MCIMX27 Multimedia Applications Processor

Reference Manual

MCIMX27RM
Rev. 0.2
9/2007

ARM POWERED®

freescale™ semiconductor

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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

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

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

***For Literature Requests Only*:**
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Contents

Chapter 4
System Control

Chapter 5
Signal Descriptions and Pin Assignments

Chapter 6
General-Purpose I/O (GPIO)

Chapter 7
JTAG Controller (JTAGC)

Chapter 8
Bootstrap Mode Operation

Chapter 9
ARM9 Platform

Chapter 10
ARM926EJ-S Interrupt Controller (AITC)

Chapter 11
Security Controller (SCC)

Chapter 12
Symmetric/Asymmetric Hashing and Random Accelerator (SAHARA2)

Chapter 13
Run-Time Integrity Checker (RTIC)

Chapter 14
IC Identification (IIM)

Chapter 15
External Memory Interface (EMI)

Chapter 16
Multi-Master Memory Interface (M3IF)

Chapter 17
Wireless External Interface Module (WEIM)

Chapter 18
Enhanced SDRAM Controller (ESDRAMC)

Chapter 19
NAND Flash Controller (NFC)

Chapter 20
Personal Computer Memory Card International Association (PCMCIA) Controller

Chapter 21
1-Wire Interface (1-Wire)

Chapter 22
Advanced Technology Attachment (ATA)

Chapter 23
Configurable Serial Peripheral Interface (CSPI)

Chapter 24
Inter-Integrated Circuit (I^2C)

Chapter 25
Keypad Port (KPP)

Chapter 26
Memory Stick Host Controller (MSHC)

Chapter 27
Secured Digital Host Controller (SDHC)

Chapter 28
Universal Asynchronous Receiver/Transmitters (UART)

Chapter 29
Fast Ethernet Controller (FEC)

Chapter 30
High-Speed USB On-The-Go (HS USB-OTG)

Chapter 31
General Purpose Timer (GPT)

Chapter 32
Pulse-Width Modulator (PWM)

Chapter 33
Real Time Clock (RTC)

Chapter 34
Watchdog Timer (WDOG)

Chapter 35
AHB-Lite IP Interface (AIPI) Module

Chapter 36
Multi-Layer AHB Crossbar Switch (MAX)

Chapter 37
Direct Memory Access Controller (DMAC)

Chapter 38
Digital Audio MUX (AUDMUX)

Chapter 39
CMOS Sensor Interface (CSI)

Chapter 40
Video Codec (Video_Codec)

Chapter 41
enhanced Multimedia Accelerator Light (eMMA_lt)

Chapter 42
Synchronous Serial Interface (SSI)

Chapter 43
Liquid Crystal Display Controller (LCDC)

Chapter 44
Smart Liquid Crystal Display Controller (SLCDC)

About This Book

The *MCIMX27 Multimedia Applications Processor Reference Manual* describes the features and operation of the i.MX27 microprocessor, the seventh generation of the DragonBall family of products. It provides the details of how to initialize, configure, and program the i.MX27 device. The manual presumes basic knowledge of ARM926EJ-S™ architecture.

Audience

The *MCIMX27 Multimedia Applications Processor Reference Manual* is intended to provide a design engineer with the necessary data to successfully integrate the i.MX27 processor into a wide variety of applications. It is assumed that the reader has a good working knowledge of the ARM926EJ-S processor. For programming information about the ARM926EJ-S processor, see the documents listed in the Suggested Reading section of this preface.

Organization

This reference manual is organized into two books:

1. Book I contains chapters that detail integration information, including the signals, clocks, power management, muxing tables, and JTAG/Boot operation of the IC.
2. Book II is divided into parts that consist of chapters that cover the operation and programming of the i.MX27 device.

Document Revision History

Since the last revision, Rev. 0.1, base addresses and memory addresses for some modules were updated for easier reader access.

Suggested Reading

The following documents are recommended for a complete description of the i.MX27 Multimedia Applications Processor, and enable proper design with the i.MX27 device. Especially for those not familiar with the ARM926EJ-S processor or previous DragonBall products, the following documents will be helpful when used in conjunction with this manual.

- *AMBA AHB specifications*, (ARM Ltd.)
- *ARM926EJ-S Platform specifications* (also named *ARM926 Platform*)
- *Hip7a KiloBit Single Port HP SRAM Compiler*, MEMCTC (May 8, 2002)
- *Hip7A SAMI ROM Compiler*, MEMCTC (November 16, 2001)
- *Hip7A KiloBit HD VIA ROM Compiler*, MEMCTC (June 28, 2002)
- *ARM926EJ-S Platform Test Guide* (ARM Ltd.)

- *ARM Architecture Reference Manual* (ARM Ltd., order number ARM DDI 0100)
- *ARM9DT1 Data Sheet Manual* (ARM Ltd., order number ARM DDI 0029)
- *ARM Technical Reference Manual* (ARM Ltd., order number ARM DDI 0151C)
- *MC9328MX1 i.MX Integrated Portable System Processor Reference Manual* (order number MC9328MX1RM)
- *MCIMX27 Multimedia Application Processor Data Sheet*—(order number MCIMX27))

These manuals can be found at the ARM Ltd. World Wide Web site at http://www.arm.com and Freescale Semiconductors World Wide Web site at http://www.freescale.com/imx. These documents can be downloaded directly from the World Wide Web site, or printed versions may be ordered. The World Wide Web site may also have useful application notes.

Conventions

This reference uses the following conventions:

- $\overline{\text{OVERBAR}}$ is used to indicate a signal that is active when pulled low: for example, $\overline{\text{RESET}}$.
- *Logic level one* is a voltage that corresponds to Boolean true (1) state.
- *Logic level zero* is a voltage that corresponds to Boolean false (0) state.
- To *set* a bit or bits means to establish logic level one.
- To *clear* a bit or bits means to establish logic level zero.
- A *signal* is an electronic construct whose state conveys or changes in state convey information.
- A *pin* is an external physical connection. The same pin can be used to connect a number of signals.
- *Asserted* means that a discrete signal is in active logic state.
  - *Active low* signals change from logic level one to logic level zero.
  - *Active high* signals change from logic level zero to logic level one.
- *Negated* means that an asserted discrete signal changes logic state.
  - *Active low* signals change from logic level zero to logic level one.
  - *Active high* signals change from logic level one to logic level zero.
- LSB means least significant bit or bits, and MSB means most significant bit or bits. References to low and high bytes or words are spelled out.
- Numbers preceded by a percent sign (%) are binary. Numbers preceded by a 0x are hexadecimal.

Definitions, Acronyms, and Abbreviations

The following list defines acronyms and abbreviations used in this document.

ADC	analog-to-digital converter
AFE	analog front end
API	application programming interface
BCD	binary coded decimal
BER	bit error ratio
CGM	clock generation module

CMOS	complimentary metal-oxide semiconductor
CRC	cyclic redundancy check
CSIC	complex instruction set computer
DAC	digital-to-analog converter
DDR RAM	double data rate RAM
DMA	direct memory access
DRAM	dynamic random access memory
DSP	digital signal processor
FEC	forward error correction
FIFO	first in first out
FIRI	fast IR interface
GPIO	general purpose input/output
I/O	Input/Output
ICE	in-circuit emulation
IrDa	infrared data association
JTAG	joint test action group
MAP	mold array process
MAPBGA	mold array process ball grid array
MIPS	million instructions per second
MMC	multimedia card
PLL	phase locked loop
PWM	pulse-width modulator
RTC	real-time clock
SD	secure digital
SDRAM	synchronous dynamic random access memory
SPI	serial peripheral interface
SRAM	static random access memory
TQFP	thin quad flat pack
UART	universal asynchronous receiver/transmitter
USB	universal serial bus
USB OTG	USB On-The-Go
XTAL	crystal
BE/LE	big endian/little endian
CCM	clock control module, also called “clkctl” module
LV	low voltage

LWB	late-write buffer
MCTL	memory controller
RAM	random access memory
ROM	read only memory
R-AHB bus	reduced advanced high-performance bus (AHB), related to ARM bus architecture
SRAM	static RAM
ARM	Advanced RISC Machines processor architecture
API	Application Programming Interface
Fabrication Path	Path within ROM Bootstrap for fabrication test execution
Flash Path	Path within ROM Bootstrap leading towards executing a Flash application.
GPCR	Global Peripheral Control Registry of the i.MX27.
HW	Hardware
iRAM	Processor-internal RAM
iROM	Processor-internal ROM
NFC	NAND Flash Controller
NAND Flash	A Flash ROM Technology
ROM Bootstrap	Internal boot code encompassing main boot flow as well as exception vectors, USB/UART Bootloader blocks.
RAM Path	Path within ROM Bootstrap leading towards downloading and executing a RAM application
SIDR	Silicon ID Register of the i.MX27
Sync Flash	A Flash ROM Technology
TBD	To Be Determined
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
V-Sync Flash	A Flash ROM Technology
Word	32 bits
TYPE	Identifier that distinguishes a production or engineering device.
UID	Unique ID; a field in the processor and CSF identifying a device or group of devices
WTLS	Wireless Transport layer Security, a part of the Wireless Application Protocol

Book I: i.MX27 Applications Processor Integration and Description

Introduction

Book I comprises detailed descriptions and information on the integration of the i.MX27 Multimedia Applications Processor. Book I includes the following chapters.

Device Introduction and Memory Map

Clocks, Power Management and Reset

Pins

Debug

Boot

Chapter 1
Introduction to the i.MX27 Multimedia Applications Processor

As part of the i.MX growing family of multimedia-focused products, the i.MX27 Multimedia Applications Processor takes the Mobile Multimedia Experience to another level. Whether you are designing mobile entertainment, a smartphone, wireless PDA, or any other portable device, the i.MX27 processor offers a high degree of integration to significantly reduce your design time while providing low-power consumption with performance to spare, and the flexibility necessary for today's competitive marketplace.

The i.MX27 processor is packaged in a 404-pin MAPBGA.

Differentiating features of the i.MX27 device include:

- Advanced and power-efficient implementation of the ARM926EJ-S™ core, operating at speeds up to 400 MHz
- *enhanced* Multimedia Accelerator Lite (eMMA_lt)—MPEG-4 and H.264 hardware encode or decode up to D1 resolution at 30 fps or encode and decode up to VGA resolution at 24 fps.
- High-Speed USB On-The-Go controller, host or client
- Smart Speed Crossbar Switch—Multi-layer AMBA-compliant bus allows any one of the six bus masters to talk to any of the three slaves without interfering with the other bus master/slave transactions to provide system level parallelism.
- PCMCIA/Compact Flash Interface—Supports hot-insertion, card insert, and removal detection
- Security—Software and hardware combined security solution allows secure e-commerce, digital rights management (DRM), information encryption, secure boot, and secure software downloads.
- Smart Power Management—Includes Run, Doze, and Sleep modes, frequency scaling, active well biasing and clock gating
- LCD panels—Supports both smart and standard LCD panels
- Fast Ethernet—Supports 10/100 baseT Ethernet MAC

1.1 i.MX27 Applications Processor Block Diagram

Figure 1-1 is a simplified functional block diagram of the i.MX27 processor.

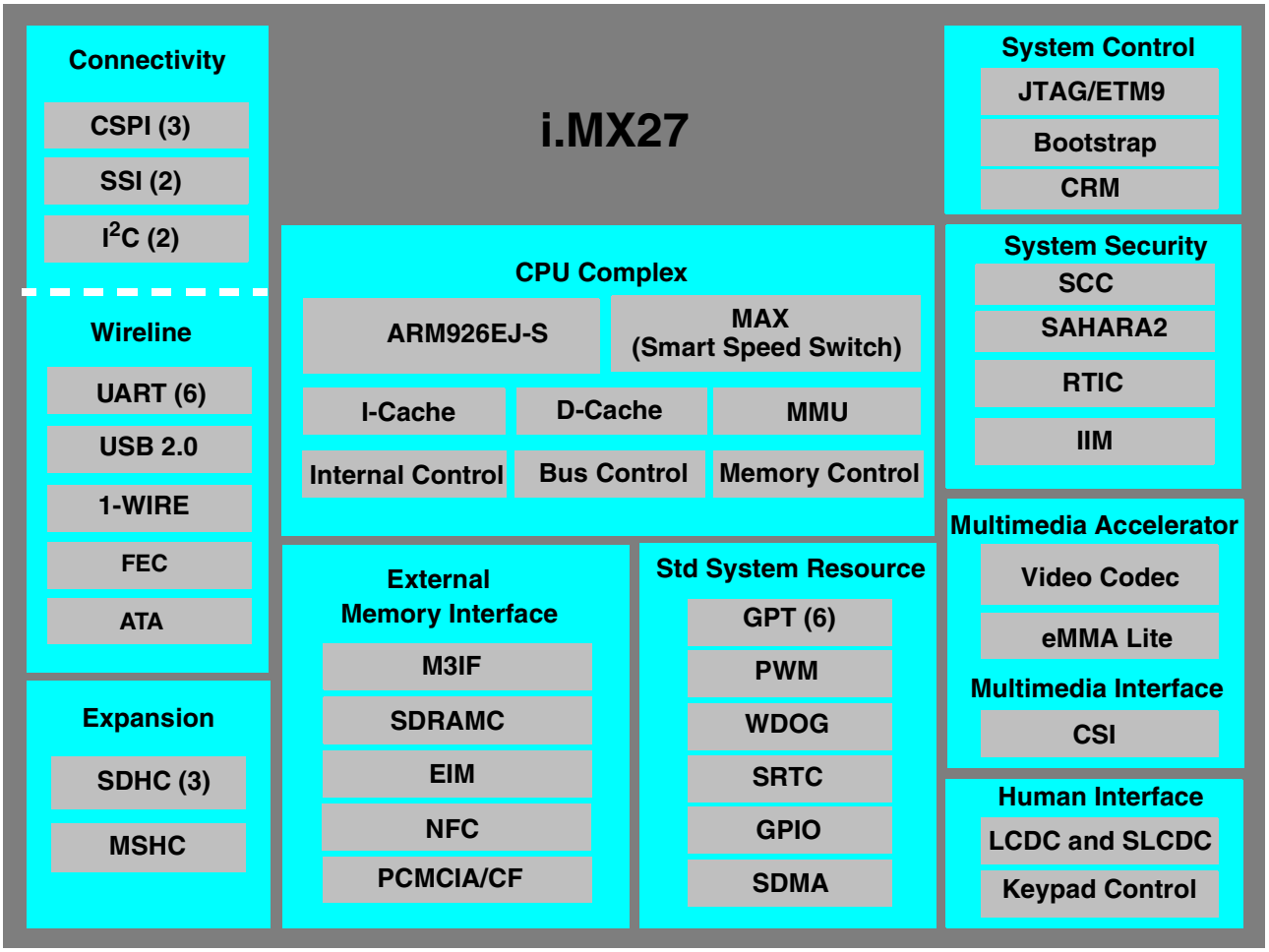


Figure 1-1. i.MX27 Processor Functional Block Diagram

1.2 Summary of Core and Modules

This section describes the ARM926EJ-S as it applies to the i.MX27 processor and the function of the modules within the i.MX27 device.

1.2.1 ARM9™ Platform

The ARM9™ Platform consists of the ARM926EJ-S core, operating at speeds up to 400 MHz at 1.6 V, and 266 MHz at 1.2 V. The ARM926EJ-S core includes a 16-Kbyte level 1 (L1) cache system, a 6×3 multi-layer AHB crossbar switch, and a 16 channel DMA.

The ARM926EJ-S is a member of the ARM9 family of general-purpose microprocessors targeted at multi-tasking applications. The ARM9 Platform provides the following features:

- ARM926EJ-S microprocessor core
  - 16K instruction cache and 16K data cache
  - High-performance ARM® 32-bit RISC engine
  - Thumb® 16-bit compressed instruction set for a leading level of code density

  - Efficient execution of Java byte codes
  - EmbeddedICE™ JTAG software debug
  - 100% user code binary compatibility with ARM7TDMI™
  - Advanced Microcontroller Bus Architecture (AMBA™) system-on-chip multi-master bus interface
  - Support for mixed loads of real-time and user applications via cache locking facilities
  - Virtual Memory Management Unit (VMMU)
- Support for Little Endian only
- CPU and System speed
  - ARM926EJ-S core: up to 400 MHz
  - System Clock: up to 133 MHz
  - External memory interface: same clock source as system, up to 133 MHz at 1.8 V supply
  - System clock is derived from the CPU clock through an integer divider
- ARM Interrupt Controller (AITC)
  - The AITC is connected to the primary AHB as a slave device and provides support for up to 64 interrupt sources. It generates normal and fast interrupts to the processor core. The AITC supports a hardware-assisted vectoring mode for automatic vectoring to reduce interrupt latency.
- Clock Control Module (CLKCTL)—The CLKCTL performs block level clock gating, ARM926EJ-S JTAG synchronization requirements, as well as other miscellaneous clock control for the platform.
- AHB to IP bus interfaces (AIPIs)—Provide a communication interface between the high-speed AHB to a lower-speed IP bus for slave peripherals
- The Multi-Layer 6×3 AHB Crossbar Switch (MAX)—The crossbar switch allows for concurrent transactions to proceed from any input port (bus master) to any output port (bus slave): That is, it is possible for all three output ports to be active at the same time as a result of three independent input or output requests.
- Well Bias Charge Pump (WBCP)—With the exception of the memories, the entire ARM9 Platform supports two active well biasing to reduce leakage current to minimum levels. The well bias enable inputs (wt_en and wt_en_dnw) are driven by the external Well Bias Charge Pump (WBCP) to the ARM9 Platform.

1.2.2 System Control

To ensure optimum power use and clock signal stability, the i.MX27 processor uses the following modules to generate, control, and distribute clock and control signals throughout the i.MX27 processor and to external devices.

1.2.2.1 Clock Controller Module (CCM)

The CCM generates clock and reset signals used throughout the i.MX27 device and for external peripherals. It also enables system software to control, customize, or read the status of the following functions:

- Chip ID
- Multiplexing of I/O signals
- I/O driving strength
- I/O pull enable control
- Well bias control
- System boot mode selection
- DPTC control

1.2.2.2 JTAG Controller (JTAGC)

The JTAGC provides debug access to the ARM926EJ-S core and boundary scan test control. The i.MX27 processor offers designers and programers with full-debug capabilities through industry-standard JTAG interface and the ability to bootload using either a serial or USB interface.

- UART Bootstrap mode function:
  - Allows system initialization and program or data download to system memory via USB or UART1
  - Accepts execution command to run program stored in system memory
  - Supports memory/register read/write operation of selectable data size of byte, half-word, or word
  - Provides a 16-byte instruction buffer for ARM instruction storage and execution
- USB Bootstrap mode function
  - Supports bootstrapping through USB OTG port
- JTAG port to support generic ARM debug tools

1.2.3 Standard System Resources

The i.MX27 processor contains various timers and resource features to optimize the control and security of both the internal modules and external devices.

1.2.3.1 General Purpose Timer (GPT)

The six General-Purpose Timer (GPT) modules contain identical general-purpose 32-bit timers with programmable prescalers and compare and capture registers with the following features:

- Automatic interrupt generation
- Programmable timer input/output pins
- Input capture channels capability with programmable trigger edge for each GPT
- Output compare channels with programmable mode for each GPT

1.2.3.2 Pulse-Width Modulator (PWM)

The Pulse-Width Modulator (PWM) has a 16-bit counter and is optimized to generate sound from stored sample audio images. It also generate tones.

The following features characterize the PWM module:

- 4×16 FIFO to minimize interrupt overhead
- 16-bit resolution
- Sound and melody generation

1.2.3.3 Real Time Clock (RTC)

The Real-Time Clock (RTC) module maintains the system clock, provides stopwatch, alarm, and interrupt functions, and supports the following features:

- 32.768 kHz and 32 kHz input operation
- Full clock features: seconds, minutes, hours, days
- Capable of counting up to 512 days
- Minute countdown timer with interrupt
- Programmable daily alarm with interrupt
- Sampling timer with interrupt
- Once-per-second, once-per-minute, once-per-hour, and once-per-day interrupts
- Interrupt generation for digitizer sampling or keyboard debouncing
- Independent power supply

1.2.3.4 Watchdog Timer Module (WDOG)

The Watchdog Timer module (WDOG Timer) module protects against system failures by providing a method for the system to recover from unexpected events or programming errors. The WDOG timer module also generates a system reset using a software write to the Watchdog Control Register (WCR), a detection of a clock monitor event, an external reset, an external JTAG reset signal, or an occurrence of a power-on reset.

The WDOG Timer provides the following:

- Programmable time out of 0.5 s to 64 s
- Resolution of 0.5 s

1.2.3.5 General-Purpose I/O Ports (GPIO)

The GPIO module provides six general purpose I/O ports. Each single GPIO port is a 32-bit port that may be multiplexed with one or more dedicated functions. The GPIO features are:

- Supports level or edge trigger interrupt and is system wake-up capable
- Most I/O signals are multiplexed with dedicated functions for pin efficiency.

1.2.3.6 Direct Memory Access Controller (DMAC)

The Direct Memory Access Controller (DMAC) provides 16 channels to support linear memory, 2D memory, FIFO, and end-of-burst enable FIFO transfers to support a wide variety of DMA operations. Features include the following:

- Supports 16 channels linear memory, 2D memory, and FIFO for both source and destination
- Supports 8-bit, 16-bit, or 32-bit FIFO port size and memory port size data transfer
- DMA burst length is configurable up to maximum of 16 words, 32 half-words, or 64 bytes for each channel
- Bus utilization control for a channel that is not triggered by DMA request
- Interrupts that are provided to interrupt handler on bulk data transfer complete or transfer error
- DMA burst time-out error to terminate DMA cycle when the burst cannot be completed in a programmed timing period
- Dedicated external DMA request and grant signal
- Supports increment, decrement, and no increment for source and destination addressing
- Supports DMA chaining

1.2.4 Power Management and Backup Modes

The i.MX27 processor’s power management features are as follows:

- SupportS 3 power modes of operation: Run, Doze, and Stop
- Aggressive clock gating within modules to minimize CMOS switching power
- Active well biasing technique to reduce standby mode current consumption
- Voltage/frequency scalable capability
- Dynamic process temperature compensation

1.2.4.1 SCC, RTC, and Oscillator Power Supply

The i.MX27 processor has a separate power domain from the main power domain for the SCC, RTC, and the 32 kHz oscillator (OSC32K) power supply, so that when the main power domain shuts down, the SCC

internal memory data and status is maintained. Also, the RTC and OSC32K works as normal using the backup power supply that is provided by power management.

1.2.4.2 Enter/Exit Mode

Power management provides the power_cut to indicate the main power cut: 1—main power_cut; 0—main power_on.

- Mode Enter—When power_cut is set to 1 by the power management chip, then the main power can shut down.
- Mode Exit—power_cut must be set to 1 when the main power is not on; after main power_on is restored (after the main power reset ends), power_cut should be set to 0.
- Power On Initial—In the initial process, power_cut must be set to 0, and $\overline{\text{POR}}$ must give a valid 0 slot for the SCC, RTC power_on reset.

1.2.4.3 Reset Strategy

The $\overline{\text{POR}}$ is active to the RTC and SCC. When power_cut is set to 1 (in the backed power mode), then the reset from the chip system will be gated, preventing any chip reset sources from resetting SCC and RTC. When power_cut is set to 0 (not in the backed power mode), then the reset from the chip system will be active to SCC and RTC, which means all chip reset sources can reset the SCC and RTC.

1.2.5 System Security

To address the need for secure wireless communication, the i.MX27 processor provides confidentiality, authentication, integrity, and legitimacy within its architecture. This section describes the modules that provide these types of security—the Security Controller, SAHARA2, Run-Time Integrity Checker, and the IC Identification Module—whose built-in features support a broad range of security-enabled products.

1.2.5.1 Security Controller Module (SCC)

The SCC is a hardware component composed of two blocks—the Secure RAM module and the Security Monitor. The Secure RAM securely stores sensitive information. The Security Monitor implements the security policy, checking algorithm sequencing, and controlling the Secure State. There is also a unique encryption key accessible only to secure RAM.

1.2.5.2 Symmetric/Asymmetric Hashing and Random Accelerator (SAHARA2)

SAHARA2 is a security co-processor within the i.MX27 processor used to implement block encryption algorithms, hashing algorithms, stream cipher algorithms, and hardware random number generation.

SAHARA2 accelerates the following security protocols and their features:

- AES encryption/decryption
  - — ECB, CBC, CTR, and CCM modes
  - — 128 bit key
- DES/3DES

  - EBC, CBC, and CTR modes
  - 56 key with parity (DES)
  - 112 bit or 168 bit key parity (3DES)
- AR4 (RC4 compatible cipher)
  - 5–16 byte key
  - Host accessible S-box
- MD5, SHA-1, SHA-224 and SHA-256 hashing algorithms
  - Message lengths are multiples of bytes
  - Auto padding supported
  - HMAC (support for IPAD and OPAD via descriptors)
  - Up to 4-Gbyte message length
- Random number generator (NIST approved PRNG – FIPS 186-2)
  - Entropy is generated via an independent free running ring oscillator

1.2.5.3 Run-Time Integrity Checkers (RTIC)

The RTIC ensures the integrity of the contents of the peripheral memory and assists with boot authentication.

The RTIC offers the following features:

- SHA-1 message authentication
- Input DMA (AMBA-AHB Lite[1] bus master) interface
- Segmented data gathering to support non-contiguous data blocks in memory (up to two segments per block)
- Works with High Assurance Boot (HAB) process
- Secure-scan DFT security
- Support for up to four independent memory blocks
- Programmable DMA bus duty cycle timer and watchdog timer
- Power-saving clock gating logic
- Hardware configurable Big/Little-Endian data format
- Full word memory reads (word-aligned addresses, multiple of 32-bit lengths)

1.2.5.4 IC Identification Module (IIM)

The IC Identification Module (IIM) provides an interface for reading, and in some cases, programming, and for overriding identification and control information stored in on-chip fuse elements.

1. AHB-Lite interface provides support for request/grant bus arbitration

1.2.6 Connectivity

This section describes how the modules within the i.MX27 processor interface with each other, and provides a high-level overview on how the architecture of the buses are configured and multiplexed.

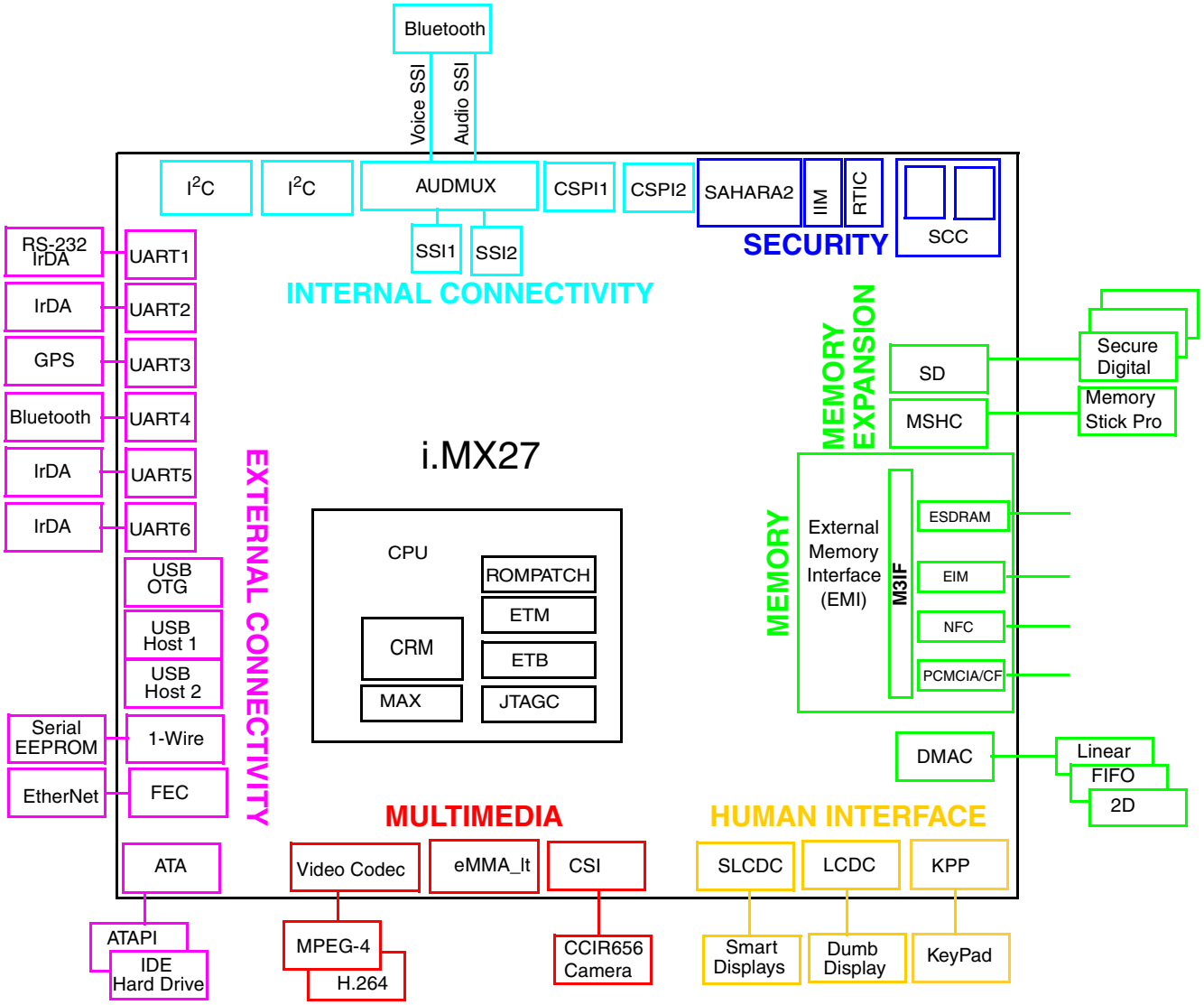


Figure 1-2. i.MX27 Connectivity Example

1.2.6.1 Configurable Serial Peripheral Interfaces (CSPI)

The i.MX27 processor has three Configurable Serial Peripheral Interface (CSPI) modules that allow rapid data communication with fewer software interrupts than conventional serial communications. Each CSPI is equipped with data FIFO and is a master/slave configurable serial peripheral interface module, enabling the i.MX27 processor to interface with external SPI master or slave devices.

- Master/slave configurable
- Two chip-selects each for master mode operation

- Up to 16-bit programmable data transfer
- 8 × 16 FIFO for both transmit and receive data

1.2.6.2 Inter-IC Connectivity (I^2C) Bus Module

The two I^2C modules are two-wire, bidirectional serial buses that provide a simple, efficient method of data exchange, minimizing the interconnection between devices. These buses are suitable for applications requiring occasional communications over a short distance between several devices. The flexible I^2C allows additional devices to be connected to the bus for expansion and system development. The I^2C features include:

- Multiple-master operation
- Software-programmable for 1 of 64 different serial clock frequencies
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation and detection
- Repeated START signal generation
- Acknowledge bit generation and detection
- Bus-busy detection

1.2.6.3 Synchronous Serial Interface (SSI)

The two synchronous serial interfaces are full-duplex, serial ports that enable the i.MX27 device to communicate with a variety of serial devices, such as standard codecs, digital signal processors (DSPs), microprocessors, peripherals, and popular industry audio codecs that implement the inter-IC sound bus standard (I^2S) and Intel AC97 standard.

Features include the following:

- Supports generic SSI interface for timeslot based communication with synchronous voice codecs
- Timeslot mode supports up to four channels for communication among devices, Bluetooth™ voice port, voice codecs, and baseband audio ports.
- Supports Philips standard Inter-IC Sound (I^2S) bus for external digital audio chip interface at 44.1 kHz and 48 kHz
- AC97 Host Controller mode with support for two audio channels supporting fixed and variable rate transfers
- Used together with the Digital Audio Mux (AUDMUX) module to provide flexible audio and voice routing options

1.2.6.4 Bus Control

The six modules that control the bus in the MX27 are listed here. This section provides a brief description for each.

- AHB-Lite IP Interface Module (AIPI)

- ARM926EJ-S Interrupt Controller (AITC)
- Intellectual Property Bus Multiplexer (IPMUX)
- Multi-layer AHB Crossbar Switch (MAX)

1.2.6.4.1 AHB-Lite IP Interface Module (AIPI)

The AIPI acts as an interface between the ARM Advanced High-performance Bus Lite. (AHB-Lite) and lower bandwidth peripherals that conform to the *IP Bus Specification, Rev 2.0*.

1.2.6.4.2 ARM926EJ-S Interrupt Controller (AITC)

AITC is connected to the primary AHB as a slave device. It generates the normal and fast interrupts to the ARM926EJ-S processor.

1.2.6.4.3 Intellectual Property Bus Multiplexer (IPMUX)

The Intellectual Property Bus Multiplexer (IPMUX) is used to select the read data, transfer wait, and transfer error signals from the various modules and pass it to the AIPI in the ARM9 Platform.

1.2.6.4.4 Multi-Layer AHB Crossbar Switch (MAX)

The ARM926EJ-S core's instruction and data buses and all alternate bus master interfaces arbitrate for resources via a 6×3 Multi-Layer AHB Crossbar Switch (MAX)—also known as a Smart Speed Switch. There are six fully functional master ports (M0–M5) and three fully functional slave ports (S0–S2). The MAX is uni-directional. All master and slave ports are AHB-Lite compliant.

1.2.7 Wireline Connectivity

The i.MX27 device provides a variety of external wireline connectivity. This section describes the modules for this support.

1.2.7.1 Universal Asynchronous Receiver/Transmitter (UART)

The i.MX27 processor includes six Universal Asynchronous Receiver/Transmitter (UART) modules that provide serial communication with external devices through either an RS-232 cable or by using IrDA-compatible infrared.

Each of the six UARTs features the following:

- Supports serial data transmit/receive operation: 7 or 8 data bits, 1 or 2 stop bits, programmable parity (even, odd, or none)
- Programmable baud rates up to 1.875 MHz
- Automatic baud rate detection
- 32-bytes FIFO for transmit and 32 half-words FIFO for receive data
- IrDA Serial Infra-Red (SIR) mode support

1.2.7.2 High Speed USB 2.0 Interface (USB)

The i.MX27 processor supports three independent USB 2.0 ports, two of which support high speed (HS) operation:

- OTG—High speed (480 Mbps)
- Host 1—High speed (480 Mbps)
- Host 2—Full speed (12 Mbps)

The USB connectivity of the i.MX27 processor provides extremely fast synchronization with a PC or between two devices. Any of the USB ports may be used for transceiver-free connection or for external transceiver-based connection.

The USB OTG port can connect to a PC as either a device or as a host to any of the following peripherals: keyboard, printer, mouse, speakers, storage device, digital camera, and so on. It supports 16 endpoints for each host and device.

USB Host 1 is typically connected to dedicated ICs that support WLAN, Bluetooth™ wireless technology, and GPS. Host 1 supports 16 endpoints. USB Host 2 is typically used to connect to ICs for baseband or WLAN, Bluetooth wireless technology, or GPS. Host 2 supports four endpoints.

1.2.7.3 1-Wire Interface (1-Wire)

The 1-Wire® module provides bi-directional communication between the ARM926EJ-S and the Add-Only-Memory EPROM (DS2502). The 1-Kbit EPROM is used to hold information about the battery and to communicate with the ARM9 Platform using the IP interface.

1.2.7.4 Advanced Technology Attachment (ATA)

The Advanced Technology Attachment (ATA) block of the i.MX27 processor is an AT attachment host interface and is used to interface with IDE hard disk drives and ATAPI optical disk drives. This feature allows designers to attach storage devices at low costs per unit, which is a critical selling point in the portable digital player market. The ATA controller interfaces with ATA devices using the industry-standard ATA-6 specification.

The ATA interface is compliant to the ATA-6 standard, and supports following protocols:

- PIO mode 0, 1, 2, 3, and 4
- Multiword DMA mode 0, 1, and 2
- Ultra DMA modes 0, 1, 2, 3, and 4 with bus clock of 50 MHz or higher
- Ultra DMA modes 5 with bus clock of 80 MHz or higher

1.2.7.5 Fast Ethernet Controller (FEC)

The Fast Ethernet Controller (FEC) performs the full set of IEEE 802.3/Ethernet CSMA/CD media access control and channel interface functions. The FEC supports connection and functionality for the 10/100

Mbps 802.3 Media Independent Interface (MII). It requires an external transceiver (PHY) to complete the interface to the media. The FEC provides the following features:

- Supports three different Ethernet physical interfaces:
  - 100 Mbps IEEE 802.3 MII
  - 10 Mbps IEEE 802.3 MII
  - 10 Mbps 7-wire interface (industry standard)
- IEEE 802.3 full-duplex flow control
- Programmable maximum frame length supports IEEE 802.1 VLAN tags and priority
- Supports full-duplex operation (200 Mbps throughput) with a minimum system clock rate of 50 MHz
- Supports half-duplex operation (100 Mbps throughput) with a minimum system clock rate of 25 MHz
- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
- Address recognition
  - Frames with broadcast address may be always accepted or always rejected
  - Exact match for single 48-bit individual (unicast) address
  - Hash (64-bit hash) check of individual (unicast) addresses
  - Hash (64-bit hash) check of group (multicast) addresses
  - Promiscuous mode

1.2.8 External Memory Interface

The External Memory Interface (EMI) of the i.MX27 processor consists of the SDRAM controller (SDRAMC), the PCMCIA controller, the NAND Flash controller (NFC), and the External Interface module (EIM), using the Multi-Master Memory Interface (M3IF) as the controller through the external memory ports. The individual features of these controllers are provided in this section.

To allow the maximum number of potential designs, the EMI supports the following memory types:

- SDRAM—133 MHz, 32/16-bit
- DDR—266 MHz, 32/16-bit
- NAND Flash—dedicated 8-bit, shared 16-bit
- PSRAM

1.2.8.1 Multi-Master Memory Interface (M3IF)

The Multi-Master Memory Interface (M3IF) controls memory accesses from one or more masters through different port interfaces to the external memory controllers SDRAM, PCMCIA, NAND Flash, and EIM.

The M3IF includes these distinctive features:

- Supports multiple requests from masters through input port interfaces

- Arbitrates requests to the different memory controllers
- Multiple requests capabilities to SDRAMC through a dedicated arbitration mechanism
- Flexible round robin access arbitration, with a programmable priority scheme to selective masters
- Programmable master that controls (locks) accesses to SDRAM/DDR, and a programmable master that controls (locks) accesses to other memories (NFC, EIM)
- Multi-endianness support for all memory controllers
- Supports memory "snooping"—that is, monitors a region in external memory for write accesses

1.2.8.2 SDRAM Controller (SDRAMC)

The SDRAM Controller (SDRAMC) provides an interface and control for synchronous DRAM memories for the system. The SDRAMC supports the following:

- Optimization of consecutive memory accesses using memory command anticipation (latency hiding)
  - Hiding latency (or "command anticipation") by optimizing the commands to both connected chip-selects
  - Monitoring open memory pages
  - Bank-wise memory address mapping
  - SDRAM burst length configuration of 4[1] or 8 bursts or full-page mode
  - MDDR burst length configuration of 8 bursts
  - Support of different internal burst length (1/4/8 words) by using burst truncate commands
  - ARM/AMBA/AHB-Lite compliant
  - Shared address and command bus to SDRAM/MDDR
- Supports 64, 128, 256, 512 Mbit, 1 Gbit, and 2 Gbit, 4 bank, single data rate, synchronous SDRAM, and MDDR
  - Two independent chip-selects
  - Up to 128 Mbytes per chip-select
  - Up to four banks active simultaneously per chip-select
  - JEDEC standard pinout/operation
- Supports mobile DDR266 devices (both 16-bit and 32-bit)
- PC133 compliant interface
  - 133-MHz system clock achievable with "–7" option PC133 compliant memories
  - Single fixed-length (4/8-word) burst or full page access
  - Access time of 9-1-1-1-1-1-1-1 at 133 MHz (for read access when the memory bus is available, the row is open and CAS latency configured to three cycles). The access time includes the M3IF delay (assuming there is no arbitration penalty).
- Software configurable for different system and memory devices requirements
  - 16-bit or 32-bit memory data bus width

1. For 16-bit memory burst length 4 is not supported.

  - Many row and column addresses
  - Row cycle delay (tRC)
  - Row precharge delay (tRP)
  - Row-to-column delay (tRCD)
  - Column-to-data delay (CAS latency)
  - Load mode register to active command (tMRD)
  - Write to precharge (tWR)
  - Write to read (tWTR) for MDDR memories only
  - MDDR exit power down to next valid command delay (tXS)
  - Active to precharge (tRAS)
  - Active to active (tRRD)
- Built-in auto-refresh timer and state machine
- Hardware and software supported self-refresh entry and exit
  - Keeps data valid during system reset and low-power modes
  - Auto Power Down timer (one per chip-select)
  - Auto Precharge timer (one per bank in each chip-select)

1.2.8.3 NAND Flash Controller (NFC)

The NAND Flash controller (NFC) interfaces standard NAND Flash memory devices to the i.MX27 processor and hides the complexities of accessing NAND Flash. The NFC features include:

- Contains hardware boot loader for automatic boot up from NAND Flash devices
- Supports all 8-bit/16-bit NAND Flash devices regardless of density and organization
- Supports 512-byte and 2-Kbyte page sizes
- Internal 2 Kbytes of buffer RAM used as boot RAM during cold startup and as read/write page buffers to relieve CPU intervention
- Automatic ECC detection and selectable correction
- Data protection for RAM buffer and NAND Flash pages

1.2.8.4 Personal Computer Memory Card International Association (PCMCIA)

The PCMCIA host adapter module provides the control logic for PCMCIA socket interfaces, and requires some additional external analog power switching logic and buffering.

The PCMCIA controller provides the following features:

- A host adapter interface fully compliant with the PCMCIA standard release 2.1 (PC Card -16)
  - Supports one PCMCIA socket
  - Supports hot-insertion
  - Supports card detection

— Mappings to common memory space, attribute memory space, and I/O space. Each space is up to 64 Mbytes in size.
— Supports five memory windows
— Generates a single interrupt to the CPU
— PC card access timing is fully programmable
— Handles interrupts from the card

- The pcmcia_if signal is part of the EMI complex and shares pins with the EIM, SDRAMC, and NFC controller.
- Supports ATA disk emulation

1.2.8.5 External Interface Module (EIM)

The External Interface Module (EIM) interfaces to devices external to the chip, including generation of chip-selects, clock and control for external peripherals, and memory. The EIM provides asynchronous and synchronous access to devices with an SRAM-like interface.

The EIM includes the following features:

- Six chip-selects for external devices, with $\overline{CS}$ [0] and $\overline{CS}$ [1] each covering a range of 128 Mbytes, and $\overline{CS}$ [2] – $\overline{CS}$ [5], each covering a range of 32 Mbytes
- $\overline{CS}$ [0] range can be increased to 256 Mbytes when collapsed with $\overline{CS}$ [1]
- Selectable protection for each chip-select
- Programmable data port size for each chip-select
- Asynchronous accesses with programmable setup and hold times for control signals
- Synchronous Memory Burst Read Mode support for AMD, Intel, and Micron burst flash memory
- Synchronous Memory Burst Write Mode support for PSRAM (CellularRAM™ from Micron, Infineon, and Cypress)
- Support for multiplexed address/data bus operation
- External cycle termination/postpone with $\overline{DTACK}$ signal
- Programmable wait-state generator for each chip-select
- Support for Big Endian and Little Endian modes of operation per access
- ARM AHB slave interface

1.2.9 Memory Expansion

The i.MX27 processor offers memory expansion options for SD, Memory Stick Pro®, and ATA-6. Each expansion port reflects the latest version of the respective specification for that interface. Brief descriptions of each expansion port follow.

1.2.9.1 Memory Stick Host Controller (MSHC)

The i.MX27 processor's Memory Stick Host Controller supports one Memory Stick Pro slot. The MSHC conforms to *Memory Stick Standard Format Specifications,* ver.1.4-00 and *Memory Stick Standard*

Memory Stick PRO Format Specification, ver.1.00-01. The MSHC communication is based on an advanced 7-pin serial bus designed to operate in a low-voltage range. In addition to multimedia cards, the module can be used to communicate to high-bit rate communication devices, such as WLAN 802.11 a/b, and Bluetooth wireless technology, among others. The MSHC is placed between the AIPI and the customer memory stick to support data transfer from the i.MX27 device to the customer memory stick.

1.2.9.2 Secured Digital Host Controller (SDHC)

The three Secured Digital Host Controllers (SDHC) in the i.MX27 device control the Secure Digital memory cards and I/O functions by sending commands to the cards and performing data accesses to and from the cards.

SDHC features include:

- Fully compatible with the *SD Memory Card Specification 1.0* and *SD I/O Specification 1.0* with 1 and 4 channel(s)
- Supports hot swappable operation
- Data rates from 25 Mbps to 100 Mbps
- Dedicated power pin

1.2.10 Video Codec and *enhanced* Multimedia Accelerator Lite (eMMA_lt)

The i.MX27 processor uses a Video Codec and an *enhanced* Multimedia Accelerator Lite (eMMA_lt) to provide H.264, MPEG-4 and H.263 hardware acceleration with pre- and post-processing.

1.2.10.1 Video Codec

The Video Codec module supports full-duplex video codec with MPEG-4 and H.264 hardware encode or decode up to D1 resolution at 30 fps or encode and decode up to VGA resolution at 24 fps, and integrates multiple video processing standards, such as H.264 BP, MPEG-4 SP, and H.263 P3. The Video Codec architecture is shown in the Figure 1-3.

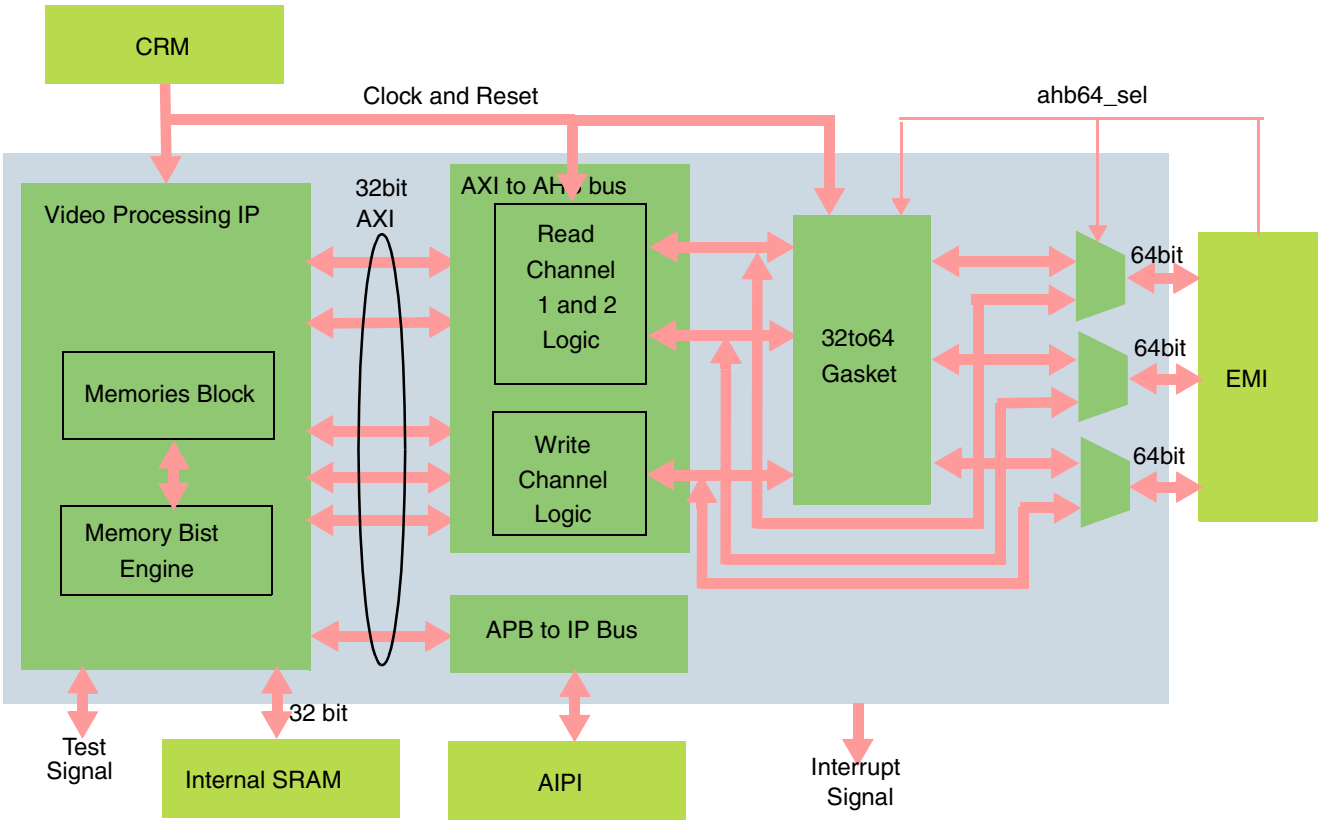


Figure 1-3. Video Codec Architecture Diagram

The Video Codec provides the following capabilities:

- Multi-standard video codec
  - MPEG-4 part-II Simple Profile (SP) encoding/decoding
  - H.264/AVC Baseline Profile (BP) encoding/decoding
  - H.263 P3 encoding/decoding
  - Multi-party call: One stream encoding and two streams decoding simultaneously
  - Multi-format: Encodes MPEG-4 bitstream, and decodes H.264 bitstream simultaneously
- Coding tools
  - High-performance motion estimation (single reference frame for both MPEG-4 and H.264 encoding)
    - Quarter-pel and half-pel accuracy motion estimation
    - [±16, ±16] search range
  - All variable block sizes are supported. (In case of encoding, 8×4, 4×8, and 4×4 block sizes are not supported.)
  - Unrestricted motion vector
  - MPEG-4 AC/DC prediction and intra-prediction (H.264)
  - H.264/AVC intra-prediction

  - — H.263 Annex I, J, K, and T are supported
  - — Error resilience tools
    - – MPEG-4 resync. marker and data-partitioning with RVLC (fixed number of bits/macroblocks between macroblocks)
    - – H.264/AVC FMO and ASO
    - – H.263 slice structured mode
    - – Bit-rate control (CBR and VBR)
- Pre/post rotation/mirroring
  - — 8 rotation/mirroring modes for image to be encoded
  - — 8 rotation/mirroring modes for image to be displayed
- Programmability
  - — Embeds C and M proprietary 16-bit DSP processor that is dedicated to processing bitstream and driving the codec hardware
  - — General purpose registers and interrupt for communication between internal host processor and Video Codec IP
- Performance
  - — Up to full-duplex VGA 24 fps encoding/decoding
  - — Up to half-duplex SD 30 fps encoding/decoding

1.2.10.2 *enhanced* Multimedia Accelerator Lite (eMMA_lt)

The i.MX27 processor comes with an *enhanced* Multimedia Accelerator Lite (eMMA_lt), which comprises independent pre-processing and post-processing stages that provide exceptional image and video quality. The eMMA_lt represents a major breakthrough to solve the problem of high MIPS required for video encode and decode operations in mobile and wireless applications. Tight integration and memory pipelining coupled with AHB master mode operation ensure minimal system loading. To further offload the CPU, live video stream data enters the eMMA_lt module directly through an internal private data interface.

The i.MX27 processor's eMMA_lt features the following:

- Enables simultaneous MPEG-4 Simple Profile (SP) video encoding and decoding
- Supports real-time video decode in any of the following advanced formats:
  - — MPEG-4 Simple Profile (SP)
  - — H.264
- Provides video and image data pre/post-processing (resizing, color conversion, filtering) that is fully hardware accelerated

The eMMA_lt architecture is shown in Figure 1-4.

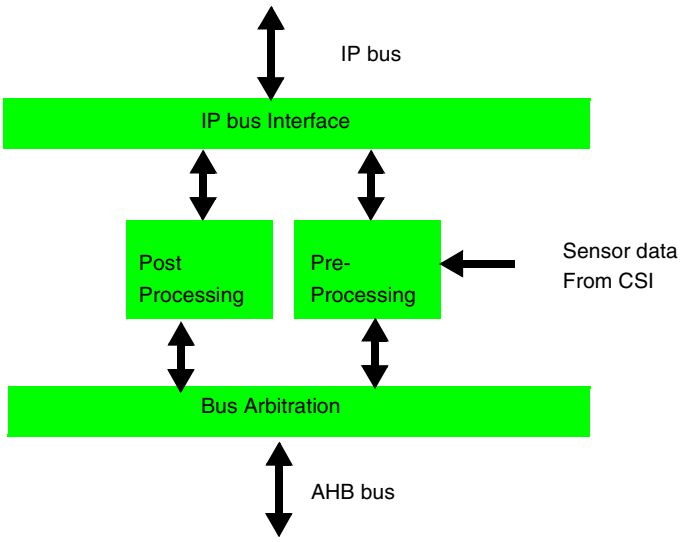


Figure 1-4. eMMA_It Architecture

1.2.10.2.1 Image Pre-Processor (PrP)

The image Pre-Processor block performs color space conversion and image resizing for the viewfinder display, and data formatting for the video encoder. It also performs color space conversions of the still image for input to either a hardware- or software-based video encoder or image compressor. The Pre-Processor has two media input and output paths and can accept input from system memory or from a private data bus connected to the CMOS Sensor Interface (CSI) module. The Pre-Processor can apply frame rate control on the live video stream from the CSI module to adjust for different processing load conditions. The Pre-Processor's two output channels are used to output RGB data for display of the local camera view and to output image data for compression by the hardware encoder or a software encoder (still image or video encode). Figure 1-5 shows the image Pre-Processor.

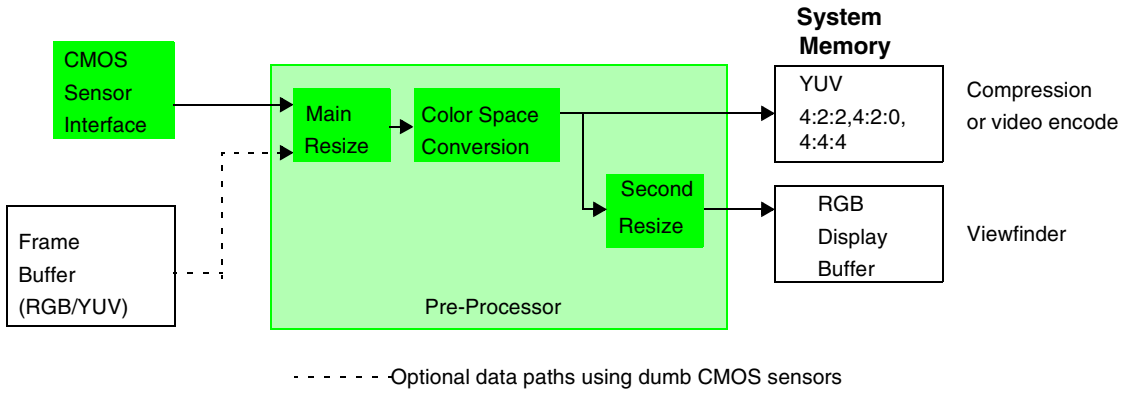


Figure 1-5. Pre-Processor Data Flow

Pre-Processor features:

- Data input:
  - System memory
  - Private DMA between CMOS Sensor Interface module and Pre-Processor

- Data input formats:
  - Arbitrarily unpacked RGB input
  - YUV 4:2:2 (Interleaved)
  - YUV 4:2:0 (Planar)
- Input image size: 2044 × 2044
- Image scaling:
  - Main resize ratio: 8:1–1:1 in integral steps, Horizontal 9:8/vertical 6:5 and Horizontal 9:8/Vertical 1:1
  - Secondary resize ratio for viewfinder: 8:1–1:1 in integral steps
- Output data format:
  - RGB565
  - YUV 4:2:2 (Interleaved)
  - YUV 4:2:0 (Planar)
- RGB data and one YUV data format can be generated concurrently

1.2.10.2.2 Post-Processor (PP)

The Post-Processor performs Deblock, Dering, Image Resize, and Color Space Conversion (CSC) functions on the input image data. These functions provide flexibility to meet various RGB formats and YUV formats for display. In addition to working in tandem with the decoder sub-block in the eMMA_lt, the Post-Processor can also be used by software decoders (other than MPEG-4) to touch up the final output before display. The sub-blocks that perform Deblock, Dering, Resize, and CSC operations can be selectively bypassed through software configuration. Figure 1-6 shows the flow for video postprocessing.

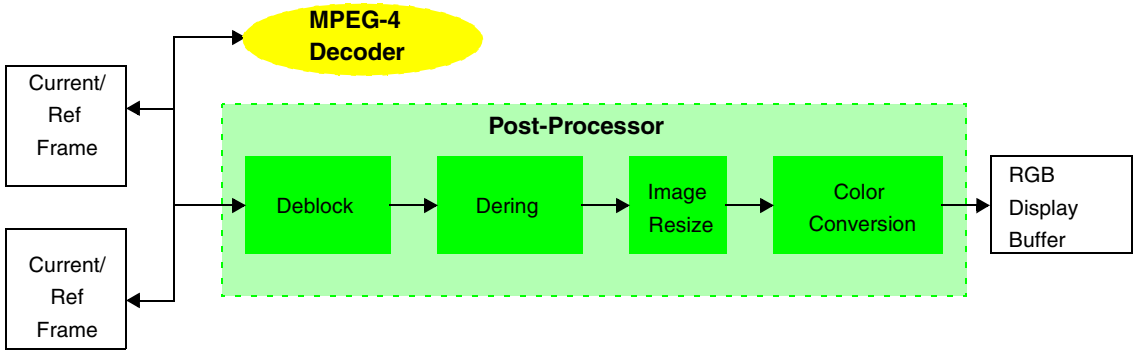


Figure 1-6. Post-Processor

Post-Processor features:

- Input data:
  - From system memory
- Input format:
  - YUV 4:2:0 (Planar)
- Output format:
  - YUV422

— RGB444
— RGB565
— RGB666
— RGB888 (unpacked)

- Input Size: Maximum size of 2044 × 2044
- Image Resize:
  — Upscaling ratios ranging from 1:1 to 1:4 in fractional steps
  — Downscaling ratios ranging from 1:1 to 2:1 in fractional steps and a fixed 4:1
  — Ratios provide scaling between QCIF, CIF, QVGA (320 × 240) and QVGA (240 × 320)

1.2.10.3 Digital Audio Multiplexer (AUDMUX)

The Digital Audio Multiplexer (AUDMUX) provides a programmable interconnect fabric for voice, audio, and synchronous data routing between the i.MX27 processor's SSI modules and external SSI, audio, and voice codecs. The AUDMUX is designed so that resource configurations do not need to be hard-wired, but instead, can be shared in many different configurations. The AUDMUX interconnections allow multiple simultaneous separate audio/voice/data flows between the ports in a point-to-point or point-to-multipoint configuration(s).

In a typical scenario, the AUDMUX and two SSI/I^{2}S modules provide interfaces to the Serial Audio Port of the cellular baseband (BB), narrowband (NB), and wideband (WB) audio ports of the external audio AD/DA, and to the audio port of the Bluetooth wireless technology on-board peripheral. See Figure 1-7.

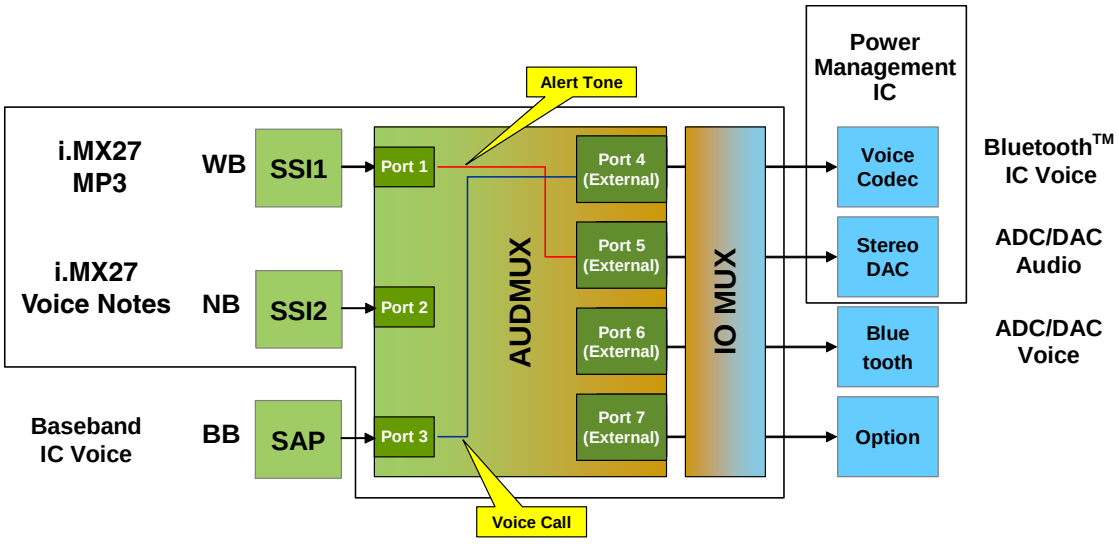


Figure 1-7. Typical AUDMUX Application

1.2.11 MultiMedia Interface

The CMOS Sensor Interface (CSI) provides multimedia interfacing.

1.2.11.1 CMOS Sensor Interface (CSI)

The CMOS Sensor Interface (CSI) is a logic interface that enables the i.MX27 device to connect directly to external CMOS sensors and a CCIR656 video source.

The sensor port provides a connection to either one or two image sensors, of which only one sensor can be active at any given time. The sensor port supports a direct parallel interface to either CMOS or CCD sensor controllers using a parallel interface with widths of 12 bits, 10 bits, 8 bits, or 4 bits at data bus rates up to 60 MHz. The sensor port may be configured to perform outputs of a still image to a non-contiguous memory buffer, enabling efficient memory use under an open OS.

The capabilities of the CSI include:

- Configurable interface logic to support common available CMOS sensors in the market
- Support traditional sensor timing interface
- Support CCIR656 video interface, progressive mode for smart sensor, interlace mode for PAL and NTSC input
- 8-bit input port for YCC, YUV, Bayer, or RGB data
- 32 × 32 FIFO storing image data supporting core data read and DMA data burst transfer to system memory
- Full control of 8-bit and 16-bit data to 32-bit FIFO packing
- Direct interface to the eMMA_lt Pre-Processing block (PrP)
- Single interrupt source to the interrupt controller from maskable sensor interrupt sources: Start of Frame, End of Frame, Change of Field, FIFO full
- Configurable master clock frequency output to sensor
- Asynchronous input logic design. Sensor master clock can be driven by either the i.MX27 processor or by an external clock source.
- Statistic data generation for Auto Exposure (AE) and Auto White Balance (AWB) control of the camera (for Bayer data only)

1.2.12 Human Interface

The i.MX27 processor can connect to a wide variety of popular display devices, such as:

- RAM-less LCD panels—up to 40 Mpix/s (for example, SVGA @ 80 fps), 262k colors. Results are dependent on end application.
- LCD panels with integrated frame buffer—up to 1024 × 1024, 14M colors. Results are dependent on end application.
- Graphics accelerators
- TV encoders

The i.MX27 processor's display ports enable simultaneous connectivity of up to two displays—an LCD without memory and a TV encoder—as well as provides connectivity to three interface types:

- Synchronous parallel (18-bit)
- Asynchronous parallel (18-bit)
- Asynchronous Serial (SPI-like) at a bus rate of up to 100 MHz

1.2.12.1 Liquid Crystal Display Controller (LCDC)

The Liquid Crystal Display Controller (LCDC) provides display data for external gray-scale or color LCD panels. The LCDC features include the following:

- Software programmable screen size (up to 800 × 600) to support single (non-split) monochrome, color STN panels, and color TFT panels
- Support for color depth for CSTN panels: 4- or 8-bit mapping from 256 × 18 table, 12-bit true color
- Support for color depth for TFT panels: 4- or 8-bit mapping from 256 × 18 table, 16-bit/18-bit/24-bit true color
- Up to 16 grey levels out of 16 palettes
- Capable of directly driving popular LCD drivers from manufacturers including Motorola, Sharp, Hitachi, and Toshiba
- Support for data bus width of 16-bit or 18-bit TFT panels
- Support for the AUO panel in 16 bpp and 24 bpp pixel modes
- Support for data bus widths of 8-bit, 4-bit, 2-bit, and 1-bit monochrome LCD panels
- Direct interface to Sharp® 320 × 240 and 240 × 320 HR-TFT panels and other generic panels
- Support for logical operation between color hardware cursor and background
- LCD contrast control using 8-bit PWM
- Support for self-refresh LCD modules
- Hardware panning (soft horizontal scrolling)
- Windowing support for one graphic or text overlay

1.2.12.2 Smart Liquid Crystal Display Controller (SLCDC)

The Smart Liquid Crystal Display Controller (SLCDC) transparently and efficiently transfers image data from system memory to an external LCD controller. The SLCDC module contains a DMA controller that transfers image and control data from system memory to the SLCDC FIFO, where it is formatted and sent out to the external LCD controller.

The SLCDC can be configured to write image data to an external LCD controller via a 4-line serial, 3-line serial, or 8- or 16-bit parallel interface. The SLCDC has two FIFOs where command and display data are loaded via DMA. The display data is tagged with commands that are used by the SLCDC to communicate display information and data to the Smart LCD panel.

The command tagged data format of the SLCDC provides flexibility and ease of connection to existing and new smart LCD panels.

1.2.12.3 Keypad Port (KPP)

The Keypad Port (KPP) is used for key pad matrix scanning or as a general purpose I/O. This peripheral simplifies the software task of scanning a keypad matrix. Features include:

- Up to 8×8 external key pad matrix support
- Open drain design
- Glitch suppression circuit prevents erroneous key detection
- Multiple keys detection
- Standby key press detection

1.2.13 Packaging Information

The i.MX27 processor is offered in the 404 MAPBGA package option. This package brings out all the new interfaces, and supplies more flexible multiplexing.

- Type: 0.65 mm pitch
- Dimensions: 17 mm $\times$ 17 mm
- Balls: 404

Chapter 2
System Memory and Register Map

2.1 Introduction

This chapter provides the i.MX27 Multimedia Applications Processor's memory maps and chip configuration registers.

2.2 Memory Space

The i.MX27 Multimedia Applications Processor, with a 32-bit address bus, is capable of addressing a 4-Gbyte physical address space. This space is divided into sections of 512-Mbyte regions within which various memories and peripherals are mapped.

Table 2-1 shows a simplified breakdown of the eight 512-Mbyte regions decoded within the 4-Gbyte address space.

Table 2-1. 4 Gbyte Memory Map Breakdown

Address	Size	Usage
0x0000_0000	512 Mbyte	ROM, Primary AHB Slaves, and Peripherals
0x2000_0000	512 Mbyte	Reserved
0x4000_0000	512 Mbyte	Reserved
0x6000_0000	512 Mbyte	Reserved
0x8000_0000	512 Mbyte	Secondary AHB Slave Port 1
0xA000_0000	1 Gbyte	Secondary AHB Slave Port 2
0xE000_0000	512 Mbyte	Primary AHB (RAM)

2.2.1 Detailed Memory Map

Figure 2-2 shows the memory space breakout view for the i.MX27 processor. The left-most column shows the eight 512-Mbyte regions. The middle column shows the breakout of primary and secondary AHB slaves, and the right-most column shows the breakout of the AIPI1 and AIPI2 address spaces.

