSSP_imr** register to set up the interrupt masks.
3. Enable the SSI / SPI interface by writing 1 to **ssi_en**.
4. If the transfer mode is Transmit and Receive (**tmod** = 0x0) or Transmit Only (**tmod** = 0x1), write the data to be transmitted into the transmit FIFO (write **SSSP_dr**). If the transfer mode is Receive Only (**tmod** = 0x2), there is no need to write data into the transmit FIFO, and the current value in the transmit shift register is retransmitted.
5. The SSI / SPI slave is now ready for the serial transfer. The transfer is initiated when the slave is selected by a serial-master device.
6. When the transfer is underway, poll the **busy** status (**SSSP_sr**) register to return the transfer status. If a transmit FIFO-empty interrupt request is made, write the transmit FIFO (write **SSSP_dr**) register. If a receive FIFO-full interrupt request is made, read the receive FIFO (read **SSSP_dr**) register.
7. The transfer is terminated when the serial master removes the select input to the SSI / SPI slave. When the transfer is completed, the **busy** status is reset to 0.
8. If the transfer mode is not Transmit Only, read the receive FIFO until it is empty.
9. Disable the SSI / SPI interface by writing 0 to **ssi_en**.

For additional DMA-related information, please refer to [Section 14.3.2.4 “AHBSP_ctl Register”](#).

16.3 SSI / SPI: Registers

The registers below are used for all SSI / SPI instances. Refer to [Section 16.1.2](#) for bus and base address information.

16.3.1 SSI / SPI Registers: Map

Register Offset	Register Name	Description
0x000	SSSP_ctrlr0	Control Register 0
0x004	SSSP_ctrlr1	Control Register 1
0x008	SSSP_ssienr	SSI Enable
0x00C		Reserved
0x010	SSSP_ser	Slave Enable
0x014	SSSP_baudr	Baud Rate Select
0x018	SSSP_txftlr	Transmit FIFO Threshold Level
0x01C	SSSP_rxftlr	Receive FIFO Threshold Level
0x020	SSSP_txflr	Transmit FIFO Level
0x024	SSSP_rxflr	Receive FIFO Level
0x028	SSSP_sr	Status Register
0x02C	SSSP_imr	Interrupt Mask
0x030	SSSP_isr	Interrupt Status
0x034	SSSP_risr	Raw Interrupt Status
0x038	SSSP_txoicr	Transmit FIFO Overflow Interrupt Clear
0x03C	SSSP_rxoicr	Receive FIFO Overflow Interrupt Clear
0x040	SSSP_rxuicr	Receive FIFO Underflow Interrupt Clear
0x044	SSSP_msticr	Multi-Master Interrupt Clear
0x048	SSSP_icr	Interrupt Clear
0x04C	SSSP_dmacr	DMA Control Register
0x050 - 0x054		Reserved
0x058	SSSP_idr	Identification
0x05C	SSSP_version	Version ID
0x060 - 0x25C	SSSP_dr	Data
0x260	SSSP_ssienpolr	SSI Slave Select Signal Polarity
0x264	SSSP_sclk_out_dly	SSIOLCK Delay Counter
0x268		Reserved
0x27C	target_frame_count	Target frame count for interrupt
0x280	frame_count	Current frame count
0x284	fcricr	Frame Counter Reaches Interrupt
0x288	tx_frame_count	Total Tx frame count

Table 16-3. SSI / SPI Register Map.

16.3.2 SSI / SPI Registers: Detail

16.3.2.1 SSSP_ctrlr0 Register

This register controls the serial data transfer. It is not possible to write to this register when the SSI / SPI interface is enabled. The SSI / SPI interface can be enabled and disabled by writing to the register **SSSP_ssienr**.

Bits	Name	Attr	Reset	Description
31:28				Reserved
27	spi_hold	RW	0	When this bit is set, SSI / SPI mode state machine will hold if TX FIFO frame does not reach the target frame count.
26	byte_wise_access_mode	RW	0	Allows byte access, half-word and word access to TXFIFO. 0: Disable 1: Enable dfs < 8, allows byte, half-word and word access. Each byte of HWDAT will be written to TX FIFO. dfs >= 8, only supports half-word and word access. Each half-word will be written to TX FIFO.
25:22	extra_rxd_margin	RW	0	Enables round chip delay compensation mode. When this bit is 1, RXD capture edge is delayed and tunable.
21	fc_en	RW	0	Frame counter enable. 0: Disable frame counter if fcr_int is not in use to save power 1: Enable frame counter
20				Reserved
19	rx_lfb_first	RW	0	RX LSB First mode. Used to indicate that the LSB bit of packets in RX FIFO was received first. 0: MSB first 1: LSB first
18	tx_lfb_first	RW	0	TX LSB First mode. Used to indicate that the LSB bit of packets in TX FIFO will be sent first. 0: MSB first 1: LSB first
17	residue_flush_mode	RW	0	Residue flush mode. Used to assert RX DMA request when the transfer completes and the frame count in RX FIFO is less than the RX FIFO threshold. This function prevents residue frames in RX FIFO and CPU intervention. 0: Disable 1: Enable
16:12				Reserved
11	srl	RW	0	Shift Register Loop Used for testing purposes only. When active internally, this bit connects the transmit shift register output to the receive shift register input. 0 - Normal Mode Operation 1 - Test Mode Operation

Bits	Name	Attr	Reset	Description
10	slv_oe	RW	0	<p>Slave Output Enable</p> <p>Relevant only when the AHB SSI is configured as a serial-slave device. When configured as a serial master, this bit field has no functionality. This bit enables or disables the setting of the SSI_OE_N output from the AHB SSI serial slave. When slv_oe = 1, the SSI_OE_N output can never be active. When the SSI_OE_N output controls the tri-state buffer on the TXD output from the slave, a high impedance state is always present on the slave TXD output when slv_oe = 1. This is useful when the master transmits in broadcast mode (master transmits data to all slave devices). Only one slave may respond with data on the master RXD line. This bit is enabled after reset and must be disabled by software (when broadcast mode is used), if the device is not to respond with data.</p> <p>0 - Slave TXD is enabled 1 - Slave TXD is disabled</p>
9:8	tmod	RW	0	<p>Transfer Mode</p> <p>Used to select the mode of transfer for serial communication. This field does not affect the transfer's duplicity. Only indicates whether the receive/transmit data is valid. In Transmit Only mode, data received from the external device is not valid and is not stored into the receive FIFO memory. It is overwritten on the next transfer. In Receive Only mode, transmitted data is not valid. After the first write to the transmit FIFO, the same word is retransmitted for the duration of the transfer.</p> <p>In Transmit and Receive mode, both transmit and receive data are valid. The transfer continues until the transmit FIFO is empty. Data received from the external device is stored into the receive FIFO memory, where it can be accessed by the host processor.</p> <p>0x0 - Transmit & Receive 0x1 - Transmit Only 0x2 - Receive Only 0x3 - Reserved</p>
7	scpol	RW	TBD	<p>Serial Clock Polarity</p> <p>Valid when the frame format (frf) is set to Motorola SPI. Used to select the polarity of the inactive serial clock. The serial clock is held inactive when the SSI / SPI master is not actively transferring data on the serial bus.</p> <p>0 - Inactive state of serial clock is low 1 - Inactive state of serial clock is high</p>
6	scph	RW	TBD	<p>Serial Clock Phase</p> <p>Valid when the frame format (frf) is set to Motorola SPI. The serial clock phase selects the relationship of the serial clock with the slave select signal.</p> <p>When scph = 0, data is captured on the first edge of the serial clock. When scph = 1, the serial clock starts toggling one cycle after the slave select line is activated, and data is captured on the second edge of the serial clock.</p> <p>0 - Serial clock toggles in middle of first data bit 1 - Serial clock toggles at start of first data bit</p>

Bits	Name	Attr	Reset	Description
5:4	frf	RW	0	Frame Format Used to select which serial protocol transfers the data. 0x0 - Motorola SPI Others - Reserved
3:0	dfs	RW	7	Data Frame Size (DFS) Used to select the data frame length. When the data frame size is programmed to be less than 16 bits, the receive data is automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded. Right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data. Refer to Table 16-5 for DFS Decode.

Table 16-4. SPI Control0 Register.

DFS Value	Description
0x0	Reserved - Undefined Operation
0x1	Reserved - Undefined Operation
0x2	Reserved - Undefined Operation
0x3	4-bit serial data transfer
0x4	5-bit serial data transfer
0x5	6-bit serial data transfer
0x6	7-bit serial data transfer
0x7	8-bit serial data transfer
0x8	9-bit serial data transfer
0x9	10-bit serial data transfer
0xA	11-bit serial data transfer
0xB	12-bit serial data transfer
0xC	13-bit serial data transfer
0xD	14-bit serial data transfer
0xE	15-bit serial data transfer
0xF	16-bit serial data transfer

Table 16-5. DFS Decode.

16.3.2.2 SSSP_ctrlr1 Register

This register exists only when the SSI / SPI is configured as a master device. When the SSI / SPI is configured as a serial slave, writing to this location has no effect; reading from this location returns zero. Control register 1 controls the end of serial transfers when in Receive Only mode. It is not possible to write to this register when the SSI / SPI interface is enabled. The SSI / SPI interface is enabled and disabled by writing to the **SSSP_ssienr** register.

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:0	ndf	RW	0	Number of Data Frames When SSSP_ctrlr0 bit tmod = 10, this register field sets the number of data frames to be received continuously by the SSI / SPI. The SSI / SPI continues to receive serial data until the number of data frames received is equal to this register value plus 1. This allows for receipt of up to 64 KB of data in a continuous transfer.

Table 16-6. SPI Control1 Register.

16.3.2.3 SSSP_ssienr Register

This register enables and disables the SSI / SPI interface.

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	ssi_en	RW	0	SSI Enable Used to enable and disable all SPI operations. When disabled, all serial transfers are halted immediately. It is not possible to program some of the SSI / SPI control registers when enabled. When disabled, the SSI_SLEEP output is set (after delay) to inform the system that it is safe to remove the SSI_CLK, thus saving power consumption in the system. FIFO buffers are cleared when enable goes from 1 to 0, but not when enable transitions from 0 to 1. In the software flow sequence, ssi_en must be programmed last.

Table 16-7. SSI / SPI Enable Register.

16.3.2.4 SSSP_ser Register

This register enables the individual slave select output lines from the SSI / SPI master. It is not possible to write to this register when SPI is **busy** (**SSSP_sr**) and when **ssi_en** = 1 (**SSSP_ssienr**).

Bits	Name	Attr	Reset	Description
31:8				Reserved

Bits	Name	Attr	Reset	Description
7:0	ser	RW	0	<p>Slave Select Enable Flag</p> <p>Each bit in this register corresponds to a slave select line from the SSI / SPI master. When a bit in this register is set (logic 1), the corresponding slave select line from the master is activated when a serial transfer begins. It should be noted that setting/clearing bits in this register have no effect on the corresponding slave select outputs until a transfer is initiated.</p> <p>Before beginning a transfer, enable the bit in this register that corresponds to the slave device with which the master wants to communicate. When not operating in broadcast mode, only one bit in this field should be set.</p> <p>0 - Not Selected 1 - Selected</p>

Table 16-8. SPI Slave Enable Register.

16.3.2.5 SSSP_baudr Register

This register is valid for the master device only. The register derives the frequency of the serial clock that regulates the data transfer. The 16-bit field in this register defines the **SSI0CLK** divider value. It is not possible to write to this register when the SSI / SPI interface is enabled. The SSI / SPI interface is enabled and disabled by writing to the **SSSP_ssienr** register.

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:0	sckdv	RW	0	<p>SSI Clock Divider (Section 16.2.1)</p> <p>The LSB for this field is always set to 0 and is unaffected by a write operation. This ensures an even value is held in this register. If the value is zero, the serial output clock (SSI0CLK) is disabled. The frequency of SSI0CLK is derived from CLK_IN with the following equation:</p> $\text{Freq}(\text{SSI0CLK}) = \text{Freq}(\text{CLK_IN}) / \text{sckdv}$ <p>where sckdv is any even value between 0 and 65534. Where CLK_IN represents GCLK_SSI or GCLK_SSI2 from the RCT module and is controlled by the SSI Clock generator in the RCT module. For Freq(CLK_IN) = 3.6864 MHz and sckdv = 2 the calculation is:</p> $\text{Freq}(\text{SSI0CLK}) = 3.6864/2 = 1.8432 \text{ MHz}$

Table 16-9. Baud Rate Select Register.

16.3.2.6 SSSP_txftlr Register

This register controls the threshold value for the transmit FIFO memory. It is not possible to write to this register when the SSI / SPI interface is enabled. The SSI / SPI interface is enabled and disabled by writing to the **SSSP_ssienr** register. TFT values differ between the Master and Slave controllers as described in the tables below.

Bits	Name	Attr	Reset	Description
31:6				Reserved
5:0	tft	RW	0	Transmit FIFO Threshold Used to control the level of entries (or below) at which the transmit FIFO controller triggers an interrupt. The FIFO depth is 32 or 64 entries. Attempts to set this value greater than or equal to the depth of the FIFO, do not write to this field and it retains its current value. When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty interrupt is triggered. If dfs < 8 and byte_wise_access_mode = 1, then FIFO depth is 64; otherwise 32.

Table 16-10. SPI Transmit FIFO Threshold Level Register.

TFT value	Description
0x00	SSI_TXE_INTR is asserted when 0 data entries are present in transmit FIFO
0x01	SSI_TXE_INTR is asserted when 1 data entries are present in transmit FIFO
0x02	SSI_TXE_INTR is asserted when 2 data entries are present in transmit FIFO
...	
0x3F	SSI_TXE_INTR is asserted when 64 data entries are present in transmit FIFO (If dfs < 8 and byte_wise_access_mode = 1, then FIFO depth is 64; otherwise 32.)

Table 16-11. SSSP_txftlr Register TFT Decode for the SSI / SPI Master Controller.

TFT value	Description
0x00	SSI_TXE_INTR is asserted when 0 data entries are present in transmit FIFO
0x01	SSI_TXE_INTR is asserted when 1 data entries are present in transmit FIFO
0x02	SSI_TXE_INTR is asserted when 2 data entries are present in transmit FIFO
...	
0x3F	SSI_TXE_INTR is asserted when 64 data entries are present in transmit FIFO (If dfs < 8 and byte_wise_access_mode = 1, then FIFO depth is 64; otherwise 32.)

Table 16-12. SSSP_txftlr Register TFT Decode for the SSI / SPI Dedicated Slave Controller.

See [Section 16.2.2 “SSI / SPI Transmit and Receive FIFO Buffers”](#) for information on SSI_TXE_INTR assertion.

16.3.2.7 SSSP_rxftlr Register

This register controls the threshold value for the receive FIFO memory. It is not possible to write to this register when the SSI / SPI interface is enabled. The SSI / SPI interface is enabled and disabled by writing to the **SSSP_ssienr** register. RFT values differ between the Master and Slave controllers as described in the tables below.

Bits	Name	Attr	Reset	Description
31:6				Reserved
5:0	rft	RW	0	<p>Receive FIFO Threshold Used to control the level of entries (or above) at which the receive FIFO controller triggers an interrupt. The FIFO depth is configurable in the range to 32 or 64. This register is sized to the number of address bits needed to access the FIFO. Attempts to set this value greater than the depth of the FIFO do not write to this field and it retains its current value.</p> <p>When the number of receive FIFO entries is greater than or equal to this value + 1, the receive FIFO full interrupt is triggered. If dfs < 8 and byte_wise_access_mode = 1, then FIFO depth is 64; otherwise 32.</p>

Table 16-13. Receive FIFO Threshold Level Register.

RFT value	Description
0x00	SSI_RXF_INTR is asserted when 1 data entries are present in receive FIFO
0x01	SSI_RXF_INTR is asserted when 2 data entries are present in receive FIFO
0x02	SSI_RXF_INTR is asserted when 3 data entries are present in receive FIFO
...	
0x3F	SSI_RXF_INTR is asserted when 64 data entries are present in receive FIFO

Table 16-14. SSSP_rxftlr Register RFT Decode for the SSI / SPI Master Controller.

RFT value	Description
0x00	SSI_RXF_INTR is asserted when 1 data entries are present in receive FIFO
0x01	SSI_RXF_INTR is asserted when 2 data entries are present in receive FIFO
0x02	SSI_RXF_INTR is asserted when 3 data entries are present in receive FIFO
...	
0x3F	SSI_RXF_INTR is asserted when 64 data entries are present in receive FIFO

Table 16-15. SSSP_rxftlr Register RFT Decode for the SSI / SPI Dedicated Slave Controller.

See [Section 16.2.2 “SSI / SPI Transmit and Receive FIFO Buffers”](#) for more information on SSI_RXF_INTR assertion.

16.3.2.8 SSSP_txflr Register

This register contains the number of valid data entries in the transmit FIFO memory.

Bits	Name	Attr	Reset	Description
31:6				Reserved
5:0	txflr	RW	0	Transmit FIFO Level

Table 16-16. Transmit FIFO Level Register.

16.3.2.9 SSSP_rxflr Register

This register contains the number of valid data entries in the transmit FIFO memory. This register can be read at any time.

Bits	Name	Attr	Reset	Description
31:6				Reserved
5:0	rxtflr	RW	0	Receive FIFO Level

Table 16-17. Receive FIFO Level Register.

16.3.2.10 SSSP_sr Register

This is a read-only register used to indicate the current transfer status, FIFO status, and transmission/reception errors that may have occurred. The status register may be read at any time. None of the bits in this register request an interrupt.

Bits	Name	Attr	Reset	Description
31:7				Reserved
6	dcol	R	0	<p>Data Collision Error Only relevant when the SSI / SPI is configured as a master device. This bit is set if the SSI / SPI master is actively transmitting when another master selects this device as a slave. This informs the processor that the last data transfer was halted before completion.</p> <p>This bit is cleared when read. 0 - No error 1 - Transmit data collision error</p>
5	txe	R	0	<p>Transmission Error Set if the transmit FIFO is empty when a transfer is started. This bit can only be set when the SSI / SPI is configured as a slave device. Data from the previous transmission is resent on the TXD line.</p> <p>This bit is cleared when read. 0 - No error 1 - Transmission error</p>

Bits	Name	Attr	Reset	Description
4	rff	R	0	<p>Receive FIFO Full</p> <p>When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared.</p> <p>0 - Receive FIFO is not full 1 - Receive FIFO is full</p>
3	rfne	R	0	<p>Receive FIFO</p> <p>Not Empty. Set when the receive FIFO contains one or more entries and is cleared when the receive FIFO is empty. This bit can be polled by software to completely empty the receive FIFO.</p> <p>0 - Receive FIFO is empty 1 - Receive FIFO is not empty</p>
2	tfe	R	1	<p>Transmit FIFO Empty</p> <p>When the transmit FIFO is completely empty, this bit is set. When the transmit FIFO contains one or more valid entries, this bit is cleared.</p> <p>This bit field does not request an interrupt.</p> <p>0 - Transmit FIFO is not empty 1 - Transmit FIFO is empty</p>
1	tfnf	R	1	<p>Transmit FIFO Not Full</p> <p>Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full.</p> <p>0 - Transmit FIFO is full 1 - Transmit FIFO is not full</p>
0	busy	R	0	<p>SSI Busy Flag</p> <p>When set, indicates that a serial transfer is in progress; when cleared indicates that the SSI / SPI is idle or disabled.</p> <p>0 - SPI is idle or disabled 1 - SPI is actively transferring data</p>

Table 16-18. SSI / SPI Status Register (SR).

16.3.2.11 SSSP_imr Register

This read/write register masks or enables all interrupts generated by the SSI / SPI interface. When the SSI / SPI is configured as a slave device, the **mstim** bit field is not present. This changes the reset value from 0x3F for serial-master configurations to 0x1F for serial-slave configurations.

Bits	Name	Attr	Reset	Description
31:9				Reserved
8	fcrim	RW	1	<p>Only valid in the master configuration.</p> <p>Frame counter reaches target value interrupt mask.</p> <p>0: Masked. 1: Not-masked</p>
7:6				Reserved
5	mstim	RW	1	<p>Only valid in the master configuration</p> <p>Multi-Master Contention Interrupt Mask</p> <p>0 - SSI_MST_INTR interrupt is masked 1 - SSI_MST_INTR interrupt is not masked</p>

Bits	Name	Attr	Reset	Description
4	rfim	RW	1	Receive FIFO Full Interrupt Mask 0 - SSI_RXF_INTR interrupt is masked 1 - SSI_RXF_INTR interrupt is not masked
3	rxiom	RW	1	Receive FIFO Overflow Interrupt Mask 0 - SSI_RXO_INTR interrupt is masked 1 - SSI_RXO_INTR interrupt is not masked
2	rxuim	RW	1	Receive FIFO Underflow Interrupt Mask 0 - SSI_RXU_INTR interrupt is masked 1 - SSI_RXU_INTR interrupt is not masked
1	txoim	RW	1	Transmit FIFO Overflow Interrupt Mask 0 - SSI_TXO_INTR interrupt is masked 1 - SSI_TXO_INTR interrupt is not masked
0	txeim	RW	1	Transmit FIFO Empty Interrupt Mask 0 - SSI_TXE_INTR interrupt is masked 1 - SSI_TXE_INTR interrupt is not masked

Table 16-19. SPI Interrupt Mask Register.

16.3.2.12 SSSP_isr Register

This register reports the status of the SSI / SPI interrupts after they have been masked.

Bits	Name	Attr	Reset	Description
31:9				Reserved
8	fcris			Only valid in the master configuration. Frame counter reaches target value interrupt status. 0: Not active 1: Active
7:6				Reserved
5	mstis	R	0	Only valid in the master configuration Multi-Master Contention Interrupt Status 0 - SSI_MST_INTR interrupt not active after masking 1 - SSI_MST_INTR interrupt is active after masking
4	rfxis	R	0	Receive FIFO Full Interrupt Status 0 - SSI_RXF_INTR interrupt is not active after masking 1 - SSI_RXF_INTR interrupt is full after masking
3	rxis	R	0	Receive FIFO Overflow Interrupt Status 0 - SSI_RXO_INTR interrupt is not active after masking 1 - SSI_RXO_INTR interrupt is active after masking
2	rxuis	R	0	Receive FIFO Underflow Interrupt Status 0 - SSI_RXU_INTR interrupt is not active after masking 1 - SSI_RXU_INTR interrupt is active after masking
1	txois	R	0	Transmit FIFO Overflow Interrupt Status 0 - SSI_TXO_INTR interrupt is not active after masking 1 - SSI_TXO_INTR interrupt is active after masking
0	txeis	R	0	Transmit FIFO Empty Interrupt Status 0 - SSI_TXE_INTR interrupt is not active after masking 1 - SSI_TXE_INTR interrupt is active after masking

Table 16-20. SPI Interrupt Status Register.

16.3.2.13 SSSP_risr Register

This read-only register reports the status of the SSI / SPI interrupts prior to masking.

Bits	Name	Attr	Reset	Description
31:9				Reserved
8	fcris			Only valid in the master configuration. Frame counter reaches target value raw interrupt status. 0: Not active prior to masking 1: Active prior to masking
7:6				Reserved
5	mstir	R	0	Only valid in the master configuration Multi-Master Contention Raw Interrupt Status 0 - SSI_MST_INTR interrupt is not active prior to masking 1 - SSI_MST_INTR interrupt is active prior to masking
4	rxfir	R	0	Receive FIFO Full Raw Interrupt Status 0 - SSI_RXF_INTR interrupt is not active prior to masking 1 - SSI_RXF_INTR interrupt is active prior to masking
3	rxoir	R	0	Receive FIFO Overflow Raw Interrupt Status 0 - SSI_RXO_INTR interrupt is not active prior to masking 1 - SSI_RXO_INTR interrupt is active prior to masking
2	rxuir	R	0	Receive FIFO Underflow Raw Interrupt Status 0 - SSI_RXU_INTR interrupt is not active prior to masking 1 - SSI_RXU_INTR interrupt is active prior to masking
1	txoir	R	0	Transmit FIFO Overflow Raw Interrupt Status 0 - SSI_TXO_INTR interrupt is not active prior to masking 1 - SSI_TXO_INTR interrupt is active prior to masking
0	txeir	R	0	Transmit FIFO Empty Raw Interrupt Status 0 - SSI_TXE_INTR interrupt is not active prior to masking 1 - SSI_TXE_INTR interrupt is active prior to masking

Table 16-21. Raw Interrupt Status Register.

16.3.2.14 SSSP_txoicr Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	txoicr	R	0	Clear Transmit FIFO Overflow Interrupt This register reflects the status of the interrupt. A read from this register clears the SSI_TXO_INTR interrupt; writing has no effect.

Table 16-22. Transmit FIFO Overflow Interrupt Clear Register.

16.3.2.15 SSSP_rxoicr Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	rxoicr	R	0	Clear Receive FIFO Overflow Interrupt This register reflects the status of the interrupt. A read from this register clears the SSI_RXO_INTR interrupt; writing has no effect.

Table 16-23. Receive FIFO Overflow Interrupt Clear Register.

16.3.2.16 SSSP_rxuicr Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	rxuicr	R	0	Clear Receive FIFO Underflow Interrupt This register reflects the status of the interrupt. A read from this register clears the SSI_RXU_INTR interrupt; writing has no effect.

Table 16-24. Receive FIFO Underflow Interrupt Clear Register.

16.3.2.17 SSSP_msticr Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	msticr	R	0	Clear Multi-Master Contention Interrupt This register reflects the status of the interrupt. A read from this register clears the SSI_MST_INTR interrupt; writing has no effect.

Table 16-25. Multi-Master Interrupt Clear Register.

16.3.2.18 SSSP_icr Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	icr	R	0	Clear Interrupts This register is set if any of the interrupts above are active. A read clears the SSI_TXO_INTR, SSI_RXU_INTR, SSI_RXO_INTR, and the SSI_MST_INTR interrupts. Writing to this register has no effect.

Table 16-26. SSI / SPI Interrupt Clear Register.

16.3.2.19 SSSP_dmacr Register

Bits	Name	Attr	Reset	Description
31:2				Reserved
1	tdmae	RW	0	<p>TX DMA request enable This field is the SSI TX DMA request enable control bit. When this field is set and the TX FIFO level is under the target value, SSI will issue a TX DMA request to the DMA controller. In non-DMA mode, this bit must be set to 1 before data can be pushed to the TX FIFO.</p> <p>0: Disable 1: Enable</p>
0	rdmae	RW	0	<p>RX DMA request enable This field is the SSI RX DMA request enable control bit. When this field is set and the RX FIFO level reaches the target value, SSI will issue an RX DMA request to the DMA controller.</p> <p>0: Disable 1: Enable</p>

Table 16-27. SSI / SPI Interrupt Clear Register.

16.3.2.20 SSSP_idr Register

This read-only register is available for use to store a peripheral identification code.

Bits	Name	Attr	Reset	Description
31:0	id	R	0x414D4241	The peripheral identification code

Table 16-28. SSI / SPI Identification Code Register.

16.3.2.21 SSSP_version Register

This read-only register is available for use to store the SSI design version.

Bits	Name	Attr	Reset	Description
31:0	version_id	R	0x312E3030	The SSI design version.

Table 16-29. SSI / SPI Version Register.

16.3.2.22 SSSP_dr Register

This SSI / SPI data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data is moved into the transmit FIFO buffer. This register can only be written when **SSSP_ssienr** bit **ssi_en** = 1. The FIFOs are reset when **ssi_en** = 0.

Bits	Name	Attr	Reset	Description
31:16				Reserved

Bits	Name	Attr	Reset	Description
15:0	dr	RW	0	Data Register When writing to this register, it is necessary to right-justify the data. Read data is automatically right-justified. Read - Receive FIFO buffer Write - Transmit FIFO buffer

Table 16-30. SSI / SPI Data Register.

16.3.2.23 SSSP_ssienpolr Register

This register is used for selecting slave enable signal polarity.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	ssienpol	RW	0	Slave Select Signal Polarity Each bit controls each SER bit. Real output depends on SSI mode and this register. 0 - Use original polarity 1 - Invert original polarity For example, if bit 0 is for a SPI interface device, then: SSIENTPOL[0] = 0 - Active low SSIENTPOL[0] = 1 - Active high

Table 16-31. SSI / SPI Slave Enable Signal Polarity Register.

16.3.2.24 SSSP_sclk_out_dly Register

This register is used for delaying **SSIOTCLK** after slave select asserting.

Bits	Name	Attr	Reset	Description
31:17				Reserved
16:0	sclk_out_dly	RW	0	Delay SSIOTCLK cycles after slave select asserted

Table 16-32. SSI / SPI **SSIOTCLK** Delay Counter Register.

16.3.2.25 SSSP_target_frame_count Register

This register is used for defining the target transferred frame count to assert an interrupt.

Bits	Name	Attr	Reset	Description
31:17				Reserved
16:0	target_frame_count	RW	0	Target frame count

Table 16-33. SSI / SPI Target Frame Count Register.

16.3.2.26 SSSP_frame_count Register

This register is used for defining the target transferred frame count to assert an interrupt.

Bits	Name	Attr	Reset	Description
31:17				Reserved
16:0	frame_count	R	0	Current frame count

Table 16-34. SSI / SPI Current Frame Count Register.

16.3.2.27 SSSP_fcricr Register

This register is used for defining the target transferred frame count to assert an interrupt.

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	fcricr	R	0	Frame Counter Reaches Interrupt This register reflects the status of the interrupt. A read from this register clears the fcris interrupt; writing has no effect.

Table 16-35. SSI / SPI Frame Count Interrupt Clear Register.

16.3.2.28 SSSP_tx_frame_count Register

This register is used to define the total number of transmit (Tx) frames to send.

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:0	tx_frame_count	RW	0	Total Tx frame count

Table 16-36. SSI / SPI Total TX Frame Count Register.

16.4 SSI / SPI: NOR-SPI Programming and Register Information

16.4.1 NOR-SPI: Controller Architecture

The following figure provides an illustration of the A12 NOR-SPI controller architecture.