Table 2-2 through Table 2-5 show the detailed breakdown of the complete memory map according to the 512-Mbyte regions. Table 2-6 and Table 2-7 show the detailed breakdown of the AIPI1 and AIPI2 modules and the different IP peripherals accessed over the AIPI1 and AIPI2.

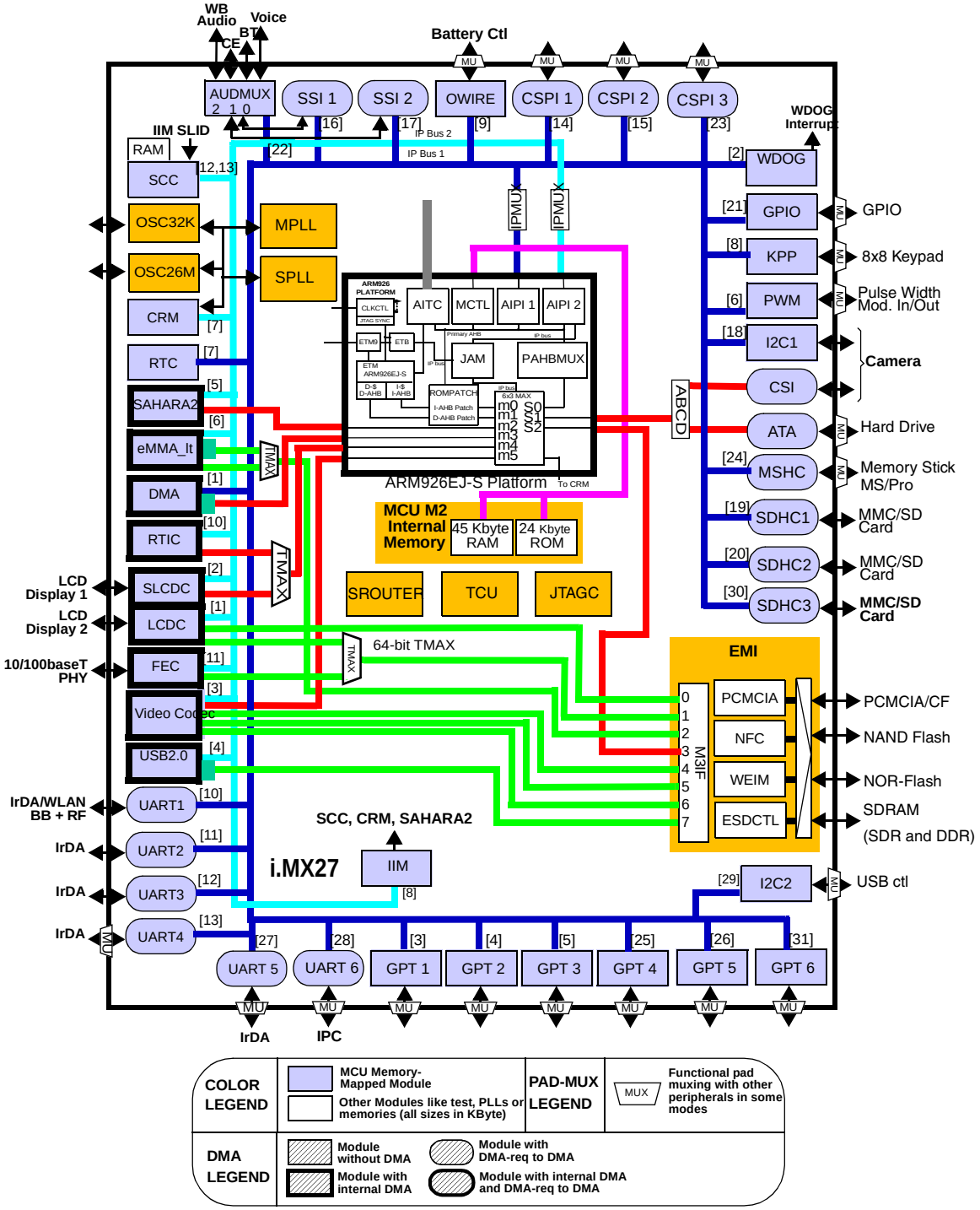


Figure 2-1. Detailed Block Diagram for the i.MX27 Processor

NOTE

Accesses to locations defined as Reserved (other than aliased RAM space) result in an AHB error response. Accesses to unimplemented locations within the AITC register spaces will be terminated, write accesses will have no effect, and read accesses will return all zeros.

Table 2-2 shows the memory map of the Primary AHB address space in the first 512-Mbyte region.

Table 2-2. Primary AHB Memory Map (Lower)

Address	Secondary AHB Slave Port 1	Size
0x0000_0000–0x0000_3FFF	BROM	16 Kbyte
0x0000_4000–0x0040_3FFF	Reserved	4 Mbyte
0x0040_4000–0x0040_5FFF	BROM	8 Kbyte
0x0040_6000– 0x007F_FFFF	BROM (Hole)	3 Mbyte + 1000 Kbyte
0x0080_0000–0x0FFF_FFFF	Reserved	248 Mbyte
0x1000_0000–0x1001_FFFF	AIPI1	128 Kbyte
0x1002_0000–0x1003_FFFF	AIPI2	128 Kbyte
0x1004_0000–0x1004_0FFF	AITC	4 Kbyte
0x1004_1000– 0x1004_1FFF	ROM Patch	4 Kbyte
0x1004_2000–0x7FFF_FFFF	Reserved	255 Mbyte + 752 Kbyte

Table 2-3 shows the memory map of the CSI and ATA modules when connected to the Secondary AHB Ports 1 via the ABCD. The BROM (hole) is split into two sections of 16 Kbytes and 8 Kbytes. The BROM (Hole) indicates that there is no BROM code present in this region. The AIPI1 and AIPI2 address space contains AIPI control registers and the IP slave registers. The AIPI1 and AIPI2 maps are shown in Table 2-6 and Table 2-7, respectively.

Table 2-3. Secondary AHB Port 1 Memory Map

Address	Secondary AHB Port 1	Size
0x8000_0000–0x8000_0FFF	CSI	4 Kbyte
0x8000_1000–0x8000_1FFF	ATA	4 Kbyte
0x8000_2000–0x9FFF_FFFF	Reserved	512 Mbyte–8 Kbyte

Figure 2-2 shows the i.MX27 processor's physical memory map (4 Gbytes).

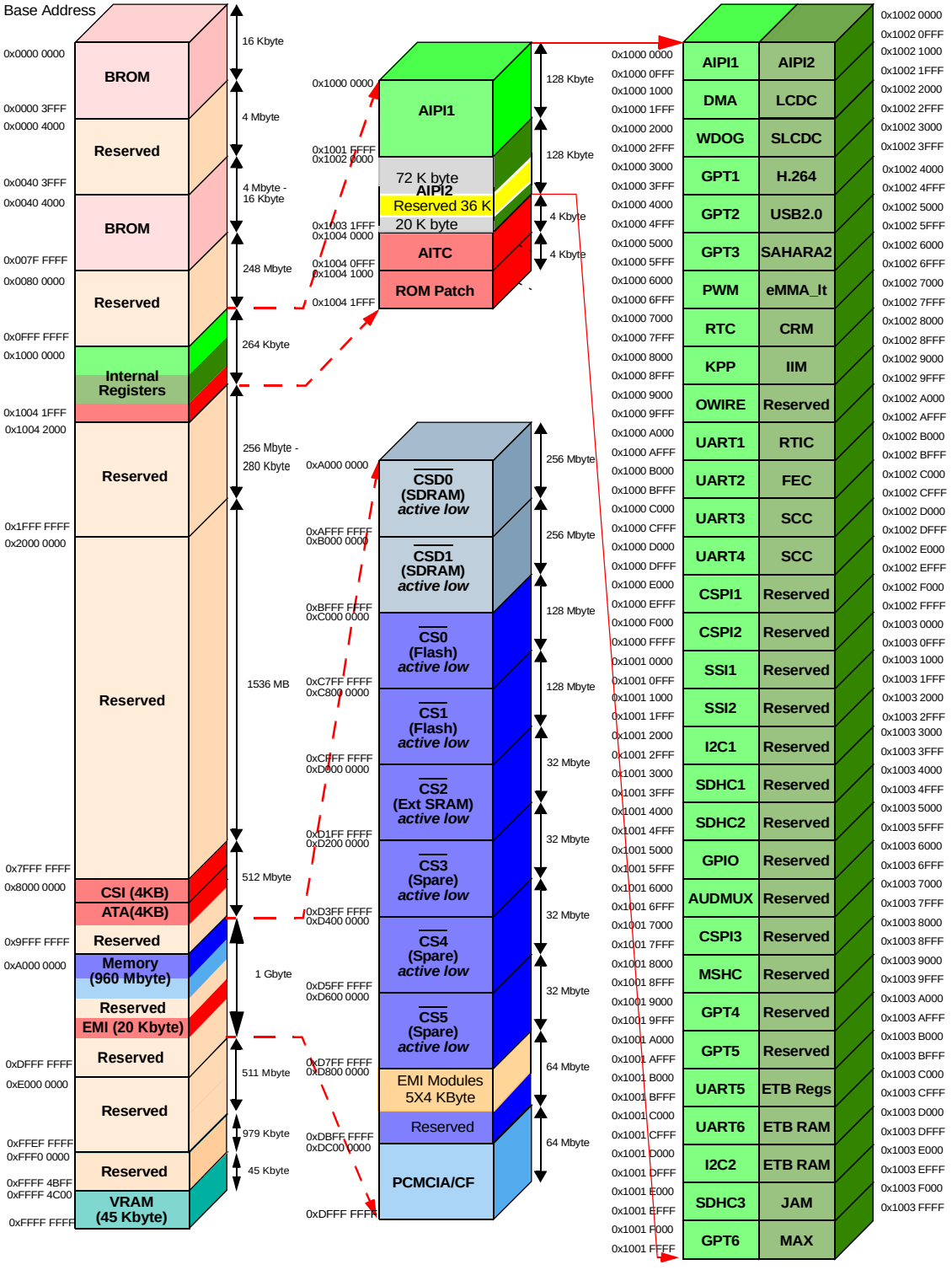


Figure 2-2. i.MX27 Processor's Physical Memory Map (4 Gbytes)

Table 2-4 shows the memory map breakdown for the Secondary AHB Port 3. The SDRAMC, WEIM, PCMCIA, and NFC module control registers and external memory are addressed via this region. The external memory regions (memory or external peripherals) are accessed via the respective chip selects. $\overline{CSD1}$ and $\overline{CSD0}$ are SDRAMC chip selects, and $\overline{CS5}$ to $\overline{CS0}$ are WEIM chip selects. $\overline{CSD1}$ and $\overline{CS0}$ chip select spaces are available for external boot. 0xBC000 0000 to 0xDFFF_FFFF is allocated for PCMCIA IO and memory space.

Table 2-4. Secondary AHB Port 2 Memory Map

Address	Secondary AHB Port 3	Size
0xA000_0000–0xAFFF_FFFF	External SDRAM/MDDR ($\overline{CSD0}$)	256 Mbyte
0xB000_0000–0xBFFF_FFFF	External SDRAM/MDDR ($\overline{CSD1}$)	256 Mbyte
0xC000_0000–0xC7FF_FFFF	WEIM External Memory ($\overline{CS0}$)	128 Mbyte
0xC800_0000–0xCFFF_FFFF	WEIM External Memory ($\overline{CS1}$)	128 Mbyte
0xD000_0000–0xD1FF_FFFF	WEIM External Memory ($\overline{CS2}$)	32 Mbyte
0xD200_0000–0xD3FF_FFFF	WEIM External Memory ($\overline{CS3}$)	32 Mbyte
0xD400_0000–0xD5FF_FFFF	WEIM External Memory ($\overline{CS4}$)	32 Mbyte
0xD600_0000–0xD7FF_FFFF	WEIM External Memory ($\overline{CS5}$)	32 Mbyte
0xD800_0000–0xD800_0FFF	NFC registers and internal RAM	4 Kbyte
0xD800_1000–0xD800_1FFF	SDRAMC registers	4 Kbyte
0xD800_2000–0xD800_2FFF	WEIM registers	4 Kbyte
0xD800_3000–0xD800_3FFF	M3IF registers	4 Kbyte
0xD800_4000–0xD800_4FFF	PCMCIA registers	4 Kbyte
0xD800_5000–0xDBFF_FFFF	Reserved	64 Mbyte–20 Kbyte
0xDC00_0000–0xDFFF_FFFF	PCMCIA Memory Space	64 Mbyte

Table 2-5 shows the last region of address space that is part of the Primary AHB Memory Map. The Vector-RAM is mapped into this region and the i.MX27 device uses the high memory (0xFFFF FF00–0xFFFF FFFF) to store the interrupt vector table (64 words). This region is aliased on a 128-Kbyte boundary.

Table 2-5. Primary AHB Memory Map (Upper)

Address	Primary AHB	Size
0xE000_0000–0xFFEF_FFFF	Reserved (aliased)	511 Mbyte
0xFFF0_0000–0xFFFF_4BFF	VRAM Space Not Use	979 Kbyte
0xFFFF_4C00–0xFFFF_FFFF	45 Kbyte VRAM	45 Kbyte

Table 2-6 and Table 2-7 show the detailed breakdown of the address space controlled by AIPI1 and AIPI2. More details on the AIPI can be found in Chapter 35, “AHB-Lite IP Interface (AIPI) Module.”

Table 2-6. AIPI1 Memory Map

	Address	AIPI1 Memory Map	Size
0	0x1000_0000–0x1000_0FFF	AIPI1 (slot 0)	4 Kbyte
1	0x1000_1000–0x1000_1FFF	DMA	4 Kbyte
2	0x1000_2000–0x1000_2FFF	WDOG	4 Kbyte
3	0x1000_3000–0x1000_3FFF	GPT1	4 Kbyte
4	0x1000_4000–0x1000_4FFF	GPT2	4 Kbyte
5	0x1000_5000–0x1000_5FFF	GPT3	4 Kbyte
6	0x1000_6000–0x1000_6FFF	PWM	4 Kbyte
7	0x1000_7000–0x1000_7FFF	RTC	4 Kbyte
8	0x1000_8000–0x1000_8FFF	KPP	4 Kbyte
9	0x1000_9000–0x1000_9FFF	OWIRE	4 Kbyte
10	0x1000_A000–0x1000_AFFF	UART1	4 Kbyte
11	0x1000_B000–0x1000_BFFF	UART2	4 Kbyte
12	0x1000_C000–0x1000_CFFF	UART3	4 Kbyte
13	0x1000_D000–0x1000_DFFF	UART4	4 Kbyte
14	0x1000_E000–0x1000_EFFF	CSPI1	4 Kbyte
15	0x1000_F000–0x1000_FFFF	CSPI2	4 Kbyte
16	0x1001_0000–0x1001_0FFF	SSI1	4 Kbyte
17	0x1001_1000–0x1001_1FFF	SSI2	4 Kbyte
18	0x1001_2000–0x1001_2FFF	I2C1	4 Kbyte
19	0x1001_3000–0x1001_3FFF	SDHC1	4 Kbyte
20	0x1001_4000–0x1001_4FFF	SDHC2	4 Kbyte
21	0x1001_5000–0x1001_5FFF	GPIO	4 Kbyte
22	0x1001_6000–0x1001_6FFF	AUDMUX	4 Kbyte
23	0x1001_7000–0x1001_7FFF	CSPI3	4 Kbyte
24	0x1001_8000–0x1001_8FFF	MSHC	4 Kbyte
25	0x1001_9000–0x1001_9FFF	GPT4	4 Kbyte
26	0x1001_A000–0x1001_AFFF	GPT5	4 Kbyte
27	0x1001_B000–0x1001_BFFF	UART5	4 Kbyte
28	0x1001_C000–0x1001_CFFF	UART6	4 Kbyte
29	0x1001_D000–0x1001_DFFF	I2C2	4 Kbyte
30	0x1001_E000–0x1001_EFFF	SDHC3	4 Kbyte
31	0x1001_F000–0x1001_FFFF	GPT6	4 Kbyte

Table 2-7. AIPI2 Memory Map

	Address	AIPI2 Memory Map	Size
0	0x1002_0000–0x1002_0FFF	AIPI2 (Slot 0)	4 Kbyte
1	0x1002_1000–0x1002_1FFF	LCDC	4 Kbyte
2	0x1002_2000–0x1002_2FFF	SLCDC	4 Kbyte
3	0x1002_3000–0x1002_3FFF	Reserved	4 Kbyte
4	0x1002_4000–0x1002_4FFF	USB2.0	4 Kbyte
5	0x1002_5000–0x1002_5FFF	SAHARA2	4 Kbyte
6	0x1002_6000–0x1002_6FFF	eMMA_lt	4 Kbyte
7	0x1002_7000–0x1002_7FFF	CRM	4 Kbyte
8	0x1002_8000–0x1002_8FFF	IIM	4 Kbyte
9	0x1002_9000–0x1002_9FFF	Reserved	4 Kbyte
10	0x1002_A000–0x1002_AFFF	RTIC	4 Kbyte
11	0x1002_B000–0x1002_BFFF	FEC	4 Kbyte
12	0x1002_C000–0x1002_CFFF	SCC	4 Kbyte
13	0x1002_D000–0x1002_DFFF	SCC	4 Kbyte
14–26	0x1002_E000–0x1003_AFFF	Reserved (slots 14–26)	52 Kbyte
27	0x1003_B000–0x1003_BFFF	ETB Regs	4 Kbyte
28	0x1003_C000–0x1003_CFFF	ETB RAM	4 Kbyte
29	0x1003_D000–0x1003_DFFF	ETB RAM	4 Kbyte
30	0x1003_E000–0x1003_EFFF	JAM	4 Kbyte
31	0x1003_F000–0x1003_FFFF	MAX	4 Kbyte

2.3 Register Map

The internal registers in the i.MX27 processor are listed in Table 2-8.

Table 2-8. Register Map

Module Name	Address	Register Name	Description
AIPI1	0x1000_0000	PSR0	Peripheral Size Register 0
AIPI1	0x1000_0004	PSR1	Peripheral Size Register 1
AIPI1	0x1000_0008	PAR	Peripheral Access Register
AIPI1	0x1000_000C	AAOR	Atomic Access Only Register
DMAC	0x1000_1000	DCR	DMA Control Register
DMAC	0x1000_1004	DISR	DMA Interrupt Status Register
DMAC	0x1000_1008	DIMR	DMA Interrupt Mask Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
DMAC	0x1000_100C	DBTOSR	DMA Burst Time-Out Status Register
DMAC	0x1000_1010	DRTOSR	DMA Request Time-Out Status Register
DMAC	0x1000_1014	DSESR	DMA Transfer Error Status Register
DMAC	0x1000_1018	DBOSR	DMA Buffer Overflow Status Register
DMAC	0x1000_101C	DBTOCR	DMA Burst Time-Out Control Register
DMAC	0x1000_1040	WSRA	W-Size Register A
DMAC	0x1000_1044	XSRA	X-Size Register A
DMAC	0x1000_1048	YSRA	Y-Size Register A
DMAC	0x1000_104C	WSRB	W-Size Register B
DMAC	0x1000_1050	XSRB	X-Size Register B
DMAC	0x1000_1054	YSRB	Y-Size Register B
DMAC	0x1000_1080	SAR0	Channel 0 Source Address Register
DMAC	0x1000_1084	DAR0	Channel 0 Destination Address Register
DMAC	0x1000_1088	CNTR0	Channel 0 Count Register
DMAC	0x1000_108C	CCR0	Channel 0 Control Register
DMAC	0x1000_1090	RSSR0	Channel 0 Request Source Select Register
DMAC	0x1000_1094	BLR0	Channel 0 Burst Length Register
DMAC	0x1000_1098	RTOR0 BUCR0	Channel 0 Request Time-Out Register Channel 0 Bus Utilization Control Register
DMAC	0x1000_109C	CCNR0	Channel 0 Channel Counter Register
DMAC	0x1000_10C0	SAR1	Channel 1 Source Address Register
DMAC	0x1000_10C4	DAR1	Channel 1 Destination Address Register
DMAC	0x1000_10C8	CNTR1	Channel 1 Count Register
DMAC	0x1000_10CC	CCR1	Channel 1 Control Register
DMAC	0x1000_10D0	RSSR1	Channel 1 Request Source Select Register
DMAC	0x1000_10D4	BLR1	Channel 1 Burst Length Register
DMAC	0x1000_10D8	RTOR1 BUCR1	Channel 1 Request Time-Out Register Channel 1 Bus Utilization Control Register
DMAC	0x1000_10DC	CCNR1	Channel 1 Channel Counter Register
DMAC	0x1000_1100	SAR2	Channel 2 Source Address Register
DMAC	0x1000_1104	DAR2	Channel 2 Destination Address Register
DMAC	0x1000_1108	CNTR2	Channel 2 Count Register
DMAC	0x1000_110C	CCR2	Channel 2 Control Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
DMAC	0x1000 1110	RSSR2	Channel 2 Request Source Select Register
DMAC	0x1000 1114	BLR2	Channel 2 Burst Length Register
DMAC	0x1000 1118	RTOR2 BUCR2	Channel 2 Request Time-Out Register Channel 2 Bus Utilization Control Register
DMAC	0x1000 111C	CCNR2	Channel 2 Channel Counter Register
DMAC	0x1000 1140	SAR3	Channel 3 Source Address Register
DMAC	0x1000 1144	DAR3	Channel 3 Destination Address Register
DMAC	0x1000 1148	CNTR3	Channel 3 Count Register
DMAC	0x1000 114C	CCR3	Channel 3 Control Register
DMAC	0x1000 1150	RSSR3	Channel 3 Request Source Select Register
DMAC	0x1000 1154	BLR3	Channel 3 Burst Length Register
DMAC	0x1000 1158	RTOR3 BUCR3	Channel 3 Request Time-Out Register Channel 3 Bus Utilization Control Register
DMAC	0x1000 115C	CCNR3	Channel 3 Channel Counter Register
DMAC	0x1000 1180	SAR4	Channel 4 Source Address Register
DMAC	0x1000 1184	DAR4	Channel 4 Destination Address Register
DMAC	0x1000 1188	CNTR4	Channel 4 Count Register
DMAC	0x1000 118C	CCR4	Channel 4 Control Register
DMAC	0x1000 1190	RSSR4	Channel 4 Request Source Select Register
DMAC	0x1000 1194	BLR4	Channel 4 Burst Length Register
DMAC	0x1000 1198	RTOR4 BUCR4	Channel 4 Request Time-Out Register Channel 4 Bus Utilization Control Register
DMAC	0x1000 119C	CCNR 4	Channel 4 Channel Counter Register
DMAC	0x1000 11C0	SAR5	Channel 5 Source Address Register
DMAC	0x1000 11C4	DAR5	Channel 5 Destination Address Register
DMAC	0x1000 11C8	CNTR5	Channel 5 Count Register
DMAC	0x1000 11CC	CCR5	Channel 5 Control Register
DMAC	0x1000 11D0	RSSR5	Channel 5 Request Source Select Register
DMAC	0x1000 11D4	BLR5	Channel 5 Burst Length Register
DMAC	0x1000 11D8	RTOR5 BUCR5	Channel 5 Request Time-Out Register Channel 5 Bus Utilization Control Register
DMAC	0x1000 11DC	CCNR5	Channel 5 Channel Counter Register
DMAC	0x1000 1200	SAR6	Channel 6 Source Address Register
DMAC	0x1000 1204	DAR6	Channel 6 Destination Address Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
DMAC	0x1000 1208	CNTR6	Channel 6 Count Register
DMAC	0x1000 120C	CCR6	Channel 6 Control Register
DMAC	0x1000 1210	RSSR6	Channel 6 Request Source Select Register
DMAC	0x1000 1214	BLR6	Channel 6 Burst Length Register
DMAC	0x1000 1218	RTOR6 BUCR6	Channel 6 Request Time-Out Register Channel 6 Bus Utilization Control Register
DMAC	0x1000 121C	CCNR6	Channel 6 Channel Counter Register
DMAC	0x1000 1240	SAR7	Channel 7 Source Address Register
DMAC	0x1000 1244	DAR7	Channel 7 Destination Address Register
DMAC	0x1000 1248	CNTR7	Channel 7 Count Register
DMAC	0x1000 124C	CCR7	Channel 7 Control Register
DMAC	0x1000 1250	RSSR7	Channel 7 Request Source Select Register
DMAC	0x1000 1254	BLR7	Channel 7 Burst Length Register
DMAC	0x1000 1258	RTOR7 BUCR7	Channel 7 Request Time-Out Register Channel 7 Bus Utilization Control Register
DMAC	0x1000 125C	CCNR7	Channel 7 Channel Counter Register
DMAC	0x1000 1280	SAR8	Channel 8 Source Address Register
DMAC	0x1000 1284	DAR8	Channel 8 Destination Address Register
DMAC	0x1000 1288	CNTR8	Channel 8 Count Register
DMAC	0x1000 128C	CCR8	Channel 8 Control Register
DMAC	0x1000 1290	RSSR8	Channel 8 Request Source Select Register
DMAC	0x1000 1294	BLR8	Channel 8 Burst Length Register
DMAC	0x1000 1298	RTOR8 BUCR8	Channel 8 Request Time-Out Register Channel 8 Bus Utilization Control Register
DMAC	0x1000 129C	CCNR8	Channel 8 Channel Counter Register
DMAC	0x1000 12C0	SAR9	Channel 9 Source Address Register
DMAC	0x1000 12C4	DAR9	Channel 9 Destination Address Register
DMAC	0x1000 12C8	CNTR9	Channel 9 Count Register
DMAC	0x1000 12CC	CCR9	Channel 9 Control Register
DMAC	0x1000 12D0	RSSR9	Channel 9 Request Source Select Register
DMAC	0x1000 12D4	BLR9	Channel 9 Burst Length Register
DMAC	0x1000 12D8	RTOR9 BUCR9	Channel 9 Request Time-Out Register Channel 9 Bus Utilization Control Register
DMAC	0x1000 12DC	CCNR9	Channel 9 Channel Counter Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
DMAC	0x1000 1300	SAR10	Channel 10 Source Address Register
DMAC	0x1000 1304	DAR10	Channel 10 Destination Address Register
DMAC	0x1000 1308	CNTR10	Channel 10 Count Register
DMAC	0x1000 130C	CCR10	Channel 10 Control Register
DMAC	0x1000 1310	RSSR10	Channel 10 Request Source Select Register
DMAC	0x1000 1314	BLR10	Channel 10 Burst Length Register
DMAC	0x1000 1318	RTOR10 BUCR10	Channel 10 Request Time-Out Register Channel 10 Bus Utilization Control Register
DMAC	0x1000 131C	CCNR10	Channel 10 Channel Counter Register
DMAC	0x1000 1340	SAR11	Channel 11 Source Address Register
DMAC	0x1000 1344	DAR11	Channel 11 Destination Address Register
DMAC	0x1000 1348	CNTR11	Channel 11 Count Register
DMAC	0x1000 134C	CCR11	Channel 11 Control Register
DMAC	0x1000 1350	RSSR11	Channel 11 Request Source Select Register
DMAC	0x1000 1354	BLR11	Channel 11 Burst Length Register
DMAC	0x1000 1358	RTOR11 BUCR11	Channel 11 Request Time-Out Register Channel 11 Bus Utilization Control Register
DMAC	0x1000 135C	CCNR11	Channel 11 Channel Counter Register
DMAC	0x1000 1380	SAR12	Channel 12 Source Address Register
DMAC	0x1000 1384	DAR12	Channel 12 Destination Address Register
DMAC	0x1000 1388	CNTR12	Channel 12 Count Register
DMAC	0x1000 138C	CCR12	Channel 12 Control Register
DMAC	0x1000 1390	RSSR12	Channel 12 Request Source Select Register
DMAC	0x1000 1394	BLR12	Channel 12 Burst Length Register
DMAC	0x1000 1398	RTOR12 BUCR12	Channel 12 Request Time-Out Register Channel 12 Bus Utilization Control Register
DMAC	0x1000 139C	CCNR12	Channel 14 Channel Counter Register
DMAC	0x1000 13C0	SAR13	Channel 13 Source Address Register
DMAC	0x1000 13C4	DAR13	Channel 13 Destination Address Register
DMAC	0x1000 13C8	CNTR13	Channel 13 Count Register
DMAC	0x1000 13CC	CCR13	Channel 13 Control Register
DMAC	0x1000 13D0	RSSR13	Channel 13 Request Source Select Register
DMAC	0x1000 13D4	BLR13	Channel 13 Burst Length Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
DMAC	0x1000 13D8	RTOR13 BUCR13	Channel 13 Request Time-Out Register Channel 13 Bus Utilization Control Register
DMAC	0x1000 13DC	CCNR13	Channel 13 Channel Counter Register
DMAC	0x1000 1400	SAR14	Channel 14 Source Address Register
DMAC	0x1000 1404	DAR14	Channel 14 Destination Address Register
DMAC	0x1000 1408	CNTR14	Channel 14 Count Register
DMAC	0x1000 140C	CCR14	Channel 14 Control Register
DMAC	0x1000 1410	RSSR14	Channel 14 Request Source Select Register
DMAC	0x1000 1414	BLR14	Channel 14 Burst Length Register
DMAC	0x1000 1418	RTOR14 BUCR14	Channel 14 Request Time-Out Register Channel 14 Bus Utilization Control Register
DMAC	0x1000 141C	CCNR14	Channel 14 Channel Counter Register
DMAC	0x1000 1440	SAR15	Channel 15 Source Address Register
DMAC	0x1000 1444	DAR15	Channel 15 Destination Address Register
DMAC	0x1000 1448	CNTR15	Channel 15 Count Register
DMAC	0x1000 144C	CCR15	Channel 15 Control Register
DMAC	0x1000 1450	RSSR15	Channel 15 Request Source Select Register
DMAC	0x1000 1454	BLR15	Channel 15 Burst Length Register
DMAC	0x1000 1458	RTOR15 BUCR15	Channel 15 Request Time-Out Register Channel 15 Bus Utilization Control Register
DMAC	0x1000 145C	CCNR15	Channel 15 Channel Counter Register
DMAC	0x1000 1480	TCR	Test Control Register
DMAC	0x1000 1484	TFIFOAR	Test FIFO A Register
DMAC	0x1000 148C	TDIPR	Test DMA In Progress Register
DMAC	0x1000 1490	TFIFOBR	Test FIFO B Register
DMAC	0x1000 1498	TDRR_L	Low 32 DMA Request Register
DMAC	0x1000 149C	TDRR_H	High 32 DMA Request Register
WDOG	0x1000 2000	WCR	Watchdog Control Register
WDOG	0x1000 2002	WSR	Watchdog Service Register
WDOG	0x1000 2004	WRSR	Watchdog Reset Status Register
GPT1	0x1000 3000	TCTL1	GPT Control Register 1
GPT1	0x1000 3004	TPRER1	GPT Prescaler Register 1
GPT1	0x1000 3008	TCMP1	GPT Compare Register 1
GPT1	0x1000 300C	TCR1	GPT Capture Register 1

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
GPT1	0x1000 3010	TCN1	GPT Counter Register 1
GPT1	0x1000 3014	TSTAT1	GPT Status Register 1
GPT2	0x1000 4000	TCTL2	GPT Control Register 2
GPT2	0x1000 4004	TPRER2	GPT Prescaler Register 2
GPT2	0x1000 4008	TCMP2	GPT Compare Register 2
GPT2	0x1000 400C	TCR2	GPT Capture Register 2
GPT2	0x1000 4010	TCN2	GPT Counter Register 2
GPT2	0x1000 4014	TSTAT2	GPT Status Register 2
GPT3	0x1000 5000	TCTL3	GPT Control Register 3
GPT3	0x1000 5004	TPRER3	GPT Prescaler Register 3
GPT3	0x1000 5008	TCMP3	GPT Compare Register 3
GPT3	0x1000 500C	TCR3	GPT Capture Register 3
GPT3	0x1000 5010	TCN3	GPT Counter Register 3
GPT3	0x1000 5014	TSTAT3	GPT Status Register 3
PWM	0x1000 6000	PWMCR	PWM Control Register
PWM	0x1000 6004	PWMSR	PWM Status Register
PWM	0x1000 6008	PWMIR	PWM Interrupt Register
PWM	0x1000 600C	PWMSAR	PWM Sample Register
PWM	0x1000 6010	PWMPR	PWM Period Register
PWM	0x1000 6014	PWMCNR	PWM Counter Register
RTC	0x1000 7000	HOURMIN	RTC Hours and Minutes Counter Register
RTC	0x1000 7004	SECONDS	RTC Seconds Counter Register
RTC	0x1000 7008	ALRM_HM	RTC Hours and Minutes Alarm Register
RTC	0x1000 700C	ALRM_SEC	RTC Seconds Alarm Register
RTC	0x1000 7010	RCCTL	RTC Control Register
RTC	0x1000 7014	RTCISR	RTC Interrupt Status Register
RTC	0x1000 7018	RTCIENR	RTC Interrupt Enable Register
RTC	0x1000 701C	STPWCH	Stopwatch Minutes Register
RTC	0x1000 7020	DAYR	RTC Days Counter Register
RTC	0x1000 7024	DAYALARM	RTC Day Alarm Register
KPP	0x1000_8000	KPCR	Keypad Control Register
KPP	0x1000_8002	KPSR	Keypad Status Register
KPP	0x1000_8004	KDDR	Keypad Data Direction Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
KPP	0x1000_8006	KPDR	Keypad Data Register
O-Wire	0x1000 9000	CONTROL	1-Wire Control Register
O-Wire	0x1000 9002	TIME_DIVIDER	1-Wire Time Divider Register
O-Wire	0x1000 9004	RESET	1-Wire Reset Register
UART1	0x1000_A000	UXRD_1	UART1 Receiver Register
UART1	0x1000_A040	UTXD_1	UART1 Transmitter Register
UART1	0x1000_A080	UCR1_1	UART1 Control Register
UART1	0x1000_A084	UCR2_1	UART1 Control Register 2
UART1	0x1000_A088	UCR3_1	UART1 Control Register 3
UART1	0x1000_A08C	UCR4_1	UART1 Control Register 4
UART1	0x1000_A090	UFCR_1	UART1 FIFO Control Register
UART1	0x1000_A094	USR1_1	UART1 Status Register 1
UART1	0x1000_A098	USR2_1	UART1 Status Register 2
UART1	0x1000_A09C	UESC_1	UART1 Escape Character Register
UART1	0x1000_A0A0	UTIM_1	UART1 Escape Timer Register
UART1	0x1000_A0A4	UBIR_1	UART1 BRM Incremental Register
UART1	0x1000_A0A8	UBMR_1	UART1 BRM Modulator Register
UART1	0x1000_A0AC	UBRC_1	UART1 Baud Rate Count Register
UART1	0x1000_A0B0	ONEMS_1	UART1 One Millisecond Register
UART1	0x1000_A0B4	UTS_1	UART1 Test Register 1
UART2	0x1000_B000	UXRD_2	UART2 Receiver Register
UART2	0x1000_B040	UTXD_2	UART2 Transmitter Register
UART2	0x1000_B080	UCR1_2	UART2 Control Register
UART2	0x1000_B084	UCR2_2	UART2 Control Register 2
UART2	0x1000_B088	UCR3_2	UART2 Control Register 3
UART2	0x1000_B08C	UCR4_2	UART2 Control Register 4
UART2	0x1000_B090	UFCR_2	UART2 FIFO Control Register
UART2	0x1000_B094	USR1_2	UART2 Status Register 1
UART2	0x1000_B098	USR2_2	UART2 Status Register 2
UART2	0x1000_B09C	UESC_2	UART2 Escape Character Register
UART2	0x1000_B0A0	UTIM_2	UART2 Escape Timer Register
UART2	0x1000_B0A4	UBIR_2	UART2 BRM Incremental Register
UART2	0x1000_B0A8	UBMR_2	UART2 BRM Modulator Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
UART2	0x1000_B0AC	UBRC_2	UART2 Baud Rate Count Register
UART2	0x1000_B0B0	ONEMS_2	UART2 One Millisecond Register
UART2	0x1000_B0B4	UTS_2	UART2 Test Register 1
UART3	0x1000_C000	UXRD_3	UART3 Receiver Register
UART3	0x1000_C040	UTXD_3	UART3 Transmitter Register
UART3	0x1000_C080	UCR1_3	UART3 Control Register
UART3	0x1000_C084	UCR2_3	UART3 Control Register 2
UART3	0x1000_C088	UCR3_3	UART3 Control Register 3
UART3	0x1000_C08C	UCR4_3	UART3 Control Register 4
UART3	0x1000_C090	UFCR_3	UART3 FIFO Control Register
UART3	0x1000_C094	USR1_3	UART3 Status Register 1
UART3	0x1000_C098	USR2_3	UART3 Status Register 2
UART3	0x1000_C09C	UESC_3	UART3 Escape Character Register
UART3	0x1000_C0A0	UTIM_3	UART3 Escape Timer Register
UART3	0x1000_C0A4	UBIR_3	UART3 BRM Incremental Register
UART3	0x1000_C0A8	UBMR_3	UART3 BRM Modulator Register
UART3	0x1000_C0AC	UBRC_3	UART3 Baud Rate Count Register
UART3	0x1000_C0B0	ONEMS_3	UART3 One Millisecond Register
UART3	0x1000_C0B4	UTS_3	UART3 Test Register 1
UART4	0x1000_D000	UXRD_4	UART4 Receiver Register
UART4	0x1000_D040	UTXD_4	UART4 Transmitter Register
UART4	0x1000_D080	UCR1_4	UART4 Control Register
UART4	0x1000_D084	UCR2_4	UART4 Control Register 2
UART4	0x1000_D088	UCR3_4	UART4 Control Register 3
UART4	0x1000_D08C	UCR4_4	UART4 Control Register 4
UART4	0x1000_D090	UFCR_4	UART4 FIFO Control Register
UART4	0x1000_D094	USR1_4	UART4 Status Register 1
UART4	0x1000_D098	USR2_4	UART4 Status Register 2
UART4	0x1000_D09C	UESC_4	UART4 Escape Character Register
UART4	0x1000_D0A0	UTIM_4	UART4 Escape Timer Register
UART4	0x1000_D0A4	UBIR_4	UART4 BRM Incremental Register
UART4	0x1000_D0A8	UBMR_4	UART4 BRM Modulator Register
UART4	0x1000_D0AC	UBRC_4	UART4 Baud Rate Count Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
UART4	0x1000_D0B0	ONEMS_4	UART4 One Millisecond Register
UART4	0x1000_D0B4	UTS_4	UART4 Test Register 1
CSPI1	0x1000 E000	RXDATA1	Receive Data Register 1
CSPI1	0x1000 E004	TXDATA1	Transmit Data Register 1
CSPI1	0x1000 E008	CONTROL_REG1	CSPI Control Register 1
CSPI1	0x1000 E00C	INT_REG1	Interrupt Control/Status Register 1
CSPI1	0x1000 E010	TEST_REG	CSPI Test Register 1
CSPI1	0x1000 E014	PERIOD1	CSPI Sample Period Control Register 1
CSPI1	0x1000 E018	CSPI_DMA1	CSPI DMA Register 1
CSPI1	0x1000 E01C	CSPI_RESET1	CSPI 1 Soft Reset Register
CSPI2	0x1000 F000	RXDATA2	Receive Data Register 2
CSPI2	0x1000 F004	TXDATA2	Transmit Data Register 2
CSPI2	0x1000 F008	CONTROL_REG2	CSPI Control Register 2
CSPI2	0x1000 F00C	INT_REG2	Interrupt Control/Status Register 2
CSPI2	0x1000 F010	TEST_REG 2	CSPI Test Register 2
CSPI2	0x1000 F014	PERIOD2	CSPI Sample Period Control Register 2
CSPI2	0x1000 F018	CSPI_DMA2	CSPI DMA Register 2
CSPI2	0x1000 F01C	CSPI_RESET2	CSPI 2 Soft Reset Register
SSI 1	0x1001 0000	STX0	SSI Transmit Data Register 0
SSI 1	0x1001 0004	STX1	SSI Transmit Data Register 1
SSI 1	0x1001 0008	SRX0	SSI Receive Data Register 0
SSI 1	0x1001 000C	SRX1	SSI Receive Data Register 1
SSI 1	0x1001 0010	SCR	SSI Control Register
SSI 1	0x1001 0014	SISR	SSI Interrupt Status Register
SSI 1	0x1001 0018	SIER	SSI Interrupt Enable Register
SSI 1	0x1001 001C	STCR	SSI Transmit Configuration Register
SSI 1	0x1001 0020	SRCR	SSI Receive Configuration Register
SSI 1	0x1001 0024	STCCR	SSI Transmit Clock Control Register
SSI 1	0x1001 0028	SRCCR	SSI Receive Clock Control Register
SSI 1	0x1001 002C	SFCSR	SSI FIFO Control/Status Register
SSI 1	0x1001 0030	STR	SSI Test Register
SSI 1	0x1001 0034	SOR	SSI Option Register
SSI 1	0x1001 0038	SACNT	SSI AC97 Control Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
SSI 1	0x1001 003C	SACADD	SSI AC97 Command Address Register
SSI 1	0x1001 0040	SACDAT	SSI AC97 Command Data Register
SSI 1	0x1001 0044	SATAG	SSI AC97 Tag Register
SSI 1	0x1001 0048	STMSK	SSI Transmit Time Slot Mask Register
SSI 1	0x1001 004C	SRMSK	SSI Receive Time Slot Mask Register
SSI 1	0x1001 0050	SACCST	SSI AC97 Channel Status Register
SSI 1	0x1001 0054	SACCEN	SSI AC97 Channel Enable Register
SSI 1	0x1001 0058	SACCDIS	SSI AC97 Channel Disable Register
SSI 2	0x1001 1000	STX0	SSI Transmit Data Register 0
SSI 2	0x1001 1004	STX1	SSI Transmit Data Register 1
SSI 2	0x1001 1008	SRX0	SSI Receive Data Register 0
SSI 2	0x1001 100C	SRX1	SSI Receive Data Register 1
SSI 2	0x1001 1010	SCR	SSI Control Register
SSI 2	0x1001 1014	SISR	SSI Interrupt Status Register
SSI 2	0x1001 1018	SIER	SSI Interrupt Enable Register
SSI 2	0x1001 101C	STCR	SSI Transmit Configuration Register
SSI 2	0x1001 1020	SRCR	SSI Receive Configuration Register
SSI 2	0x1001 1024	STCCR	SSI Transmit Clock Control Register
SSI 2	0x1001 1028	SRCCR	SSI Receive Clock Control Register
SSI 2	0x1001 102C	SFCSR	SSI FIFO Control/Status Register
SSI 2	0x1001 1030	STR	SSI Test Register
SSI 2	0x1001 1034	SOR	SSI Option Register
SSI 2	0x1001 1038	SACNT	SSI AC97 Control Register
SSI 2	0x1001 103C	SACADD	SSI AC97 Command Address Register
SSI 2	0x1001 1040	SACDAT	SSI AC97 Command Data Register
SSI 2	0x1001 1044	SATAG	SSI AC97 Tag Register
SSI 2	0x1001 1048	STMSK	SSI Transmit Time Slot Mask Register
SSI 2	0x1001 104C	SRMSK	SSI Receive Time Slot Mask Register
SSI 2	0x1001 1050	SACCST	SSI AC97 Channel Status Register
SSI 2	0x1001 1054	SACCEN	SSI AC97 Channel Enable Register
SSI 2	0x1001 1058	SACCDIS	SSI AC97 Channel Disable Register
I2C 1	0x1001 2000	IADR	I2C Address Register
I2C 1	0x1001 2004	IFDR	I2C Frequency Divider Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
I2C 1	0x1001 2008	I2CR	I2C Control Register
I2C 1	0x1001 200C	I2SR	I2C Status Register
I2C 1	0x1001 2010	I2DR	I2C Data I/O Register
SDHC1	0x1001 3000	STR_STP_CLK	MMC/SD1 Clock Control Register
SDHC1	0x1001 3004	STATUS (Read Only)	MMC/SD1 Status Register
SDHC1	0x1001 3008	CLK_RATE	MMC/SD1 Clock Rate Register
SDHC1	0x1001 300C	CMD_DAT_CONT	MMC/SD1 Command and Data Control Register
SDHC1	0x1001 3010	RESPONSE_TO	MMC/SD1 Response Time Out Register
SDHC1	0x1001 3014	READ_TO	MMC/SD1 Read Time Out Register
SDHC1	0x1001 3018	BLK_LEN	MMC/SD1 Block Length Register
SDHC1	0x1001 301C	NOB	MMC/SD1 Number of Block Register
SDHC1	0x1001 3020	REV_NO	MMC/SD1 Revision Number Register
SDHC1	0x1001 3024	INT_CNTL	MMC/SD1 Interrupt Control Register
SDHC1	0x1001 3028	CMD	MMC/SD1 Command Number Register
SDHC1	0x1001 302C	ARGH	MMC/SD1 Higher Argument Register
SDHC1	0x1001 3030	ARGL	MMC/SD1 Lower Argument Register
SDHC1	0x1001 3034	RES_FIFO (Read Only)	MMC/SD1 Response FIFO Register
SDHC1	0x1001 3038	BUFFER_ACCESS	MMC/SD1 Buffer Access Register
SDHC2	0x1001 4000	STR_STP_CLK	MMC/SD2 Clock Control Register
SDHC2	0x1001 4004	STATUS (Read Only)	MMC/SD2 Status Register
SDHC2	0x1001 4008	CLK_RATE	MMC/SD2 Clock Rate Register
SDHC2	0x1001 400C	CMD_DAT_CONT	MMC/SD2 Command and Data Control Register
SDHC2	0x1001 4010	RESPONSE_TO	MMC/SD2 Response Time Out Register
SDHC2	0x1001 4014	READ_TO	MMC/SD2 Read Time Out Register
SDHC2	0x1001 4018	BLK_LEN	MMC/SD2 Block Length Register
SDHC2	0x1001 401C	NOB	MMC/SD2 Number of Block Register
SDHC2	0x1001 4020	REV_NO	MMC/SD2 Revision Number Register
SDHC2	0x1001 4024	INT_CNTL	MMC/SD2 Interrupt Control Register
SDHC2	0x1001 4028	CMD	MMC/SD2 Command Number Register
SDHC2	0x1001 402C	ARGH	MMC/SD2 Higher Argument Register
SDHC2	0x1001 4030	ARGL	MMC/SD2 Lower Argument Register
SDHC2	0x1001 4034	RES_FIFO (Read Only)	MMC/SD2 Response FIFO Register
SDHC2	0x1001 4038	BUFFER_ACCESS	MMC/SD2 Buffer Access Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
GPIO	0x1001 5000	PTA_DDIR	Data Direction Register, Port A
GPIO	0x1001 5004	PTA_OCR1	Output Configuration Register 1 (OCR1), Port A
GPIO	0x1001 5008	PTA_OCR2	Output Configuration Register 2 (OCR2), Port A
GPIO	0x1001 500C	PTA_ICONFA1	Input Configuration Register A1 (ICONFA1), Port A
GPIO	0x1001 5010	PTA_ICONFA2	Input Configuration Register A1 (ICONFA2), Port A
GPIO	0x1001 5014	PTA_ICONFB1	Input Configuration Register B1 (ICONFB1), Port A
GPIO	0x1001 5018	PTA_ICONFB2	Input Configuration Register B2 (ICONFB2), Port A
GPIO	0x1001 501c	PTA_DR	Data Register, Port A
GPIO	0x1001 5020	PTA_GIUS	GPIO In Use Register, Port A
GPIO	0x1001 5024	PTA_SSR	Sample Status Register, Port A
GPIO	0x1001 5028	PTA_ICR1	Interrupt Configuration Register 1, Port A
GPIO	0x1001 502C	PTA_ICR2	Interrupt Configuration Register 2, Port A
GPIO	0x1001 5030	PTA_IMR	Interrupt Mask Register, Port A
GPIO	0x1001 5034	PTA_ISR	Interrupt Status Register, Port A
GPIO	0x1001 5038	PTA_GPR	General Purpose Register, Port A
GPIO	0x1001 503c	PTA_SWR	Software Reset Register, Port A
GPIO	0x1001 5040	PTA_PUEN	Pull_up Enable Register, Port A
GPIO	0x1001_5100	PTB_DDIR	Data Direction Register, Port B
GPIO	0x1001 5104	PTB_OCR1	Output Configuration Register 1 (OCR1), Port B
GPIO	0x1001 5108	PTB_OCR2	Output Configuration Register 2 (OCR2), Port B
GPIO	0x1001 510c	PTB_ICONFA1	Input Configuration Register A1 (ICONFA1), Port B
GPIO	0x1001 5110	PTB_ICONFA2	Input Configuration Register A1 (ICONFA2), Port B
GPIO	0x1001 5114	PTB_ICONFB1	Input Configuration Register B1 (ICONFB1), Port B
GPIO	0x1001 5118	PTB_ICONFB2	Input Configuration Register B2 (ICONFB2), Port B
GPIO	0x1001 511c	PTB_DR	Data Register, Port B
GPIO	0x1001 5120	PTB_GIUS	GPIO In Use Register, Port B
GPIO	0x1001 5124	PTB_SSR	Sample Status Register, Port B
GPIO	0x1001 5128	PTB_ICR1	Interrupt Configuration Register 1, Port B
GPIO	0x1001 512C	PTB_ICR2	Interrupt Configuration Register 2, Port B
GPIO	0x1001 5130	PTB_IMR	Interrupt Mask Register, Port B
GPIO	0x1001 5134	PTB_ISR	Interrupt Status Register, Port B
GPIO	0x1001 5138	PTB_GPR	General Purpose Register, Port B
GPIO	0x1001 513c	PTB_SWR	Software Reset Register, Port B

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
GPIO	0x1001 5140	PTB_PUEN	Pull_up Enable Register, Port B
GPIO	0x1001_5200	PTC_DDIR	Data Direction Register, Port C
GPIO	0x1001 5204	PTC_OCR1	Output Configuration Register 1 (OCR1), Port C
GPIO	0x1001 5208	PTC_OCR2	Output Configuration Register 2 (OCR2), Port C
GPIO	0x1001 520c	PTC_ICONFA1	Input Configuration Register A1 (ICONFA1), Port C
GPIO	0x1001 5210	PTC_ICONFA2	Input Configuration Register A1 (ICONFA2), Port C
GPIO	0x1001 5214	PTC_ICONFB1	Input Configuration Register B1 (ICONFB1), Port C
GPIO	0x1001 5218	PTC_ICONFB2	Input Configuration Register B2 (ICONFB2), Port C
GPIO	0x1001 521C	PTC_DR	Data Register, Port C
GPIO	0x1001 5220	PTC_GIUS	GPIO In Use Register, Port C
GPIO	0x1001 5224	PTC_SSR	Sample Status Register, Port C
GPIO	0x1001 5228	PTC_ICR1	Interrupt Configuration Register 1, Port C
GPIO	0x1001 522C	PTC_ICR2	Interrupt Configuration Register 2, Port C
GPIO	0x1001 5230	PTC_IMR	Interrupt Mask Register, Port C
GPIO	0x1001 5234	PTC_ISR	Interrupt Status Register, Port C
GPIO	0x1001 5238	PTC_GPR	General Purpose Register, Port C
GPIO	0x1001 523c	PTC_SWR	Software Reset Register, Port C
GPIO	0x1001 5240	PTC_PUEN	Pull_up Enable Register, Port C
GPIO	0x1001 5300	PTD_DDIR	Data Direction Register, Port D
GPIO	0x1001 5304	PTD_OCR1	Output Configuration Register 1 (OCR1), Port D
GPIO	0x1001 5308	PTD_OCR2	Output Configuration Register 2 (OCR2), Port D
GPIO	0x1001 530c	PTD_ICONFA1	Input Configuration Register A1 (ICONFA1), Port D
GPIO	0x1001 5310	PTD_ICONFA2	Input Configuration Register A1 (ICONFA2), Port D
GPIO	0x1001 5314	PTD_ICONFB1	Input Configuration Register B1 (ICONFB1), Port D
GPIO	0x1001 5318	PTD_ICONFB2	Input Configuration Register B2 (ICONFB2), Port D
GPIO	0x1001 531c	PTD_DR	Data Register, Port D
GPIO	0x1001 5320	PTD_GIUS	GPIO In Use Register, Port D
GPIO	0x1001 5324	PTD_SSR	Sample Status Register, Port D
GPIO	0x1001 5328	PTD_ICR1	Interrupt Configuration Register 1, Port D
GPIO	0x1001 532C	PTD_ICR2	Interrupt Configuration Register 2, Port D
GPIO	0x1001 5330	PTD_IMR	Interrupt Mask Register, Port D
GPIO	0x1001 5334	PTD_ISR	Interrupt Status Register, Port D
GPIO	0x1001 5338	PTD_GPR	General Purpose Register, Port D

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
GPIO	0x1001 533c	PTD_SWR	Software Reset Register, Port D
GPIO	0x1001 5340	PTD_PUEN	Pull_up Enable Register, Port D
GPIO	0x1001 5400	PTE_DDIR	Data Direction Register, Port E
GPIO	0x1001 5404	PTE_OCR1	Output Configuration Register 1 (OCR1), Port E
GPIO	0x1001 5408	PTE_OCR2	Output Configuration Register 2 (OCR2), Port E
GPIO	0x1001 540c	PTE_ICONFA1	Input Configuration Register A1 (ICONFA1), Port E
GPIO	0x1001 5410	PTE_ICONFA2	Input Configuration Register A1 (ICONFA2), Port E
GPIO	0x1001 5414	PTE_ICONFB1	Input Configuration Register B1 (ICONFB1), Port E
GPIO	0x1001 5418	PTE_ICONFB2	Input Configuration Register B2 (ICONFB2), Port E
GPIO	0x1001 541c	PTE_DR	Data Register, Port E
GPIO	0x1001 5420	PTE_GIUS	GPIO In Use Register, Port E
GPIO	0x1001 5424	PTE_SSR	Sample Status Register, Port E
GPIO	0x1001 5428	PTE_ICR1	Interrupt Configuration Register 1, Port E
GPIO	0x1001 542C	PTE_ICR2	Interrupt Configuration Register 2, Port E
GPIO	0x1001 5430	PTE_IMR	Interrupt Mask Register, Port E
GPIO	0x1001 5434	PTE_ISR	Interrupt Status Register, Port E
GPIO	0x1001 5438	PTE_GPR	General Purpose Register, Port E
GPIO	0x1001 543c	PTE_SWR	Software Reset Register, Port E
GPIO	0x1001 5440	PTE_PUEN	Pull_up Enable Register, Port E
GPIO	0x1001 5500	PTF_DDIR	Data Direction Register, Port F
GPIO	0x1001 5504	PTF_OCR1	Output Configuration Register 1 (OCR1), Port F
GPIO	0x1001 5508	PTF_OCR2	Output Configuration Register 2 (OCR2), Port F
GPIO	0x1001 550C	PTF_ICONFA1	Input Configuration Register A1 (ICONFA1), Port F
GPIO	0x1001 5510	PTF_ICONFA2	Input Configuration Register A1 (ICONFA2), Port F
GPIO	0x1001 5514	PTF_ICONFB1	Input Configuration Register B1 (ICONFB1), Port F
GPIO	0x1001 5518	PTF_ICONFB2	Input Configuration Register B2 (ICONFB2), Port F
GPIO	0x1001 551c	PTF_DR	Data Register, Port F
GPIO	0x1001 5520	PTF_GIUS	GPIO In Use Register, Port F
GPIO	0x1001 5524	PTF_SSR	Sample Status Register, Port F
GPIO	0x1001 5528	PTF_ICR1	Interrupt Configuration Register 1, Port F
GPIO	0x1001 552C	PTF_ICR2	Interrupt Configuration Register 2, Port F
GPIO	0x1001 5530	PTF_IMR	Interrupt Mask Register, Port F
GPIO	0x1001 5534	PTF_ISR	Interrupt Status Register, Port F

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
GPIO	0x1001 5538	PTF_GPR	General Purpose Register, Port F
GPIO	0x1001 553c	PTF_SWR	Software Reset Register, Port F
GPIO	0x1001 5540	PTF_PUEN	Pull_up Enable Register, Port F
GPIO	0x1001 5600	PMASK	GPIO Port Interrupt Mask
AUDMUX	0x1001 6000	HPCR1	Host Port Configuration Register 1
AUDMUX	0x1001 6004	HPCR2	Host Port Configuration Register 2
AUDMUX	0x1001 6008	HPCR3	Host Port Configuration Register 3
AUDMUX	0x1001 6010	PPCR1	Peripheral Port Configuration Register 1
AUDMUX	0x1001 6014	PPCR2	Peripheral Port Configuration Register 2
AUDMUX	0x1001 601C	PPCR3	Peripheral Port Configuration Register 3
CSPI3	0x1001 7000	RXDATA3	Receive Data Register 3
CSPI3	0x1001 7004	TXDATA3	Transmit Data Register 3
CSPI3	0x1001 7008	CONTROL_REG3	CSPI Control Register 3
CSPI3	0x1001 700C	INT_REG3	Interrupt Control/Status Register 3
CSPI3	0x1001 7010	TEST_REG3	CSPI Test Register 3
CSPI3	0x1001 7014	PERIOD3	CSPI Sample Period Control Register 3
CSPI3	0x1001 7018	CSPI_DMA3	CSPI DMA Register 3
CSPI3	0x1001 701C	CSPI_RESET3	CSPI Soft Reset Register 3
MSHC	0x1001 8000	COMMAND_REG	MSHC Command Register
MSHC	0x1001 8008	DATA_REG	MSHC Data Register
MSHC	0x1001 8010	STATUS_REG	MSHC Status Register
MSHC	0x1001 8018	SYSTEM_REG	MSHC System Register
GPT4	0x1001 9000	TCTL4	GPT Control Register 4
GPT4	0x1001 9004	TPRER4	GPT Prescaler Register 4
GPT4	0x1001 9008	TCMP4	GPT Compare Register 4
GPT4	0x1001 900C	TCR4	GPT Capture Register 4
GPT4	0x1001 9010	TCN4	GPT Counter Register 4
GPT4	0x1001 9014	TSTAT4	GPT Status Register 4
GPT5	0x1001 A000	TCTL5	GPT Control Register 5
GPT5	0x1001 A004	TPRER5	GPT Prescaler Register 5
GPT5	0x1001 A008	TCMP5	GPT Compare Register 5
GPT5	0x1001 A00C	TCR5	GPT Capture Register 5
GPT5	0x1001 A010	TCN5	GPT Counter Register 5

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
GPT5	0x1001 A014	TSTAT5	GPT Status Register 5
UART5	0x1001_B000	UXRD_5	UART5 Receiver Register
UART5	0x1001_B040	UTXD_5	UART5 Transmitter Register
UART5	0x1001_B080	UCR1_5	UART5 Control Register
UART5	0x1001_B084	UCR2_5	UART5 Control Register 2
UART5	0x1001_B088	UCR3_5	UART5 Control Register 3
UART5	0x1001_B08C	UCR4_5	UART5 Control Register 4
UART5	0x1001_B090	UFCR_5	UART5 FIFO Control Register
UART5	0x1001_B094	USR1_5	UART5 Status Register 1
UART5	0x1001_B098	USR2_5	UART5 Status Register 2
UART5	0x1001_B09C	UESC_5	UART5 Escape Character Register
UART5	0x1001_B0A0	UTIM_5	UART5 Escape Timer Register
UART5	0x1001_B0A4	UBIR_5	UART5 BRM Incremental Register
UART5	0x1001_B0A8	UBMR_5	UART5 BRM Modulator Register
UART5	0x1001_B0AC	UBRC_5	UART5 Baud Rate Count Register
UART5	0x1001_B0B0	ONEMS_5	UART5 One Millisecond Register
UART5	0x1001_B0B4	UTS_5	UART5 Test Register 1
UART6	0x1001_C000	UXRD_6	UART6 Receiver Register
UART6	0x1001_C040	UTXD_6	UART6 Transmitter Register
UART6	0x1001_C080	UCR1_6	UART6 Control Register
UART6	0x1001_C084	UCR2_6	UART6 Control Register 2
UART6	0x1001_C088	UCR3_6	UART6 Control Register 3
UART6	0x1001_C08C	UCR4_6	UART6 Control Register 4
UART6	0x1001_C090	UFCR_6	UART6 FIFO Control Register
UART6	0x1001_C094	USR1_6	UART6 Status Register 1
UART6	0x1001_C098	USR2_6	UART6 Status Register 2
UART6	0x1001_C09C	UESC_6	UART6 Escape Character Register
UART6	0x1001_C0A0	UTIM_6	UART6 Escape Timer Register
UART6	0x1001_C0A4	UBIR_6	UART6 BRM Incremental Register
UART6	0x1001_C0A8	UBMR_6	UART6 BRM Modulator Register
UART6	0x1001_C0AC	UBRC_6	UART6 Baud Rate Count Register
UART6	0x1001_C0B0	ONEMS_6	UART6 One Millisecond Register
UART6	0x1001_C0B4	UTS_6	UART6 Test Register 1

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
I2C 2	0x1001 D000	IADR	I2C Address Register
I2C 2	0x1001 D004	IFDR	I2C Frequency Divider Register
I2C 2	0x1001 D008	I2CR	I2C Control Register
I2C 2	0x1001 D00C	I2SR	I2C Status Register
I2C 2	0x1001 D010	I2DR	I2C Data I/O Register
SDHC3	0x1001 E000	STR_STP_CLK	MMC/SD2 Clock Control Register
SDHC3	0x1001 E004	STATUS (Read Only)	MMC/SD2 Status Register
SDHC3	0x1001 E008	CLK_RATE	MMC/SD2 Clock Rate Register
SDHC3	0x1001 E00C	CMD_DAT_CONT	MMC/SD2 Command and Data Control Register
SDHC3	0x1001 E010	RESPONSE_TO	MMC/SD2 Response Time Out Register
SDHC3	0x1001 E014	READ_TO	MMC/SD2 Read Time Out Register
SDHC3	0x1001 E018	BLK_LEN	MMC/SD2 Block Length Register
SDHC3	0x1001 E01C	NOB	MMC/SD2 Number of Block Register
SDHC3	0x1001 E020	REV_NO	MMC/SD2 Revision Number Register
SDHC3	0x1001 E024	INT_CNTL	MMC/SD2 Interrupt Control Register
SDHC3	0x1001 E028	CMD	MMC/SD2 Command Number Register
SDHC3	0x1001 E02C	ARGH	MMC/SD2 Higher Argument Register
SDHC3	0x1001 E030	ARGL	MMC/SD2 Lower Argument Register
SDHC3	0x1001 E034	RES_FIFO (Read Only)	MMC/SD2 Response FIFO Register
SDHC3	0x1001 E038	BUFFER_ACCESS	MMC/SD2 Buffer Access Register
GPT6	0x1001 F000	TCTL6	GPT Control Register 6
GPT6	0x1001 F004	TPRER6	GPT Prescaler Register 6
GPT6	0x1001 F008	TCMP6	GPT Compare Register 6
GPT6	0x1001 F00C	TCR6	GPT Capture Register 6
GPT6	0x1001 F010	TCN6	GPT Counter Register 6
GPT6	0x1001 F014	TSTAT6	GPT Status Register 6
AIPI2	0x1002 0000	PSR0	Peripheral Size Register0
AIPI2	0x1002 0004	PSR1	Peripheral Size Register1
AIPI2	0x1002 0008	PAR	Peripheral Access Register
AIPI2	0x1002 000C	AAOR	Atomic Access Only Register
LCDC	0x1002 1000	LSSAR	LCDC Screen Start Address Register
LCDC	0x1002 1004	LSR	LCDC Size Register
LCDC	0x1002 1008	LVPWR	LCDC Virtual Page Width Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
LCDC	0x1002 100C	LCPR	LCDC Cursor Position Register
LCDC	0x1002 1010	LCWHBR	LCDC Cursor Width Height and Blink Register
LCDC	0x1002 1014	LCCMR	LCDC Color Cursor Mapping Register
LCDC	0x1002 1018	LPCR	LCDC Panel Configuration Register
LCDC	0x1002 101C	LHCR	LCDC Horizontal Configuration Register
LCDC	0x1002 1020	LVCR	LCDC Vertical Configuration Register
LCDC	0x1002 1024	LPOR	LCDC Panning Offset Register
LCDC	0x1002 1028	LSCR	LCDC Sharp Configuration Register
LCDC	0x1002 102C	LPCCR	LCDC PWM Contrast Control Register
LCDC	0x1002 1030	LDCR	LCDC DMA Control Register
LCDC	0x1002 1034	LRMCR	LCDC Refresh Mode Control Register
LCDC	0x1002 1038	LICR	LCDC Interrupt Configuration Register
LCDC	0x1002 103C	LIER	LCDC Interrupt Enable Register
LCDC	0x1002 1040	LISR	LCDC Interrupt Status Register
LCDC	0x1002 1050	LGWSAR	LCDC Graphic Window Start Address Register
LCDC	0x1002 1054	LGWSR	LCDC Graphic Window Size Register
LCDC	0x1002 1058	LGWVPWR	LCDC Graphic Window Virtual Page Width Register
LCDC	0x1002 105C	LGWPOR	LCDC Graphic Window Panning Offset Register
LCDC	0x1002 1060	LGWPR	LCDC Graphic Window Position Register
LCDC	0x1002 1064	LGWCR	LCDC Graphic Window Control Register
LCDC	0x1002 1068	LGWDCR	LCDC Graphic Window DMA Control Register
LCDC	0x1002 1080	LAUSCR	LCDC Aus mode Control Register
LCDC	0x1002 1084	LAUSCCR	LCDC Aus mode Cursor Control Register
SLCDC	0x1002 2000	DATA_BASE_ADDR	SLCD Data Base Address Register
SLCDC	0x1002 2004	DATA_BUFF_SIZE	SLCD Data Buffer Size Register
SLCDC	0x1002 2008	CMD_BASE_ADDR	SLCD Command Buffer Base Address Register
SLCDC	0x1002 200C	CMD_BUFF_SIZE	SLCD Command Buffer Size Register
SLCDC	0x1002 2010	CMD_STR_SIZE	SLCD Command String Size Register
SLCDC	0x1002 2014	FIFO_CONFIG	SLCD FIFO Configuration Register
SLCDC	0x1002 2018	LCD_CONFIG	SLCD Configuration Register
SLCDC	0x1002 201C	LCD_XFER_CONFIG	SLCD Transfer Configuration Register
SLCDC	0x1002 2020	DMA_CTRL_STAT	SLCD DMA Control/Status Register
SLCDC	0x1002 2024	LCD_CLK_CONFIG	SLCD Clock Configuration Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
SLCDC	0x1002 2028	LCD_WRITE_DATA	SLCD Write Data Register
Video Codec	0x1002 3000	CodeRun	BIT run start
Video Codec	0x1002 3004	CodeDownLoad	Code Download Data Register
Video Codec	0x1002 3008	HostIntReq	Host Interrupt Request to BI
Video Codec	0x1002 300C	BitIntClear	BIT Interrupt Clear
Video Codec	0x1002 3010	BitIntSts	BIT Interrupt Status
Video Codec	0x1002 3100	WorkBufAddr	Working Buffer Address in External Memory
Video Codec	0x1002 3104	CodeBufAddr	Code Table Size in External Memory
Video Codec	0x1002 3108	BitStreamCtrl	Bit Stream Control
Video Codec	0x1002 310C	FrameMemCtrl	Frame Memory Control
Video Codec	0x1002 3110	SramAddr	Internal SRAM Base Address
Video Codec	0x1002 3114	SramSize	Internal SRAM Size
Video Codec	0x1002 3140	BitStreamRdPtr	Bit Stream Buffer Read Address
Video Codec	0x1002 3144	BitStreamWrPtr	Bit Stream Buffer Write Address
Video Codec	0x1002 3148	FrameNum	Encoded/Decoded Frame Number
Video Codec	0x1002 3160	BusyFlag	Processing Busy Flag
Video Codec	0x1002 3164	RunCommand	Start/Stop Codec Run Command
Video Codec	0x1002 3168	RunIndex	Run Process Index
Video Codec	0x1002 316C	RunCodStd	Run Codec Standard
Video Codec	0x1002 3180	BitBufAddr	Parameter Registers in sequence initialization. Bitstream Buffer Address
Video Codec	0x1002 3184	BitBufSize	Parameter Registers in sequence initialization. Bitstream Buffer Size
Video Codec	0x1002 3188	FrameIntAddrY	Parameter Registers in sequence initialization. Temporal Frame Y Address
Video Codec	0x1002 318C	FrameIntAddrCb	Parameter Registers in sequence initialization. Temporal Frame Cb Address
Video Codec	0x1002 3190	FrameIntAddrCr	Parameter Registers in sequence initialization. Temporal Frame Cr Address
Video Codec	0x1002 3194	EncCodStd	Parameter Registers in sequence initialization. Encode Coding Standard
Video Codec	0x1002 3198	EncSrcFormat	Parameter Registers in sequence initialization. Encode Source Frame Format
Video Codec	0x1002 319C	EncMp4Para	Parameter Registers in sequence initialization. Encode MPEG4 Parameter