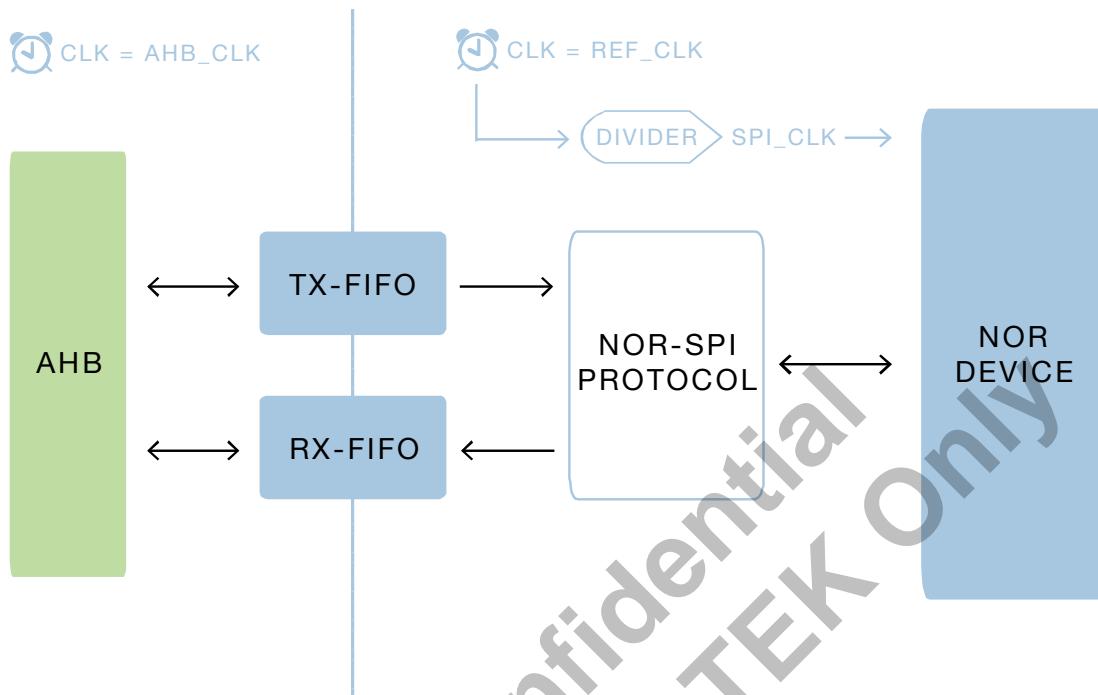


Figure 16-9. NOR-SPI Controller Architecture.

Notes:

1. The TX FIFO and RX FIFO modules are composed of asynchronous and synchronous FIFO.
2. TX FIFO size is 265 bytes. RX FIFO size is 264 bytes.
3. Based on the configuration register, the NOR-SPI Protocol unit generates a NOR protocol waveform to access the NOR device.
4. The clock divider is a register-based digital divider.
5. The NOR-SPI controller is located at AHB base address 0xE003.1000.

16.4.2 NOR-SPI: Performance Specification Notes

- Ahb_clk: Approximately 120 MHz
- Reference clock has two sources
 - 150 MHz
 - Ahb_2x clock

16.4.3 NOR-SPI: Command Format

The figures below illustrate the basic structure and timing of a NOR-SPI command. Note that the timing parameters used in this example are as follows:

- cmdlength = 1
- addrlength = 3
- dummylength = 2
- datalength = 1
- numcmdlane = 1
- numaddrlane = 1
- numdatalane = 1
- dataread = 1 (for read command)

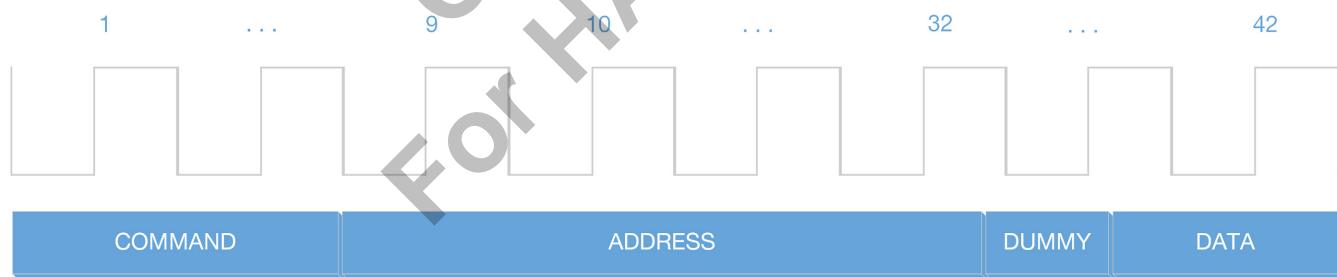


Figure 16-10. NOR-SPI Programming Command Format.

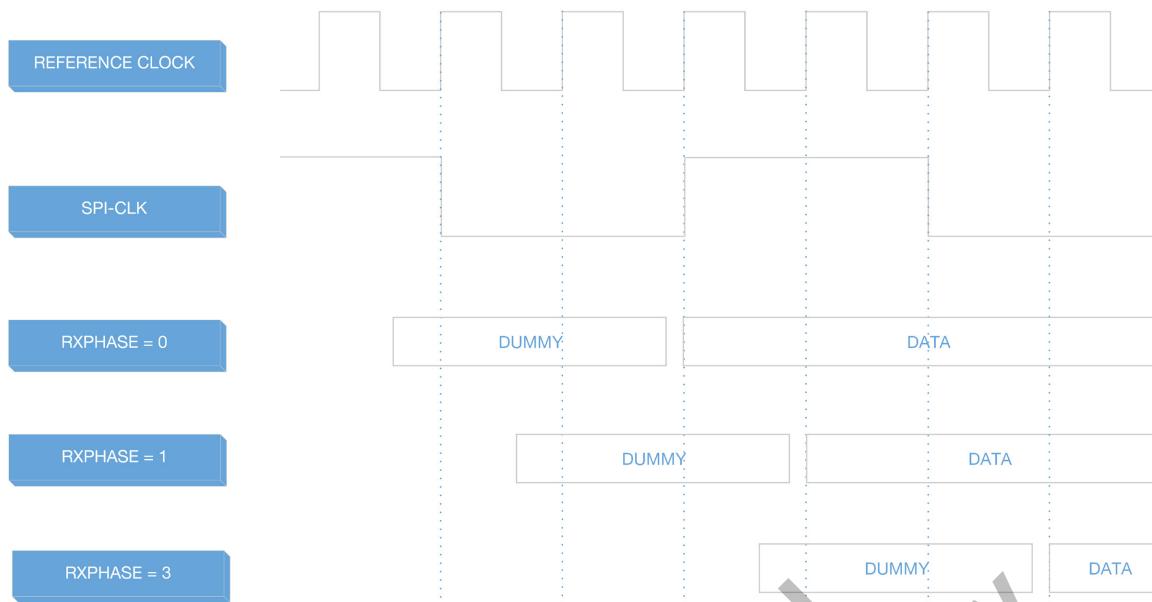


Figure 16-11. NOR-SPI: Example of SPI RX Sampling Phase.

The following table shows the possible lane combinations for a commercially-available NOR-SPI device. The A12 NOR-SPI controller provides a degree of lane configuration flexibility; please contact an Ambarella representative for details.

Command Lane	Address Lane	Data Lane
1	1	0/1/2/4
1	2	2
2	2	2
1	4	4
4	4	4

Table 16-37. NOR-SPI Lane Number Combinations.

16.4.4 NOR-SPI: Registers

When accessing NOR-SPI controller registers, the following general guidelines apply:

1. Access to the NOR-SPI device can be managed in one of two ways.
 - The device can be accessed via a DMA agent. In this case, the implementation of flow control measures can be useful for preventing TX/RX FIFO overflow/underflow when the DMA agent is otherwise occupied and fails to read or write data in a timely manner. This could be accomplished through the use of a commercially available device that supports flow control; however, the A12 NOR-SPI controller can achieve a similar result by gating the SPI clock to hold the SPI NOR transaction. For more information on DMA access to the NOR-SPI controller, please refer to [Section 16.4.5](#).
 - Alternatively, the device can be directly accessed by the Cortex-A9 processor.
2. Program the configuration field according to the NOR device being used, then write the Start Transmission/Reception register **strttrx** to 1.
3. The data-write port range is between 0x100 and 0x1FF. The data-read port range is between 0x200 and 0x2FF. Once the programmed (received or transmitted) data length has been achieved, the **datalengthreachintr** interrupt will provide notification that the transaction is complete. The **rxalmostfullintr** and **txunderflowintr** interrupts can be used to prevent a FIFO overflow or underflow.

Address Offset	Bits	Name	Attr	Reset	Description
0x00	31:30	cmdlength	RW	0x1	Command Length 0x0: 0 byte 0x1: 1 byte 0x2: 2 bytes 0x3: 3 bytes Disable the output on the interface in case of 0 byte.
	29:27	addrlength	RW	0x3	Address Length 0x0: 0 byte 0x1: 1 byte 0x2: 2 bytes ... 0x7: 7 bytes Disable the output on the interface in case of 0 byte.
	26:22	dummylength	RW	0x0	Dummy Cycle Length 0x0: 0 cycle 0x1: 1 cycle 0x2: 2 cycles ... 0x1F: 31 cycles Disable the output on the interface in case of 0 byte.

Address Offset	Bits	Name	Attr	Reset	Description
	21:0	datalength	RW	0x18	Data Access Length 0x0: 0 bytes 0x1: 1 bytes 0x2: 2 bytes ... 0x3FFFFF: 3FFFFF bytes Disable the output on the interface in case of 0 byte.
0x04	31	cmddtr	RW	0x0	Command Double Transfer Rate (DTR) DTR is only supported when reg_the clock divider >2 cmddtr 0x0: DTR Not Supported 0x1: DTR Supported
	30	address dtr	RW	0x0	Address Double Transfer Rate 0x0: DTR Not Supported 0x1: DTR Supported
	29	dummy dtr	RW	0x0	Dummy Double Transfer Rate 0x0: DTR Not Supported 0x1: DTR Supported
	28	data dtr	RW	0x0	Data Double Transfer Rate 0x0: DTR Not Supported 0x1: DTR Supported
	27:25				Reserved
	24	lsbfirst	RW	0x0	0x0: MSB first 0x1: LSB first
	15:14	numcmdlane	RW	0x0	0x0: 1 lane 0x1: 2 lane 0x2: 4 lane 0x3: 8 lane
	13:12	numaddrlane	RW	0x0	0x0: 1 lane 0x1: 2 lane 0x2: 4 lane 0x3: Reserved
	11:10	numdatalane	RW	0x0	0x0: 1 lane 0x1: 2 lane 0x2: 4 lane 0x3: 8 lane
	9	rxlane	RW	0x1	0x0: Lane 0 for one data lane Lane 0 and 1 for two data lanes Lane 0, 1, 2 and 3 for four data lanes 0x1: Lane 1 for one data lane Lane 2 and 3 for two data lanes Lane 4, 5, 6 and 7 for four data lanes Always set to 0 when transmitting only. Always set to 1 when transmitting and receiving.

Address Offset	Bits	Name	Attr	Reset	Description
	1	datawriteen	RW	0x0	Data Part Write Mode 0x0: Write Disable 0x1: Write Enable Set to 1 unless receiving only.
	0	datareaden	RW	0x1	Data Part Read Mode 0x0: Read Disable 0x1: Read Enable Set to 1 unless transmitting only.
0x08	31	flowcontrol	RW	0x1	Flow Control Enable 0x0: No Flow Control 0x1: Flow Control Enable
	30:28	hold	RW	0x0	Used for flow control purposes. Takes effect when flowcontrol =1 0x00: Using gate clock for flow control 0x01: Using dq_o[1] as a hold to perform flow control 0x02: Using dq_o[2] as a hold to perform flow control 0x06: Using dq_o[6] as a hold to perform flow control 0x07: Using dq_o[7] as a hold to perform flow control
	27	spiclkpolarity	RW	0x0	Clock will remain 1 or 0 in standby mode. scpol and scph only support (0, 0) or (1, 1)
	26	holdswitchphase	RW	0x1	0 - Hold switch one reference clock cycle before negative edge 1 - Hold switch on negative edge
	25:18	chipselect#	RW	0xFE	CEN for multiple device SELPIN = CHIPSELECT {8{SEL}}; 0 for select and vice versa
	17:10	clkdivider	RW	0x1E	Divider for reference clock 0x0: Reserved 0x1: Reserved 0x2: Reference Clock/2 ... 0xFF: Reference Clock/255
	9:5				Reserved
	4:0	rxsampledelay	RW	0	Adjust Reception Sampling Data Phase 0x0: 0 reference clock cycle shift ... 0x1F: 1F reference clock cycle shift 1 SPI cycle is required to adjust RX sampling phase
0x0C	31:24				Reserved
	23:16	cmd2	RW	0	Command 2 for SPI device
	15:8	cmd1	RW	0	Command 1 for SPI device
	7:0	cmd0	RW	3	Command 0 for SPI device
0x10	31:24				Reserved
	23:16	addr6	RW	0	ADDRESS 6 Field for SPI command

Address Offset	Bits	Name	Attr	Reset	Description
	15:8	addr5	RW	0	ADDRESS 5 Field for SPI command
	7:0	addr4	RW	0	ADDRESS 4 Field for SPI command
0x14	31:24	addr3	RW	0	ADDRESS 3 Field for SPI command
	23:16	addr2	RW	0	ADDRESS 2 Field for SPI command
	15:8	addr1	RW	0	ADDRESS 1 Field for SPI command
	7:0	addr0	RW	0	ADDRESS 0 Field for SPI command
0x18	1	txdmaenable	RW	0	Transmit DMA Enable
	0	rxdmaenable	RW	0	Receive DMA Enable
0x01C	8:0	txfifoctlv	RW	0	Transmit FIFO Threshold Level
0x20	8:0	rxfifoctlv	RW	0	Receive FIFO Threshold Level
0x24	8:0	txfifolv	RO	0	Transmit FIFO Level
0x028	8:0	rxfifolv	RO	0	Receive FIFO Level
0x02C	31:5				Reserved
	4	rxfifofull	RO		RX FIFO full
	3	rxfifoisempty	RO		
	2	txfifoempty	RO		TX FIFO empty
	1	txfifoisnotfull	RO		
	0				Reserved
0x030	31:7				Reserved
	6	txunderflowmsk			Transmit Underflow Interrupt Mask
	5	datalengthreachmsk			Transaction Complete Interrupt Mask
	4	rxalmostfullmsk			RX FIFO almost full
	3	rxoverflowmsk			Receive Overflow Interrupt Mask
	2	rxunderflowmsk			Receive Underflow Interrupt Mask
	1	txoverflowmsk			Transmit Overflow Interrupt Mask
	0	txalmostemptymsk			TX FIFO almost empty
0x034	31:7				Reserved
	6	txunderflowintr			Transmit Underflow Interrupt
	5	datalengthreachintr			Transaction Complete Interrupt
	4	rxalmostfullintr			RX FIFO almost full
	3	rxoverflowintr			Receive Overflow Interrupt
	2	rxunderflowintr			Receive Underflow Interrupt
	1	txoverflowintr			Transmit Overflow Interrupt
	0	txalmostemptynintr			TX almost empty
0x38	31:7				Reserved
	6	txunderflowirntr			Transmit Underflow Raw Interrupt Status
	5	datalengthreachirntr			Transaction Complete Raw Interrupt Status
	4	rxalmostfullirntr			RX FIFO almost full
	3	rxoverflowirntr			Receive Overflow Raw Interrupt Status
	2	rxunderflowirntr			Receive Underflow Raw Interrupt Status
	1	txoverflowirntr			Transmit Overflow Raw Interrupt Status
	0	txalmostemptyrntr			TX FIFO almost empty
0x3C	31:7				Reserved
	6	txunderflowinrc	WC		Transmit Underflow Write Clear Interrupt Status. Write 1 to clear.

Address Offset	Bits	Name	Attr	Reset	Description
	5	datalengthreachintrc	WC		Transaction Complete Write Clear Interrupt Status. Write 1 to clear.
	4	rxalmostfullintrc	WC		RX FIFO almost full
	3	rxbusyintrc	WC		Receive Busy Write Clear Interrupt Status. Write 1 to clear.
	2	rxunderflowintrc	WC		Receive Underflow Write Clear Interrupt Status. Write 1 to clear.
	1	txoverflowintrc	WC		Transmit Overflow Write Clear Interrupt Status. Write 1 to clear.
	0	txalmostemptyintrc	WC		TX FIFO almost empty
0x40	0	txfiforeset	AC	0	TX FIFO Software Reset
0x44	0	rxfiforeset	AC	0	RX FIFO Software Reset
0x50	0	strttx	AC	0	Start Transmitting or Receiving
0x100 - 0x1FF	31:0	txdata	WO		Data to be written to the NOR device
0x200 - 0x2FF	31:0	rxdata	RO		Data to be read from the NOR device

Table 16-38. SSI / SPI: NOR-SPI Registers.

16.4.5 NOR-SPI: Using DMA

Note that the use of DMA access with the A12 NOR SPI controller is subject to the requirements and constraints listed below. For additional information regarding the use of DMA access with the SPI-NOR controller, please refer to [Section 14.3.2.4 “AHBSP_ctl Register”](#).

- **tx_dma_req** will be asserted when the TX FIFO level (**txfifo_lv**) is less than or equal to **txfifo_thlv**.
- **rx_dma_req** will be asserted when the RX FIFO level (**rxfifo_lv**) exceeds **rxfifo_thlv**.
- RX DMA constraints:
 - **dma_blk_size**: **dma_ch[n].control[26:24]**
dma_byte_count: **dma_ch[n].control[21:0]**
 - **rxfifo_thlv == dma_blk_size - 1**
 - **(datalength % dma_blk_size) == 0**
 - **dma_byte_count == datalength**
- For example: RX DMA 128 bytes of NOR SPI
 - **dma_blk_size = 32 bytes (dma_ch[n].control[26:24] = 2)**
 - **dma_byte_count = 128 bytes (datalength = 128)**
 - **rxfifo_thlv = 32 - 1 = 31**

1. Wait for the RX FIFO to accumulate 32 bytes, a value larger than the FIFO threshold level → DMA request → DMA AHB read → (128 - 32 bytes (`dma_blk_size == 32 bytes`) left to be read via DMA).
2. Wait for the RX FIFO to accumulate 32 bytes, a value larger than the FIFO threshold level → DMA request → DMA AHB read → (96 - 32 bytes (`dma_blk_size == 32 bytes`) left to be read via DMA).
3. Wait for the RX FIFO to accumulate 32 bytes, a value larger than the FIFO threshold level → DMA request → DMA AHB read → (64 - 32 bytes (`dma_blk_size == 32 bytes`) left to be read via DMA).
4. Wait for the RX FIFO to accumulate 32 bytes, a value larger than the FIFO threshold level → DMA request → DMA AHB read → (final DMA transaction).
5. Note that the steps above must be used to configure the RX DMA engine; otherwise, the DMA transaction will hang due to the presence of residual data in the RX FIFO (i.e., the final DMA request will not be triggered).

For Confidential
For HAOTEX Only

17. UART

This chapter discusses the A12 Universal Asynchronous Receiver/Transmitter (UART) interface. The chapter is organized as follows:

- [\(Section 17.1\) UART: Overview](#)
- [\(Section 17.2\) UART: Clocking](#)
- [\(Section 17.3\) UART: Registers](#)

17.1 UART: Overview

The A12 chip offers two industry-standard 16550 UART programming interfaces (UART0 and UART1). Features of the interfaces include:

- 32-bit data width and internal FIFO with a depth of 16
- Support for auto- and manual- flow control on one UART interface (UART1)
- Loopback mode control support

UART1 supports flow control and DMA access by the DMA Engine via the AHB bus. UART0 does not support flow control, and certain register fields are not applicable to this interface.

For additional DMA-related information, please refer to [Section 14.3.2.4 “AHBSP_ctl Register”](#).

17.2 UART: Clocking

A12 system software generally sets and configures the source of the baud rate clock and the UART clock GCLK_UART. The register programming defined in this chapter is used for adjustment and refinement.

17.3 UART: Registers

Each UART instance is mapped to a unique address range on both the AHB and APB buses. UART1 is located at base address 0xE003.2000 on the AHB bus, while UART0 is located on the APB bus at base address 0xE800.5000.

17.3.1 UART Registers: Map

UART registers are multi-functional. Address offsets 0x00 and 0x04 use the value of a particular bit as a switch that triggers read and write operations to different registers at the same address. Address offset 0x08 also provides read or write access to different registers. The map below outlines these dependencies. Note that DMA-related register fields are not applicable to the UART 0 interface.

0x00	UART_rbr	Receive Buffer	Read access when dlab = 0
	UART_thr	Transmit Holding	Write access when dlab = 0
	UART_dll	Divisor Latch (Low)	Read and Write access when dlab = 1
0x04	UART_iер	Interrupt Enable	Read and Write access when dlab = 0
	UART_dlh	Divisor Latch (High)	Read and Write access when dlab = 1
0x08	UART_iir	Interrupt Identity	Read access only
	UART_fcr	FIFO Control	Write access only
0x0C	UART_lcr	Line Control	
0x10	UART_mcr	Modem Control	
0x14	UART_lsr	Line Status	
0x18	UART_msr	Modem Status	
0x1C	UART_scr	Scratchpad	
0x28	UART_dmae	DMA Enable in UART1	
0x40 - 0x5F	UART_thr (Remapped)	Mapping to Transmit Holding - This allows the DMA Engine to access UART1 with AHB incrementing burst transaction.	Write access when dlab = 0 (Any write access to this section is equivalent to writing UART_thr)
	UART_rbr (Remapped)	Mapping to Receive Buffer - This allows the DMA Engine to access UART1 with AHB incrementing burst transaction.	Read access when dlab = 0 (Any read access to this section is equivalent to reading UART_rbr)
0x7C	UART_usr	UART Status	
0x80	UART_tfl	Transmit FIFO Level	
0x84	UART_rfl	Receive FIFO Level	
0x88	UART_srr	Software Reset	
0x8C - 0xA0		Reserved	
0xA4	UART_htx	Halt Tx	
0xA8 - 0xF0		Reserved	
0xF4	UART_cpr	Component Parameter	
0xF8	UART_ucv	UART Component Version	
0xFC	UART_ctr	Component Type	

Table 17-1. UART Interface Registers.

17.3.2 UART Registers: Detail

17.3.2.1 UART_rbr Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	receive_buffer	R	0	<p>Receive Buffer</p> <p>The Receive Buffer register (UART_rbr) contains the data byte received on the serial input pin UARTRX. The data in UART_rbr is valid only if the Data Ready bit (dr) in the Line status register (UART_Isr) is set.</p> <p>In non-FIFO mode (FIFO Depth = NONE) or when the FIFOs are programmed OFF, the data in UART_rbr must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error.</p> <p>In FIFO mode (FIFO Depth != NONE) and FIFOs programmed ON, this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost and an overrun error will occur.</p>

Table 17-2. *UART Receive Buffer Register (RBR)*.

17.3.2.2 UART_thr Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	transmit_holding	W	0	<p>Transmit Holding</p> <p>The Transmit Holding register (UART_thr) contains data to be transmitted on the serial output port (UARTOTX pin). Data can be written to UART_thr any time the Transmit Holding Register Empty bit (thre) of the Line Status register (UART_Isr) is set. If FIFOs are not enabled and thre is set, writing a single character to UART_thr clears thre. Any additional writes to UART_thr before thre is set again causes the UART_thr data to be overwritten. If FIFOs are enabled and thre is set, x number of characters of data may be written to UART_thr before the FIFO is full. The number x (default = 16) is determined by the value of FIFO Depth that software sets during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p>

Table 17-3. *UART Transmit Holding Register (THR)*

17.3.2.3 **UART_dlh/ UART_dll Registers**

The Divisor Latch High register (**UART_dlh**), in conjunction with Divisor Latch Low register (**UART_dll**), forms a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. It is accessed by first setting Line Control register (**UART_lcr**) bit 7 (**dlab**). The output baud rate is equal to the serial clock frequency divided by sixteen times the value of the baud rate divisor, as follows:

$$\text{baud rate} = \text{GCLK_UART} / (16 * \text{divisor})$$

Note that when the Divisor Latch registers (**UART_dll** and **UART_dlh**) are set to zero, the baud clock is disabled and no serial communications occur.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	divisor_latch_high	RW	0	Divisor Latch (High) Upper 8 bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.

Table 17-4. *UART Divisor Latch High Register (DLH)*.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	divisor_latch_low	RW	0	Divisor Latch (Low) Lower 8 bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.

Table 17-5. *UART Divisor Latch Low Register (DLL)*.

17.3.2.4 **UART_iер Register**

The Interrupt Enable register (**UART_iер**) is a read/write register that contains bits to enable the generation of interrupts. See [Section 17.3.2.5](#) for details on the interrupt operation.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7	ptime	RW	0	Programmable THRE Interrupt Mode Enable. Always readable. This is used to enable/disable the generation of THRE Interrupt. 0 - Disabled 1 - Enabled
6				Reserved
5	etoi	RW	0	Enable/disable Time Out Interrupt for a time out event 0 - Disabled 1 - Enabled
4	ebdi	RW	0	Enable/disable Busy Detect Indication for writes to UART_lcr during a busy state 0 - Disabled 1 - Enabled

Bits	Name	Attr	Reset	Description
3	edssi	RW	NR	Enable Modem Status Interrupt This enables/disables generation of a Modem Status Interrupt. This is the fourth-highest priority interrupt. 0 - Disabled 1 - Enabled
2	elsi	RW	NR	Enable Receiver Line Status Interrupt This enables/disables generation of a Receiver Line Status Interrupt. This is the highest priority interrupt. 0 - Disabled 1 - Enabled
1	etbei	RW	NR	Enable Transmit Holding Register Empty Interrupt This enables/disables generation of a Transmitter Holding Register Empty Interrupt. This is the third-highest priority interrupt. 0 - Disabled 1 - Enabled
0	erbf1	RW	NR	Enable Received Data Available Interrupt This enables/disables generation of a Received Data Available Interrupt. This is the second-highest priority interrupt. 0 - Disabled 1 - Enabled

Table 17-6. UART Interrupt Enable Register (IER).

17.3.2.5 UART_iir Register

The Interrupt Identity register (**UART_iir**) is a read-only register that identifies the source of an interrupt. The upper two bits of the register are the FIFO enable bits. These bits will be 0x0 if the FIFOs are disabled, and 0x3 if they are enabled. The lower four bits identify the highest-priority pending interrupt as shown in [Table 17-8](#) below.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:6	fifos_enabled	R	0	FIFOs Enabled 0x0 - FIFOs disabled 0x3 - FIFOs enabled
5:4				Reserved
3:0	interrupt_id	R	1	Interrupt ID This indicates the highest-priority pending interrupt which can be one of the following types: 0x0 - Modem status changed 0x1 - No interrupt pending 0x2 - UART_thr empty 0x4 - Received data available 0x6 - Receiver line status 0x7 - Busy detect 0x12 - Character time out Bit 3 indicates an interrupt can occur only when the FIFOs are enabled and used to distinguish a Character Timeout condition interrupt.

Table 17-7. UART Interrupt ID Register (IIR).

FIFO Mode Only		Interrupt Identification Register		Interrupt Set and Reset Functions				
Bit 3	Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control	
0	0	0	1		None	None	None	
0	1	1	0	Highest	Receiver line status	Overrun/parity/framing errors or interrupt	Reading the Line Status register	
0	1	0	0	Second	Received data available	Receiver data available (non-FIFO mode or FIFOs disabled) or Rx FIFO trigger level reached (FIFO mode and FIFOs enabled)	Reading the Receiver Buffer register	
1	1	0	0	Second	Character time out indication	No characters in or out of the Rx FIFO during the last four character times and there is at least one character in it during this time	Reading the Receiver Buffer register	
0	0	1	0	Third	Transmitter Holding Register empty	Transmitter Holding Register empty (Prog. THRE Mode disabled) or Tx FIFO at or below threshold (Prog. THRE Mode enabled)	Reading the IIR register (if source of interrupt) or Writing to THR (FIFOs or THRE Mode not selected or disabled) or Tx FIFO above threshold (FIFOs and THRE Mode selected and enabled).	
0	0	0	0	Fourth	Modem status	Clear to send or data set ready or ring indicator or data carrier detect. Note that if auto flow control mode is enabled, a change in cts (that is, setting dcts) does not cause an interrupt.	Reading the Modem Status register	
0	1	1	1	Fifth	Busy detect indication	Master has tried to write to the Line Control register while the UART is busy (UART_usr[0] is 1).	Reading the UART status register	

Table 17-8. *UART Interrupt Control Functions.*

17.3.2.6 UART_fcr Register

The FIFO Control register (UART_fcr) shares the address with UART_iir but is write-only. Note that bit 3 in this register is used with the UART1 interface only.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:6	rcvr_trigger	W	0	<p>Rx Trigger</p> <p>This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated. In auto flow control mode it is used to determine when the UART_AHB_RTS_N signal is deasserted. It also determines when the external DMA_RX_REQ_N signal is asserted in certain modes of operation. The following trigger levels are supported:</p> <ul style="list-style-type: none"> 0x0 - One character in the FIFO 0x1 - FIFO 1/4 full 0x2 - FIFO 1/2 full 0x3 - FIFO 2 less than full
5:4	tx_empty_trigger	W	0	<p>Tx Empty Trigger</p> <p>This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active. It also determines when the DMA_TX_REQ_N signal is asserted when in certain modes of operation. The following trigger levels are supported:</p> <ul style="list-style-type: none"> 0x0 - FIFO empty 0x1 - Two characters in the FIFO 0x2 - FIFO 1/4 full 0x3 - FIFO 1/2 full
3	dma_mode	W	0	<p>DMA Mode (Note: UART1 only)</p> <p>This determines the DMA signaling mode used for the DMS_TX_REQ_N and DMA_RX_REQ_N output signals when additional DMA handshaking signals are not selected (DMA_EXTRA = NO).</p> <p>0 = Mode 0 1 = Mode 1</p> <p>Note: When using Mode 1 with UART1, the following constraints apply. For the Tx FIFO: tx_empty_trigger can be 0, 1 or 2, and the maximum transfer length is 8 bytes. For the Rx FIFO: rcvr_trigger can be 0, 1, 2 or 3.</p>
2	xmit_fifo_reset	W_SC	0	<p>Tx FIFO Reset</p> <p>This resets the control portion of the transmit FIFO and treats the FIFO as empty. This also deasserts the DMA Tx request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA = YES).</p> <p>Note that this bit is self-clearing. It is not necessary to clear this bit.</p>

Bits	Name	Attr	Reset	Description
1	rcvr_fifo_reset	W_SC	0	<p>Rx FIFO Reset</p> <p>This resets the control portion of the receive FIFO and treats the FIFO as empty. This also deasserts the DMA Rx request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA = YES).</p> <p>Note that this bit is self-clearing. It is not necessary to clear this bit.</p>
0	fifo_enable	W	0	<p>FIFO Enable</p> <p>This enables/disables the transmit (Tx) and receive (Rx) FIFOs. Whenever the value of this bit is changed both the Tx and Rx controller portion of FIFOs is reset.</p> <p>Do not change this register when the UART is operating. The UART must be reset by the software (e.g., UART_srr fields rfr, xfr, and ur) after changing fifo_enable.</p>

Table 17-9. FIFO Control Register (FCR).

17.3.2.7 **UART_lcr** Register

The Line Control register (**UART_lcr**) controls the format of the data that is transmitted and received by the UART.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7	dlab	RW	0	<p>Divisor Latch Access Bit</p> <p>Setting this bit enables reading and writing of the Divisor Latch register (UART_dll and UART_dlh) to set the baud rate of the UART. This bit must be cleared after initial baud rate setup in order to access other registers.</p>
6	break	RW	0	<p>Break Control</p> <p>Setting this bit sends a break signal by holding the SOUT line low (when not in Loopback mode, as determined by Modem Control register (UART_mcr) bit 4, until break is cleared. When in Loopback mode, the break condition is internally looped back to the receiver.</p>
5				Reserved
4	eps	RW	0	<p>Even Parity Select</p> <p>This is used to select between even and odd parity.</p> <p>0 - An odd number of logic 1s is transmitted or checked</p> <p>1 - An even number of logic 1s is transmitted or checked</p>
3	pen	RW	0	<p>Parity Enable</p> <p>This bit is used to enable and disable parity generation and detection in transmitted and received serial character respectively.</p> <p>0 - Disable</p> <p>1 - Enable</p>

Bits	Name	Attr	Reset	Description
2	stop	RW	0	<p>Stop</p> <p>This is used to select the number of stop bits per character that the peripheral transmits and receives. If set to zero, one stop bit is transmitted in the serial data.</p> <p>If set to one and the data bits are set to 5 (LCR[1:0] set to zero) one and a half stop bits is transmitted. Otherwise, two stop bits are transmitted.</p> <p>Note that regardless of the number of stop bits selected, the receiver checks only the first stop bit.</p> <p>0 - 1 stop bit 1 - 1.5 stop bits when UART_Lcr[1:0] or dls is 0, else 2 stop bits</p>
1:0	dls (cls)	RW	0	<p>Data Length Select (or previously CLS)</p> <p>This is used to select the number of data bits per character that the peripheral transmits and receives. The number of bits that may be selected are as follows:</p> <p>0x0 - 5 bits 0x1 - 6 bits 0x2 - 7 bits 0x3 - 8 bits</p>

Table 17-10. UART Line Control Register (LCR).

17.3.2.8 UART_mcr Register

Bits	Name	Attr	Reset	Description
31:6				Reserved
5	afce	RW	0	0 - Auto Flow Control Mode disabled 1 - Auto Flow Control Mode enabled
4	loopback	RW	0	<p>LoopBack Bit</p> <p>This is used to put the UART into a diagnostic mode for test purposes.</p> <p>If operating in UART mode (SIR_MODE != Enabled or not active, UART_mcr[6] is 0), data on the SOUT line is held high, while serial data output is looped back to the sin line, internally. In this mode all the interrupts are fully functional. Also, in loopback mode, the modem control inputs (DSR_N, UART_AHB_CTS_N, RI_N, DCD_N) are disconnected and the modem control outputs (DTR_N, UART_AHB_RTS_N, OUT1_N, OUT2_N) are looped back to the inputs, internally.</p>
3	out2	RW	0	<p>OUT2</p> <p>This is used to directly control the user-designated Output2 (OUT2_N) output. The value written to this location is inverted and driven out on OUT2_N, that is:</p> <p>0 - OUT2_N deasserted (logic 1) 1 - OUT2_N asserted (logic 0)</p> <p>Note that in Loopback mode (UART_mcr[4] is 1), the OUT2_N output is held inactive high while the value of this location is internally looped back to an input.</p>

Bits	Name	Attr	Reset	Description
2	out1	RW	0	<p>OUT1 This is used to directly control the user-designated Output1 (OUT1_N) output. The value written to this location is inverted and driven out on OUT1_N, that is:</p> <p>0 - OUT1_N deasserted (logic 1) 1 - OUT1_N asserted (logic 0)</p> <p>Note that in Loopback mode (UART_mcr[4] is 1), the OUT1_N output is held inactive high while the value of this location is internally looped back to an input.</p>
1	rts	RW	0	<p>Request to Send This is used to directly control the Request to Send (UART_AHB_RTS_N) output. The Request To Send (UART_AHB_RTS_N) output is used to inform the modem or dataset that the UART is ready to exchange data. When Auto RTS Flow Control is not enabled (UART_mcr[5] is 0), the UART_AHB_RTS_N signal is set low by programming MCR[1] (RTS) to a high. In Auto Flow Control, AFCE_MODE == Enabled and active (UART_mcr[5] is 1) and FIFOs enable (UART_fcr[0] is 1), the UART_AHB_RTS_N output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (UART_AHB_RTS_N is inactive high when above the threshold). The UART_AHB_RTS_N signal is deasserted when UART_mcr[1] is set low.</p> <p>Note that in Loopback mode (UART_mcr[4] is 1), the UART_AHB_RTS_N output is held inactive high while the value of this location is internally looped back to an input.</p> <p>Note that auto flow control mode is enabled when afce is 1. Do not program UART_mcr without turning off auto flow control (set afce to 0).</p>
0	dtr	RW	0	<p>Data Terminal Ready This bit directly controls the Data Terminal Ready (DTR_N) output. The value written to this location is inverted and driven out on DTR_N, that is:</p> <p>0 - DTR_N deasserted (logic 1) 1 - DTR_N asserted (logic 0)</p> <p>The Data Terminal Ready output is used to inform the modem or data set that the UART is ready to establish communications.</p> <p>Note that in Loopback mode (UART_mcr[4] is 1), the DTR_N output is held inactive high while the value of this location is internally looped back to an input.</p>

Table 17-11. UART Modem Control Register (MCR).

17.3.2.9 **UART_Isr** Register

The Line Status register (**UART_Isr**) contains status of the receiver and transmitter data transfers. This status register can be read by the programmer at any time.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7	rfe	R	0	<p>Receiver FIFO Error bit This bit is only relevant when FIFO_MODE != NONE AND FIFOs are enabled (UART_fcr[0] is 1). This indicates if there is at least one parity error, framing error, or break indication in the FIFO. 0 - No error in Rx FIFO 1 - Error in Rx FIFO</p> <p>This bit is cleared when UART_Isr is read and the character with the error is at the top of the receiver FIFO and there are no subsequent errors in the FIFO.</p>
6	temt	R	1	<p>Transmitter Empty If in FIFO mode (FIFO_MODE != NONE) and FIFOs enabled (UART_fcr[0] is 1), this bit is set whenever the Transmitter Shift register and the FIFO are both empty. If in non-FIFO mode or FIFOs are disabled, this bit is set whenever the Transmitter Holding register (UART_thr) and the Transmitter Shift register are both empty.</p>
5	thre	R	1	<p>Transmit Holding Register Empty THRE If THRE mode is disabled (the UART_iер bit 7 (ptime) is set to zero) and regardless of FIFO's being implemented/enabled or not, this bit indicates that UART_thr or Tx FIFO is empty. This bit is set whenever data is transferred from UART_thr or Tx FIFO to the Transmitter Shift register and no new data has been written to UART_thr or Tx FIFO. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled (UART_iер bit etbei). If FIFO_MODE != NONE and this mode is active (UART_iер bit 7 is set to one), the functionality is switched to indicate the transmitter FIFO is full, and no longer controls THRE interrupts, which are then controlled by the UART_fcr bits 5:4 threshold setting.</p>
4	bi	R	0	<p>Break Interrupt This is used to indicate the detection of a break sequence on the serial input data. If in UART mode, it is set whenever the serial input, sin, is held in a logic 0 state for longer than the sum of start time + data bits + parity + stop bits. In FIFO mode, the character associated with the break condition is carried through the FIFO and is revealed when the character is at the top of the FIFO. Reading UART_Isr clears bi. In the non-FIFO mode, the bi indication occurs immediately and persists until UART_Isr is read.</p>

Bits	Name	Attr	Reset	Description
3	fe	R	0	<p>Framing Error</p> <p>This bit indicates a framing error has occurred when the receiver does not detect a valid stop bit in the received data. Because the framing error is associated with a character received, it is revealed when the character with the framing error is at the top of the FIFO. When a framing error occurs, the UART attempts to resynchronize. It does this by assuming that the error was due to the start bit of the next character and then continues receiving the other bit data, and/or parity and stop.</p> <p>The fe bit is set if a break interrupt bi has occurred (UART_Isr[4]).</p> <p>0 - No framing error 1 - Framing error</p> <p>Note that oe, pe and fe are reset when UART_Isr is read.</p>
2	pe	R	0	<p>Parity Error</p> <p>This bit indicates a parity error in the receiver if the Parity Enable (pen) bit (UART_Icr[3]) is set. In the FIFO mode, since the parity error is associated with a character received, it is revealed when the character with the parity error arrives at the top of the FIFO. It should be noted that the Parity Error (pe) bit (UART_Isr[2]) is set if a break interrupt has occurred, as indicated by Break Interrupt (bi) bit (UART_Isr[4]).</p> <p>0 - No parity error 1 - Parity error</p> <p>Note that oe, pe and fe are reset when UART_Isr is read.</p>
1	oe	R	0	<p>Overrun Error</p> <p>This bit indicates the occurrence of an overrun error. This error occurs if a new data character was received before the previous data was read.</p> <p>In non-FIFO mode, the oe bit is set when a new character arrives in the receiver before the previous character was read from UART_rbr. When this happens, the data in UART_rbr is overwritten. In the FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver. The data in the FIFO is retained and the data in the receive shift register is lost.</p> <p>0 - No overrun error 1 - Overrun error</p> <p>Note that oe, pe and fe are reset when UART_Isr is read.</p>
0	dr	R	0	<p>Data Ready</p> <p>This is used to indicate that the receiver contains at least one character in UART_rbr or the receiver FIFO.</p> <p>0 - No data ready 1 - Data ready</p> <p>This bit is cleared when the UART_rbr is read in non-FIFO mode, or when the receiver FIFO is empty, in FIFO mode.</p>

Table 17-12. *UART Line Status Register (LSR)*.

17.3.2.10 UART_msr Register

In UART0 this register is not used. The Modem Status register (UART_msr) contains the current status of the modem control input lines and indicates whether they have changed.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7	dcd	R	0	<p>Data Carrier Detect</p> <p>This is used to indicate the current state of the modem control line DCD_N. This bit is the complement of DCD_N. When the Data Carrier Detect input (DCD_N) is asserted it is an indication that the carrier has been detected by the modem or data set.</p> <p>0 - DCD_N input is deasserted (logic 1) 1 - DCD_N input is asserted (logic 0)</p> <p>In Loopback Mode (UART_mcr[4] is 1), dcd is the same as UART_mcr[3] or out2.</p>
6	ri	R	0	<p>Ring Indicator</p> <p>This bit indicates the current state of the modem control line for Ring Indicator input or RI_N. This bit is the complement of RI_N. When RI_N is asserted it indicates that a telephone ringing signal has been received by the modem or data set.</p> <p>0 - RI_N input is deasserted (logic 1) 1 - RI_N input is asserted (logic 0)</p> <p>In Loopback Mode (UART_mcr[4] is 1), ri is the same as UART_mcr[2] or out1.</p>
5	dsr	R	0	<p>Data Set Ready</p> <p>This is used to indicate the current state of the modem control line for Data Set Ready input or DSR_N. This bit is the complement of DSR_N. When DSR_N is asserted it indicates that the modem or data set is ready to establish communications with the UART.</p> <p>0 - DSR_N input is deasserted (logic 1) 1 - DSR_N input is asserted (logic 0)</p> <p>In Loopback Mode (UART_mcr[4] is 1), DSR is the same as UART_mcr[0] or dtr.</p>
4	cts	R	0	<p>Clear to Send</p> <p>This is used to indicate the current state of the modem control line UART_AHB_CTS_N. This bit is the complement of UART_AHB_CTS_N. When the Clear to Send input (UART_AHB_CTS_N) is asserted it is an indication that the modem or data set is ready to exchange data with the UART.</p> <p>0 - UART_AHB_CTS_N input is deasserted (logic 1) 1 - UART_AHB_CTS_N input is asserted (logic 0)</p> <p>In Loopback Mode (UART_mcr[4] is 1), cts is the same as UART_mcr[1] or rts.</p>

Bits	Name	Attr	Reset	Description
3	ddcd	R	0	<p>Delta Data Carrier Detect</p> <p>This indicates that the modem control line DCD_N has changed since the last time the UART_msr was read.</p> <p>0 - No change on DCD_N since last read of UART_msr 1 - Change on DCD_N since last read of UART_msr</p> <p>Reading the UART_msr clears the ddcd bit. In Loopback Mode (UART_mcr[4] is 1), ddcd reflects changes on UART_mcr[3] or out2.</p> <p>Note that if ddcd is not set and the DCD_N signal is asserted (low) and a reset occurs (software or otherwise), then the ddcd bit is set when the reset is removed if the DCD_N signal remains asserted.</p>
2	teri	R	0	<p>Trailing Edge of Ring Indicator</p> <p>This is used to indicate that a change on the input RI_N (from an active-low to an inactive-high state) has occurred since the last time the UART_msr was read.</p> <p>0 - No change on RI_N since last read of UART_msr 1 - Change on RI_N since last read of UART_msr</p> <p>Reading the UART_msr clears the teri bit. In Loopback Mode (UART_mcr[4] is 1), teri reflects when MCR[2] or out1 has changed state from high to a low.</p>
1	ddsr	R	0	<p>Delta Data Set Ready</p> <p>This is used to indicate that the modem control line DSR_N has changed since the last time the UART_msr was read.</p> <p>0 - No change on DSR_N since last read of UART_msr 1 - Change on DSR_N since last read of UART_msr</p> <p>Reading the UART_msr clears the ddsr bit. In Loopback Mode (UART_mcr[4] is 1), ddsr reflects changes on MCR[0] or dtr.</p> <p>Note that if the ddsr bit is not set and the DSR_N signal is asserted (low) and a reset occurs (software or otherwise), then the ddsr bit is set when the reset is removed if the DSR_N signal remains asserted.</p>
0	dcts	R	0	<p>Delta Clear to Send</p> <p>This is used to indicate that the modem control line UART_AHB_CTS_N has changed since the last time the UART_msr was read.</p> <p>0 - No change on CTSDSR_N since last read of UART_msr 1 - Change on CTSDSR_N since last read of UART_msr</p> <p>Reading the UART_msr clears the dcts bit. In Loopback Mode (UART_mcr[4] is 1), dcts reflects changes on MCR[1] or RTS.</p> <p>Note, if the dcts bit is not set and the UART_AHB_CTS_N signal is asserted (low) and a reset occurs (software or otherwise), then the dcts bit is set when the reset is removed if the UART_AHB_CTS_N signal remains asserted.</p>

Table 17-13. UART Modem Status Register (MSR).

17.3.2.11 UART_scr Register

The Scratchpad register (UART_scr) is an 8-bit read/write register for programmers to use as a temporary storage space. It has no defined purpose in the UART.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	scratchpad	RW	0	Scratchpad Has no defined purpose in the UART. Use for temporary storage.

Table 17-14. *UART Scratchpad Register (SCR)*.

17.3.2.12 UART_dmae Register

Bits	Name	Attr	Reset	Description
31:2				Reserved
1	tdmae	RW	0	Transmit DMA Enable (UART1 only). This enables/disables the transmit DMA function.
0	rdmae	RW	0	Receive DMA Enable (UART1 only). This enables/disables the receive DMA function.

Table 17-15. *UART DMA Enable (UART1) Register (DMAE)*.

17.3.2.13 UART_usr Register

Bits	Name	Attr	Reset	Description
31:5				Reserved
4	rff	R	0	Receive FIFO Full This bit is valid only when fifo_stat = YES. This is used to indicate that the receive FIFO is completely full. 0 - Receive FIFO not full 1 - Receive FIFO full This bit is cleared when the Rx FIFO is no longer full.
3	rfne	R	0	Receive FIFO Not Empty This bit is used to indicate that the receive FIFO contains one or more enattempts. 0 - Receive FIFO is empty 1 - Receive FIFO is not empty This bit is cleared when the Rx FIFO is empty.
2	tfe	R	1	Transmit FIFO Empty This bit is valid only when fifo_stat = YES. This is used to indicate that the transmit FIFO is completely empty. 0 - Transmit FIFO is not empty 1 - Transmit FIFO is empty This bit is cleared when the Tx FIFO is no longer empty.

Bits	Name	Attr	Reset	Description
1	tfnf	R	1	<p>Transmit FIFO Not Full This bit is used to indicate that the transmit FIFO is not full. 0 - Transmit FIFO is full 1 - Transmit FIFO is not full This bit is cleared when the Tx FIFO is full.</p>
0	busy	R	0	<p>UART Busy Clearing this bit indicates that the UART is idle or inactive. 0 - UART is idle or inactive 1 - UART is busy (actively transferring data)</p> <p>Note that it is possible for busy to be cleared even though a new character may have been sent from another device. That is, if the UART has no data in UART_thr and UART_rbr; there is no transmission in progress; and a start bit of a new character has just reached the UART. This is because a valid start is not seen until the middle of the bit period, and this duration depends on the baud divisor which has been programmed. If a second system clock has been implemented, the assertion of this bit also is delayed by several cycles of the slower clock.</p>

Table 17-16. *UART Status Register (USR).*

17.3.2.14 **UART_tfl** Register

Bits	Name	Attr	Reset	Description
31:13				Reserved
12:0	tx_fifo	R	0	<p>Transmit FIFO Level This indicates the number of data enattempts in the transmit FIFO.</p>

Table 17-17. *UART Transmit FIFO Level Register (TFL).*

17.3.2.15 **UART_rfl** Register

Bits	Name	Attr	Reset	Description
31:13				Reserved
12:0	rx_fifo	R	0	<p>Receive FIFO Level This indicates the number of data enattempts in the receive FIFO.</p>

Table 17-18. *UART Receive FIFO Level Register (RFL).*

17.3.2.16 **UART_srr** Register

Bits	Name	Attr	Reset	Description
31:5				Reserved

Bits	Name	Attr	Reset	Description
2	xfr	W_SC	0	<p>Tx FIFO Reset</p> <p>This is a shadow register for the Tx FIFO Reset bit (UART_fcr[2]). This can be used to remove the burden on the software to store previously written UART_fcr values (which are relatively static) to reset the transmit FIFO. This resets the control portion of the transmit FIFO and treats the FIFO as empty. This also deasserts the DMA Tx request and single signals when additional DMA handshaking signals are selected.</p> <p>Note that this bit is self-clearing. It is not necessary to clear this bit. Dependencies are that writes have no effect when FIFO mode is not enabled.</p>
1	rfr	W_SC	0	<p>Rx FIFO Reset</p> <p>This is a shadow register for the Rx FIFO Reset bit (UART_fcr[1]). This can be used to remove the burden on the software to store previously written UART_fcr values (which are relatively static) to reset the receive FIFO. This resets the control portion of the receive FIFO and treats the FIFO as empty. This also deasserts the DMA Rx request and single signals when additional DMA handshaking signals are selected.</p> <p>Note that this bit is self-clearing. It is not necessary to clear this bit. Dependencies are that writes have no effect when FIFO mode is not enabled.</p>
0	ur	W	0	<p>UART Reset</p> <p>This asynchronously resets the UART and synchronously removes the reset assertion. For a two-clock implementation both PCLK and SCLK domains are reset.</p> <p>Note that this bit does not auto clear. When ur is asserted, the programmer should deassert ur as needed.</p>

Table 17-19. *UART Software Reset Register (SRR)*.

17.3.2.17 **UART_htx** Register

Bits	Name	Attr	Reset	Description
31:13				Reserved
12:0	halt_tx	RW	0	<p>This register is used to halt transmissions for testing, allowing the transmit FIFO to be filled by the master when FIFOs are implemented and enabled.</p> <p>0 - Halt Tx disabled 1 - Halt Tx enabled</p> <p>Note that if FIFOs are implemented and not enabled, setting the halt Tx register has no effect on operation. Dependencies are that writes have no effect when FIFO mode is not enabled.</p>

Table 17-20. *UART Halt Tx Register (HTX)*.

17.3.2.18 UART_cpr Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	fifo_mode	R	1	0x00 - 0 0x01 - 16 0x02 - 32 to 0x80 - 2048 0x81 - 0xFF - Reserved
15:14				Reserved
13	dma_extra	R	0	0 - False 1 - True
12	uart_add_encoded_params	R	1	0 - False 1 - True
11	shadow	R	1	0 - False 1 - True
10	fifo_stat	R	1	0 - False 1 - True
9	fifo_access	R	1	0 - False 1 - True
8	additional_feat	R	1	0 - False 1 - True
7	sir_lp_mode	R	0	0 - False 1 - True
6	sir_mode	R	0	0 - False 1 - True
5	thre_mode	R	1	0 - False 1 - True
4	afce_mode	R	1	0 - False 1 - True
3:2				Reserved
1:0	apb_data_width	R	2	0x0 - 8 bits 0x1 - 16 bits 0x2 - 32 bits 0x3 - Reserved

Table 17-21. UART Component Parameter Register (CPR).

17.3.2.19 UART_ucv Register

Bits	Name	Attr	Reset	Description
31:13				Reserved
12:0	uart_component	R	0x3130_302A	ASCII value for each number in the version. For example 31_30_30_2A represents the version 1.00#

Table 17-22. UART Component Version Register (UCV).

17.3.2.20 UART_ctr Register

Bits	Name	Attr	Reset	Description
31:13				Reserved
12:0	peripheral_id	R	0x414D_4241	This register contains the peripheral identification code.

Table 17-23. *UART Component Type Register (CTR)*.

For Confidential
For HAOTEX Only

18. IDC

This chapter contains information regarding the A12 Inter-Integrated Circuit (IDC) interface for two-wire communication. The chapter is organized as follows:

- [\(Section 18.1\) IDC: Overview](#)
- [\(Section 18.2\) IDC: Clock](#)
- [\(Section 18.3\) IDC: Registers](#)

18.1 IDC: Overview

The A12 IDC interface allows communication with external IDC devices with bit rates of up to 400 Kbps. There are up to three IDC controller interfaces (IDC0, IDC1 and IDC2) included on the A12 chip. These interfaces are located on the APB bus at the following base addresses:

- IDC0: 0xE800.3000
- IDC1: 0xE800.1000 (dedicated for use with HDMI) ( A12 chips only)
- IDC2: 0xE800.7000

18.1.1 IDC Turbo Mode Operation

The A12 IDC interface provides an operational mode which utilizes a 63-entry FIFO to buffer the commands and data bytes transmitted to a slave. This is referred to as Turbo Mode (and is also occasionally referred to as burst mode). Registers **IDC_fmcontrol** and **IDC_fmdata** ([Section 18.3.2.6](#)) are the write ports to the control portion and the data portion, respectively, of the FIFO. Note that Turbo Mode is intended for write operations only.

The **IDC_fmcontrol** and **IDC_fmdata** registers behave in a similar way to the **IDC_control** ([Section 18.3.2.2](#)) and **IDC_data** ([Section 18.3.2.3](#)) registers. The difference is that the **IDC_fmcontrol** and **IDC_fmdata** registers allow the software to write a batch of commands and data sequences rapidly and without the accompanying wait for each write onto the IDC bus to complete.

Note that when switching from Turbo Mode to Non-Turbo mode, the software must ensure that the FIFOs used during Turbo Mode are empty.

18.2 IDC: Clock

The period for a given IDC controller is configured according to:

$$\text{IDC Clock Frequency} = (\text{gclk_apb}) / ((4 + \text{dutycycle}[1:0]) * (\text{prescale}[15:0] + 1) + 2)$$

where the APB bus clock (gclk_apb) is used with a divider. The field prescale[15:0] equals the 16-bit value specified in the **IDC_prescaleh** and **IDC_prescalel** registers.

18.3 IDC: Registers

18.3.1 IDC Registers: Map

Register Offset	Register Name	Description
0x00	IDC_enable	Enable
0x04	IDC_control	Control
0x08	IDC_data	Data
0x0C	IDC_status	Status
0x10	IDC_prescalel	Prescaler (low byte)
0x14	IDC_prescaleh	Prescaler (high byte)
0x18	IDC_fmcontrol	FIFO Turbo Mode control
0x1C	IDC_fmdata	FIFO Turbo Mode data
0x20	IDC_prescalehs	Prescaler for High-Speed Mode
0x24	IDC_dutycycle	Duty cycle setting for Standard (Non-High Speed) Mode
0x28	IDC_stretch	Extend SCL clock for START/STOP setup and hold timing

Table 18-1. APB Registers of the IDC Interface.

18.3.2 IDC Registers: Detail

18.3.2.1 IDC_enable Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	en	RW	0	0 - IDC disabled (conserve power) 1 - IDC enabled

Table 18-2. IDC Enable Register.

18.3.2.2 IDC_control Register

Bits	Name	Attr	Reset	Description
31:6				Reserved
5	clear	WO		Write 1 to clear the 63-entry FIFO used during Turbo Mode
4	hs_mode	RW	0	1 - Enables High-Speed Mode
3	stop	RW	0	1 - Generates a STOP condition
2	start	RW	0	1 - Generates a START condition
1	if	RW	0	Interrupt flag. The software clears this flag by writing 0 0 - No interrupt 1 - Interrupt pending

Bits	Name	Attr	Reset	Description
0	ack	RW	0	Read: When receiving, 0/1 indicates ACK/NACK needs to be transmitted to the slave device. When transmitting, 0/1 indicates ACK/NACK received from slave. Write: Write 0/1 to transmit ACK/NACK to slave device.

Table 18-3. IDC Control Register.

18.3.2.3 IDC_data Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	data	RW	0	IDC data register for single-byte operations only Write: Send data Read: Receive data

Table 18-4. IDC Data Register.

18.3.2.4 IDC_status Register

Bits	Name	Attr	Reset	Description
31:14				Reserved
13:8	fifocnt	R	0	FIFO count of empty entries
7:4	state	R	0	IDC Finite State Machine (FSM) status
3				Reserved
2	fifoemp	R	0	FIFO empty
1	fifofull	R	0	FIFO full
0	mode	R	0	0 - Master transmitter 1 - Master receiver

Table 18-5. IDC Status Register.

18.3.2.5 IDC_prescalel / IDC_prescaleh Registers

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	scale	RW	0	See description of IDC period in Section 18.1 "IDC: Overview"

Table 18-6. IDC Prescaler Registers (Low and High).

18.3.2.6 IDC_fmcontrol / IDC_fmdata Registers

The **IDC_fmcontrol** and **IDC_fmdata** registers are the write ports to the control portion and the data portion, respectively, of the FIFO when using Turbo Mode. Refer to [Section 18.1.1](#) for further information on this mode.

The following is an example of writing 61 data words to the IDC device using the **IDC_fmcontrol** and **IDC_fmdata** registers.

```

@ program base address of idc
    mov r0, #APB_BASE_ADDRESS           @ load up idc base address
register
    orr r0, r0, #IDC_DEVICE_OFFSET
    mov r1, #0x01                      @ configure prescale registers
    str r1, [r0, #IDC_PRE_SCALE_LOW]
    mov r1, #0x01                      @ enable the module
    str r1, [r0, #IDC_ENABLE]
    mov r1, #0x04                      @ push START command to FIFO
    str r1, [r0, #IDC_FMCTRL]          @ place slave address and r/w bit
into FIFO
    mov r1, #0x2A                      @ slave address = 0x15 / r/w bit = 0 (w)
    str r1, [r0, #IDC_FMDATA]
    mov r1, #0x0
fill_data_FIFO:                         @ place 61 data items in FIFO
    str r1, [r0, #IDC_FMDATA]
    add r1, r1, #0x01
    cmp r1, #0x3d
    bne fill_data_FIFO
    mov r1, #0x0A                      @ push INT/STOP command to control FIFO
    str r1, [r0, #IDC_FMCTRL]
@
@ software can then wait for IDC Interrupt or poll the "ack" field of
the IDC_Control register to
@ if the write sequence has completed.
@

```

Note that if the hardware fails to see an acknowledgment from the device in executing a sequence of writes (e.g., the address is incorrect), the IDC hardware will hang. To avoid this, the software should implement an appropriate time-out and error recovery mechanism. In recovering the IDC hardware from a hang state, the software should clear the FIFO.

Bits	Name	Attr	Reset	Description
31:5				Reserved
4	hs_mode	RW	0	1 - Enables High-Speed Mode
3	stop	RW	0	1 - The next process generates a STOP condition
2	start	RW	0	1 - The next process generates a START condition
1	is	RW	0	Enable bit for FIFO mode Assert interrupt to ARM at the completion of this command
0				Reserved

Table 18-7. IDC Turbo-Mode Control Register.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	fmdata	RW	0	Data for Turbo Mode

Table 18-8. IDC Turbo-Mode Data Register.

18.3.2.7 IDC_prescalerhs Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	prescalerhs	RW	0	Prescaler for High-Speed Mode

Table 18-9. IDC Turbo-Mode Prescaler Register.

18.3.2.8 IDC_dutycycle Register

Bits	Name	Attr	Reset	Description
31:4				Reserved
3	sda	RW	1	SDA (data line) pin current source enable
2	scl	RW	1	SCL (clock line) pin current source enable
1:0	dutycycle	RW	1	Duty cycle setting for Standard (Non-High Speed) Mode 0 - 1:1 1 - 2:3 2 - 1:2

Table 18-10. IDC Standard (Non-Turbo) Mode Duty Cycle Register.