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
Video Codec	0x1002 31A0	Enc263Para	Parameter Registers in sequence initialization. Encode H.263 Parameter
Video Codec	0x1002 31A4	Enc264Para	Parameter Registers in sequence initialization. Encode H.264 Parameter
Video Codec	0x1002 31A8	EncSliceMode	Parameter Registers in sequence initialization. Encode Slice Mode
Video Codec	0x1002 31AC	EncGopNum	Parameter Registers in sequence initialization. Encode GOP Number
Video Codec	0x1002 31B0	EncPictureQs	Parameter Registers in sequence initialization. Encode Picture Quantize Step
Video Codec	0x1002 31C0	RetStatus	Parameter Registers in sequence initialization. Command Executing Result Status
Video Codec	0x1002 31C4	RetSrcFormat	Parameter Registers in sequence initialization. Decoded Source Format
Video Codec	0x1002 31C8	RetMp4Info	Parameter Registers in sequence initialization. Decoded MPEG4 Sequence Information
Video Codec	0x1002 31CC	Ret263Info	Parameter Registers in sequence initialization. Decoded H.263 Sequence Information
Video Codec	0x1002 31D0	Ret264Info	Parameter Registers in sequence initialization. Decoded H.264 Sequence Information
Video Codec	0x1002 3180	FrameSrcAddrY	Parameter Register in Processing Running. Source Frame Y Address
Video Codec	0x1002 3184	FrameSrcAddrCb	Parameter Register in Processing Running. Source Frame Cb Address
Video Codec	0x1002 3188	FrameSrcAddrCr	Parameter Register in Processing Running. Source Frame Cr Address
Video Codec	0x1002 318C	FrameDecAddrY	Parameter Register in Processing Running. Decode Frame Y Address
Video Codec	0x1002 3190	FrameDecAddrCb	Parameter Register in Processing Running. Decode Frame Cb Address
Video Codec	0x1002 3194	FrameDecAddrCr	Parameter Register in Processing Running. Decode Frame Cr Address
Video Codec	0x1002 31C0	RetStatus	Parameter Register in Processing Running. Command Executing Result Status
USBOTG	0x1002 4000	UOG_ID	ID (UOG_ID)
USBOTG	0x1002 4004	UOG_HWGENERAL	Hardware General (UOG_HWGENERAL)
USBOTG	0x1002 4008	UOG_HWHOST	Host Hardware Parameters (UOG_HWHOST)
USBOTG	0x1002 400C	UOG_HWDEVICE	Device Hardware Parameters (UOG_HWDEVICE)
USBOTG	0x1002 4010	UOG_HWTXBUF	TX Buffer Hardware Parameters (UOG_HWTXBUF)

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
USBOTG	0x1002 4014	UOG_HWRXBUF	RX Buffer Hardware Parameters (UOG_HWRXBUF)
USBOTG	0x1002 4080	GPTIMER0LD	General Purpose Timer #0 Load (GPTIMER0LD)
USBOTG	0x1002 4084	GPTIMER0CTRL	General Purpose Timer #0 Controller (GPTIMER0CTRL)
USBOTG	0x1002 4088	GPTIMER0LD	General Purpose Timer #1 Load (GPTIMER0LD)
USBOTG	0x1002 408C	GPTIMER0CTRL	General Purpose Timer #1 Controller (GPTIMER0CTRL)
USBOTG	0x1002 4100	UOG_CAPLENGTH	Capability Register Length (UOG_CAPLENGTH)
USBOTG	0x1002 4102	UOG_HCIVERSION	Host Interface Version (UOG_HCIVERSION)
USBOTG	0x1002 4104	UOG_HCSPARAMS	Host Control Structural Parameters (UOG_HCSPARAMS)
USBOTG	0x1002 4108	UOG_HCCPARAMS	Control Capability Parameters (UOG_HCCPARAMS)
USBOTG	0x1002 4120	UOG_DCIVERSION	Device Interface Version (UOG_DCIVERSION)
USBOTG	0x1002 4124	UOG_DCCPARAMS	Device Controller Capability Parameters (UOG_DCCPARAMS)
USBOTG	0x1002 4140	UOG_USBCMD	USB Command Register (UOG_USBCMD)
USBOTG	0x1002 4144	UOG_USBSTS	USB Status Register (UOG_USBSTS)
USBOTG	0x1002 4148	UOG_USBINTR	Interrupt Enable Register (UOG_USBINTR)
USBOTG	0x1002 414C	UOG_FRINDEX	USB Frame Index (UOG_FRINDEX)
USBOTG	0x1002 4154	UOG_PERIODICLISTBASE	Host Controller Frame List Base Address (UOG_PERIODICLISTBASE)
USBOTG	0x1002 4158	UOG_ASYNCLISTADDR	Host Controller Next Asynch. Address (UOG_ASYNCLISTADDR)
USBOTG	0x1002 4160	UOG_BURSTSIZE	Host Controller Embedded TT Asynch. Buffer Status (UOG_BURSTSIZE)
USBOTG	0x1002 4164	UOG_TXFILLTUNING	TX FIFO Fill Tuning (UOG_TXFILLTUNING)
USBOTG	0x1002 4170	ULPIVIEW	ULPI Viewport (ULPIVIEW)
USBOTG	0x1002 4180	UOG_CFGFLAG	Config Flag (UOG_CFGFLAG)
USBOTG	0x1002 4184	UOG_PORTSC1	Port Status and Control (UOG_PORTSC1)
USBOTG	0x1002 41A4	UOG_OTGSC	On-The-Go Status and control (UOG_OTGSC)
USBOTG	0x1002 41A8	UOG_USBMODE	USB Device Mode (UOG_USBMODE)
USBOTG	0x1002 41AC	UOG_ENDPTSETUPSTAT	Endpoint Setup Status (UOG_ENDPTSETUPSTAT)
USBOTG	0x1002 41B0	UOG_ENDPTPRIME	Endpoint Initialization (UOG_ENDPTPRIME)
USBOTG	0x1002 41B4	UOG_ENDPTFLUSH	Endpoint De-Initialize (UOG_ENDPTFLUSH)
USBOTG	0x1002 41B8	UOG_ENDPTSTAT	Endpoint Status (UOG_ENDPTSTAT)

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
USBOTG	0x1002 41BC	UOG_ENDPTCOMPLETE	Endpoint Complete (UOG_ENDPTCOMPLETE)
USBOTG	0x1002 41C0	ENDPTCTRL0	Endpoint Control 0 (ENDPTCTRL0)
USBOTG	0x1002 41C4	ENDPTCTRL1	Endpoint Control 1 (ENDPTCTRL1)
USBOTG	0x1002 41C8	ENDPTCTRL2	Endpoint Control 2 (ENDPTCTRL2)
USBOTG	0x1002 41CC	ENDPTCTRL3	Endpoint Control 3 (ENDPTCTRL3)
USBOTG	0x1002 41D0	ENDPTCTRL4	Endpoint Control 4 (ENDPTCTRL4)
USB OTG	0x1002 41D4	ENDPTCTRL5	Endpoint Control 5 (ENDPTCTRL5)
USB OTG	0x1002 41D8	ENDPTCTRL6	Endpoint Control 6 (ENDPTCTRL6)
USBOTG	0x1002 41DC	ENDPTCTRL7	Endpoint Control 7 (ENDPTCTRL7)
USBOTG	0x1002 4200	UH1_ID	Host 1 ID (UH1_ID)
USBOTG	0x1002 4204	UH1_HWGENERAL	Hardware General (UH1_HWGENERAL)
USBOTG	0x1002 4208	UH1_HWHOST	Host Hardware Parameters (UH1_HWHOST)
USBOTG	0x1002 4210	UH1_HWTXBUF	TX Buffer Hardware Parameters (UH1_HWTXBUF)
USBOTG	0x1002 4214	UH1_HWRXBUF	RX Buffer Hardware Parameters (UH1_HWRXBUF)
USBOTG	0x1002 4280	GPTIMER0LD	General Purpose Timer #0 Load (GPTIMER0LD)
USBOTG	0x1002 4284	GPTIMER0CTRL	General Purpose Timer #0 Controller (GPTIMER0CTRL)
USBOTG	0x1002 4288	GPTIMER0LD	General Purpose Timer #1 Load (GPTIMER0LD)
USBOTG	0x1002 428C	GPTIMER0CTRL	General Purpose Timer #1 Controller (GPTIMER0CTRL)
USBOTG	0x1002 4300	UH1_CAPLENGTH	Capability Register Length (UH1_CAPLENGTH)
USBOTG	0x1002 4302	UH1_HCIVERSION	Host Interface Version (UH1_HCIVERSION)
USBOTG	0x1002 4304	UH1_HCSPARAMS	Host Control Structural Parameters (UH1_HCSPARAMS)
USBOTG	0x1002 4308	UH1_HCCPARAMS	Control Capability Parameters (UH1_HCCPARAMS)
USBOTG	0x1002 4340	UH1_USBCMD	USB Command Register (UH1_USBCMD)
USBOTG	0x1002 4344	UH1_USBSTS	USB Status Register (UH1_USBSTS)
USBOTG	0x1002 4348	UH1_USBINTR	Interrupt Enable Register (UH1_USBINTR)
USBOTG	0x1002 434C	UH1_FRINDEX	USB Frame Index (UH1_FRINDEX)
USBOTG	0x1002 4354	UH1_PERIODICLISTBASE	Host Controller Frame List Base Address (UH1_PERIODICLISTBASE)
USBOTG	0x1002 4358	UH1_ASYNCLISTADDR	Host Controller Next Asynch. Address (UH1_ASYNCLISTADDR)
USBOTG	0x1002 4360	UH1_BURSTSIZE	Host Controller Embedded TT Asynch. Buffer Status (UH1_BURSTSIZE)

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
USBOTG	0x1002 4364	UH1_TXFILLTUNING	TX FIFO Fill Tuning (UH1_TXFILLTUNING)
USBOTG	0x1002 4380		Reserved
USBOTG	0x1002 4838	Reserved	Port Status and Control (UH1_PORTSC1)
USBOTG	0x1002 4384	UH1_PORTSC1	Port Status and Control (UH1_PORTSC1)
USBOTG	0x1002 483C	Reserved	Reserved
USBOTG	0x1002 43A8	UH1_USBMODE	USB Device Mode (UH1_USBMODE)
USBOTG	0x1002 4400	UH2_ID	ID (UH2_ID)
USBOTG	0x1002 4404	UH2_HWGENERAL	Hardware General (UH2_HWGENERAL)
USBOTG	0x1002 4408	UH2_HWHOST	Host Hardware Parameters (UH2_HWHOST)
USBOTG	0x1002 4410	UH2_HWTXBUF	TX Buffer Hardware Parameters (UH2_HWTXBUF)
USBOTG	0x1002 4414	UH2_HWRXBUF	RX Buffer Hardware Parameters (UH2_HWRXBUF)
USBOTG	0x1002 4480	GPTIMER0LD	General Purpose Timer #0 Load (GPTIMER0LD)
USBOTG	0x1002 4484	GPTIMER0CTRL	General Purpose Timer #0 Controller (GPTIMER0CTRL)
USBOTG	0x1002 4488	GPTIMER0LD	General Purpose Timer #1 Load (GPTIMER0LD)
USBOTG	0x1002 448C	GPTIMER0CTRL	General Purpose Timer #1 Controller (GPTIMER0CTRL)
USBOTG	0x1002 4500	UH2_CAPLENGTH	Capability Register Length (UH2_CAPLENGTH)
USBOTG	0x1002 4502	UH2_HCIVERSION	Host Interface Version (UH2_HCIVERSION)
USBOTG	0x1002 4504	UH2_HCSPARAMS	Host Control Structural Parameters (UH2_HCSPARAMS)
USBOTG	0x1002 4508	UH2_HCCPARAMS	Control Capability Parameters (UH2_HCCPARAMS)
USBOTG	0x1002 4540	UH2_USBCMD	USB Command Register (UH2_USBCMD)
USBOTG	0x1002 4544	UH2_USBSTS	USB Status Register (UH2_USBSTS)
USBOTG	0x1002 4548	UH2_USBINTR	Interrupt Enable Register (UH2_USBINTR)
USBOTG	0x1002 454C	UH2_FRINDEX	USB Frame Index (UH2_FRINDEX)
USBOTG	0x1002 4554	UH2_PERIODICLISTBASE	Host Controller Frame List Base Address (UH2_PERIODICLISTBASE)
USBOTG	0x1002 4558	UH2_ASYNCLISTADDR	Host Controller Next Asynch. Address (UH2_ASYNCLISTADDR)
USBOTG	0x1002 4560	UH2_BURSTSIZE	Host Controller Embedded TT Asynch. Buffer Status (UH2_BURSTSIZE)
USBOTG	0x1002 4564	UH2_TXFILLTUNING	TX FIFO Fill Tuning (UH2_TXFILLTUNING)
USBOTG	0x1002 4570	ULPIVIEW	ULPI Viewport (ULPIVIEW)
USBOTG	0x1002 4580		Reserved

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
USBOTG	0x1002 4584	UH2_PORTSC1	Port Status and Control (UH2_PORTSC1)
USBOTG	0x1002 45A8	UH2_USBMODE	USB Device Mode (UH2_USBMODE)
USBOTG	0x1002 4600	USB_CTRL	USB Control Register (USB_CTRL)
USBOTG	0x1002 4604	USB_OTG_MIRROR	USB OTG Mirror Register (USB_OTG_MIRROR)
SAHARA2	0x1002 5000	VER_ID	SAHARA2 Version ID Register
SAHARA2	0x1002 5004	DSC_ADR	SAHARA2 Descriptor Address Register
SAHARA2	0x1002 5008	CONTROL	SAHARA2 Control Register
SAHARA2	0x1002 500C	COMMAND	SAHARA2 Command Register
SAHARA2	0x1002 5010	STAT	SAHARA2 Status Register
SAHARA2	0x1002 5014	ERR_STAT	SAHARA2 Error Status Register
SAHARA2	0x1002 5018	FAULT_ADR	SAHARA2 Fault Address Register
SAHARA2	0x1002 501C	C_DSC_ADR	SAHARA2 Current Descriptor Address Register
SAHARA2	0x1002 5020	I_DSC_ADR	SAHARA2 Initial Descriptor Address Register
SAHARA2	0x1002 5024	BUFF_LVL	SAHARA2 Buffer Level Register
SAHARA2	0x1002 5080	DSC_A	Location to Store Descriptor
SAHARA2	0x1002 5084	DSC_B	Location to Store Descriptor
SAHARA2	0x1002 5088	DSC_C	Location to Store Descriptor
SAHARA2	0x1002 508C	DSC_D	Location to Store Descriptor
SAHARA2	0x1002 5090	DSC_E	Location to Store Descriptor
SAHARA2	0x1002 5094	DSC_F	Location to Store Descriptor
SAHARA2	0x1002 50A0	LNK_1_A	Location to Store Link Data
SAHARA2	0x1002 50A4	LNK_1_B	Location to Store Link Data
SAHARA2	0x1002 50A8	LNK_1_C	Location to Store Link Data
SAHARA2	0x1002 50B0	LNK_2_A	Location to Store Link Data
SAHARA2	0x1002 50B4	LNK_2_B	Location to Store Link Data
SAHARA2	0x1002 50B8	LNK_2_C	Location to Store Link Data
SAHARA2	0x1002 50C0	FLOW_CTRL	SAHARA2 Internal Buffer and Data-Paths Control Register
SAHARA2	0x1002 5100	SKHA_MODE	SKHA Mode Register
SAHARA2	0x1002 5104	SKHA_KEY_SIZE	SKHA Key Size Register
SAHARA2	0x1002 5108	SKHA_DATA_SIZE	SKHA Data Size Register
SAHARA2	0x1002 510C	SKHA_STAT	SKHA Status Register
SAHARA2	0x1002 5110	SKHA_ERR_STAT	SKHA Error Status Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
SAHARA2	0x1002 5114	SKHA_End-of-Message	SKHA End-of-Message Register
SAHARA2	0x1002 5140–0x1002 517F	SKHA_CXT	SKHA Context Register
SAHARA2	0x1002 5180–0x1002 51FF	SKHA Key	SKHA Key Register
SAHARA2	0x1002 5200	MDHA_MODE	MDHA Mode Register
SAHARA2	0x1002 5204	MDHA_KEY_SIZE	MDHA Key Size Register
SAHARA2	0x1002 5208	MDHA_DATA_SIZE	MDHA Data Size Register
SAHARA2	0x1002 520C	MDHA_STAT	MDHA Status Register
SAHARA2	0x1002 5210	MDHA_ERR_STAT	MDHA Error Status Register
SAHARA2	0x1002 5214	MDHA_End-of-Message	MDHA End-of-Message Register
SAHARA2	0x1002 5240–0x1002 5254	MDHA_Digest and Length	MDHA Message Digest and Length Register
SAHARA2	0x1002 5280–0x1002 52FF	MDHA_Key	MDHA Keys
SAHARA2	0x1002 5300	RNG_Mode	RNG Mode Register
SAHARA2	0x1002 5308	RNG_Data_SIZE	RNG Data Size Register
SAHARA2	0x1002 530C	RNG_STAT	RNG Status Register
SAHARA2	0x1002 5310	RNG_ERROR_STAT	RNG Error Status Register
SAHARA2	0x1002 5314	RNG_End-of-Message	RNG End-of-Message Register
SAHARA2	0x1002 5340–0x1002 537F	RNG_VERIFICATION	RNG Verification Register
SAHARA2	0x1002 5380	RNG_ENTROPY	RNG Entropy Register
SAHARA2	0x1002 5400–0x1002 54FF	Data Input Buffer	Data Input Buffers
SAHARA2	0x1002 5500–0x1002 55FF	Data Output Buffer	Data Output Buffers
SAHARA2	0x1002 5600–0x1002 57FF	SBOX CONTEXT	SBOX Context
eMMA_lt	0x1002 6000	PP_CNTL	PP Control Register
eMMA_lt	0x1002 6004	PP_INTRCNTL	PP Interrupt Control Register
eMMA_lt	0x1002 6008	PP_INTRSTATUS	PP interrupt Status Register
eMMA_lt	0x1002 600C	PP_SOURCE_Y_PTR	PP Source “Y” FramE Data Pointer Register
eMMA_lt	0x1002 6010	PP_SOURCE_CB_PTR	PP Source “CB” Frame Data Pointer Register
eMMA_lt	0x1002 6014	PP_SOURCE_CR_PTR	PP Source “CR” Frame Data Pointer Register
eMMA_lt	0x1002 6018	PP_DEST_RGB_PTR	PP Destination “RGB” Frame Start Address Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
eMMA_lt	0x1002 601C	PP_QUANTIZER_PTR	PP Quantizer Start Address Register
eMMA_lt	0x1002 6020	PP_PROCESS_FRAME_PARA	PP Process Frame Parameter, Width And Height Register
eMMA_lt	0x1002 6024	PP_SOURCE_FRAME_WIDTH	PP Source Frame Width Register
eMMA_lt	0x1002 6028	PP_DEST_DISPLAY_WIDTH	PP Destination Display Width Register
eMMA_lt	0x1002 602C	PP_DEST_IMAGE_SIZE	PP Destination Image Size Register
eMMA_lt	0x1002 6030	PP_DEST_FRAME_FMT_CNTL	PP Destination Frame Format Control Register
eMMA_lt	0x1002 6034	PP Resize Table index Reg	PP Resize Table Index Register
eMMA_lt	0x1002 6038	PP_CSC_COEFF_012	PP CSC Coefficient 0, 1, and 2 Register
eMMA_lt	0x1002 603C	PP_CSC_COEFF_34	PP CSC Coefficient 3 and 4 Register
eMMA_lt	0x1002 6100–0x1002 617C	PP_RESIZE_COEF_TBL	PP Resize Coefficient Table Register
eMMA_lt	0x1002 6400	PrP_CNTL	PrP Control Register
eMMA_lt	0x1002 6404	PrP_INTRCNTL	PrP Interrupt Control Register
eMMA_lt	0x1002 6408	PrP_INTRSTATUS	PrP interrupt Status Register
eMMA_lt	0x1002 640C	PrP_SOURCE_Y_PTR	PrP Source “Y” Frame Start Address Register
eMMA_lt	0x1002 6410	PrP_SOURCE_CB_PTR	PrP Source “CB” Frame Start Address Register
eMMA_lt	0x1002 6414	PrP_SOURCE_CR_PTR	PrP Source “CR” Frame Start Address Register
eMMA_lt	0x1002 6418	PrP_DEST_RGB1_PTR	PrP Destination “RGB” Frame-1 Start Address Register
eMMA_lt	0x1002 641C	PrP_DEST_RGB2_PTR	PrP Destination “RGB” Frame-2 Start Address Register
eMMA_lt	0x1002 6420	PrP_DEST_Y_PTR	PrP Destination “Y” Frame Start Address Register
eMMA_lt	0x1002 6424	PrP_DEST_CB_PTR	PrP Destination “CB” Frame Start Address Register
eMMA_lt	0x1002 6428	PrP_DEST_CR_PTR	PrP Destination “CR” Frame Start Address Register
eMMA_lt	0x1002 642C	PrP_SOURCE_FRAME_SIZE	PrP Source Frame Size Register
eMMA_lt	0x1002 6430	PrP_CH1_LINE_STRIDE	PrP Channel-1 Line stride Register
eMMA_lt	0x1002 6434	PrP_SRC_PIXEL_FORMAT_CNTL	PrP Source Pixel Format Control Register
eMMA_lt	0x1002 6438	PrP_CH1_PIXEL_FORMAT_CNTL	PrP CH1 Pixel Format Control Register
eMMA_lt	0x1002 643C	PrP_CH1_OUT_IMAGE_SIZE	PrP CH1 Output Image Size Register
eMMA_lt	0x1002 6440	PrP_CH2_OUT_IMAGE_SIZE	PrP CH2 Output Image Size Register
eMMA_lt	0x1002 6444	PrP_SOURCE_LINE_STRIDE	PrP Source Line Stride Register
eMMA_lt	0x1002 6448	PrP_CSC_COEFF_012	PrP CSC Coefficients 0, 1, and 2 Register
eMMA_lt	0x1002 644C	PrP_CSC_COEFF_345	PrP CSC Coefficients 3, 4, and 5 Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
eMMA_lt	0x1002 6450	PrP_CSC_COEFF_678	PrP CSC Coefficients 6, 7, and 8 Register
eMMA_lt	0x1002 6454	PrP_CH1_HRESIZE_COEFF1	PrP CH1 Horizontal Resize Coefficients Register
eMMA_lt	0x1002 6458	PrP_CH1_HRESIZE_COEFF2	PrP CH1 Horizontal Resize Coefficients Register
eMMA_lt	0x1002 645C	PrP_CH1_HRESIZE_VALID	PrP CH1 Horizontal Resize Valid Register
eMMA_lt	0x1002 6460	PrP_CH1_VRESIZE_COEFF1	PrP CH1 Vertical Resize Coefficients Register
eMMA_lt	0x1002 6464	PrP_CH1_VRESIZE_COEFF2	PrP CH1 Vertical Resize Coefficients Register
eMMA_lt	0x1002 6468	PrP_CH1_VRESIZE_VALID	PrP CH1 Vertical Resize Valid Register
eMMA_lt	0x1002 646C	PrP_CH2_HRESIZE_COEFF1	PrP CH2 Horizontal Resize Coefficients Register
eMMA_lt	0x1002 6470	PrP_CH2_HRESIZE_COEFF2	PrP CH2 Horizontal Resize Coefficients Register
eMMA_lt	0x1002 6474	PrP_CH2_HRESIZE_VALID	PrP CH2 Horizontal Resize Valid Register
eMMA_lt	0x1002 6478	PrP_CH2_VRESIZE_COEFF1	PrP CH2 Vertical Resize Coefficients Register
eMMA_lt	0x1002 647C	PrP_CH2_VRESIZE_COEFF2	PrP CH2 Vertical Resize Coefficients Register
eMMA_lt	0x1002 6480	PrP_CH2_VRESIZE_VALID	PrP CH2 Vertical Resize Valid Register
PLLCLK	0x1002 7000	CSCR	Clock Source Control Register
PLLCLK	0x1002 7004	MPCTL0	MPLL Control Register 0
PLLCLK	0x10027008	MPCTL1	MPLL Control Register 1
PLLCLK	0x1002700C	SPCTL0	SPLL Control Register 0
PLLCLK	0x10027010	SPCTL1	SPLL Control Register 1
PLLCLK	0x10027014	OSC26MCTL	Oscillator 26M Register
PLLCLK	0x10027018	PCDR0	Peripheral Clock Divider Register 0
PLLCLK	0x1002701C	PCDR1	Peripheral Clock Divider Register 1
PLLCLK	0x10027020	PCCR0	Peripheral Clock Control Register 0
PLLCLK	0x10027024	PCCR1	Peripheral Clock Control Register 1
PLLCLK	0x10027028	CCSR	Clock Control Status Register
PLLCLK	0x1002702C	PMCTL	PMOS Switch Control Register
PLLCLK	0x10027030	PMCOUNT	PMOS Switch Counter Register
PLLCLK	0x10027034	WKGDCTL	Wakeup Guard Mode Control Register
SYSCTRL	0x10027800	CID	Chip ID Register
SYSCTRL	0x10027814	FMCR	Function Multiplexing Control Register
SYSCTRL	0x10027818	GPCR	Global Peripheral Control Register
SYSCTRL	0x1002781C	WBCR	Well Bias Control Register
SYSCTRL	0x10027820	DSCR1	Driving Strength Control Register 1
SYSCTRL	0x10027824	DSCR2	Driving Strength Control Register 2

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
SYSCTRL	0x10027828	DSCR3	Driving Strength Control Register 3
SYSCTRL	0x1002782C	DSCR4	Driving Strength Control Register 4
SYSCTRL	0x10027830	DSCR5	Driving Strength Control Register 5
SYSCTRL	0x10027834	DSCR6	Driving Strength Control Register 6
SYSCTRL	0x10027838	DSCR7	Driving Strength Control Register 7
SYSCTRL	0x1002783C	DSCR8	Driving Strength Control Register 8
SYSCTRL	0x10027840	DSCR9	Driving Strength Control Register 9
SYSCTRL	0x1002 7844	DSCR10	Driving Strength Control Register 10
SYSCTRL	0x1002 7848	DSCR11	Driving Strength Control Register 11
SYSCTRL	0x1002 784C	DSCR12	Driving Strength Control Register 12
SYSCTRL	0x1002 7850	DSCR13	Driving Strength Control Register 13
SYSCTRL	0x1002 7854	PSCR	Pull Strength Control Register
SYSCTRL	0x1002 7858	PCSR	Priority Control and Select Register
SYSCTRL	0x1002 7860	PMCR	Power Management Control Register
SYSCTRL	0x1002 7864	DCVR0	DPTC Comparator Value Register 0
SYSCTRL	0x1002 7868	DCVR‘	DPTC Comparator Value Register 1
SYSCTRL	0x1002 786C	DCVR2	DPTC Comparator Value Register 2
SYSCTRL	0x1002 7870	DCVR3	DPTC Comparator Value Register 3
IIM	0x1002_8000	STAT	Status Register
IIM	0x1002_8004	STATM	Status IRQ Mask Register
IIM	0x1002_8008	ERR	Module Errors Register
IIM	0x1002_800C	EMASK	Error IRQ Mask Register
IIM	0x1002_8010	FCTL	Fuse Control Register
IIM	0x1002_8014	UA	Upper Address Register
IIM	0x1002_8018	LA	Lower Address Register
IIM	0x1002_801C	SDAT	Explicit Sense Data Register
IIM	0x1002_8020	PREV	Product Revision Register
IIM	0x1002_8024	SREV	Silicon Revision Register
IIM	0x1002_8028	PROG_P	Program Protection Register
IIM	0x1002_802C	SCS0	Software_Controllable Signals Register 0
IIM	0x1002_8030	SCS1	Software_Controllable Volatile Hardware—Visible Signals Register (1–3)
IIM	0x1002_8034	SCS2	
IIM	0x1002_8038	SCS3	

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
IIM	0x1002_803C	FBAC0	Fuse Bank 0 Access Protection Register
IIM	0x1002_8804	WORD1_BANK0	Word 1 of Fusebank 0
IIM	0x1002_8808	WORD2_BANK0	Word 2 of Fusebank 0
IIM	0x1002_880C	WORD3_BANK0	Word 3 of Fusebank 0
IIM	0x1002_8810	WORD4_BANK0	Word 4 of Fusebank 0
IIM	0x1002_8814–0x1002_8828	SUID	Silicon_Unique_ID[47:0]
IIM	0x1002_882C–0x1002_887C	SCC_KEY	SCC_KEY[167:0]
IIM	0x1002_8C00	FBAC1	Fuse Bank 0 Access Protection register
IIM	0x1002_8C04–0x1002_8C18	MAC_ADDR	MAC Address of Ethernet
IIM	0x1002_8C1C–0x1002_8C7C	RESERVED	Reserved for future use
RTIC	0x1002 A000	RTICSR	RTIC Status Register
RTIC	0x1002 A004	RTICCMDR	RTIC Command Register
RTIC	0x1002 A008	RTICCNTLR	RTIC Control Register
RTIC	0x1002 A00C	RTICTR	RTIC Throttle Register
RTIC	0x1002 A010	RTICAMSAR1	RTIC Memory A Start Address Register 1
RTIC	0x1002 A014	RTICAMLR1	RTIC Memory A Len Register 1
RTIC	0x1002 A0018	RTICAMSAR2	RTIC Memory A Start Address Register 2
RTIC	0x1002 A01C	RTICAMLR2	RTIC Memory A Len Register 2
RTIC	0x1002 A030	RTICBMSAR1	RTIC Memory B Start Address Register 1
RTIC	0x1002 A034	RTICBMLR1	RTIC Memory B Len Register 1
RTIC	0x1002 A038	RTICBMSAR2	RTIC Memory B Start Address Register 2
RTIC	0x1002 A003C	RTICBMLR2	RTIC Memory B Len Register 2
RTIC	0x1002 A050	RTICCMSAR1	RTIC Memory C Start Address Register 1
RTIC	0x1002 A054	RTICCMLR1	RTIC Memory C Len Register 1
RTIC	0x1002 A058	RTICCMSAR2	RTIC Memory C Start Address Register 2
RTIC	0x1002 A05C	RTICCMLR2	RTIC Memory C Len Register 2
RTIC	0x1002 A070	RTICDMSAR1	RTIC Memory D Start Address Register 1
RTIC	0x1002 A074	RTICDMLR1	RTIC Memory D Len Register 1
RTIC	0x1002 A078	RTICDMSAR2	RTIC Memory D Start Address Register 2
RTIC	0x1002 A07C	RTICDMLR2	RTIC Memory D Len Register 2

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
RTIC	0x1002 A090	RTICFAR	RTIC Fault Address Register
RTIC	0x1002 A094	RTICWR	RTIC Watchdog Register
RTIC	0x1002 A0A0	RTICAMHR1	RTIC Memory A Hash Result [159:128]
RTIC	0x1002 A0A4	RTICAMHR2	RTIC Memory A Hash Result [127:96]
RTIC	0x1002 A0A8	RTICAMHR3	RTIC Memory A Hash Result [95:64]
RTIC	0x1002 A0AC	RTICAMHR4	RTIC Memory A Hash Result [63:32]
RTIC	0x1002 A0B0	RTICAMHR5	RTIC Memory A Hash Result [31:0]
RTIC	0x1002 A0C0	RTICBMHR1	RTIC Memory B Hash Result [159:128]
RTIC	0x1002 A0C4	RTICBMHR2	RTIC Memory B Hash Result [127:96]
RTIC	0x1002 A0C8	RTICBMHR3	RTIC Memory B Hash Result [95:64]
RTIC	0x1002 A0CC	RTICBMHR4	RTIC Memory B Hash Result [63:32
RTIC	0x1002 A0D0	RTICBMHR5	RTIC Memory B Hash Result [31:0]
RTIC	0x1002 A0E0	RTICCMHR1	RTIC Memory C Hash Result [159:128]
RTIC	0x1002 A0E4	RTICCMHR2	RTIC Memory C Hash Result [127:96]
RTIC	0x1002 A0E8	RTICCMHR3	RTIC Memory C Hash Result [95:64]
RTIC	0x1002 A0EC	RTICCMHR4	RTIC Memory C Hash Result [63:32]
RTIC	0x1002 A0F0	RTICCMHR5	RTIC Memory C Hash Result [31:0]
RTIC	0x1002 A100	RTICDMHR1	RTIC Memory D Hash Result [159:128]
RTIC	0x1002 A104	RTICDMHR2	RTIC Memory D Hash Result [127:96]
RTIC	0x1002 A108	RTICDMHR3	RTIC Memory D Hash Result [95:64]
RTIC	0x1002 A10C	RTICDMHR4	RTIC Memory D Hash Result [63:32]
RTIC	0x1002 A110	RTICDMHR5	RTIC Memory D Hash Result [31:0]
FEC	0x1002_B004	EIR	Interrupt Event Register
FEC	0x1002_B008	EIMR	Interrupt Mask Register
FEC	0x1002_B010	RDAR	Receive Descriptor Active Register
FEC	0x1002_B014	TDAR	Transmit Descriptor Active Register
FEC	0x1002_B024	ECR	Ethernet Control Register
FEC	0x1002_B040	MMFR	MII Management Frame Register
FEC	0x1002_B044	MSCR	MII Speed Control Register
FEC	0x1002_B064	MIBC	MIB Control/Status Register
FEC	0x1002_B084	RCR	Receive Control Register
FEC	0x1002_B0C4	TCR	Transmit Control Register
FEC	0x1002_B0E4	PALR	Physical Address Low Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
FEC	0x1002_B0E8	PAUR	Physical Address High+ Type Field
FEC	0x1002_B0EC	OPD	Opcode + Pause Duration
FEC	0x1002_B118	IAUR	Upper 32 bits of Individual Hash Table
FEC	0x1002_B11C	IALR	Lower 32 Bits of Individual Hash Table
FEC	0x1002_B120	GAUR	Upper 32 bits of Group Hash Table
FEC	0x1002_B124	GALR	Lower 32 bits of Group Hash Table
FEC	0x1002_B144	TFWR	Transmit FIFO Watermark
FEC	0x1002_B14C	FRBR	FIFO Receive Bound Register
FEC	0x1002_B150	FRSR	FIFO Receive FIFO Start Registers
FEC	0x1002_B180	ERDSR	Pointer to Receive Descriptor Ring
FEC	0x1002_B184	ETDSR	Pointer to Transmit Descriptor Ring
FEC	0x1002_B188	EMRBR	Maximum Receive Buffer Size
FEC	0x1002_B200	RMON_T_DROP	Count of frames not counted correctly
FEC	0x1002_B204	RMON_T_PACKETS	RMON Tx packet count
FEC	0x1002_B208	RMON_T_BC_PKT	RMON Tx Broadcast Packets
FEC	0x1002_B20C	RMON_T_MC_PKT	RMON Tx Multicast Packets
FEC	0x1002_B210	RMON_T_CRC_ALIGN	RMON Tx Packets w CRC/Align error
FEC	0x1002_B214	RMON_T_UNDERSIZE	RMON Tx Packets < 64 bytes, good crc
FEC	0x1002_B218	RMON_T_OVERSIZE	RMON Tx Packets > MAX_FL bytes, good crc
FEC	0x1002_B21C	RMON_T_FRAG	RMON Tx Packets < 64 bytes, bad crc
FEC	0x1002_B220	RMON_T_JAB	RMON Tx Packets > MAX_FL bytes, bad crc
FEC	0x1002_B224	RMON_T_COL	RMON Tx collision count
FEC	0x1002_B228	RMON_T_P64	RMON Tx 64 byte packets
FEC	0x1002_ 22C	RMON_T_P65TO127	RMON Tx 65 to 127 byte packets
FEC	0x1002_B230	RMON_T_P128TO255	RMON Tx 128 to 255 byte packets
FEC	0x1002_B234	RMON_T_P256TO511	RMON Tx 256 to 511 byte packets
FEC	0x1002_B238	RMON_T_P512TO1023	RMON Tx 512 to 1023 byte packets
FEC	0x1002_B23C	RMON_T_P1024TO2047	RMON Tx 1024 to 2047 byte packets
FEC	0x1002_B240	RMON_T_P_GTE2048	RMON Tx packets w > 2048 bytes
FEC	0x1002_B244	RMON_T_OCTETS	RMON Tx Octets
FEC	0x1002_B248	IEEE_T_DROP	Count of frames not counted correctly
FEC	0x1002_B24C	IEEE_T_FRAME_OK	Frames Transmitted OK
FEC	0x1002_B250	IEEE_T_1COL	Frames Transmitted with Single Collision

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
FEC	0x1002_B254	IEEE_T_MCOL	Frames Transmitted with Multiple Collisions
FEC	0x1002_B258	IEEE_T_DEF	Frames Transmitted after Deferral Delay
FEC	0x1002_B25C	IEEE_T_LCOL	Frames Transmitted with Late Collision
FEC	0x1002_B260	IEEE_T_EXCOL	Frames Transmitted with Excessive Collisions
FEC	0x1002_B264	IEEE_T_MACERR	Frames Transmitted with Tx FIFO Underrun
FEC	0x1002_B268	IEEE_T_CSERR	Frames Transmitted with Carrier Sense Error
FEC	0x1002_B26C	IEEE_T_SQE	Frames Transmitted with SQE Error
FEC	0x1002_B270	IEEE_T_FDXFC	Flow Control Pause frames transmitted
FEC	0x1002_B274	IEEE_T_OCTETS_OK	Octet count for Frames Transmitted w/o Error
FEC	0x1002_B284	RMON_R_PACKETS	RMON Rx packet count
FEC	0x1002_B288	RMON_R_BC_PKT	RMON Rx Broadcast Packets
FEC	0x1002_B28C	RMON_R_MC_PKT	RMON Rx Multicast Packets
FEC	0x1002_B290	RMON_R_CRC_ALIGN	RMON Rx Packets w CRC/Align error
FEC	0x1002_B294	RMON_R_UNDERSIZE	RMON Rx Packets < 64 bytes, good crc
FEC	0x1002_B298	RMON_R_OVERSIZE	RMON Rx Packets > MAX_FL bytes, good crc
FEC	0x1002_B29C	RMON_R_FRAG	RMON Rx Packets < 64 bytes, bad crc
FEC	0x1002_B2A0	RMON_R_JAB	RMON Rx Packets > MAX_FL bytes, bad crc
FEC	0x1002_B2A4	RMON_R_RESVD_0	Reserved
FEC	0x1002_B2A8	RMON_R_P64	RMON Rx 64 byte packets
FEC	0x1002_B2AC	RMON_R_P65TO127	RMON Rx 65 to 127 byte packets
FEC	0x1002_B2B0	RMON_R_P128TO255	RMON Rx 128 to 255 byte packets
FEC	0x1002_B2B4	RMON_R_P256TO511	RMON Rx 256 to 511 byte packets
FEC	0x1002_B2B8	RMON_R_P512TO1023	RMON Rx 512 to 1023 byte packets
FEC	0x1002_B2BC	RMON_R_P1024TO2047	RMON Rx 1024 to 2047 byte packets
FEC	0x1002_B2C0	RMON_R_P_GTE2048	RMON Rx packets w > 2048 bytes
FEC	0x1002_B2C4	RMON_R_OCTETS	RMON Rx Octets
FEC	0x1002_B2C8	IEEE_R_DROP	Count of frames not counted correctly
FEC	0x1002_B2CC	IEEE_R_FRAME_OK	Frames Received OK
FEC	0x1002_B2D0	IEEE_R_CRC	Frames Received with CRC Error
FEC	0x1002_B2D4	IEEE_R_ALIGN	Frames Received with Alignment Error
FEC	0x1002_B2D8	IEEE_R_MACERR	Receive Fifo Overflow count
FEC	0x1002_B2DC	IEEE_R_FDXFC	Flow Control Pause frames received
FEC	0x1002_B2E0	IEEE_R_OCTETS_OK	Octet count for Frames Rcvd w/o Error

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
SCC	0x1002_C000	RED_START	SCM Red Memory Start Addr Register
SCC	0x1002_C004	BLACK_START	SCM Black Memory Start Addr Register
SCC	0x1002_C008	LENGTH	SCM Encrypt/Decrypt Data Length Register
SCC	0x1002_C010	SCM_STAT	SCM Status Register
SCC	0x1002_C014	SCM_ERROR	SCC Error Status Register
SCC	0x1002_C018	INTERRUPT_CONTROL	SCC Interrupt Control Register
SCC	0x1002_C01C	CONFIGURATION	SCC Configuration Register
SCC	0x1002_C020	INIT_VECTOR 0	Initial Vector 0 Register
SCC	0x1002_C024	INIT_VECTOR1	Initial Vector 1 Register
SCC	0x1002_C400 – 0x1002_C7FF	SCM_RED_MEM	SCM Red Memory
SCC	0x1002_C800 – 0x1002_CBFF	SCM_BLACK_MEM	SCM Black Memory
SCC	0x1002_D000	SMN_STAT	SMN Status Register
SCC	0x1002_D004	SMN_COMMAND	SMN Command Register
SCC	0x1002_D008	SEQ_START	Sequence Start Value Register
SCC	0x1002_D00C	SEQ_END	Sequence End Value Register
SCC	0x1002_D010	SEQ_CHECK	Sequence Check Register
SCC	0x1002_D014	BIT_COUNT	Bit Count Register
SCC	0x1002_D018	BIT_BANK_INC_SIZE	Bit Bank Increment Size Register
SCC	0x1002_D01C	BIT_BANK_DEC	Bit Bank Decrement
SCC	0x1002_D020	CMP_SIZE	Compare Size Register
SCC	0x1002_D024	PLAINTEXT_CHECK	Plaintext Check Register
SCC	0x1002_D028	CIPHER CHECK	Ciphertext Check Register
SCC	0x1002_D02C	TIMER IV	Timer Initial Vector Register
SCC	0x1002_D030	TIMER CONTROL	Timer Control Register
SCC	0x1002_D034	DEBUG DETECTOR STATUS	Debug Port Detection Status Register
SCC	0x1002_D038	TIMER	Timer Register
ETB REG	0x1003_B000	ETB_ID	ETB Identify Register
ETB REG	0x1003_B004	ETB_RAM_DEPTH	ETB RAM depth Register
ETB REG	0x1003_B008	ETB_RAM_WIDTH	ETB RAM width Register
ETB REG	0x1003_B00C	ETB_STATUS	ETB Status Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
ETB REG	0x1003_B010	ETB_DATA	ETB Data Register
ETB REG	0x1003_B014	ETB_READ_POINTER	ETB Read Pointer Register
ETB REG	0x1003_B018	ETB_WRITE_POINTER	ETB Write Pointer Register
ETB REG	0x1003_B01C	ETB_TRIGGER_COUNTER	ETB Trigger Counter Register
ETB REG	0x1003_B020	ETB_CONTROL	ETB Control Register
JAM	0x1003_E000	JAM_ARM9P_GPR0	JAM ARM9P General Purpose Register 0
JAM	0x1003_E010	JAM_ARM9P_GPR4	JAM ARM9P General Purpose Register 4
MAX	0x1003_F000	MPR0	Master Priority Register for Slave Port 0
MAX	0x1003_F100	MPR1	Master Priority Register for Slave Port 1
MAX	0x1003_F200	MPR2	Master Priority Register for Slave Port 2
MAX	0x1003_F004	AMPR0	Alternate Master Priority Register for Slave Port 0
MAX	0x1003_F104	AMPR1	Alternate Master Priority Register for Slave Port 1
MAX	0x1003_F204	AMPR2	Alternate Master Priority Register for Slave Port 2
MAX	0x1003_F010	SGPCR0	General Purpose Control Register for Slave Port 0
MAX	0x1003_F110	SGPCR1	General Purpose Control Register for Slave Port 1
MAX	0x1003_F210	SGPCR2	General Purpose Control Register for Slave Port 2
MAX	0x1003_F014	ASGPCR0	Alternate SGPCR for Slave Port 0
MAX	0x1003_F114	ASGPCR1	Alternate SGPCR for Slave Port 1
MAX	0x1003_F214	ASGPCR2	Alternate SGPCR for Slave Port 2
MAX	0x1003_F800	MGPCR0	General Purpose Control Register for Master Port 0
MAX	0x1003_F900	MGPCR1	General Purpose Control Register for Master Port 1
MAX	0x1003_FA00	MGPCR2	General Purpose Control Register for Master Port 2
MAX	0x1003_FB00	MGPCR3	General Purpose Control Register for Master Port 3
MAX	0x1003_FC00	MGPCR4	General Purpose Control Register for Master Port 4
MAX	0x1003_FD00	MGPCR5	General Purpose Control Register for Master Port 5
AITC	0x1004_0000	INTCNTL	Interrupt Control Register
AITC	0x1004_0004	NIMASK	Normal Interrupt Mask Register
AITC	0x1004_0008	INTENNUM	Interrupt Enable Number Register
AITC	0x1004_000C	INTDISNUM	Interrupt Disable Number Register
AITC	0x1004_0010	INTENABLEH	Interrupt Enable Register High
AITC	0x1004_0014	INTENABLEL	Interrupt Enable Register Low
AITC	0x1004_0018	INTTYPEH	Interrupt Type Register High
AITC	0x1004_001C	INTTYPEL	Interrupt Type Register Low

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
AITC	0x1004_0020	NIPRIORITY7	Normal Interrupt Priority Level Register 7
AITC	0x1004_0024	NIPRIORITY6	Normal Interrupt Priority Level Register 6
AITC	0x1004_0028	NIPRIORITY5	Normal Interrupt Priority Level Register 5
AITC	0x1004_002C	NIPRIORITY4	Normal Interrupt Priority Level Register 4
AITC	0x1004_0030	NIPRIORITY3	Normal Interrupt Priority Level Register 3
AITC	0x1004_0034	NIPRIORITY2	Normal Interrupt Priority Level Register 2
AITC	0x1004_0038	NIPRIORITY1	Normal Interrupt Priority Level Register 1
AITC	0x1004_003C	NIPRIORITY0	Normal Interrupt Priority Level Register 0
AITC	0x1004_0040	NIVECSR	Normal Interrupt Vector and Status Register
AITC	0x1004_0044	FIVECSR	Fast Interrupt Vector and Status Register
AITC	0x1004_0048	INTSRCH	Interrupt Source Register High
AITC	0x1004_004C	INTSRCL	Interrupt Source Register Low
AITC	0x1004_0050	INTFRCH	Interrupt Force Register High
AITC	0x1004_0054	INTFRCL	Interrupt Force Register Low
AITC	0x1004_0058	NIPNDH	Normal Interrupt Pending Register High
AITC	0x1004_005C	NIPNDL	Normal Interrupt Pending Register Low
AITC	0x1004_0060	FIPNDH	Fast Interrupt Pending Register High
AITC	0x1004_0064	FIPNDL	Fast Interrupt Pending Register Low
CSI	0x8000_0000	CSICR1	CSI Control Register 1
CSI	0x8000_0004	CSICR2	CSI Control Register 2
CSI	0x8000_0008	CSISR	CSI Status Register
CSI	0x8000_000C	CSISTATR	CSI Statistic FIFO Register
CSI	0x8000_0010	CSIRXR	CSI RxFIFO Register
CSI	0x8000_0014	CSIRXCNT	CSI RX Count Register
CSI	0x8000_0018	CSIDEBUG	CSI Debug Register
CSI	0x8000_001C	CSICR3	CSI Control Register 3
ATA	0x8000_1000	TIME_CONFIG0	ATA timing parameter 0.
ATA	0x8000_1004	TIME_CONFIG1	ATA timing parameter 1.
ATA	0x8000_1008	TIME_CONFIG2	ATA timing parameter 2.
ATA	0x8000_100C	TIME_CONFIG3	ATA timing parameter 3.
ATA	0x8000_1010	TIME_CONFIG4	ATA timing parameter 4.
ATA	0x8000_1014	TIME_CONFIG5	ATA timing parameter 5.
ATA	0x8000_1018	FIFO_DATA_32	32-bit wide data port to/from FIFO

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
ATA	0x8000_101C	FIFO_DATA_16	16-bit wide data port to/from FIFO
ATA	0x8000_1020	FIFO_FILL	FIFO Filling in Halfwords
ATA	0x8000_1024	ATA_CONTROL	ATA Interface Control Register
ATA	0x8000_1028	INT_PENDING	Interrupt Pending Register
ATA	0x8000_102c	INT_ENABLE	Interrupt Enable Register
ATA	0x8000_1030	INT_CLEAR	Interrupt Clear Register
ATA	0x8000_1034	FIFO_ALARM	FIFO Alarm Threshold
ATA	0x8000_10A0	DCTR	Drive Data Register
ATA	0x8000_10A4	DDTR	Drive Features Register
ATA	0x8000_10A8	DFTR	Drive Sector Count Register
ATA	0x8000_10AC	DSCR	Drive Sector Number Register
ATA	0x8000_10B0	DSNR	Drive Cylinder Low Register
ATA	0x8000_10B4	DCLR	Drive Cylinder High Register
ATA	0x8000_10B8	DCHR	Drive Device Head Register
ATA	0x8000_10BC	DDHR	Drive Command Register (W)/ Drive Status Register (R)
ATA	0x8000_10D8	DCDR	Drive Alternate Status Register (W)/ Drive Control Register (R)
NFC	0xD800_0E00	NFC_BUFSIZE	Internal SRAM Size
NFC	0xD800_0E02	Reserved	Reserved
NFC	0xD800_0E04	RAM_BUFFER_ADDRESS	Buffer Number for Page Data Transfer To/ From Flash Memory
NFC	0xD800_0E06	NAND_FLASH_ADD	NAND Flash Address
NFC	0xD800_0E08	NAND_FLASH_CMD	NAND Flash Command
NFC	0xD800_0E0A	NFC_CONFIGURATION	NFC Internal Buffer Lock Control
NFC	0xD800_0E0C	ECC_STATUS_RESULT	Controller Status/Result of Flash Operation
NFC	0xD800_0E0E	ECC_RSLT_MAIN_AREA	ECC Error Position of Main Area Data Error
NFC	0xD800_0E10	ECC_RSLT_SPARE_AREA	ECC Error Position of Spare Area Data Error
NFC	0xD800_0E12	NF_WR_PROT	Nand Flash Write Protection
NFC	0xD800_0E14	UNLOCK_START_BLK_ADD	Start Address for Write Protection Unlock
NFC	0xD800_0E16	UNLOCK_END_BLK_ADD	End Address for Write Protection Unlock
NFC	0xD800_0E18	NAND_FLASH_WR_PR_ST	Current Nand Flash Write Protection Status
NFC	0xD800_0E1A	NAND_FLASH_CONFIG1	Nand Flash Operation Configuration 1
NFC	0xD800_0E1C	NAND_FLASH_CONFIG2	Nand Flash Operation Configuration 2

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
ESDCTL	0xD800_1000	ESD_ESDCTL0	SDRAM/MDDR 0 Control Register
ESDCTL	0xD800_1004	ESD_ESDCFG0	SDRAM/MDDR 0 Timing Config Register
ESDCTL	0xD800_1008	ESD_ESDCTL1	SDRAM/MDDR 1 Control Register
ESDCTL	0xD800_100C	ESD_ESDCFG1	SDRAM/MDDR 1 Timing Config Register
ESDCTL	0xD800_1010	ESD_ESDMISC	SDRAM/MDDR Miscellaneous Register
WEIM	0xD800_2000	CS0U	Chip Select 0 Upper Control Register
WEIM	0xD800_2004	CS0L	Chip Select 0 Lower Control Register
WEIM	0xD800_2008	CS0A	Chip Select 0 Addition Control Register
WEIM	0xD800 2010	CS1U	Chip Select 1 Upper Control Register
WEIM	0xD800_2014	CS1L	Chip Select 1 Lower Control Register
WEIM	0xD800_2018	CS1A	Chip Select 1 Addition Control Register
WEIM	0xD800_2020	CS2U	Chip Select 2 Upper Control Register
WEIM	0xD800_2024	CS2L	Chip Select 2 Lower Control Register
WEIM	0xD800_2028	CS2A	Chip Select 2 Addition Control Register
WEIM	0xD800_2030	CS3U	Chip Select 3 Upper Control Register
WEIM	0xD800_2034	CS3L	Chip Select 3 Lower Control Register
WEIM	0xD800_2038	CS3A	Chip Select 3 Addition Control Register
WEIM	0xD800_2040	CS4U	Chip Select 4 Upper Control Register
WEIM	0xD800_2044	CS4L	Chip Select 4 Lower Control Register
WEIM	0xD800_2048	CS4A	Chip Select 4 Addition Control Register
WEIM	0xD800_2050	CS5U	Chip Select 5 Upper Control Register
WEIM	0xD800_2054	CS5L	Chip Select 5 Lower Control Register
WEIM	0xD800_2058	CS5A	Chip Select 5 Addition Control Register
WEIM	0xD800_2060	EIM	EIM Configuration Register
M3IF	0xD800_3000	M3IF_CTL	M3IF control register
M3IF	0xD800_3028	M3IF_SCFG0	M3IF Snooping Configuration Register 0
M3IF	0xD800_302C	M3IF_SCFG1	M3IF Snooping Configuration Register 1
M3IF	0xD800_3030	M3IF_SCFG2	M3IF Snooping Configuration Register 2
M3IF	0xD800_3034	M3IF_SSR0	M3IF Snooping Status Register 0
M3IF	0xD800_3038	M3IF_SSR1	M3IF Snooping Status Register 1
M3IF	0xD800_3040	M3IFMLWE0	M3IF Master Lock WEIM CS0 Register
M3IF	0xD800_3044	M3IFMLWE1	M3IF Master Lock WEIM CS1 Register
M3IF	0xD800_3048	M3IFMLWE2	M3IF Master Lock WEIM CS2 Register

Table 2-8. Register Map (continued)

Module Name	Address	Register Name	Description
M3IF	0xD800_304C	M3IFMLWE3	M3IF Master Lock WEIM CS3 Register
M3IF	0xD800_3050	M3IFMLWE4	M3IF Master Lock WEIM CS4 Register
M3IF	0xD800_3054	M3IFMLWE5	M3IF Master Lock WEIM CS5 Register
PCMCIA	0xD800_4000	PCMCIA_PIPR	PCMCIA Input Pins Register
PCMCIA	0xD800_4004	PCMCIA_PSCR	PCMCIA Status Changed Register
PCMCIA	0xD800_4008	PCMCIA_PER	PCMCIA Enable Register
PCMCIA	0xD800_400C	PCMCIA_PBR0	PCMCIA Base Register 0
PCMCIA	0xD800_4010	PCMCIA_PBR1	PCMCIA Base Register 1
PCMCIA	0xD800_4014	PCMCIA_PBR2	PCMCIA Base Register 2
PCMCIA	0xD800_4018	PCMCIA_PBR3	PCMCIA Base Register 3
PCMCIA	0xD800_401C	PCMCIA_PBR4	PCMCIA Base Register 4
PCMCIA	0xD800_4020	PCMCIA_PBR5	PCMCIA Base Register 5
PCMCIA	0xD800_4024	PCMCIA_PBR6	PCMCIA Base Register 6
PCMCIA	0xD800_4028	PCMCIA_POR0	PCMCIA Option Register 0
PCMCIA	0xD800_402C	PCMCIA_POR1	PCMCIA Option Register 1
PCMCIA	0xD800_4030	PCMCIA_POR2	PCMCIA Option Register 2
PCMCIA	0xD800_4034	PCMCIA_POR3	PCMCIA Option Register 3
PCMCIA	0xD800_4038	PCMCIA_POR4	PCMCIA Option Register 4
PCMCIA	0xD800_403C	PCMCIA_POR5	PCMCIA Option Register 5
PCMCIA	0xD800_4040	PCMCIA_POR6	PCMCIA Option Register 6
PCMCIA	0xD800_4044	PCMCIA_POFR0	PCMCIA Offset Register 0
PCMCIA	0xD800_4048	PCMCIA_POFR1	PCMCIA Offset Register 1
PCMCIA	0xD800_404C	PCMCIA_POFR2	PCMCIA Offset Register 2
PCMCIA	0xD800_4050	PCMCIA_POFR3	PCMCIA Offset Register 3
PCMCIA	0xD800_4054	PCMCIA_POFR4	PCMCIA Offset Register 4
PCMCIA	0xD800_4058	PCMCIA_POFR5	PCMCIA Offset Register 5
PCMCIA	0xD800_405C	PCMCIA_POFR6	PCMCIA Offset Register 6
PCMCIA	0xD800_4060	PCMCIA_PGCR	PCMCIA General Control Register
PCMCIA	0xD800_4064	PCMCIA_PGSR	PCMCIA General Status Register

Chapter 3
Clocks, Power Management, and Reset Control

3.1 Introduction

There are two clock controller modules in the i.MX27 Multimedia Applications Processor: The ARM9 Platform Clock Controller and the PLL Clock Controller module (CCM), which produces the clock signals used and distributed by the ARM9 Platform Clock Controller.

- ARM9 Platform Clock Controller—The primary function of the ARM9 Platform Clock Controller is to take the clock signals from the PLL Clock Controller and distribute them to various peripherals on the ARM9 Platform. The clock control module contains the logic to turn clocks on or off and to determine when the ARM9 Platform's clock can be turned off. This module also synchronizes the JTAG interface to the CLK domain.
- PLL Clock Controller—This module generates clock signals used throughout the i.MX27 chip and external peripherals. The PLL Clock Controller also serves as the interface between the ARM9 Platform and the peripherals on the i.MX27 device.

The ARM9 Platform Clock Controller is not a user-programmable or accessible module, whereas the PLL Clock Controller is accessible—therefore, only the PLL Clock Controller is described in this chapter.

3.2 Clock Controller Architecture Block Diagram

There are two DPLLs in the PLL Clock Controller—the MCU/System PLL (MPLL) and the Serial Peripheral PLL (SPLL), which uses digital and mixed analog/digital circuits to provide clock frequencies for wireless communication and other applications. The MPLL primarily generates the CLK signal to the ARM9 and HCLK (also called System clock) for the system bus and for most of the on-chip peripherals, including the LCDC pixel clock and the NAND Flash Controller clock. The SPLL produces the primary clock to the clock dividers for USB OTG, SSI1, and SSI2.

Both MPLL and SPLL accept either the output of the FPM or the OSC26M as a source from which to generate the required frequencies for the ARM9 Platform and/or peripherals using a fractional frequency multiplication method. Detailed information about the calculation of the DPLL settings is shown in Section 3.2.2, "Output Frequency Calculations."

To produce the wide range of on-chip clock frequencies required by the i.MX27 processor, the core clock generator uses a two-stage phase locked loop. The first stage is a Frequency Pre-Multiplier PLL (FPM), which multiplies the input frequency by a factor of 1024. If the input crystal frequency is 32.768 kHz, the premultiplier multiplies it by a factor of 1024 to 33.554 MHz (32.768 MHz for a 32.0 KHz crystal). The output of the FPM is one of the clock sources for the MPLL and SPLL. Power management in the i.MX27 device is accomplished by controlling the clock output of the MPLL and SPLL units.

The distribution of clocks in the i.MX27 processor is shown in the general block diagram, Figure 3-1. There are two external clock sources to the PLL Clock Controller, as follows:

- 32 kHz external crystal
- 26 MHz external source/crystal

Settings in the Clock Source Control Register (CSCR) are used to independently configure the external clock sources applied to the MPLL and SPLL.

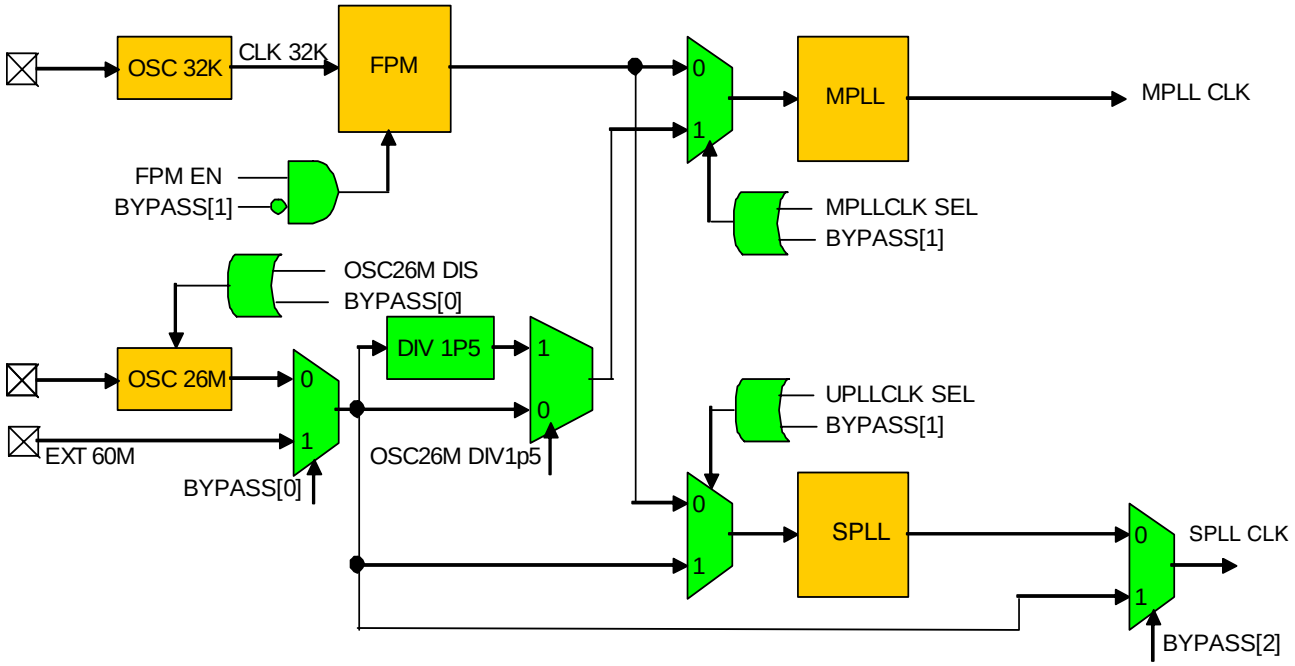


Figure 3-1. i.MX27 Clock Distribution Block Diagram (1 of 2)

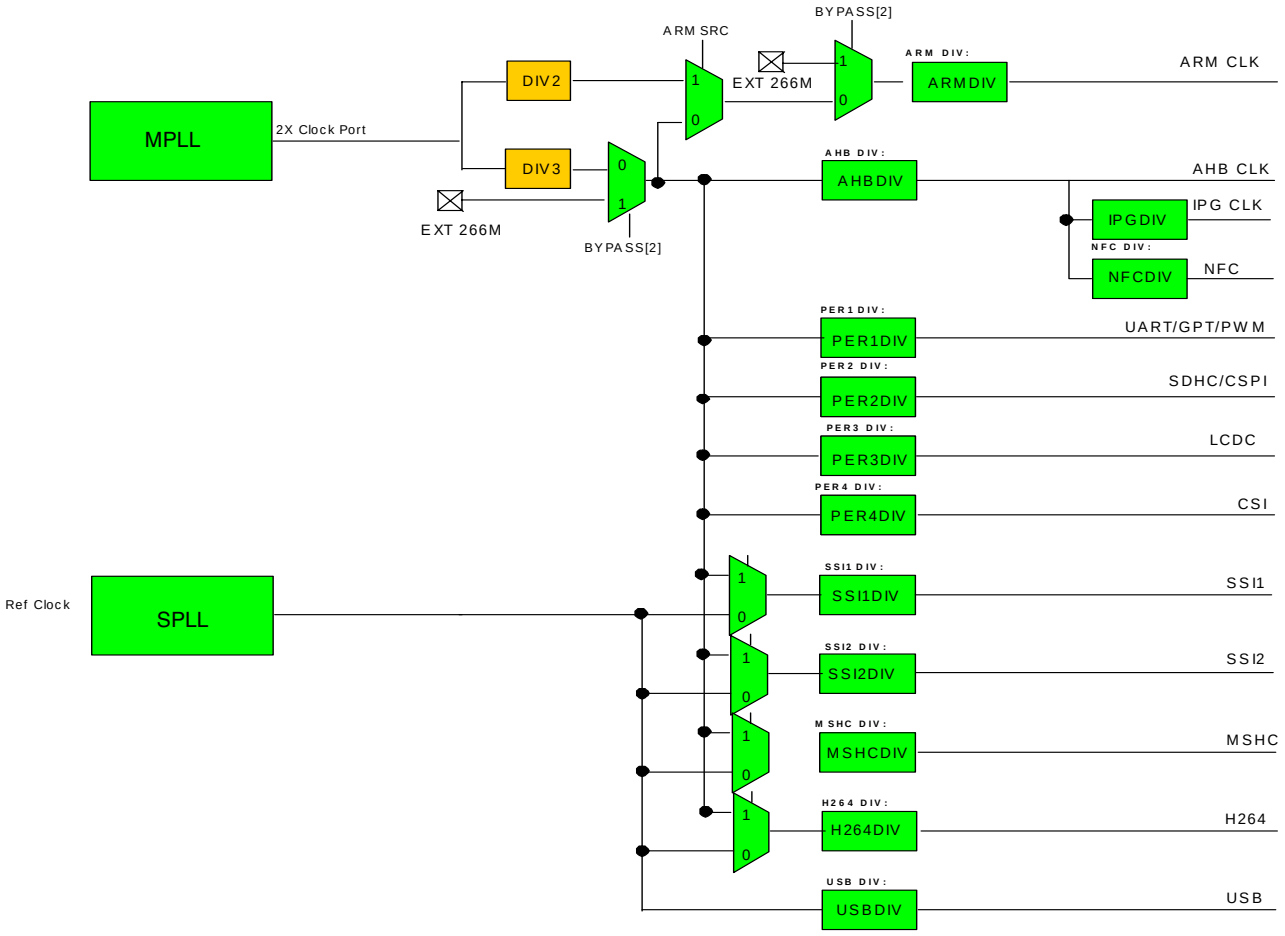


Figure 3-2. i.MX27 Clock Distribution Block Diagram (2 of 2)

Table 3-1. PLL Clock Controller Signal Descriptions

Signal Names	Description
CLK	Fast clock used only by ARM9 Platform for internal operations, such as executing instructions from the cache. Can be gated during Doze and Sleep mode when all the criteria are met to enter a low power.
HCLK	System clock. Appears as the BCLK input to the CPU and the HCLK to the system. This is a continuous clock (when the system is not in Sleep mode). It can be gated during Doze and Sleep mode when all the criteria are met to enter a low power.
HCLKEN	Used to signify the rising edge of CLK that corresponds to the rising edge of HCLK. It is used by the ARM9 Platform only.
CLK60M	60 MHz clock for the USB OTG module
SSI1CLK	Divided clock output for the SSI1 module
SSI2CLK	Divided clock output for the SSI2 module
NFCCLK	Divided clock output for the NAND Flash Controller module
H264CCLK	Divided clock output for the H264 module
MSHCCLK	Divided clock output for the MSHC module

Table 3-1. PLL Clock Controller Signal Descriptions (continued)

Signal Names	Description
PERCLK1	Divided clock output for peripheral set 1 (UART, Timer, PWM)
PERCLK2	Divided clock output for peripheral set 2 (SDHC, CSPI)
PERCLK3	Divided clock output for the LCDC
PERCLK4	Divided clock output for the CSI
CLKO	Selected internal clock output to the CLKO pin
A9P_CLK_OFF	Control signal from the ARM9 Platform Clock Controller

3.2.1 High Frequency Clock Source and Distribution

Two DPLLs—MPLL and SPLL—on the i.MX27 device are used to generate two separate clock frequencies from either the Frequency Pre-Multiplier (FPM) or an external high frequency source (CLK26M). The clock source for each DPLL is individually selected using bits in the Clock Source Control Register (CSCR).