18.3.2.9 IDC_stretch Register

Bits	Name	Attr	Reset	Description
31:4				Reserved
3:0	stretchscl	RW	0	Extend the SCL clock START/STOP setup and hold timing

Table 18-11. IDC SCL Clock Stretch Register.

19. ADC, PWC AND RTC

This chapter details the interfaces for the Analog-to-Digital Converter (ADC), Power Controller (PWC) and Real-Time Clock (RTC) as follows:

- [\(Section 19.1\) ADC / PWC / RTC: Overview](#)
- [\(Section 19.2\) ADC / PWC / RTC: Registers](#)
- [\(Section 19.3\) ADC / PWC / RTC: ADC Programming Notes](#)
- [\(Section 19.4\) ADC / PWC / RTC: PWC and RTC Programming Notes](#)

19.1 ADC / PWC / RTC: Overview

The A12 SoC includes up to four channels of 12-bit analog-to-digital converters intended for low-speed monitoring. Potential uses include battery voltage monitoring, zoom position sense, and general purpose input (digital on/off events encoded as analog voltage). The ADC interface provides threshold control, allowing interrupts to be sent to the VIC if, for example, a sampled voltage exceeds the maximum or minimum threshold value. This capability enables the software to use an interrupt scheme to monitor voltage changes instead of polling the ADC channel voltages continuously. The ADC can handle inputs between 0 and 3.3 V.

The ADC register is located at offset E801.D000 on the APB bus.

The PWC and RTC modules share a register space located on the APB bus at 0xE801.5000. This register space is also used for GPIO pull-control functions.

19.2 ADC / PWC / RTC: Registers

The ADC / PWC / RTC instances on the APB bus use 32-bit registers as shown below.

19.2.1 ADC / PWC / RTC Registers: Map

Register Off-set	Register Name	Description
0x000	ADC_status	ADC Status
0x004	ADC_control	ADC Control
0x008	ADC_counter	ADC Counter to Delay Tuning Between Sampling Points
0x00C	ADC_slot_num	ADC Time Slot Number
0x010	ADC_slot_period	ADC Slot Period
0x014 - 0x01C		Reserved
0x020	PWC_seq1	PWC Set Delay Between PWC_WKUP and PWC_PSEQ1 Signals
0x024	PWC_seq2	PWC Set Delay Between PWC_PSEQ1 and PWC_PSEQ2 Signals
0x028	PWC_seq3	PWC Set Delay Between PWC_PSEQ2 and PWC_PSEQ3 Signals
0x02C	PWC_alat_write	RTC Alarm Time Programming
0x030	PWC_curt_write	RTC Current Time Programming
0x034	PWC_curt_read	RTC Current Time Value
0x038	PWC_alat_read	RTC Alarm Time Value
0x040	PWC_reset	RTC Reset
0x044	ADC_ctrl_int_table	ADC Control Interrupt Table
0x048	ADC_data_int_table	ADC Channel Interrupt Table
0x04C	ADC_fifo_int_table	ADC FIFO Status Interrupt Table
0x050	ADC_err_status	ADC Error Status Table
0x054 - 0x078		Reserved
0x07C	PWC_reg_wo	PWC Extra Wakeup Status
0x080	GPIO_pull_en_0	Enables Pull-Up/Pull-Down function for GPIO[0:31]
0x084	GPIO_pull_en_1	Enables Pull-Up/Pull-Down function for GPIO[32:63]
0x088	GPIO_pull_en_2	Enables Pull-Up/Pull-Down function for GPIO[64:95]
0x08C	GPIO_pull_en_3	Enables Pull-Up/Pull-Down function for GPIO[96:113]
0x090 - 0x100		Reserved
0x094	GPIO_pull_sel_0	Selects Pull-Up/Pull-Down direction for GPIO[0:31]
0x098	GPIO_pull_sel_1	Selects Pull-Up/Pull-Down direction for GPIO[32:63]
0x09C	GPIO_pull_sel_2	Selects Pull-Up/Pull-Down direction for GPIO[64:95]
0x0A0	GPIO_pull_sel_3	Selects Pull-Up/Pull-Down direction for GPIO[96:113]
0x0A4 - 0x10C		Reserved
0x0A8	PWC_dnalert	PWC is forced off by pressing PWC_WKUP longer than 16 s
0x0AC	PWC_lbat	Signals Low Battery Level
0x0B0		Reserved
0x0B4	PWC_reg_sta	PWC Current Value of Status Bits
0x0B8	PWC_disp2	PWC Ties PWC_PSEQ2 low
0x0BC	PWC_disp3	PWC Ties PWC_PSEQ3 low
0x0C0	PWC_sta	PWC Sets Status Bits
0x0C4 - 0x0CC		Reserved
0x0D0	PWC_seq4	PWC Sets Delay Between PWC_PSEQ3 and PWC_RSTOB

Register Off-set	Register Name	Description
0x0D4		Reserved
0x0D8		Reserved
0x0DC		Reserved
0x0E0	PWC_enp1	PWC Ties PWC_PSEQ1 high
0x0E4	PWC_enp2	PWC Ties PWC_PSEQ2 high
0x0E8	PWC_enp3	PWC Ties PWC_PSEQ3 high
0x0EC		Reserved
0x0F8	PWC_disp1	PWC Ties PWC_PSEQ1 low
0x0FC		Reserved
0x100	ADC_slot_ctrl_0	ADC Control for Slot 0
0x104	ADC_slot_ctrl_1	ADC Control for Slot 1
0x108	ADC_slot_ctrl_2	ADC Control for Slot 2
0x10C	ADC_slot_ctrl_3	ADC Control for Slot 3
0x110	ADC_slot_ctrl_4	ADC Control for Slot 4
0x114	ADC_slot_ctrl_5	ADC Control for Slot 5
0x118	ADC_slot_ctrl_6	ADC Control for Slot 6
0x11C	ADC_slot_ctrl_7	ADC Control for Slot 7
0x120	ADC_int_ctrl_0	ADC Channel 0 Interrupt Control
0x124	ADC_int_ctrl_1	ADC Channel 1 Interrupt Control
0x128	ADC_int_ctrl_2	ADC Channel 2 Interrupt Control
0x12C	ADC_int_ctrl_3	ADC Channel 3 Interrupt Control
0x150	ADC_data_0	ADC Data for Channel 0
0x154	ADC_data_1	ADC Data for Channel 1
0x158	ADC_data_2	ADC Data for Channel 2
0x15C	ADC_data_3	ADC Data for Channel 3
0x180	ADC_fifo_ctrl_0	ADC FIFO 0 Control
0x184	ADC_fifo_ctrl_1	ADC FIFO 1 Control
0x188	ADC_fifo_ctrl_2	ADC FIFO 2 Control
0x18C	ADC_fifo_ctrl_3	ADC FIFO 3 Control
0x190	ADC_fifo_ctrl	ADC FIFO Control
0x194 - 0x19C		Reserved
0x1A0	ADC_fifo_status_0	ADC FIFO 0 Status
0x1A4	ADC_fifo_status_1	ADC FIFO 1 Status
0x1A8	ADC_fifo_status_2	ADC FIFO 2 Status
0x1AC	ADC_fifo_status_3	ADC FIFO 3 Status
0x1B0	ADC_ec_control	Event counter control register
0x1B4	ADC_ec_counter	Event counter
0x1B8	ADC_ec_counter_th	Event counter interrupt threshold
0x1BC	ADC_ec_param_2	Event counter parameter register (Reserved)
0x1C0	ADC_ec_adc	Event counter ADC channel register
0x1C4	ADC_ec_refval	Event counter reference values register
0x1C8	ADC_ec_reshape	Event counter ADC reshaping register
0x1CC - 0x19C		Reserved
0x200 - 0x27F	ADC_fifo_data_0	0th FIFO Read Offset
0x280 - 0x2FF	ADC_fifo_data_1	1st FIFO Read Offset
0x300 - 0x37F	ADC_fifo_data_2	2nd FIFO Read Offset

Register Off-set	Register Name	Description
0x380 - 0x3FF	ADC_fifo_data_3	3rd FIFO Read Offset
0xC0	PWC_sts	RTC Status

Table 19-1. ADC Registers on the APB Bus.

i Note:

A number of the above-mentioned registers may not be applicable to specific A12 or A12m SoCs. Please consult the relevant chip datasheet for a comprehensive list of supported features and specifications.

For Confidential
For HAOTEX Only

19.2.2 ADC / PWC / RTC Registers: Detail

19.2.2.1 ADC_status Register

Bits	Name	Attr	Reset	Description
31:2				Reserved
1	sampling_done	R	0	1 - Signal reverts to 1 as the ADC completes sampling
0	adc_status	R	0	0 - Value reverts to 0 after the 'start' command and before the first conversion is complete 1 - At least one conversion since the last 'start' command is complete. Writing to this register clears this bit until the first valid sample.

Table 19-2. ADC Status Register.

19.2.2.2 ADC_control Register

Bits	Name	Attr	Reset	Description
31:4				Reserved
3	adc_start	RW	0	1 - Start ADC conversion (self-clear)
2	adc_enable	RW	0	Enable ADC 0 - Disable 1 - Enable
1	adc_mode	RW	0	Continuous or single mode 0 - Sample one iteration then stop 1 - Run the ADC continuously until stopped by software
0	adc_clear	RW	0	Software reset 1 - Clear ADC (self-clear)

Table 19-3. ADC Control Register.

19.2.2.3 ADC_counter Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	counter	RW	0	STC edge delay tuning in AHB clock cycles. This value can be roughly set to the number of "tuning cycles in HCLK period – 2".

Table 19-4. ADC Counter Register.

19.2.2.4 ADC_slot_num Register

Bits	Name	Attr	Reset	Description
31:3				Reserved
2:0	slot_num	RW	0	The number of valid time slots = slot_num + 1, valid range from 0 to 7

Table 19-5. ADC Time Slot Number Register.

19.2.2.5 ADC_slot_period Register

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:0	slot_period	RW	0	This value indicates the period of ADC sampling slots in CKIN clock cycles -1. The period shall be not less than the maximum of active channels multiplied by six (each channel takes six CKIN cycles) in one slot.

Table 19-6. ADC Slot Period Register.

19.2.2.6 PWC_seq[n] Registers

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	seq[n]	RW	0	PWC Sequence[n] Used to set delay time (mS)

Table 19-7. Registers for PWC Sequence Signal Delay Control.

There are four registers used for delay control:

1. **PWC_seq1** sets the delay between the **PWC_WKUP** and **PWC_PSEQ1** signals.
2. **PWC_seq2** sets the delay between the **PWC_PSEQ1** and **PWC_PSEQ2** signals.
3. **PWC_seq3** sets the delay between the **PWC_PSEQ2** and **PWC_PSEQ3** signals.
4. **PWC_seq4** sets the delay between the **PWC_PSEQ3** and **PWC_RSTOB** signals.

19.2.2.7 PWC_alat_write Register

Bits	Name	Attr	Reset	Description
31:0	pwc_alat_write	RW	0	Current alarm time programming in RTC Writing to this register sets the alarm clock value.

Table 19-8. RTC Alarm Time Programming Register.

19.2.2.8 PWC_curt_write Register

Bits	Name	Attr	Reset	Description
31:0	pwc_curt_write	RW	0	Current time programming in RTC Writing to this register sets the current wall clock time.

Table 19-9. RTC Current Time Programming Register.

19.2.2.9 PWC_curt_read Register

Bits	Name	Attr	Reset	Description
31:0	pwc_curt_read	R	0	Current time value in RTC This register should be read to retrieve the current wall clock time.

Table 19-10. RTC Current Time Value Register.
19.2.2.10 PWC_alat_read Register

Bits	Name	Attr	Reset	Description
31:0	pwc_alat_read	R	0	Alarm value in RTC This register should be read to retrieve the current alarm clock value.

Table 19-11. RTC Alarm Value Register.
19.2.2.11 PWC_sta Register

Bits	Name	Attr	Reset	Description
31:4				Reserved
3	wkup	R	0	RTC wake state with the PWC_WKUP pin
2	ala_wk	R	0	RTC alarm wake up state.
1				Reserved
0	rtc_clk	R	0	RTC clock value.

Table 19-12. RTC Status Register.
19.2.2.12 PWC_reset Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	reset	RW	0	Reset the RTC Set to 1 to clear the RTC status. Also used as PCRST strobe for alarm and current time setting. All bits must be refreshed at the same time.

Table 19-13. RTC Reset Register.
19.2.2.13 ADC_ctrl_int_table Register

Bits	Name	Attr	Reset	Description
31:2				Reserved
1	vcm_th_int	RW	0	Event counter interrupt

Bits	Name	Attr	Reset	Description
0	error_event	RW	0	Error event 0 - No ADC error 1 - ADC error occurs Reverts to 1 when ADC error occurs. Errors include (1) slot period error and (2) FIFO errors, such as an invalid setting or FIFO underflow.

Table 19-14. ADC Control Interrupt Table Register.

19.2.2.14 ADC_data_int_table Register

Bits	Name	Attr	Reset	Description
31:4				Reserved
3	data_th_int_3	RW	0	ADC channel 3 interrupt. ADC channel value exception
2	data_th_int_2	RW	0	ADC channel 2 interrupt. ADC channel value exception
1	data_th_int_1	RW	0	ADC channel 1 interrupt. ADC channel value exception
0	data_th_int_0	RW	0	ADC channel 0 interrupt. ADC channel value exception

Table 19-15. ADC Data Interrupt Table Register.

19.2.2.15 ADC_fifo_int_table Register

Bits	Name	Attr	Reset	Description
31:12				Reserved
11	fifo_over_int_3	RW	0	FIFO 3 overflow exception, latest FIFO data dropped
10	fifo_over_int_2	RW	0	FIFO 2 overflow exception, latest FIFO data dropped
9	fifo_over_int_1	RW	0	FIFO 1 overflow exception, latest FIFO data dropped
8	fifo_over_int_0	RW	0	FIFO 0 overflow exception, latest FIFO data dropped
7	fifo_under_int_3	RW	0	FIFO 3 underflow exception, error occurs
6	fifo_under_int_2	RW	0	FIFO 2 underflow exception, error occurs
5	fifo_under_int_1	RW	0	FIFO 1 underflow exception, error occurs
4	fifo_under_int_0	RW	0	FIFO 0 underflow exception, error occurs
3	fifo_th_int_3	RW	0	FIFO 3 interrupt. FIFO to be full
2	fifo_th_int_2	RW	0	FIFO 2 interrupt. FIFO to be full
1	fifo_th_int_1	RW	0	FIFO 1 interrupt. FIFO to be full
0	fifo_th_int_0	RW	0	FIFO 0 interrupt. FIFO to be full

Table 19-16. ADC FIFO Interrupt Table Register.

19.2.2.16 ADC_err_status Register

Bits	Name	Attr	Reset	Description
31:2				Reserved

Bits	Name	Attr	Reset	Description
1	fifo_param	R	0	FIFO parameter error 0 - Error not detected 1 - Invalid FIFO parameters or FIFO underflow
0	over_period	R	0	Over period error 0 - Error not detected 1 - ADC sampling period over its limit

Table 19-17. ADC Error Status Register.

19.2.2.17 PWC_reg_wo Register

Bits	Name	Attr	Reset	Description
31:3				Reserved
2:0	pwc_reg_wo	R	0	Extra wake up state 0x0 - No extra wake up 0x1 - Extra wake up with WK[0] signal (i.e., PWC_WKUP pin) 0x2 - Extra wake up with WK[1] signal (i.e., PWC_WKUP1 pin) 0x4 - Reserved

Table 19-18. RTC Extra Wakeup Status Register.

19.2.2.18 GPIO_pull_en_0 Register

Bits	Name	Attr	Reset	Description
31:0	gpio_pull_en[31:0]	RW	0x7FFF.FFFF	GPIO pin pull-enable control 0: Disable I/O pull control 1: Enable I/O pull control

Table 19-19. GPIO Pin Pull-Up / Pull-Down Enable 0 Register.

GPIO Pin Name	Bit
GPIO_0	gpio_pull_en[0]
GPIO_1	gpio_pull_en[1]
GPIO_2	gpio_pull_en[2]
GPIO_3	gpio_pull_en[3]
GPIO_4	gpio_pull_en[4]
GPIO_5	gpio_pull_en[5]
GPIO_6	gpio_pull_en[6]
SC_A0	gpio_pull_en[7]
SC_A1	gpio_pull_en[8]
SC_A2	gpio_pull_en[9]
SC_A3	gpio_pull_en[10]
SC_B0	gpio_pull_en[11]
SC_B1	gpio_pull_en[12]
SC_B2	gpio_pull_en[13]

GPIO Pin Name	Bit
SC_B3	gpio_pull_en[14]
SC_C0	gpio_pull_en[15]
SC_C1	gpio_pull_en[16]
SC_C2	gpio_pull_en[17]
SC_C3	gpio_pull_en[18]
SC_D0	gpio_pull_en[19]
SC_D1	gpio_pull_en[20]
SC_D2	gpio_pull_en[21]
SC_D3	gpio_pull_en[22]
SC_E0	gpio_pull_en[23]
TIMER0	gpio_pull_en[24]
TIMER1	gpio_pull_en[25]
TIMER2	gpio_pull_en[26]
IDCCLK	gpio_pull_en[27]
IDCDATA	gpio_pull_en[28]
IDC2CLK	gpio_pull_en[29]
IDC2DATA	gpio_pull_en[30]
IDC3CLK	gpio_pull_en[31]

Table 19-20. GPIO Pin Pull-Up / Pull-Down Enable 0 Register and Pin Name Detail.

19.2.2.19 GPIO_pull_en_1 Register

Bits	Name	Attr	Reset	Description
31:0	gpio_pull_en[63:32]	RW	0x1400.17E2	GPIO pin pull-enable control 0: Disable I/O pull control 1: Enable I/O pull control

Table 19-21. GPIO Pin Pull-Up / Pull-Down Enable 1 Register.

GPIO Pin Name	Bit
IDC3DATA	gpio_pull_en[32]
IR_IN	gpio_pull_en[33]
SSIOCLK	gpio_pull_en[34]
SSIOMOSI	gpio_pull_en[35]
SSIOMISO	gpio_pull_en[36]
SSIOENO	gpio_pull_en[37]
SSIOEN1	gpio_pull_en[38]
UART0RX	gpio_pull_en[39]
UART0TX	gpio_pull_en[40]
I2S_CLK	gpio_pull_en[41]
I2S_SI	gpio_pull_en[42]
I2S_SO	gpio_pull_en[43]
I2S_WS	gpio_pull_en[44]
CLK_AU	gpio_pull_en[45]

GPIO Pin Name	Bit
ENET_TXEN	gpio_pull_en[46]
ENET_TXD_0	gpio_pull_en[47]
ENET_TXD_1	gpio_pull_en[48]
ENET_RXD_0	gpio_pull_en[49]
ENET_RXD_1	gpio_pull_en[50]
ENET_RX_ER	gpio_pull_en[51]
ENET_CRS_DV	gpio_pull_en[52]
ENET_REF_CLK	gpio_pull_en[53]
WP	gpio_pull_en[54]
SMIO_0	gpio_pull_en[55]
SMIO_1	gpio_pull_en[56]
SMIO_2	gpio_pull_en[57]
SMIO_3	gpio_pull_en[58]
SMIO_4	gpio_pull_en[59]
SMIO_5	gpio_pull_en[60]
SMIO_6	gpio_pull_en[61]
SMIO_7	gpio_pull_en[62]
SMIO_8	gpio_pull_en[63]

Table 19-22. GPIO Pin Pull-Up / Pull-Down Enable 1 Register and Pin Name Detail.

19.2.2.20 GPIO_pull_en_2 Register

Bits	Name	Attr	Reset	Description
31:0	gpio_pull_en[95:64]	RW	0x197D.FE00	GPIO pin pull-enable control 0: Disable I/O pull control 1: Enable I/O pull control

Table 19-23. GPIO Pin Pull-Up / Pull-Down Enable 2 Register.

GPIO Pin Name	Bit
SMIO_9	gpio_pull_en[64]
SMIO_10	gpio_pull_en[65]
SMIO_11	gpio_pull_en[66]
SMIO_12	gpio_pull_en[67]
SMIO_13	gpio_pull_en[68]
SMIO_14	gpio_pull_en[69]
SMIO_15	gpio_pull_en[70]
SMIO_16	gpio_pull_en[71]
SMIO_17	gpio_pull_en[72]
SMIO_18	gpio_pull_en[73]
SMIO_19	gpio_pull_en[74]
SMIO_20	gpio_pull_en[75]
SMIO_21	gpio_pull_en[76]
SMIO_22	gpio_pull_en[77]

GPIO Pin Name	Bit
SMIO_23	gpio_pull_en[78]
SMIO_24	gpio_pull_en[79]
SMIO_25	gpio_pull_en[80]
SMIO_26	gpio_pull_en[81]
SMIO_27	gpio_pull_en[82]
SMIO_28	gpio_pull_en[83]
SMIO_29	gpio_pull_en[84]
SMIO_30	gpio_pull_en[85]
SMIO_31	gpio_pull_en[86]
SMIO_32	gpio_pull_en[87]
SMIO_33	gpio_pull_en[88]
HPD	gpio_pull_en[89]
CEC	gpio_pull_en[90]
SVSYNC	gpio_pull_en[91]
SHSYNC	gpio_pull_en[92]
VDO_OUT_0	gpio_pull_en[93]
VDO_OUT_1	gpio_pull_en[94]
VDO_OUT_2	gpio_pull_en[95]

Table 19-24. GPIO Pin Pull-Up / Pull-Down Enable 2 Register and Pin Name Detail.

19.2.2.21 GPIO_pull_en_3 Register

Bits	Name	Attr	Reset	Description
31:18				Reserved
17:0	gpio_pull_en[113:96]	RW	0x0	GPIO pin pull-enable control 0: Disable I/O pull control 1: Enable I/O pull control

Table 19-25. GPIO Pin Pull-Up / Pull-Down Enable 3 Register.

GPIO Pin Name	Bit
VDO_OUT_3	gpio_pull_en[96]
VDO_OUT_4	gpio_pull_en[97]
VDO_OUT_5	gpio_pull_en[98]
VDO_OUT_6	gpio_pull_en[99]
VDO_OUT_7	gpio_pull_en[100]
VDO_OUT_8	gpio_pull_en[101]
VDO_OUT_9	gpio_pull_en[102]
VDO_OUT_10	gpio_pull_en[103]
VDO_OUT_11	gpio_pull_en[104]
VDO_OUT_12	gpio_pull_en[105]
VDO_OUT_13	gpio_pull_en[106]
VDO_OUT_14	gpio_pull_en[107]

GPIO Pin Name	Bit
VDO_OUT_15	gpio_pull_en[108]
VDO_CLK	gpio_pull_en[109]
VDO_VSYNC	gpio_pull_en[110]
VDO_HSYNC	gpio_pull_en[111]
VDO_HVLD	gpio_pull_en[112]
VD_PWM	gpio_pull_en[113]

Table 19-26. GPIO Pin Pull-Up / Pull-Down Enable 3 Register and Pin Name Detail.

19.2.2.22 GPIO_pull_sel_0 Register

Bits	Name	Attr	Reset	Description
31:0	gpio_pull_sel[31:0]	RW	0x7FFF.FFFF	GPIO pin pull value selection when the corresponding pull-enable bit is set. 0: Pull Down 1: Pull High

Table 19-27. GPIO Pin Pull-Up / Pull-Down Direction Select 0 Register.

GPIO Pin Name	Bit
GPIO_0	gpio_pull_sel[0]
GPIO_1	gpio_pull_sel[1]
GPIO_2	gpio_pull_sel[2]
GPIO_3	gpio_pull_sel[3]
GPIO_4	gpio_pull_sel[4]
GPIO_5	gpio_pull_sel[5]
GPIO_6	gpio_pull_sel[6]
SC_A0	gpio_pull_sel[7]
SC_A1	gpio_pull_sel[8]
SC_A2	gpio_pull_sel[9]
SC_A3	gpio_pull_sel[10]
SC_B0	gpio_pull_sel[11]
SC_B1	gpio_pull_sel[12]
SC_B2	gpio_pull_sel[13]
SC_B3	gpio_pull_sel[14]
SC_C0	gpio_pull_sel[15]
SC_C1	gpio_pull_sel[16]
SC_C2	gpio_pull_sel[17]
SC_C3	gpio_pull_sel[18]
SC_D0	gpio_pull_sel[19]
SC_D1	gpio_pull_sel[20]
SC_D2	gpio_pull_sel[21]
SC_D3	gpio_pull_sel[22]
SC_E0	gpio_pull_sel[23]

GPIO Pin Name	Bit
TIMER0	gpio_pull_sel[24]
TIMER1	gpio_pull_sel[25]
TIMER2	gpio_pull_sel[26]
IDCCLK	gpio_pull_sel[27]
IDCDATA	gpio_pull_sel[28]
IDC2CLK	gpio_pull_sel[29]
IDC2DATA	gpio_pull_sel[30]
IDC3CLK	gpio_pull_sel[31]

Table 19-28. GPIO Pin Pull-Up / Pull-Down Direction Select 0 Register and Pin Name Detail.

19.2.2.23 GPIO_pull_sel_1 Register

Bits	Name	Attr	Reset	Description
31:0	gpio_pull_sel[63:32]	RW	0x1400.17E2	GPIO pin pull value selection when the corresponding pull-enable bit is set. 0: Pull Down 1: Pull High

Table 19-29. GPIO Pin Pull-Up / Pull-Down Direction Select 1 Register.

GPIO Pin Name	Bit
IDC3DATA	gpio_pull_sel[32]
IR_IN	gpio_pull_sel[33]
SSIOCLK	gpio_pull_sel[34]
SSIOMOSI	gpio_pull_sel[35]
SSIOMISO	gpio_pull_sel[36]
SSIOENO	gpio_pull_sel[37]
SSIOEN1	gpio_pull_sel[38]
UARTORX	gpio_pull_sel[39]
UARTOTX	gpio_pull_sel[40]
I2S_CLK	gpio_pull_sel[41]
I2S_SI	gpio_pull_sel[42]
I2S_SO	gpio_pull_sel[43]
I2S_WS	gpio_pull_sel[44]
CLK_AU	gpio_pull_sel[45]
ENET_TXEN	gpio_pull_sel[46]
ENET_TXD_0	gpio_pull_sel[47]
ENET_TXD_1	gpio_pull_sel[48]
ENET_RXD_0	gpio_pull_sel[49]
ENET_RXD_1	gpio_pull_sel[50]
ENET_RX_ER	gpio_pull_sel[51]
ENET_CRS_DV	gpio_pull_sel[52]
ENET_REF_CLK	gpio_pull_sel[53]

GPIO Pin Name	Bit
WP	gpio_pull_sel[54]
SMIO_0	gpio_pull_sel[55]
SMIO_1	gpio_pull_sel[56]
SMIO_2	gpio_pull_sel[57]
SMIO_3	gpio_pull_sel[58]
SMIO_4	gpio_pull_sel[59]
SMIO_5	gpio_pull_sel[60]
SMIO_6	gpio_pull_sel[61]
SMIO_7	gpio_pull_sel[62]
SMIO_8	gpio_pull_sel[63]

Table 19-30. GPIO Pin Pull-Up / Pull-Down Direction Select 1 Register and Pin Name Detail.

19.2.2.24 GPIO_pull_sel_2 Register

Bits	Name	Attr	Reset	Description
31:0	gpio_pull_sel[95:64]	RW	0x197D.FE00	GPIO pin pull value selection when the corresponding pull-enable bit is set. 0: Pull Down 1: Pull High

Table 19-31. GPIO Pin Pull-Up / Pull-Down Direction Select 2 Register.

GPIO Pin Name	Bit
SMIO_9	gpio_pull_sel[64]
SMIO_10	gpio_pull_sel[65]
SMIO_11	gpio_pull_sel[66]
SMIO_12	gpio_pull_sel[67]
SMIO_13	gpio_pull_sel[68]
SMIO_14	gpio_pull_sel[69]
SMIO_15	gpio_pull_sel[70]
SMIO_16	gpio_pull_sel[71]
SMIO_17	gpio_pull_sel[72]
SMIO_18	gpio_pull_sel[73]
SMIO_19	gpio_pull_sel[74]
SMIO_20	gpio_pull_sel[75]
SMIO_21	gpio_pull_sel[76]
SMIO_22	gpio_pull_sel[77]
SMIO_23	gpio_pull_sel[78]
SMIO_24	gpio_pull_sel[79]
SMIO_25	gpio_pull_sel[80]
SMIO_26	gpio_pull_sel[81]
SMIO_27	gpio_pull_sel[82]
SMIO_28	gpio_pull_sel[83]

GPIO Pin Name	Bit
SMIO_29	gpio_pull_sel[84]
SMIO_30	gpio_pull_sel[85]
SMIO_31	gpio_pull_sel[86]
SMIO_32	gpio_pull_sel[87]
SMIO_33	gpio_pull_sel[88]
HPD	gpio_pull_sel[89]
CEC	gpio_pull_sel[90]
SVIDEO	gpio_pull_sel[91]
SHSYNC	gpio_pull_sel[92]
VDO_OUT_0	gpio_pull_sel[93]
VDO_OUT_1	gpio_pull_sel[94]
VDO_OUT_2	gpio_pull_sel[95]

Table 19-32. GPIO Pin Pull-Up / Pull-Down Direction Select 2 Register and Pin Name Detail.

19.2.2.25 GPIO_pull_sel_3 Register

Bits	Name	Attr	Reset	Description
31:18				Reserved
17:0	gpio_pull_sel[113:96]	RW	0x0	GPIO pin pull value selection when the corresponding pull-enable bit is set. 0: Pull Down 1: Pull High

Table 19-33. GPIO Pin Pull-Up / Pull-Down Direction Select 3 Register.

GPIO Pin Name	Bit
VDO_OUT_3	gpio_pull_sel[96]
VDO_OUT_4	gpio_pull_sel[97]
VDO_OUT_5	gpio_pull_sel[98]
VDO_OUT_6	gpio_pull_sel[99]
VDO_OUT_7	gpio_pull_sel[100]
VDO_OUT_8	gpio_pull_sel[101]
VDO_OUT_9	gpio_pull_sel[102]
VDO_OUT_10	gpio_pull_sel[103]
VDO_OUT_11	gpio_pull_sel[104]
VDO_OUT_12	gpio_pull_sel[105]
VDO_OUT_13	gpio_pull_sel[106]
VDO_OUT_14	gpio_pull_sel[107]
VDO_OUT_15	gpio_pull_sel[108]
VDO_CLK	gpio_pull_sel[109]
VDO_VSYNC	gpio_pull_sel[110]
VDO_HSYNC	gpio_pull_sel[111]
VDO_HVLD	gpio_pull_sel[112]

GPIO Pin Name	Bit
VD_PWM	gpio_pull_sel[113]

Table 19-34. GPIO Pin Pull-Up / Pull-Down Direction Select 3 Register and Pin Name Detail.

19.2.2.26 PWC_dnalert Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	dnalert	R	0	PWC is forced off by pressing PWC_WKUP longer than 16 s.

Table 19-35. PWC Alert Register.

19.2.2.27 PWC_lbat Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	lbat	R	0	The battery level is low when this bit is asserted

Table 19-36. PWC Low Battery Signal Register.

19.2.2.28 PWC_reg_sta Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	sta	R	0	Current value of status bits

Table 19-37. PWC Status Read Register.

19.2.2.29 PWC_sta Register

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	reg_sta	RW	0	Sets the status bits

Table 19-38. PWC Status Set Register.

19.2.2.30 PWC_enp[n]_enable Registers

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	enp[n]	RW	0	Ties PWC_PSEQ[N] high when asserted

Table 19-39. Registers to Enable Each PWC Sequence[n].

The **PWC_enp[n]** and **PWC_disp[n]** registers control the PWC outputs **PWC_PSEQ[N]** as follows:

- The **PWC_PSEQ[N]** pins are asserted high if power sequence[n] is ready when **enp[n] = disp[n]**.
- If **enp[n]** is 1 and **disp[n]** is 0, **PWC_PSEQ[N]** is always 1.
- If **enp[n]** is 0 and **disp[n]** is 1, **PWC_PSEQ[N]** is always 0.

19.2.2.31 **PWC_disp[n]**_disable Registers

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	disp[n]	RW	0	Ties PWC_PSEQ[N] low when asserted

Table 19-40. PWC Sequence[n] Disable Registers.

19.2.2.32 **ADC_slot_ctrl_[n]** Registers

Bits	Name	Attr	Reset	Description
31:4				Reserved
3:0	slot_channel_x	RW	0	Selection of sampling channels of slot x, one hot encoding. For example, 0x00F indicates sampling channels from #0 to #3. Do not set to zero when the slot is activated.

Table 19-41. ADC Slot Control Register.

19.2.2.33 **ADC_int_ctrl_[n]** Registers

Bits	Name	Attr	Reset	Description
31	int_en_ch_[n]	RW	0	Enable threshold interrupt trigger
30:28				Reserved
27:16	hi_int_ch_[n]	RW	0	High threshold value for ADC channel N
15:12				Reserved
11:0	low_int_ch_[n]	RW	0	Low threshold value for ADC channel N

Table 19-42. ADC Interrupt Control Register.

19.2.2.34 **ADC_data_[n]** Registers

Bits	Name	Attr	Reset	Description
31:12				Reserved
11:0	adc_data_[n]	R	0	ADC converted data channel N

Table 19-43. ADC Data Registers.

19.2.2.35 ADC_fifo_ctrl_0/1/2/3 Registers

Bits	Name	Attr	Reset	Description
31	fifo_over_int_en_x	RW	0	Enables x-th FIFO overflow interrupt
30	fifo_undr_int_en_x	RW	0	Enables x-th FIFO underflow interrupt
29:17				Reserved
26:16	fifo_th_x	RW	0	FIFO data interrupt threshold, INT while fifo_cnt_x > fifo_th_x
15:12	fifo_cid_x	RW	0	Channel number associated with x-th FIFO
11				Reserved
10:0	fifo_depth_x	RW	0	FIFO depth for ADC samples. Each sample is 12 bits. Value set when fifo_clear . Sum of four values shall not exceed 1024 .

Table 19-44. ADC FIFO Control 0/1/2/3 Register.
19.2.2.36 ADC_fifo_ctrl Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	fifo_clear	RW	0	Clear and set FIFOs when initializing or on occurrence of error. Self-clearing.

Table 19-45. ADC FIFO Control Register.
19.2.2.37 ADC_fifo_status_0/1/2/3 Registers

Bits	Name	Attr	Reset	Description
31:17				Reserved
16	fifo_active_0	R	0	0 - FIFO not activated 1 - FIFO activated
15:11				Reserved
10:0	fifo_cnt_x	R	0	FIFO data counter

Table 19-46. ADC FIFO Status 0/1/2/3 Register.
19.2.2.38 ADC_ec_control Register

Bits	Name	Attr	Reset	Description
31:2				Reserved
1	vcm_option_1	RW	0	VCM counting rule options (see Figure 21-1 below) 0 - Counting rule uses option 1 1 - Counting ruler uses option 2

Bits	Name	Attr	Reset	Description
0	vcm_enable	RW	0	Enables VCM event counting. This value is active after parameter register are set.

Table 19-47. ADC Event Counter Control Register.

19.2.2.39 ADC_ec_counter Register

Bits	Name	Attr	Reset	Description
31:17				Reserved
16	vcm_ec_set	RW	0	Set initial value to event counter Self-clearing
15:0	vcm_ec_counter	RW	0	Event counter value (-32768 to 32767)

Table 19-48. ADC Event Counter General Register.

19.2.2.40 ADC_ec_counter_th Register

Bits	Name	Attr	Reset	Description
31:15				Reserved
16	vcm_int_en	R/W	0	Interrupt enable control
15:0	vcm_int_th	R/W	0	Interrupt threshold of event counter.

Table 19-49. ADC Event Counter TH Register.

19.2.2.41 ADC_ec_adc Register

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:12	vcm_ref_cid_b	RW	0	ADC channel ASEL value of reference of B. Value 0xF indicates reference set through internal register.
11:8	vcm_ref_cid_a	RW	0	ADC channel ASEL value of reference of A. Value 0xF indicates reference set through internal register.
7:4	vcm_adc_cid_b	RW	0	ADC channel ASEL value of signal B (0 - 3)
3:0	vcm_adc_cid_a	RW	0	ADC channel ASEL value of signal A (0 - 3)

Table 19-50. ADC Event Counter ADC Register.

19.2.2.42 ADC_ec_refval Register

Bits	Name	Attr	Reset	Description
31:28				Reserved
27:16	vcm_ref_b	RW	0	Reference to ADC value of signal B in VCM (0 - 4095)
15:12				Reserved

Bits	Name	Attr	Reset	Description
11:0	vcm_ref_a	RW	0	Reference to ADC value of signal A in VCM (0 - 4095)

Table 19-51. ADC Event Counter Reference Value Register.

19.2.2.43 ADC_ec_reshape Register

Bits	Name	Attr	Reset	Description
31:29				Reserved
28:16	vcm_prime_hi_th	RW	0	High threshold value for waveform reshaper (Schmitt trigger). Values from -4096 to 4095.
15:13				Reserved
12:0	vcm_prime_low_th	RW	0	Low threshold for waveform reshaper (Schmitt trigger). Values from -4096 to 4095.

Table 19-52. ADC Event Counter Reshape Register.

19.2.2.44 ADC_fifo_data_0/1/2/3 Register

The A12 chip provides four 4-KB blocks to support FIFO 0/1/2/3 burst read operations.

Bits	Name	Attr	Reset	Description
1016:0	fifo_data_block_0	R	0	4-KB data FIFO for burst read

Table 19-53. ADC FIFO Data 0/1/2/3 Register.

19.3 ADC / PWC / RTC: ADC Programming Notes

This section provides programming notes related to the ADC interface. The section is organized as follows:

- [\(Section 19.3.1\) ADC Initialization Sequence](#)
- [\(Section 19.3.2\) ADC Clocking](#)
- [\(Section 19.3.3\) ADC Time Slot Settings](#)

19.3.1 ADC Initialization Sequence

1. Clear the ADC by writing the **ADC_control** register bit 0 to 1.
2. Set the ADC mode using the **ADC_control** register field **adc_mode**.
3. Enable the ADC by writing the **ADC_control** register bit 2 to 1.
4. Wait 3 microseconds (uS) for the ADC to stabilize.
5. Set ADC parameters with the following registers:

- ADC_counter
- ADC_slot_num
- ADC_slot_period
- ADC_ctrl_int_table
- ADC_data_int_table
- ADC_fifo_int_table
- ADC_slot_ctrl_[n]
- ADC_int_ctrl_[n]
- ADC_data_[n]
- ADC_fifo_ctrl_[n]
- ADC_fifo_ctrl
- ADC_fifo_status_[n]
- ADC_ec_control
- ADC_ec_counter
- ADC_ec_counter_th
- ADC_ec_adc
- ADC_ec_refval
- ADC_ec_reshape

6. Use the **ADC_control** register **adc_start** bit to trigger the sampler.
7. Wait for the **ADC_status** register **status** bit to become 1.
8. Read the Data Registers.
9. Wait for the **ADC_status** register **sampling_done** bit to become 1 at the conclusion of ADC sampling.

19.3.2 ADC Clocking

The ADC clock GCLK_ADC is generally configured by A12 system software. The register programming information provided in this chapter is used for adjustment and refinement.

19.3.3 ADC Time Slot Settings

Each ADC time slot can be used to enable channel sampling independently. The number of valid time slots can be selected as well.

For example, if the targets are:

- Channels 10/11 run at 200 KS/s for the hall sensor
- Channels 8/9 run at 100 KS/s for the gyro sensor
- Other channels run at 25 KS/s

Then the ADC clock can be set to 4 MHz for up to 800 KS/s, and registers should be programmed for:

- `slot_num = 7`
- `channel_sel_0 = 0xF00` (enable channel 8, 9, 10, 11)
- `channel_sel_1 = 0xC03` (enable channel 0, 1, 10, 11)
- `channel_sel_2 = 0xF00` (enable channel 8, 9, 10, 11)
- `channel_sel_3 = 0xC0C` (enable channel 2, 3, 10, 11)
- `channel_sel_4 = 0xF00` (enable channel 8, 9, 10, 11)
- `channel_sel_5 = 0xC30` (enable channel 4, 5, 10, 11)
- `channel_sel_6 = 0xF00` (enable channel 8, 9, 10, 11)
- `channel_sel_7 = 0xCC0` (enable channel 6, 7, 10, 11)

Note that the A12 chip supports up to four ADC channels only. The above example is provided for illustrative purposes.

19.4 ADC / PWC / RTC: PWC and RTC Programming Notes

This section provides programming notes related to the PWC and RTC interfaces. The section is organized as follows::

- [\(Section 19.4.1\) PWC Port Definitions](#)
- [\(Section 19.4.2\) PWC and RTC Power-Up/Down](#)

19.4.1 PWC Port Definitions

Port Name	Input/Output	Description
DVDD	Input	1-V core power; for PWC interface only
PC_VDD	Input	3.3-V analog power Connected to external battery/adaptor clamping circuit. When PC_VDD is less than 2.3 V, all wake-ups will be blocked.
PC_VSS	Input	Analog ground
RTC_CP	Input	2.6-V power for on-chip RTC oscillator When RTC_CP is less than 1.4 V, the power controller will be powered down and all registers will be reset.
AVDD33	Input	3.3-V analog power for the output sequences
PC_REF	Input	Maximum is 1.8 V. If less than 1.2 V, then REG_LBAT is issued to core.
PD	Input	Power down (from core); pulled down In power-on state, a positive pulse of PD will trigger power-off sequences that consist of de-asserting sequence-3 state, followed by de-asserting sequence-2 state, followed by de-asserting sequence-1 state, followed by the power-off state. The core will be reset when this trigger is asserted.
RSTINB	Input	Vilmax=1.2 V, Vihmin=2.4 V; Low effective to reset power controller logic. The current time registers will not be affected by this reset.
PWC_WKUP	Input	Maximum is 3.3 V, Vih, min=1.7 V; Connected to external power switch to turn on the power-on state. Normally low.
PWC_WKUP[3:1]	Input	Same as PWC_WKUP.
WKUPC[15:0]	Input	From GPIO, to wake up system when WKENC=1
CLEARC	Input	From core, to clear REG_WKUPC[15:0] when asserted high
WKENC[15:0]	Input	From core, enable monitoring GPIO with WKUPC[15:0] to wake-up power sequences when asserted high
IO1P8V[15:0]	Input	From core, default 0, which means GPIO output is 3.3 V. Assert to high when GPIO is 1.8 V.
PCRST	Input	From core, the positive pulse of this signal is used to set values of registers,
SEQ1[7:0]	Input	From core, used to set delay-time of power-on sequence 1
SEQ2[7:0]	Input	From core, used to set delay-time of power-on sequence 2
SEQ3[7:0]	Input	From core, used to set delay-time of power-on sequence 3
SEQ4[7:0]	Input	From core, used to set delay-time of RSTOB
ALAT[31:0]	Input	From core, used to set alarm time
CURT[31:0]	Input	From core, used to set current time

Port Name	Input/Output	Description
Sta[7:0]	Input	From core, used to set the value of reg_sta[7:0] at the positive pulse of PCRST
Bc[2:0]	Input	From core, to set bias current in oscillator
XI_RTC	I/O	Connected to 1 pin of crystal
VBG	Output	Bandgap output voltage
ENP1C	Input	From core: to set PSEQ1C to high
DISP1C	Input	From core: to set PSEQ1C to low
ENP2C	Input	From core: to set PSEQ2C to high
DISP2C	Input	From core: to set PSEQ2C to low
ENP3C	Input	From core: to set PSEQ3C to high
DISP3C	Input	From core: to set PSEQ3C to low
ENP1	Input	From core: to set PSEQ1 to high
DISP1	Input	From core: to set PSEQ1 to low
ENP2	Input	From core: to set PSEQ2 to high
DISP2	Input	From core: to set PSEQ2 to low
ENP3	Input	From core: to set PSEQ3 to high
DISP3	Input	From core: to set PSEQ3 to low
PSEQ1	Output	To pad, asserted high if power sequence1 is ready when ENP1=DISP1; Always high when ENP1=1 and DISP1=0; Always low when ENP1=0 and DISP1=1
PSEQ2	Output	To pad, asserted high if power sequence2 is ready when ENP2=DISP2; Always high when ENP2=1 and DISP2=0; Always low when ENP2=0 and DISP2=1
PSEQ3	Output	To pad, asserted high if power sequence3 is ready when ENP3=DISP3; Always high when ENP3=1 and DISP3=0; Always low when ENP3=0 and DISP3=1
PSEQ1C	Output	To pad, asserted high if power sequence1 is ready when ENP1C=DISP1C; Always high when ENP1C=1 and DISP1C=0; Always low when ENP1C=0 and DISP1C=1
PSEQ2C	Output	To pad, asserted high if power sequence2 is ready when ENP2C=DISP2C; Always high when ENP2C=1 and DISP2C=0; Always low when ENP2C=0 and DISP2C=1
PSEQ3C	Output	To pad, asserted high if power sequence3 is ready when ENP3C=DISP3C; Always high when ENP3C=1 and DISP3C=0; Always low when ENP3C=0 and DISP3C=1
RSTOB	Output	To pad, reset external circuits in power-off state
DNALERT	Output	To core, asserted high when both PWC_WKUP and PWC_WKUP[3] are pressed high for longer than 4s
REG_ALAWK	Output	To core, set for 1 s when alarm time equals current time
REG_LBAT	Output	To core, set to high whenever PC_VDDP drops below 1.2/(PC_REF/PC_VDD) V
REG_WKUP	Output	To core, stores status of PWC_WKUP
REG_WKUPC[15:0]	Output	To core, stores status of WKUPC[15:0]
REG_WO[2:0]	Output	To core, stores status of PWC_WKUP[3:1]
REG_CURT[31:0]	Output	To core, values of current time
REG_ALAT[31:0]	Output	To core, values of alarm time
REG_STA[7:0]	Output	To core, to store the power sequence information for software.
CLK_RTC	Output	To core, Clock, frequency = 32.768 K

Port Name	Input/Output	Description
RTC_CLK1S	Output	To core, Clock, frequency=1
XO_RTC	I/O	Connect to another pin of crystal

Table 19-54. PWC Port Definitions.

 Note:

A12m SoCs include a single wake-up pin only.

19.4.2 PWC and RTC Power-Up/Down

The PWC can be powered on by: (1) **PWC_WKUP/1/2/3** signals, (2) alarm, or (3) GPIO input signals. All methods require **PWC_PC_VDD** to reach a specified minimum voltage (typically, greater than 1.5 V). Please refer to the A12 datasheet for electrical requirements.

- [\(Section 19.4.2.1\) PWC Power-on: PWC_WKUP](#)
- [\(Section 19.4.2.2\) PWC Power-on: Alarm](#)
- [\(Section 19.4.2.3\) PWC Power-on: GPIO](#)
- [\(Section 19.4.2.4\) PWC Power Down](#)
- [\(Section 19.4.2.5\) PWC: Power Sequence Control](#)
- [\(Section 19.4.2.6\) PWC: Load Registers](#)
- [\(Section 19.4.2.7\) PWC: Reset Power Sequences and Registers](#)

19.4.2.1 PWC Power-on: PWC_WKUP

- In POWER-OFF mode, the positive edge of **PWC_WKUP/1/2/3** triggers the PWC to enter POWER-ON mode.
- The **enp1/2/3** and **disp1/2/3** signals are the enable and disable signals for the corresponding power sequence **PWC_PSEQ1/2/3**.
 - If the **enp1/2/3** and **disp1/2/3** values both are 0 or 1, **PWC_WKUP/1/2/3** controls the **PWC_PSEQ1/2/3** sequences to power-on.
 - **PWC_RSTOB** always powers on when entering POWER-ON mode.
- The sequences power-on in an order of **PWC_PSEQ1**, **PWC_PSEQ2**, **PWC_PSEQ3**, then **PWC_RSTOB**.
- Use the following registers for time intervals:
 - PWC Sequence 1 (**PWC_seq1**, 0x20) to set the delays (mS) between **PWC_PSEQ1** and **PWC_WKUP**.
 - PWC Sequence 2 (**PWC_seq2**, 0x24) to set the delays (mS) between **PWC_PSEQ1** and **PWC_PSEQ2**.
 - PWC Sequence 3 (**PWC_seq3**, 0x28) to set the delays (mS) between **PWC_PSEQ2** and **PWC_PSEQ3**.

- PWC Sequence 4 (**PWC_seq4**, 0xD0) to set the delays (mS) between **PWC_PSEQ3** and **PWC_RSTOB**.
- Repetitious assertion of **PWC_WKUP/1/2/3** in POWER-ON mode is ignored.
- When the PWC successfully enters POWER-ON mode, the **PWC_WKUP** and **PWC_WKUP1/2/3** signals, respectively, are stored in register fields **wkup** (register **PWC_sta**, at 0xC0, bit 3) and **pwc_reg_wo** (register **PWC_reg_wo**, at 0x7C, bits 0:2).
- Initial states of **PWC_WKUP** and **PWC_WKUP[3]** should be low, and initial states of **PWC_WKUP1** and **PWC_WKUP2** can be either low or high.

19.4.2.2 PWC Power-on: Alarm

- The registers **PWC_curt_read/write** (0x34/0x30, bits 0:31) and **PWC_alat_read/write** (0x2C/0x38, bits 0:31) store/set the current time and alarm time, respectively.
- **PWC_sta** (0xC0, bit 3) enables the alarm wake-up function.
- When **PWC_curt_read** equals **PWC_alat_read**, the **RTC_status** register **ala_wk** signal (0x3C, bit 2) is asserted high.
- If the **PWC_sta** register **wkup** bit (0xC0, bit 3) is set high, the power sequences are generated in the order mentioned above.

19.4.2.3 PWC Power-on: GPIO

- The WKENC[15:0] bits enable power-on via GPIO.
- When WKENC[i] is asserted high by the positive pulse of PCRST, the power sequences can be powered on by the corresponding GPIO-monitored signal WKUPC[i]. In this case, WKUPC[i] has the same function as **PWC_WKUP/1/2/3**.
- REG_WKUPC[i] is asserted high at the positive edge of WKUPC[i], and is cleared when the battery is low or when the CLEARC or PD signal is issued. Note that WKUPC[i] should be set back to low after power-on.
- If GPIO[i] voltage is 1.8 V, then the corresponding IO1P8V[i] should be set to high. If GPIO[i] voltage is greater than 2.8 V, then the corresponding IO1P8V[i] should be set to low. The default value for IO1P8V[15:0] is 0.

19.4.2.4 PWC Power Down

- In POWER-ON mode, a positive edge of the power-down PD signal triggers the PWC to enter POWER-OFF mode.
- If the **enp1/2/3** value is zero and the **disp1/2/3** value is one, the power sequences **PWC_PSEQ1/2/3** will be pulled low instantly.
- The power sequences **PWC_PSEQ1/2/3** are pulled down in this order: (1) **PWC_RSTOB**, (2) **PWC_PSEQ3**, (3) **PWC_PSEQ2**, and (4) **PWC_PSEQ1**.

- PWC Sequence 4 (**PWC_seq4**, 0xD0) sets the delay interval between power-down signal PD and **PWC_RSTOB**.
- PWC Sequence 3 (**PWC_seq3**, 0x28) sets the delay interval between **PWC_RSTOB** and **PWC_PSEQ3**.
- PWC Sequence 2 (**PWC_seq2**, 0x24) sets the delay interval between **PWC_PSEQ3** and **PWC_PSEQ2**.
- PWC Sequence 1 (**PWC_seq1**, 0x20) sets the delay interval between **PWC_PSEQ2** and **PWC_PSEQ1**.
- A PD signal issued while the sequences are powering-off is ignored.

19.4.2.5 PWC: Power Sequence Control

- In addition to control by external pin **PWC_WKUP/1/2/3** and internal power-down PD signals, the **PWC_PSEQ1/2/3** pins can be forced to high or low independently by setting corresponding registers {**enp1/2/3**, **disp1/2/3**} = 2'b10 or 2'b01.
- **PWC_sta** (0xC0, bit 2) enables DRAM self-refreshing mode. If **PWC_sta** (0xC0, bit 2) is asserted high and the joint decoding of register fields {**enp3**, **disp3**}≠2'b01, **PWC_PSEQ3** will stay high .

19.4.2.6 PWC: Load Registers

- A positive PCRST pulse causes the PWC to load the values of **PWC_seq1/2/3/4** (bits 0:7), **PWC_sta** (bits 0:7), **PPWC_curt_read/write** (0x34/0x30, bits 0:31) and **PWC_alat_read/write** (0x2C/0x38, bits 0:31) to corresponding registers. Ensure that all register data is ready when programming specific registers.
- The data must be valid on these inputs at least 3 ms before the assertion of PCRST.
- Both the data and PCRST signals should be maintained for at least 4 ms.
- At least one programming task is required after **PWC_RTC_CP** is on or recovered above the minimum voltage (see the A12 chip datasheet for electrical details).
- After programming is completed, the PWC controller should reset both inputs and PCRST to 0.
- The default value for **PWC_seq1/2/3/4** (bits 0:7) is 32. The default value for all other registers is 0.

19.4.2.7 PWC: Reset Power Sequences and Registers

- When external pin **PWC_WKUP** is held high for longer than 16 s, all registers will be reset and all power sequences will be pulled low.
- When the **PWC_WKUP** and **PWC_WKUP3** signals are simultaneously held high for longer than 4 s, the DNALERT signal will be generated (refer to the **PWC_dnalert** register described in [Section 22.2.2.15](#) for additional detail).
- If **PWC_WKUP** and **PWC_WKUP3** are simultaneously held high for longer than 8 s, all power sequences will be pulled low and the **PWC_reg_sta** register bits 6:7 will be reset to low.
- If **PWC_WKUP1/2/3** are not used, tie **PWC_WKUP3** to **PWC_WKUP**, and tie **PWC_WKUP1/2** to

ground.

- Refer to the A12 chip datasheet for electrical characteristics and conditions.

For Confidential
HAOTEK Only

20. INFRARED (IR) REMOTE INTERFACE

This chapter provides register programming information for the A12 InfraRed Remote interface as follows:

- [\(Section 20.1\) IR: Overview](#)
- [\(Section 20.2\) IR: Remote Receiver Modules](#)
- [\(Section 20.3\) IR: Clocking](#)
- [\(Section 20.4\) IR: Registers](#)

20.1 IR: Overview

The A12 chip provides an interface that manages digital signals from InfraRed (IR) remote controllers.

There are a variety of IR remote control protocols used by different consumer electronics manufacturers, most of which are variants of the RC-5 and REC-80 standards. Under these standards, a remote command consists of a fixed-length binary code word. Each bit of a code word is encoded using either bi-phase coding (RC-5) or pulse-distance coding (REC-80).

The various protocols differ as to the number of bits and the sub-fields within each code word (“start bit” field, “address” field, “command” field, and so on). In bi-phase coding, a logical 0 is encoded as a high-to-low transition, and a logical 1 is encoded as a low-to-high transition. In pulse-distance coding, a logical 0 is encoded as a high level of duration T followed by a low level of duration 2T. A logical 1 is encoded as a high level of duration T followed by a low level of duration 3T. From there, the bi-phase or pulse-distance encoded bits are then used to amplitude-modulate a carrier. The carrier frequency is typically in the range of 30 kHz to 50 kHz. The modulated signal is then transmitted using an IR LED.

The table below summarizes several popular protocols:

Protocol	Carrier Frequency (kHz)	Bit Encoding	Payload Burst Length (bits)	Inter-Burst Gap (ms)
NEC	38	Pulse-Distance	32	110
Sharp	38	Pulse-Distance	13	40
Sony	40	Pulse-Distance	12	25
Philips RC5	36	Bi-Phase	11	114

Table 20-1. Popular InfraRed Remote Protocols.

Note that a protocol not mentioned above is the IIT protocol which does not use a carrier but can be categorized as a form of pulse-distance coding. In this case, the T duration is shorter: 0.1 ms for logic 0 and 0.2 ms for logic 1. A typical burst bit rate is approximately 1 Kbps.

20.2 IR: Remote Receiver Modules

Integrated IR remote receiver modules are offered by a number of manufacturers (e.g., Vishay, Rohm and NEC). These receivers typically consist of a photodiode, AGC, bandpass filter, and demodulator. The output of these receivers is the bi-phase or pulse-distance bit stream.

In many consumer electronics designs (e.g., DVD players), the IR receiver is connected to a GPIO pin of a microcontroller and the decoding is performed via software. A degree of hardware-assist is used to reduce the frequency of software wake-ups required in response to a state change of the GPIO pin.

The external interface to the IR module consists of the A12 input pin (**IR_IN**). This pin can be configured for GPIO usage, if desired. Internally, the IR interface consists of a 16-bit counter, driven at a rate of **CLK_IR** from the RCT module. At the detection of a rising edge or a falling edge of the external input pin, the value of the counter is latched and put into a FIFO. The counter is then reset and continues to count until the next edge is detected (i.e., the data in the FIFO represents the number of clock cycles between signal edges).

When the internal 16-bit counter reaches 0xFFFF, it saturates until being reset upon the next signal edge. Software can determine the beginning of a burst by comparing the clock cycle count against the inter-burst gap of the particular protocol in use.

20.3 IR: Clocking

The remote IR clock **GCLK_IR** is generally configured by A12 system software. The register programming information provided in this chapter is used for adjustment and refinement.

20.4 IR: Registers

20.4.1 IR Registers: Map

There are three APB registers which control the IR interface. The registers are located at base address 0xE800.6000 on the APB bus.

Register Offset	Register Name	Description
0	IR_control	Control
4	IR_status	Status
8	IR_data	Read port of data FIFO

Table 20-2. APB registers of IR Interface Module.

20.4.2 IR Registers: Detail

20.4.2.1 IR_control Register

Bits	Name	Attr	Reset	Description
31:15				Reserved
14	reset	W		1 - Reset the entire IR module (all other bits go to reset state)
13	enb	RW	0	0 - IR interface disabled 1 - IR interface enabled
12	levint	RW	0	Level interrupt flag 0 - No level interrupt 1 - Level interrupt Software writes 1 to clear this bit
11:10				Reserved
9:4	intlev	RW	0	Interrupt level When post-increment FIFO fullness is \geq (intlev + 1), generate interrupt to ARM
3	fifo_ov	RW	0	FIFO overflow 0 - No overflow 1 - Overflow occurred Software writes 1 to clear this bit
2	intenb	RW	0	Interrupt enable 0 - Disabled 1 - Enabled
1:0				Reserved

Table 20-3. IR Remote Control Register.

20.4.2.2 IR_status Register

Bits	Name	Attr	Reset	Description
31:6				Reserved
5:0	count	R	0	FIFO fullness

Table 20-4. IR Remote Status Register.

20.4.2.3 IR_data Register

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:0	data	R	NR	Number of clock cycles since last edge

Table 20-5. IR Remote Data Register.

21. STEPPER, MICRO-STEPPER AND PWM

This chapter provides register programming information for the A12 Stepper Motor Controller, which includes Stepper (ST), Pulse Width Modulator (PWMB0/B1/C0/C1), and Micro-Stepper (MS) interfaces. The chapter is organized as follows.

- ([Section 21.1](#)) ST / MS / PWM: Overview
- ([Section 21.2](#)) ST / MS / PWM: Block Diagram
- ([Section 21.3](#)) ST / MS / PWM: External Pin Muxing
- ([Section 21.4](#)) ST / MS / PWM: Clocking
- ([Section 21.5](#)) ST / MS / PWM: Stepper Interface
- ([Section 21.6](#)) ST / MS / PWM: PWMB0/B1/C0/C1 Interface
- ([Section 21.7](#)) ST / MS / PWM: Micro Stepper Interface
- ([Section 21.8](#)) ST / MS / PWM: Registers

 Note that the A12m SoC does not support motor control.