The MCU/System PLL (MPLL) is configured by the MPCTL registers (MPCTL0, MPCTL1) to produce system clock signals that are divided down to output the FCLK (for example 266 MHz) and the HCLK (for example, 133 MHz) clock signals. MPLL serves as the clock source for the PERCLK4, PERDIV3, PERDIV2, and PERDIV1. FCLK serves as the clock source for the NFCDIV divider. These dividers produce the clock signals for the following:

- NAND Flash Controller (NFC)
- Peripheral set 1 (PERCLK1): UART, Timer, and PWM
- Peripheral set 2 (PERCLK2): SDHC and CSPI
- LCDC Pixel Clock (PERCLK3)
- CSI (PERCLK4)

Serial Peripheral PLL (SPLL) is configured by SPCTL registers (SPCTL0, SPCTL1) and produces input signals for the USBDIV, SSI1DIV, SSI2DIV, H264DIV, and MSHCDIV dividers, which generate clock signals for serial peripherals that require special clock frequencies:

- CLK60M—for the USB OTG
- SSI1CLK—Clock signal for SSI1
- SSI2CLK—Clock signal for SSI2
- H264CCLK—Clock signal for H264
- MSHCCLK—Clock signal for MSHC

The clock source for the SSI1DIV and SSI2DIV dividers can be the MPLL or SPLL. Source selection is controlled by the respective bits in the Clock Source Control register (CSCR).

3.2.2 Output Frequency Calculations

Both DPLLs produce a high frequency clock that exhibits a low frequency jitter and a low phase jitter. The DPLL output clock frequency (f_{dpll}) is determined by the Equation 3-1:

$$f_{dpll} = 2f_{ref} \bullet \frac{MFI + MFN / (MFD+1)}{PD+1} \qquad \textit{Eqn. 3-1}$$

where:

- f_{ref} is the reference frequency (1024 × 32.768 kHz, 1024 × 32.0 kHz, or 26 MHz).
- MFI is an integer part of a multiplication factor (MF).
- MFN is the numerator and MFD is the denominator of the MF.
- PD is the predivider factor.

NOTE

In bootstrap mode, the PLL registers assume a source clock of 32.768 KHz. If using bootstrap mode, use a 32.768 KHz crystal.

3.3 Power Management

The PLL Clock Controller module is designed with clock control at various stages of clock supply to achieve optimum power savings. The operation of the PLL and clock controller at different stages of power management is described in the following sections.

3.3.1 PLL Operation at Power-Up

The crystal oscillator begins oscillating within several hundred milliseconds of initial power-up. While system reset remains asserted the PLL begins the lockup sequence and locks 1 ms after the crystal oscillator becomes stable. Both DPLLs are enabled on power-up. The system reset is held asserted by the PLL Clock Controller for 300 ms + 14 cycles of the 32 kHz, as shown in Figure 3-17.

3.3.2 PLL Operation at Wake-Up

When the device is awakened from Sleep mode by a wake-up event, the DPLL locks within 350 µs. The crystal oscillator is always on after initial power-up, so crystal startup time is not a factor. The PLL output clock starts operating as soon as it achieves lock.

3.3.3 i.MX27 Processor Low-Power Modes

The i.MX27 processor provides two power saving modes—Doze mode and Sleep mode:

- In Doze mode, the ARM9 executes a wait for interrupt (WFI) instruction. System clocks are still active.
- In Sleep mode, the ARM9 executes a wait for interrupt (WFI) instruction. The output of the MPLL and SPLL are shut down and only the 32 kHz clock is running.

These modes are controlled by the clock control logic and a sequence of CPU instructions. Most of the peripheral modules can enable or disable the incoming clock signal through clock gating circuitry from the peripheral bus. Each module has a module enable bit which, when disabled, disables the operational clock to the module.

The i.MX27 PLL Clock Controller provides the Low-Power mode information to the Watchdog (WDOG) module.

3.3.3.1 Doze Mode

Doze mode is defined as when the ARM9 executes a wait for an interrupt instruction, after which the buffered clock supply to the MCU is turned off.

The sequence of operation to set the system to Doze mode is as follows:

1. Enable desired interrupts for wake-up from Doze mode.
2. Disable watchdog timer interrupt.
3. Execute wait-for-interrupt instruction.

The ARM9 executes a wait for interrupt instruction if all required conditions are met (no irq, fiq, or debug requests pending), the ARM9 Platform generates an A9P_CLK_OFF signal to the PLL Clock Controller module. The CLK signal to the MCU is immediately turned off when the A9P_CLK_OFF signal goes active. CLK_ALWAYS and system bus (HCLK) remain running. HCLK is required by the Cross Bar Switch within the ARM9 Platform for continuous operation of peripheral modules. When an unmasked interrupt event occurs, the CLK signal to the ARM9 is re-enabled.

3.3.3.2 Sleep Mode

Sleep mode is defined as when all the DPLLs clock outputs are disabled. A sequence of operations and criteria must be satisfied before the system turns off the MPLL and SPLL. The Sleep mode sequence is initiated when the MPEN bit in the CSCR register is cleared disabling the MPLL. This action also automatically turns off the SPLL.

The sequence to put the system into Sleep mode is as follows:

1. Disable AHB peripherals from bus accesses.
2. Enable desired interrupts to be used for system wake-up.
3. Disable watchdog timer interrupt.
4. Set the required value to the SD_CNT (CSCR register) for shutdown countdown.
5. Disable the MPLL by clearing the MPEN bit (CSCR register).
6. Execute wait-for-interrupt instruction.

The example of programming setup to enter Sleep mode is as follows:

Code Example 3-1. Programming Setup for Entering Sleep Mode

```
MRS    r0, CPSR                          ; Enable interrupts
AND    r1, r0, #(ENABLE_IRQ+ENABLE_FIQ+MODE_BITS)
MSR    CPSR_c, r1
LDR    r3, =WDG_BASEADDR                 ; Disable WDG Timer
LDRH   r4, [r3, #0x0]
ORR    r4, r4, #0x00000001
STRH   r4, [r3, #0x0]
LDR    r1, =CRM_BASEADDR                 ; Set SDCNT to '01'
LDR    r2, [r1, #0x0]
ORR    r2, r2, #0x0100_0000
STR    r2, [r1, #0x0]
BIC    r2, r2, #0x00000001               ; Disable MPEN
STR    r2, [r1, #0x0]
LDR    r1, 0x00000000
MCR    p15, 0, r1, c7, c0, 4             ; WFI
```

The MPLL and SPLL are turned off when the countdown value in SD_CNT is satisfied. For the MPLL, there are a number of conditions that must be satisfied before the Clock Controller module turns off the DPLL. The conditions to be satisfied before the PLL Clock Controller actually turns off the MPLL are as follows:

1. Clock Controller module has successfully mastered the system bus.
2. The A9P_CLK_OFF signal from the ARM9 Platform is active.
3. SDRAM controller has successfully placed the external SDRAM into Self-Refresh mode.
4. After the above conditions are satisfied, the countdown based on the value in the SD_CNT field will be initiated.
5. SD_CNT countdown completes.

When the conditions listed above are satisfied the MPLL and the SPLL will be turned off. The Frequency Premultiplier (FPM) is also disabled in the Sleep mode. The FPM_EN bit (CSCR register) must not be cleared if the FPM is providing the clock source to the DPLL.

When an unmasked interrupt event occurs, the FPM and then the MPLL are re-enabled and the MPLL enable bit (MPEN) automatically restored to its enable setting. The SPLL is restored to its original state based on the setting of the SPEN bit before Sleep mode. If the SPLL was not enabled before entering Sleep mode the SPLL will not be enabled.

The total start-up time from Sleep mode is the sum of the FPM lock time and the DPLL lock time.

In Sleep mode, the i.MX27 device retains all RAM data and register configuration values. Data to output terminals is also maintained and thus will continue to sink/source static current.

NOTE

System software must ensure that if there are any clocks being sourced by the i.MX27 processor to external peripherals (for example, SSI MCLK), then the corresponding PLL must not be turned off. In such cases, the i.MX27 processor must remain in Doze mode.

3.3.4 SDRAM Power Modes

When the SDRAM controller (SDRAMC) is enabled, the external SDRAM operates in Distributed-Refresh mode or in Self-Refresh mode (as shown in Table 3-2). The SDRAM wake-up latency is approximately 20 system clock cycles (HCLK). The SDRAMC can wake up from Self-Refresh mode when it is in a SDRAM cycle.

In Doze and Run mode, the Power Down timers within the SDRAMC can be enabled to cause the SDRAM to enter Power Down mode on detecting no activity. The SDRAMC still controls the refresh and it takes the SDRAM out of the Power Down mode to perform refresh when needed and then put it back into the Power Down mode. In Power Down mode the clock to the SDRAM is gated off and the CKE pin goes low. In addition since the SDRAM will be in self refresh just when the system get into Sleep mode, no bus cycle can access the SDRAM to cause it to exit the Self-Refresh mode. Exit from Self-Refresh mode will happen when the chip will exit the Sleep mode and re-enable the MPEN.

Table 3-2. SDRAM Operation During Power Modes

SDRAM	Run	Doze	Stop
SDRAM	Distributed-refresh, Note 1	Distributed-refresh, Note 1	Self-refresh

3.3.5 Power Management in the PLL Clock Controller

The i.MX27 device has a very efficient clock control scheme that enables clocking control of the modules and devices at various stages. Power management in the i.MX27 device is achieved by controlling the duty cycles of the clock system efficiently. The clocking control scheme is shown in Table 3-3.

Table 3-3. Power Management in the Clock Controller

Device/Signal	Shut-Down Conditions	Wake-Up Conditions
MPLL	When 0 is written to the MPEN bit and the PLL shut-down count times out (for details see the SD_CNT settings in Table 3-6).	When $\overline{\text{IRQ}}$ or $\overline{\text{FIQ}}$ is asserted
SPLL	When 0 is written to the SPEN bit.	When the SPEN bit is set to 1
FPM	When 0 is written to the FPMEN bit.	When the FPMEN bit is set to 1
CLK32	Continuously running.	Continuously running

Most modules in the i.MX27 processor have a module enable bit assigned which must be enabled before the module is active. Enabling the module enables the clock source for the module to be provided for its main operations. The clock input to the dividers from the SPLL is also controlled separately in the same manner.

3.3.6 Power Management Using Frequency Control

The i.MX27 processor has DPTC, but does not support the DVFS feature. The i.MX27 device provides a way for software to save power under different operating conditions.

- Software determines whether to change the operation frequency or not.
- Software uses DPTC to determine whether to reduce or increase the power supply.

- After the power supply has been changed, software can update MPLL configuration.
- It restarts the MPLL and operates with new frequency.

3.4 Memory Map and Register Definition

The PLL Clock Controller module includes six user-accessible 32-bit registers. Table 3-4 provides the memory map for the PLL Clock Controller.

Table 3-4. PLL Clock Controller Memory Map

Address	Register	Access	Reset Value	Section/Page
General Registers				
0x1002_7000 (CSCR)	Clock Source Control Register	R/W	0x33F0_1307	3.4.2/3-10
0x1002_7004 (MPCTL0)	MPLL Control Register 0	R/W	0x0021_1803	3.4.3/3-13
0x1002_7008 (MPCTL1)	MPLL Control Register 1	R/W	0x0000_8000	3.4.4/3-14
0x1002_700C (SPCTL0)	SPLL Control Register 0	R/W	0x8403_1C53	3.4.6/3-16
0x1002_7010 (SPCTL1)	SPLL Control Register 1	R/W	0x0000_8000	3.4.7/3-17
0x1002_7014 (OSC26MCTL)	Oscillator 26M Register	R/W	0x0000_3F00	3.4.8/3-18
0x1002_7018 (PCDR0)	Peripheral Clock Divider Register 0	R/W	0x2008_3403	3.4.9/3-20
0x1002_701C (PCDR1)	Peripheral Clock Divider Register 1	R/W	0x1204_1303	3.4.10/3-22
0x1002_7020 (PCCR0)	Peripheral Clock Control Register 0	R/W	0x0401_01C0	3.4.11/3-23
0x1002_7024 (PCCR1)	Peripheral Clock Control Register 1	R/W	0xFF4B_6848	3.4.12/3-26
0x1002_7028 (CCSR)	Clock Control Status Register	R/W	0x0000_0300	3.4.13/3-29
0x1002_7034 (WKGDCTL)	Wakeup Guard Mode Control Register	R/W	0x0000_0000	3.4.14/3-31

3.4.1 Register Summary

The conventions in Figure 3-3 and Table 3-5 serve as a key for the register summary and individual register diagrams.

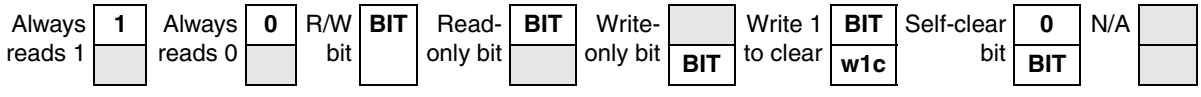


Figure 3-3. Key to Register Fields

Table 3-5 provides a key for register figures and tables and the register summary.

Table 3-5. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

3.4.2 Clock Source Control Register (CSCR)

The Clock Source Control Register controls the various clock sources to the internal modules of the i.MX27 processor. Figure 3-4 shows the register and Table 3-6 provides the field descriptions.

0x1002_7000 (CSCR) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	USB_DIV			0	0	SD_CNT		SSi2_SEL	SSI1_SEL	H264_SEL	MSHC_SEL	SPLL_RESTART	MPLL_RESTART	SP_SEL	MCU_SEL
W																
Reset	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ARM SRC	0	ARMDIV		0	0	AHBDIV		0	0	0	OSC26M_DIV1P5	OSC26M_DIS	FPM_EN	SPEN	MPEN
W																
Reset	0	0	0	1	0	0	1	1	0	0	0	0	0	1	1	1

Figure 3-4. Clock Source Control Register (CSCR)

Table 3-6. Clock Source Control Register Field Descriptions

Field	Description
31 UPDATE_DIS	Disable source selection and divider update until next MPLL lock. This bit is cleared automatically. When reprogramming the PLL and corresponding CSCR settings, this bit must first be set before the CSCR is updated and DPLL is reprogrammed to ensure that erratic clock behavior does not occur.
30–28 USB_DIV	USB Clock Divider. Contains the 3-bit integer divider value for generation of CLK60M. 000 SPLL_CLK divided by 1 001 SPLL_CLK divided by 2 ... 111 SPLL_CLK divided by 8
27–26	Reserved. These bits are reserved and should read 0.
25–24 SD_CNT	Shut-Down Control. Contains the value that determines duration of DPLL clock output before it goes off after a 0 is written to the MPEN or SPEN bit. **Note:** Power controller requests the bus before SPLL shutdown. Any unmasked interrupt event will enable MPLL. 00 DPLL shuts down after the next rising edge of CLK32 is detected and the current bus cycle is completed. A minimum of 16 HCLK cycles is occurs after writing 0 to MPEN bit. 01 DPLL shuts down after second rising edge of CLK32 is detected and the current bus cycle is completed. 10 DPLL shuts down after third rising edge of CLK32 is detected and the current bus cycle is completed. 11 DPLL shuts down after forth rising edge of CLK32 is detected and the current bus cycle is completed.
23 SSI2_SEL	SSI2 Baud Source Select. Selects the clock source to SSI2 fractional divider (SSI2_DIV). 0 Source clock to SSI2 fractional divider from SPLL 1 Source clock to SSI2 fractional divider from MPLL
22 SSI1_SEL	SSI1 Baud Source Select. Selects the clock source to SSI1 fractional divider (SSI1_DIV). 0 Source clock to SSI1 fractional divider from SPLL 1 Source clock to SSI1 fractional divider from MPLL
21 H264_SEL	H264 CCLK Source Select. Selects the clock source to H264 divider (H264_DIV). 0 Source clock to H264 divider is from SPLL 1 Source clock to H264 divider is from MPLL
20 MSHC_SEL	MSHC CCLK Source Select. Selects the clock source to MSHC divider (MSHC_DIV). 0 Source clock to MSHC divider is from SPLL 1 Source clock to MSHC divider is from MPLL
19 SPLL_RESTART	SPLL Restart. Restarts SPLL at the new assigned frequency. SPLL_RESTART self-clears after 1 (min) or 2 (max) cycles of CLK32. 0 No Effect 1 Restarts SPLL at new frequency
18 MPLL_RESTART	MPLL Restart. Restarts MPLL at the new assigned frequency. MPLL_RESTART self-clears after 1 (min) or 2 (max) cycles of CLK32. 0 No Effect 1 Restarts MPLL at new frequency
17 SP_SEL	SPLL Select. Selects clock source of SPLL input. When set, the external high frequency clock input is selected. 0 Clock source is the internal premultiplier. Register map shows this bit as reserved also conflicts with 1 Clock source is the external high frequency clock

Table 3-6. Clock Source Control Register Field Descriptions (continued)

Field	Description
16 MCU_SEL	MPLL Select. Selects clock source of MPLL input. When set, the external high frequency clock input is selected. 0 Clock source is the internal premultiplier. 1 Clock source is the external high frequency clock.
15 ARM SRC	ARMSRC. It selects the ARM clock source. 0 MPLL CLK * 2 / 3 1 MPLL CLK
13–12 ARM_DIV	ARM_DIV. Divider value for arm clk. 00 divide by 1 01 divide by 2 10 divide by 3 11 divide by 4
11–10	Reserved
9-8 AHB_DIV	AHB_DIV. Divider value for AHB clk. 00 divide by 1 01 divide by 2 10 divide by 3 11 divide by 4
7–5	Reserved
4 OSC26M_DIV1P5	Oscillator 26M Divide Enable. Divides osc26m output by 1 or 1.5. 0 osc26m output divide by 1 (default) 1 osc26m output divide by 1.5
3 OSC26M_DIS	Oscillator Disable. Disables the internal (on-chip) 26 MHz oscillator circuit when this bit is set to 1. 0 Enable the internal 26 MHz oscillator circuit 1 Disable the internal 26 MHz oscillator circuit
2 FPM_EN	Frequency Premultiplier Enable. Enables/disables FPM when set/cleared. This bit is set automatically on system reset. When the software writes a 0 to this bit, FPM is shut down immediately. This bit must remain at 1 prior and during Sleep mode if FPM is providing the source to the DPLL. 0 Disable the frequency premultiplier circuit 1 Enable the frequency premultiplier circuit
1 SPEN	Serial Peripheral PLL Enable. Enables/disables the SPLL. When software writes 0 to SPEN, SPLL shuts down after timeout determined by SD_CNT. SPEN sets automatically when SPLLEN asserts, and on system reset. 0 Serial Peripheral PLL disabled 1 Serial Peripheral PLL enabled
0 MPEN	MPLL Enable. Enables/disables the MPLL. When software writes 0 to MPEN, MPLL shuts down after SDCNT timeout. MPEN sets automatically when MPLLEN asserts, and on system reset. 0 MCU and Serial PLL disabled 1 MCU and Serial PLL enabled

NOTE

When PRESC and BCLKDIV are modified at the same time, it must be performed in two programming steps: The first step is to change BCLKDIV, and then to change PRESC.

3.4.3 MPLL Control Register 0 (MPCTL0)

The MCU and System PLL Control Register 0 (MPCTL0) is a 32-bit register that controls the operation of the MPLL. The MPCTL0 control bits are described in the following sections.

Figure 3-5 shows the register and Table 3-7 provides the field descriptions.

The following is the recommended procedure for changing the MPLL settings:

1. Program the desired values of PD, MFD, MFI, and MFN into the MPCTL0.
2. Set the MPLL_RESTART bit in the CSCR (it will self-clear).
3. New MPLL settings will take effect.
4. The new PLL clock output is valid upon the assertion of the DPLL lock flag.

0x1002_7004 (MPCTL0) Access: User R/W

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

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

Figure 3-5. MPLL Control Register 0 (MPCTL0)

Table 3-7. MPLL Register 0 Field Descriptions

Field	Description
31 CPLM	Phase Lock Mode. DPLL operates in the Frequency Only Lock mode (FOL) when CPLM bit is cleared, and in Frequency and Phase Lock mode (FPL) when the bit is set. FPL mode can be used for both integer and fractional multiplication factor, but phase skew elimination is accomplished only for integer MF. 0 FOL 1 FPL
30	Reserved. This bit is reserved and should read 0.
29–26 PD	Predivider Factor. Defines the predivider factor (PD) applied to MPLL input frequency. PD is an integer between 0 and 15 (inclusive). PD is chosen to ensure that the resulting output frequency remains within the specified range. When a new value is written into PD, MPLL loses its lock; and after a time delay, MPLL re-locks. MPLL output is determined by Equation 3-1. 0000 0 0001 1 … 1111 15

Table 3-7. MPLL Register 0 Field Descriptions (continued)

Field	Description
25–16 MFD	Multiplication Factor (Denominator Part). Defines the denominator part of BRM value for MF. When a new value is written into the MFD bits, MPLL loses its lock; and after a time delay, MPLL re-locks. 000 Reserved 001 1 … 3FF 1023
15–14	Reserved. These bits are reserved and should read 0.
13–10 MFI	Multiplication Factor (Integer). Defines the integer part of BRM value for MF. MFI is encoded so that MFI < 5 results in MFI = 5. When a new value is written into the MFI bits, PLL loses its lock: and after a time delay, PLL re-locks. VCO oscillates at a frequency determined by Equation 3-1. Where PD is the division factor of the predivider, MFI is the integer part of total MF, MFN is the numerator of the fractional part of MF, and MFD is its denominator part. MF is chosen to ensure that the resulting VCO output frequency remains within the specified range. 0000–01015 0110 6 ... 1111 15
9–0 MFN	Multiplication Factor (Numerator). Defines the numerator of BRM value for MF. The MFN is the only part in the DPLL Configuration that can be changed after the DPLL was locked without resetting the DPLL (on the fly).The bit 9 is the sign bit. When MFN is zero, the circuitry for fractional division is disabled to save power. 000 0 001 1 ... 1FE 510 1FF reserved ... 3FE –510 3FF Reservoir

The recommended settings for MPLL and SPLL that produce the least amount of signal jitter are shown in Table 3-8.

Table 3-8. Recommend Settings for Frequency Stability

Ref Frequency	Target Frequency	MFI	MFN	MFD	PD	MPCTL0 Setting	Actual Calculated Frequency
32.768 kHz	399 MHz	5	469	495	0	0x01EF15D5	399.000
32.000 kHz	399 MHz	6	3	21	0	0x00211803	398.998
26 MHz	399 MHz	7	35	51	0	0x00331C23	399

3.4.4 MCU and System PLL Control Register 1 (MPCTL1)

The MCU and System PLL Control Register 1 (MPCTL1) is a 32-bit register that directs the operation of the on-chip MCU PLL. Figure 3-6 shows the register and Table 3-9 provides the field descriptions.

0x1002_7008 (MPCTL1) Access: User R/W

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

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

Figure 3-6. MCU and System PLL Control Register 1 (MPCTL1)

Table 3-9. MCU and System PLL Control Register 1 Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15 LF	Lock Flag. Indicates whether MPLL is locked or not. When set, MPLL clock output is valid. When cleared, MPLL clock output remains at logic high. 0 MPLL is not locked. 1 MPLL is locked.
14–7	Reserved. These bits are reserved and should read 0.
6 BRMO	BRM Order. Controls the BRM order which affects jitter performance of MPLL. The first order BRM is used if a MF fractional part is more than 1/10 and less than 9/10. In other cases, second order BRM is used. BRMO bit is cleared by a hardware reset. A delay of reference cycles is required between two write accesses to BRMO. 0 BRM contains first order. 1 BRM contains second order.
5–0	Reserved. These bits are reserved and should read 0.

3.4.5 Programming the Serial Peripheral PLL (SPLL)

One of the clock frequencies that the SPLL generates is for the USB OTG module (CLK60M). Its frequency is set to 60 MHz using the SPLL control registers assuming a default input clock frequency 32.768 MHz. This input clock frequency assumes a 32 kHz crystal input. The predivider/multiplier output depends on the input clock frequency. Recommended settings are provided for the Serial Peripheral PLL as shown in Table 3-10.

Table 3-10. Serial PLL Multiplier Factor

Ref Frequency	Target Frequency	MFI	MFN	MFD	PD	SPCTL0 Setting	Actual Calculated Frequency
32.768 kHz	300 MHz	8	111	117	1	0x0475206F	299.99937 MHz
32.768 kHz	240 MHz	7	9	58	1	0x043A1C09	239.99950 MHz
32 kHz	300 MHz	9	25	160	1	0x04A02419	300.00020 MHz
32 kHz	240 MHz	7	83	255	1	0x04FF1C53	240 MHz
26 MHz	300 MHz	11	7	12	1	0x040C2C07	300 MHz
26 MHz	240 MHz	9	3	12	1	0x040C2403	240 MHz

3.4.6 SPLL Control Register 0 (SPCTL0)

The Serial Peripheral PLL Control Register 0 (SPCTL0) is a 32-bit register that controls the operation of the SPLL. The SPCTL0 control bits are described in the following sections.

Figure 3-7 shows the register and Table 3-11 provides the field descriptions.

The following is a procedure for changing the Serial Peripheral PLL settings:

1. Program the desired values of PD, MFD, MFI, and MFN into the SPCTL0.
2. Set the SPLL_RESTART bit in the CSCR (it will self-clear).
3. New PLL settings will take effect.
4. The new PLL clock output is valid upon the assertion of the DPLL lock flag.

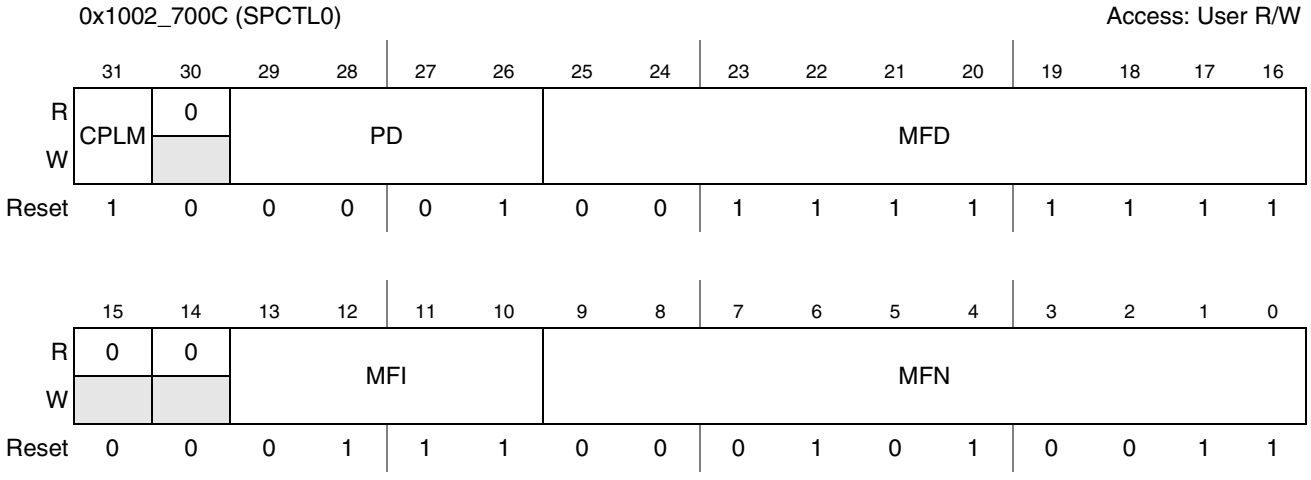


Figure 3-7. SPLL Control Register 0 (SPCTL0)

Table 3-11. SPLL Control Register 0 Field Descriptions

Field	Description
31 CPLM	Phase Lock Mode. DPLL operates in the Frequency Only Lock mode (FOL) when CPLM bit is cleared, and in Frequency and Phase Lock mode (FPL) when the bit is set. FPL mode can be used for both integer and fractional multiplication factor, but phase skew elimination is accomplished only for integer MF. 0 FOL 1 FPL
30	Reserved. These bit is reserved and should read 0.
29–26 PD	Predivider Factor. Defines the predivider factor (PD) that is applied to the PLL input frequency. PD is an integer between 0 and 15 (inclusive). SPLL oscillates at a frequency determined by Equation 3-1. PD is chosen to ensure that the resulting VCO output frequency remains within the specified range. When a new value is written into the PD bits, SPLL loses its lock: and after a time delay, SPLL re-locks. 0000 0 0001 1 … 111115
25–16 MFD	Multiplication Factor (Denominator Part). Defines the denominator part of BRM value for the MF. When a new value is written into the MFD9 to MFD0 bits, PLL loses its lock: and after a time delay, PLL re-locks. 000 Reserved 001 1 … 3FF 1023
15–14	Reserved. These bits are reserved and should read 0.
13–10 MFI	Multiplication Factor (Integer Part). Defines the integer part of BRM value for MF. MFI is decoded so that MFI < 5 results in MFI = 5. SPLL oscillates at a frequency determined by Equation 3-1. Where PD is the division factor of the predivider, MFI is the integer part of total MF, MFN is the numerator of fractional part of MF, and MFD is the denominator part of MF. MF is chosen to ensure that the resulting VCO output frequency remains within the specified range. When a new value is written into the MFI bits, SPLL loses its lock; and after a time delay, SPLL re-locks. 0000–01015 0110 6 ... 1111 15
9–0 MFN	Multiplication Factor (Numerator). Defines the numerator of BRM value for MF. The MFN is the only part in the DPLL Configuration that can be changed after the DPLL was locked without resetting the DPLL (on the fly).The bit 9 is the sign bit. When MFN is zero, the circuitry for fractional division is disabled to save power. 0x0000 0x0011 ... 0x1FE 510 0x1FF Reserved ... 0x3FE-510 0x3FF Reserved

3.4.7 SPLL Control Register 1 (SPCTL1)

The Serial PLL Control Register 1 (SPCTL1) is a 32-bit read/write register that directs the operation of the SPLL. The SPCTL1 control bits are described in this section. Figure 3-8 shows the register and Table 3-12 provides the field descriptions.

0x1002_7010 (SPCTL1) Access: User R/W

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

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

Figure 3-8. SPLL Control Register 1 (SPCTL1)

Table 3-12. Serial Peripheral PLL Control Register 1 Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15 LF	Lock Flag. Indicates whether SPLL is locked or not. When set, SPLL clock output is valid. When cleared, SPLL clock output remains at logic high. 0 SPLL is not locked. 1 SPLL is locked.
14–7	Reserved. These bits are reserved and should read 0.
6 BRMO	BRM Order. Controls the BRM order which affect SPLL jitter performance. The first order BRM is used if a MF fractional part is more than 1/10 and less than 9/10. In other cases, second order BRM is used. BRMO bit is cleared by a hardware reset. A delay of reference cycles is required between two write accesses to BRMO. 0 BRM contains first order. 1 BRM contains second order.
5–0	Reserved. These bits are reserved and should read 0.

3.4.8 Oscillator 26M Register

This register is use to program the 26 MHz oscillator test modes as well as the gain control. Trimming of the oscillator is necessary only on initial power up; the trim may be stored in Flash for future reference.

Figure 3-9 shows the register and Table 3-13 provides the field descriptions.

0x1002_7014 (OSC26MCTL) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OSC26M_PEAK	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 3-9. Oscillator 26M Control Register (OSC26MCTL)

Table 3-13. Oscillator 26M Control Register Field Descriptions

Field	Description
31–18	Reserved. These bits are reserved and should read 0.
17–16 OSC26M_PEAK	OSC26M_PEAK. These bits indicates the current amplitude status from the oscillator. 00 Amplitude in desired operating range 01 Amplitude too low; trim higher 10 Amplitude too high; trim lower 11 Invalid state
13–8 AGC	Automatic Gain Control. These bits sets the magnitude of crystal oscillations based on OSC26M_PEAK status. Optimum settings for these bits is determined using the algorithm in Section 3.4.8.1, "Adjusting the 26 MHz Oscillator Trim."
7–0	Reserved. These bits are reserved and should read 0.
Reserved Bits 7–0	Reserved—These bits are reserved and should read 0.

3.4.8.1 Adjusting the 26 MHz Oscillator Trim

To ensure a proper startup of the 26 MHz oscillator on power-up or system reset use the following steps to determine the optimum trim of the oscillator AGC. To ensure proper startup of 26 MHz oscillator on power-up or system reset, use Example 3-2 to determine optimum trim. This algorithm must be run to determine optimum AGC setting. Once done, software must read the trim value from external memory and write it to OSC26M_AGC[5:0].

Example 3-2. 26 MHz Oscillator Trim Programming Algorithm

1. At power up or system reset, OSC26M_AGC[5:0] bits in the OSC26MCTL register are reset to logic 1 (done in hardware, no software interaction required).
2. Read the peak amplitude value in bits OSC26M_PEAK[1:0] in the OSC26MCTL register.
3. If the amplitude is not in the desired range, adjust by decrementing the OSC26M_AGC[5:0] by 1 count.

4. Wait at least 30.5 us (1 cycle of 32 kHz clock) for system to update OSC26M_PEAK bits.
5. Repeat steps 2 to 4 until trimmed in desired range.
6. Decrement 4 additional counts to provide a margin of error for temperature drift.
7. Store trim value in an external memory—that is, Flash, for future use.

It is suggested that the proceeding algorithm be run to determine the optimum AGC setting. Once this is done on power-up or system reset the software must read the trim value from the external memory and write it to the OSC26M_AGC[5:0].

3.4.9 Peripheral Clock Divider Register 0 (PCDR0)

The Peripheral Clock Divider Register 0 (PCDR0) contains the divider values for the peripheral clock dividers in the PLL Clock Controller. Peripherals in the i.MX27 device require special clock frequency which is divided down from the MPLL and the SPLL clock output. Each of these peripheral modules receive their clock input from the respective clock divider. These modules will still have the clock gating scheme as with other modules for power saving advantages.

Figure 3-10 shows the register and Table 3-14 provides the field descriptions.

Table 3-16 lists the clock sources associated with the i.MX27 peripherals given in the PCDR0.

0x1002_7018 (PCDR0) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SSI2DIV						CLKO_EN		CLKO_DIV				SSI1DIV			
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	H264DIV						NFCDIV				MSHCDIV					
Reset	0	0	0	1	0	1	0	0	1	1	0	0	0	0	1	1

Figure 3-10. Peripheral Clock Divider Register 0 (PCDR0)

Table 3-14. Peripheral Clock Divider Register 0 Field Descriptions

Field	Description
31–26 SSI2DIV	SSI2 Baud Clock Divider. Contains 6-bit fractional divider that produces the clock for SSI2CLK clock signal for the peripherals. The value of the divider starts from 0. 0 2 1 2.5 2 3 .... 63 33.5 **Note:** Formula for all others: clkin / (2 + 0.5 * SSI2DIV)
25 CLKO_EN	Clock Out Enable. Enable bit for CLKO pin. 0 disable CLKO output 1 enable CLKO output
24–22 CLKO_DIV	Clock Out Divider. Contains the 3-bit divider that divides output clocks to CLKO pin. 000 Divide by 1 001 Divide by 2 … 111 Divide by 8
21–16 SSI1DIV	SSI1 Baud Clock Divider. Contains 6-bit fractional divider that produces the clock for SSI1CLK clock signal for the peripherals. The value of the divider starts from 0. 0 2 1 2.5 2 3 .... 63 33.5 **Note:** Formula for all others: clkin / (2 + 0.5 * SSI1DIV)
15–10 H264DIV	H264 Baud Clock Divider. Contains 6-bit fractional divider that produces the clock for H264CLK clock signal for the peripherals. The value of the divider starts from 0. 0 2 1 2.5 2 3 .... 63 33.5 **Note:** Formula for all others: clkin / (2 + 0.5 * H264DIV)
9–6 NFCDIV	NAND Flash Controller Clock Divider. Contains 4-bit divider that produces the clock for NFCCLK clock signal of the NAND Flash Controller. 0000 Divide by 1 0001 Divide by 2 … 1111 Divide by 16
5–0 MSHCDIV	MSHC Clock Divider. Contains 6-bit divider that produces the clock for MSHCCLK clock signal of MSHC. 000000 Divide by 1 000001 Divide by 2 … 111111 Divide by 64

3.4.10 Peripheral Clock Divider Register 1 (PCDR1)

The Peripheral Clock Divider Register 1 (PCDR1) contains the divider values for the peripheral clock dividers in the PLL Clock Controller. Peripherals in i.MX27 requires special clock frequency which is divided down from the MPLL and the SPLL clock output. Each of these peripheral modules receive their clock input from the respective clock divider. These modules will still have the clock gating scheme as with other modules for power saving advantages. Figure 3-11 shows the register and Table 3-15 provides the field descriptions.

Table 3-16 lists the clock sources associated with the i.MX27 peripherals given in the PCDR1.

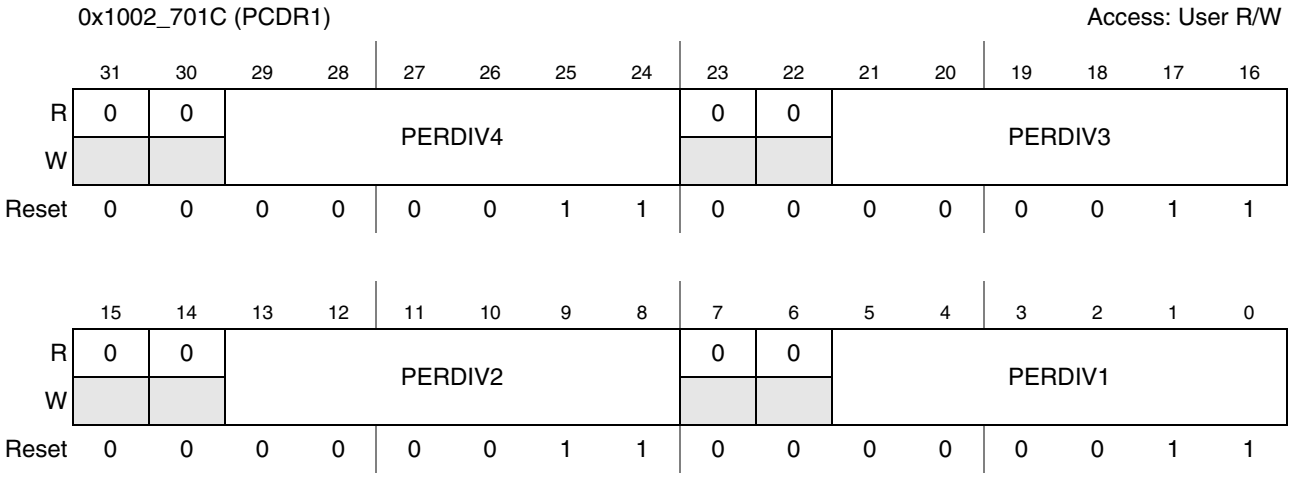


Figure 3-11. Peripheral Clock Divider Register 1(PCDR1)

Table 3-15. Peripheral Clock Divider Register 1 Field Descriptions

Field	Description
31–30	These are reserved bits and should read 0.
29–24 PERDIV4	Peripheral Clock Divider 4. Contains 6-bit integer divider that produces PERCLK4 clock signal for CSI MCLK Clock. 000000 Divide by 1 000001 Divide by 2 … 111111 Divide by 64
23–22	These are reserved bits and should read 0.
21–16 PERDIV3	Peripheral Clock Divider 3. Contains 6-bit integer divider that produces PERCLK3 clock signal for LCDC Pixel Clock. 000000 Divide by 1 000001 Divide by 2 … 111111 Divide by 64
15–14	These are reserved bits and should read 0.

Table 3-15. Peripheral Clock Divider Register 1 Field Descriptions (continued)

Field	Description
13–8 PERDIV2	Peripheral Clock Divider 2. Contains 6-bit integer divider that produces PERCLK2 clock signal for the peripheral 2set (CSPI and SDHC). 000000 Divide by 1 000001 Divide by 2 … 111111 Divide by 64
7–6	These are reserved bits and should read 0.
5–0 PERDIV1	Peripheral Clock Divider 1. Contains 6-bit integer divider that produces PERCLK1 clock signal for the peripheral 1 set (UART, GPT, PWM). 000000 Divide by 1 000001 Divide by 2 … 111111 Divide by 64

Table 3-16. Clock Sources for i.MX27 Peripherals

Clock Source	Peripherals	Clock Source	Peripherals
SSI1CLK	SSI1	NFCCLK	NFC
SSI2CLK	SSI2	MSHCCLK	MSHC
H264CCLC	H264		

3.4.11 Peripheral Clock Control Register 0 (PCCR0)

The Peripheral Clock Control Register 0 (PCCR0) provides additional power saving capabilities by controlling the clocks in the i.MX27 modules. It also controls the clock source for Bootstrap mode. The PCCR0 allows for gating of HCLK to modules or peripherals that access the AHB bus and perform AHB bus transfers and also allows for gating of the ipg clk (PERCLK) to specific peripherals.

Figure 3-12 shows the register and Table 3-17 provides the field descriptions.

0x1002_7020 (PCCR0) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CSPI1_EN	CSPI2_EN	CSPI3_EN	DMA_EN	EMMA_EN	FEC_EN	GPIO_EN	GPT1_EN	GPT2_EN	GPT3_EN	GPT4_EN	GPT5_EN	GPT6_EN	I2C1_EN	I2C2_EN	IIM_EN
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KPP_EN	LCDC_EN	MSHC_EN	OWIRE_EN	PWM_EN	0	RTC_EN	RTIC_EN	SAHARA_EN	SCC_EN	SDHC1_EN	SDHC2_EN	SDHC3_EN	SLCDC_EN	SSI1_EN	SSI2_EN
Reset	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0

Figure 3-12. Peripheral Clock Control Register 0 (PCCR0)

Table 3-17. Peripheral Clock Control Register 0 Field Descriptions

Field	Description
31 CSPI1_EN	CSPI1 IPG Clock Enable. Enables/Disables IPG clock input to CSPI1 module. 0 CSPI1 IPG clock input is disabled. 1 CSPI1 IPG clock input is enabled.
30 CSPI2_EN	CSPI2 IPG Clock Enable. Enables/Disables IPG clock input to CSPI2 module. 0 CSPI2 IPG clock input is disabled. 1 CSPI2 IPG clock input is enabled.
29 CSPI3_EN	CSPI3 IPG Clock Enable. Enables/Disables IPG clock input to CSPI3 module. 0 CSPI3 IPG clock input is disabled. 1 CSPI3 IPG clock input is enabled.
28 DMA_EN	DMA IPG Clock Enable. Enables/Disables IPG clock input to DMA module. 0 DMA IPG clock input is disabled. 1 DMA IPG clock input is enabled.
27 EMMA_EN	EMMA IPG Clock Enable. Enables/Disables IPG clock input to EMMA module. 0 EMMA IPG clock input is disabled. 1 EMMA IPG clock input is enabled.
26 FEC_EN	FEC IPG Clock Enable. Enables/Disables IPG clock input to FEC module. 0 FEC IPG clock input is disabled. 1 FEC IPG clock input is enabled.
25 GPIO_EN	GPIO IPG Clock Enable. Enables/Disables IPG clock input to GPIO module. 0 GPIO IPG clock input is disabled. 1 GPIO IPG clock input is enabled.
24 GPT1_EN	GPT1 IPG Clock Enable. Enables/Disables IPG clock input to GPT1 module. 0 GPT1 IPG clock input is disabled. 1 GPT1 IPG clock input is enabled.

Table 3-17. Peripheral Clock Control Register 0 Field Descriptions (continued)

Field	Description
23 GPT2_EN	GPT2 IPG Clock Enable. Enables/Disables IPG clock input to GPT2 module. 0 GPT2 IPG clock input is disabled. 1 GPT2 IPG clock input is enabled.
22 GPT3_EN	GPT3 IPG Clock Enable. Enables/Disables IPG clock input to GPT3 module. 0 GPT3 IPG clock input is disabled. 1 GPT3 IPG clock input is enabled.
21 GPT4_EN	GPT4 IPG Clock Enable. Enables/Disables IPG clock input to GPT4 module. 0 GPT4 IPG clock input is disabled. 1 GPT4 IPG clock input is enabled.
20 GPT5_EN	GPT5 IPG Clock Enable. Enables/Disables IPG clock input to GPT5 module. 0 GPT5 IPG clock input is disabled. 1 GPT5 IPG clock input is enabled.
19 GPT6_EN	GPT6 IPG Clock Enable. Enables/Disables IPG clock input to GPT6 module. 0 GPT6 IPG clock input is disabled. 1 GPT6 IPG clock input is enabled.
18 I2C1_EN	I2C1 IPG Clock Enable. Enables/Disables IPG clock input to I2C1 module. 0 I2C1 IPG clock input is disabled. 1 I2C1 IPG clock input is enabled.
17 I2C2_EN	I2C2 IPG Clock Enable. Enables/Disables IPG clock input to I2C2 module. 0 I2C2 IPG clock input is disabled. 1 I2C2 IPG clock input is enabled.
16 IIM_EN	IIM IPG Clock Enable. Enables/Disables IPG clock input to IIM module. 0 IIM IPG clock input is disabled. 1 IIM IPG clock input is enabled.
15 KPP_EN	KPP IPG Clock Enable. Enables/Disables IPG clock input to KPP module. 0 KPP IPG clock input is disabled. 1 KPP IPG clock input is enabled.
14 LCDC_EN	LCDC IPG Clock Enable. Enables/Disables IPG clock input to LCDC module. 0 LCDC IPG clock input is disabled. 1 LCDC IPG clock input is enabled.
13 MSHC_EN	MSHC IPG Clock Enable. Enables/Disables IPG clock input to MSHC module. 0 MSHC IPG clock input is disabled. 1 MSHC IPG clock input is enabled.
12 OWIRE_EN	OWIRE IPG Clock Enable. Enables/Disables IPG clock input to OWIRE module. 0 OWIRE IPG clock input is disabled. 1 OWIRE IPG clock input is enabled.
11 PWM_EN	PWM IPG Clock Enable. Enables/Disables IPG clock input to PWM module. 0 PWM IPG clock input is disabled. 1 PWM IPG clock input is enabled.
10	Reserved. This bit is reserved.
9 RTC_EN	RTC IPG Clock Enable. Enables/Disables IPG clock input to RTC module. 0 RTC IPG clock input is disabled. 1 RTC IPG clock input is enabled.

Table 3-17. Peripheral Clock Control Register 0 Field Descriptions (continued)

Field	Description
8 RTIC_EN	RTIC IPG Clock Enable. Enables/Disables IPG clock input to RTIC module. 0 RTIC IPG clock input is disabled. 1 RTIC IPG clock input is enabled.
7 SAHARA_EN	SAHARA IPG Clock Enable. Enables/Disables IPG clock input to SAHARA module. 0 SAHARA IPG clock input is disabled. 1 SAHARA IPG clock input is enabled.
6 SCC_EN	SCC IPG Clock Enable. Enables/Disables IPG clock input to SCC_EN module. 0 SCC_EN IPG clock input is disabled. 1 SCC_EN IPG clock input is enabled.
5 SDHC1_EN	SDHC1 IPG Clock Enable. Enables/Disables IPG clock input to SDHC1 module. 0 SDHC1 IPG clock input is disabled. 1 SDHC1 IPG clock input is enabled.
4 SDHC2_EN	SDHC2 IPG Clock Enable. Enables/Disables IPG clock input to SDHC2 module. 0 SDHC2 IPG clock input is disabled. 1 SDHC2 IPG clock input is enabled.
3 SDHC3_EN	SDHC3 IPG Clock Enable. Enables/Disables IPG clock input to SDHC3 module. 0 SDHC3 IPG clock input is disabled. 1 SDHC3 IPG clock input is enabled.
2 SLCDC_EN	SLCDC IPG Clock Enable. Enables/Disables IPG clock input to SLCDC module. 0 SLCDC IPG clock input is disabled. 1 SLCDC IPG clock input is enabled.
1 SSI1_EN	SSI1 IPG Clock Enable. Enables/Disables IPG clock input to SSI1 module. 0 SSI1 IPG clock input is disabled. 1 SSI1 IPG clock input is enabled.
0 SSI2_EN	SSI2 IPG Clock Enable. Enables/Disables IPG clock input to SSI2 module. 0 SSI2 IPG clock input is disabled. 1 SSI2 IPG clock input is enabled.

3.4.12 Peripheral Clock Control Register 1 (PCCR1)

The Peripheral Clock Control Register 1 (PCCR1) provides additional power saving capabilities by controlling the clocks in the i.MX27 modules. It also controls the clock source for Bootstrap mode. The PCCR1 allows for gating of the ipg clk (PERCLK) to specific peripherals. Figure 3-13 shows the register and Table 3-18 provides the field descriptions.

0x1002_7024 (PCCR1) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	UART1_EN	UART2_EN	UART3_EN	UART4_EN	UART5_EN	UART6_EN	USB_EN	WDT_EN	HCLK_ATA	HCLK_BROM	HCLK_CSI	HCLK_DMA	HCLK_EMI	HCLK_EMMA	HCLK_FEC	HCLK_H264
Reset	1	1	1	1	1	1	1	1	0	1	0	0	1	0	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HCLK_LCDC	HCLK_RTIC	HCLK_SAHARA	HCLK_SLCDC	HCLK_USB	PERCLK1_EN	PERCLK2_EN	PERCLK3_EN	PERCLK4_EN	H264_BAUDEN	SSI1_BAUDEN	SSI2+BAUDEN	NFC_BAUDEN	MSHC_BAUDEN	0	0
W																
Reset	0	1	1	0	1	0	0	0	0	1	0	0	1	0	0	0

Figure 3-13. Peripheral Clock Control Register 1(PCCR1)

Table 3-18. Peripheral Clock Control Register 1 Field Descriptions

Field	Description
31 UART1_EN	UART1 IPG Clock Enable. Enables/Disables IPG clock input to UART1 module. 0 UART1 IPG clock input is disabled. 1 UART1 IPG clock input is enabled.
30 UART2_EN	UART2 IPG Clock Enable. Enables/Disables IPG clock input to UART2 module. 0 UART2 IPG clock input is disabled. 1 UART2 IPG clock input is enabled.
29 UART3_EN	UART3 IPG Clock Enable. Enables/Disables IPG clock input to UART3 module. 0 UART3 IPG clock input is disabled. 1 UART3 IPG clock input is enabled.
28 UART4_EN	UART4 IPG Clock Enable. Enables/Disables IPG clock input to UART4 module. 0 UART4 IPG clock input is disabled. 1 UART4 IPG clock input is enabled.
27 UART5_EN	UART5 IPG Clock Enable. Enables/Disables IPG clock input to UART5 module. 0 UART5 IPG clock input is disabled. 1 UART5 IPG clock input is enabled.
26 UART6_EN	UART6 IPG Clock Enable. Enables/Disables IPG clock input to UART6 module. 0 UART6 IPG clock input is disabled. 1 UART6 IPG clock input is enabled.
25 USB_EN	USB IPG Clock Enable. Enables/Disables IPG clock input to USB module. 0 USB IPG clock input is disabled. 1 USB IPG clock input is enabled.
24 WDT_EN	WDT IPG Clock Enable. Enables/Disables IPG clock input to WDT module. 0 WDT IPG clock input is disabled. 1 WDT IPG clock input is enabled.

Table 3-18. Peripheral Clock Control Register 1 Field Descriptions (continued)

Field	Description
23 HCLK_ATA	ATA AHB Clock Enable. Enables/Disables AHB clock input to ATA module. 0 ATA AHB clock input is disabled. 1 ATA AHB clock input is enabled.
22 HCLK_BROM	BROM AHB Clock Enable. Enables/Disables AHB clock input to BROM module. 0 BROM AHB clock input is disabled. 1 BROM AHB clock input is enabled.
21 HCLK_CSI	CSI AHB Clock Enable. Enables/Disables AHB clock input to CSI module. 0 CSI AHB clock input is disabled. 1 CSI AHB clock input is enabled.
20 HCLK_DMA	DMA AHB Clock Enable. Enables/Disables AHB clock input to DMA module. 0 DMA AHB clock input is disabled. 1 DMA AHB clock input is enabled.
19 HCLK_EMI	EMI AHB Clock Enable. Enables/Disables AHB clock input to EMI module. 0 EMI AHB clock input is disabled. 1 EMI AHB clock input is enabled.
18 HCLK_EMMA	EMMA AHB Clock Enable. Enables/Disables AHB clock input to EMMA module. 0 EMMA AHB clock input is disabled. 1 EMMA AHB clock input is enabled.
17 HCLK_FEC	FEC AHB Clock Enable. Enables/Disables AHB clock input to FEC module. 0 FEC AHB clock input is disabled. 1 FEC AHB clock input is enabled.
16 HCLK_H264	H264 AHB Clock Enable. Enables/Disables AHB clock input to H264 module. 0 H264 AHB clock input is disabled. 1 H264 AHB clock input is enabled.
15 HCLK_LCDC	LCDC AHB Clock Enable. Enables/Disables AHB clock input to LCDC module. 0 LCDC AHB clock input is disabled. 1 LCDC AHB clock input is enabled.
14 HCLK_RTIC	RTIC AHB Clock Enable. Enables/Disables AHB clock input to RTIC module. 0 RTIC AHB clock input is disabled. 1 RTIC AHB clock input is enabled.
13 HCLK_SAHARA	SAHARA AHB Clock Enable. Enables/Disables AHB clock input to SAHARA module. 0 SAHARA AHB clock input is disabled. 1 SAHARA AHB clock input is enabled.
12 HCLK_SLCDC	SLCDC AHB Clock Enable. Enables/Disables AHB clock input to SLCDC module. 0 SLCDC AHB clock input is disabled. 1 SLCDC AHB clock input is enabled.
11 HCLK_USB	USB AHB Clock Enable. Enables/Disables AHB clock input to USB module. 0 USB AHB clock input is disabled. 1 USB AHB clock input is enabled.
10 PERCLK1_EN	PERCLK1 Clock Enable. Enables/Disables Peripheral clock1. 0 Peripheral clock1 is disabled. 1 Peripheral clock1 is enabled.

Table 3-18. Peripheral Clock Control Register 1 Field Descriptions (continued)

Field	Description
9 PERCLK2_EN	PERCLK2 Clock Enable. Enables/Disables Peripheral clock2. 0 Peripheral clock2 is disabled. 1 Peripheral clock2 is enabled.
8 PERCLK3_EN	PERCLK3 Clock Enable. Enables/Disables Peripheral clock3. 0 Peripheral clock3 is disabled. 1 Peripheral clock3 is enabled.
7 PERCLK4_EN	PERCLK4 Clock Enable. Enables/Disables Peripheral clock4. 0 Peripheral clock4 is disabled. 1 Peripheral clock4 is enabled.
6 H264_BAUDEN	H264 BAUD Clock Enable. Enables/Disables BAUD clock input to H264 module. 0 H264 BAUD clock input is disabled. 1 H264 BAUD clock input is enabled.
5 SSI1_BAUDEN	SSI1 BAUD Clock Enable. Enables/Disables BAUD clock input to SSI1 module. 0 SSI1 BAUD clock input is disabled. 1 SSI1 BAUD clock input is enabled.
4 SSI2_BAUDEN	SSI2 BAUD Clock Enable. Enables/Disables BAUD clock input to SSI2 module. 0 SSI2 BAUD clock input is disabled. 1 SSI2 BAUD clock input is enabled.
3 NFC_BAUDEN	NFC BAUD Clock Enable. Enables/Disables BAUD clock input to NFC module. 0 NFC BAUD clock input is disabled. 1 NFC BAUD clock input is enabled.
2 MSHC_BAUDEN	MSHC BAUD Clock Enable. Enables/Disables BAUD clock input to MSHC module. 0 MSHC BAUD clock input is disabled. 1 MSHC BAUD clock input is enabled.
1–0	Reserved. These bits are reserved and should read 0.

3.4.13 Clock Control Status Register (CCSR)

The Clock Control Status Register (CCSR) provides information on the configuration of the Analog and Digital block. The clocks within the chip can also be monitored by the CLKO_SEL programming.

Figure 3-14 shows the register and Table 3-19 provides the field descriptions.

0x1002_7028 (CCSR) Access: User R/W

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	32K _SR	0	0	0	0	0	CLKMODE		0	0	0	CLKO_SEL				
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Figure 3-14. Clock Control Status Register (CCSR)

Table 3-19. Clock Control Status Register Field Descriptions

Field	Description
31–16	Reserved. These bit are reserved and should read 0.
15 32K_SR	32K Status Register. It contains status information of 32 KHz clock. It is cleared to zero during the assertion of HARD_ASYNC_RESET signal. The sampled 32KHz clock phase is continuously registered into the bit upon the de-assertion of HARD_ASYNC_RESET signal. 0 CLK32 in low phase 1 CLK32 in high phase
14–12	Reserved. These bits are reserved and should read 0.
9–8 CLKMODE	CLKMODE. Determines the configuration of FPM, OSC26M and DPLL on the chip. Its reset value depends on CLKMODE input signals. 00 DPLL, FPM, OSC26M bypassed 01 FPM bypassed. 10 FPM and OSC26M bypassed 11 FPM and DPLL in use (Default)

Table 3-19. Clock Control Status Register Field Descriptions (continued)

Field	Description
7–5	Reserved. These bits are reserved and should read 0.
4–0 CLKO_SEL	CLKO Select. Selects which clock signal source is the output of CLKO pin. 00000 CLK32 00001 PREMCLK 00010 CLK26M 00011 MPLL Reference CLK 00100 SPLL Reference CLK 00101 HCLK Source (MPLL 2x clock output / 3) 00110 SPLL CLK 00111 FCLK 01000 HCLK 01001 IPG_CLK 01010 PERCLK1 01011 PERCLK2 01100 PERCLK3 01101 PERCLK4 01110 SSI 1 Baud 01111 SSI 2 Baud 10000 NFC Baud 10001 MSHC_Baud 10010 H264 Baud 10011 CLK60M Always 10100 CLK32K Always 10101 CLK60M 10110 DPTC Reference Clock

3.4.14 Wakeup Guard Mode Control Register (WKGDCTL)

The Wakeup Guard mode Control Register (WKGDCTL) provides the configuration of the Wakeup Guard mode. This is a write once only bit in order to be compatible with the watchdog behavior. After enable/disable, it will not be modifiable. When enabled, the battery detector external to the chip provides a glitch free signal through the TIN pin. Battery must be intact for the chip to wakeup from sleep.

Figure 3-15 shows the register and Table 3-20 provides the field descriptions.

0x1002_7034 (WKGDCTL) Access: User write-once

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

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

Figure 3-15. Wakeup Guard Mode Control Register (WKGDCTL)

Table 3-20. Wakeup Guard Mode Control Register Field Descriptions

Field	Description
31–1	Reserved. These bits are reserved and should read 0.
0 WKDG_EN	Wakeup Guard Mode Enable. Enables /disables the wakeup guard logic. Write- once-only bit and can only be cleared through system reset. Once enabled, battery indicator through TIN will be used to qualify the wakeup process. When battery is intact, that is, TIN=1, wakeup from sleep proceed as per normal. When WKGD_EN=1 and battery is removed, 32 kHz clock to watchdog module is gated off. Clock resumes when battery is back in place. 0 Wakeup Guard mode is disabled. 1 Wakeup Guard mode is enabled.

3.5 Functional Description of the Reset Module

The reset module controls or distributes all of the system reset signals used by the i.MX27 processor. A simplified block diagram of the reset module is shown in Figure 3-16. The reset module generates two distinct events—a global reset and an ARM9 Platform reset.

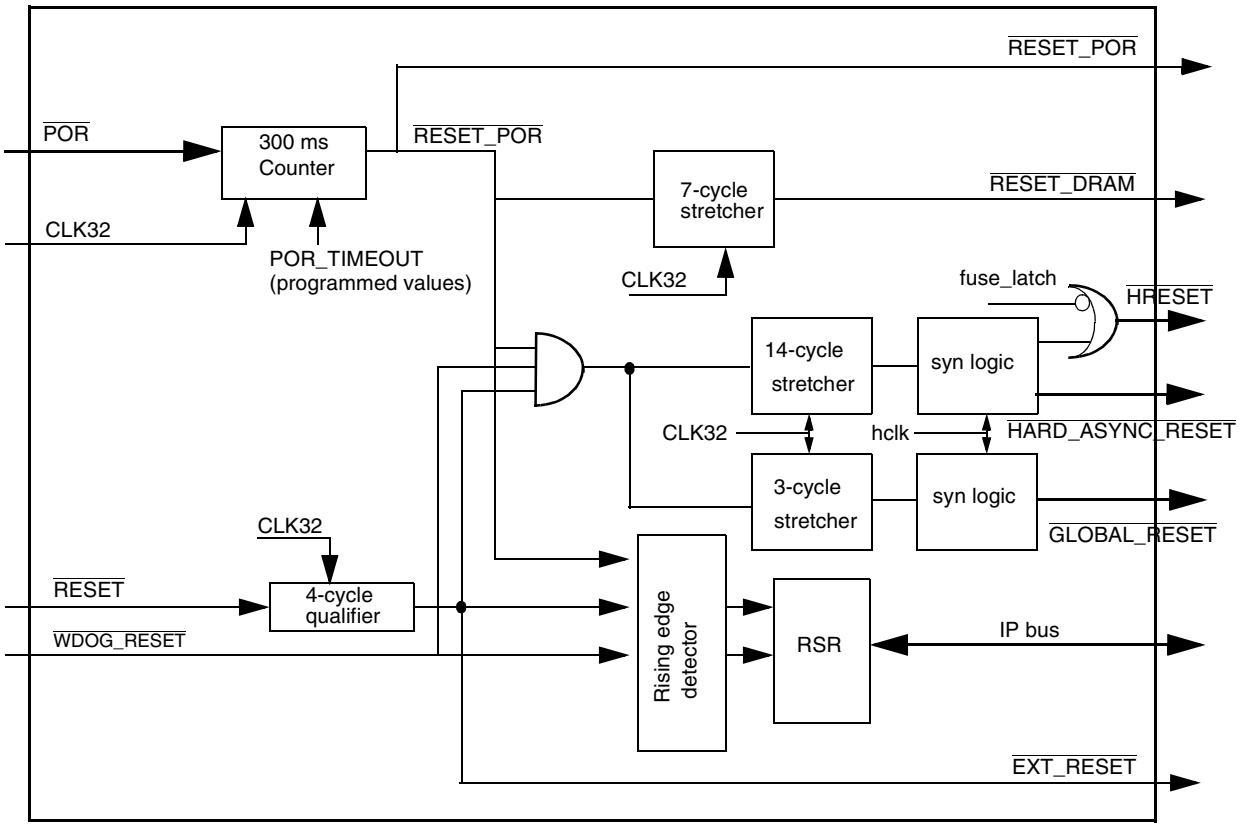


Figure 3-16. Reset Module Clock Diagram

3.5.1 Global Reset

A global reset is produced by the simultaneous assertion of the following resets:

- $\overline{\text{RESET\_DRAM}}$
- $\overline{\text{HRESET}}$
- $\overline{\text{HARD\_ASYNC\_RESET}}$
- $\overline{\text{RESET\_POR}}$

There is one source capable of generating a global reset: A low condition on the $\overline{\text{POR}}$ pin when the 32 kHz crystal oscillator is running.

The $\overline{\text{HRESET}}$ and $\overline{\text{HARD\_ASYNC\_RESET}}$ are armed simultaneously; they remain in that state for 14 CLK32 cycles.

$\overline{\text{RESET\_DRAM}}$ is deasserted seven CLK32 cycles before $\overline{\text{HRESET}}$ and $\overline{\text{HARD\_ASYNC\_RESET}}$ are deasserted. The SDRAM executes the necessary self refresh operations during this time.

The timing diagram in Figure 3-16 shows the relationship of the reset signal timings. See Table 3-21 for reset module signal and pin definitions.

The following signal conditions are not capable of generating a global reset, however their assertion will reset the ARM9 Platform:

- An external qualified low condition on the $\overline{\text{RESET\_IN}}$ pin
- A low condition on $\overline{\text{WDOG\_RESET}}$

Furthermore, these reset conditions will not reset the SDRAMC, Real Time Clock, WatchDog module, or allow a change in Boot mode—that is, changes made to BOOT[3:0] during these resets conditions will not take effect. Only the global reset is capable of this.

The source of the last hardware reset can be determined in the watchdog status register.

NOTE

Due to the asynchronous nature of the $\overline{\text{RESET\_IN}}$ signal, the time period required to qualify the signal may vary, and the $\overline{\text{HRESET}}$ timing relative to the rising edge of the $\overline{\text{RESET\_IN}}$ is also affected. A $\overline{\text{RESET\_IN}}$ signal shorter than three CLK32 cycles will not be qualified, a $\overline{\text{RESET\_IN}}$ signal equal to or longer than four CLK32 cycles will always be qualified, and any period length that is more than three and less than four CLK32 cycles is undefined.

$\overline{\text{POR}}$ is the reset signal for all the reset module flip-flops. For this reason, an external reset signal is qualified if it lasts more than four CLK32 cycles when $\overline{\text{POR}}$ is deasserted.

During power on the user must ensure that $\overline{\text{POR}}$ stay asserted (low) long enough for the 32kHz crystal to stabilize. The time it takes for the crystal to stabilize depends upon on the crystal used. Consult the crystal's specification for details about its stabilization timing.

NOTE

Refer to the *i.MX27 Multimedia Applications Processor Data Sheet* for power-up and power-down sequence requirements.

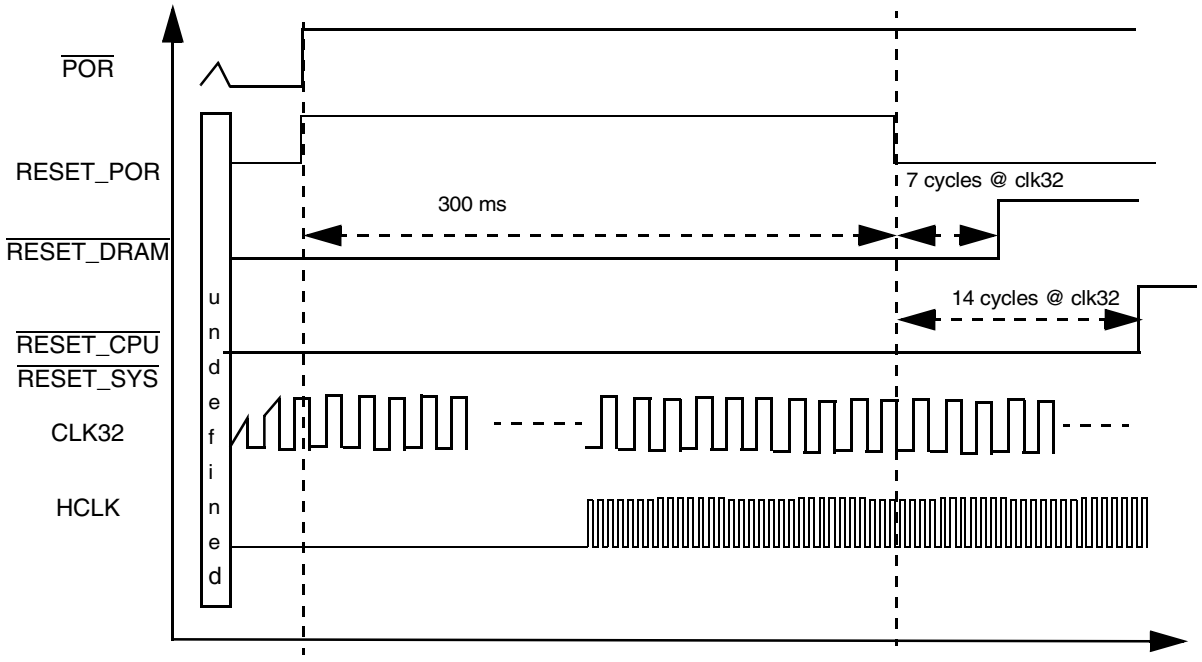


Figure 3-17. DRAM and Internal Reset Timing Diagram

3.5.2 ARM9 Platform Reset

Any qualified global reset signal resets the ARM9 Platform and all related peripherals to their default state. After the internal reset is deasserted, the ARM9 processor begins fetching code from the internal bootstrap ROM or CS0 space. The memory location of the fetch depends on the configuration of the BOOT pins and the value of the TEST pin on the rising edge of the HRESET.

Table 3-21. Reset Module Pin and Signal Descriptions

Signal Name	Direction	Signal Description
CLK32	IN	32 kHz Clock—A 32 kHz clock signal derived from the 32.768 KHz or 32.0 KHz crystal oscillator circuit in the PLL Clock Controller.
POR	IN	Power-On Reset—An internal active Schmitt trigger signal from the POR pin. The POR signal is normally generated by an external RC circuit designed to detect a power-up event.
RESET_IN	IN	Reset—An external active low Schmitt trigger signal from the RESET_IN pin. When this signal goes active, all modules (except the SDRAMC, Real Time Clock, WatchDog, and the BOOT[3:0] signals) are reset.
WDOG_RESET	IN	Watchdog Timer Reset—An active low signal generated by the watchdog timer when a time-out period has expired. Resets the same modules as RESET_IN.
HARD_ASYN_RESET	OUT	Hard Asynchronous Reset—An active low signal that resets all peripheral modules except the watchdog module's status register. The rising edge of this signal is synchronous with IPG_CLK.

Table 3-21. Reset Module Pin and Signal Descriptions (continued)

Signal Name	Direction	Signal Description
$\overline{\text{HRESET}}$	OUT	Hard Reset—An active low signal that resets the ARM9 Platform. This signal is deasserted during the low phase of HCLK. This signal also appears on the $\overline{\text{RESET\_OUT}}$ pin of the i.MX27.
$\overline{\text{RESET\_DRAM}}$	OUT	DRAM Reset—An active low signal that resets the SDRAM controller.

Chapter 4
System Control

4.1 Introduction

This chapter describes the system control module of the i.MX27 microprocessor. The system control module enables system software to control, customize, or read the status of the following functions:

- Chip ID
- Multiplexing of I/O signals
- I/O Drive Strength
- I/O Pull Enable Control
- Well Bias Control
- System boot mode selection
- DPTC Control

4.2 Memory Map and Register Definition

The system control module includes one 32-bit Silicon ID and twenty-four user-accessible 32-bit registers. Table 4-1 summarizes these registers and their addresses.

Table 4-1. Block Memory Map

Address	Register	Access	Reset Value	Section/Page
General Registers				
0x1002_7800 (CID)	Chip ID Register	R/W	0x1882_181D	4.2.1/4-3
0x1002_7814 (FMCR)	Function Multiplexing Control Register	R/W	0xFFFF_FFCB	4.2.2/4-4
0x1002_7818 (GPCR)	Global Peripheral Control Register	R/W	0x0000_0808	4.2.3/4-6
0x1002_781C (WBCR)	Well Bias Control Register	R/W	0x0000_0000	4.2.5/4-8
0x1002_7820 (DSCR1)	Drive Strength Control Register 1	R/W	0x0000_0000	4.2.6/4-10
0x1002_7824 (DSCR2)	Drive Strength Control Register 2	R/W	0x0000_0000	4.2.7/4-12
0x1002_7828 (DSCR3)	Drive Strength Control Register 3	R/W	0x0000_0000	4.2.8/4-14
0x1002_782C (DSCR4)	Drive Strength Control Register 4	R/W	0x0000_0000	4.2.9/4-17

Table 4-1. Block Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x1002_7830 (DSCR5)	Drive Strength Control Register 5	R/W	0x0000_0000	4.2.10/4-18
0x1002_7834 (DSCR6)	Drive Strength Control Register 6	R/W	0x0000_0000	4.2.11/4-21
0x1002_7838 (DSCR7)	Drive Strength Control Register 7	R/W	0x0000_0000	4.2.12/4-23
0x1002_783C (DSCR8)	Drive Strength Control Register 8	R/W	0x0000_0000	4.2.13/4-25
0x1002_7840 (DSCR9)	Drive Strength Control Register 9	R/W	0x0000_0000	4.2.14/4-28
0x1002_7844 (DSCR10)	Drive Strength Control Register 10	R/W	0x0000_0000	4.2.15/4-30
0x1002_7848 (DSCR11)	Drive Strength Control Register 11	R/W	0x0000_0000	4.2.16/4-32
0x1002_784C (DSCR12)	Drive Strength Control Register 12	R/W	0x0000_0000	4.2.17/4-34
0x1002_7850 (DSCR13)	Drive Strength Control Register 13	R/W	0x0000_0000	4.2.18/4-36
0x1002_7854 (PSCR)	Pull Strength Control Register	R/W	0x0000_0000	4.2.19/4-38
0x1002_7858 (PCSR)	Priority Control and Select Register	R/W	0x0000_0003	4.2.20/4-40
0x1002_7860 (PMCR)	Power Management Control Register	R/W	0x0000_0000	4.2.21/4-41
0x1002_7864 (DCVR0)	DPTC Comparator Value Register 0	R/W	0x0000_0000	4.2.22/4-43
0x1002_7868 (DCVR1)	DPTC Comparator Value Register 1	R/W	0x0000_0000	4.2.23/4-43
0x1002_786C (DCVR2)	DPTC Comparator Value Register 2	R/W	0x0000_0000	4.2.24/4-44
0x1002_7870 (DCVR3)	DPTC Comparator Value Register 3	R/W	0x0000_0000	4.2.25/4-45

The conventions in Figure 4-1 and Table 4-2 serve as a key for the register summary and individual register diagrams.

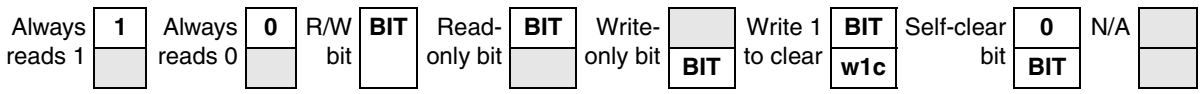


Figure 4-1. Key to Register Fields

Table 4-2 provides a key for register figures and tables and the register summary.

Table 4-2. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

4.2.1 Chip ID Register (CID)

The Chip ID register contains the chip identification number. Figure 4-2 shows the register and Table 4-3 provides its field descriptions.

0x1002_7800 (CID) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	VERSION ID				PART NUMBER											
Reset	0	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PART NUMBER				MANUFACTURER ID											
Reset	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	1

Figure 4-2. Chip ID Register (CID)

Table 4-3. Chip ID Register Field Descriptions

Field	Description
31–28 VERSION ID	VERSION ID. This field contains the 4-bit version ID number.
27–12 PART NUMBER	PART NUMBER. This field contains the 16-bit part number of the chip.
11–0 MANUFACTURER ID	MANUFACTURER ID. This field contains the 12-bit manufacturer ID number of the chip.

4.2.2 Function Multiplexing Control Register (FMCR)

The FMCR controls the multiplexing of the signal lines shared by the SLCDC module, UART module, and Keypad module as well as the SDRAM chip select lines. The FMCR also allows control or indicates the boot status of the NAND Flash page size and data port size. Figure 4-3 shows the register and Table 4-4 provides its field descriptions.

0x1002_7814 (FMCR) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	1	1	1	1	1	1	UART4_RXD_CTL	UART4_RTS_CTL	1	1	1	1	1	KP_COL6_CTL	KP_ROW7_CTL	KP_ROW6_CTL
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	1	PC_WAIT_B_CTL	PC_READY_CTL	PC_VS1_CTL	PC_VS2_CTL	PC_BVD1_CTL	PC_BVD2_CTL	IOIS16_CTL	1	1	NF_FMS	NF_16BIT_SEL	1	SLCDC_SEL	SDCS1_SEL	SDCS0_SEL
W																
Reset	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	1

Figure 4-3. Function Multiplexing Control Register (FMCR)

Table 4-4. Function Multiplexing Control Register Description

Field	Description
31–26	Reserved. These bits are reserved and should read 1.
25 UART4_RXD_CTL	UART4 RXD Control. When set, the alternate signal of USBH1_RXDP (PB31) is input to RXD of UART4. When 0, the USBH1_TXDP (PB29) GPIO's AOUT is input to RXD of UART4. With either setting, the user must also ensure that the proper GPIO registers have been programmed to select the desired multiplexing. 0 The USBH1_TXDP (PB29) GPIO's AOUT is input to RXD of UART4. 1 The alternate signal of USBH1_RXDP (PB31) is input to RXD of UART4.

Table 4-4. Function Multiplexing Control Register Description (continued)

Field	Description
24 UART4_RTS_CTL	UART4 RTS Control. When set, the alternate signal of USBH1_FS (PB26) is input to RTS of UART4. When 0, the USBH1_RXDP (PB31) GPIO's AOUT is input to RTS of UART4. With either setting, the user must also ensure that the proper GPIO registers have been programmed to select the desired multiplexing. 0 The USBH1_RXDP (PB31) GPIO's AOUT is input to RTS of UART4. 1 The alternate signal of USBH1_FS (PB26) is input to RTS of UART4.
23–19	Reserved. These bits are reserved and should read 1.
18 KP_COL6_CTL	Keypad Column 6 Control. When set, the alternate signal of UART2_TXD (PE6) is input to column 6 of keypad. When 0, the alternate signal of TEST_WB2 (PE0) is input to column 6 of keypad. With either setting, the user must also ensure that the proper GPIO registers have been programmed to select the desired multiplexing. 0 The alternate signal of TEST_WB2 (PE0) is input to column 6 of keypad. 1 The alternate signal of UART2_TXD (PE6) is input to column 6 of keypad.
17 KP_ROW7_CTL	Keypad Row 7 Control. When set, the alternate signal of UART2_RTS (PE4) is input to row 7of keypad. When 0, the alternate signal of TEST_WB0 (PE2) is input to row 7 of keypad. With either setting, the user must also ensure that the proper GPIO registers have been programmed to select the desired multiplexing. 0 The alternate signal of TEST_WB0 (PE2) is input to row 7 of keypad. 1 The alternate signal of UART2_RTS (PE4) is input to row 7of keypad.
16 KP_ROW6_CTL	Keypad Row 6 Control. When set, the alternate signal of UART2_RXD (PE7) is input to row 6 of keypad. When 0, the alternate signal of TEST_WB1 (PE1) is input to row 6 of keypad. With either setting, the user must also ensure that the proper GPIO registers have been programmed to select the desired multiplexing. 0 The alternate signal of TEST_WB1 (PE1) is input to row 6 of keypad. 1 The alternate signal of UART2_RXD (PE7) is input to row 6 of keypad.
15	Reserved. These bits are reserved and should read 1.
14 PC_WAIT_B_CTL	PC_WAIT_B Control. When set, signal pc_wait_b of PCMCIA is input from PC_WAIT_B. When 0, it is input from BOUT of GPIO PORT C[31]. 0 The signal pc_wait_b of PCMCIA is input from BOUT of GPIO PORT C[31]. 1 The signal pc_wait_b of PCMCIA is input from PC_WAIT_B.
13 PC_READY_CTL	PC_READY Control. When set, signal pc_ready of PCMCIA is input from PC_READY. When 0, it is input from BOUT of GPIO PORT C[30]. 0 The signal pc_ready of PCMCIA is input from BOUT of GPIO PORT C[30]. 1 The signal pc_ready of PCMCIA is input from PC_READY.
12 PC_VS1_CTL	PC_VS1 Control. When set, signal pc_vs1 of PCMCIA is input from PC_VS1. When 0, it is input from BOUT of GPIO PORT C[29]. 0 The signal pc_vs1 of PCMCIA is input from BOUT of GPIO PORT C[29]. 1 The signal pc_vs1 of PCMCIA is input from PC_VS1.
11 PC_VS2_CTL	PC_VS2 Control. When set, signal pc_vs2 of PCMCIA is input from PC_VS2. When 0, it is input from BOUT of GPIO PORT C[28]. 0 The signal pc_vs2 of PCMCIA is input from BOUT of GPIO PORT C[28]. 1 The signal pc_vs2 of PCMCIA is input from PC_VS2.
10 PC_BVD1_CTL	PC_BVD1 Control. When set, signal pc_bvd1 of PCMCIA is input from PC_BVD1. When 0, it is input from BOUT of GPIO PORT C[19]. 0 The signal pc_bvd1 of PCMCIA is input from BOUT of GPIO PORT C[19]. 1 The signal pc_bvd1 of PCMCIA is input from PC_BVD1.

Table 4-4. Function Multiplexing Control Register Description (continued)

Field	Description
9 PC_BVD2_CTL	PC_BVD2 Control. When set, signal PC_BVD2 of PCMCIA is input from PC_BVD2. When 0, it is input from BOUT of GPIO PORT C[18]. 0 The signal pc_bvd2 of PCMCIA is input from BOUT of GPIO PORT C[18]. 1 The signal pc_bvd2 of PCMCIA is input from PC_BVD2.
8 IOIS16_CTL	IOIS16 Control. When set, signal iois16 of PCMCIA is input from IOIS16. When 0, it is input from BOUT of GPIO PORT C[17]. 0 The signal iois16 of PCMCIA is input from BOUT of GPIO PORT C[17]. 1 The signal iois16 of PCMCIA is input from IOIS16.
7–6	Reserved. These bits are reserved and should read 1.
5 NF_FMS	Flash Memory Select. When Boot[3:0] = 0010 or 0011, the NF_FMS will be set, otherwise it will be 0. After boot up, this bit is user programmable. 0 NAND Flash with 512B page size (64Mb/128Mb/256Mb/512Mbyte/ 1Gbyte DDP) 1 NAND Flash with 2 Kbyte page size (1Gbyte/2Gbyte DDP/2Gbyte) **Note:** DDP means Double Density Package.
4 NF_16BIT_SEL	NAND Flash 16-bit Select. Selects 16-bit NF operation. Setting this bit forces the NAND Flash into 16-bit operation and the NAND Flash upper data is available to the pins. Clearing this bit forces the NF to 8-bit operation and the A[25:21]signals become the function pins. The muxing is done in the EMI module, not I/O MUX module. During system boot up, if the BOOT[3:0] input pins are configured to select 16-bit mode, this NF_16BIT_SEL bit is set. 0 NAND Flash 8-bit operation 1 NAND Flash 16-bit operation
3	Reserved. This bit is reserved and should read 0.
2 SLCDC_SEL	SLCDC Select. Selects whether a BaseBand chip (BB) or the i.MX27 processor drives the SLCDC display port in serial mode. 0 On Chip SLCDC drives the SLCDC port. 1 BB can write directly to the SLCDC port.
1 SDCS1_SEL	SDRAM Chip Select. Selects the function of the $\overline{\text{CS3/CSD1}}$ pin. 0 $\overline{\text{CS3}}$ is selected. 1 $\overline{\text{CSD1}}$ is selected.
0 SDCS0_SEL	SDRAM Chip Select. Selects the function of the $\overline{\text{CS2/CSD0}}$ pin. 0 $\overline{\text{CS2}}$ is selected. 1 $\overline{\text{CSD0}}$ is selected.

4.2.3 Global Peripheral Control Register (GPCR)

The Global Peripheral Control Register (GPCR) displays the current boot mode of the i.MX27 device. The clock gating to the processor's modules is also controlled by this register. Figure 4-4 shows the register and Table 4-5 provides its field descriptions.

0x1002_7818 (GPCR) Access: User R/W

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ETM9_PAD_EN	USB_Burst_Override	PP_Burst_Override	DMA_Burst_Override	0	0	0	0	CLOCK_GATING_EN	DDR_MODE	CLK_DDR_MODE	DDR_INPUT
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

Figure 4-4. Global Peripheral Control Register (GPCR)

Table 4-5. Global Peripheral Control Register Descriptions

Field	Description
31–20	Reserved. These bits are reserved and should read 0.
19–16 BOOT	Boot Mode. These are 4-bit system boot mode for the i.MX27 device. 0000 Bootstrap from UART/USB 0001 Reserved 0010 8-bit NAND Flash (2 Kbyte per page) 0011 16-bit Nand Flash (2 Kbyte per page) 0100 16-bit Nand Flash (512 bytes per page) 0101 16-bit CS0 0110 32-bit CS0 0111 8 bit Nand Flash (512 bytes per page) 1xxx Reserved
15–12	Reserved. These bits are reserved and should read 0.
11 ETM9_PAD_EN	ETM9 Pad Enable. When this bit is set, pads for ETM9 are enabled. 0 Disable ETM9 pads 1 Enable ETM9 pads
10 USB_Burst_Override	USB Burst Override Control. When this bit is set, the burst type of USB will be forced to INCR8. 0 Bypass. The burst type will not be forced. 1 Burst type of USB is INCR8
9 PP_Burst_Override	EMMA PP Burst Override Control. When this bit is set, the burst type of EMMA PP will be forced to INCR4 or INCR8. 0 Bypass. The burst type will not be forced. 1 Burst type of EMMA PP is INCR4 or INCR8.
8 DMA_Burst_Override	DMA Burst Override Control. When this bit is set, the burst type of DMA will be forced to INCR4 or INCR8. 0 Bypass. The burst type will not be forced. 1 Burst type of DMA is INCR4 or INCR8.
7–4	Reserved. These bits are reserved and should read 0.

Table 4-5. Global Peripheral Control Register Descriptions (continued)

Field	Description
3 CLOCK_GATING_EN	Clock Gating Enable. When set to 1, the peripheral register access clocks are gated by the AIPI modules. For example, when there is a register read or write access to the peripherals of AIPI1, the ipg_clk_s1 clock will be running, otherwise if no access is taking place the clock will shut off and when there is a register read or write access to the peripherals of AIPI2. The clock is running, otherwise if no access is taking place the clock shuts off. When this bit is cleared to 0 then the AIPI clocks become a continuous clock, regardless of peripheral accesses. It is recommended for maximum power savings to ensure this bit is set to 1.
2 DDR_MODE	DDR Drive Strength Control. used to select DDR drive strength of all DDR pads except the SDCLK pad. 0 Drive strength is selected by associated fields in the DSCRx registers. 1 Drive strength of about 20 mA, as defined in SSTL_18
1 CLK_DDR_MODE	CLK DDR MODE. used to select DDR drive strength of SDCLK pad. 0 Drive strength is selected by associated fields in the DSCR8 registers. 1 Drive strength of about 20 mA, as defined in SSTL_18
0 DDR_INPUT	DDR_INPUT. Used to force input mode of DDR pads to CMOS input mode. 0 No force on input mode of DDR pads 1 DDR pads will be forced to CMOS input mode.

4.2.4 Well Bias System

The i.MX27 processor employs an innovative system feature to help reduce leakage current called Well Biasing. The Well Bias System reduces the leakage current of the QVDDx sub-system during low-power mode by increasing the threshold voltage of the QVDDx sub-system transistors. The i.MX27 contains two Well Biasing System, one for ARM core logic and one for EMI module. The following section describes how to enable and take advantage of this power saving feature.

4.2.5 Well Bias Control Register (WBCR)

The Well Bias Control Register (WBCR) allows the user to enable the A926P Well Biasing System and EMI Well Biasing System. The default setting is both Well Biasing Systems are disabled. A926P Well Biasing System can operate under both Doze mode and Sleep mode, while EMI Well Biasing System can operate under Sleep mode only. To enable Well Biasing Systems and take advantage of this power saving feature, CRM_WBFA or CRM_WBFA_EMI bit must be set to 1.

Figure 4-5 shows the register and Table 4-6 provides its field descriptions.

0x1002_781C (WBCR) Access: User R/W

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	CRM_SPA				0	0	0	0	CRM_WBFA		CRM_WBM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-5. Well Bias Control Register (WBCR)

Table 4-6. Well Bias Control Register Field Descriptions

Field	Description
31–28	Reserved. These bits are reserved and should read 0.
27–26 CRM_SPA_EMI	EMI PWELL Set Point Adjust. Describe the configuration of the EMI PWELL bias circuit's set point or regulation level. 00 Minimum Back Bias applied to the Pwells. 01 Decreased Back Bias applied to the Pwells. 10 Moderate Back Bias applied to the Pwells. 11 Increased Back Bias applied to the Pwells.
25–24 CRM_SPA_EMI	EMI NWELL Set Point Adjust. Describe the configuration of the EMI NWELL bias circuit's set point or regulation level. 00 Minimum Back Bias applied to the Nwells. 01 Decreased Back Bias applied to the Nwells. 10 Moderate Back Bias applied to the Nwells. 11 Increased Back Bias applied to the Nwells.
23–20	Reserved. These bits are reserved and should read 0.
19 CRM_WBFA_EMI	Well Bias Frequency Adjust. For optimal power savings, the user should set this bit to 1 when EMI Well Bias is enabled. 0 Standard 1 Adjusted Suggested setting for optimal power savings when Well Bias is enabled. **Note:** This bit has no effect when Well Bias is disabled.
18–16 CRM_WBM_EMI	CRM_WBM. Enables or disables EMI Well Bias System during Sleep mode. To enable Well Bias during Sleep mode, these bits must be set to 001. To disable Well Bias, these bits must be set to 000. All other bit settings are reserved. 000 Well Bias not applied 001 Well Bias @ Sleep 010–111Well Bias not applied
15–12	Reserved. These bits are reserved and should read 0.
11–10 CRM_SPA	A926P PWELL Set Point Adjust. Describes the configuration of the A926P PWELL bias circuit's set point or regulation level. 00 Minimum Back Bias applied to the Pwells. 01 Decreased Back Bias applied to the Pwells. 10 Moderate Back Bias applied to the Pwells. 11 Increased Back Bias applied to the Pwells.

Table 4-6. Well Bias Control Register Field Descriptions (continued)

Field	Description
9–8 CRM_SPA	A926P NWELL Set Point Adjust. Describe the configuration of the A926P Nwell bias circuit's set point or regulation level. 00 Minimum Back Bias applied to the Nwells. 01 Decreased Back Bias applied to the Nwells. 10 Moderate Back Bias applied to the Nwells. 11 Increased Back Bias applied to the Nwells.
7–4	Reserved. These bits are reserved and should read 0.
3 CRM_WBFA	Well Bias Frequency Adjust. For optimal power savings, the user should set this bit to 1 when A926P Well Bias is enabled. 0 Standard. 1 Adjusted Suggested setting for optimal power savings when Well Bias is enabled This bit has no effect when Well Bias is disabled.
2–0 CRM_WBM	CRM_WBM. Controls when the A926P well bias will be applied.Enables or disables Well Bias System during Sleep mode. To enable Well Bias during Sleep mode, these bits must be set to 001. To disable Well Bias, these bits must be set to 000. All other bit settings are reserved. 000 Well Bias not applied 001 Well Bias @ Sleep 010 Well Bias @ Sleep and DOZE 100–111Well Bias not applied

4.2.6 Drive Strength Control Register 1 (DSCR1)

The Drive Strength Control Register 1 (DSCR1) controls the driving force parameters of all slow I/O signals in the i.MX27 device. Figure 4-6 shows the register and Table 4-7 provides its field descriptions.

0x1002_7820 (DSCR1) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	DS_SLOW11		DS_SLOW10		DS_SLOW9	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DS_SLOW8		DS_SLOW7		DS_SLOW6		DS_SLOW5		DS_SLOW4		DS_SLOW3		DS_SLOW2		DS_SLOW1	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-6. Drive Strength Control Register (DSCR1)

Table 4-7. Drive Strength Control Register 1 Field Description

Field	Description
31–22	Reserved. These bits are reserved and should read 0.
21–20 DS_SLOW11	Drive Strength Slow I/O. Controls the driving strength of slow I/O group 11. (DVS_PMIC) 00 Normal 01 High 10 Max high 11 Max high
19–18 DS_SLOW10	Drive Strength Slow I/O. Controls the driving strength of slow I/O group 10. (SDHC1 and CSPI3) 00 Normal 01 High 10 Max high 11 Max high
17–16 DS_SLOW9	Drive Strength Slow I/O. Controls the driving strength of slow I/O group 9. (JTAG) 00 Normal 01 High 10 Max high 11 Max high
15–14 DS_SLOW8	Drive Strength Slow I/O. Controls the driving strength of slow I/O group 8. (PWM, KPP, UART1, UART2, UART3, and RESET_OUT_B) 00 Normal 01 High 10 Max high 11 Max high
13–12 DS_SLOW7	Drive Strength Slow I/O. Controls the driving strength of slow I/O group 7. (CSPI1 and CSPI2) 00 Normal 01 High 10 Max high 11 Max high
11–10 DS_SLOW6	Drive Strength Slow I/O. Controls the driving strength of slow I/O group 6. (SSI1, SSI2, SAP, SSI3, GPT4, and GPT5) 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_SLOW5	Drive Strength Slow I/O. Controls the driving strength of slow I/O group 5. (GPT1, I2C1, and I2C2) 00 Normal 01 High 10 Max high 11 Max high
7–6 DS_SLOW4	Drive Strength Slow I/O. Controls the driving strength of slow I/O group 4. (USBH1, UART4, and USBG) 00 Normal 01 High 10 Max high 11 Max high

Table 4-7. Drive Strength Control Register 1 Field Description (continued)

Field	Description
5–4 DS_SLOW3	Drive Strength Slow I/O. Controls the driving strength of slow I/O group 3. (CSI, UART5, and UART6) 00 Normal 01 High 10 Max high 11 Max high
3–2 DS_SLOW2	Drive Strength Slow I/O. Controls the driving strength of slow I/O group 2.(SDHC2 and MSHC) 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_SLOW1	Drive Strength Slow I/O. Controls the driving strength of slow I/O group 1. (LCDC) 00 Normal 01 High 10 Max high 11 Max high

4.2.7 Drive Strength Control Register 2 (DSCR2)

The Drive Strength Control Register 2 (DSCR2) controls the driving force parameters of the fast I/O signals in the i.MX27 processor. Figure 4-7 shows the register and Table 4-8 provides its field descriptions.

0x1002_7824 (DSCR2) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DS_FAST16		DS_FAST15		DS_FAST14		DS_FAST13		DS_FAST12		DS_FAST11		DS_FAST10		DS_FAST9	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DS_FAST8		DS_FAST7		DS_FAST6		DS_FAST5		DS_FAST4		DS_FAST3		DS_FAST2		DS_FAST1	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-7. Drive Strength Control Register 2 (DSCR2)

Table 4-8. Drive Strength Control Register 2 Field Descriptions

Field	Description
31–30 DS_FAST16	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 16 (D15). 00 Normal 01 High 10 Max high 11 Max high
29–28 DS_FAST15	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 15 (D14). 00 Normal 01 High 10 Max high 11 Max high
27–26 DS_FAST14	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 14 (D13). 00 Normal 01 High 10 Max high 11 Max high
25–24 DS_FAST13	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 13 (D12). 00 Normal 01 High 10 Max high 11 Max high
23–22 DS_FAST12	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 12 (D11). 00 Normal 01 High 10 Max high 11 Max high
21–20 DS_FAST11	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 11(D10). 00 Normal 01 High 10 Max high 11 Max high
19–18 DS_FAST10	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 10 (D9). 00 Normal 01 High 10 Max high 11 Max high
17–16 DS_FAST9	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 9 (D8). 00 Normal 01 High 10 Max high 11 Max high
15–14 DS_FAST8	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 8 (D7). 00 Normal 01 High 10 Max high 11 Max high

Table 4-8. Drive Strength Control Register 2 Field Descriptions (continued)

Field	Description
13–12 DS_FAST7	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 7 (D6). 00 Normal 01 High 10 Max high 11 Max high
11–10 DS_FAST6	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 6 (D5). 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_FAST5	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 5 (D4). 00 Normal 01 High 10 Max high 11 Max high
7–6 DS_FAST4	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 4 (D3). 00 Normal 01 High 10 Max high 11 Max high
5–4 DS_FAST3	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 3 (D2). 00 Normal 01 High 10 Max high 11 Max high
3–2 DS_FAST2	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 2 (D1). 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_FAST1	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 1 (D0). 00 Normal 01 High 10 Max high 11 Max high

4.2.8 Drive Strength Control Register 3

The Drive Strength Control Register 3 (DSCR3) controls the driving force parameters of the fast I/O signals in the i.MX27. Figure 4-8 shows the register and Table 4-9 provides its field descriptions.

0x1002_7828 (DSCR3) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DS_FAST32		DS_FAST31		DS_FAST30		DS_FAST29		DS_FAST28		DS_FAST27		DS_FAST26		DS_FAST25	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DS_FAST24		DS_FAST23		DS_FAST22		DS_FAST21		DS_FAST20		DS_FAST19		DS_FAST18		DS_FAST17	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-8. Drive Strength Control Register 3 (DSCR3)

Table 4-9. Drive Strength Control Register 3 Field Descriptions

Field	Description
31–30 DS_FAST32	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 32 (A15). 00 Normal 01 High 10 Max high 11 Max high
29–28 DS_FAST31	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 31 (A14). 00 Normal 01 High 10 Max high 11 Max high
27–26 DS_FAST30	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 30 (A13). 00 Normal 01 High 10 Max high 11 Max high
25–24 DS_FAST29	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 29 (A12). 00 Normal 01 High 10 Max high 11 Max high
23–22 DS_FAST28	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 28 (A11). 00 Normal 01 High 10 Max high 11 Max high
21–20 DS_FAST27	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 27 (A10). 00 Normal 01 High 10 Max high 11 Max high

Table 4-9. Drive Strength Control Register 3 Field Descriptions (continued)

Field	Description
19–18 DS_FAST26	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 26 (A9). 00 Normal 01 High 10 Max high 11 Max high
17–16 DS_FAST25	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 25 (A8). 00 Normal 01 High 10 Max high 11 Max high
15–14 DS_FAST24	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 24 (A7). 00 Normal 01 High 10 Max high 11 Max high
13–12 DS_FAST23	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 23 (A6). 00 Normal 01 High 10 Max high 11 Max high
11–10 DS_FAST22	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 22 (A5). 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_FAST21	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 21 (A4). 00 Normal 01 High 10 Max high 11 Max high
7–6 DS_FAST20	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 20 (A3). 00 Normal 01 High 10 Max high 11 Max high
5–4 DS_FAST19	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 19 (A2). 00 Normal 01 High 10 Max high 11 Max high

Table 4-9. Drive Strength Control Register 3 Field Descriptions (continued)

Field	Description
3–2 DS_FAST18	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 18 (A1). 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_FAST17	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 17 (A0). 00 Normal 01 High 10 Max high 11 Max high

4.2.9 Drive Strength Control Register 4

The Drive Strength Control Register 4 (DSCR4) controls the driving force parameters of the fast I/O signals in the i.MX27 device. Figure 4-9 shows the register and Table 4-10 provides its field descriptions.

0x1002_782C (DSCR4) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	DS_FAST42		DS_FAST41	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DS_FAST40		DS_FAST39		DS_FAST38		DS_FAST37		DS_FAST36		DS_FAST35		DS_FAST34		DS_FAST33	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-9. Drive Strength Control Register 4 (DSCR4)

Table 4-10. Drive Strength Control Register 4 Field Descriptions

Field	Description
31–20	Reserved. These bits are reserved and should read 0.
19–18 DS_FAST42	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 42 (A25). 00 Normal 01 High 10 Max high 11 Max high
17–16 DS_FAST41	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 41 (A24). 00 Normal 01 High 10 Max high 11 Max high

Table 4-10. Drive Strength Control Register 4 Field Descriptions (continued)

Field	Description
15–14 DS_FAST40	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 40 (A23). 00 Normal 01 High 10 Max high 11 Max high
13–12 DS_FAST39	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 39 (A22). 00 Normal 01 High 10 Max high 11 Max high
11–10 DS_FAST38	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 38 (A21). 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_FAST37	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 37 (A20). 00 Normal 01 High 10 Max high 11 Max high
7–6 DS_FAST36	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 36 (A19). 00 Normal 01 High 10 Max high 11 Max high
5–4 DS_FAST35	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 35 (A18). 00 Normal 01 High 10 Max high 11 Max high
3–2 DS_FAST34	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 34 (A17). 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_FAST33	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 33 (A16). 00 Normal 01 High 10 Max high 11 Max high

4.2.10 Drive Strength Control Register 5

The Drive Strength Control Register 5 (DSCR5) controls the driving force parameters of the fast I/O signals in the i.MX27 processor. Figure 4-10 shows the register and Table 4-11 provides its field descriptions.

0x1002_7830 (DSCR5) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DS_FAST64		DS_FAST63		DS_FAST62		DS_FAST61		DS_FAST60		DS_FAST59		DS_FAST58		DS_FAST57	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DS_FAST56		DS_FAST55		DS_FAST54		DS_FAST53		DS_FAST52		DS_FAST51		DS_FAST50		DS_FAST49	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-10. Drive Strength Control Register 5 (DSCR5)

Table 4-11. Drive Strength Control Register 5 Field Descriptions

Field	Description
31–30 DS_FAST64	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 64 (SD15). 00 Normal 01 High 10 Max high 11 Max high
29–28 DS_FAST63	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 63 (SD14). 00 Normal 01 High 10 Max high 11 Max high
27–26 DS_FAST62	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 62 (SD13). 00 Normal 01 High 10 Max high 11 Max high
25–24 DS_FAST61	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 61 (SD12). 00 Normal 01 High 10 Max high 11 Max high
23–22 DS_FAST60	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 60 (SD11). 00 Normal 01 High 10 Max high 11 Max high
21–20 DS_FAST59	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 59 (SD10). 00 Normal 01 High 10 Max high 11 Max high

Table 4-11. Drive Strength Control Register 5 Field Descriptions (continued)

Field	Description
19–18 DS_FAST58	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 58 (SD9). 00 Normal 01 High 10 Max high 11 Max high
17–16 DS_FAST57	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 57 (SD8). 00 Normal 01 High 10 Max high 11 Max high
15–14 DS_FAST56	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 56 (SD7). 00 Normal 01 High 10 Max high 11 Max high
13–12 DS_FAST55	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 55 (SD6). 00 Normal 01 High 10 Max high 11 Max high
11–10 DS_FAST54	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 54 (SD5). 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_FAST53	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 53 (SD4). 00 Normal 01 High 10 Max high 11 Max high
7–6 DS_FAST52	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 52 (SD3). 00 Normal 01 High 10 Max high 11 Max high
5–4 DS_FAST51	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 51 (SD2). 00 Normal 01 High 10 Max high 11 Max high

Table 4-11. Drive Strength Control Register 5 Field Descriptions (continued)

Field	Description
3–2 DS_FAST50	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 50 (SD1). 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_FAST49	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 49 (SD0). 00 Normal 01 High 10 Max high 11 Max high

4.2.11 Drive Strength Control Register 6

The Drive Strength Control Register 6 (DSCR6) controls the driving force parameters of the fast I/O signals in the i.MX27 device. Figure 4-11 shows the register and Table 4-12 provides its field descriptions.

0x1002_7834 (DSCR6) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DS_FAST80		DS_FAST79		DS_FAST78		DS_FAST77		DS_FAST76		DS_FAST75		DS_FAST74		DS_FAST73	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DS_FAST72		DS_FAST71		DS_FAST70		DS_FAST69		DS_FAST68		DS_FAST67		DS_FAST66		DS_FAST65	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-11. Drive Strength Control Register 6 (DSCR6)

Table 4-12. Drive Strength Control Register 6 Field Descriptions

Field	Description
31–30 DS_FAST80	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 80 (SD31). 00 Normal 01 High 10 Max high 11 Max high
29–28 DS_FAST79	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 79 (SD30). 00 Normal 01 High 10 Max high 11 Max high
27–26 DS_FAST78	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 78 (SD29). 00 Normal 01 High 10 Max high 11 Max high

Table 4-12. Drive Strength Control Register 6 Field Descriptions (continued)

Field	Description
25–24 DS_FAST77	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 77 (SD28). 00 Normal 01 High 10 Max high 11 Max high
23–22 DS_FAST7	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 76 (SD27). 00 Normal 01 High 10 Max high 11 Max high
21–20 DS_FAST75	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 75 (SD26). 00 Normal 01 High 10 Max high 11 Max high
19–18 DS_FAST74	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 74 (SD25). 00 Normal 01 High 10 Max high 11 Max high
17–16 DS_FAST73	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 73 (SD24). 00 Normal 01 High 10 Max high 11 Max high
15–14 DS_FAST72	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 72 (SD23). 00 Normal 01 High 10 Max high 11 Max high
13–12 DS_FAST71	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 71 (SD22). 00 Normal 01 High 10 Max high 11 Max high
11–10 DS_FAST70	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 70 (SD21). 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_FAST69	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 69 (SD20). 00 Normal 01 High 10 Max high 11 Max high

Table 4-12. Drive Strength Control Register 6 Field Descriptions (continued)

Field	Description
7–6 DS_FAST68	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 68 (SD19). 00 Normal 01 High 10 Max high 11 Max high
5–4 DS_FAST67	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 67 (SD18). 00 Normal 01 High 10 Max high 11 Max high
3–2 DS_FAST66	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 66 (SD17). 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_FAST65	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 65 (SD16). 00 Normal 01 High 10 Max high 11 Max high

4.2.12 Drive Strength Control Register 7

The Drive Strength Control Register 7 (DSCR7) controls the driving force parameters of the fast I/O signals in the i.MX27 processor. Figure 4-12 shows the register and Table 4-13 provides its field descriptions.

0x1002_7838 (DSCR7) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DS_FAST95		DS_FAST94		DS_FAST93		DS_FAST92		DS_FAST91		DS_FAST90		DS_FAST89	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DS_FAST88		DS_FAST87		DS_FAST86		DS_FAST85		DS_FAST84		DS_FAST83		DS_FAST82		DS_FAST81	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-12. Drive Strength Control Register 7 (DSCR7)

Table 4-13. Drive Strength Control Register 7 Field Descriptions

Field	Description
31–30	Reserved. These bits are reserved and should read 0.
29–28 DS_FAST95	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 95 (RW_B). 00 Normal 01 High 10 Max high 11 Max high
27–26 DS_FAST94	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 94 (BCLK). 00 Normal 01 High 10 Max high 11 Max high
25–24 DS_FAST93	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 93 (LBA_B). 00 Normal 01 High 10 Max high 11 Max high
23–22 DS_FAST92	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 92 (OE_B). 00 Normal 01 High 10 Max high 11 Max high
21–20 DS_FAST91	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 91 (ECB_B). 00 Normal 01 High 10 Max high 11 Max high
19–18 DS_FAST90	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 90 (CS5_B). 00 Normal 01 High 10 Max high 11 Max high
17–16 DS_FAST89	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 89 (CS4_B). 00 Normal 01 High 10 Max high 11 Max high
15–14 DS_FAST88	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 88 (CS3_B). 00 Normal 01 High 10 Max high 11 Max high
13–12 DS_FAST87	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 87 (CS2_B). 00 Normal 01 High 10 Max high 11 Max high

Table 4-13. Drive Strength Control Register 7 Field Descriptions (continued)

Field	Description
11–10 DS_FAST86	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 86 (CS1_B). 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_FAST85	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 85 (CS0_B). 00 Normal 01 High 10 Max high 11 Max high
7–6 DS_FAST84	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 84 (EB1_B). 00 Normal 01 High 10 Max high 11 Max high
5–4 DS_FAST83	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 83 (EB0_B). 00 Normal 01 High 10 Max high 11 Max high
3–2 DS_FAST82	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 82 (SDBA1). 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_FAST81	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 81 (SDBA0). 00 Normal 01 High 10 Max high 11 Max high

4.2.13 Drive Strength Control Register 8

The Drive Strength Control Register 8 (DSCR8) controls the driving force parameters of the fast I/O signals in the i.MX27 device. Figure 4-13 shows the register and Table 4-14 provides its field descriptions.

0x1002_783C (DSCR8) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DS_FAST111		DS_FAST110		DS_FAST109		DS_FAST108		DS_FAST107		DS_FAST106		DS_FAST105	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DS_FAST104		DS_FAST103		DS_FAST102		DS_FAST101		DS_FAST100		DS_FAST99		DS_FAST98		DS_FAST97	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-13. Drive Strength Control Register 8 (DSCR8)

Table 4-14. Drive Strength Control Register 8 Field Descriptions

Field	Description
31–30	Reserved. These bits are reserved and should read 0.
29–28 DS_FAST111	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 111 (SDQS3). 00 Normal 01 High 10 Max high 11 Max high
27–26 DS_FAST110	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 110 (SDQS2). 00 Normal 01 High 10 Max high 11 Max high
25–24 DS_FAST109	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 109 (SDQS1). 00 Normal 01 High 10 Max high 11 Max high
23–22 DS_FAST108	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 108 (SDQS0). 00 Normal 01 High 10 Max high 11 Max high
21–20 DS_FAST107	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 107 (SDCLK). 00 Normal 01 High 10 Max high 11 Max high
19–18 DS_FAST106	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 106 (SDCKE1). 00 Normal 01 High 10 Max high 11 Max high

Table 4-14. Drive Strength Control Register 8 Field Descriptions (continued)

Field	Description
17–16 DS_FAST105	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 105 (SDCKE0). 00 Normal 01 High 10 Max high 11 Max high
15–14 DS_FAST104	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 104 (SDWE_B). 00 Normal 01 High 10 Max high 11 Max high
13–12 DS_FAST103	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 103 (CAS_B). 00 Normal 01 High 10 Max high 11 Max high
11–10 DS_FAST102	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 102 (RAS_B). 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_FAST101	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 101 (MA10). 00 Normal 01 High 10 Max high 11 Max high
7–6 DS_FAST100	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 100 (DQM3). 00 Normal 01 High 10 Max high 11 Max high
5–4 DS_FAST99	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 99 (DQM2). 00 Normal 01 High 10 Max high 11 Max high
3–2 DS_FAST98	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 98 (DQM1). 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_FAST97	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 97 (DQM0). 00 Normal 01 High 10 Max high 11 Max high

4.2.14 Drive Strength Control Register 9

The Drive Strength Control Register 9 (DSCR9) controls the driving force parameters of the fast I/O signals in the i.MX27 device. Figure 4-14 shows the register and Table 4-15 provides its field descriptions.

0x1002_7840 (DSCR9) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DS_FAST127		DS_FAST126		DS_FAST125		DS_FAST124		DS_FAST123		DS_FAST122		DS_FAST121	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DS_FAST120		DS_FAST119		DS_FAST118		DS_FAST117		DS_FAST116		DS_FAST115		DS_FAST114		DS_FAST113	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-14. Drive Strength Control Register 9 (DSCR9)

Table 4-15. Drive Strength Control Register 9 Field Descriptions

Field	Description
31–30	Reserved. These bits are reserved and should read 0.
29–28 DS_FAST127	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 127 (M_REQUEST). 00 Normal 01 High 10 Max high 11 Max high
27–26 DS_FAST126	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 126 (M_GRANT). 00 Normal 01 High 10 Max high 11 Max high
25–24 DS_FAST125	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 125 (IOIS16). 00 Normal 01 High 10 Max high 11 Max high
23–22 DS_FAST124	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 124 (PC_POE). 00 Normal 01 High 10 Max high 11 Max high
21–20 DS_FAST123	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 123 (PC_RW_B). 00 Normal 01 High 10 Max high 11 Max high

Table 4-15. Drive Strength Control Register 9 Field Descriptions (continued)

Field	Description
19–18 DS_FAST122	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 122 (PC_RST). 00 Normal 01 High 10 Max high 11 Max high
17–16 DS_FAST121	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 121 (PC_BVD2). 00 Normal 01 High 10 Max high 11 Max high
15–14 DS_FAST120	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 120 (PC_BVD1). 00 Normal 01 High 10 Max high 11 Max high
13–12 DS_FAST119	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 119 (PC_VS2). 00 Normal 01 High 10 Max high 11 Max high
11–10 DS_FAST118	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 118 (PC_VS1). 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_FAST117	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 117 (PC_PWRON). 00 Normal 01 High 10 Max high 11 Max high
7–6 DS_FAST116	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 116 (PC_READY). 00 Normal 01 High 10 Max high 11 Max high
5–4 DS_FAST115	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 115 (PC_WAIT_B). 00 Normal 01 High 10 Max high 11 Max high

Table 4-15. Drive Strength Control Register 9 Field Descriptions (continued)

Field	Description
3–2 DS_FAST114	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 114 (PC_CD2_B). 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_FAST113	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 113 (PC_CD1_B). 00 Normal 01 High 10 Max high 11 Max high

4.2.15 Drive Strength Control Register 10

The Drive Strength Control Register 10 (DSCR10) controls the driving force parameters of the fast I/O signals in the i.MX27 device. Figure 4-15 shows the register and Table 4-16 provides its field descriptions.

0x1002_7844 (DSCR10) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DS_FAST143		DS_FAST142		DS_FAST141		DS_FAST140		DS_FAST139		DS_FAST138		DS_FAST137	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DS_FAST136		DS_FAST135		DS_FAST134		DS_FAST133		DS_FAST132		DS_FAST131		DS_FAST130		DS_FAST129	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-15. Drive Strength Control Register 10 (DSCR10)

Table 4-16. Drive Strength Control Register 10 Field Descriptions

Field	Description
31–30	Reserved. These bits are reserved and should read 0.
29–28 DS_FAST143	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 143 (SD3_CMD). 00 Normal 01 High 10 Max high 11 Max high
27–26 DS_FAST142	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 142 (SD3_CLK). 00 Normal 01 High 10 Max high 11 Max high

Table 4-16. Drive Strength Control Register 10 Field Descriptions (continued)

Field	Description
25–24 DS_FAST141	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 141 (SD2_CLK). 00 Normal 01 High 10 Max high 11 Max high
23–22 DS_FAST140	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 140 (LSCLK). 00 Normal 01 High 10 Max high 11 Max high
21–20 DS_FAST139	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 139 (CSI_MCLK). 00 Normal 01 High 10 Max high 11 Max high
19–18 DS_FAST138	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 138 (CSI_PIXCLK). 00 Normal 01 High 10 Max high 11 Max high
17–16 DS_FAST137	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 137 (CLKO). 00 Normal 01 High 10 Max high 11 Max high
15–14 DS_FAST136	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 136. 00 Normal 01 High 10 Max high 11 Max high
13–12 DS_FAST135	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 135 (NFWE_B). 00 Normal 01 High 10 Max high 11 Max high
11–10 DS_FAST134	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 134 (NFRE_B). 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_FAST133	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 133 (NFALE). 00 Normal 01 High 10 Max high 11 Max high

Table 4-16. Drive Strength Control Register 10 Field Descriptions (continued)

Field	Description
7–6 DS_FAST132	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 132 (NFCLE). 00 Normal 01 High 10 Max high 11 Max high
5–4 DS_FAST131	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 131 (NFWP_B). 00 Normal 01 High 10 Max high 11 Max high
3–2 DS_FAST130	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 130 (NFCE_B). 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_FAST129	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 129 (NFRB). 00 Normal 01 High 10 Max high 11 Max high

4.2.16 Drive Strength Control Register 11

The Drive Strength Control Register 11 (DSCR11) controls the driving force parameters of the fast I/O signals in the i.MX27 device. Figure 4-16 shows the register and Table 4-17 provides its field descriptions.

0x1002_7848 (DSCR11) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DS_FAST160		DS_FAST159		DS_FAST158		DS_FAST157		DS_FAST156		DS_FAST155		DS_FAST154		DS_FAST153	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DS_FAST152		DS_FAST151		DS_FAST150		DS_FAST149		DS_FAST148		DS_FAST147		DS_FAST146		DS_FAST145	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-16. Drive Strength Control Register 11 (DSCR11)

Table 4-17. Drive Strength Control Register 11 Field Descriptions

Field	Description
31–30 DS_FAST160	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 160 (ATA_DATA15). 00 Normal 01 High 10 Max high 11 Max high
29–28 DS_FAST159	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 159 (ATA_DATA14). 00 Normal 01 High 10 Max high 11 Max high
27–26 DS_FAST158	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 158 (ATA_DATA13). 00 Normal 01 High 10 Max high 11 Max high
25–24 DS_FAST157	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 157 (ATA_DATA12). 00 Normal 01 High 10 Max high 11 Max high
23–22 DS_FAST156	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 156 (ATA_DATA11). 00 Normal 01 High 10 Max high 11 Max high
21–20 DS_FAST155	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 155 (ATA_DATA10). 00 Normal 01 High 10 Max high 11 Max high
19–18 DS_FAST154	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 154 (ATA_DATA9). 00 Normal 01 High 10 Max high 11 Max high
17–16 DS_FAST153	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 153 (ATA_DATA8). 00 Normal 01 High 10 Max high 11 Max high
15–14 DS_FAST152	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 152 (ATA_DATA7). 00 Normal 01 High 10 Max high 11 Max high

Table 4-17. Drive Strength Control Register 11 Field Descriptions (continued)

Field	Description
13–12 DS_FAST151	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 151 (ATA_DATA6). 00 Normal 01 High 10 Max high 11 Max high
11–10 DS_FAST150	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 150 (ATA_DATA5). 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_FAST149	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 149 (ATA_DATA4). 00 Normal 01 High 10 Max high 11 Max high
7–6 DS_FAST148	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 148 (ATA_DATA3). 00 Normal 01 High 10 Max high 11 Max high
5–4 DS_FAST147	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 147 (ATA_DATA2). 00 Normal 01 High 10 Max high 11 Max high
3–2 DS_FAST146	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 146 (ATA_DATA1). 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_FAST145	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 145 (ATA_DATA0). 00 Normal 01 High 10 Max high 11 Max high

4.2.17 Drive Strength Control Register 12

The Drive Strength Control Register 12 (DSCR12) controls the driving force parameters of the fast I/O signals in the i.MX27 device. Figure 4-17 shows the register and Table 4-18 provides its field descriptions.

0x1002_784C (DSCR12) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	DS_FAST172		DS_FAST171		DS_FAST170		DS_FAST169	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DS_FAST 168		DS_FAST 167		DS_FAST 166		DS_FAST 165		DS_FAST164		DS_FAST163		DS_FAST162		DS_FAST161	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-17. Drive Strength Control Register 12 (DSCR12)

Table 4-18. Drive Strength Control Register 12 Field Descriptions

Field	Description
31–24	Reserved. These bits are reserved and should read 0.
23–22 DS_FAST172	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 172 (USBOTG_CLK). 00 Normal 01 High 10 Max high 11 Max high
21–20 DS_FAST171	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 171 (USBOTG_NXT). 00 Normal 01 High 10 Max high 11 Max high
19–18 DS_FAST170	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 170 (USBOTG_STP). 00 Normal 01 High 10 Max high 11 Max high
17–16 DS_FAST169	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 169 (USBOTG_DIR). 00 Normal 01 High 10 Max high 11 Max high
15–14 DS_FAST168	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 168 (USBOTG_DATA7). 00 Normal 01 High 10 Max high 11 Max high
13–12 DS_FAST167	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 167 (USBOTG_DATA6). 00 Normal 01 High 10 Max high 11 Max high

Table 4-18. Drive Strength Control Register 12 Field Descriptions (continued)

Field	Description
11–10 DS_FAST166	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 166 (USBOTG_DATA5). 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_FAST165	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 165 (USBOTG_DATA4). 00 Normal 01 High 10 Max high 11 Max high
7–6 DS_FAST164	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 164 (USBOTG_DATA3). 00 Normal 01 High 10 Max high 11 Max high
5–4 DS_FAST163	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 163 (USBOTG_DATA2). 00 Normal 01 High 10 Max high 11 Max high
3–2 DS_FAST162	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 162 (USBOTG_DATA1). 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_FAST161	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 161 (USBOTG_DATA0). 00 Normal 01 High 10 Max high 11 Max high

4.2.18 Drive Strength Control Register 13

The Drive Strength Control Register 13 (DSCR13) controls the driving force parameters of the fast I/O signals in the i.MX27 device. Figure 4-18 shows the register and Table 4-19 provides its field descriptions.

0x1002_7850 (DSCR13) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	DS_FAST188		DS_FAST187		DS_FAST186		DS_FAST185	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DS_FAST184		DS_FAST183		DS_FAST182		DS_FAST181		DS_FAST180		DS_FAST179		DS_FAST178		DS_FAST177	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-18. Drive Strength Control Register 13 (DSCR13)

Table 4-19. Drive Strength Control Register 13 Field Descriptions

Field	Description
31–24	Reserved. These bits are reserved and should read 0.
23–22 DS_FAST188	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 188 (USBH2_CLK). 00 Normal 01 High 10 Max high 11 Max high
21–20 DS_FAST187	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 187 (USBH2_DIR). 00 Normal 01 High 10 Max high 11 Max high
19–18 DS_FAST186	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 186 (USBH2_NXT). 00 Normal 01 High 10 Max high 11 Max high
17–16 DS_FAST185	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 185 (USBH2_STP). 00 Normal 01 High 10 Max high 11 Max high
15–14 DS_FAST184	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 184 (USBH2_DATA7). 00 Normal 01 High 10 Max high 11 Max high
13–12 DS_FAST183	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 183 (USBH2_DATA6). 00 Normal 01 High 10 Max high 11 Max high

Table 4-19. Drive Strength Control Register 13 Field Descriptions (continued)

Field	Description
11–10 DS_FAST182	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 182 (USBH2_DATA5). 00 Normal 01 High 10 Max high 11 Max high
9–8 DS_FAST181	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 181 (USBH2_DATA4). 00 Normal 01 High 10 Max high 11 Max high
7–6 DS_FAST180	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 180.(USBH2_DATA3) 00 Normal 01 High 10 Max high 11 Max high
5–4 DS_FAST179	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 179 (USBH2_DATA2) 00 Normal 01 High 10 Max high 11 Max high
3–2 DS_FAST178	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 178 (USBH2_DATA1) 00 Normal 01 High 10 Max high 11 Max high
1–0 DS_FAST177	Drive Strength Fast I/O. Controls the driving strength of fast I/O group 177 (USBH2_DATA0) 00 Normal 01 High 10 Max high 11 Max high

4.2.19 Pull Strength Control Register (PSCR)

The Pull Strength Control Register (PSCR) controls the pull strength value and direction for the chip. Figure 4-19 shows the register and Table 4-20 provides its field descriptions.

0x1002_7854 (PSCR) Access: User R/W

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUENCR7		PUENCR6		PUENCR5		PUENCR4		PUENCR3		PUENCR2		PUENCR1		PUENCR0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-19. Pull Strength Control Register (PSCR)

Table 4-20. Pull Strength Control Register Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15–14 PUENCR	PUEN Strength Control 7. Bit selects direction (up or down) and strength. (SD2_D0_MSHC_DATA0) 00 100k pull-down 01 100k pull-up 10 47k pull-up 11 22k pull-up
13–12 PUENCR6	PUEN Strength Control 6. Bit selects direction (up or down) and strength. (SD2_D1_MSHC_DATA1) 00 100k pull-down 01 100k pull-up 10 47k pull-up 11 22k pull-up
11–10 PUENCR5	PUEN Strength Control 5. Bit selects direction (up or down) and strength. (SD2_D2_MSHC_DATA2) 00 100k pull-down 01 100k pull-up 10 47k pull-up 11 22k pull-up
9–8 PUENCR4	PUEN Strength Control 4. Bit selects direction (up or down) and strength. (SD2_D3_MSHC_DATA3) 00 100k pull-down 01 100k pull-up 10 47k pull-up 11 22k pull-up
7–6 PUENCR3	PUEN Strength Control 3. Bit selects direction (up or down) and strength. (SD2_CMD_MSHC_BS) 00 100k pull-down 01 100k pull-up 10 47k pull-up 11 22k pull-up

Table 4-20. Pull Strength Control Register Field Descriptions (continued)

Field	Description
5–4 PUENCR2	PUEN Strength Control 2. Bit selects direction (up or down) and strength. (SD2_CLK_MSHC_SCLK) 00 100k pull-down 01 100k pull-up 10 47k pull-up 11 22k pull-up
3–2 PUENCR1	PUEN Strength Control 1. Bit selects direction (up or down) and strength. (SD1_D3_CSPI3_SS) 00 100k pull-down 01 100k pull-up 10 47k pull-up 11 22k pull-up
1–0 PUENCR0	PUEN Strength Control 0. Bit selects direction (up or down) and strength. (ATA_DATA3_SD3_D3S) 00 100k pull-down 01 100k pull-up 10 47k pull-up 11 22k pull-up

4.2.20 Priority Control and Select Register (PCSR)

The Priority Control and Select Register (PCSR) consist of the master high priority and slave alternate context priority select to the ARM9 Platform. Figure 4-20 shows the register and Table 4-21 provides its field descriptions.

Address 0x1002_7858 (PCSR) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	S3_AMPR_SEL	S2_AMPR_SEL	S1_AMPR_SEL	S0_AMPR_SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	M5_HIGH_PRIORITY	M4_HIGH_PRIORITY	M3_HIGH_PRIORITY	M2_HIGH_PRIORITY	M1_HIGH_PRIORITY	M0_HIGH_PRIORITY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 4-20. Priority Control and Select Register (PCSR)

Table 4-21. Priority Control and Select Register Field Descriptions

Field	Description
31–20	Reserved. These bits are reserved and should read 0.
19–16 S3_AMPR_SEL S2_AMPR_SEL S1_AMPR_SEL S0_AMPR_SEL	Slave Alternate Context Priority Select. Inputs to the ARM9 Platform to select the priority determination and control source for the appropriate slave port. (Note s0 is the primary AHB and does not come out of the ARM9 Platform. 0 Priority determination and control is made by regular registers. 1 Priority determination and control is made by alternate registers set in the Crossbar switch.
15–6	Reserved. These bits are reserved and should read 0.
5–0 M5_HIGH_PRIORITY M4_HIGH_PRIORITY M3_HIGH_PRIORITY M2_HIGH_PRIORITY M1_HIGH_PRIORITY M0_HIGH_PRIORITY	Master High Priority. Inputs to the ARM9 Platform to elevate to highest appropriate master ports priority level to each slave above all other master ports priority levels which do not have this input asserted. If more than one master has its high priority input asserted, priority level is determined by the software programmed priority assignments inside the Crossbar switch. 0 Master port low priority 1 Master port high priority

4.2.21 Power Management Control Register (PMCR)

The Power Management Control Register (PMCR) controls the DPTC function of the chip. Figure 4-21 shows the register and Table 4-22 provides its field descriptions.

0x1002_7860 (PMCR) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MC	EM	UP	LO	0	REFCOUNTER										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RVEN	0	VSTBY		0	0	DCR	RCLK ON	DRCE 3	DRCE 2	DRCE 1	DRCE 0	DIM		DIE	DPTE N
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-21. Power Management Control Register (PMCR)

Table 4-22. Power Management Control Register Field Descriptions

Field	Description
31 MC	MC. Measure complete status bit 0 On progress or idle 1 Measure completed
30 EM	EM. Emergency interrupt state bit 0 No Emergency interrupt 1 Emergency interrupt is detected.
29 UP	UP. Upper_limit interrupt state bit 0 No Upper_limit interrupt is detected. 1 Upper_limit interrupt is detected.

Table 4-22. Power Management Control Register Field Descriptions (continued)

Field	Description
28 LO	LO. Lower_limit interrupt state bit. 0 No lower_limit interrupt. 1 Lower_limit interrupt is detected.
27	Reserved. These bits are reserved and should read 0.
26–16 REFCOUNTER	Reference Counter Value. These bits contains the value of reference counter in comparison stage.
15 RVEN	Reduced Voltage Mode Enable. This bit controls whether enable RV mode when chip is in sleep mode. 0 Disable RV mode is in sleep mode. 1 Enable RV mode is in sleep mode.
14	Reserved. These bits are reserved and should read 0.
13–12 VSTBY	Voltage Standby Control. These two bits will be put on Boot1 and Boot0 when chip is in sleep mode. And used to inform PMIC to change the voltage to the chip.
11–10	Reserved. These bits are reserved and should read 0.
9 DCR	DPTC counting range. This bit sets how many times the system clock may increment and the reference circuits remain active (and their output signals will be counted). Value of ‘1’ causes a 256 system clock count. Value of ‘0’ causes a 128 system clock count. 0 128 system clock count 1 256 system clock count
8 RCLKON	DPTC Reference Clock Monitor On. Enable Reference clock for debug. 0 Normal operation 1 Reference clock always on
7 DRCE3	DPTC reference circuit3 enable. This bit defines if reference circuit3 is enabled during DPTC operation. 0 DPTC reference circuit3 is disabled. 1 DPTC reference circuit3 is enabled.
6 DRCE2	DPTC reference circuit2 enable. This bit defines if reference circuit2 is enabled during DPTC operation. 0 DPTC reference circuit2 is disabled. 1 DPTC reference circuit2 is enabled.
5 DRCE1	DPTC reference circuit1 enable. This bit defines if reference circuit1 is enabled during DPTC operation. 0 DPTC reference circuit1 is disabled. 1 DPTC reference circuit1 is enabled.
4 DRCE0	DPTC reference circuit0 enable. This bit defines if reference circuit0 is enabled during DPTC operation. 0 DPTC reference circuit0 is disabled. 1 DPTC reference circuit0 is enabled.
3–2 DIM	DPTC interrupt mask. these bits control how DPTC generate its interrupt. 00 DPTC will generate an interrupt in all cases. 01 DPTC will generate an interrupt only in lower_limit case. 10 DPTC will generate an interrupt only in upper_limit case. 11 DPTC will generate an interrupt only in emergency case.

Table 4-22. Power Management Control Register Field Descriptions (continued)

Field	Description
1 DIE	DPTC Interrupt enable. This bit enables DPTC interrupt generation. 0 No interrupt will be generated. 1 Enable interrupt generation
0 DPTEN	DPTC enable. This bit enables the DPTC block and starts the reference circuit clock counting and compares this to look-up table values. 0 DPTC is disabled. 1 DPTC is enabled.

4.2.22 DPTC Comparator Value Register 0 (DCVR0)

The DPTC Comparator Value Register 0 (DCVR0) contains the DPTC comparator value for the DPTC in the i.MX27 processor. Figure 4-22 shows the register and Table 4-23 provides its field descriptions.

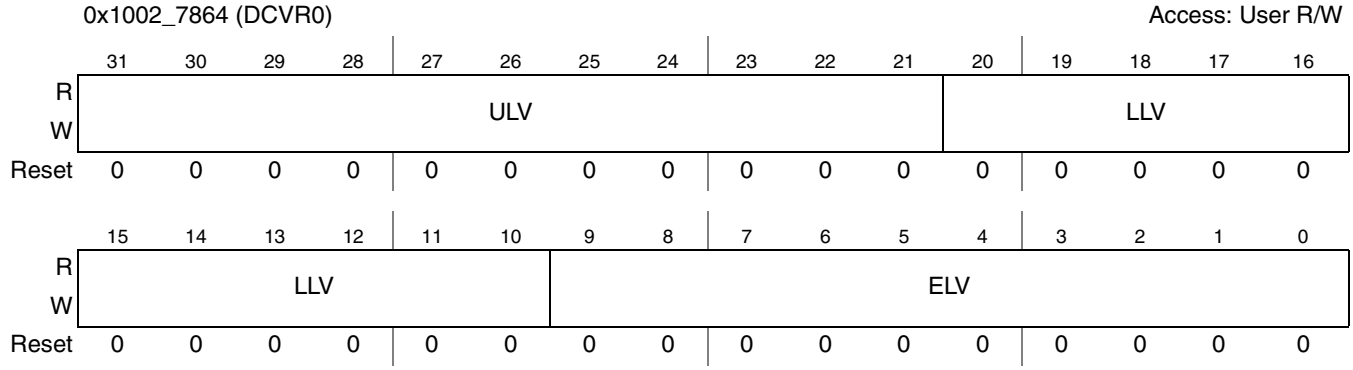


Figure 4-22. DPTC Comparator Value Register 0 (DCVR0)

Table 4-23. DPTC Comparator Value Register 0 Field Description

Field	Description
31–21 ULV	Upper Limit. Value for the upper performance limit of the reference circuit 0 clock counter.
20–10 LLV	Lower Limit. Value for the lower performance limit of the reference circuit 0 clock counter.
9–0 ELV	Emergency Limit. Value for the lower performance limit of the reference circuit 0 clock counter. This serves as an “emergency” lower limit, which indicates a critical value.

4.2.23 DPTC Comparator Value Register 1 (DCVR1)

The DPTC Comparator Value Register 1 (DCVR1) contains the DPTC comparator value for the DPTC in the i.MX27 processor. Figure 4-23 shows the register and Table 4-24 provides its field descriptions.

0x1002_7868 (DCVR1) Access: User R/W

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

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

Figure 4-23. DPTC Comparator Value Register 1 (DCVR1)

Table 4-24. DPTC Comparator Value Register 1 Field Descriptions

Field	Description
31–21 ULV	Upper Limit. Value for the upper performance limit of the reference circuit 1 clock counter.
20–10 LLV	Lower Limit. Value for the lower performance limit of the reference circuit 1 clock counter.
9–0 ELV	Emergency Limit. Value for the lower performance limit of the reference circuit 1 clock counter. This serves as an “emergency” lower limit, which indicates a critical value

4.2.24 DPTC Comparator Value Register 2

The DPTC Comparator Value Register 2 (DCVR2) contains the DPTC comparator value for the DPTC in the i.MX27 processor. Figure 4-24 shows the register and Table 4-25 provides its field descriptions.