21.1 ST / MS / PWM: Overview

The A12 stepper motor interface is designed to assist the ARM host processor in controlling stepper/DC motors often used in digital still camera (DSC) lens modules. Features of the stepper motor controller include:

- Five sets of Stepper (ST) interfaces (A, B, C, D, and E).
 - Interfaces A through D consist of four external pins each. The four 4-pin sets are: **SC_A[0:3]**, **SC_B[0:3]**, **SC_C[0:3]**, and **SC_D[0:3]**.
 - Interface E consists of one pin: **SC_E0**.
 - Steppers can be advanced forwards or backwards.
 - Note that pins from multiple stepper channels should not be used to connect to the same motor driver channel.
- Four sets of Micro-Stepper (MS) interfaces (A, B, C, and D).
 - Each set consists of four external pins. The four 4-pin sets are: **SC_A[0:3]**, **SC_B[0:3]**, **SC_C[0:3]**, and **SC_D[0:3]**.
 - Configurable SIN wave table.
 - Each phase has 128 micro-steps.
 - Micro Steppers are grouped together for micro-stepper motor control. For example:
 - **SC_A0** connects to Motor_A
 - **SC_A1** connects to Motor_A#
 - **SC_A2** connects to Motor_B
 - **SC_A3** connects to Motor_B#
 - In this example, if Motor_A and Motor_A# are connected to stepper channel A, the remaining micro-stepper channels should not be used for Motor_B and Motor_B#.

- Four Pulse Width Modulation (PWMB0/B1/C0/C1) interfaces. The four PWM outputs are referred to as **PWM_[0:3]** in the A12 chip datasheet. Functionally:
 - PWMB0 is associated with PWM_0
 - PWMB1 is associated with PWM_1
 - PWMC0 is associated with PWM_2
 - PWMC1 is associated with PWM_3
- Note that in addition to the PWM controller embedded in the stepper motor controller, the A12 external pin **VD_PWM** can also serve as a PWM controller.

21.2 ST / MS / PWM: Block Diagram

The Stepper Motor Controller provides interfaces for the Stepper Motor ([Section 21.5](#)), the Micro Stepper ([Section 21.7](#)) and general-purpose Pulse Width Modulators ([Section 21.6](#)). Below is a block diagram showing the Stepper Motor Controller and its associated interfaces.

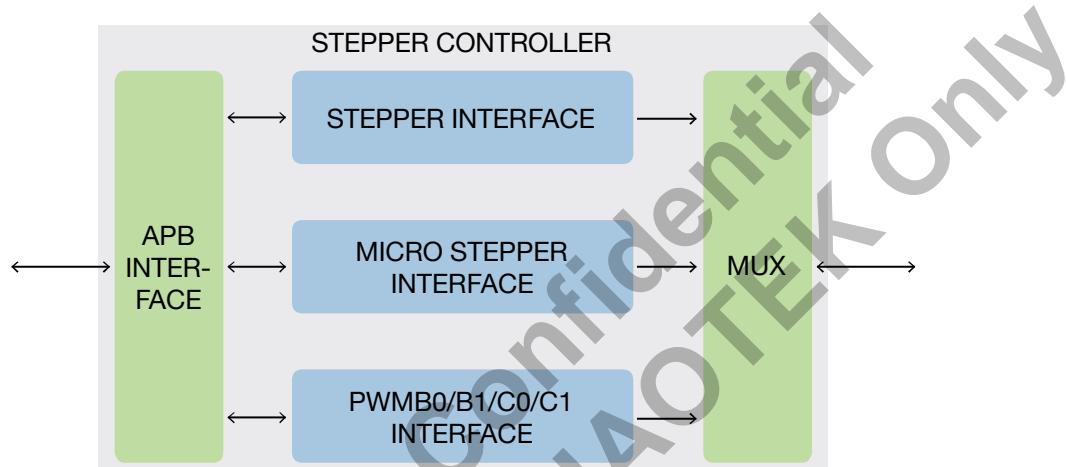


Figure 21-1. Block Diagram of the Stepper Motor Controller.

21.3 ST / MS / PWM: External Pin Muxing

The Stepper, Micro Stepper and PWM functions are multiplexed on GPIO pins. Refer to the GPIO programming chapter in this manual for information regarding function selection when using GPIO pins.

21.4 ST / MS / PWM: Clocking

The Stepper, Micro Stepper and PWM clock frequencies are generally set by system software. The register programming information provided in this chapter can be used for adjustment and refinement. Contact an Ambarella representative for additional detail.

- [Section 21.5.2](#) provides register programming information for the Stepper clocks.
- [Section 21.6.2](#) provides register programming information for the PWMB0/B1/C0/C1 clocks.
- [Section 21.6.2](#) provides an example of PWM clock tuning.

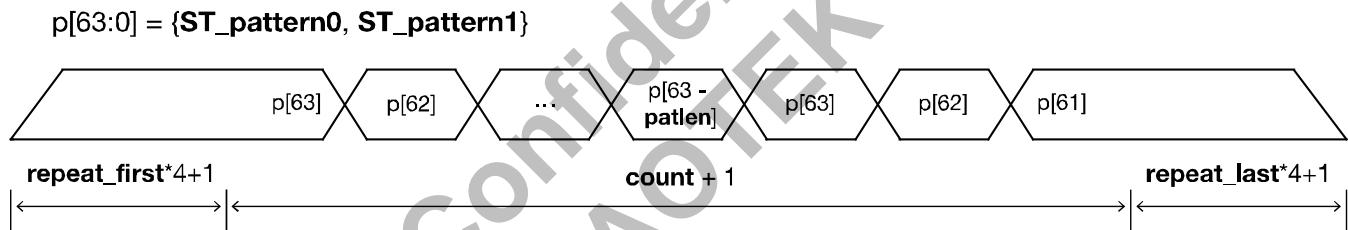
21.5 ST / MS / PWM: Stepper Interface

Register programming information for the Stepper Interface is provided in this section. The section is organized as shown below.

- ([Section 21.5.1](#)) Stepper: Overview
- ([Section 21.5.2](#)) Stepper: Clocking Overview
- ([Section 21.5.3](#)) Stepper: Clocking Accelerate / Decelerate
- ([Section 21.5.4](#)) Stepper: Interrupt

21.5.1 Stepper: Overview

Each **ST_pattern0/1_nx** register pair defines the bit sequence for each pin of each stepper interface. The maximum length of a pattern is 64 bits. For Stepper Interface A, a sequence of **ST_pattern0_0a** followed by **ST_pattern1_0a** defines the bit sequence of **SC_A0** and so on. Each stepper interface maintains an internal six-bit counter, which is driven by the phase clock of each interface. The counter counts from 0 through the value of the **ST_control_x** pattern length bit (**patlen**) and wraps around to 0 again. The counter provides the address for output to the external pin. For example, when the counter value is 0, the left-most bit of the **ST_pattern0_nx** register is used.



*Figure 21-2. Stepper Timing Diagram with 64-bit MSB (**ST_pattern0_nx**) LSB (**ST_pattern1_nx**) Pattern $p[64:0]$.*

The software initiates a new output sequence by writing to the **ST_count_x** register. An output sequence is complete when the hardware has decremented the **count** bit (**ST_count_x**) to zero.

The first phase of the sequence is reached when the **count** bit (**ST_count_x**) has not been decremented. The last phase of the sequence is reached when the **count** bit has been decremented to 0. The first phase (and/or last phase) can be repeated if the **repeat_first** (or **repeat_last**) bit (**ST_count_x**) is non-zero. For example, if **repeat_first** equals 0, the first phase is output $(0^*4 + 1) = 1$ time (i.e., no repeat). If **repeat_first** equals 1, the first phase is output $(1^*4 + 1) = 5$ times. A similar property applies to **repeat_last**.

Note that it is possible to terminate an active sequence early by writing to **ST_count**. The previous sequence outputs 1 to 3 more phases before the new sequence output starts. Setting **rewind_en** to terminate early causes an error in the prior sequence. Setting **ST_count** to 0 stops the previous sequence early; i.e., without **repeat_last**. Inserting an early termination generates an interrupt. Use software to read **ST_pre_counter_x** to determine how many steps of the old sequence were executed after receiving the interrupt.

Note that the hardware maintains an internal pointer for each of the five sets of stepper interfaces. This internal pointer points to the bit to be output. This internal pointer is reset to point to bit 63 of the 64-bit pattern when the **ST_control_x** bit **reset** is written to 1. Otherwise this pointer maintains its value. If the software changes the **ST_control_x** bit **patlen**, a reset should also be written. If the pointer is not reset and **patlen** is changed, then the pointer could be out of range and cause unexpected behavior.

For example, setting the **ST_control_0** bit **patlen** to 3 will define a four-bit pattern covering bits 63, 62, 61, and 60 of the 64-bit pattern register. A write to **ST_count_0** (with **count** = 6, **repeat_first** = **repeat_last** = 0, and **rewind_en** = 0) produces an output bit sequence of b63, b62, b61, b60, b63, b62, and b61. After the output sequence ends, the internal bit pointer points to bit 60 and remains there. A subsequent write to **ST_count_0** (with **count** = 2, **repeat_first** = **repeat_last** = 0, and **rewind_en** = 0) then produces an output sequence of b60, b63, and b62. At the end of the second sequence, the internal bit pointer will point to bit 61 and remain there. Then, if **rewind_en** is asserted, the pointer will jump to b63. Again, a subsequent write to **ST_count_0** (with **count** = 2, **repeat_first** = **repeat_last** = 0, and **rewind_en** = 1) will produce an output sequence of b63, b60, and b61. At the end of this sequence, the internal bit pointer will point to bit 62 and remain there if the **rewind_en** value remains 1. The internal bit pointer will be reset to b63 when a write of 1 resets the **ST_control_0** register.

21.5.2 Stepper: Clocking Overview

The Stepper Interface module includes internal counters driven by the GCLK_MOTOR clock. The GCLK_MOTOR clock can be further divided by the **clkdiv** field of the **ST_control_x** register to produce the phase clock for each stepper interface:

$$\text{Phase Clock for Stepper Interface} = \text{GCLK_MOTOR} / [2^{*(\text{clkdiv} + 1)}]$$

- In the equation, GCLK_MOTOR is the current clock frequency.
- Set the stepper phase clock divider with **ST_control_x** bit 0 **clkdiv** ([Section 21.8.2.1](#))
- The stepper phase counter is **ST_count_x** bits 15:0 **count** ([Section 21.8.2.3](#)).
- Use the **ST_status_x** to check and query the phase clock for the interface ([Section 21.8.2.4](#))

As **clkdiv** is an integer with minimum value of 0, the maximum phase clock is 24 MHz / 2 = 12 MHz assuming the system clock CLK_REF (source) is 24 MHz.

21.5.3 Stepper: Clocking Accelerate / Decelerate

The Stepper Phase Clock for a given stepper may be accelerated or decelerated with register programming. The clock divider is **clkdiv** when acceleration and / or deceleration are not enabled ([Section 21.5.2](#)) and otherwise is **acc_clkdiv0/1x** as described below.

- Set **ST_acc_count_x** bit **acc_type** to enable accelerate/decelerate functions ([Section 21.8.2.5](#)).
- The phase clock for a given acceleration or deceleration phase (acc[n]/dec[n]) is:

$$\text{Phase Clock for acc[n]/dec[n]} = \text{GCLK_MOTOR} / [2^{*(\text{acc_clkdiv}[n] + 1)}]$$

- GCLK_MOTOR frequency is the current stepper clock frequency.

- With acceleration / deceleration, the clock divider is determined using the 10-bit field `acc_clkdiv[n]` of the register `ST_acc_clkdiv_0x` for phase 0 and phase 1 or `acc0/dec0` and `acc1/dec1` ([Section 21.8.2.6](#)) and register `ST_acc_clkdiv_1x` for phase 2 and phase 3 or `acc2/dec2` and `acc3/dec3` ([Section 21.8.2.7](#))
- Phase number during acceleration / deceleration is set to `acc_count + 1` using `ST_acc_count_x` bit 9:0 `acc_count` ([Section 21.8.2.5](#)).

When performing acceleration and / or deceleration, the clock divider changes according to the current phase (step). Initially, the clock divider is `acc_clkdiv0`, then it transitions with the phase to `acc_clkdiv1/acc_clkdiv2/acc_clkdiv3`, keeping pace with the stepper (`acc_count + 1` phases).

After acceleration, the stepper uses the `clkdiv` clock divider and proceeds for $(\text{count}+1)-8*(\text{acc_count}+1)$ phases.

Finally during deceleration, the clock divider transitions from `acc_clkdiv3/acc_clkdiv2/acc_clkdiv1/ acc_clkdiv0`, and for each clock divider, the stepper proceeds (`acc_count+1`) phases.

The table below summarizes the range of outputs during acceleration phases (`acc[n]`), normal state, and deceleration phases (`dec[n]`). As the total phase count does not change, $(\text{count} + 1)$ either must be larger than $8*(\text{acc_count} + 1)$ when `acc_type = 0x3`, or must be larger than $4*(\text{acc_count} + 1)$ when `acc_type = 0x1` or `0x2`. If the accelerate function is enabled, the clock divider of `repeat_first` is `acc_clkdiv0/1x`, otherwise (else) it is `clkdiv`. If the decelerate function is enabled, the clock divider of `repeat_last` is `acc_clkdiv0` and otherwise it is `clkdiv`.

Phase	Phase Count Acc[n]/Dec[n] ¹	Programmed Phases ²	Clock Divider ³
Repeat First			
acc0	<code>acc_count + 1</code>		<code>acc_clkdiv0</code>
acc1	<code>acc_count + 1</code>		<code>acc_clkdiv0</code>
acc2	<code>acc_count + 1</code>		<code>acc_clkdiv1</code>
acc3	<code>acc_count + 1</code>		<code>acc_clkdiv2</code>
Repeat Normal			<code>acc_clkdiv3</code>
dec3	<code>acc_count + 1</code>	<code>count + 1</code>	<code>clkdiv</code>
dec2	<code>acc_count + 1</code>		<code>acc_clkdiv3</code>
dec1	<code>acc_count + 1</code>		<code>acc_clkdiv2</code>
dec0	<code>acc_count + 1</code>		<code>acc_clkdiv1</code>
Repeat Last			<code>acc_clkdiv0</code>

Table 21-1. Stepper Interface Enabling Accelerate / Decelerate Function.

Notes:

1. For register definitions see [Section 21.8.2.5](#).
2. For register definitions see [Section 21.8.2.3](#).
3. For register definitions see [Section 21.8.2.1](#), [Section 21.8.2.6](#) and [Section 21.8.2.7](#).

As an example, in the figure below both acceleration and deceleration are enabled (`acc_type` = 0x3) with `acc_count`=5 and `count`=65 and `ST_pattern0/1_nx` bit sequence fields `pattern` at {1,0}. The phase number for the four acceleration/deceleration states is 6 (or equivalently `acc_count` + 1), and the phase for a `ST_count_x` bits `repeat_first` or `repeat_last` state is 66-8*6=18 (or equivalently `(count + 1)-8*(acc_count + 1)`).

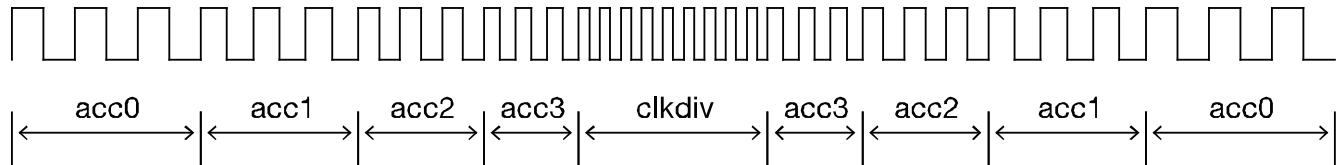


Figure 21-3. Stepper Acceleration / Deceleration Timing.

21.5.4 Stepper: Interrupt

After the end of the output phase (including `repeat_last`) or the last phase of the prior sequence which terminated early, an interrupt will be asserted, and the `ST_interrupt` register bit `interrupt_x` will go to high. The `interrupt_x` bit must be cleared by the software.

21.6 ST / MS / PWM: PWMB0/B1/C0/C1 Interface

This section provides programming information for the PWMB0/B1/C0/C1 interfaces. The section is organized as follows:

- [\(Section 21.6.1\) PWMB0/B1/C0/C1: Overview](#)
- [\(Section 21.6.2\) PWMB0/B1/C0/C1: Clocking](#)

21.6.1 PWMB0/B1/C0/C1: Overview

The PWMB0/B1/C0/C1 outputs (`pwm_[0:3]`) are shared across multiple pins. When enabled, a given PWM will put out a repeated sequence of logic-high and logic-low. The duration of each logic-high and logic-low is programmable (`xon` and `xoff`).

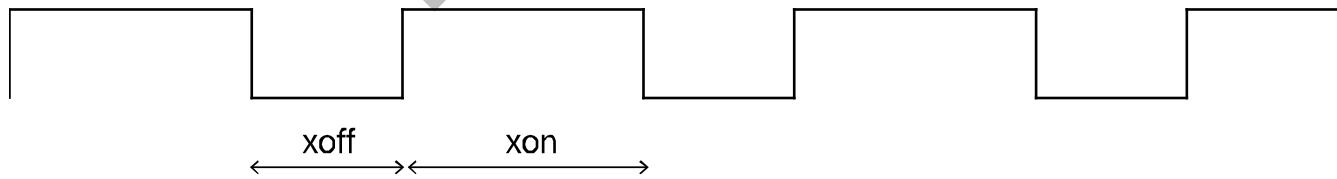


Figure 21-4. PWMB0/B1/C0/C1 Timing Diagram.

PWMB0/B1/C0/C1 generally use the PWM clock (`GCLK_PWM`) as a source clock, and `PWM_CLK = SOURCE_CLK / (divider+1)`. Refer to [Section 21.6.2 “PWMB0/B1/C0/C1: Clocking”](#) for an overview of this register programming.

21.6.2 PWMB0/B1/C0/C1: Clocking

As mentioned previously, the Pulse Width Modulators PWMB0/B1/C0/C1 generally use GCLK_PWM, which is set by system software. GCLK_PWM is typically sourced to GCLK_APB. Contact an Ambarella representative for additional detail.

The PWMB0/B1/C0/C1 pulse rates are calculated using GCLK_PWM with a divider defined on PWM registers ([Section 21.8.2.11](#), [Section 21.8.2.12](#) and [Section 21.8.2.13](#)). The relationship is illustrated below for SOURCE_CLK = GCLK_PWM:



Figure 21-5. PWMB0/B1/C0/C1 Clock Settings.

- GCLK_PWM is the current frequency.
- The PWMB0/B1/C0/C1 Pulse Rates are calculated with the PWM_xn_enable bit divider and PWM_xn_data bits **xon** and **xoff**. The calculation for one of the pulse rates is:

$$\text{PWM}[n] \text{ Pulse Rate} = \frac{\text{gclk_pwm} / (\text{divider} + 1)}{[(\text{xon} + 1) + (\text{xoff} + 1)]}$$

21.7 ST / MS / PWM: Micro Stepper Interface

This section contains programming information for the Micro Stepper (MS) interface. The section is organized as follows:

- [\(Section 21.7.1\) Micro Stepper: Overview](#)
- [\(Section 21.7.2\) Micro Stepper: Clocking](#)

21.7.1 Micro Stepper: Overview

The Micro Stepper interface sources to GLK_CORE ([Section 21.7.2](#)) and uses its own PWMs ([Section 21.8.2.15](#)) to generate a continuous micro-stepper wave. The figure below shows the output of the four micro steppers (MS0/1/2/3) after electrical circuit integration.

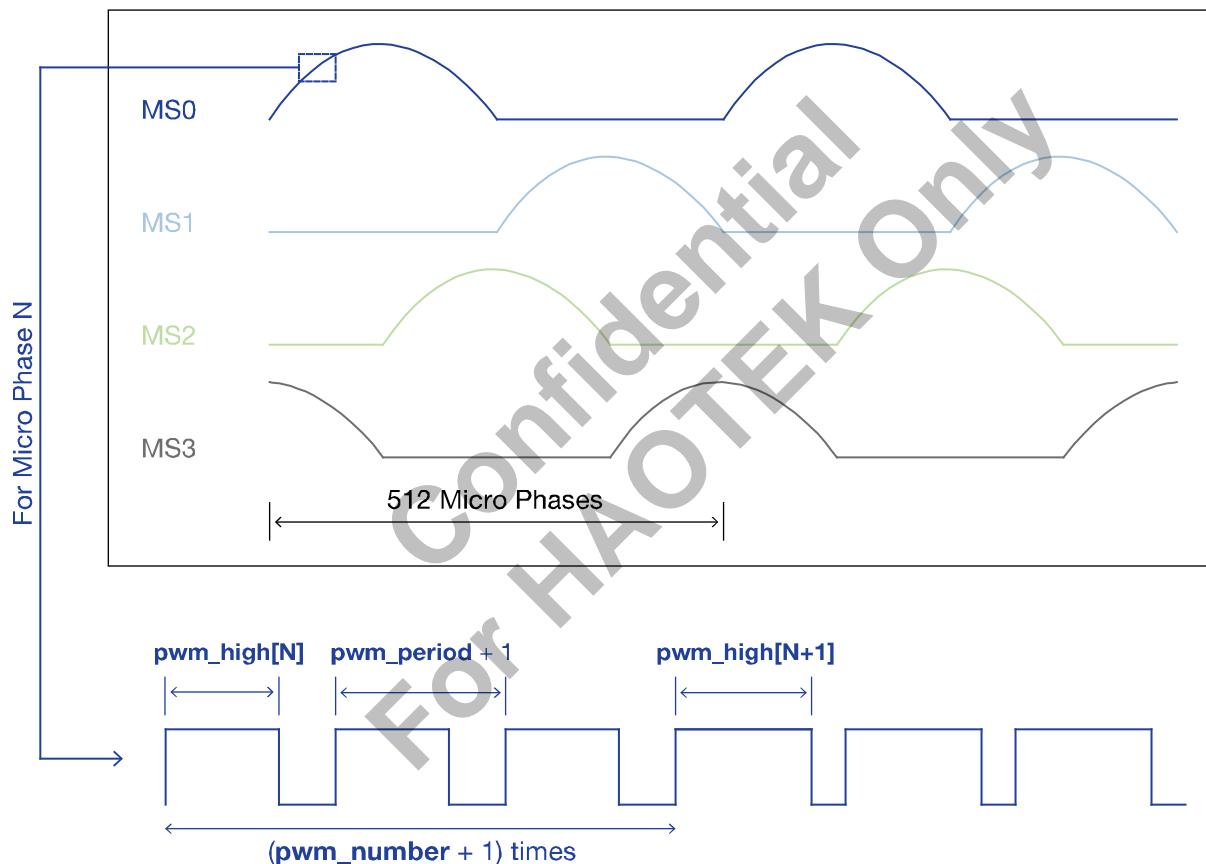


Figure 21-6. Micro Stepper Timing Diagram with Output From MS0/1/2/3 After Electrical Circuit Integration.

The PWM associated with a given microstepper (MS_PWM) is automatically enabled when that MS is enabled.

The cycle count of each PWM pulse is (`pwm_period + 1`), the number of PWM pulse widths per micro-step is (`pwm_number + 1`), and the number of micro-steps per stepper phase is 128 (i.e., four phases per sine wave). Use `MS_wave_xn [offset t]` registers `pwm_high0/1/2/3` to control the high period of a given microstep ([Section 21.8.2.20](#)).

There are four sets of micro-stepping MS motor control outputs, and the pins are shared with stepper ST interfaces `SC_A[0:3]`, `SC_B[0:3]`, `SC_C[0:3]`, and `SC_D[0:3]`. When enabled, the pins output micro stepper signals whether the PWM is enabled or disabled. To use this controller, follow the steps below:

1. Initialize the sine wave table:

- Before using the micro stepping motor controller, the software must create the sine wave table `MS_wave_xn` ([Section 21.8.2.20](#)) to set the PWM duty cycle for each micro step. Offset 0x800 is for the A0/A1 micro step 0 - 3 (`w1[0], w1[1], w1[2], w1[3]`); offset 0x804 is for the A0/A1 micro step 4 - 7. And so on.
- At micro step position n, the duty cycle of `a1 = w1[n]` if `n < 256` else 0; the duty cycle of `A2 = w1[n - 256]` if `n >= 256` else 0; the duty cycle of `A3 = w3[n-128]` if `128 <= n < 384` else 0; and the duty cycle of `A4 = w3[n + 128]` if `n < 128`, `a4 = w3[n-384]` if `n >= 384`, else 0; (`0 <= n < 512`).

2. Push the micro stepping motor:

- Set `MS_control_x` ([Section 21.8.2.14](#)), `MS_pwm_x` ([Section 21.8.2.15](#)), and `MS_count0_x` ([Section 21.8.2.16](#)) first. Next set `MS_count1_x` ([Section 21.8.2.17](#)) to start the motor. When the motor is running, changing `MS_pwm_x` changes the output speed. Software can check `MS_status_x` to determine the motor status and position.
- If the `MS_control_x` register bit `cont_repeat = 0`, the output sequence is `repeat_first, normal_output` followed by `repeat_last`. If `cont_repeat = 1`, the output is `repeat_first, normal_output` until `cont_repeat = 0`, followed by `repeat_last`.
- The interrupt occurs in the last phase of `repeat_last` when `cont_repeat = 0` or the last phase of `normal_output` when `cont_repeat = 1`.

3. Force stop:

- Setting the `MS_control_x` register bit `force_stop` stops the motor immediately, and the interrupt occurs.

4. `MS_pre_count`:

- When the interrupt occurs, the current position (`MS_control_x` register bit `repeat_count`) is stored into the `MS_pre_count` register `pre_repeat_count` field ([Section 21.8.2.19](#)). The user reads this register to learn the previous status.

5. MOB interrupt:

- When the MOB interrupt is enabled the motor sends an interrupt at each corresponding position according to the setting of the `ST_interrupt_en` bit field `freq_mob_x` ([Section 21.8.2.10](#)). For example, if `freq_mob_x = 0`, the interrupt is sent at 45, 90, 135, ..., 360 (0) degrees.

21.7.2 Micro Stepper: Clocking

The Micro Stepper sources directly to GCLK_CORE, which is set by system software. Contact an Ambarella representative for additional detail.

$$\text{MS Clock Frequency} = \text{GCLK_CORE} / (2 * (\text{clk_div} + 1)).$$

The maximum MS Clock Frequency equals the GCLK_CORE maximum.

The cycle count of each PWM pulse is (**pwm_period** + 1), the number of PWM pulse widths per micro-step is (**pwm_number** + 1), and the number of micro-steps per stepper phase is 128 (i.e., four phases per sine wave). The phase frequency, therefore, is:

$$\text{MS Phase Frequency} = \text{GCLK_CORE} / (2 * (\text{clk_div} + 1) * (\text{pwm_period} + 1) * (\text{pwm_number} + 1) * 128).$$

For Confidential
For HAOTEX Only

21.8 ST / MS / PWM: Registers

21.8.1 ST / MS / PWM Registers: Map

The software interface includes the following visible registers:

Register Offset	Register Name	Description
0x000	ST_control_a	Control Register of Interface A
0x004	ST_pattern0_0a	MSB Pattern for SC_A0 of Interface A
0x008	ST_pattern1_0a	LSB Pattern for SC_A0 of Interface A
0x00C	ST_pattern0_1a	MSB Pattern for SC_A1 of Interface A
0x010	ST_pattern1_1a	LSB Pattern for SC_A1 of Interface A
0x014	ST_pattern0_2a	MSB Pattern for SC_A2 of Interface A
0x018	ST_pattern1_2a	LSB Pattern for SC_A2 of Interface A
0x01C	ST_pattern0_3a	MSB Pattern for SC_A3 of Interface A
0x020	ST_pattern1_3a	LSB Pattern for SC_A3 of Interface A
0x024	ST_count_a	Count Register of Interface A
0x028	ST_status_a	Status Register of Interface A
0x02C	ST_acc_count_a	Accelerate/Decelerate Count
0x030	ST_acc_clkdiv_0a	Accelerate/Decelerate Clock Divider 0/1
0x034	ST_acc_clkdiv_1a	Accelerate/Decelerate Clock Divider 2/3
0x038	ST_pre_counter	Previous Running Counter
0x03C - 0x07C		Reserved
0x080 - 0x0B8	ST registers of B	Registers of Interface B
0x0BC - 0x0FC		Reserved
0x100 - 0x138	ST registers of C	Registers of Interface C
0x13C - 0x17C		Reserved
0x180 - 0x1B8	ST registers of D	Registers of Interface D
0x1BC - 0x1FC		Reserved
0x200 - 0x238	ST registers of E	Registers of Interface E
0x23C - 0x2FC		Reserved
0x300		Reserved
0x304	PWM_b0_enable	PWMB0 Enable
0x308		Reserved
0x30C	PWM_b1_enable	PWMB1 Enable
0x310		Reserved
0x314	PWM_c0_enable	PWMC0 Enable
0x318		Reserved
0x31C	PWM_c1_enable	PWMC1 Enable
0x320	PWM_b0_data1	Register Bank for PWMB0
0x324	PWM_b1_data1	Register Bank for PWMB1
0x328	PWM_c0_data1	Register Bank for PWMC0
0x32C	PWM_c1_data1	Register Bank for PWMC1
0x330 - 0x37C		Reserved
0x380	ST_interrupt	Interrupt Status and Clear Control
0x384	ST_interrupt_en	Interrupt Enable
0x388 - 0x3FC		Reserved
0x400	MS_control_a	Control Register of Micro Stepper A

Register Offset	Register Name	Description
0x404	MS_pwm_a	PWM Control Register of Micro Stepper A
0x408	MS_count0_a	Count Register 0 of Micro Stepper A
0x40C	MS_count1_a	Count Register 1 of Micro Stepper A
0x410	MS_status_a	Status Register of Micro Stepper A
0x414	MS_pre_count_a	Previous Running Counter of Micro Stepper A
0x500 - 0x514	MS registers of b	Registers of Micro Stepper B
0x600 - 0x614	MS registers of c	Registers of Micro Stepper C
0x700 - 0x714	MS registers of d	Registers of Micro Stepper D
0x800 - 0x8FC	MS_wave_a0	Configuration of each micro step of Micro Stepper A0/A1
0x900 - 0x9FC	MS_wave_a2	Configuration of each micro step of Micro Stepper A2/A3
0xA00 - 0xAF0	MS_wave_b0	Configuration of each micro step of Micro Stepper B0/B1
0xB00 - 0xBFC	MS_wave_b2	Configuration of each micro step of Micro Stepper B2/B3
0xC00 - 0xCF0	MS_wave_c0	Configuration of each micro step of Micro Stepper C0/C1
0xD00 - 0xDF0	MS_wave_c2	Configuration of each micro step of Micro Stepper C2/C3
0xE00 - 0xEF0	MS_wave_d0	Configuration of each micro step of Micro Stepper D0/D1
0xF00 - 0xFF0	MS_wave_d2	Configuration of each micro step of Micro Stepper D2/D3

Table 21-2. Stepper Interface Register Map.