0x1002_786C (DCVR2) Access: User R/W

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

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

Figure 4-24. DPTC Comparator Value Register 2 (DCVR2)

Table 4-25. DPTC Comparator Value Register 2 Field Descriptions

Field	Description
31–21 ULV	Upper Limit. Value for the upper performance limit of the reference circuit 2 clock counter.

Table 4-25. DPTC Comparator Value Register 2 Field Descriptions (continued)

Field	Description
20–10 LLV	Lower Limit. Value for the lower performance limit of the reference circuit 2 clock counter.
9–0 ELV	Emergency Limit. Value for the lower performance limit of the reference circuit 2 clock counter. This serves as an “emergency” lower limit, which indicates a critical value.

4.2.25 DPTC Comparator Value Register 3

The DPTC Comparator Value Register 3 (DCVR3) contains the DPTC comparator value for the DPTC in the i.MX27 processor. Figure 4-25 shows the register and Table 4-26 provides its field descriptions.

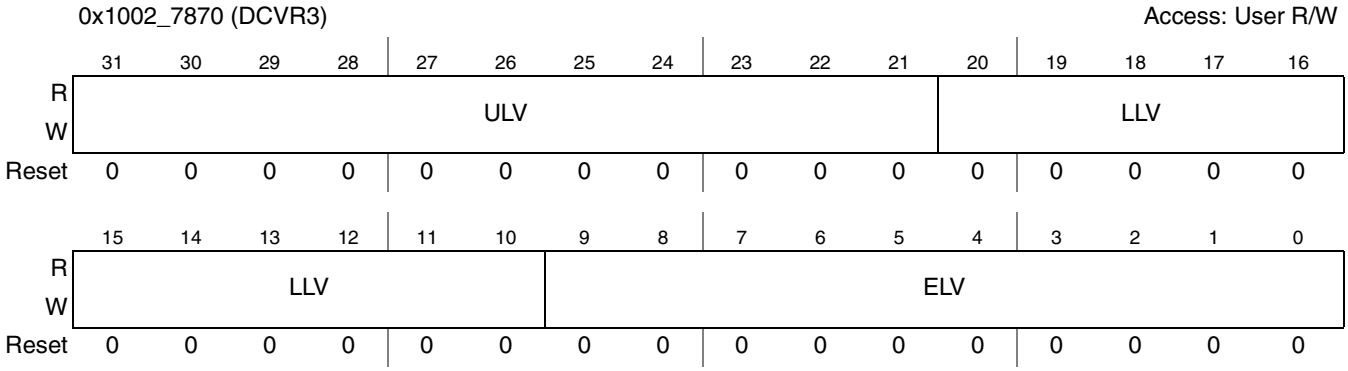


Figure 4-25. DPTC Comparator Value Register 3 (DCVR3)

Table 4-26. DPTC Comparator Value Register 3 Description

Field	Description
31–21 ULV	Upper Limit. Value for the upper performance limit of the reference circuit 3 clock counter.
20–10 LLV	Lower Limit. Value for the lower performance limit of the reference circuit 3 clock counter.
9–0 ELV	Emergency Limit. Value for the lower performance limit of the reference circuit 3 clock counter. This serves as an “emergency” lower limit, which indicates a critical value.

4.2.26 PMIC Pad Control Register (PPCR)

The PMIC Pad Control Register (PPCR) contains control bits of Boot0 and Boot1 pads which used for RV function in low power mode. Figure 4-26 shows the register and Table 4-27 provides its field descriptions.

0x1002_7874 (PPCR) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PUS1		PUE1	DSE1		OE1	0	0	PUS0		PUE0	DSE0		OE0
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1

Figure 4-26. PMIC Pad Control Register (PPCR)

Table 4-27. PMIC Pad Control Register Field Description

Field	Description
31–14	Reserved. These bits are reserved and should read 0.
13–12 PUS1	PUS1. PUS control of BOOT1 pad. Only used when RVEN bit is set.
11 PUE1	PUE1. PUE control of BOOT1 pad. Only used when RVEN bit is set.
10–9 DSE1	DSE1. DSE control of BOOT1 pad. Only used when RVEN bit is set.
8 OE1	OE1. OE control of BOOT1 pad. Only used when RVEN bit is set.
7–16	Reserved. These bits are reserved and should read 0.
5–4 PUS0	PUS0. PUS control of BOOT0 pad. Only used when RVEN bit is set.
3 PUE0	PUE0. PUE control of BOOT0 pad. Only used when RVEN bit is set.
2–1 DSE0	DSE0. DSE control of BOOT0 pad. Only used when RVEN bit is set.
0 OE0	OE0. OE control of BOOT0 pad. Only used when RVEN bit is set.

4.3 System Boot Mode Selection

The operational system boot mode of the i.MX27 processor upon system reset is determined by the configuration of the four external input pins, BOOT[3:0]. The settings of these pins control where the system is boot from and the memory port size.

The i.MX27 processor always begins fetching instruction from the address 0x00000000 after reset. The BOOT[3:0] pins control the memory region that is mapped to the address 0x0. Upon power up, if the BOOT_INT is 1, the Boot Address will always be 0x00000000. If the fuse of the BOOT_INT is blown, the BOOT Address will be generated based on the BOOT[3:0] information. The boot modes are defined in Table 4-28. These boot modes information are registered during the system reset. When an external chip

select is enabled by the BOOT[3:0] pins, the reset vector 0x0 will jump to the corresponding boot address space.

NOTE

The BOOT pins must not change once the i.MX27 device is out of reset. For proper operation, BOOT[3] must always be tied to VSS.

Table 4-28. System Boot Mode Selection

Inputs BOOT[3:0]	Output Signals Active Device (Boot Internal)	Output Signals Active Device (Boot External)	Boot Address
0000	iROM (Bootstrap USB/UART)	iROM Bootstrap USB/UART	0x00000030
0010	iROM (8-bit 2 Kbyte NAND Flash)	8-bit 2 Kbyte NAND Flash	0xD8000000
0011	iROM (16-bit 2 Kbyte NAND Flash)	16-bit 2 Kbyte NAND Flash	0xD8000000
0100	iROM (16-bit 512 byte NAND Flash)	16-bit 512 Kbyte NAND Flash	0xD8000000
0101	iROM (16-bit CS0 at D[15:0] (NOR Flash))	16-bit CS0 at D[15:0] (NOR Flash)	0xC0000000
0110	Reserved	Reserved	0xC0000000
0111	iROM (8-bit 512 byte NAND Flash)	8-bit 512B NAND Flash	0xD8000000

Chapter 5
Signal Descriptions and Pin Assignments

5.1 Introduction

This chapter identifies and describes the i.MX27 signals and their pin assignments.

5.2 Signal Descriptions

The i.MX27 signals are described in Table 5-1. Most of the signals shown in Table 5-1 are multiplexed with other signals. For simplicity, only the primary signal names are shown. See Table 5-2 for complete information on the signal multiplexing schemes of these signals.

Table 5-1. i.MX27 Signal Descriptions

Pad Name	Function/Notes
External Bus/Chip Select (EMI)	
A [13:0]	Address bus signals, shared with SDRAM/MDDR, WEIM and PCMCIA, A[10] for SDRAM/MDDR is not the address but the pre-charge bank select signal.
MA10	Address bus signals for SDRAM/MDDR
A [25:14]	Address bus signals, shared with WEIM and PCMCIA
SDBA[1:0]	SDRAM/MDDR bank address signals
SD[31:0]	Data bus signals for SDRAM, MDDR
SDQS[3:0]	MDDR data sample strobe signals
DQM0–DQM3	SDRAM data mask strobe signals
EB0	Active low external enable byte signal that controls D [15:8], shared with PCMCIA $\overline{\text{PC\_REG}}$.
EB1	Active low external enable byte signal that controls D [7:0], shared with PCMCIA $\overline{\text{PC\_IORD}}$.
$\overline{\text{OE}}$	Memory Output Enable—Active low output enables external data bus, shared with PCMCIA $\overline{\text{PC\_IOWR}}$.
$\overline{\text{CS}}$ [5:0]	Chip Select—The chip select signals $\overline{\text{CS}}$ [3:2] are multiplexed with $\overline{\text{CSD}}$ [1:0] and are selected by the Function Multiplexing Control Register (FMCR) in the System Control chapter. By default $\overline{\text{CSD}}$ [1:0] is selected. $\overline{\text{DTACK}}$ is multiplexed with $\overline{\text{CS4}}$. CS[5:4] are multiplexed with ETMTRACECLK and ETMTRACESYNC; PF22, 21.
ECB	Active low input signal sent by flash device to the EIM whenever the flash device must terminate an on-going burst sequence and initiate a new (long first access) burst sequence.
LBA	Active low signal sent by flash device causing external burst device to latch the starting burst address.

Table 5-1. i.MX27 Signal Descriptions (continued)

Pad Name	Function/Notes
BCLK	Clock signal sent to external synchronous memories (such as burst flash) during burst mode.
RW	$\overline{RW}$ signal—Indicates whether external access is a read (high) or write (low) cycle. This signal is also shared with the PCMCIA $\overline{PC\_WE}$.
RAS	SDRAM/MDDR Row Address Select signal
CAS	SDRAM/MDDR Column Address Select signal
SDWE	SDRAM Write Enable signal
SDCKE0	SDRAM Clock Enable 0
SDCKE1	SDRAM Clock Enable 1
SDCLK	SDRAM Clock
SDCLK_B	SDRAM Clock_B
NFWE_B	NFC Write enable signal, multiplexed with ETMPIPESTAT2; PF6
NFRE_B	NFC Read enable signal, multiplexed with ETMPIPESTAT1; PF5
NFALE	NFC Address latch signal, multiplexed with ETMPIPESTAT0; PF4
NFCLE	NFC Command latch signal, multiplexed with ETMTRACEPKT0; PF1
NFWP_B	NFC Write Permit signal, multiplexed with ETMTRACEPKT1; PF2
NFCE_B	NFC Chip enable signal, multiplexed with ETMTRACEPKT2; PF3
NFRB	NFC read Busy signal, multiplexed with ETMTRACEPKT3; PF0
D[15:0]	Data Bus signal, shared with EMI, PCMCIA, and NFC
PC_CD1_B	PCMCIA card detect signal, multiplexed with ATA ATA_DIOR signal; PF20
PC_CD2_B	PCMCIA card detect signal, multiplexed with ATA ATA_DIOW signal; PF19
PC_WAIT_B	PCMCIA WAIT signal, multiplexed with ATA ATA_CS1 signal; PF18
PC_READY	PCMCIA READY/IRQ signal, multiplexed with ATA ATA_CS0 signal; PF17
PC_PWRON	PCMCIA signal, multiplexed with ATA ATA_DA2 signal; PF16
PC_VS1	PCMCIA voltage sense signal, multiplexed with ATA ATA_DA1 signal; PF14
PC_VS2	PCMCIA voltage sense signal, multiplexed with ATA ATA_DA0 signal; PF13
PC_BVD1	PCMCIA Battery voltage detect signal, multiplexed with ATA ATA_DMARQ signal; PF12
PC_BVD2	PCMCIA Battery voltage detect signal, multiplexed with ATA ATA_DMACK signalPF11
PC_RST	PCMCIA card reset signal, multiplexed with ATA ATA_RESET_B signal; PF10
IOIS16	PCMCIA mode signal, multiplexed with ATA ATA_INTRQ signal; PF9
PC_RW_B	PCMCIA read write signal, multiplexed with ATA ATA_IORDY signal; PF8
PC_POE	PCMCIA output enable signal, multiplexed with ATA ATA_BUFFER_EN signal; PF7

Table 5-1. i.MX27 Signal Descriptions (continued)

Pad Name	Function/Notes
Clocks and Resets	
CLKO	Clock Out signal selected from internal clock signals. Refer to the clock controller for internal clock selection; PF15.
EXT_60M	This is a special factory test signal. To ensure proper operation, connect this signal to ground.
EXT_266M	This is a special factory test signal. To ensure proper operation, connect this signal to ground.
OSC26M_TEST	This is a special factory test signal. To ensure proper operation, leave this signal as a no connect.
RESET_IN	Master Reset—External active low Schmitt trigger input signal. When this signal goes active, all modules (except the reset module, SDRAMC module, and the clock control module) are reset.
RESET_OUT	Reset_Out—Output from the internal Hreset_b; and the Hreset can be caused by all reset source: power on reset, system reset (RESET_IN), and watchdog reset.
POR	Power On Reset—Active low Schmitt trigger input signal. The $\overline{\text{POR}}$ signal is normally generated by an external RC circuit designed to detect a power-up event.
XTAL26M	Oscillator output to external crystal
EXTAL26M	Crystal input (26 MHz), or a 16 MHz to 32 MHz oscillator (or square-wave) input when internal oscillator circuit is shut down.
CLKMODE[1:0]	These are special factory test signals. To ensure proper operation, do not connect to these signals.
EXTAL32K	32 kHz crystal input (Note: in the RTC power domain)
XTAL32K	Oscillator output to 32 kHz crystal (Note: in the RTC power domain)
Power_cut	(Note: in the RTC power domain)
Power_on_reset	(Note: in the RTC power domain)
osc32K_bypass	The signal for osc32k input bypass (Note: in the RTC power domain)
Bootstrap	
BOOT [3:0]	System Boot Mode Select—The operational system boot mode of the i.MX27 processor upon system reset is determined by the settings of these pins. BOOT[1:0] are also used as handshake signals to PMIC(VSTBY).
JTAG	
JTAG_CTRL	JTAG Controller select signal—JTAG_CTRL is sampled during rising edge of TRST. Must be pulled to logic high for proper JTAG interface to debugger. Pulling JTAG_CRTL low is for internal test purposes only.
TRST	Test Reset Pin—External active low signal used to asynchronously initialize the JTAG controller.
TDO	Serial Output for test instructions and data. Changes on the falling edge of TCK.
TDI	Serial Input for test instructions and data. Sampled on the rising edge of TCK.
TCK	Test Clock to synchronize test logic and control register access through the JTAG port.

Table 5-1. i.MX27 Signal Descriptions (continued)

Pad Name	Function/Notes
TMS	Test Mode Select to sequence JTAG test controller's state machine. Sampled on rising edge of TCK.
RTCK	JTAG Return Clock used to enhance stability of JTAG debug interface devices. This signal is multiplexed with 1-Wire; thus, utilizing 1-Wire will render RTCK unusable and vice versa; PE16.
Secure Digital Interface (X2)	
SD1_CMD	SD Command bidirectional signal—If the system designer does not want to make use of the internal pull-up, via the Pull-up enable register, a 4. 7K–69 K external pull up resistor must be added. This signal is multiplexed with CSPI3_MOSI; PE22.
SD1_CLK	SD Output Clock. This signal is multiplexed with CSPI3_SCLK; PE23.
SD1_D[3:0]	SD Data bidirectional signals—If the system designer does not want to make use of the internal pull-up, via the Pull-up enable register, a 50 K–69 K external pull up resistor must be added. SD1_D[3] is muxed with CSPI3_SS while SD1_D[0] is muxed with CSPI3_MISO PE21–18.
SD2_CMD	SD Command bidirectional signal. This signal is multiplexed with MSHC_BS; through GPIO multiplexed with SLCDC1_CS; PB8.
SD2_CLK	SD Output Clock signal. This signal is multiplexed with MSHC_SCLK, through GPIO multiplexed with SLCDC1_CLK; PB9.
SD2_D[3:0]	SD Data bidirectional signals. SD2_D[3:0] multiplexed with MSHC_DATA[0:3], also through GPIO SD2_1:0] multiplexed with SLCDC1_RS and SLDCD1_D0; PB7–PB4.
SD3_CMD	SD Command bidirectional signal. This signal is multiplexed with ETMTRACEPKT15 and also through GPIO PD1 multiplexed with FEC_TXD1.
SD3_CLK	SD Output Clock signal. This signal is through GPIO PD0 multiplexed with FEC_TXD0.
Note: SD3_DATA is multiplexed with ATA_DATA3–0.	
UARTs (X6)	
UART1_RTS	Request to Send input signal; PE15
UART1_CTS	Clear to Send output signal; PE14
UART1_RXD	Receive Data input signal; PE13
UART1_TXD	Transmit Data output signal, PE12
UART2_RXD	Receive Data input signal. This signal is multiplexed with KP_ROW6 signal from KPP; PE7.
UART2_TXD	Transmit Data output signal. This signal is multiplexed with KP_COL6 signal from KPP; PE6.
UART2_RTS	Request to Send input signal. This signal is multiplexed with KP_ROW7 signal from KPP; PE4.
UART2_CTS	Clear to Send output signal. This signal is multiplexed with KP_COL7 signal from KPP; PE3.
UART3_RTS	Request to Send input signal, PE11
UART3_CTS	Clear to Send output signal; PE10
UART3_RXD	Receive Data input signal; PE9
UART3_TXD	Transmit Data output signal; PE8

Table 5-1. i.MX27 Signal Descriptions (continued)

Pad Name	Function/Notes
Note: UART 4, 5, and 6 are multiplexed with COMS Sensor Interface signals.	
Keypad	
KP_COL[5:0]	Keypad Column selection signals. KP_COL[7:6] are multiplexed with $\overline{\text{UART2\_CTS}}$ and UART2_TXD respectively. Alternatively, KP_COL6 is also available on the internal factory test signal TEST_WB2. The Function Multiplexing Control Register in the System Control chapter must be used in conjunction with programming the GPIO multiplexing (to select the alternate signal multiplexing) to choose which signal KP_COL6 is available.
KP_ROW[5:0]	Keypad Row selection signals. KP_ROW[7:6] are multiplexed with $\overline{\text{UART2\_RTS}}$ and UART2_RXD signals respectively. The Function Multiplexing Control Register in the System Control chapter must be used in conjunction with programming the GPIO multiplexing (to select the alternate signal multiplexing) to choose which signals KP_ROW6 and KP_ROW7 are available.
Note: KP_COL[7:6] and KP_ROW[7:6] are multiplexed with UART2 signals as show above, also see UARTs table.	
PWM	
PWMO	PWM Output. This signal is multiplexed with PC_SPKOUT of PCMCIA, as well as TOUT2 and TOUT3 of the General Purpose Timer module; PE5.
CSPI (X3)	
CSPI1_MOSI	Master Out/Slave In signal, PD31
CSPI1_MISO	Master In/Slave Out signal, PD30
CSPI1_SS[2:0]	Slave Select (Selectable polarity) signal, the CSPI1_SS2 is multiplexed with USBH2_DATA5/RCV; and CSPI1_SS1 is multiplexed with $\overline{\text{EXT\_DMAGRANT}}$; PD26–28.
CSPI1_SCLK	Serial Clock signal, PD29
CSPI1_$\overline{\text{RDY}}$	Serial Data Ready signal, shared with Ext_DMAReq_B signal; PD25
CSPI2_MOSI	Master Out/Slave In signal, multiplexed with USBH2_DATA1/TXDP; PD24
CSPI2_MISO	Master In/Slave Out signal, multiplexed with USBH2_DATA2/TXDm; PD23
CSPI2_SS[2:0]	Slave Select (Selectable polarity) signals, multiplexed with USBH2_DATA4/RXDM, USBH2_DATA3/RXDP, USBH2_DATA6/SPEED; PD19–PD21
CSPI2_SCLK	Serial Clock signal, multiplexed with USBH2_DATA0/OEn; PD22
Note: CSPI3 CSPI3_MOSI, CSPI3_MISO, CSPI3_SS, andCSPI3_SCLK are multiplexed with SD1 signals.	
I^2C	
I2C2_SCL	I^2C2 Clock, through GPIO, multiplexed with SLCDC_data8; PC6
I2C2_SDA	I^2C2 Data, through GPIO, multiplexed with SLCDC_data7; PC5
I2C_CLK	I^2C1 Clock; PD18
I2C_DATA	I^2C1 Data; PD17
CMOS Sensor Interface	
CSI_HSYNC	Sensor port horizontal sync, multiplexed with UART5_RTSP; PB21

Table 5-1. i.MX27 Signal Descriptions (continued)

Pad Name	Function/Notes
CSI_VSYNC	Sensor port vertical sync, multiplexed with UART5_CTS; PB20
CSI_D7	Sensor port data, multiplexed with UART5_RXD; PB19
CSI_D6	Sensor port data, multiplexed with UART5_TXD; PB18
CSI_D5	Sensor port data; PB17
CSI_PIXCLK	Sensor port data latch clock; PB16
CSI_MCLK	Sensor port master clock, PB15
CSI_D4	Sensor port data, PD14
CSI_D3	Sensor port data, multiplexed with UART6_RTS; PB13
CSI_D2	Sensor port data, multiplexed with UART6_CTS; PB12
CSI_D1	Sensor port data, multiplexed with UART6_RXD; PB11
CSI_D0	Sensor port data, multiplexed with UART6_TXD; PB10
Serial Audio Port—SSI (Configurable to I2S Protocol and AC97) (2 to 4)	
SSI1_CLK	Serial clock signal that is output in master or input in slave; PC23
SSI1_TXD	Transmit serial data; PC22
SSI1_RXD	Receive serial data; PC21
SSI1_FS	Frame Sync signal that is output in master and input in slave; PC20
SSI2_CLK	Serial clock signal that is output in master or input in slave, multiplexed with GPT4_TIN. PC27
SSI2_TXD	Transmit serial data signal, multiplexed with GPT4_TOUT; PC26
SSI2_RXD	Receive serial data, multiplexed with GPT5_TIN; PC25
SSI2_FS	Frame Sync signal which is output in master and input in slave, multiplexed with GPT5_TOUT: PC24
SSI3_CLK	Serial clock signal which is output in master or input in slave. This signal is multiplexed with SLCDC2_CLK; through GPIO multiplexed with PC_WAIT_B; PC31.
SSI3_TXD	Transmit serial data signal which is multiplexed with SLCDC2_CS, through GPIO multiplexed with PC_READY; PC30
SSI3_RXD	Receive serial data which is multiplexed with SLCDC2_RS; through GPIO multiplexed with PC_VS1; PC29
SSI3_FS	Frame Sync signal which is output in master and input in slave. This signal is multiplexed with SLCDC2_D0; through GPIO multiplexed with PC_VS1; PC28.
SSI4_CLK	Serial clock signal which is output in master or input in slave; through GPIO multiplexed with PC_BVD1; PC19
SSI4_TXD	Transmit serial data; through GPIO multiplexed with PC_BVD2; PC18
SSI4_RXD	Receive serial data; through GPIO multiplexed with IOIS16; PC17
SSI4_FS	Frame Sync signal which is output in master and input in slave; PC16

Table 5-1. i.MX27 Signal Descriptions (continued)

Pad Name	Function/Notes
General Purpose Timers (X6)	
TIN	Timer Input Capture or Timer Input Clock—The signal on this input is applied to GPT 1–3 simultaneously. This signal is muxed with the Walk-up Guard Mode $\overline{\text{WKGD}}$ signal in the PLL, Clock, and Reset Controller module, and is also multiplexed with GPT6_TOUT; PC15.
TOUT1	Timer Output signal from General Purpose Timer1 (GPT1). This signal is multiplexed with SSI1_MCLK and SSI2_MCLK signal of SSI1 and SSI2. The pin name of this signal is simply TOUT, and is also multiplexed with GPT6_TIN; PC14.
Note: TOUT2, TOUT3 are multiplexed with PWMO pad; GPT4 and GPT5 signals are multiplexed with SSI2 pads.	
USB2.0	
USBOTG_DIR/TXDM	USB OTG direction/Transmit Data Minus signal, multiplexed with KP_ROW7A; PE2
USBOTG_STP/TXDM	USB OTG Stop signal/Transmit Data Minus signal, multiplexed with KP_ROW6A; PE1
USBOTG_NXT/TXDM	USB OTG NEXT/Transmit Data Minus signal, multiplexed with KP_COL6A; PE0
USBOTG_CLK/TXDM	USB OTG Clock/Transmit Data Minus signal, PE24
USBOTG_DATA7/SUSPEND	USB OTG Data7/Suspend signal, PE25
USBH2_STP/TXDM	USB Host2 Stop signal/Transmit Data Minus signal, PA4
USBH2_NXT/TXDM	USB Host2 NEXT/Transmit Data Minus signal, PA3
USBH2_DATA7/SUSPEND	USB Host2 Data7/Suspend signal, PA2
USBH2_DIR/TXDM	USB Host2 Direction/Transmit Data Minus signal, PA1
USBH2_CLK/TXDM	USB Host2 Clock/Transmit Data Minus signal; PA0
USBOTG_DATA3/RXDP	USB OTG data4/Receive Data Plus signal; multiplexed with SLCDC1_DAT15 through PC13
USBOTG_DATA4/RXDM	USB OTG data4/Receive Data Minus signal; multiplexed with SLCDC1_DAT14 through PC12
USBOTG_DATA1/TXDP	USB OTG data1/Transmit Data Plus signal; multiplexed with SLCDC1_DAT13 through PC11
USBOTG_DATA2/TXDm	USB OTG data2/Transmit Data Minus signal; multiplexed with SLCDC1_DAT12 through PC10
USBOTG_DATA0/Oen	USB OTG data0/Output Enable signal; multiplexed with SLCDC1_DAT11 through PC9
USBOTG_DATA6/SPEED	USB OTG data6/Suspend signal; multiplexed with SLCDC1_DAT10 and USBG_TXR_INT_B through PC8
USBOTG_DATA5/RCV	USB OTG data5/RCV signal; multiplexed with SLCDC1_DAT9 through PC7
USBH1_RXDP	USB Host1 Receive Data Plus signal, multiplexed with UART4_RXD; multiplexed with SLCDC1_DAT6 and UART4_RTS_ALT through PB31
USBH1_RXDM	USB Host1 Receive Data Minus signal; multiplexed with SLCDC1_DAT5 and UART4_CTS through PB30
USBH1_TXDP	USB Host1 Transmit Data Plus signal; multiplexed with UART4_CTS, multiplexed with SLCDC1_DAT4 and UART4_RXD_ALT through PB29
USBH1_TXDM	USB Host1 Transmit Data Minus signal; multiplexed with UART4_TXD, multiplexed with SLCDC1_DAT3 through PB28
USBH1_OE_B	USB Host1 Output Enable signal; multiplexed with SLCDC1_DAT2 through PB27

Table 5-1. i.MX27 Signal Descriptions (continued)

Pad Name	Function/Notes
USBH1_FS	USB Host1 Full Speed output signal, multiplexed with UART4_RTS, multiplexed with SLCDC1_DAT1 through PB26
USBH1_RCV	USB Host1 RCV signal; multiplexed with SLCDC1_DAT0 through PB25
USB_OC_B	USB OC signal. PB24
USB_PWR	USB Power signal; PB23
USBH1_SUSP	USB Host1 Suspend signal; PB22
LCD Controller and Smart LCD Controller	
OE_ACD	Alternate Crystal Direction/Output Enable; PA31
CONTRAST	This signal is used to control the LCD bias voltage as contrast control; PA30
VSYNC	Frame Sync or Vsync—This signal also serves as the clock signal output for gate; driver (dedicated signal SPS for Sharp panel HR-TFT); PA29.
HSYNC	Line Pulse or HSync; PA28
SPL_SPR	Sampling start signal for left and right scanning. Through GPIO, this signal is multiplexed with the SLCDC1_CLK; PA27.
PS	Control signal output for source driver (Sharp panel dedicated signal). This signal is multiplexed with the SLCDC1_CS; PA26.
CLS	Start signal output for gate driver. This signal is invert version of PS (Sharp panel dedicated signal). This signal is multiplexed with the SLCDC1_RS; PA25.
REV	Signal for common electrode driving signal preparation (Sharp panel dedicated signal). This signal is multiplexed with SLCDC1_D0; PA24.
LD [17:0]	LCD Data Bus—All LCD signals are driven low after reset and when LCD is off. Through GPIO, LD[15:0] signals are multiplexed with SLCDC1_DAT[15:0], SLCDC. PA23–PA6.
LSCLK	Shift Clock; PA5
Note: SLCDC signals are multiplexed with LCDC signals.	
ATA	
ATA_DATA15–0	ATA Data Bus, [15:0] are multiplexed with ETMTRACEPKT4–12, FEC_MDIO, ETMTRACEPKT13–14 SD3_D3–0; Through GPIO also are multiplexed with SLCDC 15–0, and FEC signals; PF23, PD16–PD2.
Noisy I/O Supply Pins	
N_{VDD}1–15, A_{VDD}	Noisy Supply for the I/O pins. There are 16 I/O voltage pads, N_{VDD}1 through N_{VDD}15 + A_{VDD}.

Table 5-1. i.MX27 Signal Descriptions (continued)

Pad Name	Function/Notes
Analog Supply Pins	
FPM_{VDD} $MPLL_{VDD}$ $OSC26_{VDD}$ $UPLL_{VDD}$ $OSC32_{VDD}$	Supply for analog blocks
FPMVSS MPLLVSS OSC26VSS OSC32VSS UPLLVSS	Quiet GND for analog blocks
Q_{VDD} Internal Power Supply	
Q_{VDD}	Power supply pins for silicon internal circuitry
QVSS	GND pins for silicon internal circuitry
$FUSE_{VDD}$	For $Fuse_{VDD}$
RTC_{VDD}	For RTC, SCC power supply
RTCVSS	For RTC, SCC GND
Note: Note: Both 1-Wire and Fast Ethernet Controller signals are multiplexed with other signals. As a result these signal names do not appear in this list. The signals are listed below with the named signal that they are multiplexed. 1-Wire Signals: The 1-Wire input and output signal is multiplexed with JTAG RTCK pad, PE16. Fast Ethernet Controller (FEC) Signals: FEC_TX_EN: Transmit enable signal, through GPIO multiplexed with ATA_DATA15 pad; PF23 FEC_TX_ER: Transmit Data Error; through GPIO multiplexed with ATA_DATA14 pad; PD16 FEC_COL: Collision signal; through GPIO multiplexed with ATA_DATA13 pad; PD15 FEC_RX_CLK: Receive Clock signal; through GPIO multiplexed with ATA_DATA12 pad; PD14 FEC_RX_DV: Receive data Valid signal; through GPIO multiplexed with ATA_DATA11 pad; PD13 FEC_RXD0: Receive Data0; through GPIO multiplexed with ATA_DATA10 pad; PD12 FEC_TX_CLK: Transmit Clock signal; through GPIO multiplexed with ATA_DATA9 pad; PD11 FEC_CRS: Carrier Sense enable; through GPIO multiplexed with ATA_DATA8 pad; PD10 FEC_MDC: Management Data Clock; through GPIO multiplexed with ATA_DATA7 pad; PD9 FEC_MDIO: Management Data Input/Output, multiplexed with ATA_DATA6 pad; PD8 FEC_RXD3–1: Receive Data; through GPIO multiplexed with ATA_DATA5–3 pad; PD7–5 FEC_RX_ER: Receive Data Error; through GPIO multiplexed with ATA_DATA2 pad; PD4 FEC_TXD3–2: Transmit Data; through GPIO multiplexed with ATA_DATA1–0; pad; PD3–2 FEC_TXD1: Transmit Data; through GPIO multiplexed with SD3_CLK pad; PD1 FEC_TXD0: Transmit Data; through GPIO multiplexed with SD3_CMD pad; PD0	
Note: The Rest ATA signals are multiplexed with PCMCIA Pads.	

5.3 I/O Power Supply and Signal Multiplexing Scheme

This section describes information about both the power supply for each I/O pin and its functional multiplexing scheme. Section 5.3.3, “I/O Mode and Supply Level” provides information on how to configure the power supply scheme for each device in the system (memory and external peripherals). The

functional multiplexing information shown in Table 5-2 enables the user to select the function of each pin by configuring the appropriate GPIO registers when those pins are multiplexed to provide different functions. In some cases, the use of the Function Multiplexing Control Register (FMCR) in Chapter 4, "System Control" may be required to select multiplex functionality.

5.3.1 Pull/Pull Strength/Open Drain Descriptions

For Table 5-2, the following notes describe the abbreviations used in the Pull/Pull Strength/Open Drain section.

- KP—Keeper Circuit permanently On when in Primary/Alternate Mode
- PU—Pull Up permanently On when in Primary/Alternate Mode
- PD—Pull Down permanently On when in Primary/Alternate Mode
- PUEN—Pull Up controllable from Module when in Primary/Alternate Mode
- PDEN—Pull Down controllable from Module when in Primary/Alternate Mode
- OD—Open Drain permanently On when in Primary/Alternate Mode
- ODEN—Open Drain Enable controllable from Module when in Primary/Alternate Mode

5.3.2 GPIO Default and Pull-Up Configuration

- The term Primary name is the package contact name.
- The Default column contains the GPIO default configuration as it appears after chip reset.
- Pull-up configuration and pull strength—Pin mux with GPIO means that Pull Up is controlled by the GPIO PUEN register (in Primary, Alternate or GPIO Mode), and the default pull strength is 100 K for all GPIO use.

5.3.3 I/O Mode and Supply Level

The supply level shown in Table 5-2 relates to the power bank segment. The same bank pad can be supplied same voltage. The voltage limitation relates to the I/O mode selected.

- I/O type of DDR mode—Voltage rating 1.65–1.95 V. Supply level 1.8 V is recommended for 100% duty cycle.
- I/O type of slow mode—Voltage rating 1.65–3.3 V. Supply level 3.05 V is recommended for 100% duty cycle;
- I/O type of fast mode—Voltage rating 1.65–3 V. Supply level 2.8 V is recommended for 100% duty cycle;
- For every analog pad a recommended supply voltage is shown.

Table 5-2. i.MX27 Pin MUX Table

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	Default
T19	AVDD	Supply	AVSS	static												AVSS
U18	AVDD	Supply	AVDD	static												AVDD
U19	AVDD	Slow/Hyst	BOOT2	I												BOOT2
V23	AVDD	Slow/Hyst	BOOT0	I												BOOT0
Y22	AVDD	Slow/Hyst	BOOT3	I												BOOT3
Y23	AVDD	Slow/Hyst	BOOT1	I												BOOT1
M18	FPMVDD	Supply	FPMVDD	static												FPMVDD
P15	FPMVDD	Supply	FPMVSS	static												FPMVSS
R18	FUSEVDD	Supply	FUSEVDD	static												FUSEVDD
R19	FUSEVDD	Supply	FUSEVSS	static												FUSEVSS
R15	MPLLVDD	Supply	MPLLVSS	static												MPLLVSS
T18	MPLLVDD	Supply	MPLLVDD	static												MPLLVDD
GND	NVDD1	Supply	NVSS1	static												NVSS1
H1	NVDD1	Fast	NFRB	I		ETMTRACEPKT3	O		PF0	PUEN						NFRB
J1	NVDD1	Fast	NFWP_B	O		ETMTRACEPKT1	O		PF2	PUEN						NFWP_B

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
K1	NVDD1	Fast	NFALE	O		ETMPIPESTAT0	O		PF4	PUEN						NFALE
L1	NVDD1	Fast	NFWE_B	O		ETMPIPESTAT2	O		PF6	PDEN						NFWE_B
L2	NVDD1	Fast	NFCE_B	O		ETMTRACEPKT2	O		PF3	PUEN						NFCLE
L5	NVDD1	Fast	NFRE_B	O		ETMPIPESTAT1	O		PF5	PUEN						NFRE_B
L6	NVDD1	Fast	NFCLE	O		ETMTRACEPKT0	O		PF1	PUEN						NFCE_B
M1	NVDD1	DDR	D14	B	KP											D14
M2	NVDD1	DDR	D15	B	KP											D15
M3	NVDD1	DDR	D11	B	KP											D11
M5	NVDD1	DDR	D13	B	KP											D13
M6	NVDD1	DDR	D9	B	KP											D9
N1	NVDD1	DDR	D12	B	KP											D12
N2	NVDD1	DDR	D7	B	KP											D7
N3	NVDD1	DDR	D5	B	KP											D5
N5	NVDD1	DDR	D3	B	KP											D3
N6	NVDD1	DDR	D1	B	KP											D1

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	Default
NVDD 1	NVDD1	Supply	NVDD1	static												NVDD1
P1	NVDD1	DDR	D10	B	KP											D10
P2	NVDD1	DDR	D8	B	KP											D8
R1	NVDD1	DDR	D6	B	KP											D6
R2	NVDD1	DDR	D4	B	KP											D4
T1	NVDD1	DDR	D2	B	KP											D2
T2	NVDD1	DDR	D0	B	KP											D0
A10	NVDD10	Slow/Hyst	SSI2_RXDAT	B		GPT5_TIN	I		PC25	PUEN						PC25
A11	NVDD10	Slow/Hyst	SSI3_RXDAT	B		SLCDC2_RS	I		PC29	PUEN					PC_VS1	PC29
A12	NVDD10	Slow_/Hyst	KP_ROW1	B	PU/100k											KP_ROW1
A13	NVDD10	Slow1/Hyst	KP_ROW5	B	PU/100k											KP_ROW5
A8	NVDD10	Slow/Hyst	SSI4_RXDAT	B					PC17	PUEN					IOIS16	PC17
A9	NVDD10	Slow/Hyst	SSI1_RXDAT	B					PC21	PUEN						PC21
B10	NVDD10	Slow/Hyst	SSI2_CLK	B		GPT4_TIN	I		PC27	PUEN						PC27

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	Default
B11	NVDD10	Slow/Hyst	SSI3_CLK	B		SLCDC2_CLK	I		PC31	PUEN					PC_WAIT_B	PC31
B12	NVDD10	Slow/Hyst	KP_ROW3	B	PU/100k											KP_ROW3
B7	NVDD10	Slow/Hyst	TIN	I					PC15	PUEN	GPT6_TOUT			WKGD_B		PC15
B8	NVDD10	Slow/Hyst	SSI4_CLK	B					PC19	PUEN					PC_BVD1	PC19
B9	NVDD10	Slow/Hyst	SSI1_CLK	B					PC23	PUEN						PC23
C12	NVDD10	Slow/Hyst	KP_ROW2	B	PU/100k											KP_ROW2
C9	NVDD10	Slow/Hyst	SSI3_TXDAT	B		SLCDC2_CS	I		PC30	PUEN					PC_READY	PC30
E10	NVDD10	Slow/Hyst	SSI3_FS	B		SLCDC2_D0	I		PC28	PUEN					PC_VS2	PC28
E11	NVDD10	Slow/Hyst	KP_ROW4	B	PU/100k											KP_ROW4
E8	NVDD10	Slow/Hyst	TOUT	O					PC14	PUEN	SSI_MCLK1	SSI_MCLK2		GPT6_TIN		PC14
E9	NVDD10	Slow/Hyst	SSI1_TXDAT	B					PC22	PUEN						PC22
F10	NVDD10	Slow/Hyst	SSI2_TXDAT	B		GPT4_TOUT	O		PC26	PUEN						PC26

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
F11	NVDD10	Slow/Hyst	KP_ROW0	B	PU/100k											KP_ROW0
F8	NVDD10	Slow/Hyst	SSI4_FS	B					PC16	PUEN						PC16
F9	NVDD10	Slow/Hyst	SSI1_FS	B					PC20	PUEN						PC20
G11	NVDD10	Supply	NVDD10	static												NVDD10
G8	NVDD10	Slow/Hyst	SSI4_TXDAT	B					PC18	PUEN					PC_BVD2	PC18
G9	NVDD10	Slow/Hyst	SSI2_FS	B		GPT5_TOUT	O		PC24	PUEN						PC24
GND	NVDD10	Supply	NVSS10	static												NVSS10
A5	NVDD11	Slow/Hyst	CSI_D3	I		UART6_RTS	I		PB13	PUEN			LCDC_TEST9			PB13
A6	NVDD11	Slow/Hyst	CSI_D5	I					PB17	PUEN			LCDC_TEST11			PB17
A7	NVDD11	Slow/Hyst	CSI_HSYNC	I		UART5_RTS	I		PB21	PUEN			LCDC_TEST15			PB21
B4	NVDD11	Slow/Hyst	CSI_D1	I		UART6_RXD	I		PB11	PUEN			LCDC_TEST7			PB11
B5	NVDD11	Fast/Hyst	CSI_MCLK	O					PB15	PUEN						PB15

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
B6	NVDD11	Slow/Hyst	CSI_D7	I		UART5_RXD	I		PB19	PUEN			LCDC_TEST13			PB19
C4	NVDD11	Slow/Hyst	CSI_D0	I		UART6_TXD	O		PB10	PUEN			LCDC_TEST6			PB10
E6	NVDD11	Slow/Hyst	CSI_D2	I		UART6_CTS	O		PB12	PUEN			LCDC_TEST8			PB12
E7	NVDD11	Fast/Hyst	CSI_PIXCLK	I					PB16	PUEN						PB16
F6	NVDD11	Slow/Hyst	CSI_D4	I					PB14	PUEN			LCDC_TEST10			PB14
F7	NVDD11	Slow/Hyst	CSI_D6	I		UART5_TXD	O		PB18	PUEN			LCDC_TEST12			PB18
G10	NVDD11	Supply	NVDD11	static												NVDD11
G7	NVDD11	Slow/Hyst	CSI_VSYNC	I		UART5_CTS	O		PB20	PUEN			LCDC_TEST14			PB20
GND	NVDD11	Supply	NVSS11	static												NVSS11
B3	NVDD12	Slow/Hyst	SPL_SPR	O					PA27	PDEN	SLCDC1_CLK					PA27

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	Default
C2	NVDD12	Slow/Hyst	CONTRAST	O					PA30	PUEN						PA30
D1	NVDD12	Slow/Hyst	HSYNC	O					PA28	PUEN						PA28
D2	NVDD12	Slow/Hyst	PS	O					PA26	PDEN	SLCDC1_CS					PA26
D3	NVDD12	Slow/Hyst	OE_ACD	O					PA31	PUEN						PA31
E1	NVDD12	Slow/Hyst	REV	O					PA24	PDEN	SLDCD1_D0					PA24
E2	NVDD12	Slow/Hyst	LD16	O					PA22	PUEN	Ext_DMAGrant_B					PA22
F1	NVDD12	Slow/Hyst	LD14	O					PA20	PUEN	SLCDC1_DAT14		SLCDC1_DAT6			PA20
F2	NVDD12	Slow/Hyst	LD10	O					PA16	PUEN	SLCDC1_DAT10		SLCDC1_DAT2			PA16
F5	NVDD12	Slow/Hyst	VSYNC	O					PA29	PUEN						PA29
G1	NVDD12	Slow/Hyst	LD8	O					PA14	PUEN	SLCDC1_DAT8		SLCDC1_DAT0			PA14
G2	NVDD12	Slow/Hyst	LD6	O					PA12	PUEN	SLCDC1_DAT6					PA12

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
G5	NVDD12	Slow/Hyst	LD17	O					PA23	PUEN						PA23
G6	NVDD12	Slow/Hyst	CLS	O					PA25	PDEN	SLCD C1_RS					PA25
GND	NVDD12	Supply	NVSS12	static												NVSS12
H2	NVDD12	Slow/Hyst	LD4	O					PA10	PUEN	SLCD C1_DAT4					PA10
H3	NVDD12	Slow/Hyst	LD12	O					PA18	PUEN	SLCD C1_DAT12		SLCD C1_DAT4			PA18
H5	NVDD12	Slow/Hyst	LD13	O					PA19	PUEN	SLCD C1_DAT13		SLCD C1_DAT5			PA19
H6	NVDD12	Slow/Hyst	LD15	O					PA21	PUEN	SLCD C1_DAT15		SLCD C1_DAT7			PA21
J2	NVDD12	Slow/Hyst	LD0	O					PA6	PUEN	SLCD C1_DAT0					PA6
J3	NVDD12	Slow/Hyst	LD2	O					PA8	PUEN	SLCD C1_DAT2					PA8
J5	NVDD12	Slow/Hyst	LD7	O					PA13	PUEN	SLCD C1_DAT7					PA13

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
J6	NVDD12	Slow/Hyst	LD5	O					PA11	PUEN	SLCD C1_DAT5					PA11
J7	NVDD12	Slow/Hyst	LD11	O					PA17	PUEN	SLCD C1_DAT11		SLCD C1_DAT3			PA17
K2	NVDD12	Fast/Hyst	LSCLK	O					PA5	PUEN						PA5
K5	NVDD12	Slow/Hyst	LD3	O					PA9	PUEN	SLCD C1_DAT3					PA9
K6	NVDD12	Slow/Hyst	LD1	O					PA7	PUEN	SLCD C1_DAT1					PA7
K7	NVDD12	Slow/Hyst	LD9	O					PA15	PUEN	SLCD C1_DAT9		SLCD C1_DAT1			PA15
L7	NVDD12	Supply	NVDD12	static												NVDD12
GND	NVDD13	Supply	NVSS13	static												NVSS13
L24	NVDD13	Slow/Hyst	osc32k_bypass	I												osc32k_bypass
M19	NVDD13	Supply	NVDD13	static												NVDD13
N19	NVDD13	Slow/Hyst	Power_on_reset	I	PU/100k											Power_on_reset

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	Default
N22	NVDD13	Slow/Hyst	Power_cut	I	PD/100k											Power_cut
C23	NVDD14	Fast/Hyst	CSPI2_SS1	B		USBH2_DATA3/RXDP	B		PD20	PUEN						PD20
C24	NVDD14	Slow/Hyst	USBH1_OE_B	B					PB27	PUEN	SLCDC1_DAT2					PB27
D22	NVDD14	Fast/Hyst	CSPI2_SS2	B		USBH2_DATA4/RXDM	B		PD19	PUEN						PD19
D23	NVDD14	Fast/Hyst	CSPI2_SCLK	B		USBH2_DATA0/OEn	B		PD22	PUEN						PD22
D24	NVDD14	Slow/Hyst	USBH1_TXDP	O		UART4_CTS	O		PB29	PDEN	SLCDC1_DAT4			UART4_RXD_ALT		PB29
E19	NVDD14	Slow/Hyst	USBH1_FS	B		UART4_RTS	I		PB26	PDEN	SLCDC1_DAT1					PB26
E22	NVDD14	Fast/Hyst	CSPI1_SS2	B		USBH2_DATA5/RCV	B		PD26	PUEN						PD26
E23	NVDD14	Fast/Hyst	CSPI2_MOSI	B		USBH2_DATA1/TXDP	B		PD24	PUEN						PD24
E24	NVDD14	Slow/Hyst	USBH1_RXDP	B		UART4_RXD	I		PB31	PDEN	SLCDC1_DAT6			UART4_RTS_ALT		PB31

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
F19	NVDD14	Slow/Hyst	USBH1_TXDM	O		UART4_TXD	O		PB28	PDEN	SLCDC1_DAT3					PB28
F20	NVDD14	Fast/Hyst	CSPI2_SS0	B		USBH2_DATA6/SPEED	B		PD21	PUEN						PD21
F23	NVDD14	Slow/Hyst	USB_PWR	O					PB23	PUEN						PB23
F24	NVDD14	Slow/Hyst	I2C2_SCL	B	OD				PC6	PUEN	SLCDC1_DAT8					PC6
G19	NVDD14	Slow/Hyst	USBH1_SUSP	B					PB22	PUEN						USB_BYP_B
G20	NVDD14	Fast/Hyst	CSPI2_MISO	B		USBH2_DATA2/TXDm	B		PD23	PUEN						PD23
GND	NVDD14	Supply	NVSS14	static												NVSS14
H18	NVDD14	Supply	NVDD14	static												NVDD14
H20	NVDD14	Slow/Hyst	USB_OC_B	I					PB24	PUEN						PB24
H22	NVDD14	Slow/Hyst	USBH1_RCV	B					PB25	PUEN	SLCDC1_DAT0					PB25
J20	NVDD14	Slow/Hyst	USBH1_RXDM	B					PB30	PDEN	SLCDC1_DAT5		UART4_CTS			PB30

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
J22	NVDD14	Slow/Hyst	I2C2_SDA	B	OD				PC5	PUEN	SLCDC1_DAT7					PC5
A3	NVDD15	Slow/Hyst	SD2_D3	B	PU/PD/100k	MSHC_DATA3	B	PU/PD/100k	PB7	PDEN	SLCDC1_RS		LCDC_TEST3			PB7
A4	NVDD15	Fast/Hyst	SD2_CLK	O		MSHC_SCLK	O		PB9	PDEN			LCDC_TEST5			PB9
C1	NVDD15	Slow/Hyst	SD2_D0	B	PU/100k	MSHC_DATA0	B	PD/100k	PB4	PDEN			LCDC_TEST0			PB4
C5	NVDD15	Slow/Hyst	SD2_CMD	B	PU/100k	MSHC_BS	O	PD/100k	PB8	PDEN	SLCDC1_CS		LCDC_TEST4			PB8
C8	NVDD15	Slow/Hyst	SD2_D2	B	PU/100k	MSHC_DATA2	B	PD/100k	PB6	PDEN	SLDCD1_D0		LCDC_TEST2			PB6
E3	NVDD15	Slow/Hyst	SD2_D1	B	PU/100k	MSHC_DATA1	B	PD/100k	PB5	PDEN	SLCDC1_CLK		LCDC_TEST1			PB5
GND	NVDD15	Supply	NVSS15	static												NVSS15
H7	NVDD15	Supply	NVDD15	static												NVDD15
AA1	NVDD2	DDR	SD30	B	KP											SD30
AA2	NVDD2	DDR	A24	O	KP											A24

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
AA3	NVDD2	DDR	SD27	B	KP											SD27
AB1	NVDD2	DDR	A23	O	KP											A23
AB2	NVDD2	DDR	SD24	B	KP											SD24
AB4	NVDD2	DDR	A21	O	KP											A21
AB5	NVDD2	DDR	SD21	B	KP											SD21
AB8	NVDD2	DDR	SD10	B	KP											SD10
AC3	NVDD2	DDR	A22	O	KP											A22
AC4	NVDD2	DDR	SD20	B	KP											SD20
AC5	NVDD2	DDR	SD17	B	KP											SD17
AC6	NVDD2	DDR	A18	O	KP											A18
AC7	NVDD2	DDR	A17	O	KP											A17
AD3	NVDD2	DDR	SD22	B	KP											SD22
AD4	NVDD2	DDR	SD19	B	KP											SD19
AD5	NVDD2	DDR	SD16	B	KP											SD16
AD6	NVDD2	DDR	SD14	B	KP											SD14
AD7	NVDD2	DDR	SD11	B	KP											SD11
GND	NVDD2	Supply	NVSS2	static												NVSS2

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
NVDD 2	NVDD2	Supply	NVDD2	static												NVDD2
P5	NVDD2	DDR	A9	B	KP											A9
P6	NVDD2	DDR	A12	B	KP											A12
R5	NVDD2	DDR	A5	B	KP											A5
R6	NVDD2	DDR	A7	B	KP											A7
T3	NVDD2	DDR	MA10	O												MA10
T5	NVDD2	DDR	SDBA1	O	KP											SDBA1
T6	NVDD2	DDR	A1	B	KP											A1
U1	NVDD2	DDR	A13	B	KP											A13
U2	NVDD2	DDR	A11	B	KP											A11
U3	NVDD2	DDR	A3	B	KP											A3
U5	NVDD2	DDR	SD31	B	KP											SD31
U6	NVDD2	DDR	A25	O	KP											A25
V1	NVDD2	DDR	A8	B	KP											A8
V2	NVDD2	DDR	A6	B	KP											A6
V5	NVDD2	DDR	SD26	B	KP											SD26
V6	NVDD2	DDR	SD28	B	KP											SD28

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	Default
V7	NVDD2	DDR	SD29	B	KP											SD29
V8	NVDD2	DDR	A19	O	KP											A19
W1	NVDD2	DDR	A4	B	KP											A4
W2	NVDD2	DDR	A2	B	KP											A2
W5	NVDD2	DDR	SD23	B	KP											SD23
W6	NVDD2	DDR	SDQS2	B	KP											SDQS2
W7	NVDD2	DDR	SD25	B	KP											SD25
W8	NVDD2	DDR	SDQS1	B	KP											SDQS1
W9	NVDD2	DDR	SD13	B	KP											SD13
Y1	NVDD2	DDR	A0	B	KP											A0
Y2	NVDD2	DDR	SDBA0	O	KP											SDBA0
Y3	NVDD2	DDR	SDQS3	B	KP											SDQS3
Y6	NVDD2	DDR	A20	O	KP											A20
Y7	NVDD2	DDR	SD18	B	KP											SD18
Y8	NVDD2	DDR	SD15	B	KP											SD15
Y9	NVDD2	DDR	SD12	B	KP											SD12
AB12	NVDD3	DDR	SD0	B	KP											SD0
AB9	NVDD3	DDR	A14	B	KP											A14

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
AC10	NVDD3	DDR	SD4	B	KP											SD4
AC11	NVDD3	DDR	SD1	B	KP											SD1
AC8	NVDD3	DDR	SD9	B	KP											SD9
AC9	NVDD3	DDR	SD5	B	KP											SD5
AD10	NVDD3	DDR	SD3	B	KP											SD3
AD11	NVDD3	DDR	DQM3	O	KP											DQM3
AD8	NVDD3	DDR	SD7	B	KP											SD7
AD9	NVDD3	DDR	SDQS0	B	KP											SDQS0
GND	NVDD3	Supply	NVSS3	static												NVSS3
NVDD 3	NVDD3	Supply	NVDD3	static												NVDD3
W10	NVDD3	DDR	SD6	B	KP											SD6
W11	NVDD3	DDR	A16	O	KP											A16
Y10	NVDD3	DDR	SD8	B	KP											SD8
Y11	NVDD3	DDR	A15	B	KP											A15
Y12	NVDD3	DDR	SD2	B	KP											SD2
AB13	NVDD4	DDR	RAS_B	O	KP											RAS_B
AB16	NVDD4	DDR	CS1_B	O												CS1_B

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
AB17	NVDD4	DDR	BCLK	O												BCLK
AC12	NVDD4	DDR	A10	B	KP											A10
AC13	NVDD4	DDR	CAS_B	O	KP											CAS_B
AC14	NVDD4	DDR	SDCKE0	O	KP											SDCKE0
AC15	NVDD4	DDR	RW_B	O												RW_B
AC16	NVDD4	Fast	ECB_B	I	PU/100k											ECB_B
AC17	NVDD4	DDR	EB1_B	O												EB1_B
AC18	NVDD4	Fast	JTAG_CTRL	I	PU/100k											JTAG_CTRL
AD12	NVDD4	DDR	DQM0	O	KP											DQM0
AD13	NVDD4	DDR_CLK	SDCLK	B												SDCLK
AD14	NVDD4	DDR_CLK	SDCLK_B	B												SDCLK_B
AD15	NVDD4	Fast	CS4_B	O		ETMTRACESYNC	O		PF21	PUEN				CS5_DTACK		CS4_B
AD16	NVDD4	DDR	CS0_B	O												CS0_B
AD17	NVDD4	Fast	CLKO	O					PF15	PUEN						CLKO
AD18	NVDD4	Fast	EXT_266M	I												EXT_266M
GND	NVDD4	Supply	NVSS4	static												NVSS4

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
V13	NVDD4	Supply	NVDD4	static												NVDD4
W12	NVDD4	DDR	DQM1	O	KP											DQM1
W13	NVDD4	DDR	SDWE_B	O	KP											SDWE_B
W14	NVDD4	DDR	CS3_B	O	KP											CS3_B
W15	NVDD4	Fast	CS5_B	O		ETMTRACECLK	O		PF22	PUEN						CS5_B
W16	NVDD4	DDR	EB0_B	O												EB0_B
W17	NVDD4	Fast	EXT_60M	I												EXT_60M
Y13	NVDD4	DDR	DQM2	O	KP											DQM2
Y14	NVDD4	DDR	SDCKE1	O	KP											SDCKE1
Y15	NVDD4	DDR	CS2_B	O	KP											CS2_B
Y16	NVDD4	DDR	LBA_B	O												LBA_B
Y17	NVDD4	DDR	OE_B	O												OE_B
AA22	NVDD5	Slow/Hyst	RESET_OUT_B	O					PE17	PUEN		PC_TEST_AHBST0				RESET_OUT_B
AB20	NVDD5	Slow/Hyst	CLKMODE0	I	PU/100k											CLKMODE0
AB21	NVDD5	Slow/Hyst	CLKMODE1	I	PU/100k											CLKMODE1

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	Default
AC19	NVDD5	Slow	PC_CD2_B	I		ATA_DIOW	O		PF19	PUEN						PC_CD2_B
AC20	NVDD5	Slow	PC_VS1	I		ATA_DA1	O		PF14	PUEN						PC_VS1
AC21	NVDD5	Slow	PC_RST	O		ATA_RESET_B	O		PF10	PUEN						PC_RST
AC22	NVDD5	Slow/Hyst	RESET_IN_B	I	PU/100k											RESET_IN_B
AD19	NVDD5	Slow	PC_READY	I		ATA_CS0	O		PF17	PUEN						PC_READY
AD20	NVDD5	Slow	PC_BVD1	I		ATA_DMARQ	I		PF12	PUEN						PC_BVD1
AD21	NVDD5	Slow	PC_RW_B	O		ATA_IORDY	I		PF8	PUEN						PC_RW_B
AD22	NVDD5	Slow/Hyst	POR_B	I												POR_B
GND	NVDD5	Supply	NVSS5	static												NVSS5
NVDD5	NVDD5	Supply	NVDD5	static												NVDD5
U20	NVDD5	Slow	IOIS16	I		ATA_INTRQ	I		PF9	PUEN						IOIS16
V20	NVDD5	Slow	PC_POE	O		ATA_BUFFER_EN	O		PF7	PUEN						PC_POE
W18	NVDD5	Slow	PC_CD1_B	I		ATA_DIOR	O		PF20	PUEN						PC_CD1_B

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
W19	NVDD5	Slow	PC_VS2	I		ATA_DA0	O		PF13	PUEN						PC_VS2
W20	NVDD5	Slow	PC_BVD2	I		ATA_DMACK	O		PF11	PUEN						PC_BVD2
Y18	NVDD5	Slow	PC_WAIT_B	I		ATA_CS1	O		PF18	PUEN						PC_WAIT_B
Y19	NVDD5	Slow	PC_PWRON	I		ATA_DA2	O		PF16	PDEN						PC_PWRON
GND	NVDD6	Supply	NVSS6	static												NVSS6
NVDD6	NVDD6	Supply	NVDD6	static												NVDD6
P19	NVDD6	Slow/Hyst	ATA_DATA6	B		FEC_MDIO	B		PD8	PUEN			SLCDC1_DAT6			ATA_DATA6
P20	NVDD6	Slow/Hyst	ATA_DATA2	B		SD3_D2	B		PD4	PUEN			SLCDC1_DAT2	FEC_RX_ER		ATA_DATA2
P23	NVDD6	Slow/Hyst	SD3_CMD	B					PD0	PUEN	FEC_TXD0					SD3_CMD
P24	NVDD6	Slow/Hyst	SD3_CLK	O		ETMTRACEPKT15	O		PD1	PUEN	FEC_TXD1					SD3_CLK
R20	NVDD6	Slow/Hyst	ATA_DATA10	B		ETMTRACEPKT9	O		PD12	PUEN			SLCDC1_DAT10	FEC_RXD0		ATA_DATA10

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
R23	NVDD6	Slow/Hyst	ATA_DATA0	B		SD3_D0	B		PD2	PUEN	FEC_TXD2		SLCDC1_DAT0			ATA_DATA0
R24	NVDD6	Slow/Hyst	ATA_DATA1	B		SD3_D1	B		PD3	PUEN	FEC_TXD3		SLCDC1_DAT1			ATA_DATA1
T20	NVDD6	Slow/Hyst	ATA_DATA14	B		ETMTRACEPKT5	O		PD16	PDEN	FEC_TX_ER		SLCDC1_DAT14			ATA_DATA14
T22	NVDD6	Slow/Hyst	ATA_DATA4	B		ETMTRACEPKT14	O		PD6	PUEN			SLCDC1_DAT4	FEC_RXD2		ATA_DATA4
T23	NVDD6	Slow/Hyst	ATA_DATA5	B		ETMTRACEPKT13	O		PD7	PUEN			SLCDC1_DAT5	FEC_RXD3		ATA_DATA5
T24	NVDD6	Slow/Hyst	ATA_DATA3	B		SD3_D3	B	PU/PD/100k	PD5	PDEN			SLCDC1_DAT3	FEC_RXD1		ATA_DATA3
U22	NVDD6	Slow/Hyst	ATA_DATA8	B		ETMTRACEPKT11	O		PD10	PUEN			SLCDC1_DAT8	FEC_CRS		ATA_DATA8
U23	NVDD6	Slow/Hyst	ATA_DATA12	B		ETMTRACEPKT7	O		PD14	PUEN			SLCDC1_DAT12	FEC_RX_CLK		ATA_DATA12
U24	NVDD6	Slow/Hyst	ATA_DATA7	B		ETMTRACEPKT12	O		PD9	PUEN	FEC_MDC		SLCDC1_DAT7			ATA_DATA7

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
V24	NVDD6	Slow/Hyst	ATA_DATA9	B		ETMTRACEPKT10	O		PD11	PUEN			SLCDC1_DAT9	FEC_TX_CLK		ATA_DATA9
W23	NVDD6	Slow/Hyst	ATA_DATA11	B		ETMTRACEPKT8	O		PD13	PUEN			SLCDC1_DAT11	FEC_RX_DV		ATA_DATA11
W24	NVDD6	Slow/Hyst	ATA_DATA13	B		ETMTRACEPKT6	O		PD15	PUEN			SLCDC1_DAT13	FEC_COL		ATA_DATA13
Y24	NVDD6	Slow/Hyst	ATA_DATA15	B		ETMTRACEPKT4	O		PF23	PDEN	FEC_TX_EN		SLCDC1_DAT15			ATA_DATA15
G18	NVDD7	Fast/Hyst	USBOTG_DATA1/TXDP	B					PC11	PUEN	SLCDC1_DAT13					PC11
G23	NVDD7	Fast/Hyst	USBOTG_DATA2/TXDm	B					PC10	PUEN	SLCDC1_DAT12					PC10
G24	NVDD7	Fast/Hyst	USBOTG_DATA6/SPEED	B					PC8	PUEN	SLCDC1_DAT10			USBG_TXR_INT_B		PC8
GND	NVDD7	Supply	NVSS7	static												NVSS7

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
H19	NVDD7	Fast/Hyst	USBOTG_DATA5/RCV	B					PC7	PUEN	SLCDC1_DAT9					PC7
H23	NVDD7	Fast/Hyst	USBH2_CLK/TXDM	I					PA0	PUEN						PA0
H24	NVDD7	Fast/Hyst	USBOTG_DATA4/RXDM	B					PC12	PUEN	SLCDC1_DAT14					PC12
J19	NVDD7	Fast/Hyst	USBOTG_DATA0/Oen	B					PC9	PUEN	SLCDC1_DAT11					PC9
J23	NVDD7	Fast/Hyst	USBH2_STP/TXDM	O					PA4	PUEN						PA4
J24	NVDD7	Fast/Hyst	USBH2_DATA7/SUSPEND	B					PA2	PUEN						PA2
K20	NVDD7	Fast/Hyst	USBOTG_DATA3/RXDP	B					PC13	PUEN	SLCDC1_DAT15					PC13
K23	NVDD7	Fast/Hyst	USBH2_DIR/TXDM	I					PA1	PUEN						PA1
K24	NVDD7	Fast/Hyst	USBOTG_CLK/TXDM	I					PE24	PUEN						PE24

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
L20	NVDD7	Fast/Hyst	USBH2_NXT/TXDM	I					PA3	PUEN						PA3
L23	NVDD7	Fast/Hyst	USBOTG_STP/TXDM	O		KP_ROW6A	B		PE1	PUEN						PE1
M20	NVDD7	Fast/Hyst	USBOTG_NXT/TXDM	I		KP_COL6A	B	ODEN	PE0	PUEN						PE0
M22	NVDD7	Fast/Hyst	USBOTG_DATA7/SUSPEND	B					PE25	PUEN						PE25
N20	NVDD7	Fast/Hyst	USBOTG_DIR/TXDM	I		KP_ROW7A	B		PE2	PUEN						PE2
NVDD7	NVDD7	Supply	NVDD7	static												NVDD7
A19	NVDD8	Slow/Hyst	RTCK	O		OWIRE	B	OD	PE16	PUEN						RTCK
A20	NVDD8	Slow/Hyst	SD1_D0	B		CSPI3_MISO	I		PE18	PUEN		PC_TEST_AHBST1				PE18
A21	NVDD8	Slow/Hyst	SD1_CMD	B		CSPI3_MOSI	O		PE22	PUEN		PC_TEST_INT_ERR				PE22
A22	NVDD8	Slow/Hyst	CSPI1_MISO	B					PD30	PUEN						PD30

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
B18	NVDD8	Slow/Hyst	TDI	I	PU/100k											TDI
B19	NVDD8	Slow/Hyst	TMS	I	PU/100k											TMS
B20	NVDD8	Slow/Hyst	SD1_D2	B					PE20	PUEN		PC_TEST_CARDST1				PE20
B21	NVDD8	Slow/Hyst	CSPI1_RDY/Ext_DMAREQ_B	I					PD25	PUEN						PD25
B22	NVDD8	Slow/Hyst	CSPI1_SS0	B					PD28	PUEN						PD28
C17	NVDD8	Slow/Hyst	TRST_B	I	PU/100k											TRST_B
C20	NVDD8	Slow/Hyst	CSPI1_SS1	B					PD27	PUEN	Ext_DMAGrant_B		Ext_DMAGrant_B			PD27
C21	NVDD8	Slow/Hyst	CSPI1_MOSI	B					PD31	PUEN						PD31
E16	NVDD8	Slow/Hyst	TDO	O												TDO
E17	NVDD8	Slow/Hyst	SD1_D1	B					PE19	PUEN		PC_TEST_CARDST0				PE19

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							Default
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	
E18	NVDD8	Slow/Hyst	SD1_D3	B	PU/PD/100k	CSPI3_SS	O	PU/PD	PE21	PDEN		PC_TEST_CARDST2				PE21
F17	NVDD8	Slow/Hyst	TCK	I	PU/100k											TCK
F18	NVDD8	Slow/Hyst	CSPI1_SCLK	B					PD29	PUEN						PD29
G15	NVDD8	Supply	NVDD8	static												NVDD8
G17	NVDD8	Slow/Hyst	SD1_CLK	O		CSPI3_SCLK	O		PE23	PUEN		PC_TEST_INT_ALL				PE23
GND	NVDD8	Supply	NVSS8	static												NVSS8
A14	NVDD9	Slow2/Hyst	UART2_RTS	I		KP_ROW7	B		PE4	PUEN						UART2_RTS
A15	NVDD9	Slow/Hyst	KP_COL2	B	PU/100k/ODEN											KP_COL2
A16	NVDD9	Slow/Hyst	UART2_TXD	O		KP_COL6	B	ODEN	PE6	PUEN						UART2_TXD
A17	NVDD9	Slow/Hyst	UART3_CTS	O					PE10	PUEN						PE10
A18	NVDD9	Slow/Hyst	UART1_CTS	O					PE14	PUEN						UART1_CTS
B13	NVDD9	Slow/Hyst	I2C_CLK	B	OD				PD18	PUEN						PD18

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	Default
B14	NVDD9	Slow/Hyst	KP_COL0	B	PU/100k/ ODEN											KP_COL0
B15	NVDD9	Slow/Hyst	KP_COL4	B	PU/100k/ ODEN											KP_COL4
B16	NVDD9	Slow/Hyst	UART3_TX D	O					PE8	PUEN						PE8
B17	NVDD9	Slow/Hyst	UART1_TX D	O					PE12	PUEN						UART1_T XD
C13	NVDD9	Slow/Hyst	PWMO	O					PE5	PUEN	PC_S PKOU T	TOUT 2	TOUT 3			PE5
C16	NVDD9	Slow/Hyst	UART1_RT S	I					PE15	PUEN						UART1_R TS
E12	NVDD9	Slow/Hyst	UART2_CT S	O		KP_COL7	B	ODEN	PE3	PUEN						UART2_C TS
E13	NVDD9	Slow/Hyst	KP_COL3	B	PU/100k/ ODEN											KP_COL3
E14	NVDD9	Slow/Hyst	UART2_RX D	I		KP_ROW6	B		PE7	PUEN						UART2_R XD
E15	NVDD9	Slow/Hyst	UART3_RT S	I					PE11	PUEN						PE11
F12	NVDD9	Slow/Hyst	I2C_DATA	B	OD				PD17	PUEN						PD17
F13	NVDD9	Slow/Hyst	KP_COL1	B	PU/100k/ ODEN											KP_COL1

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	Default
F14	NVDD9	Slow/Hyst	KP_COL5	B	PU/100k/ ODEN											KP_COL5
F15	NVDD9	Slow/Hyst	UART3_RX D	I					PE9	PUEN						PE9
F16	NVDD9	Slow/Hyst	UART1_RX D	I					PE13	PUEN						UART1_R XD
G14	NVDD9	Supply	NVDD9	static												NVDD9
GND	NVDD9	Supply	NVSS9	static												NVSS9
AA23	OSC26VDD	Supply	OSC26VD D	static												OSC26V DD
AA24	OSC26VDD	Analog_By p	XTAL26M	static												XTAL26M
AB23	OSC26VDD	Supply	OSC26VS S	static												OSC26V SS
AB24	OSC26VDD	Analog_By p	EXTAL26M	static												EXTAL26 M
V19	OSC26VDD	Analog	OSC26M_ TEST	I												OSC26M _TEST
M23	OSC32VDD	Supply	OSC32VD D	static												OSC32V DD
M24	OSC32VDD	Analog	EXTAL32K	I												EXTAL32 K
N23	OSC32VDD	Supply	OSC32VS S	static												OSC32V SS

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	Default
N24	OSC32VDD	Analog	XTAL32K	I												XTAL32K
GND	QVDD10	Supply	QVSS10	static												QVSS10
QVDD	QVDD10	Supply	QVDD10	static												QVDD10
GND	QVDD12	Supply	QVSS12	static												QVSS12
QVDD	QVDD12	Supply	QVDD12	static												QVDD12
GND	QVDD2	Supply	QVSS2	static												QVSS2
QVDD	QVDD2	Supply	QVDD2	static												QVDD2
GND	QVDD3	Supply	QVSS3	static												QVSS3
QVDD	QVDD3	Supply	QVDD3	static												QVDD3
GND	QVDD5	Supply	QVSS5	static												QVSS5
QVDD	QVDD5	Supply	QVDD5	static												QVDD5
GND	QVDD6	Supply	QVSS6	static												QVSS6
QVDD	QVDD6	Supply	QVDD6	static												QVDD6
GND	QVDD7	Supply	QVSS7	static												QVSS7
QVDD	QVDD7	Supply	QVDD7	static												QVDD7
GND	QVDD8	Supply	QVSS8	static												QVSS8

Table 5-2. i.MX27 Pin MUX Table (continued)

Ball map Location	Power Bank	I/O Type	Primary			Alternate			GPIO							
			Signal/Pad Name	Direction[1]	Pull-up/Pull Strength/Open Drain[2]	Signal	Direction[2]	Pull-up /Pull Strength /Open Drain	Mux[3]	PUEN/PDEN	AIN	BIN	CIN	AOUT	BOUT	Default
QVDD	QVDD8	Supply	QVDD8	static												QVDD8
K18	RTCVDD	Supply	RTCVSS	static												RTCVSS
K19	RTCVDD	Supply	RTCVDD	static												RTCVDD
J18	UPLLVDD	Supply	UPLLVDD	static												UPLLVDD
M15	UPLLVDD	Supply	UPLLVSS	static												UPLLVSS

1 Indicates direction of primary signal. It may not indicate the direction of the pin as it may be dependent on other functions.