(i) Note:

A number of stepper-related registers are not applicable to A12M SoCs. Please consult the relevant chip data-sheet for a comprehensive list of supported features and specifications.

21.8.2 ST / MS / PWM Registers: Detail

21.8.2.1 ST_control_x Register

Bits	Name	Attr	Reset	Description
31	reset	RW	0	Internal bit pointer reset. Clears automatically.
30:22				Reserved
21:16	patlen	RW	0	Only the left (patlen +1) bits of pattern are valid.
15:14				Reserved
13	set_last	RW	0	0 - Hold last phase state 1 - Set pins to last_pol after last phase
12	last_pol	RW	0	Last polarity 0 - Set pins to logic low after last phase if set_last = 1 1 - Set pins to logic high after last phase if set_last = 1
11:0	clkdiv	RW	0	The phase clock of the interface is GCLK_MOTOR / [2*(clkdiv + 1)]

Table 21-3. ST Control Register for Stepper Interface A/B/C/D/E.

Note that system software generally sets / configures the Stepper source clock GCLK_MOTOR, and register programming is used for adjustment. See [Section 21.5.2](#) for details.

21.8.2.2 ST_pattern0/1_nx Registers

Bits	Name	Attr	Reset	Description
31:0	pattern	RW	0	Left most bit of ST_pattern0_px is the first bit of the pattern, and right most bit of ST_pattern1_px is the last bit. Bits from left most bit up to and including patlen (ST_control_x[21:16]) are valid. (Valid pattern can be less than 64 bits)

Table 21-4. ST LSB and MSB Patterns for Pins 0 - 4 of Stepper Interface A/B/C/D/E.

21.8.2.3 ST_count_x Register

Bits	Name	Attr	Reset	Description
31				Reserved
30:24	repeat_last	RW	0	Output the last phase (repeat_last *4+1) times
23	rewind_en	RW	0	Rewind enable
22:16	repeat_first	RW	0	Output the first phase (repeat_first *4+1) times
15:0	count	RW	0	Do (count +1) phase

Table 21-5. ST Count Register of Stepper Interface A/B/C/D/E.

21.8.2.4 ST_status_x Register

Bits	Name	Attr	Reset	Description
31:28				Reserved
27	active_flag	R	0	Active flag status
26	repeat_first_flag	R	0	Repeat first flag status
25	repeat_flag	R	0	Repeat flag status
24	repeat_last_flag	R	0	Repeat last flag status
23:16				Reserved
15:0	repeat_count	R	0	Repeat counter

Table 21-6. ST Status Register of Stepper Interface A/B/C/D/E.

21.8.2.5 ST_acc_count_x Register

Bits	Name	Attr	Reset	Description
31:18				Reserved
17:16	acc_type	RW	0	0x0 - No acceleration and deceleration 0x1 - Acceleration without deceleration 0x2 - Deceleration without acceleration 0x3 - Both acceleration and deceleration
15:10				Reserved
9:0	acc_count	RW	0	Accelerate/decelerate (acc_count + 1) each step.

Table 21-7. ST Accelerate/Decelerate Count Register of Stepper Interface A/B/C/D/E.

21.8.2.6 ST_acc_clkdiv_0x Register

Bits	Name	Attr	Reset	Description
31:28				Reserved
27:16	acc_clkdiv1	RW	0	The phase clock of acc1/dec1 is GCLK_MOTOR / [2*(acc_clkdiv1 + 1)]
15:12				Reserved
11:0	acc_clkdiv0	RW	0	The phase clock of acc0/dec0 is GCLK_MOTOR / [2*(acc_clkdiv0 + 1)]

Table 21-8. ST Accelerate/Decelerate Clock Divider 0/1 of Stepper Interface A/B/C/D/E.

Note that system software generally sets / configures the Stepper source clock GCLK_MOTOR, and register programming is used for adjustment. See [Section 21.5.2](#) for details.

21.8.2.7 ST_acc_clkdiv_1x Register

Bits	Name	Attr	Reset	Description
31:28				Reserved
27:16	acc_clkdiv3	RW	0	The phase clock of acc3/dec3 is GCLK_MOTOR / [2*(acc_clkdiv3+1)]
15:12				Reserved
11:0	acc_clkdiv2	RW	0	The phase clock of acc2/dec2 is GCLK_MOTOR / [2*(acc_clkdiv2+1)]

Table 21-9. ST Accelerate/Decelerate Clock Divider 2/3 of Stepper Interface A/B/C/D/E.

Note that system software generally sets / configures the Stepper source clock GCLK_MOTOR, and register programming is used for adjustment. See [Section 21.5.2](#) for details.

21.8.2.8 ST_pre_counter_x Register

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:0	pre_counter	R	0	The counter of the previous sequence's end point. It updates when the sequence ends or terminates early.

Table 21-10. ST Previous Running Counter of Stepper Interface A/B/C/D/E.

21.8.2.9 ST_interrupt Register

Bits	Name	Attr	Reset	Description
31:13				Reserved
12	interrupt_mob_d	RW	0	The interrupt status of Micro Stepper D, asserted when Micro Stepper D is in MOB (at each 45 degree position), cleared by the software.
11	interrupt_mob_c	RW	0	The interrupt status of Micro Stepper C, asserted when Micro Stepper C is in MOB (at each 45 degree position), cleared by the software.
10	interrupt_mob_b	RW	0	The interrupt status of Micro Stepper C, asserted when Micro Stepper B is in MOB (at each 45 degree position), cleared by the software.
9	interrupt_mob_a	RW	0	The interrupt status of Micro Stepper C, asserted when Micro Stepper A is in MOB (at each 45 degree position), cleared by the software.
8	interrupt_ms_d	RW	0	The interrupt status of Micro Stepper D, asserted when Micro Stepper D is at end, cleared by the software.
7	interrupt_ms_c	RW	0	The interrupt status of Micro Stepper C, asserted when Micro Stepper C is at end, cleared by the software.
6	interrupt_ms_b	RW	0	The interrupt status of Micro Stepper B, asserted when Micro Stepper B is at end, cleared by the software.
5	interrupt_ms_a	RW	0	The interrupt status of Micro Stepper A, asserted when Micro Stepper A is at end, cleared by the software.
4	interrupt_e	RW	0	The interrupt status of Stepper E, asserted when Stepper E is at end, cleared by the software.
3	interrupt_d	RW	0	The interrupt status of Stepper D, asserted when Stepper D is at end, cleared by the software.
2	interrupt_c	RW	0	The interrupt status of Stepper C, asserted when Stepper C is at end, cleared by the software.
1	interrupt_b	RW	0	The interrupt status of Stepper B, asserted when Stepper B is at end, cleared by the software.
0	interrupt_a	RW	0	The interrupt status of Stepper A, asserted when Stepper A is at end, cleared by the software.

Table 21-11. ST Interrupt Status and Clear Control Register.

21.8.2.10 ST_interrupt_en Register

Bits	Name	Attr	Reset	Description
31:21				Reserved
20:19	freq_mob_d	RW	0	The frequency of interrupt_mob_d 0 - Interrupt every 45 degrees 1 - Interrupt every 90 degrees 2 - Interrupt every 180 degrees 3 - Interrupt every 360 degrees
18:17	freq_mob_c	RW	0	The frequency of interrupt_mob_c 0 - Interrupt every 45 degrees 1 - Interrupt every 90 degrees 2 - Interrupt every 180 degrees 3 - Interrupt every 360 degrees
16:15	freq_mob_b	RW	0	The frequency of interrupt_mob_b 0 - Interrupt every 45 degrees 1 - Interrupt every 90 degrees 2 - Interrupt every 180 degrees 3 - Interrupt every 360 degrees
14:13	freq_mob_a	RW	0	The frequency of interrupt_mob_a 0 - Interrupt every 45 degrees 1 - Interrupt every 90 degrees 2 - Interrupt every 180 degrees 3 - Interrupt every 360 degrees
12	int_en_mob_d	RW	0	The interrupt enable of interrupt_mob_d
11	int_en_mob_c	RW	0	The interrupt enable of interrupt_mob_c
10	int_en_mob_b	RW	0	The interrupt enable of interrupt_mob_b
9	int_en_mob_a	RW	0	The interrupt enable of interrupt_mob_a
8	int_en_ms_d	RW	0	The interrupt enable of interrupt_ms_d
7	int_en_ms_c	RW	0	The interrupt enable of interrupt_ms_c
6	int_en_ms_b	RW	0	The interrupt enable of interrupt_ms_b
5	int_en_ms_a	RW	0	The interrupt enable of interrupt_ms_a
4	int_en_e	RW	0	The interrupt enable of interrupt_e
3	int_en_d	RW	0	The interrupt enable of interrupt_d
2	int_en_c	RW	0	The interrupt enable of interrupt_c
1	int_en_b	RW	0	The interrupt enable of interrupt_b
0	int_en_a	RW	0	The interrupt enable of interrupt_a

Table 21-12. ST Interrupt Enable Register.

21.8.2.11 PWM_x0_enable Register

Bits	Name	Attr	Reset	Description
31				Reserved
30:1	divider	RW	0	Clock divider of PWM (PWM_CLK = SOURCE_CLK/(divider+1))
0	pwm_en	RW	0	PWM enable

Table 21-13. PWM Enable Register for PWMB0 and PWMC0.

Note that system software generally sets / configures the PWM source clock, and register programming is used

for adjustment. See [Section 21.6.2](#) for details.

21.8.2.12 PWM_x1_enable Register

Bits	Name	Attr	Reset	Description
31	inv_en	RW	0	When 1, set x1 becomes an inverted version of x0, and the register settings for x1 are ignored.
30:1	divider	RW	0	Clock divider of PWM (PWM_CLK = SOURCE_CLK/(divider+1))
0	pwm_en	RW	0	PWM enable

Table 21-14. PWM Enable Register for PWMB1 and PWMC1.

Note that system software generally sets / configures the PWM source clock, and register programming is used for adjustment. See [Section 21.6.2](#) for details.

21.8.2.13 PWM_xn_data1 Register

Bits	Name	Attr	Reset	Description
31:16	xon	RW	0	Banked xon data pattern for PWM A read from this register will read the data from bank 1, and a write to this register will write to both banks.
15:0	xoff	RW	0	Banked xoff data pattern for PWM A read from this register will read the data from bank 1, and a write to this register will write to both banks.

Table 21-15. PWM Register Bank for PWMB0/B1/C0/C1.

Note that system software generally sets / configures the PWM source clock, and register programming is used for adjustment. See [Section 21.6.2](#) for details.

21.8.2.14 MS_control_x Register

Bits	Name	Attr	Reset	Description
31:12				Reserved
11	ms2_enable	RW	0	Micro Stepper x2 / x3 output enable
10	ms0_enable	RW	0	Micro Stepper x0 / x1 output enable
9	reset	RW	0	Internal bit pointer reset (automatically clears)
8	force_stop	RW	0	Force stop (automatically clears)
7	cont_repeat	RW	0	Continuous repeat
6	set_last	RW	0	0 - Hold last phase state 1 - Set pins to last_pol after last phase
5	last_pol	RW	0	0 - Set pins to logic low after last phase if set_last = 1 1 - Set pins to logic high after last phase if set_last = 1
4	pwm_inv	RW	0	Invert PWM polarity
3:0	clk_div	RW	0	Clock divider, CLK = GCLK_CORE / (clk_div + 1)

Table 21-16. MS Control Register of Micro Stepper A/B/C/D.

Note that the Micro Stepper sources directly to GCLK_CORE. The GCLK_CORE frequency is controlled by the

A12 system software. See [Section 21.7.2](#) for details.

21.8.2.15 MS_pwm_x Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:16	pwm_period	RW	127	Period of PWM = (pwm_period + 1)
15:0	pwm_number	RW	7	Each micro step contains (pwm_number + 1) PWM waves

Table 21-17. MS PWM Control Register of Micro Stepper A/B/C/D.

21.8.2.16 MS_count0_x Register

Bits	Name	Attr	Reset	Description
31:16				Reserved
15:8	repeat_last	RW	0	Repeat the last phase (repeat_last *256) times
7:0	repeat_first	RW	0	Output the first phase (repeat_first *256) times

Table 21-18. MS Count Register 0 of Micro Stepper A/B/C/D.

21.8.2.17 MS_count1_x Register

Bits	Name	Attr	Reset	Description
31:25				Reserved
24	rewind_en	RW	0	Rewind enable
23:0	count	RW	0	Do (count +1) micro steps

Table 21-19. MS Count Register 1 of Micro Stepper A/B/C/D.

21.8.2.18 MS_status_x Register

Bits	Name	Attr	Reset	Description
31:28				Reserved
27	active_flag	R	0	The flag of active status
26	repeat_first_flag	R	0	The flag of repeat first status
25	repeat_flag	R	0	The flag of repeat status
24	repeat_last_flag	R	0	The flag of repeat last status
23:0	repeat_count	R	0	Repeat counter

Table 21-20. MS Status Register of Micro Stepper A/B/C/D.

21.8.2.19 MS_pre_count_x Register

Bits	Name	Attr	Reset	Description
31:24				Reserved
23:0	pre_repeat_count	R	0	The counter of the previous sequence's end point. The counter will be updated when the sequence ends or is forced to stop. Note that reading when the micro stepping motor is active may produce unexpected behavior.

Table 21-21. MS Previous Running Counter of Micro Stepper A/B/C/D.
21.8.2.20 MS_wave_xn [offset t] Register

Bits	Name	Attr	Reset	Description
31:24	pwm_high3	RW		PWM high period of micro step [t+3]
23:16	pwm_high2	RW		PWM high period of micro step [t+2]
15:8	pwm_high1	RW		PWM high period of micro step [t+1]
7:0	pwm_high0	RW		PWM high period of micro step [t]

Table 21-22. MS PWM Wave Register for Micro Stepper A/B/C/D.

22. PULSE WIDTH MODULATOR (PWM)

This chapter overviews the A12 pulse width modulator (PWM) controller interface as follows:

- [\(Section 22.1\) PWM Overview](#)
- [\(Section 22.2\) PWM Registers: Map](#)

22.1 PWM Overview

The A12 Pulse Width Modulator interfaces (PWM0/1/2/3) are included as components of the Stepper Motor Controller ([Chapter 21](#)). The base address for the PWM interface is 0xE800.8000.

The following features are provided:

- Four Pulse Width Modulation interfaces. The four PWM outputs are referred to as PWM_[0:3] in the A12 chip datasheet. Functionally:
 - PWMB0 is associated with PWM_0
 - PWMB1 is associated with PWM_1
 - PWMC0 is associated with PWM_2
 - PWMC1 is associated with PWM_3
- Note that in addition to the PWM controller embedded in the stepper motor controller, the A12 external pin **VD_PWM** can also serve as a PWM controller. **VD_PWM** is identical to PWM_0.

22.2 PWM Registers: Map

The software interface includes the visible registers shown below. For further information, including detailed register definitions, please refer to [Section 21.6 \(ST / MS / PWM: PWMB0/B1/C0/C1 Interface\)](#) and [Section 21.8 \(ST / MS / PWM: Registers\)](#).

Register Offset	Register Name	Description
0x000		Reserved
0x004	PWM_0_enable	PWM0 Enable
0x008		Reserved
0x00C	PWM_1_enable	PWM1 Enable
0x010		Reserved
0x014	PWM_2_enable	PWM2 Enable
0x018		Reserved
0x01C	PWM_3_enable	PWM3 Enable
0x020	PWM_0_data1	Register Bank for PWM0
0x024	PWM_1_data1	Register Bank for PWM1
0x028	PWM_2_data1	Register Bank for PWM2
0x02C	PWM_3_data1	Register Bank for PWM3
0x030 - 0xFFC		Reserved

Table 22-1. PWM Interface Register Map.

23. INTERVAL TIMERS

This chapter provides information for the A12 Interval Timers as follows:

- [\(Section 23.1\) Interval Timer: Overview](#)
- [\(Section 23.2\) Interval Timer: Clocking](#)
- [\(Section 23.3\) Interval Timer: Registers](#)

23.1 Interval Timer: Overview

Features of the Interval Timers include:

- Programmable 32-bit timer/counters
- 8 sets of independent counters for each timer instance.
- Interrupts generated at underflow (i.e., reached limit) or at regular intervals
- Two match registers
- Current value of each count register can be read at any time
- Sensitive to both edges when driven by an external clock source

23.2 Interval Timer: Clocking

The interval timers can be independently configured to be clocked internally or externally. When clocked internally, the timer runs at the same frequency as the APB bus clock.

The timers also support being clocked by an external source. When clocked by an external clock source, the interval timer is sensitive to both edges of the clock. In other words, the timer decrements on a rising edge as well as on a falling edge of the external clock.

Please refer to the chip A12 datasheet for details on the external pins used in A12 programming.

23.3 Interval Timer: Registers

23.3.1 Interval Timer Registers: Map

Register Offset	Register Name	Description
0x00	TM_cnt1_sts	Counter 1 Status
0x04	TM_cnt1_reload	Counter 1 Reload Value
0x08	TM_cnt1_match1	Counter 1 First Match
0x0C	TM_cnt1_match2	Counter 1 Second Match
0x10	TM_cnt2_sts	Counter 2 Status
0x14	TM_cnt2_reload	Counter 2 Reload Value
0x18	TM_cnt2_match1	Counter 2 First Match
0x1C	TM_cnt2_match2	Counter 2 Second Match
0x20	TM_cnt3_sts	Counter 3 Status
0x24	TM_cnt3_reload	Counter 3 Reload Value
0x28	TM_cnt3_match1	Counter 3 First Match
0x2C	TM_cnt3_match2	Counter 3 Second Match
0x30	TM_ctrl	Control
0x34	TM_cnt4_sts	Counter 4 Status
0x38	TM_cnt4_reload	Counter 4 Reload Value
0x3C	TM_cnt4_match1	Counter 4 First Match
0x40	TM_cnt4_match2	Counter 4 Second Match
0x44	TM_cnt5_sts	Counter 5 Status
0x48	TM_cnt5_reload	Counter 5 Reload Value
0x4C	TM_cnt5_match1	Counter 5 First Match
0x50	TM_cnt5_match2	Counter 5 Second Match
0x54	TM_cnt6_sts	Counter 6 Status
0x58	TM_cnt6_reload	Counter 6 Reload Value
0x5C	TM_cnt6_match1	Counter 6 First Match
0x60	TM_cnt6_match2	Counter 6 Second Match
0x64	TM_cnt7_sts	Counter 7 Status
0x68	TM_cnt7_reload	Counter 7 Reload Value
0x6C	TM_cnt7_match1	Counter 7 First Match
0x70	TM_cnt7_match2	Counter 7 Second Match
0x74	TM_cnt8_sts	Counter 8 Status
0x78	TM_cnt8_reload	Counter 8 Reload Value
0x7C	TM_cnt8_match1	Counter 8 First Match
0x80	TM_cnt8_match2	Counter 8 Second Match

Table 23-1. Interval Timer Registers.

23.3.2 Interval Timer Registers: Details

23.3.2.1 TM_cnt[n].sts Register

Bits	Name	Attr	Reset	Description
31:0	cnt[n].sts	RW	0	Counter status TM_cnt[n].sts register stores the current status of counter[n]. When the enable bit of TM_ctrl register is set, the counter will start to decrement. Software can modify the register value at any time.

Table 23-2. Interval Timer Counter Status Register.

23.3.2.2 TM_cnt[n].reload Register

Bits	Name	Attr	Reset	Description
31:0	reload[n]	RW	0	Counter[n] reload value register. The reload value is the interval. When the counter is decremented to 0, the reload value will be reloaded to the counter value automatically. For example, if reload is set to 2 the counter sequence will be 2,1,0,2,1,... Trigger one-shot mode by setting the TM_cnt[n].reload bit reload=0, the TM_cnt[n].match1 bit match1!=0, the TM_cnt[n].match2 bit match2!=0, and TM_ctrl bit of[n] =0.

Table 23-3. Interval Timer Reload Value Register.

23.3.2.3 TM_cnt[n].match1 Register

Bits	Name	Attr	Reset	Description
31:0	match1[n]	RW	0	First match register When counter[n] matches this register, the timer will generate an edge trigger interrupt to the ARM core. Trigger one-shot mode by setting the TM_cnt[n].reload bit reload=0, the TM_cnt[n].match1 bit match1!=0, the TM_cnt[n].match2 bit match2!=0, and TM_ctrl bit of [n] =0.

Table 23-4. Interval Timer Match 1 Register.

23.3.2.4 TM_cnt[n].match2 Register

Bits	Name	Attr	Reset	Description
31:0	match2[n]	RW	0	<p>Second set match register</p> <p>When counter[n] matches this register, the timer will generate an edge trigger interrupt to the ARM.</p> <p>Trigger one-shot mode by setting the TM_cnt[n].reload bit reload=0, the TM_cnt[n].match1 bit match1!=0, the TM_cnt[n].match2 bit match2!=0, and TM_ctrl bit of[n]=0.</p>

Table 23-5. Interval Timer Match 2 Register.

23.3.2.5 TM_ctrl 0x30 Register

Trigger one-shot mode by setting the TM_cnt[n].reload bit reload=0, the TM_cnt[n].match1 bit match1!=0, the TM_cnt[n].match2 bit match2!=0, and TM_ctrl bit of[n]=0.

Bits	Name	Attr	Reset	Description
31				Reserved
30	of8	RW	0	<p>Overflow for Timer/Counter 8</p> <p>0 - No interrupt on counter underflow</p> <p>1 - Generate interrupt on counter underflow</p>
29	clksel8	RW	0	<p>Clock selection for Timer/Counter 8</p> <p>0 - APB clock (PCLK)</p> <p>1 - External clock (EXTCLK)</p>
28	enable8	RW	0	<p>Timer enable for Timer/Counter 8</p> <p>0 - Disable</p> <p>1 - Enable</p> <p>Notes: When timer is disabled, all actions for counter, reload, and interrupt will be halted.</p>
27				Reserved
26	of7	RW	0	<p>Overflow for Timer/Counter 7</p> <p>0 - No interrupt on counter underflow</p> <p>1 - Generate interrupt on counter underflow</p>
25	clksel7	RW	0	<p>Clock selection for Timer/Counter 7</p> <p>0 - APB clock (PCLK)</p> <p>1 - External clock (EXTCLK)</p>
24	enable7	RW	0	<p>Timer enable for Timer/Counter 7</p> <p>0 - Disable</p> <p>1 - Enable</p> <p>Notes: When timer is disabled, all actions for counter, reload, and interrupt will be halted.</p>
23				Reserved
22	of6	RW	0	<p>Overflow for Timer/Counter 6</p> <p>0 - No interrupt on counter underflow</p> <p>1 - Generate interrupt on counter underflow</p>
21	clksel6	RW	0	<p>Clock selection for Timer/Counter 6</p> <p>0 - APB clock (PCLK)</p> <p>1 - External clock (EXTCLK)</p>

Bits	Name	Attr	Reset	Description
20	enable6	RW	0	Timer enable for Timer/Counter 6 0 - Disable 1 - Enable Notes: When timer is disabled, all action for counter, reload, and interrupt will be gated.
19				Reserved
18	of5	RW	0	Overflow for Timer/Counter 5 0 - No interrupt on counter underflow 1 - Generate interrupt on counter underflow
17	clksel5	RW	0	Clock selection for Timer/Counter 5 0 - APB clock (PCLK) 1 - External clock (EXTCLK)
16	enable5	RW	0	Timer enable for Timer/Counter 5 0 - Disable 1 - Enable Notes: When timer is disabled, all action for counter, reload, and interrupt will be gated.
15				Reserved
14	of4	RW	0	Overflow for Timer/Counter 4 0 - No interrupt on counter underflow 1 - Generate interrupt on counter underflow
13	clksel4	RW	0	Clock selection for Timer/Counter 4 0 - APB clock (PCLK) 1 - External clock (EXTCLK)
12	enable4	RW	0	Timer enable for Timer/Counter 4 0 - Disable 1 - Enable Notes: When timer is disabled, all action for counter, reload, and interrupt will be gated.
11				Reserved
10	of3	RW	0	Overflow for Timer/Counter 3 0 - No interrupt on counter underflow 1 - Generate interrupt on counter underflow
9	clksel3	RW	0	Clock selection for Timer/Counter 3 0 - APB clock (PCLK) 1 - External clock (EXTCLK)
8	enable3	RW	0	Timer enable for Timer/Counter 3 0 - Disable 1 - Enable Notes: When timer is disabled, all action for counter, reload, and interrupt will be gated.
7				Reserved
6	of2	RW	0	Overflow for Timer/Counter 2 0 - No interrupt on counter underflow 1 - Generate interrupt on counter underflow
5	clksel2	RW	0	Clock selection for Timer/Counter 2 0 - APB clock (PCLK) 1 - External clock (EXTCLK)

Bits	Name	Attr	Reset	Description
4	enable2	RW	0	Timer enable for Timer/Counter 2 0 - Disable 1 - Enable Notes: When timer is disabled, all action for counter, reload, and interrupt will be gated.
3				Reserved
2	of1	RW	0	Overflow for Timer/Counter 1 0 - No interrupt on counter underflow 1 - Generate interrupt on counter underflow
1	clksel1	RW	0	Clock selection for Timer/Counter 1 0 - APB clock (PCLK) 1 - External clock (EXTCLK)
0	enable1	RW	0	Timer enable for Timer/Counter 1 0 - Disable 1 - Enable Notes: When timer is disabled, all action for counter, reload, and interrupt will be gated.

Table 23-6. Interval Timer Control Register.

24. WATCH DOG TIMER (WDT)

This chapter provides information regarding the A12 Watch Dog Timer (WDT). The chapter is organized as follows:

- [\(Section 24.1\) WDT: Overview](#)
- [\(Section 24.2\) WDT: Registers](#)

24.1 WDT: Overview

The WDT is used to prevent system deadlock. When enabled, the software must reload the WDT periodically to prevent the watch dog counter from underflowing (decrementing to zero). The WDT runs at the same rate as the APB bus clock and can be programmed to:

- Generate a system reset signal
- Generate an interrupt to the system interrupt controller
- Perform none of the above

24.2 WDT: Registers

24.2.1 WDT Registers: Map

Register Offset	Register Name	Description
0x00	WDT_cnt_sts	Counter Status
0x04	WDT_reload	Counter Reload Value
0x08	WDT_restart	Counter Restart
0x0C	WDT_ctrl	Control
0x10	WDT_timeout	Timeout Status
0x14	WDT_clr	Clear Timeout Status
0x18	WDT_RST_WD	Reset Width

Table 24-1. APB Registers of the Watch Dog Timer (WDT)

24.2.2 WDT Registers: Details

24.2.2.1 WDT_cnt_sts Register

When enabled, the watch dog counter decrements by 1 every APB bus clock period. The current counter value is contained in this Counter Status register. Decrementing this register below 0 will cause an underflow condition to occur. Software can prevent this underflow by writing a new value into the Watch Dog Restart register.

Bits	Name	Attr	Reset	Description
31:0	cnt_sts	R	0x3EF1480	Current counter value of the watch dog timer The cnt_sts starts to decrement once the WDT_ctrl bit 0 (enable) is set.

Table 24-2. WDT Counter Status Register.

24.2.2.2 WDT_reload Register

Bits	Name	Attr	Reset	Description
31:0	reload	RW	0	This value reloads to the WDT_cnt_sts register when it underflows or when software writes “0x4755” to the restart register.

Table 24-3. WDT Counter Reload Value Register.

24.2.2.3 WDT_restart Register

To prevent the watch dog timer from underflowing, software must write to this register periodically and in a timely manner.

Bits	Name	Attr	Reset	Description
15:0	restart	W	0	When software writes “0x4755” to this register, values in the reload register will be loaded to the WDT_cnt_sts register.

Table 24-4. WDT Counter Restart Register.

24.2.2.4 WDT_ctrl Register

Bits	Name	Attr	Reset	Description
31:3				Reserved
2	intenable	RW	0	Interrupt enable 0 - Disable interrupt on counter underflow 1 - Enable interrupt on counter underflow
1	rstenable	RW	0	Reset system after timeout 0 - Disable system reset on counter underflow 1 - Enable system reset on counter underflow
0	enable	RW	0	WDT enable signal 0 - Disable watch dog timer 1 - Enable watch dog timer

Table 24-5. WDT Control Register.

24.2.2.5 WDT_timeout Register

Upon coming out of a reset, the software can examine the Timeout Status Register to determine if the last system reset was caused by a Watch Dog timeout.

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	timeout	R	0	Indicate timeout If the WDT has reached the timeout condition, this bit will be set automatically. 0 - Timeout has not occurred 1 - Timeout has occurred

Table 24-6. WDT Timeout Status Register.

24.2.2.6 WDT_clr Register

Bits	Name	Attr	Reset	Description
31:1				Reserved
0	clr	W	0	Clear timeout status Write 1 value into this bit to clear timeout register

Table 24-7. WDT Clear Timeout Status Register.

24.2.2.7 WDT_RST_WD Register

The watch dog timer can generate an interrupt to the VIC, as well as a reset signal to the RCT. The pulse width of the reset signal and the reset interrupt is controlled by this register.

Bits	Name	Attr	Reset	Description
31:8				Reserved
7:0	rstwd	RW	0	Reset width

Table 24-8. WDT Reset Width Register.

25. CRYPTOGRAPHY ENGINE

This chapter covers the A12 Cryptography Engine as follows:

- [\(Section 25.1\) Cryptography: Overview](#)
- [\(Section 25.2\) Cryptography: Restrictions](#)
- [\(Section 25.3\) Cryptography: Registers and Programming](#)

25.1 Cryptography: Overview

The A12 chip provides a cryptography engine (Cryptography Engine 0) on the Cortex-A9 AXI bus at base address 0xF002.0000.

The cryptography engine performs Advanced Encryption Standard (AES) and Data Encryption Standard (DES) operations, including 3DES using a software wrapper. The engine also provides hardware acceleration for the MD5 and SHA-1 hashing algorithms. Note that the A12 cryptography engine is not binary-compatible with micro-code from Ambarella chips A5, A5M, A6, or A7.

The following is a list of features of the A12 cryptography engine:

- Up to two sets of commands can be queued for each block:
 - AES: Two sets of keys / One key set, one data set / Two data sets, as applicable.
 - DES: Two sets of data / Any number of keys.
 - MD5 and SHA-1: Two sets of inputs to hash for any one of the blocks.
- Writing operation code is not required. Instead, encryption or decryption is performed depending upon the address to which the data is written.
- Result reading can occur in one of two ways depending on whether or not interrupts are enabled for the particular unit:
 - If interrupts are enabled, the interrupt service routine is triggered when the results are ready.
 - If interrupts are disabled, the ready-register of the block should be polled to determine when the results become ready.
- For the MD5 / SHA-1 block, a new set of inputs can be written if the block is ready for input (a separate register to be polled). It is not necessary to read the result of the previous inputs; however, for AES and DES, a maximum of two operations can be outstanding in the block. The result from the earlier operation must be read before a new operation is enqueued.

25.2 Cryptography: Restrictions

Programming of the cryptography module should adhere to the following restrictions:

- MD5 and SHA-1 operations can not be interleaved; i.e., to enqueue an MD5 operation, all pending SHA-1 results must be read-out (and vice versa).
- The MD5 / SHA-1 interrupt enable registers cannot be modified while there are pending MD5 / SHA-1 operations.
- AES and DES keys can be programmed only if all pending results have been read.

25.3 Cryptography: Registers and Programming

- [\(Section 25.3.1\) Cryptography Registers: Addressing](#)
- [\(Section 25.3.2\) Cryptography Registers: Map](#)
- [\(Section 25.3.3\) Cryptography Registers: Example Programming](#)
- [\(Section 25.3.4\) Cryptography Registers: Interrupts](#)

25.3.1 Cryptography Registers: Addressing

Note that the A12 cryptography block uses 64-bit wide registers. These registers should be accessed only on 64-bit boundaries. Accessing the cryptography block on 32-bit boundaries may result in undefined behavior.

25.3.2 Cryptography Registers: Map

Coprocessor ID Offset [11:9]	Register Offset [8:0]	Register Name	Description
0x0	0x0	DES_key_offset	DES Key
0x0	0x8	DES_encrypt_input_offset	DES Encryption Input
0x0	0x10	DES_decrypt_input_offset	DES Decryption Input
0x0	0x18	DES_output_offset	DES Output
0x0	0x20	DES_output_ready_offset	DES Output Ready Status
0x0	0x28	DES_interrupt_enable	DES Interrupt Enable
0x1	0x0	AES_key_256_255_192_offset	256 Bit AES Key [255:192]
0x1	0x8	AES_key_256_191_128_offset	256 Bit AES Key [191:128]
0x1	0x10	AES_key_256_127_64_offset	256 Bit AES Key [127:64]
0x1	0x18	AES_key_256_63_0_offset	256 Bit AES Key [63:0]
0x1	0x20	AES_key_192_191_128_offset	192 Bit AES Key [191:128]
0x1	0x28	AES_key_192_127_64_offset	192 Bit AES Key [127:64]
0x1	0x30	AES_key_192_63_0_offset	192 Bit AES Key [63:0]
0x1	0x38	AES_key_128_127_64_offset	128 Bit AES Key [127:64]
0x1	0x40	AES_key_128_63_0_offset	128 Bit AES Key [63:0]
0x1	0x48	AES_encrypt_input_127_64_offset	AES Encryption Input [127:64]

Coprocessor ID Offset [11:9]	Register Offset [8:0]	Register Name	Description
0x1	0x50	AES_encrypt_input_63_0_offset	AES Encryption Input [63:0]
0x1	0x58	AES_decrypt_input_127_64_offset	AES Decryption Input [127:64]
0x1	0x60	AES_decrypt_input_63_0_offset	AES Decryption Input [63:0]
0x1	0x68	AES_output_127_64_offset	AES Output [127:64]
0x1	0x70	AES_output_63_0_offset	AES Output [63:0]
0x1	0x78	AES_output_ready	AES Output Ready Status
0x1	0x80	AES_interrupt_enable	AES Interrupt Enable
0x2	0x0	SHA1_init_hash_63_0_offset	SHA-1 Initial Hash [63:0]
0x2	0x8	SHA1_init_hash_127_64_offset	SHA-1 Initial Hash [127:64]
0x2	0x10	SHA1_init_hash_159_128_offset	SHA-1 Initial Hash [159:128]
0x2	0x18	SHA1_input_63_0_offset	SHA-1 Input [63:0]
0x2	0x20	SHA1_input_127_64_offset	SHA-1 Input [127:64]
0x2	0x28	SHA1_input_191_128_offset	SHA-1 Input [191:128]
0x2	0x30	SHA1_input_255_192_offset	SHA-1 Input [255:192]
0x2	0x38	SHA1_input_319_256_offset	SHA-1 Input [319:256]
0x2	0x40	SHA1_input_383_320_offset	SHA-1 Input [383:320]
0x2	0x48	SHA1_input_447_384_offset	SHA-1 Input [447:384]
0x2	0x50	SHA1_input_511_448_offset	SHA-1 Input [511:448]
0x2	0x58	SHA1_output_63_0_offset	SHA-1 Output [63:0]
0x2	0x60	SHA1_output_127_64_offset	SHA-1 Output [127:64]
0x2	0x68	SHA1_output_159_128_offset	SHA-1 Output [159:128]
0x2	0x70	SHA1_output_ready_offset	SHA-1 Output Ready Status
0x2	0x78	SHA1_interrupt_enable	SHA-1 Interrupt Enable
0x2	0x80	MD5_init_hash_63_0_offset	MD5 Initial Hash [63:0]
0x2	0x88	MD5_init_hash_127_64_offset	MD5 Initial Hash [127:64]
0x2	0x90	MD5_input_63_0_offset	MD5 Input [63:0]
0x2	0x98	MD5_input_127_64_offset	MD5 Input [127:64]
0x2	0xA0	MD5_input_191_138_offset	MD5 Input [191:128]
0x2	0xA8	MD5_input_255_192_offset	MD5 Input [255:192]
0x2	0xB0	MD5_input_319_256_offset	MD5 Input [319:256]
0x2	0xB8	MD5_input_383_320_offset	MD5 Input [383:320]
0x2	0xC0	MD5_input_447_384_offset	MD5 Input [447:384]
0x2	0xC8	MD5_input_511_448_offset	MD5 Input [511:448]
0x2	0xD0	MD5_output_63_0_offset	MD5 Output [63:0]
0x2	0xD8	MD5_output_127_64_offset	MD5 Output [127:64]
0x2	0xE0	MD5_output_ready_offset	MD5 Output Ready Status
0x2	0xE8	MD5_interrupt_enable	MD5 Interrupt Enable
0x2	0xF0	MD5_SHA1_ready_for_input	MD5 And SHA-1 Engine ‘Ready To Accept’ Inputs Status

Table 25-1. Software-Visible Cryptography Registers (All 64-bit).

25.3.3 Cryptography Registers: Example Programming

- [\(Section 25.3.3.1\) Cryptography Example: Setting the Coprocessor ID](#)
- [\(Section 25.3.3.2\) Cryptography Example: Set the DES Key / Input](#)
- [\(Section 25.3.3.3\) Cryptography Example: Read the DES Output](#)
- [\(Section 25.3.3.4\) Cryptography Example: Set the AES Key / Input](#)
- [\(Section 25.3.3.5\) Cryptography Example: Read the AES Output](#)
- [\(Section 25.3.3.6\) Cryptography Example: Write to the MD5 / SHA-1 Engine](#)
- [\(Section 25.3.3.7\) Cryptography Example: Initialize the Hashes for MD5 / SHA-1](#)
- [\(Section 25.3.3.8\) Cryptography Example: Write Inputs to MD5 / SHA-1](#)
- [\(Section 25.3.3.9\) Cryptography Example: Read Results from MD5 / SHA-1](#)
- [\(Section 25.3.3.10\) Cryptography Example: Interrupt Enables](#)

25.3.3.1 Cryptography Example: Setting the Coprocessor ID

Depending on the co-processor to be programmed, the appropriate co-processor ID (bits 11:9 of the register address) is loaded into the register used for addressing. As can be seen from the Register Map, 0x0 corresponds to AES; 0x1 to DES; and 0x2 to MD5 / SHA-1.

For the instruction sequences in [Section 25.3.3.2 “Cryptography Example: Set the DES Key / Input”](#) and [Section 25.3.3.3 “Cryptography Example: Read the DES Output”](#), the addressing registers should be prepared as below:

```
    . . .
MOV           R1, #DES_CPROC_ID
ORR           R1, R0, R1, lsl #9
@ Now, r1 has DES offset complete inside it
    . . .
```

For the instruction sequences in [Section 25.3.3.4 “Cryptography Example: Set the AES Key / Input”](#) and [Section 25.3.3.5 “Cryptography Example: Read the AES Output”](#), the addressing registers should be prepared as shown below:

```
    . . .
MOV           R1, #AES_CPROC_ID
ORR           R1, R0, R1, lsl #9
@ Now, r1 has AES offset complete inside it
    . . .
```

Writing to the MD5 / SHA-1 engine is discussed in [Section 25.3.3.6](#). For the instruction sequences in [Section 25.3.3.7 “Cryptography Example: Initialize the Hashes for MD5 / SHA-1”](#), [Section 25.3.3.8 “Cryptography Example: Write Inputs to MD5 / SHA-1”](#) and [Section 25.3.3.9 “Cryptography Example: Read Results from MD5 / SHA-1”](#), the addressing registers are prepared as below:

```

MOV          R1, #MD5_SHA1_CPROC_ID
ORR          R1, R0, R1, lsl #9
@ Now, r1 has MD5 / SHA1 offset complete inside it
...

```

25.3.3.2 Cryptography Example: Set the DES Key / Input

Assume the key to be utilized is stored in Registers 3 and 4. Register 3 holds bits 63:32, and Register 4 holds bits 31:0. In this case, the sequence of instructions to write the key to the DES engine is as below.

```

ORR          R2, R1, #DES_KEY_OFFSET
STM          R2, {R3-R4}
...

```

The programming above sets details such as:

Bits	Name	Attr	Reset	Description
63:0	des_key_offset	W	0	DES key register holding Key[63:0]

Table 25-2. Cryptography Engine Example DES Key Offset Register Definition.

Assume the input data to be encrypted or decrypted is stored in Registers 3 and 4. Register 3 holds bits 63:32, and Register 4 holds bits 31:0. In this case the sequence of instructions to write the input to the DES engine (for a purpose such as encryption) is as below. For decryption, R2 and **DES_decrypt_input_offset** should be joined using the ORR instruction, which computes the bitwise OR of the two values.

```

ORR          R2, R1, #DES_ENCRYPT_INPUT_OFFSET
STM          R2, {R3-R4}
...

```

This programming sets details such as:

Bits	Name	Attr	Reset	Description
63:0	des_enc_dec_offset	W	0	DES input register holding Data[63:0] for encryption or decryption

Table 25-3. Cryptography Engine Example DES Encryption/Decryption Input Offset Register Definition.

25.3.3.3 Cryptography Example: Read the DES Output

Prior to reading the output of the DES operation from the cryptography engine, the software should ensure that the output is ready. Polling can be used for this purpose. Alternatively, if interrupts are enabled, an interrupt service routine can be used to automatically read results when they become available. The program segment in this subsection uses the polling method and reads the results into Registers 3 and 4 of the CPU.

```

        . . .
        _waitForDes_${LabelCtr}:
        LDR          R2, [R1, #DES_OUTPUT_RDY_OFFSET]
        CMP          R2, #1
        BNE          _waitForDes_${LabelCtr}
        ORR          R2, R1, #DES_OUTPUT_OFFSET
        LDM          R2, {R3-R4}
        . . .

```

This sets registers as follows:

Bits	Name	Attr	Reset	Description
63:1	readout	R	0	Bits [63:1] are read back as 0
0	des_output_rdy_offset	R	0	DES output ready status

Table 25-4. Cryptography Engine Example DES Input Offsets Register Definition (Part A).

Bits	Name	Attr	Reset	Description
63:0	des_output_offset	R	0	DES input register holding Data[63:0] for encryption or decryption

Table 25-5. Cryptography Engine Example DES Input Offsets Register Definition (Part B).

25.3.3.4 Cryptography Example: Set the AES Key / Input

Setting the AES key is very similar to the corresponding DES operation, with the exception that the appropriate number of registers in the multiple-store instruction must be chosen. Assume the bits 255:0 are stored in terms of 32 bits each in Registers 3, 4, 5, 6, 7, 8, 9 and 10, with bits 255:224 in Register 3 and so on through bits 31:0 in Register 10. For this case, the sequence of instructions to write 128, 192 and 256 bit keys would be:

```

        . . .
        @ 128 bit key
        ORR          R2, R1, #AES_KEY_128_127_64_OFFSET
        STMIA        R2, {R7-R10}
        . . .
        . . .

```

```

@ 192 bit key
ORR          R2, R1, #AES_KEY_192_191_127_OFFSET
STMIA        R2, {R5-R10}
. . .
. . .
@ 256 bit key
ORR          R2, R1, #AES_KEY_256_255_192_OFFSET
STMIA        R2, {R3-R10}
. . .

```

The programming above sets details such as:

Bits	Name	Attr	Reset	Description
63:0	aes_key_[x]_[y]_[z]_offset	W	0	AES Data for key type (128,192 or 256) encryption or decryption

Table 25-6. Cryptography Engine Example AES Key Offsets Register.

Next, assume that the input data to be encrypted or decrypted is stored in Registers 3 through 6; i.e., Register 3 holds bits 127:96, and Register 6 hold bits 31:0. In this case, the sequence of instructions to write the input to the AES engine is as described below. Note that the input write address determines whether encryption or decryption is performed. Further, it is not possible to interleave encrypt and decrypt operations with each other; i.e., the four encrypt or decrypt inputs must be written in sequence.

```

. . .
ORR          R2, R1, #AES_ENCRYPT_INPUT_127_64_OFFSET
STMIA        R2, {R3-R6}
. . .
. . .
ORR          R2, R1, #AES_DECRYPT_INPUT_127_64_OFFSET
STMIA        R2, {R3-R6}
. . .

```

This programming sets details such as:

Bits	Name	Attr	Reset	Description
63:0	aes_enc_dec_inp_[y]_[z]_offset	W	0	AES input Data for encryption or decryption

Table 25-7. Cryptography Engine Example AES Input Offsets Register.

25.3.3.5 Cryptography Example: Read the AES Output

Before reading the output of the AES operation from the cryptography engine, the software must verify the output is ready. Polling can be used for this purpose. Alternatively, if interrupts are enabled, an interrupt service routine can be utilized to automatically read the results when they become available. The program segment in this sub-section uses the polling method and reads the results into Registers 3,4,5 and 6 of the CPU.

```

    . .
    _WaitForAes_${LabelCtr}:
    LDR          R2, [R1, #AES_OUTPUT_RDY_OFFSET]
    CMP          R2, #1
    BNE          _WaitForAes_${LabelCtr}

    ORR          R2, R1, #AES_OUTPUT_127_64_OFFSET
    LDMIA        R2, {R3-R6}
    . .

```

This programs set details such as:

Bits	Name	Attr	Reset	Description
63:1	readout	R	0	Bits [63:1] are read back as 0
0	aes_output_rdy_offset	R	0	AES output ready status

Table 25-8. Cryptography Engine Example AES Output Offsets Register (Part A).

Bits	Name	Attr	Reset	Description
63:0	aes_output_[y]_[z]_offset	R	0	AES operation output

Table 25-9. Cryptography Engine Example AES Output Offsets Register (Part B).

25.3.3.6 Cryptography Example: Write to the MD5 / SHA-1 Engine

Before enqueueing an operation to the MD5 / SHA-1 engine, the software must ensure that the engine is ready for input. The engine buffers can hold up to two sets of operations (one set of initialization hashes and one set of inputs / two sets of inputs / two sets of initialization hashes). The following program segment decides whether it is safe to write to the MD5 / SHA-1 engine:

```

    . .
    _WaitForShaMd5InputRdy_${LabelCtr}:
    LDR          R4, [R1, #MD5_SHA1_READY_FOR_INPUT]
    CMP          R4, #1
    BNE          _WaitForShaMd5InputRdy_${LabelCtr}
    . .

```

This programming sets details such as:

Bits	Name	Attr	Reset	Description
63:1	readout	R	0	Bits [63:1] are read back as 0
0	md5_sha1_rdy_for_input	R	0	'Ready for input' status of the hash engines

Table 25-10. Cryptography Engine Example MD5 / SHA-1 Ready for Input Offset Register.

25.3.3.7 Cryptography Example: Initialize the Hashes for MD5 / SHA-1

Once it has been verified that the hash engine can accept inputs (by using code such as the segment in [Section 25.3.3.6](#)), the hash initialization for the MD5 and SHA-1 engines is performed as described below. For SHA-1, it is assumed the initialization hashes are in registers 3 to 7 of the CPU, with register 3 holding the lowest-order word. For MD5, it is assumed the initialization hashes are in registers 3 to 6 of the CPU, with register 3 holding the lowest-order word.

```

    . . .
    ORR          R2, R1, #SHA1_INIT_HASH_63_0_OFFSET
    STMIA        R2, {R3-R7}
    . . .
    . . .
    ORR          R2, R1, #MD5_INIT_HASH_63_0_OFFSET
    STMIA        R2, {R3-R6}
    . . .

```

This programming sets details such as:

Bits	Name	Attr	Reset	Description
63:0	md5_sha1_init_hash_[y]_[z].offset	W	0	Initial hash entry for MD5 or SHA-1 hash engines

Table 25-11. Cryptography Engine Example MD5 / SHA-1 Initial Hash Offset Register.

25.3.3.8 Cryptography Example: Write Inputs to MD5 / SHA-1

Once it has been verified that the hash engine can accept inputs (by using code such as the segment in [Section 25.3.3.7](#)), the software can write inputs to the MD5 and SHA-1 engines as shown below. It is assumed the 512-bit input to be hashed is split into two 256-bit chunks, each loaded into registers 3 to 10 of the CPU before the appropriate writes are made to the engine. Note that it is the responsibility of the software to ensure message padding.

```

    . . .
    ORR          R2, R1, #SHA1_INPUT_63_0_OFFSET
    STMIA        R2, {R3-R10}
    . . .
    ORR          R2, R1, #SHA1_INPUT_319_256_OFFSET
    STMIA        R2, {R3-R10}
    . . .
    . . .
    ORR          R2, R1, #MD5_INPUT_63_0_OFFSET
    STMIA        R2, {R3-R10}
    . . .
    ORR          R2, R1, #MD5_INPUT_319_256_OFFSET
    STMIA        R2, {R3-R10}
    . . .

```

This programming sets details such as:

Bits	Name	Attr	Reset	Description
63:0	md5_sha1_input_[y]_[z]_offset	W	0	Input Data for MD5 or SHA-1 hash engines

Table 25-12. Cryptography Engine Example MD5 / SHA-1 Input Offset Register.

25.3.3.9 Cryptography Example: Read Results from MD5 / SHA-1

Prior to reading the output of the MD5 / SHA-1 operation, the software must verify the output is ready. Polling can be used for this purpose. Alternatively, if interrupts are enabled, an interrupt service routine can be utilized to automatically read the results when they become available. The program segment in this subsection uses the polling method and reads the results into Registers 3 to 7 of the CPU for SHA-1, and Registers 3 to 6 for MD5.

```

    . . .
    _WaitForSha1_${LabelCtr}:
    LDR          R2, [R1, #SHA1_OUTPUT_RDY_OFFSET]
    CMP          R2, #1
    BNE          _WaitForSha1_${LabelCtr}

    ORR          R2, R1, #SHA1_OUTPUT_63_0_OFFSET
    LDMIA        R2, {R3-R7}
    . . .
    . . .
    _WaitForMd5_${LabelCtr}:
    LDR          R2, [R1, #MD5_OUTPUT_RDY_OFFSET]
    CMP          R2, #1
    BNE          _WaitForMd5_${LabelCtr}

    ORR          R2, R1, #MD5_OUTPUT_63_0_OFFSET
    LDMIA        R2, {R3-R6}
    . . .

```

This programming sets details such as:

Bits	Name	Attr	Reset	Description
63:1	readout	R	0	Bits [63:1] are read back as 0
0	md5_sha1_output_rdy_offset	R	0	MD5 or SHA-1 output ready status

Table 25-13. Cryptography Engine Example MD5 / SHA-1 Output Offsets Register (Part A).

Bits	Name	Attr	Reset	Description
63:0	md5_sha1_output_y_z_offset	R	0	MD5 or SHA-1 operation output

Table 25-14. Cryptography Engine Example MD5 / SHA-1 Output Offsets Register (Part B).

25.3.3.10 Cryptography Example: Interrupt Enables

Writing the DES, AES, SHA-1, and MD5 Interrupt Enable registers can be performed through STR instructions.

Bits	Name	Attr	Reset	Description
63:1	readout	R	0	Bits [63:1] are ignored
0	des_aes_sha1_md5_intr_en	W	0	DES / AES / SHA-1 or MD5 interrupt enable

Table 25-15. Cryptography Engine Example Interrupt Enable Registers.

25.3.4 Cryptography Registers: Interrupts

The AES / DES and MD5 / SHA-1 engines can be set to generate interrupts upon completion of operations. The signals interfacing with the VIC are single-cycle pulses. To capture and process them, the VIC must be configured properly. This subsection presents the program segment used to set up the VIC before issuing operations to the Cryptography Engine.

```

. . .
MOV          R0, #AHB_REG_BASE_ADDRESS
ORR          R0, R0, #VIC2_OFFSET
@ Enable MD5 / SHA1, AES and DES interrupts on rising edge
MOV          R1, #0x02000000
ORR          R1, R1, #0x3000
STR          R1, [R0, #0x10]
STR          R1, [R0, #0x2c]
. . .

```

26. AMBARELLA CONTACT INFORMATION

For complete Ambarella contact information, please visit www.ambarella.com.

For Confidential
For HAOTEK Only

27. IMPORTANT NOTICE

All Ambarella design specifications, datasheets, drawings, files, and other documents (together and separately, "materials") are provided on an "as is" basis, and Ambarella makes no warranties, expressed, implied, statutory, or otherwise with respect to the materials, and expressly disclaims all implied warranties of noninfringement, merchantability, and fitness for a particular purpose. The information contained herein is believed to be accurate and reliable. However, Ambarella assumes no responsibility for the consequences of use of such information.

Ambarella Incorporated reserves the right to correct, modify, enhance, improve, and otherwise change its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

All products are sold subject to Ambarella's terms and conditions of sale supplied at the time of order acknowledgement. Ambarella warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with its standard warranty. Testing and other quality control techniques are used to the extent Ambarella deems necessary to support this warranty.

Ambarella assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using Ambarella components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

Ambarella does not warrant or represent that any license, either expressed or implied, is granted under any Ambarella patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Ambarella products or services are used. Information published by Ambarella regarding third-party products or services does not constitute a license from Ambarella to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from Ambarella under the patents or other intellectual property of Ambarella.

Reproduction of information from Ambarella documents is not permissible without prior approval from Ambarella.

Ambarella products are not authorized for use in safety-critical applications (such as life support) where a failure of the product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Customers acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of Ambarella products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by Ambarella. Further, Customers must fully indemnify Ambarella and its representatives against any damages arising out of the use of Ambarella products in such safety-critical applications.

Ambarella products are neither designed nor intended for use in automotive and military/aerospace applications or environments. Customers acknowledge and agree that any such use of Ambarella products is solely at the Customer's risk, and they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

28. TYPOGRAPHICAL CONVENTIONS

28.1 Formatting

This manual uses consistent formatting that conveys concepts at a glance and helps the reader navigate effectively through the text. Typographical conventions include:

Example	Description
AmbaGuiGen , DirectUSB Save , File > Save Power , Reset, Home	Software names GUI commands and command sequences Computer / Hardware buttons
Flash_IO_control da, status, enable	Register names and register fields. For example, Flash_IO_control is the register for global control of Flash I/O, and bit 17 (da) is used for DMA acknowledgement.
GPIO81, CLK_AU	Hardware external pins
VIL, VIH, VOL, VOH	Hardware pin parameters
INT_O, RXDATA_I	Hardware pin signals
amb_performance_t amb_operating_mode_t amb_set_operating_mode()	API details (e.g., functions, structures, and type definitions)
/usr/local/bin	User entries into software dialogues and GUI windows File names and paths Command line scripting and Code

Table 28-1. Typographical Conventions for Technical Documents.

Additional Ambarella typographical conventions include:

- Acronyms are given in UPPER CASE using the default font (e.g., AHB, ARM11 and DDRIO).
- Names of Ambarella documents and publicly available standards, specifications, and databooks appear in *italic* type.

28.2 Register Map and Details

This manual provides register maps for each peripheral (or peripheral function) followed by details regarding each register. The AHB / APB registers on the chip are 32-bits wide and word-addressed unless otherwise stated. The map includes the offset, name, and a brief description for each register. The details focus on the individual fields, their function, how to access the fields, and reset values.

28.3 Register Access

Each register description specifies how to access the register fields using the following conventions:

Access Type	Description
Read Only (R)	An application can read the register field only. Writes made to Read-Only fields have no effect.
Write Only (W)	The register field can be written by the application. Reading a Write-Only field will return undefined data.
Read and Write (RW)	The register field can be read and written by the application. For example, the application can set this field by writing 1 and can clear it by writing 0.
Read, Write, and Self Clear (RW_SC)	The register field can be read and written by the application (Read and Write), and cleared (to 0) by the hardware (Self Clear). The method used by the hardware to clear a given field is detailed in that field's description.
Read, Self Set, and Write Clear (R_WC)	The register field can be read by the application (Read), can be set (e.g., to 1) by the hardware on the occurrence of a certain internal event (Self Set), and can be cleared (e.g., to 0) by the application with a register write of 1 (Write Clear). A register write of 0 has no effect on this field. The conditions under which the hardware sets this field are detailed in that field's description (e.g., interrupt bits).
Read, Write Set, and Self Clear (R_WS_SC)	The register field can be read by the application, can be set (to 1) by the application with a register write (of 1), and is cleared (to 0) by the hardware. The application cannot clear this type of field, and a register write (of 0) to this bit has no effect. The conditions under which the hardware sets this field are detailed in that field's description (e.g., reset signals).
Read, Self Set, and Self Clear or Write Clear (R_SS_SC_WC)	Register field can be read by the application, can be set (to 1) by the hardware on the occurrence of a certain internal event, and can be cleared (to 0) either by hardware (Self Clear) or by the application with a register write of 0 (Write Clear). A register write of 1 to this bit has no effect. The method used by the hardware to set or clear a given field is detailed in that field's description.
Read Only and Write Trigger (R_WT)	Register field can be read by the application, and when a write operation is performed with any data value, an event is triggered, as specified in the field's description (e.g., Tx Poll Demand register).
Hardware Initialized (HWI)	The value is initialized at reset by system firmware. These bits are Write-Once after power-on reset, then they become Read-Only.

Table 28-2. Register Access Options.

28.4 Register Field Values

Register detail tables provide reset values for all registers. Typically these are numeric values. A notation “NR” is used to specify registers that are not reset and therefore are in an undefined state. Please note that the register field descriptions provide options for reset situations that are not predefined. The pound sign # is used to denote reset values with dependencies that are described in the table.

29. REVISION HISTORY

Our goal is to provide our customers with the highest-quality documentation possible, and to continuously improve our publications to ensure that your experience with Ambarella's products is a positive one. If you have any questions or comments regarding this document, please contact the Technical Writing team at docs@ambarella.com. Your feedback is welcomed and appreciated.

NOTE: Page/chapter numbers for previous drafts may differ from those of the current version.

Version	Date	Comments
0.1	27 Aug 2013	Draft Original
0.2	9 Sept 2013	Update to Chapters 2, 4, 14, 15, 16, 17, 18 and 21
0.3	18 Sept 2013	Update memory map table; base addresses; VIC registers
0.5	19 Nov 2013	Add Secure / Scratchpad chapter; add Cryptography chapter; update ADC / PWC / RTC chapter; update VIC chapter
0.6	20 Nov 2013	Update UART chapter; update ADC chapter
0.7	25 April 2014	Update Overview, DMA, VIN, GPIO, IDC, SSI, Flash and Stepper chapters; add PWM chapter; update formatting
0.8	13 May 2014	Add NOR-SPI programming notes; I2S channel clarification; PWM offset address update
0.9	29 July 2014	Update UART, EHCI interrupt, PWC information
1.0	22 Oct 2014	Updated HDMI register DI_aunit_src; Added bit to Ethernet Controller MAC Register 0; Clarified IO sharing for SD controllers; Marked the led_control bit as reserved for the SD_hpbw_control register
1.1	15 Dec 2014	Add DMA_ch0_spare_desc_addr; Add 1-bit eMMC boot mode information
1.2	23 Feb 2015	Updated Section 13.4.7.13 ; Correct the VIC_int_en_clear register description
1.3	29 May 2015	Changed pwc_status(0x3c) to pwc_sta(0xc0) ; Remove note restricting pending/ongoing transactions from Section 13.1 ; Update Section 13.1.2
1.4	26 January 2016	Modify description of strapping pin 31 (VDO_HVLD) operations in Section 2.3.10 (removed bits 4 and 5 from the description) ; Changed CLK_CSR_I references to Core Clock in Section 10.3.2.5 ; Added Section 13.4 covering the SD PHY ; Modified trdelay description in Section 13.3.3.7 ; Modified description of bit 15 in Section 10.3.2.1
1.5	9 May 2016	Modified the comments in the third row of Table 13-9 ("The tg field of Flash_IO_dma_control") ; Removed the following statement from Section 13.5.1: "The SD and NAND flash or other logic may share I/O pins" ; Marked bit 3 in Table 13-6 as Reserved
1.6	14 June 2016	Modified the descriptions of the WDT_reload and WDT_restart registers in Sections 24.2.2.2 and 24.2.2.3
1.7	22 December 2016	Removed the eMMC boot descriptions in POC[15:14] ; Removed references to sensor clock GCLK_SO ; Added DMA_R10_axi_bus_mode Register to Ethernet chapter

Table 29-1. Revision History