2 **KP** = Keeper Circuit permanently On when in Primary/Alternate Mode; **PU** = Pull Up permanently On when in Primary/Alternate Mode; **PUEN** = Pull Up controllable from Module when in Primary/Alternate Mode; **OD** = Open Drain permanently On when in Primary/Alternate Mode; **ODEN** = Open Drain controllable from Module when in Primary/Alternate Mode.

3 Pin mux with GPIO has its Pull Up controlled by the GPIO PUEN register (in Primary, Alternate, or GPIO Mode)

5.4 Package Pin Assignments

Table 5-3 through Table 5-6 identifies the i.MX27 full package MAPBGA pin assignments. The connection of these pins depends solely upon the user application, however there are a few factory test signals that are not used in normal applications. Following is a list of these signals and how they are to be terminated for proper operation of the i.MX27 processor:

- CLKMODE[1:0]: To ensure proper operation, leave these signals as no connects.
- OSC26M_TEST: To ensure proper operation, leave this signal as no connect.
- EXT_48M: To ensure proper operation, connect this signal to ground.
- EXT_266M: To ensure proper operation, connect this signal to ground.

Table 5-3. i.MX27 Full Package MAPBGA Pin Assignment (1 of 4)

	1	2	3	4	5	6	7	8
A	GND	GND	SD2_D3_MSHC_DATA3_PB7_PAD	SD2_CLK_MSHC_SCLK_PB9_PAD	CSI_D3_UART6_RTS_PB13_PAD	CSI_D5_PB17_PAD	CSI_HSYNC_UART5_RTS_PB21_PAD	SSI4_RXDAT_PC17_PAD
B	GND	GND	SPL_SPR_PA27_PAD	CSI_D1_UART6_RXD_PB11_PAD	CSI_MCLK_PB15_PAD	CSI_D7_UART5_RXD_PB19_PAD	TIN_PC15_PAD	SSI4_CLK_PC19_PAD
C	SD2_D0_MSHC_DATA0_PB4_PAD	CONTRAST_PA30_PAD		CSI_D0_UART6_TXD_PB10_PAD	SD2_CMD_MSHC_BS_PB8_PAD			SD2_D2_MSHC_DATA2_PB6_PAD
D	HSYNC_PA28_PAD	PS_PA26_PAD	OE_ACD_PA31_PAD					
E	REV_PA24_PAD	LD16_PA22_PAD	SD2_D1_MSHC_DATA1_PB5_PAD			CSI_D2_UART6_CTS_PB12_PAD	CSI_PIXCLK_PB16_PAD	TOUT_PC14_PAD
F	LD14_PA20_PAD	LD10_PA16_PAD			VSYNC_PA29_PAD	CSI_D4_PB14_PAD	CSI_D6_UART5_TXD_PB18_PAD	SSI4_FS_PC16_PAD
G	LD8_PA14_PAD	LD6_PA12_PAD			LD17_PA23_PAD	CLS_PA25_PAD	CSI_VSYNC_UART5_CTS_PB20_PAD	SSI4_TXDAT_PC18_PAD
H	NFRB_ETMTRACEPKT3_PF0	LD4_PA10_PAD	LD12_PA18_PAD		LD13_PA19_PAD	LD15_PA21_PAD	NVDD15	
J	NFWP_B_ETMTRACEPKT1_PF2	LD0_PA6_PAD	LD2_PA8_PAD		LD7_PA13_PAD	LD5_PA11_PAD	LD11_PA17_PAD	
K	NFALE_ETMPIPESTAT0_PF4	LSCLK_PA5_PAD			LD3_PA9_PAD	LD1_PA7_PAD	LD9_PA15_PAD	
L	NFWE_B_ETMPIPESTAT2_PF6	NFCE_B_ETMTRACEPKT2_PF3			NFRE_B_ETMPIPESTAT1_PF5	NFCLE_ETMTRACEPKT0_PF1	NVDD12	
M	D14_PAD	D15_PAD	D11_PAD		D13_PAD	D9_PAD	NVDD1	

Table 5-4. i.MX27 Full Package MAPBGA Pin Assignment (2 of 4)

13	14	15	16	17	18	19	20	21	22	23	24	
KP_ROW5_PAD	UART2_RTS_KP_ROW7_PE4_PAD	KP_COL2_PAD	UART2_TXD_KP_COL6_PE6_PAD	UART3_CTS_PE10_PAD	UART1_CTS_PE14_PAD	RTCK_OWIRE_PE16_PAD	SD1_D0_CSPI3_MISO_PE18_PAD	SD1_CMD_CSPI3_MOSI_PE22_PAD	CSPI1_MISO_PD30_PAD	GND	GND	A
I2C_CLK_PD18_PAD	KP_COL0_PAD	KP_COL4_PAD	UART3_TXD_PE8_PAD	UART1_TXD_PE12_PAD	TDI_PAD	TMS_PAD	SD1_D2_PE20_PAD	CSPI1_RDY_PD25_PAD	CSPI1_SS0_PD28_PAD	GND	GND	B
PWMO_PE5_PAD			UART1_RTS_PE15_PAD	TRST_B_PAD			CSPI1_SS1_PD27_PAD	CSPI1_MOSI_PD31_PAD		CSPI2_SS1_USBH2_DATA3_PD20	USBHL_OE_B_PB27_PAD	C
									CSPI2_SS2_USBH2_DATA4_PD19	CSPI2_SCLK_USBH2_DATA0_PD22	USBHL_TXDP_UART4_CTS_PB29	D
KP_COL3_PAD	UART2_RXD_KP_ROW6_PE7_PAD	UART3_RTS_PE11_PAD	TDO_PAD	SD1_D1_PE19_PAD	SD1_D3_CSPI3_SS_PE21_PAD	USBHL_FS_UART4_RTS_PB26_PAD			CSPI1_SS2_USBH2_DATA5_PD26	CSPI2_MOSI_USBH2_DATA1_PD24	USBHL_RXDP_UART4_RXD_PB31	E
KP_COL1_PAD	KP_COL5_PAD	UART3_RXD_PE9_PAD	UART1_RXD_PE13_PAD	TCK_PAD	CSPI1_SCLK_PD29_PAD	USBHL_TXDM_UART4_TXD_PB28	CSPI2_SS0_USBH2_DATA6_PD21			USB_PWR_PB23_PAD	I2C2_SCL_PC6_PAD	F
QVDD	NVDD9	NVDD8	QVDD	SD1_CLK_CSPI3_SCLK_PE23_PAD	USBOTG_DATA1_PC11_PAD	USBHL_SUSP_PB22_PAD	CSPI2_MISO_USBH2_DATA2_PD23			USBOTG_DATA2_PC10_PAD	USBOTG_DATA6_PC8_PAD	G
					NVDD14	USBOTG_DATA5_PC7_PAD	USB_OC_B_PB24_PAD		USBHL_RCV_PB25_PAD	USBH2_CLK_PA0_PAD	USBOTG_DATA4_PC12_PAD	H
					UPLLVDD_PAD	USBOTG_DATA0_PC9_PAD	USBHL_RXDM_PB30_PAD		I2C2_SDA_PC5_PAD	USBH2_STP_PA4_PAD	USBH2_DATA7_PA2_PAD	J
GND	GND	GND			RTCVSS_PAD	RTCVDD_PAD	USBOTG_DATA3_PC13_PAD			USBH2_DIR_PA1_PAD	USBOTG_CLK_PE24_PAD	K
GND	GND	GND			NVDD7	NVDD7	USBH2_NXT_PA3_PAD			USBOTG_STP_KP_ROW6A_PE1_PAD	OSC32K_BYPASS_PAD	L
GND	GND	UPLLVSS_PAD			FPMVDD_PAD	NVDD13	USBOTG_NXT_KP_COL6A_PE0_PAD		USBOTG_DATA7_PE25_PAD	OSC32VDD_PAD	EXTAL32K_PAD	M

Table 5-5. i.MX27 Full Package MAPBGA Pin Assignment (3 of 4)

N	D12_PAD	D7_PAD	D5_PAD		D3_PAD	D1_PAD	NVDD1			GND	GND	GND
P	D10_PAD	D8_PAD			A9_PAD	A12_PAD	QVDD			GND	GND	GND
R	D6_PAD	D4_PAD			A5_PAD	A7_PAD	NVDD2			GND	GND	GND
T	D2_PAD	D0_PAD	MA10_PAD		SDBA1_PAD	A1_PAD	NVDD2					
U	A13_PAD	A11_PAD	A3_PAD		SD31_PAD	A25_PAD	NVDD2					
V	A8_PAD	A6_PAD			SD26_PAD	SD28_PAD	SD29_PAD	A19_PAD	NVDD2	NVDD2	NVDD3	NVDD3
W	A4_PAD	A2_PAD			SD23_PAD	SDQS2_PAD	SD25_PAD	SDQS1_PAD	SD13_PAD	SD6_PAD	A16_PAD	DQM1_PAD
Y	A0_PAD	SDBA0_PAD	SDQS3_PAD			A20_PAD	SD18_PAD	SD15_PAD	SD12_PAD	SD8_PAD	A15_PAD	SD2_PAD
AA	SD30_PAD	A24_PAD	SD27_PAD									
AB	A23_PAD	SD24_PAD		A21_PAD	SD21_PAD			SD10_PAD	A14_PAD			SD0_PAD
AC	GND	GND	A22_PAD	SD20_PAD	SD17_PAD	A18_PAD	A17_PAD	SD9_PAD	SD5_PAD	SD4_PAD	SD1_PAD	A10_PAD
AD	GND	GND	SD22_PAD	SD19_PAD	SD16_PAD	SD14_PAD	SD11_PAD	SD7_PAD	SDQS0_PAD	SD3_PAD	DQM3_PAD	DQM0_PAD
	1	2	3	4	5	6	7	8	9	10	11	12

Table 5-6. i.MX27 Full Package MAPBGA Pin Assignment (4 of 4)

GND	GND	GND			NVDD6	POWER_ON_RESET_PAD	USBOTG_DIR_KP_ROW7A_PE2_PAD		POWER_CUT_PAD	OSC32VSS_PAD	XTAL32K_PAD	N
GND	GND	FPMVSS_PAD			NVDD6	ATA_DATA6_FEC_MDIO_PD8_PAD	ATA_DATA2_SD3_D2_PD4_PAD			SD8_CMD_PD0_PAD	SD8_CLK_ETMTRACEPKT15_PD1	P
GND	GND	MPLLVSS_PAD			FUSEVDD_PAD	FUSEVSS_PAD	ATA_DATA10_ETMTRACEPKT9_PD12			ATA_DATA0_SD3_D0_PD2_PAD	ATA_DATA1_SD3_D1_PD3_PAD	R
					MPLLVDD_PAD	AVSS_PAD	ATA_DATA14_ETMTRACEPKT5_PD16		ATA_DATA4_ETMTRACEPKT14_PD6	ATA_DATA5_ETMTRACEPKT13_PD7	ATA_DATA3_SD3_D3_PD5_PAD	T
					AVDD_PAD	BOOT2_PAD	ICS16_ATA_INTRQ_PF9_PAD		ATA_DATA8_ETMTRACEPKT11_PD10	ATA_DATA12_ETMTRACEPKT7_PD14	ATA_DATA7_ETMTRACEPKT12_PD9	U
NVDD4	QVDD	QVDD	QVDD	NVDD5	NVDD5	OSC26M_TEST_PAD	PC_POE_ATA_BUFFER_EN_PF7_PAD			BOOT0_PAD	ATA_DATA9_ETMTRACEPKT10_PD11	V
SDWE_B_PAD	CS3_B_PAD	CS5_B_ETMTRACECLK_PF22_PAD	EB0_B_PAD	EXT_60M_PAD	PC_CD1_B_ATA_DIOR_PF20_PAD	PC_VS2_ATA_DA0_PF13_PAD	PC_BVD2_ATA_DMACK_PF11_PAD			ATA_DATA11_ETMTRACEPKT8_PD13	ATA_DATA13_ETMTRACEPKT6_PD15	W
DQM2_PAD	SDCKE1_PAD	CS2_B_PAD	LBA_B_PAD	OE_B_PAD	PC_WAIT_B_ATA_CS1_PF18_PAD	PC_PWRON_ATA_DA2_PF16_PAD			BOOT3_PAD	BOOT1_PAD	ATA_DATA15_ETMTRACEPKT4_PF23	Y
									RESET_OUT_B_PE17_PAD	OSC26VDD_PAD	XTAL26M_PAD	AA
RAS_B_PAD			CS1_B_PAD	BCLK_PAD			CLKMODE0_PAD	CLKMODE1_PAD		OSC26VSS_PAD	EXTAL26M_PAD	AB
CAS_B_PAD	SDCKE0_PAD	RW_B_PAD	EB3_B_PAD	EB1_B_PAD	JTAG_CTRL_PAD	PC_CD2_B_ATA_DIOW_PF19_PAD	PC_VS1_ATA_DA1_PF14_PAD	PC_RST_ATA_RESET_B_PF10_PAD	RESET_IN_B_PAD	GND	GND	AC
SDCLK_PAD	SDCLK_PAD_B	CS4_B_ETMTRACESYNC_PF21_PAD	CS0_B_PAD	CLKO_PF15_PAD	EXT_266M_PAD	PC_READY_ATA_CS0_PF17_PAD	PC_BVD1_ATA_DMARQ_PF12_PAD	PC_RW_B_ATA_IORDY_PF8_PAD	POR_B_PAD	GND	GND	AD
13	14	15	16	17	18	19	20	21	22	23	24	

Chapter 6
General-Purpose I/O (GPIO)

6.1 Introduction

The GPIO module in i.MX27 processor provides six general purpose I/O (GPIO) ports (PA, PB, PC, PD, PE, and PF). Each GPIO port is a 32-bit port that may be multiplexed with one or more dedicated functions.

This chapter contains the description of the top level i.MX27 I/O multiplexing strategy that consists of two parts:

- Software controllable multiplexing done in the GPIO module
- Hardware multiplexing done by the IOMUX module

The I/O multiplexing strategy is designed to configure the inputs and outputs of the BONO Device chip in different modes. It allows a user to use the same I/O pad for alternative purposes of the chip. The design of I/O multiplexer is targeted to be as flexible as possible. Refer to Chapter 5, “Signal Descriptions and Pin Assignments” for detailed I/O multiplexing information.

Figure 6-1 shows the block diagram of the GPIO and IOMUX modules’ partition at the top level of the BONO processor. Figure 6-2 shows a block diagram of an individual port of the GPIO module.

NOTE

A_IN, B_IN, C_IN, A_OUT, and B_OUT are internal signals and do not represent individual port signals.

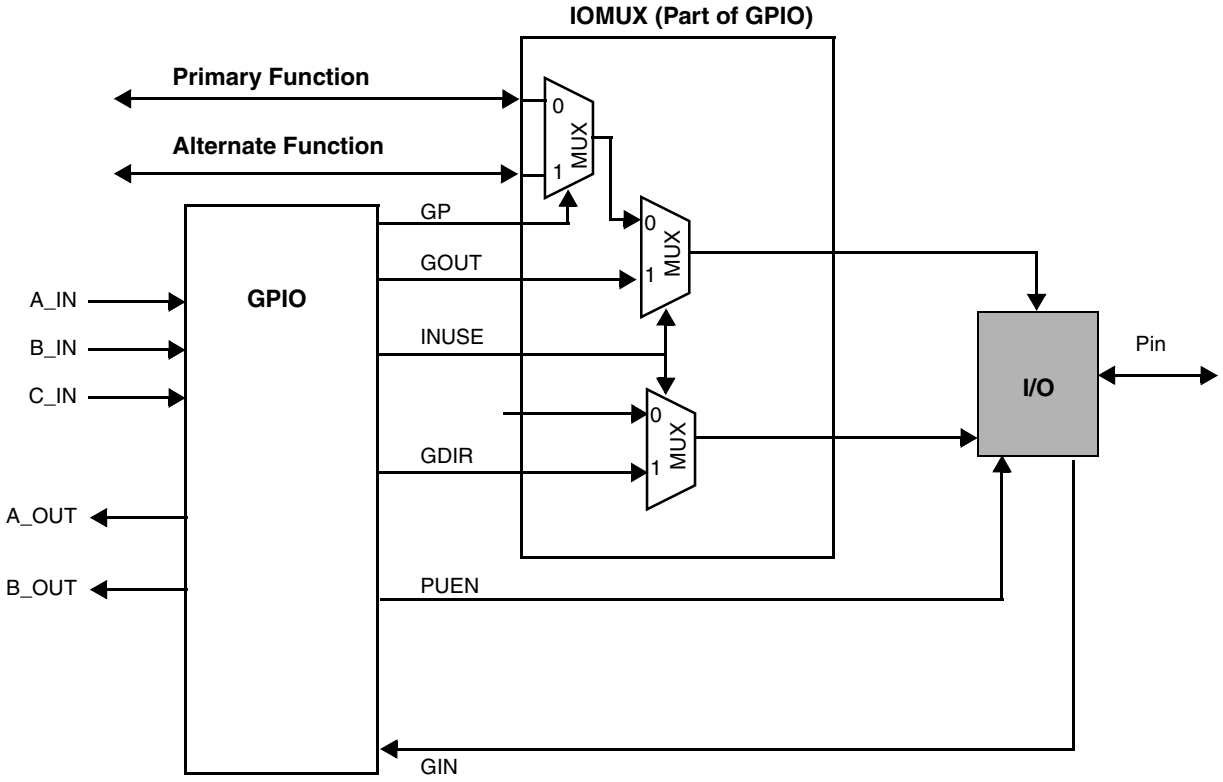


Figure 6-1. Functional Block Diagram of GPIO

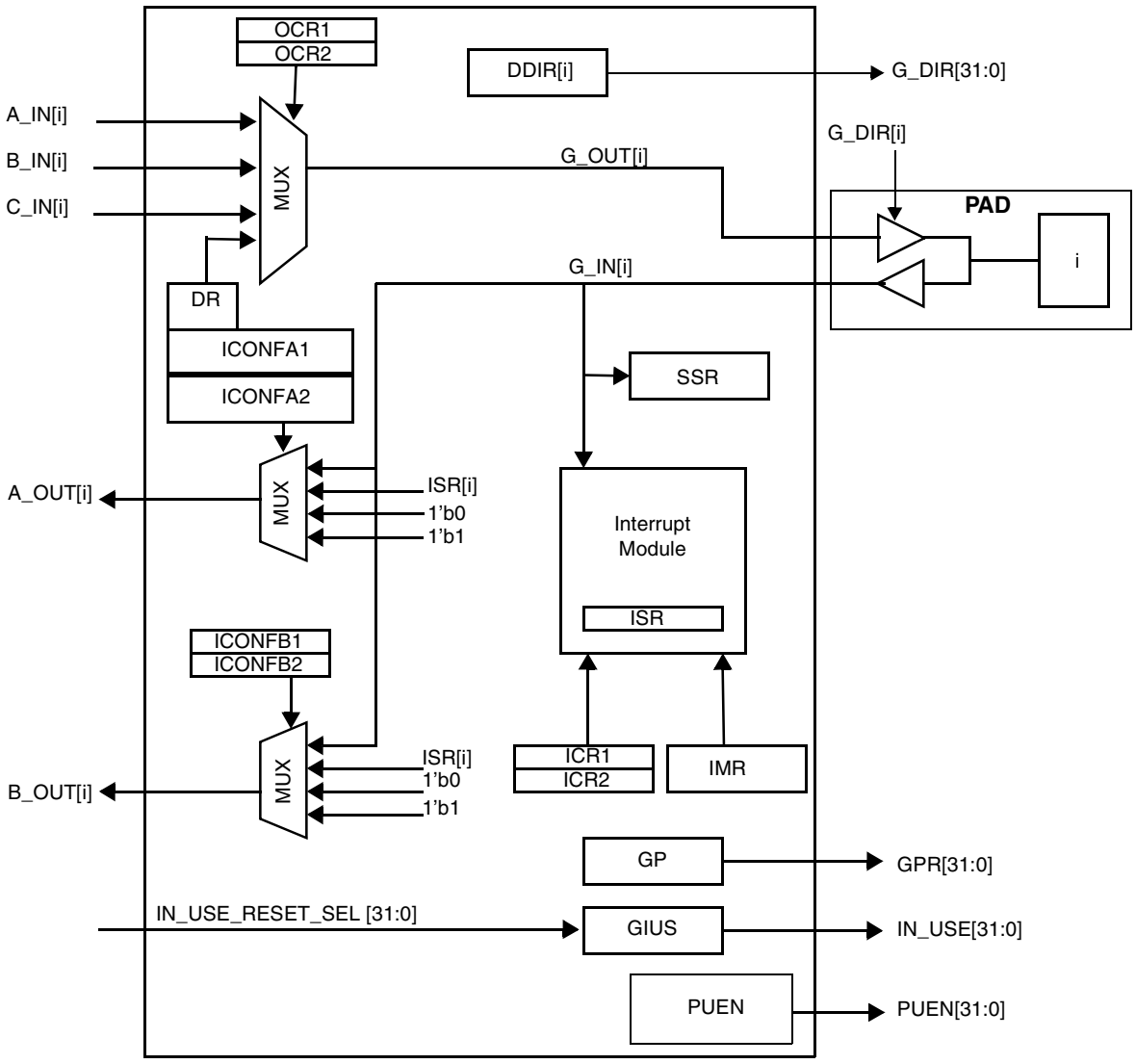


Figure 6-2. GPIO Block Diagram for an Individual Port

6.2 Overview

The GPIO module provides General Purpose I/O capability to the device. Each I/O port can be programmed as either a general purpose input or general purpose output. In addition to GPIO functionality, pins can be changed from their default dedicated functions to alternate functions. Input and output signals of peripherals are connected to the IOMUX module at the dedicated (primary) or alternate inputs. In the output direction, one out of three alternate sources (originating from peripherals) can be selected. From the input direction, one out of two alternate destinations (input to a peripheral) can be selected.

6.3 GPIO Features

The following list contains the GPIO features:

- Six 32-bit ports, each with direction-configurable pins

- Software control for input/output pin configuration through 32-bit direction register
- Software control for multiplexing one out of four different sources for every output. Three of them are functional pins from internal modules while the fourth is from the data register of the module.
- Software control for routing of every input to two different destinations
- Input data can be sampled to the data register.
- Inputs can be internally tied to a logic 1 or 0 to ensure any transitions attempted to be processed are ignored.
- One 32-bit general purpose register is dedicated to each GPIO port. These registers may be used for software control of IOMUX block of the GPIO.
- Every input is configurable as an interrupt and each interrupt can be defined as either:
  - Rising-edge triggered
  - Falling-edge triggered
  - Level sensitive
- The interrupts can be masked using a 32-bit mask register.
- Two levels of interrupt masking are provided. Interrupts can be individually masked at the bit level or at the port level.
- Software reset function: when the SWR bit (SWR register, 0 bit) is written as a 1, the entire GPIO module is reset immediately, and this reset signal is asserted for three system cycles. After this, the reset signal will be released automatically.

6.4 External Signals Description

Refer to Chapter 5, “Signal Descriptions and Pin Assignments” for details on the I/O multiplexing scheme and external connection to the GPIO module.

6.5 Interrupts

Every external input passes through the interrupt module in the GPIO module. Inside this module, the interrupts may be defined as rising-edge triggered, or falling-edge triggered. Each interrupt can be masked and also be designated as a high-level interrupt, or a low-level sensitive interrupt. The interrupt status register bits corresponding to the interrupts waiting for service are stored as a value of 1. The interrupt status register is Write 1 to Clear (w1c). The user is responsible for clearing the interrupt status register bit after it has been serviced.

6.6 Memory Map and Register Definitions

The GPIO module has six ports and each port has 17 registers. In total, the GPIO has 102 registers. The registers, other than the Sample Status Register (SSR) and the Interrupt Status Register (ISR), have both read and write capability. The Sample Status Register is a read only register, while the Interrupt Status Register is a w1c register; the register can be read, but writing a 1 to any register bit clears the bit. Writing a value of 0 to the bit has no effect.

While there are six GPIO ports, each capable of representing 32 GPIO configurable pins as inputs or outputs, not all bits are mapped to a pin and hence these bits do not have any effect and are marked as reserved. These reserved bits are indicated in Section 6.6.10, "GPIO IN USE Registers (GIUS)."

Table 6-1 shows the GPIO memory map.

Table 6-1. GPIO Memory Map

Address	Register	Access	Reset Value	Section/Page
General Registers				
0x1001_5000 (PTA_DDIR)	Data Direction Register	R/W	0x0000_0000	6.6.2/6-11
0x1001_5100 (PTB_DDIR)	Data Direction Register	R/W	0x0000_0000	6.6.2/6-11
0x1001_5200 (PTC_DDIR)	Data Direction Register	R/W	0x0000_0000	6.6.2/6-11
0x1001_5300 (PTD_DDIR)	Data Direction Register	R/W	0x0000_0000	6.6.2/6-11
0x1001_5400 (PTE_DDIR)	Data Direction Register	R/W	0x0000_0000	6.6.2/6-11
0x1001_5500 (PTF_DDIR)	Data Direction Register	R/W	0x0000_0000	6.6.2/6-11
0x1001_5004 (PTA_OCR1)	Output Configuration Register 1)	R/W	0x0000_0000	6.6.3/6-11
0x1001_5104 (PTB_OCR1)	Output Configuration Register 1	R/W	0x0000_0000	6.6.3/6-11
0x1001_5204 (PTC_OCR1)	Output Configuration Register 1)	R/W	0x0000_0000	6.6.3/6-11
0x1001_5304 (PTD_OCR1)	Output Configuration Register 1	R/W	0x0000_0000	6.6.3/6-11
0x1001_5404 (PTE_OCR1)	Output Configuration Register 1	R/W	0x0000_0000	6.6.3/6-11
0x1001_5504 (PTF_OCR1)	Output Configuration Register 1	R/W	0x0000_0000	6.6.3/6-11
0x1001_5008 (PTA_OCR2)	Output Configuration Register 2	R/W	0x0000_0000	6.6.4/6-12
0x1001_5108 (PTB_OCR2)	Output Configuration Register 2	R/W	0x0000_0000	6.6.4/6-12
0x1001_5208 (PTC_OCR2)	Output Configuration Register 2	R/W	0x0000_0000	6.6.4/6-12
0x1001_5308 (PTD_OCR2)	Output Configuration Register 2	R/W	0x0000_0000	6.6.4/6-12
0x1001_5408 (PTE_OCR2)	Output Configuration Register 2	R/W	0x0000_0000	6.6.4/6-12

Table 6-1. GPIO Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x1001_5508 (PTF_OCR2)	Output Configuration Register 2	R/W	0x0000_0000	6.6.4/6-12
0x1001_500C (PTA_ICONFA1)	Input Configuration Register A1	R/W	0xFFFF_FFFF	6.6.5/6-13
0x1001_510C (PTB_ICONFA1)	Input Configuration Register A1	R/W	0xFFFF_FFFF	6.6.5/6-13
0x1001_520C (PTC_ICONFA1)	Input Configuration Register A1	R/W	0xFFFF_FFFF	6.6.5/6-13
0x1001_530C (PTD_ICONFA1)	Input Configuration Register A1	R/W	0xFFFF_FFFF	6.6.5/6-13
0x1001_540C (PTE_ICONFA1)	Input Configuration Register A1	R/W	0xFFFF_FFFF	6.6.5/6-13
0x1001_550C (PTF_ICONFA1)	Input Configuration Register A1	R/W	0xFFFF_FFFF	6.6.5/6-13
0x1001_5010 (PTA_ICONFA2)	Input Configuration Register A2	R/W	0xFFFF_FFFF	6.6.6/6-14
0x1001_5110 (PTB_ICONFA2)	Input Configuration Register A2	R/W	0xFFFF_FFFF	6.6.6/6-14
0x1001_5210 (PTC_ICONFA2)	Input Configuration Register A2	R/W	0xFFFF_FFFF	6.6.6/6-14
0x1001_5310 (PTD_ICONFA2)	Input Configuration Register A2	R/W	0xFFFF_FFFF	6.6.6/6-14
0x1001_5410 (PTE_ICONFA2)	Input Configuration Register A2	R/W	0xFFFF_FFFF	6.6.6/6-14
0x1001_5510 (PTF_ICONFA2)	Input Configuration Register A2	R/W	0xFFFF_FFFF	6.6.6/6-14
0x1001_5014 (PTA_ICONFB1)	Input Configuration Register B1	R/W	0xFFFF_FFFF	6.6.7/6-15
0x1001_5114 (PTB_ICONFB1)	Input Configuration Register B1	R/W	0xFFFF_FFFF	6.6.7/6-15
0x1001_5214 (PTC_ICONFB1)	Input Configuration Register B1	R/W	0xFFFF_FFFF	6.6.7/6-15
0x1001_5314 (PTD_ICONFB1)	Input Configuration Register B1	R/W	0xFFFF_FFFF	6.6.7/6-15
0x1001_5414 (PTE_ICONFB1)	Input Configuration Register B1	R/W	0xFFFF_FFFF	6.6.7/6-15
0x1001_5514 (PTF_ICONFB1)	Input Configuration Register B1	R/W	0xFFFF_FFFF	6.6.7/6-15
0x1001_5018 (PTA_ICONFB2)	Input Configuration Register B2	R/W	0xFFFF_FFFF	6.6.8/6-16

Table 6-1. GPIO Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x1001_5118 (PTB_ICONFB2)	Input Configuration Register B2	R/W	0xFFFF_FFFF	6.6.8/6-16
0x1001_5218 (PTC_ICONFB2)	Input Configuration Register B2	R/W	0xFFFF_FFFF	6.6.8/6-16
0x1001_5318 (PTD_ICONFB2)	Input Configuration Register B2	R/W	0xFFFF_FFFF	6.6.8/6-16
0x1001_5418 (PTE_ICONFB2)	Input Configuration Register B2	R/W	0xFFFF_FFFF	6.6.8/6-16
0x1001_5518 (PTF_ICONFB2)	Input Configuration Register B2	R/W	0xFFFF_FFFF	6.6.8/6-16
0x1001_501C (PTA_DR)	Data Register	R/W	0x0000_0000	6.6.9/6-17
0x1001_511C (PTB_DR)	Data Register	R/W	0x0000_0000	6.6.9/6-17
0x1001_521C (PTC_DR)	Data Register	R/W	0x0000_0000	6.6.9/6-17
0x1001_531C (PTD_DR)	Data Register	R/W	0x0000_0000	6.6.9/6-17
0x1001_541C (PTE_DR)	Data Register	R/W	0x0000_0000	6.6.9/6-17
0x1001_551C (PTF_DR)	Data Register	R/W	0x0000_0000	6.6.9/6-17
0x1001_5020 (PTA_GIUS)	GPIO In Use Register A	R/W	0xFFFF_FFFF	6.6.11/6-19
0x1001_5120 (PTB_GIUS)	GPIO In Use Register B	R/W	0xFF3F_FFF3	6.6.11/6-19
0x1001_5220 (PTC_GIUS)	GPIO In Use Register C	R/W	0xFFFF_FFFF	6.6.11/6-19
0x1001_5320 (PTD_GIUS)	GPIO In Use Register D	R/W	0xFFFE_0000	6.6.11/6-19
0x1001_5420 (PTE_GIUS)	GPIO In Use Register E	R/W	0xFFFC_0F27	6.6.11/6-19
0x1001_5520 (PTF_GIUS)	GPIO In Use Register F	R/W	0xFF00_0000	6.6.11/6-19
0x1001_5024 (PTA_SSR)	Sample Status Register	R	0x0000_0000	6.6.12/6-22
0x1001_5124 (PTB_SSR)	Sample Status Register	R	0x0000_0000	6.6.12/6-22
0x1001_5224 (PTC_SSR)	Sample Status Register	R	0x0000_0000	6.6.12/6-22

Table 6-1. GPIO Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x1001_5324 (PTD_SSR)	Sample Status Register	R	0x0000_0000	6.6.12/6-22
0x1001_5424 (PTE_SSR)	Sample Status Register	R	0x0000_0000	6.6.12/6-22
0x1001_5524 (PTF_SSR)	Sample Status Register	R	0x0000_0000	6.6.12/6-22
0x1001_5028 (PTA_ICR1)	Interrupt Configuration Register 1	R/W	0x0000_0000	6.6.13/6-23
0x1001_5128 (PTB_ICR1)	Interrupt Configuration Register 1	R/W	0x0000_0000	6.6.13/6-23
0x1001_5228 (PTC_ICR1)	Interrupt Configuration Register 1	R/W	0x0000_0000	6.6.13/6-23
0x1001_5328 (PTD_ICR1)	Interrupt Configuration Register 1	R/W	0x0000_0000	6.6.13/6-23
0x1001_5428 (PTE_ICR1)	Interrupt Configuration Register 1	R/W	0x0000_0000	6.6.13/6-23
0x1001_5528 (PTF_ICR1)	Interrupt Configuration Register 1	R/W	0x0000_0000	6.6.13/6-23
0x1001_502C (PTA_ICR2)	Interrupt Configuration Register 2	R/W	0x0000_0000	6.6.14/6-24
0x1001_512C (PTB_ICR2)	Interrupt Configuration Register 2	R/W	0x0000_0000	6.6.14/6-24
0x1001_522C (PTC_ICR2)	Interrupt Configuration Register 2	R/W	0x0000_0000	6.6.14/6-24
0x1001_532C (PTD_ICR2)	Interrupt Configuration Register 2	R/W	0x0000_0000	6.6.14/6-24
0x1001_542C (PTE_ICR2)	Interrupt Configuration Register 2	R/W	0x0000_0000	6.6.14/6-24
0x1001_552C (PTF_ICR2)	Interrupt Configuration Register 2	R/W	0x0000_0000	6.6.14/6-24
0x1001_5030 (PTA_IMR)	Interrupt Mask Register	R/W	0x0000_0000	6.6.15/6-25
0x1001_5130 (PTB_IMR)	Interrupt Mask Register	R/W	0x0000_0000	6.6.15/6-25
0x1001_5230 (PTC_IMR)	Interrupt Mask Register	R/W	0x0000_0000	6.6.15/6-25
0x1001_5330 (PTD_IMR)	Interrupt Mask Register	R/W	0x0000_0000	6.6.15/6-25
0x1001_5430 (PTE_IMR)	Interrupt Mask Register	R/W	0x0000_0000	6.6.15/6-25

Table 6-1. GPIO Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x1001_5530 (PTF_IMR)	Interrupt Mask Register	R/W	0x0000_0000	6.6.15/6-25
0x1001_5034 (PTA_ISR)	Interrupt Status Register	R/W	0x0000_0000	6.6.16/6-26
0x1001_5134 (PTB_ISR)	Interrupt Status Register	R/W	0x0000_0000	6.6.16/6-26
0x1001_5234 (PTC_ISR)	Interrupt Status Register	R/W	0x0000_0000	6.6.16/6-26
0x1001_5334 (PTD_ISR)	Interrupt Status Register	R/W	0x0000_0000	6.6.16/6-26
0x1001_5434 (PTE_ISR)	Interrupt Status Register	R/W	0x0000_0000	6.6.16/6-26
0x1001_5534 (PTF_ISR)	Interrupt Status Register	R/W	0x0000_0000	6.6.16/6-26
0x1001_5038 (PTA_GPR)	General Purpose Register	R/W	0x0000_0000	6.6.17/6-27
0x1001_5138 (PTB_GPR)	General Purpose Register	R/W	0x0000_0000	6.6.17/6-27
0x1001_5238 (PTC_GPR)	General Purpose Register	R/W	0x0000_0000	6.6.17/6-27
0x1001_5338 (PTD_GPR)	General Purpose Register	R/W	0x0000_0000	6.6.17/6-27
0x1001_5438 (PTE_GPR)	General Purpose Register	R/W	0x0000_0000	6.6.17/6-27
0x1001_5538 (PTF_GPR)	General Purpose Register	R/W	0x0000_0000	6.6.17/6-27
0x1001_503C (PTA_SWR)	Software Reset Register	R	0x0000_0000	6.6.18/6-28
0x1001_513C (PTB_SWR)	Software Reset Register	R	0x0000_0000	6.6.18/6-28
0x1001_513C (PTB_SWR)	Software Reset Register	R	0x0000_0000	6.6.18/6-28
0x1001_533C (PTD_SWR)	Software Reset Register	R	0x0000_0000	6.6.18/6-28
0x1001_543C (PTE_SWR)	Software Reset Register	R	0x0000_0000	6.6.18/6-28
0x1001_553C (PTF_SWR)	Software Reset Register	R	0x0000_0000	6.6.18/6-28
0x1001_5040 (PTA_PUEN)	Pull-Up Enable Register	R/W	0xFFFF_FFFF	6.6.19/6-29

Table 6-1. GPIO Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x1001_5140 (PTB_PUEN)	Pull-Up Enable Register	R/W	0xFFFF_FFFF	6.6.19/6-29
0x1001_5240 (PTC_PUEN)	Pull-Up Enable Register	R/W	0xFFFF_FFFF	6.6.19/6-29
0x1001_5340 (PTD_PUEN)	Pull-Up Enable Register	R/W	0xFFFF_FFFF	6.6.19/6-29
0x1001_5440 (PTE_PUEN)	Pull-Up Enable Register	R/W	0xFFFF_FFFF	6.6.19/6-29
0x1001_5540 (PTF_PUEN)	Pull-Up Enable Register	R/W	0xFFFF_FFFF	6.6.19/6-29
0x1001_5600 (PMASK)	Port Interrupt Mask Register	R/W	0x0000_003F	6.6.20/6-30

6.6.1 Register Summary

The conventions in Figure 6-3 and Table 6-2 serve as a key for the register summary and individual register diagrams.

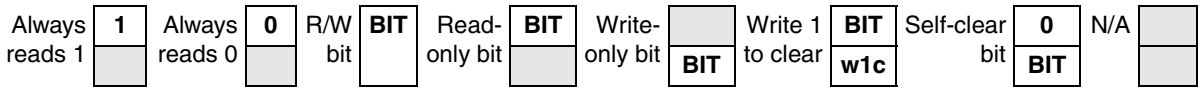


Figure 6-3. Key to Register Fields

Table 6-2 provides a key for register figures and tables and the register summary.

Table 6-2. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to zero.
1	Resets to one.

Table 6-2. Register Conventions (continued)

Convention	Description
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

6.6.2 Data Direction Register (PTn_DDIR)

The Data Direction registers determine whether each port pin operates as an input or an output pin. Figure 6-4 shows the register and Table 6-3 provides its field descriptions.

0x1001_5000 (PTA_DDIR)
0x1001_5100 (PTB_DDIR)
0x1001_5200 (PTC_DDIR)
0x1001_5300 (PTD_DDIR)
0x1001_5400 (PTE_DDIR)
0x1001_5500 (PTF_DDIR)

Access: User R/W

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

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

Figure 6-4. Data Direction Register (PTn_DDIR)

Table 6-3. Data Direction Register Field Descriptions

Field	Description
31–0 DDIR	Data Direction. This is a read/write register that defines the current direction of the 32 pins of a port in the GPIO module. 0 Pin operates as an input. 1 Pin operates as an output.

6.6.3 Output Configuration Register 1 (OCR1)

Each port consists of 32-pins. Because the output configuration for each pin is described using a two-bit combination the output configuration of the pins is controlled by two identical 32-bit registers (OCR1 and OCR2). The Output Configuration register 1 (OCR1) configures the output signal for lower 16 pins (0–15) of the associated port. Figure 6-5 shows the register and Table 6-4 provides its field descriptions.

0x1001_5004 (PTA_OCR1)
0x1001_5104 (PTB_OCR1)
0x1001_5204 (PTC_OCR1)
0x1001_5304 (PTD_OCR1)
0x1001_5404 (PTE_OCR1)
0x1001_5504 (PTF_OCR1)

Access: User R/W

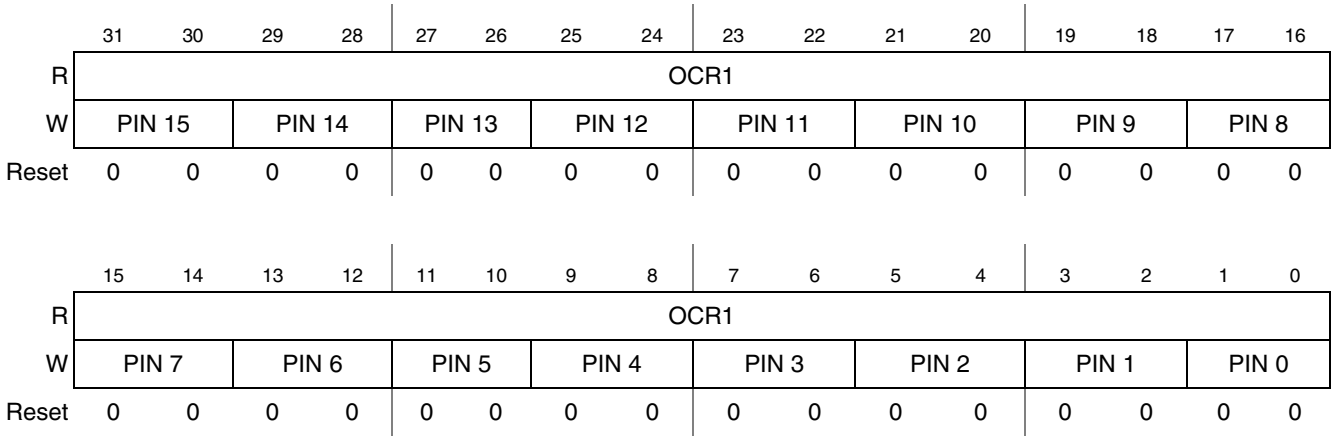


Figure 6-5. Output Configuration Register 1 (OCR1)

Table 6-4. Output Configuration Register 1 Field Descriptions

Field	Description
31–0 OCR1	Output Configuration Register 1. Each field selects how each pin (0–15) is used as an output by the GPIO. 00 Input A_IN output selected. 01 Input B_IN output selected. 10 Input C_IN output selected. 11 Data Register output selected.

6.6.4 Output Configuration Register 2 (OCR2)

The Output Configuration register 2 (OCR2) specifies the output signal for upper 16 pins (16–31) of the associated port. The output configuration for each pin is described with a two-bit combination. Figure 6-6 shows the register and Table 6-5 provides its field descriptions.

0x1001_5008 (PTA_OCR2)
0x1001_5108 (PTB_OCR2)
0x1001_5208 (PTC_OCR2)
0x1001_5308 (PTD_OCR2)
0x1001_5408 (PTE_OCR2)
0x1001_5508 (PTF_OCR2)

Access: User R/W

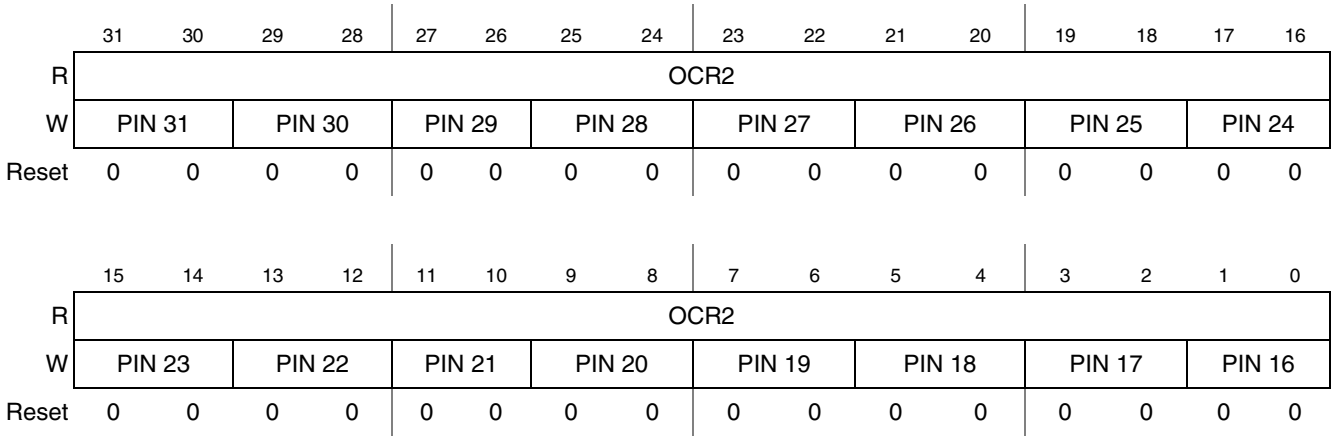


Figure 6-6. Output Configuration Register 2 (OCR2)

Table 6-5. Output Configuration Register 2 Field Descriptions

Field	Description
31–0 OCR2	Output Configuration Register 2. Each field selects how each pin (16–31) is used as an output by the GPIO. 00 Input A_IN output selected. 01 Input B_IN output selected. 10 Input C_IN output selected. 11 Data Register output selected.

6.6.5 Input Configuration Register A1 (ICONFA1)

The input configuration registers (ICONFA1) specify the signal or value driven to the A_OUT signals that is connected to internal modules of the BONO Device processor. Each port pin is defined by two bits in the input configuration registers. Figure 6-7 shows the register and Table 6-6 provides its field descriptions.

0x1001_500C (PTA_ICONFA1)
0x1001_510C (PTB_ICONFA1)
0x1001_520C (PTC_ICONFA1)
0x1001_530C (PTD_ICONFA1)
0x1001_540C (PTE_ICONFA1)
0x1001_550C (PTF_ICONFA1)

Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ICONFA1															
W	PIN 15		PIN 14		PIN 13		PIN 12		PIN 11		PIN 10		PIN 9		PIN 8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ICONFA1															
W	PIN 7		PIN 6		PIN 5		PIN 4		PIN 3		PIN 2		PIN 1		PIN 0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 6-7. Input Configuration Register A1 (ICONFA1)

Table 6-6. Input Configuration Register A1 Field Descriptions

Field	Description
31–0 ICONFA1	Input Configuration. Corresponds to port pins 0–15 and defines which one of the four options is driven to A_OUT. Each port pin requires two ICONFA1 bits to determine the input value. 00 GPIO_In 01 Interrupt Status Register 10 0 11 1

6.6.6 Input Configuration Register A2 (ICONFA2)

The input configuration registers (ICONFA2) specify the signal or value driven to the A_OUT signals connected to internal modules. There are two bits in the input configuration registers for each port pin. Figure 6-8 shows the register and Table 6-7 provides its field descriptions.

0x1001_5010 (PTA_ICONFA2)
0x1001_5110 (PTB_ICONFA2)
0x1001_5210 (PTC_ICONFA2)
0x1001_5310 (PTD_ICONFA2)
0x1001_5410 (PTE_ICONFA2)
0x1001_5510 (PTF_ICONFA2)

Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ICONFA2															
W	PIN 31		PIN 30		PIN 29		PIN 28		PIN 27		PIN 26		PIN 25		PIN 24	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ICONFA2															
W	PIN 23		PIN 22		PIN 21		PIN 20		PIN 19		PIN 18		PIN 17		PIN 16	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 6-8. Input Configuration Register A2 (ICONFA2)

Table 6-7. Input Configuration Register A2 Field Descriptions

Field	Description
31–0 ICONFA2	Input Configuration. Corresponds to port pins 16–31 and defines which one of the four options is driven to A_OUT. Each port pin requires two ICONFA2 bits to determine the input value. 00 GPIO_In 01 Interrupt Status Register 10 0 11 1

6.6.7 Input Configuration Register B1 (ICONFB1)

The input configuration registers ICONFB1 specify the signal or value driven to the B_OUT signals connected to internal modules. There are two bits in the input configuration registers for each port pin. Figure 6-9 shows the register and Table 6-8 provides its field descriptions.

0x1001_5014 (PTA_ICONFB1)
0x1001_5114 (PTB_ICONFB1)
0x1001_5214 (PTC_ICONFB1)
0x1001_5314 (PTD_ICONFB1)
0x1001_5414 (PTE_ICONFB1)
0x1001_5514 (PTF_ICONFB1)

Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ICONFB1															
W	PIN 15		PIN 14		PIN 13		PIN 12		PIN 11		PIN 10		PIN 9		PIN 8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ICONFB1															
W	PIN 7		PIN 6		PIN 5		PIN 4		PIN 3		PIN 2		PIN 1		PIN 0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 6-9. Input Configuration Register B1 (ICONFB1)

Table 6-8. Input Configuration Register B1 Field Descriptions

Name	Description
31–0 ICONFB1	Input Configuration. Corresponds to pins 0–15 of the port and defines which one of the four options is driven to b_OUT. Each port pin requires two ICONFB1 bits to determine the input value. 00 GPIO_IN 01 Interrupt Status register 10 0 11 1

6.6.8 Input Configuration Register B2 (ICONFB2)

The input configuration registers ICONFB2 specify the signal or value driven to the B_OUT signals connected to internal modules. There are two bits in the input configuration registers for each port pin. Figure 6-10 shows the register and Table 6-9 provides its field descriptions.

0x1001_5018 (PTA_ICONFB2)
0x1001_5118 (PTB_ICONFB2)
0x1001_5218 (PTC_ICONFB2)
0x1001_5318 (PTD_ICONFB2)
0x1001_5418 (PTE_ICONFB2)
0x1001_5518 (PTF_ICONFB2)

Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ICONFB2															
W	PIN 31		PIN 30		PIN 29		PIN 28		PIN 27		PIN 26		PIN 25		PIN 24	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ICONFB2															
W	PIN 23		PIN 22		PIN 21		PIN 20		PIN 19		PIN 18		PIN 17		PIN 16	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 6-10. Input Configuration Register B1 (ICONFB2)

Table 6-9. Input Configuration Register B2 Description

Name	Description
31–0 ICONFB2	Input Configuration. Corresponds to pins 16–31 of the port and defines which one of the four options is driven to b_OUT. Each port pin requires two ICONFB2 bits to determine the input value. 00 GPIO_IN 01 Interrupt Status register 10 0 11 1

6.6.9 Data Register (DR)

The Data Register holds data for output from an associated port when a pin is configured as an output and the Data Register is chosen using Output Configuration Register 1 and Output Configuration Register 2. Figure 6-11 shows the register and Table 6-10 provides its field descriptions.

0x1001_501C (PTA_DR)
0x1001_511C (PTB_DR)
0x1001_521C (PTC_DR)
0x1001_531C (PTD_DR)
0x1001_541C (PTE_DR)
0x1001_551C (PTF_DR)

Access: User R/W

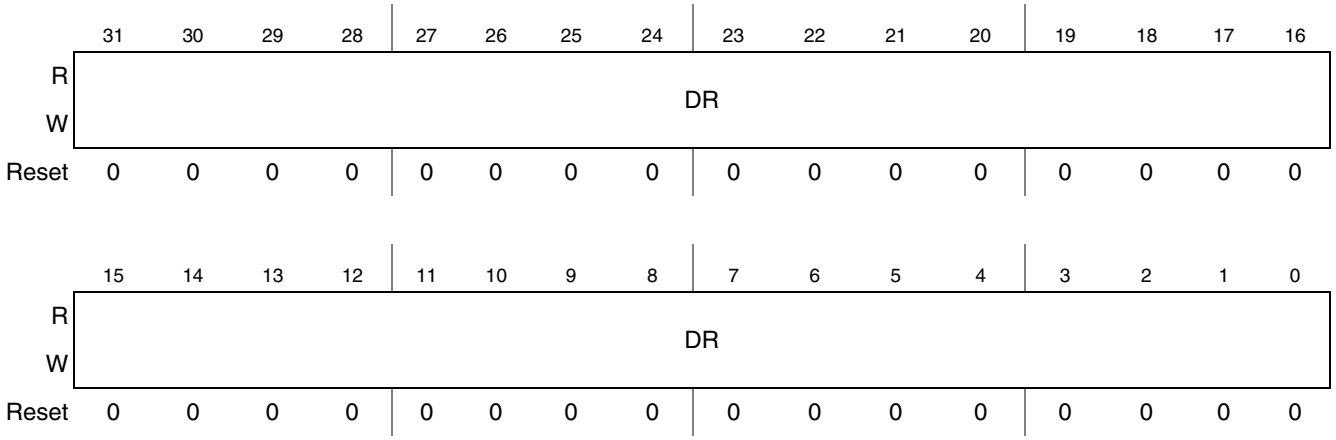


Figure 6-11. Data Register (DR)

Table 6-10. Data Register Field Descriptions

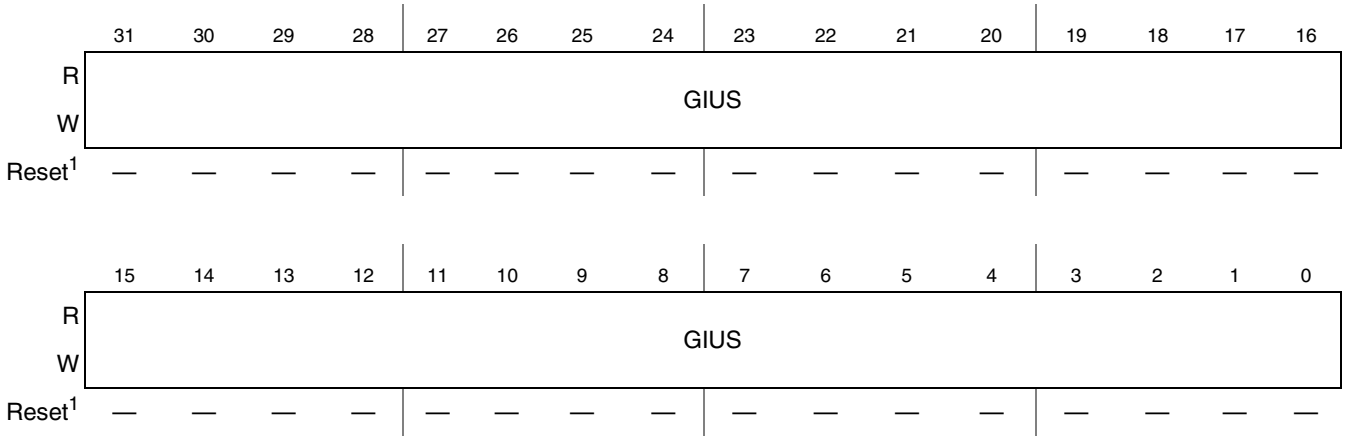
Field	Description
31–0 DR	Data Register. Contains the GPIO output values when the Output Configuration Registers select the Data Register as the output for the pin (selection 11). 0 Drives the output signal is low. 1 Drives the output signal is high.

6.6.10 GPIO IN USE Registers (GIUS)

The GPIO In Use Registers control a multiplexer in the IOMUX module. The settings in these registers choose whether a pin is utilized for a peripheral function or for its GPIO function. If the register is set to a zero for a corresponding pin, then this register is used in conjunction with the GPR register to control the peripheral functionality. Figure 6-12 shows a GIUS overview register and Table 6-11 provides field descriptions of the GIUS registers. Reset values for individual registers are shown in the following sections.

0x1001_5020 (PTA_GIUS)
0x1001_5120 (PTB_GIUS)
0x1001_5220 (PTC_GIUS)
0x1001_5320 (PTD_GIUS)
0x1001_5420 (PTE_GIUS)
0x1001_5520 (PTF_GIUS)

Access: User R/W



1 The reset value of this register is determined by the input value of the signal INUSE_RESET_SEL [31:0].

Figure 6-12. GPIO IN USE Register (GIUS)

Table 6-11. GPIO In Use Register Field Descriptions

Field	Description
31–0 GIUS	GPIO In Use. Informs the IOMUX module whether the port pin is utilized for its GPIO function. When the pin is utilized for its GPIO function, the multiplexed functions are not available. The reset value of this register is determined by the input value of the signal INUSE_RESET_SEL [31:0]. 0 Pin utilized for multiplexed function 1 Pin utilized for GPIO function

6.6.11 GPIO IN USE Register Reset Values

The following sections describe the GPIO In Use (GIUS) reset values for the various ports. Additionally, the registers also indicate the reserved bits (unimplemented GPIO bits) of the GPIO ports.

6.6.11.1 GPIO IN USE Register A (PTA_GIUS)

The reset value of the PTA_GIUS register is (0xFFFF_FFFF). Figure 6-13 shows the register.

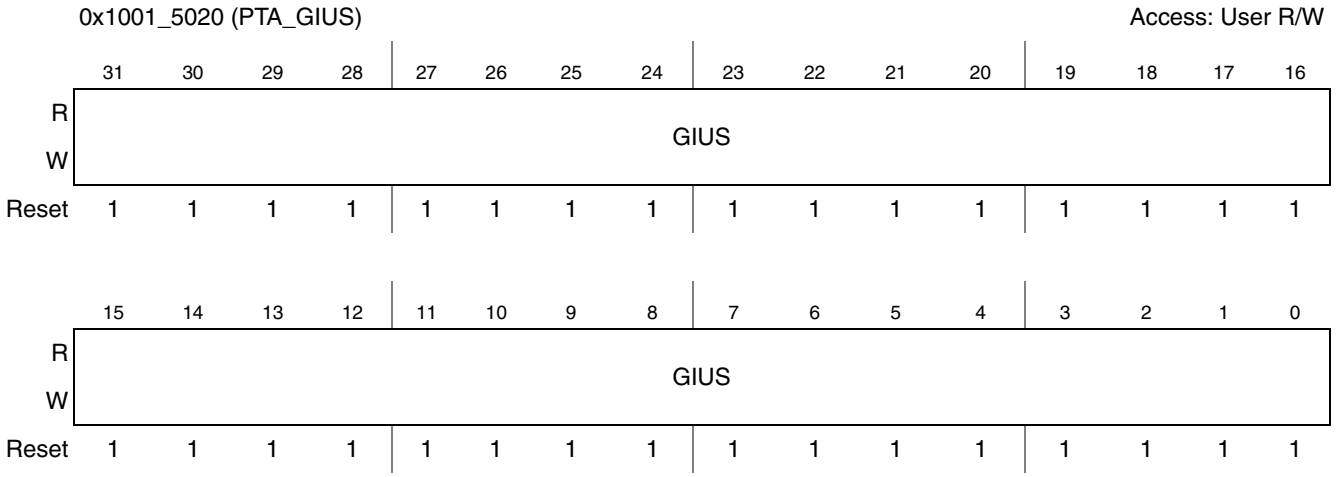


Figure 6-13. GPIO IN USE Register A Reset Values (PTA_GIUS)

6.6.11.2 GPIO IN USE Register B (PTB_GIUS)

The reset value of the PTB_GIUS register is (0xFF3F_FFF3). Figure 6-14 shows the register.

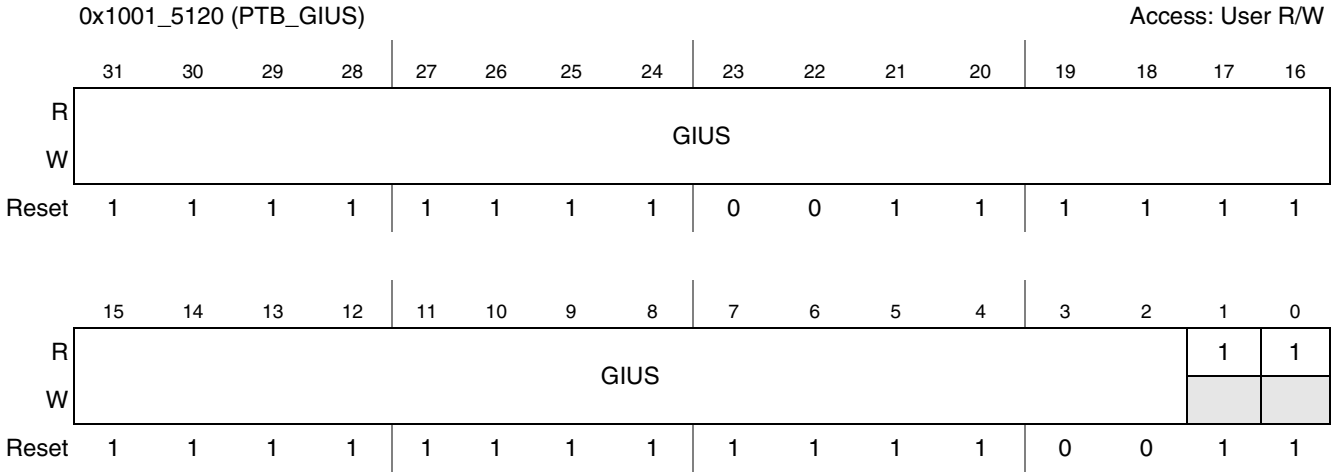


Figure 6-14. GPIO IN USE Register B Reset Values (PTB_GIUS)

6.6.11.3 GPIO IN USE Register C (PTC_GIUS)

The reset value of the PTC_GIUS register is (0xFFFF_FFFF). Figure 6-15 shows the register.

0x1001_5220 (PTC_GIUS) Access: User R/W

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

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

Figure 6-15. GPIO IN USE Register C Reset Values (PTC_GIUS)

6.6.11.4 GPIO IN USE Register D (PTD_GIUS)

The reset value of the PTD_GIUS register is (0xFFFE_0000). Figure 6-16 shows the register.

0x1001_5320 (PTD_GIUS) Access: User R/W

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

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

Figure 6-16. GPIO IN USE Register D Reset Values (PTD_GIUS)

6.6.11.5 GPIO IN USE Register E (PTE_GIUS)

The reset value of the PTE_GIUS register is (0xFFFC_0F27). Figure 6-17 shows the register.

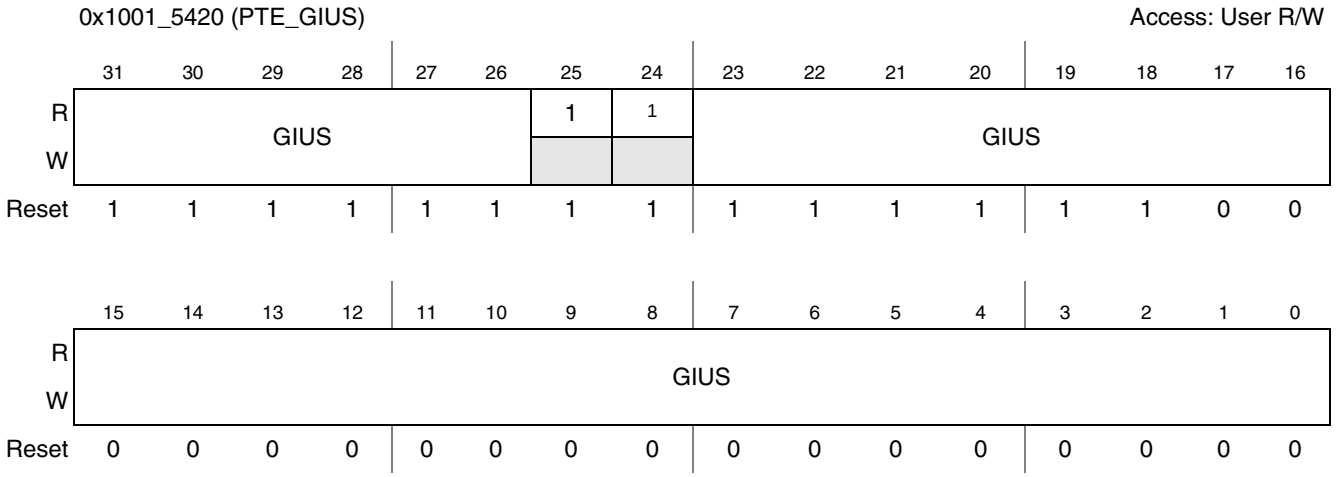


Figure 6-17. GPIO IN USE Register E Reset Values (PTE_GIUS)

6.6.11.6 GPIO IN USE Register F (PTF_GIUS)

The reset value of the PTF_GIUS register is (0xFF00_0000). Figure 6-17 shows the register.

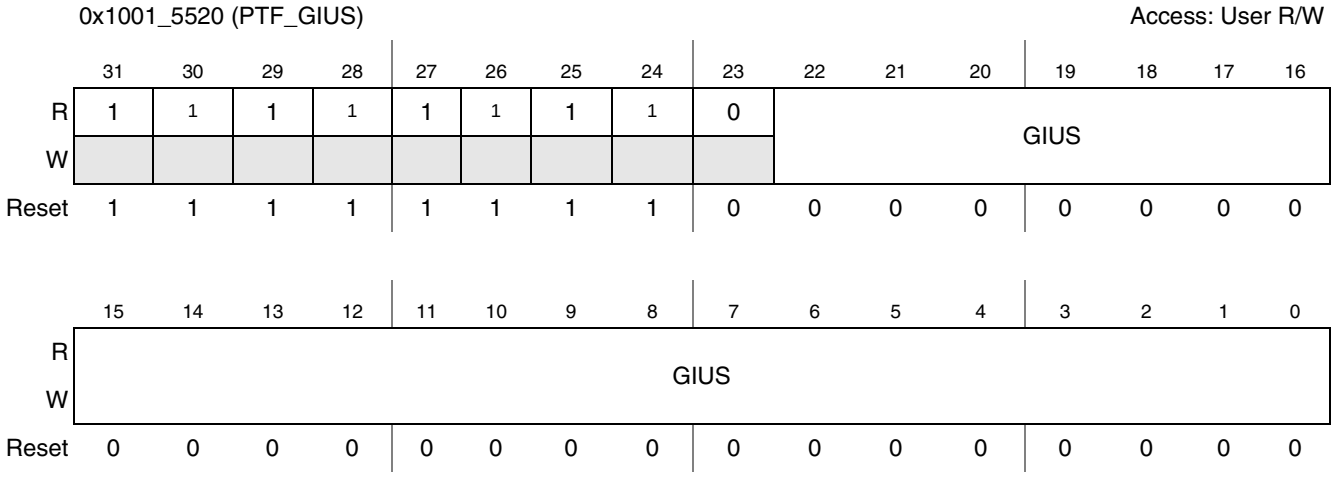


Figure 6-18. GPIO IN USE Register F Reset Values (PTF_GIUS)

6.6.12 Sample Status Register (SSR)

The read-only Sample Status Registers contain the value of the GPIO pins for each associated port. The register is updated on every clock tick. The contents are used as a status indicator when the pins are configured as inputs. Figure 6-19 shows the register and Table 6-12 provides its field descriptions.

0x1001_5024 (PTA_SSR)
0x1001_5124 (PTB_SSR)
0x1001_5224 (PTC_SSR)
0x1001_5324 (PTD_SSR)
0x1001_5424 (PTE_SSR)
0x1001_5524 (PTF_SSR)

Access: User read-only

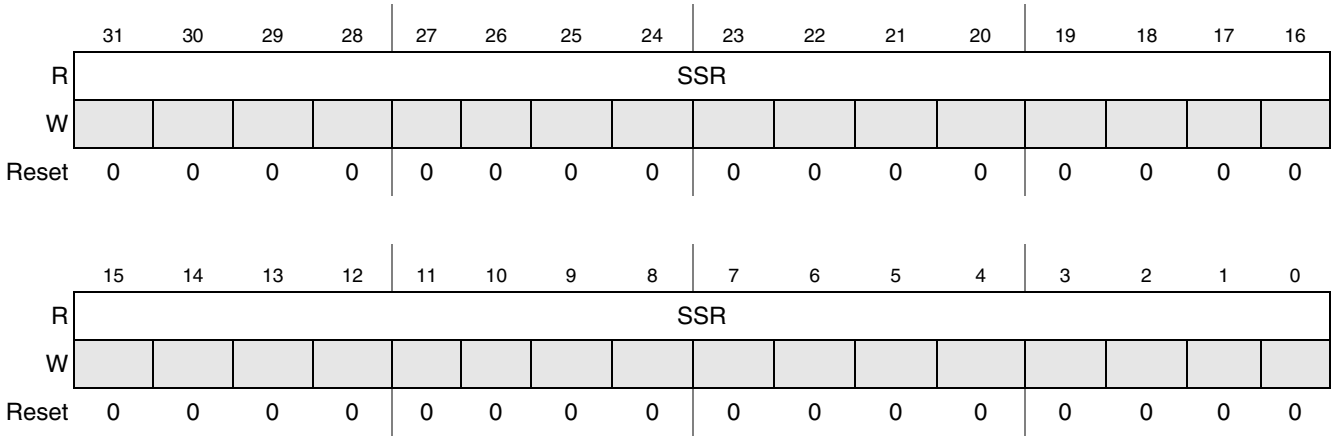


Figure 6-19. Sample Status Register (SSR)

Table 6-12. Sample Status Register Field Descriptions

Field	Description
31–0 SSR	Sample Status. Contains the value of the GPIO pin [i]. It is sampled on every clock. 0 Pin value is low. 1 Pin value is high.

6.6.13 Interrupt Configuration Register 1 (ICR1)

This register specifies the external interrupt configuration for each of the lower 16 interrupts of a port. There are two bits in the register for each port pin. Figure 6-20 shows the register and Table 6-13 provides its field descriptions.

0x1001_5028 (PTA_ICR1)
0x1001_5128 (PTB_ICR1)
0x1001_5228 (PTC_ICR1)
0x1001_5328 (PTD_ICR1)
0x1001_5428 (PTE_ICR1)
0x1001_5528 (PTF_ICR1)

Access: User R/W

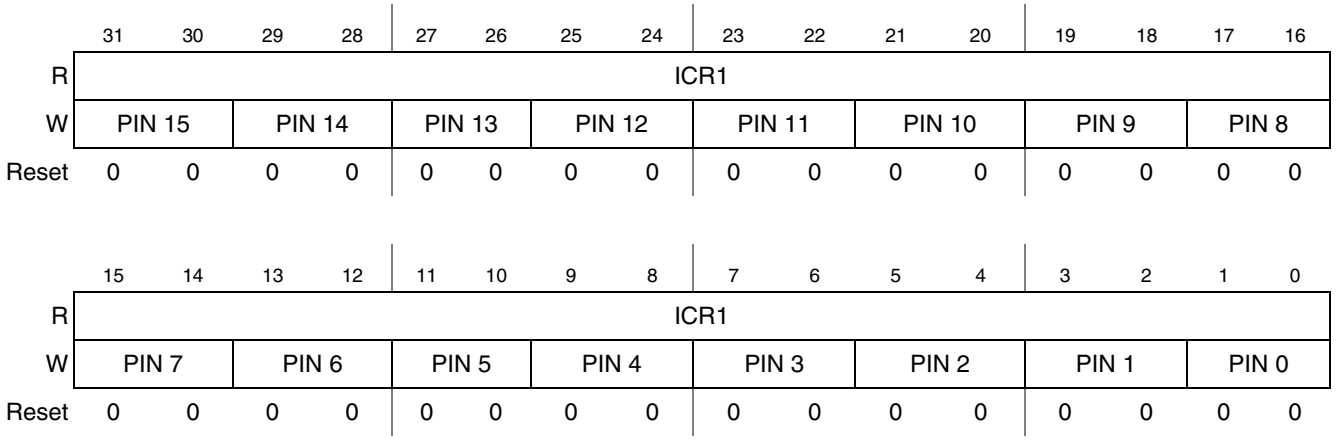


Figure 6-20. Interrupt Configuration Register 1 (ICR1)

Table 6-13. Interrupt Configuration Register 1 Field Descriptions

Field	Description
31–0 ICR1	Interrupt Configuration. Corresponds to interrupts 0–15 of the port and defines which one of the four options is the sensitivity of the interrupt. Each interrupt [i] (i= 0 through 15) requires two ICR1 bits to determine the sensitivity. 00 Rising edge sensitive 01 Falling edge sensitive 10 High level sensitive 11 Low level sensitive

6.6.14 Interrupt Configuration Register 2 (ICR2)

This register specify the external interrupt configuration for each of the upper 16 interrupts of the port. There are two bits in the register for each port pin. Figure 6-21 shows the register and Table 6-14 provides its field descriptions.

0x1001_502C (PTA_ICR2)
0x1001_512C (PTB_ICR2)
0x1001_522C (PTC_ICR2)
0x1001_532C (PTD_ICR2)
0x1001_542C (PTE_ICR2)
0x1001_552C (PTF_ICR2)

Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ICR2															
W	PIN 31		PIN 30		PIN 29		PIN 28		PIN 27		PIN 26		PIN 25		PIN 24	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ICR2															
W	PIN 23		PIN 22		PIN 21		PIN 20		PIN 19		PIN 18		PIN 17		PIN 16	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6-21. Interrupt Configuration Register 2 (ICR2)

Table 6-14. Interrupt Configuration Register 2 Field Descriptions

Field	Description
31–0 ICR2	Interrupt Configuration. Corresponds to interrupts 16–31 of the port and defines which one of the four options is the sensitivity of the interrupt. Each interrupt requires two ICR2 bits to determine the sensitivity. 00 Rising edge sensitive 01 Falling edge sensitive 10 High level sensitive 11 Low level sensitive

6.6.15 Interrupt Mask Register (IMR)

The Interrupt Mask Registers (IMR) determine if an interrupt will be asserted when an interrupt event occurs and when the pin and corresponding bit is configured in an interrupt mode. An interrupt is asserted when corresponding bits in the IMR and ISR are set. Figure 6-22 shows the register and Table 6-15 provides its field descriptions.

0x1001_5030 (PTA_IMR)
0x1001_5130 (PTB_IMR)
0x1001_5230 (PTC_IMR)
0x1001_5330 (PTD_IMR)
0x1001_5430 (PTE_IMR)
0x1001_5530 (PTF_IMR)

Access: User R/W

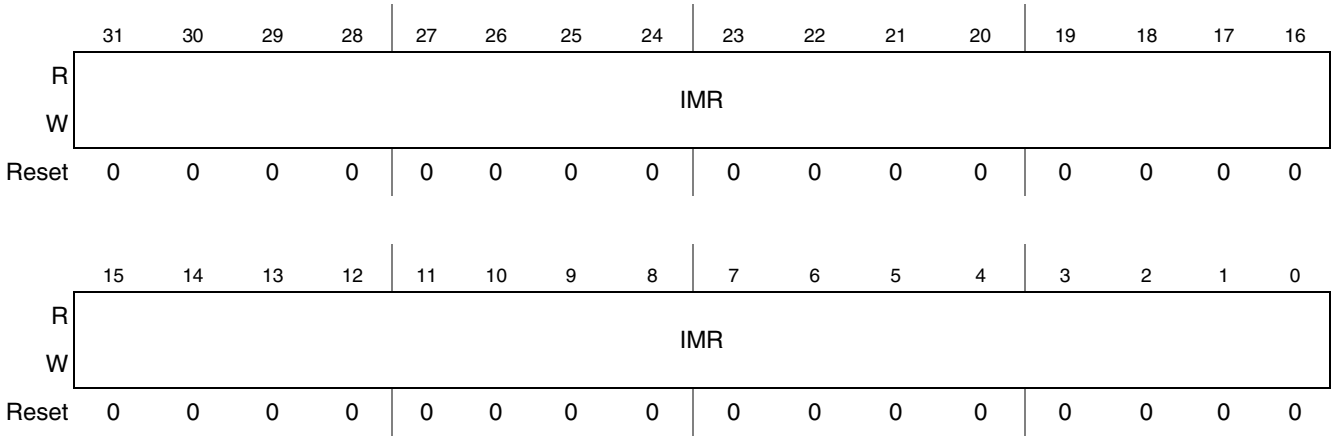


Figure 6-22. Interrupt Mask Register (IMR)

Table 6-15. Interrupt Mask Register Description

Name	Description
31–0 IMR	Interrupt Mask. Masks the interrupts for this module. 0 Interrupt is masked. 1 Interrupt is not masked.

6.6.16 Interrupt Status Register (ISR)

The Interrupt Status Registers (ISR) indicate if an interrupt has occurred. When an interrupt event occurs, the bit in this register is set. The condition necessary to set the bit is determined by the Interrupt Configuration Registers (ICR) and the inputs satisfying the interrupt condition. Figure 6-23 shows the register and Table 6-16 provides its field descriptions.

0x1001_5034 (PTA_ISR)
0x1001_5134 (PTB_ISR)
0x1001_5234 (PTC_ISR)
0x1001_5334 (PTD_ISR)
0x1001_5434 (PTE_ISR)
0x1001_5534 (PTF_ISR)

Access: User R/W

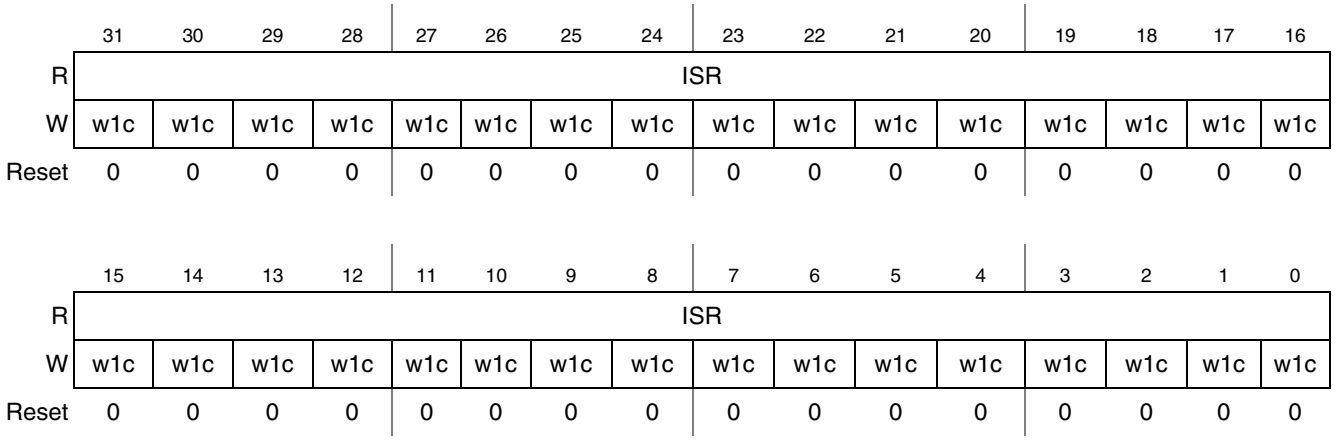


Figure 6-23. Interrupt Status Register (ISR)

Table 6-16. Interrupt Status Register Field Descriptions

Field	Description
31–0 ISR	Interrupt Status. Indicates whether the interrupt [i] has occurred for in the GPIO module. The bits of this register are write 1 to clear. The w1c bit is cleared when a value of 1 is written to the associated bit. 0 Interrupt has not occurred. 1 Interrupt has occurred.

6.6.17 General Purpose Register (GPR)

The General Purpose Registers (GPR) control a multiplexer in the IOMUX module. When the corresponding bit in the associated GIUS register is set to zero, the settings in these registers determine whether a pin is utilized for its primary peripheral function or for its alternate peripheral function. When the corresponding bit in the GIUS is set, the settings of this register have no effect. Figure 6-24 shows the register and Table 6-17 provides its field descriptions.

0x1001_5038 (PTA_GPR)
0x1001_5138 (PTB_GPR)
0x1001_5238 (PTC_GPR)
0x1001_5338 (PTD_GPR)
0x1001_5438 (PTE_GPR)
0x1001_5538 (PTF_GPR)

Access: User R/W

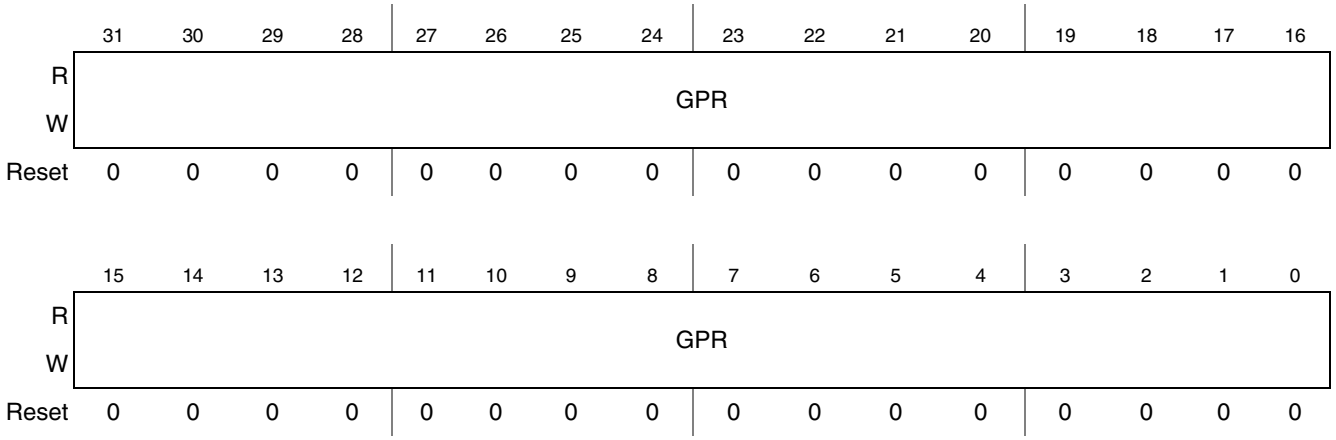


Figure 6-24. General Purpose Register

Table 6-17. General Purpose Register Field Descriptions

Field	Description
31–0 GPR	General Purpose Register. Selects between the primary and alternate functions of the pin. When the associated bit in the GIUS register is set, this bit has no meaning. **Note:** Ensure that this bit is cleared when there is not an alternate function for the associated pin. 0 Select primary pin function 1 Select alternate pin function

6.6.18 Software Reset Register (SWR)

The Software Reset Register (SWR) controls the reset of the individual ports in the GPIO module. When the SWR bit of the Software Reset Register is set, the GPIO circuitry for the individual port resets immediately.

The total time of the software reset sequence will take six clock cycles. The reset will be asserted from the third cycle and remains asserted for three clocks.

Figure 6-25 shows the register and Table 6-18 provides its field descriptions.

0x1001_503C (PTA_SWR)
0x1001_513C (PTB_SWR)
0x1001_523C (PTC_SWR)
0x1001_533C (PTD_SWR)
0x1001_543C (PTE_SWR)
0x1001_553C (PTF_SWR)

Access: User R/W

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

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

Figure 6-25. Software Reset Register (SWR)

Table 6-18. Software Reset Register Field Descriptions

Field	Description
31–1	Reserved. These bits are reserved and should read 0
0 SWR	Software Reset. Controls software reset of the port. The reset signal is active for 3 system clock cycles and then it is released automatically. It is a self-clearing bit. 0 No effect 1 GPIO circuitry for Port X reset

6.6.19 Pull-Up Enable Register (PUEN)

The Pull-Up Enable (PUEN) Registers enable or disable a 69 kΩ pull-up resistor on the associated pin. The pull-up can be applied to any GPIO pin regardless of whether it is configured as primary, alternate or GPIO function. The pin is tri-stated when the pull-up is disabled and the pin is not driven. Figure 6-26 shows the register and Table 6-19 provides its field descriptions.

NOTE

Bits 27–24 on Port A (PTA_PUEN) enables or disables a 69 kΩ pull-down resistor on the associated pin.

Bits 31–28, 26, and 9 on Port B (PTB_PUEN) enables or disables a 69 kΩ pull-down resistor on the associated pin.

0x1001_5040 (PTA_PUEN)
0x1001_5140 (PTB_PUEN)
0x1001_5240 (PTC_PUEN)
0x1001_5340 (PTD_PUEN)
0x1001_5440 (PTE_PUEN)
0x1001_5540 (PTF_PUEN)

Access: User R/W

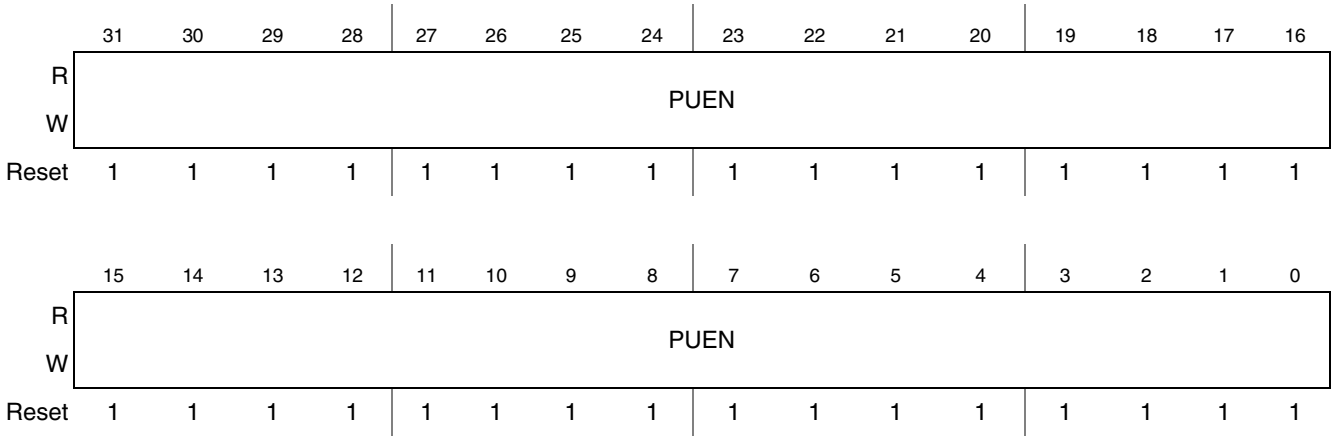


Figure 6-26. Pull-Up Enable Register (PUEN)

Table 6-19. Pull-Up Enable Register Field Descriptions

Field	Description
31–0 PUEN	Pull-Up Enable. Determines whether the corresponding pad is pulled up to a logic-high or tri-stated. When the pin is configured as an input, clearing this bit causes the signal to be tri-stated when not driven by an external source. When the pin is configured as an output, clearing this bit causes the signal to be tri-stated when it is not enabled. 0 Pin [i] is tri-stated when not driven internally or externally. 1 Pin [i] is pulled high1 when not driven internally or externally.

6.6.20 Port Interrupt Mask Register (PMASK)

The GPIO has six ports, each with interrupt generation capability. The PMASK register provides interrupt masking capability at the port level while the Interrupt Mask Register provides control over individual interrupts. If a bit is zero, then all interrupts for that port are masked. A software reset on a port (SWR is set) will clear the corresponding mask bit of the port in this register. Figure 6-27 shows the register and Table 6-20 provides its field descriptions.

0x1001_5600 (PMASK) Access: User R/W

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	PTF	PTE	PTD	PTC	PTB	PTA
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 6-27. Port Interrupt Mask Register (PMASK)

Table 6-20. Port Interrupt Mask Register Field Descriptions

Field	Description
31–6	Reserved. These bits are reserved and should read 0.
5 PTF	Port F. The bit helps in masking the Port F interrupt. The bit clears during software reset of Port F. 0 Interrupt is masked. 1 Interrupt is not masked.
4 PTE	Port E. The bit helps in masking the Port E interrupt. The bit clears during software reset of Port E. 0 Interrupt is masked. 1 Interrupt is not masked.
3 PTD	Port D. The bit helps in masking the Port D interrupt. The bit clears during software reset of Port D. 0 Interrupt is masked. 1 Interrupt is not masked.
2 PTC	Port C. The bit helps in masking the Port C interrupt. The bit clears during software reset of Port C. 0 Interrupt is masked. 1 Interrupt is not masked.
1 PTB	Port B. The bit helps in masking the Port B interrupt. The bit clears during software reset of Port B. 0 Interrupt is masked. 1 Interrupt is not masked.
0 PTA	Port A. The bit helps in masking the Port A interrupt. The bit clears during software reset of Port A. 0 Interrupt is masked. 1 Interrupt is not masked.

Chapter 7
JTAG Controller (JTAGC)

7.1 Introduction

The JTAG Controller (JTAGC) module supports Debug access to ARM926 core and tri-state enabling of the I/O pads. The JTAGC is compatible with 1EEE1149.1 Standard Test Access Port and Boundary Scan Architecture.

7.2 Features

The Test and debug features of JTAG provide the following capabilities:

- Provide debug access to ARM926 core and execute its specific JTAG instructions independently
- Controls tri-state enable of I/O pads

7.3 Implementation

The JTAG Controller consists of the JTAG Controller state machine, Instruction Register (IR), Bypass Register, Boundary Scan Register, Instruction decode, and various user specific data registers collectively reside inside the ExtraDebug register.

The TDO output from the JTAG Controller is the muxed output based on whether i.MX27 JTAG Controller or ARM926 Platform JTAG mode is active. It changes on falling edge of TCK. The TDO output enable is selected based on whether i.MX27 JTAG Controller or ARM926 Platform JTAG mode is active.

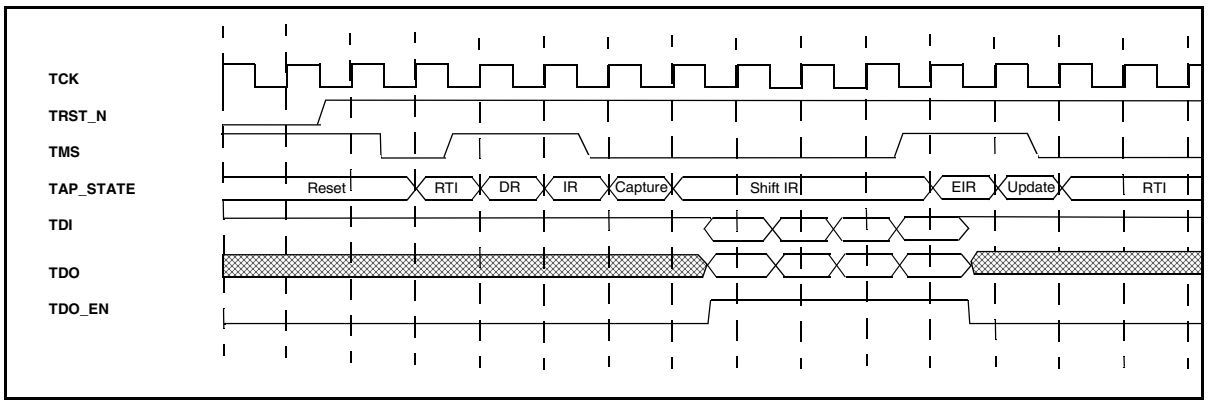


Figure 7-1. JTAG Signals Timing Diagram

The Test Mode Select (TMS) input from external pin by default connects to the ARM926 Platform after gating with the laser fuse output. At the rising edge of TRST_B, the JTAG_control input controls whether the TMS pin should be connected to ARM926 Platform or to i.MX27 JTAG Controller.

When JTAG_control input is HIGH (by default), the TMS pin will be connected to ARM926 Platform after gating with the laser fuse output. The TMS input of i.MX27 JTAG Controller will be held HIGH in this case. When JTAG_control input is LOW, the TMS pin will be connected to i.MX27 JTAG Controller. The TMS input of ARM926 Platform will be held HIGH.

- The Test Reset (TRST_B) input from external pin will be connected to both ARM926 Platform and i.MX27 JTAG Controller.
- The Test Data Input (TDI) input from external pin will be connected to both ARM926 Platform and i.MX27 JTAG Controller.
- The Test Clock (TCK) input from external pin will be connected to both ARM926 Platform and i.MX27 JTAG Controller.

7.4 JTAG Controller Pin List

Table 7-1 provides a list of the JTAGC pins.

Table 7-1. JTAGC Pin List

Pin Name	Direction	Description
tdo	Output	Test Data Output TDO is asserted during rising edge of TCK
tck	Input	Test Clock Test Clock input is used to synchronize the Test Logic. This includes an internal pull-up resistor.
tdi	Input	Test Data Input TDI is captured during rising edge of TCK. TDI includes an internal pull-up resistor.
tms	Input	Test Mode Select TMS is captured during rising edge of TCK. TMS includes an internal pull-up resistor. TMS input for the ARM926 Platform and JTAG Controller is gated by the system logic.
trst_b	Input	Test Reset TRST_B includes an internal pull-up resistor

7.5 JTAG Overview

Figure 7-2 shows the i.MX27 JTAG block diagram.

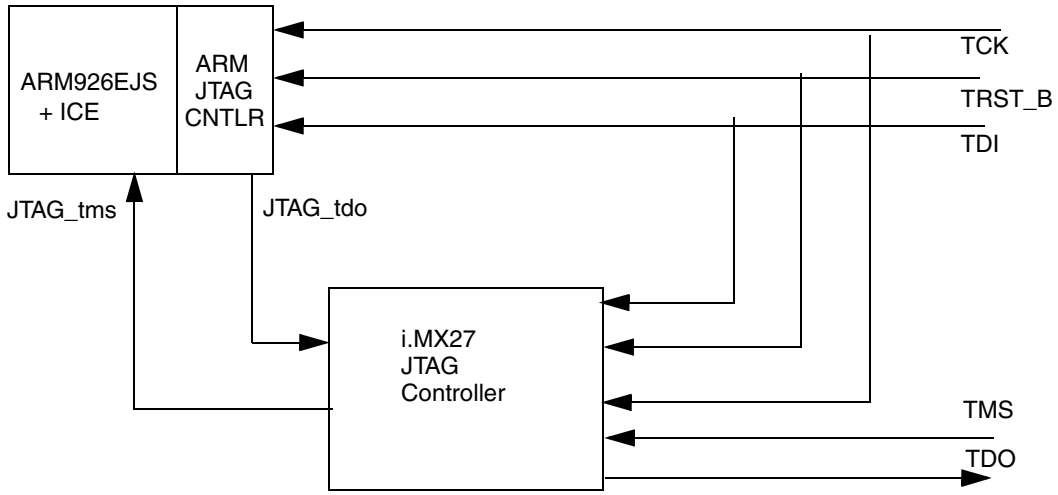


Figure 7-2. i.MX27 JTAG Block Diagram

7.6 JTAG Modes

Two JTAG modes are created based on the I/O pin JTAG_control. These modes are used to maintain compatibility to ARM MCU Multi-ICE^{TM} products as well as maintain IEEE JTAG standards.

7.6.1 ARM926 Platform mode

This mode connects the processed TMS input to the ARM926 Platform. TRST_B must be asserted to exit this mode.

7.6.2 i.MX27 JTAG Controller mode

This mode will connect the processed TMS input to the i.MX27 JTAG Controller. This will provide a dedicated user-accessible test access port that uses the same communication style as the IEEE1149.1 Standard. TRST_B or POR_B must be asserted to leave this mode.

In this mode, i.MX27 JTAG Controller supports the following capabilities:

- Query identification information (manufacturer, part number and version) of i.MX27 (IDCODE)
- Tri-state I/O pads for iddq test (HIGHZ)
- BYPASS instruction

7.7 Boundary Scan Register

The boundary scan register (BSR) in the i.MX27 JTAG implementation contains bits for all device signals and clock pins and associated control signals. All i.MX27 bidirectional pins have a single register bit in the boundary scan register for pin data, and are controlled by an associated control bit in the boundary scan register.

7.8 Instruction Register

The JTAG Instruction register is 3 bits wide. The settings of the IR is shown in Table 7-2.

Table 7-2. JTAG Instruction Register

Bit2	Bit1	Bit0	Instruction
0	0	0	IDCODE
0	0	1	SAMPLE/PRELOAD
0	1	0	EXTEST
0	1	1	ENABLE_ExtraDebug
1	0	0	HIGHZ
1	0	1	ACCESS_GENERIC_MBIST
1	1	0	CLAMP
1	1	1	BYPASS

The instruction register is reset to 3’b000 which is equivalent to the IDCODE instruction.

During the capture-IR state, the parallel inputs to the instruction register are loaded with the code 01 in the least significant bits as required by the IEEE standard, the most significant bits are loaded with the values 0, leading to a capture value of 3’b001.

7.8.1 EXTEST Instruction

The EXTEST instruction selects the boundary scan register, and the 1149.1 test logic has control of the I/O pins. EXTEST also asserts internal reset for the Core to force a predictable internal state while performing external boundary scan operations.

By using the TAP Controller, the register is capable of:

- Scanning user-defined values into the output buffers
- Capturing values presented to input pins controlling the direction of bidirectional pins
- Controlling the output drive of tri-statable output pins

For more details on the function and use of EXTEST, refer to the IEEE 1149.1 document.

7.8.2 SAMPLE/PRELOAD Instruction

This selects the boundary scan register and the system logic controls the I/O pins. The SAMPLE/PRELOAD instruction provides two separate functions. First, it provides a means to obtain a snapshot of system data and control signals. The snapshot occurs on the rising edge of TCK in the capture-DR Controller state. The data can be observed by shifting it transparently through the boundary scan register.

NOTE

Since there is no internal synchronization between the JTAG clock (TCK) and the system clock (CLK), the user must provide some form of external synchronization to achieve meaningful results.

The second function of SAMPLE/PRELOAD is to initialize the boundary scan register output cells prior to selection of EXTEST. This initialization ensures that known data will appear on the outputs when entering the EXTEST instruction.

7.8.3 IDCODE Instruction

This selects the ID register and the system logic controls the I/O pins. This instruction is a public instruction to allow the manufacturer, part number and the version of the IC to be available through TAP. Figure 7-3 shows the ID register configuration.

0x0000_7000 (IDCODE) Access: User R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Version Information				Design Center Part Number						Device Number [21:12]					
W																
Reset	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Device Number [21:12]				MFG											1
W																
Reset	0	0	0	0	1	0	1	1	0	0	0	1	1	1	0	1

Figure 7-3. ID Register Configuration

7.8.4 ENABLE_ExtraDebug Instruction

The ExtraDebug register consists of 44 bits comprising a 40-bits register (maximum), a 3-bit address field and one read/write bit. The register data field does not need to be filled in during register read. The particular ExtraDebug register connected between TDI and TDO is selected by the ExtraDebug Controller based on the currently decoded address during Update_DR state. All communication with the ExtraDebug Controller is done through the Select-DR-Scan path of the i.MX27 JTAG Controller.

7.8.5 HIGHZ Instruction

All output drivers, including the two-state drivers, are turned off (that is, high impedance). The instruction selects the bypass register. The HIGHZ instruction also asserts internal reset for the Core to force a predictable internal state while performing external boundary scan operations. In this mode, all internal pull-up resistors on all the pins (except for the TMS TDI TCK TRST COLD_START MUXCTL pins) will be disabled.

7.8.6 CLAMP Instruction

Selects the 1-bit bypass register as the serial path between TDI and TDO while allowing signals driven from the component pins to be determined from the boundary scan register. During testing of ICs on PCB, it may be necessary to place static guarding values on signals that control operation of logic not involved in the test. The EXTEST instruction could be used for this purpose, but since it selects the boundary-scan register the required guarding signals would be loaded as part of the complete serial data stream shifted in, both at the start of the test and each time a new test pattern is entered. Since the CLAMP instruction allows guarding values to be applied using the boundary-scan register of the appropriate ICs while selecting their bypass registers, it allows much faster testing than does the EXTEST instruction. Data in the boundary scan cell remains unchanged until a new instruction is shifted in or the JTAG state machine is set to its reset state. The CLAMP instruction also asserts internal reset for the Core to force a predictable internal state while performing external boundary scan operations.

7.8.7 BYPASS Instruction

Selects the single bit Bypass register and the system logic controls the I/O pins. This creates a shift register path from TDI to the bypass register and finally to TDO.

When the bypass register is selected by the current instruction, the shift-register stage is set to a logic zero on the rising edge of TCK in the capture-DR controller state. The first bit to be shifted out after selecting the bypass register will always be a logic zero.

7.9 TMS Sequences

7.9.1 TMS Sequence to Check ID Code

The following table shows the TMS sequence to check the ID Code value, starting from any point in the state machine.

Table 7-1. TMS Sequence To Check ID Code

Step	TCK	TMS	State	Comment
0	x5	1	Test Logic Reset	SEQUENCE IS : IDCODE READ
1	x1	0	Run-Test/Idle	
2	x1	1	Select DR	
3	x1	1	Select IR	IR Path : Loading 'Idcode' instr.
4	x1	0	Capture IR	
5	x1	0	Shift IR	Shift 'Idcode' inst.= 3'b010 thru TDI
6	x2	0	Shift	
7	x1	1	Exit1	
8	x1	1	Update	Select Idcode register
9	x1	0	Run-Test/Idle	

Table 7-1. TMS Sequence To Check ID Code

Step	TCK	TMS	State	Comment
10	x1	1	Select DR	DR Path: Reading Idcode reg.
11	x1	0	Capture DR	Capture Idcode value
12	x1	0	Shift DR	Shift out Idcode on 32bits
13	x31	0	Shift	
14	x1	1	Exit1	
15	x1	1	Update	
16	x1	0	Run-Test/Idle	

7.9.2 TMS Sequence to Write to ExtraDebug Register

Table 7-2 shows the TMS sequence to Write to any of the ExtraDebug registers, starting from any point in the state machine.

Table 7-2. TMS Sequence to Write to ExtraDebug Register

Step	TCK	TMS	State	Comment
0	x5	1	Test Logic Reset	SEQUENCE IS : WRITE ExtraDebug Register
1	x1	0	Run-Test/Idle	
2	x1	1	Select DR	
3	x1	1	Select IR	IR Path : Select ExtraDebug register.
4	x1	0	Capture IR	
5	x1	0	Shift IR	Shift ‘Enable ExtraDebug’ inst.= 3'b011 thru TDI
6	x2	0	Shift	
7	x1	1	Exit1	
8	x1	1	Update	Select ExtraDebug register
9	x1	0	Run-Test/Idle	
10	x1	1	Select DR	DR Path: Select Extradebug register to write data
11	x1	0	Capture DR	
12	x1	0	Shift DR	Shift In Writ bit(1’b0) + Register Address + Data
13	x43	0	Shift	
14	x1	1	Exit1	
15	x1	1	Update	Write to the ExtraDebug register
16	x1	0	Run-Test/Idle	

7.9.3 TMS Sequence to Read ExtraDebug Register

The following table shows the TMS sequence to READ any of the ExtraDebug registers, starting from any any point in the state machine.

Table 7-3. TMS Sequence to Read ExtraDebug Register

Step	TCK	TMS	State	Comment
0	x5	1	Test Logic Reset	SEQUENCE IS : READ ExtraDebug Register
1	x1	0	Run-Test/Idle	
2	x1	1	Select DR	
3	x1	1	Select IR	IR Path : Select ExtraDebug register.
4	x1	0	Capture IR	
5	x1	0	Shift IR	Shift 'Enable ExtraDebug' inst.= 3'b011 thru TDI
6	x2	0	Shift	
7	x1	1	Exit1	
8	x1	1	Update	Select ExtraDebug register
9	x1	0	Run-Test/Idle	
10	x1	1	Select DR	DR Path: Select Extradebug register to Read
11	x1	0	Capture DR	
12	x1	0	Shift DR	Shift In Read bit(1'b1) + Register Address
13	x3	0	Shift	
14	x1	1	Exit1	
15	x1	1	Update	Decode the 4 bits shifted in
16	x1	0	Run-Test/Idle	
17	x1	1	Select DR	2nd DR Path: ExtraDebug Read access
18	x1	0	Capture DR	Read the ExtraDebug register
19	x1	0	Shift DR	Shift out the captured value
20	x39	0	Shift	
21	x1	1	Exit1	
22	x1	1	Update	
23	x1	0	Run-Test/Idle	

7.10 i.MX27 JTAG Restrictions

TRST_b must be externally asserted to force the selection of ARM926 Platform TAP or i.MX27 JTAG Controller. During POR_B assertion, ARM926 Platform TAP is selected.

If TMS either remains unconnected or connected to VDD, then the TAP controller cannot leave the Test-Logic-Reset state regardless of TCK.

Chapter 8
Bootstrap Mode Operation

8.1 Introduction

The bootstrap program is a small program that resides in the internal ROM of the i.MX27 processor. It is activated when the BOOT[3:0] selection pins are set to 4'b0000 or if there is any exception during the HAB checking during boot-up. The bootstrap operation handles the commands from either USB or UART1 to establish a channel to interface the i.MX27 processor's hardware and the external machine such as PC. It provides the following functions.

1. For HAB Enable-type silicon, it downloads authenticated binary image code to memory so as to execute in run time or perform Flash update.
2. For HAB Disable-type silicon, it downloads binary image code to memory as to execute in run time or perform Flash update.

For HAB Enable-type silicon, a shell provides essential information such as the signatures, optimized commands, and the authenticated binary image to the iROM to validate before the core to execute.

8.2 UART/USB Configuration

The configuration for RS 232 is using baud rate 115200, 8 Data bits, No Parity, 1 Stop bits, and No Flow Control. The Configuration for USB is for Control Endpoint 0 with Max Packet Size equal 8 byte. Bulk IN at Endpoint 2 with Max Packet Size equal 64 bytes, Bulk OUT at Endpoint 1 with Max Packet Size equal 64 bytes.

NOTE

Current ROM code only supports Full Speed transmission over Full Speed transceiver(ISP1301 and Atlas) and High Speed USB transceiver (ISP 1504 or the like). The ROM code does not support high-speed transmission over high-speed USB transceiver (ISP 1504 or the like).

8.3 Enter Bootstrap Mode Configuration

The i.MX27 processor enters bootstrap mode under the following conditions:

1. BOOT[3:0] is selected bootstrap mode

   or
2. For HAB Enabled type of silicon: HAB authentication fails when booting from Flash (for example NAND Flash, NOR Flash)

Refer to Chapter 4, "System Control" for the details of bootstrap mode configuration and operation.

8.4 Bootstrap Flow

The overall flow of the bootstrap program is shown in Figure 8-1.

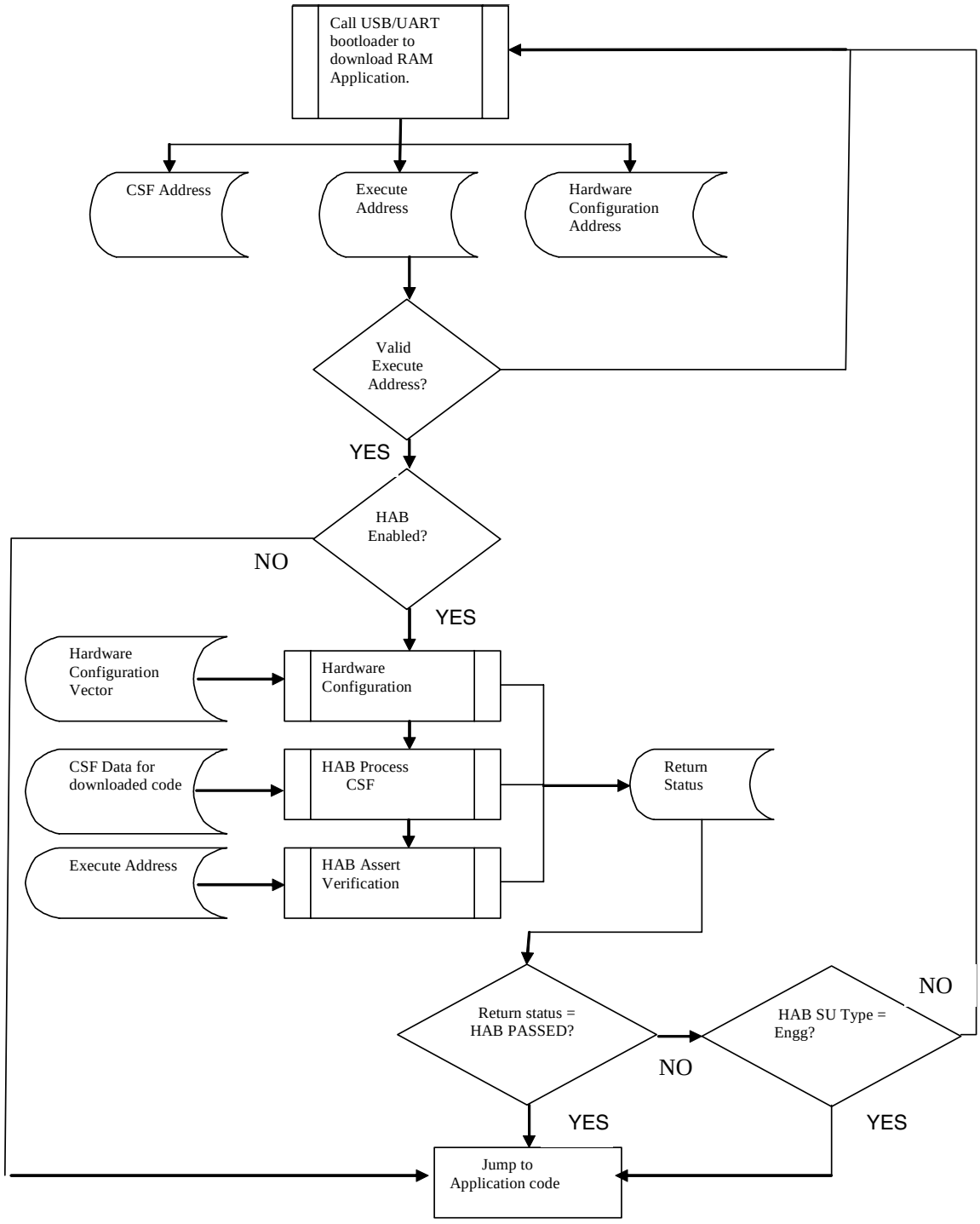


Figure 8-1. Flow Diagram for Bootstrap Mode

8.4.1 Bootstrap Protocol and Definition

In this section, bootstrap protocol and the command, response definition is defined. For the i.MX27 processor's boot-up sequence, refer to System Boot. For the CSF, HW Configuration, Image definition, refer to the High Assurance Boot (HAB).

8.4.1.1 Synchronization Operation

When bootstrap is firstly entered, the status of the iROM can be obtained by issuing the command shown in Figure 8-2.

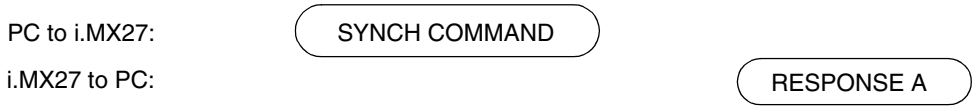


Figure 8-2. iROM Status Command

The SYNC COMMAND consists of 16 bytes using the format shown in Table 8-1.

Table 8-1. Synch Command Response Definition

Header (2 bytes)	Address (4 bytes)	Format (1 byte)	Bytecount (4 bytes)	Data (4 bytes)	End (1 byte)
0505	00000000	00	00000000	00000000	00

RESPONSE A is 4 bytes long using the format shown in Table 8-2.

Table 8-2. Response A Definition

Byte 0	Byte 1	Byte 2	Byte 3
STATUS CODE	STATUS CODE	STATUS CODE	STATUS CODe

8.4.1.2 Write Register Operation

To write to a register through bootstrap, requires a specific protocol. After the command is sent from PC to MX27 processor, two responses are returned from MX27. One is used to indicate the type of silicon (either HAB enable or disable), the other is used to indicate whether the write operation is successful.

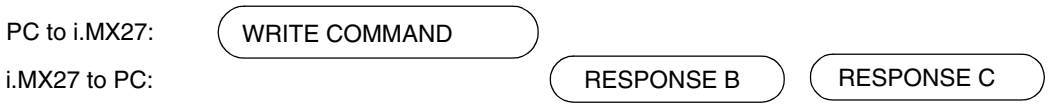


Figure 8-3. Write Register Command

WRITE COMMAND is 16 bytes long using the format shown in Table 8-3.

Table 8-3. Write Register Command Definition

Header (2 bytes)	Address (4 bytes)	Format (1 byte)	Bytecount (4 bytes)	Data (4 bytes)	End (1 byte)
0202	Address to be written	Format to be written (08: byte access 10: halfword access 20: word access)	00	Data to be written to the register	00

RESPONSE B indicates type of silicon. It is composed of 8 bytes using the format shown in Table 8-4.

Table 8-4. Response B Definition

	Byte 0	Byte 1	Byte 2	Byte 3
HAB Disable/ Development	56	78	78	56
HAB Enable	12	34	34	12

RESPONSE C indicates the success of a write operation as shown in Table 8-5.

Table 8-5. Response C Definition

Byte 0	Byte 1	Byte 2	Byte 3
12	8A	8A	12

Remarks: For HAB enabled silicon, users can only write the following range of registers:

1. System Control registers (address: 0x10027800-0x10027870)
2. Phase-Locked Loop, Clock and Reset Controller registers (address: 0x10027000–0x10027034)
3. NFC registers (address: 0xD8000000–0xD8000FFF)
4. SDRAMC registers (address: 0xD8001000–0xD8001FFF)
5. WEIM registers (address: 0xD8002000–0xD8002FFF)
6. Memory area of CS0, CS1, CS2, CS3, CS4, CS5, CSD0, and CSD1 (address: 0xA0000000–0xD7FFFFFF)

8.4.1.3 Download Operation

Memory is initialized before downloading a binary file to it. The following command can be used:

Figure 8-4. Download Command

The DOWNLOAD COMMAND is 16 bytes long using the formats shown in Table 8-6.

Table 8-6. Download Command Definition

	Header (2 bytes)	Address (4 Bytes)	Format (1 byte)	Bytecount (4 bytes)	Data (4 bytes)	End (1 byte)
CSF	0404	Start address where the binary data is to be downloaded	00	Number of byte to be written in Hex	Start address in memory where data is to be written	CC
HWC	0404	Start address where the binary data is to be downloaded	00	Number of byte to be written in Hex	Start address in memory where data is to be written	EE
Image file	0404	Start address where the binary data is to be downloaded	00	Number of byte to be written in Hex (max 0x1F0000)	Start address in memory where data is to be written	00
Image file	0404	Start address where the binary data is to be downloaded	00	Number of byte to be written in Hex	Start address in memory where data is to be written	AA

RESPONSE B indicates the type of silicon. It is 8 bytes long using the format shown in Table 8-7.

Table 8-7. Response B Silicon Type Definition

	BYTE 0	BYTE 1	BYTE 2	BYTE 3
HAB Disable/ Development	56	78	78	56
HAB Enable	12	34	34	12

After the RESPONSE B is received by the MX27 processor, the attached PC can start to download the binary data to MX27 until all the BYTECOUNT is downloaded. Each time the Image File is downloaded through HEADER (0404), the maximum data to be download is 0x1F0000. Thus, if the Image File size is greater then 0x1F0000, it will send the command repeatedly with END (0x00). After all the data is downloaded, PC must send a DOWNLOAD command with END (AA) to the target execution address.

8.4.1.4 Bootstrap End Indication Operation

After all the bootstrap operations are completed, the i.MX27 processor will send RESPONSE D to PC after the Application Pointer was sent to indicate bootstrap was completed. After RESPONSE D, it will enter iROM to perform the authentication check for HAB enable silicon or to execute the image for HAB disable/development silicon.

Figure 8-5. Bootstrap End Indication Operation Diagram

RESPONSE D is indicates the success of the write operation as shown in Table 8-8.

Table 8-8. Bootstrap End Indication Operation Diagram

BYTE 0	BYTE 1	BYTE 2	BYTE 3
88	88	88	88

Book II:
Applications Processors' Core and Peripherals

Introduction

Book II comprises detailed information on the applications processors' core and peripherals. Book II includes the following chapters.

Book II, Part 1: ARM9 Core and Interrupts

Book II, Part 2: Security

Book II, Part 3: External Interfaces

Book II, Part 4: Connectivity Peripherals

Book II, Part 5: Timer Peripherals

Book II, Part 6: System Control Peripherals

Book II, Part 7: Multimedia Peripherals

Book II, Part 1:
ARM9 Core and Interrupts

Introduction

This part provides an overview of the modules that make up the ARM9 core and interrupts.

Chapter 9, “ARM9 Platform,” on page 9-3

Chapter 10, “ARM926EJ-S Interrupt Controller (AITC),” on page 10-1

ARM9 Platform

The ARM9 Platform consists of the ARM926EJ-S processor, ETM9, ETB9, a 6 x 3 Multi-Layer AHB crossbar switch (MAX), and a “primary AHB” complex. The instruction bus of the ARM926EJ-S processor (I-AHB) is connected directly to MAX Master Port 0. The data bus of the ARM926EJ-S processor (D-AHB) is connected directly to MAX Master Port 1. All four alternate bus master interfaces are connected to MAX Master Ports 2-5. The three slave ports of the MAX are AHB-Lite compliant buses. Slave Port 0 is designated as the “primary” AHB. The primary AHB is internal to the platform and has six slaves connected to it: the AITC interrupt module, the MCTL memory controller, two AIPI peripheral interface gaskets, and a ROMPATCH module. Slave ports 1 and 2 of the MAX are referred to as “secondary” AHBs. Each of the secondary AHB interfaces is only accessible off platform.

ARM936EJ-S Interrupt Controller (AITC)

The ARM926EJ-S Interrupt Controller (AITC) is a 32-bit peripheral which collects interrupt requests from up to 64 sources and provides an interface to the ARM926EJ-S core. The AITC includes software controlled priority levels for normal interrupts.

Chapter 9
ARM9 Platform

9.1 Introduction

The ARM9 Platform consists of the ARM926EJ-S processor, ETM9, ETB9, a 6 x 3 Multi-Layer AHB crossbar switch (MAX), and a “primary AHB” complex. The instruction bus of the ARM926EJ-S processor (I-AHB) is connected directly to MAX Master Port 0. The data bus of the ARM926EJ-S processor (D-AHB) is connected directly to MAX Master Port 1. All four alternate bus master interfaces are connected to MAX Master Ports 2–5. The three slave ports of the MAX are AHB-Lite compliant buses. Slave Port 0 is designated as the “primary” AHB. The primary AHB is internal to the platform and has six slaves connected to it: the AITC interrupt module, the MCTL memory controller, two AIPI peripheral interface gaskets, and a ROMPATCH module. Slave ports 1 and 2 of the MAX are referred to as “secondary” AHBs. Each of the secondary AHB interfaces is only accessible off platform.

The four alternate bus master ports on the ARM9 Platform, which are connected directly to master ports of the Multi-Layer Crossbar Switch (MAX), are designed to support connections to multiple AHB masters external to the platform. An external arbitration and AHB control module is needed if multiple external masters are desired to share an ARM9 Platform alternate bus master port. However, the alternate bus master ports on the platform support seamless connection to a single master with no external interface logic required.

A PAHBMUX module (primary AHBMUX) performs address decoding, read data muxing, bus watchdog, and other miscellaneous functions for the primary AHB within the platform. A clock control module (CLKCTL) is provided to support a power conscious design methodology as well as implementation of several clock synchronization circuits.

The JAM (Just Another Module) implements the platform’s general purpose registers and also contains miscellaneous platform logic.

A block diagram of the ARM9 Platform can be seen in Figure 9-1.

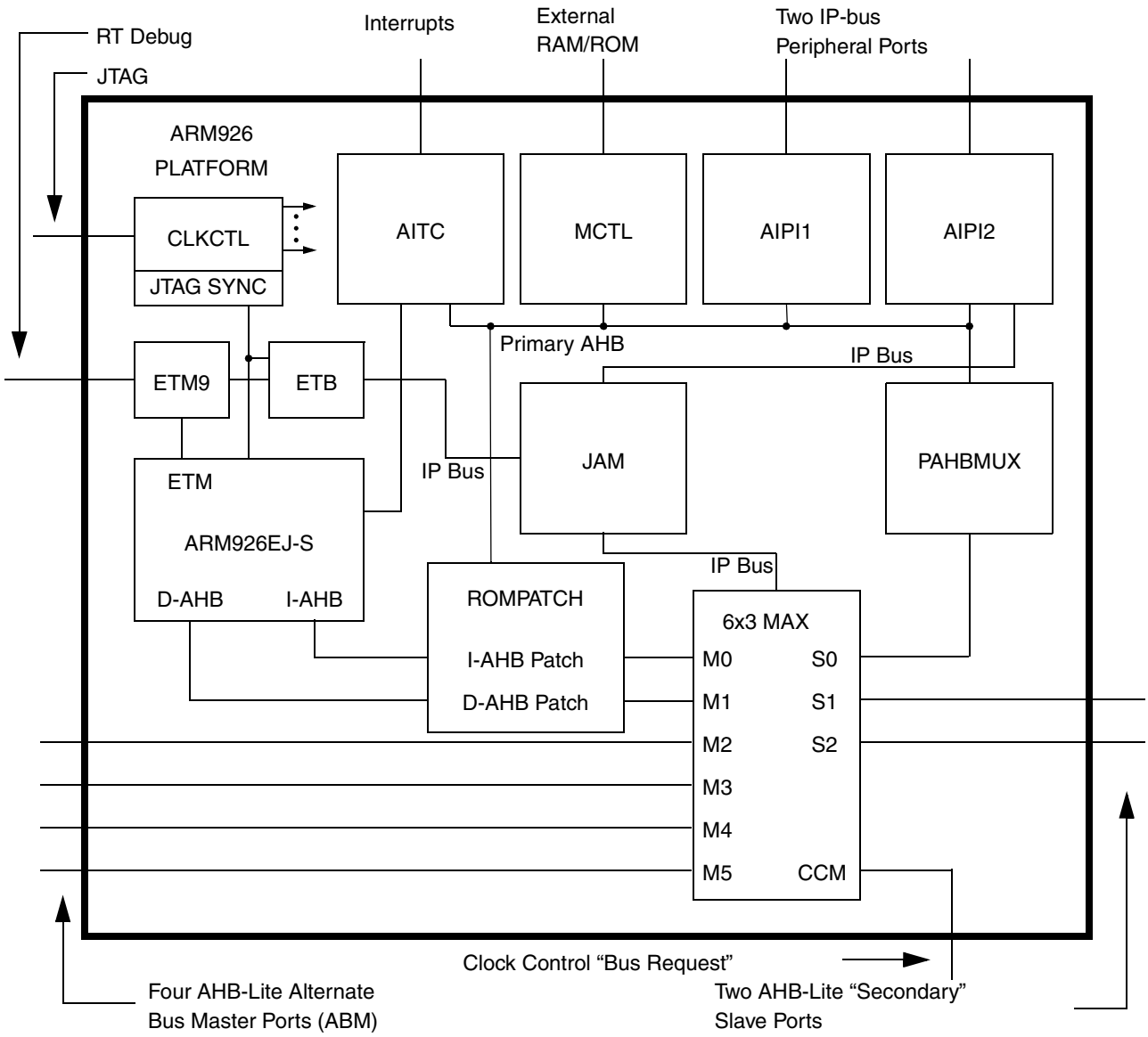


Figure 9-1. ARM9 Platform Block Diagram

9.1.1 Design Methodology Summary

Other than the CPU and ETB memories, this platform is a fully synthesizable, mux-D, rising edge clock based design. DFT goals are for 95% fault coverage and the design includes BIST engines for all memories implemented on the ARM926EJ-S, the ETB module and RAM or ROM controlled by the MCTL module. The platform will be designed with a DFT friendly scan wrapper to allow for deeply embedded integrations.

9.1.2 Performance Characteristics

The ARM9 Platform will support two main clocks—clk and hclk. clk will be connected to the ARM926EJ-S processor, ETM9, ETB and the Clock Control Module only. The remainder of the platform will be connected to hclk. The I-AHB and D-AHB from the ARM926EJ-S BIU module will run at the hclk frequency and will be controlled by a single hclken input—that is, the two buses cannot be decoupled. Refer to Section 9.6, "Platform Clocking" for more information on ARM926EJ-S clock control.

9.1.2.1 Performance Target

The ARM9 Platform team has committed to making an operating frequency of 266MHz characterized at the 1.1V, 105C WCS 3 sigma cmos90lp standard Vt library and 400MHz characterized at the 1.45V, 105C WCS 3 sigma cmos90lp standard Vt library. This is the level of performance required to meet functional requirements. Due to the increased leakage of the cmos90lp library, well back-biasing will be employed along with other standard low power clocking methodologies.

9.2 ARM9 Platform Sub-Modules

The sub-modules of the platform are listed below along with short functional descriptions.

9.2.1 ARM926EJ-S Processor

The ARM926EJ-S (ARM926) is a member of the ARM9 family of general-purpose microprocessors targeted at multi-tasking applications. The ARM926 supports the 32-bit ARM and 16-bit THUMB instructions sets. The ARM926 includes features for efficient execution of Java byte codes. A JTAG port is provided to support the ARM Debug Architecture, along with associated signals to support the ETM9 real-time trace module. The ARM926EJ-S is a Harvard cached architecture including an ARM9EJ-S integer core, a Memory Management Unit (MMU), separate instruction and data AMBA AHB interfaces, separate instruction and data caches, and separate instruction and data tightly coupled memory (TCM) interfaces. The ARM926 co-processor, instruction TCM, and data TCM interfaces will be tied off within the ARM9 Platform and will not be available for external connection.

The ARM926EJ-S processor is a fully synthesizable macrocell with a configurable memory system. Both instruction and data caches will be 16 Kbytes on the platform. The cache is virtually accessed and virtually tagged. The data cached has physical tags as well. The MMU provides virtual memory facilities which are required to support various platform operating systems such as Symbian OS, Windows CE, and Linux. The MMU contains eight fully associative TLB entries for lockdown and 64 set associative entries. Refer to the ARM926EJ-S Technical Reference Manual for more information.

9.2.1.1 ARM926EJ-S Co-Processor Interface

The co-processor interface will not exit the ARM9 platform and will be tied off internally and synthesized away to improve routing congestion and timing.

9.2.1.2 TCM Interfaces

Both instruction and data tightly-coupled memory (TCM) interfaces will not exit the ARM9 platform and will be tied off internally and synthesized away to improve routing congestion and timing.

9.2.2 ARM9 Embedded Trace Macrocell and Embedded Trace Buffer

The ARM9 platform will include an ARM9 Embedded Trace Macrocell (ETM9) and Embedded Trace Buffer (ETB) supporting real-time instruction and data tracing. The ETM9/ETB external interface may run at the ARM926EJ-S clock frequency or at half the ARM926EJ-S clock frequency. The Embedded Trace Buffer is sized at 2048x 32 and can be used as general scratch pad memory when not being used for real-time tracing. This scratch pad memory is accessible via AIPI2 slots 27 and 28. The ETB registers can be accessed via AIPI2 slot 29. Refer to the ETM9 and ETB technical reference manuals for more information.

9.2.3 The 6 x 3 Multi-Layer AHB Crossbar Switch (MAX)

The ARM926EJ-S processor's instruction and data buses and all alternate bus master interfaces arbitrate for resources via a 6 X 3 Multi-Layer AHB Crossbar Switch (MAX). There are six (M0–M5) fully functional master ports and three (S0–S2) fully functional slave ports. The MAX is uni-directional. All master and slave ports are AHB-Lite compliant. See Section 9.9.1, "Definition of AHB-Lite" for an explanation of AHB-Lite.

The design of the crossbar switch allows for concurrent transactions to proceed from any master port to any slave port. That is, it is possible for all three slave ports to be active at the same time as a result of three independent master requests. If a particular slave port is simultaneously requested by more than one master port, arbitration logic exists inside the crossbar to allow the higher priority master port to be granted the bus, while stalling the other requestor(s) until that transaction has completed. The slave port arbitration schemes supported are fixed, programmable fixed, programmable default input port parking, and a round robin arbitration scheme.

The Crossbar Switch also monitors the ccm_br input (clock control module bus request) which request a bus grant from all four slave ports. The priority of ccm_br is programmable and defaults to the highest. Upon receiving bus grants for all four output ports, the ccm_bg output will assert. At this point, the clock control module can turn off hclk and be assured there are no outstanding AHB transactions in progress. Once the CCM is granted a port, no other master will receive a grant on that port until the CCM bus request (ccm_br) negates.

Brief descriptions below provide more detail on the MAX. For complete functionality, refer to the ARM9 Platform "Multi-Layer AHB Crossbar Switch" Module (MAX) specification.

9.2.3.1 MAX Configuration Registers

The Crossbar Switch has configuration and control registers accessible via the IP Bus (slot 31 of AIPI2). Programmable registers exist to control arbitration schemes, bus parking, as well as other crossbar bus switch functionality. Alternate master priority registers exists within the MAX module for each slave

output port. The alternate priority register can be selected for use by the internal arbitration logic by driving the sx_ampr_sel (x=0-2) input high.

A write-block sticky bit is implemented for those applications where it is desirable to prevent changes to the MAX registers after boot. Refer to the MAX module design specification for more details.

9.2.3.2 Master Ports

Master Port 0 of the MAX is connected directly to the ARM926EJ-S I-AHB. Master Port 1 of the MAX is connected directly to the ARM926EJ-S D-AHB. The other four master ports exit the platform and will be connected to external alternate bus masters. Multiple external masters may be attached to a single alternate bus master port via use of an external arbiter. See Section 9.9.3, "Single Master Seamless Connection to ABM Port" and Section 9.9.4, "Multiple External Masters Connection to ABM Port" for more information on how to connect either a single master or multiple masters to a single alternate master port.

Master Port priorities are determined by the MAX priority register bit settings. Refer to the MAX module design specification for more details.

9.2.3.3 Slave Ports

Slave port 0 through 2 are identical AHB-Lite buses. Slave Port 0 is designated as the "Primary AHB" bus, and is internal to the platform. Slave port 1 and 2 are identical "Secondary AHB" buses and are available external to the platform. See Section 9.9.6, "MAX AHB Slave Ports" for more details.

9.2.3.4 Debug Support

In addition to the JTAG, ETM9 and ETB9 interfaces, several internal ARM926EJ-S signals, several internal primary AHB signals as well as some internal signals from Master Ports 0 and 1, have been brought out of the platform. These signals, along with alternate bus master and secondary AHB signals already available on the top-level of the platform, enable the user to gain insight into the operation of the processor and the MAX. Specifically, it is possible to monitor these signals and determine which master currently owns each slave port. In addition, it is possible to determine which slave each master is targeting for its next request.

9.2.4 ARM Interrupt Controller (AITC)

The ARM9 platform's interrupt controller is called the AITC and is connected to the primary AHB as a slave device. It will generate the normal and fast interrupts to the ARM926EJ-S processor. The AITC also supports hardware assisted vectoring.

NOTE

If hardware assisted vectoring is used, the vector space must be marked non-cacheable. Since the vector is dynamically "jammed" in by the AITC during a vector fetch, there would be no way to do this if cached.

Refer to the AITC specification for more details.

9.2.5 Memory Controller and BIST Engine (MCTL)

The MCTL module interfaces the primary AHB to RAM and ROM. BIST engines are provided for both RAM and ROM.

9.2.5.1 RAM

The ram_connect input on the ARM9 Platform must be tied high if RAM exists on the MCTL RAM interface. The MCTL module supports a minimum of 1 Kbyte of RAM and a maximum of 1 Mbyte. Non power-of-two sizes between 1 Kbyte and 1 Mbyte are supported by strapping the ram_max_addr[9:0] inputs, which correspond to the primary AHB's haddr[19:10]. The ram_wait input should be tied high at integration time if a wait state is required to make read data timing on RAM accesses (writes will still be zero wait state). The RAM interface will support single clock edge non late-write style compiled memories, and will implement an internal write buffer to mimic the late-write capability for improved performance. A configurable BIST engine is provided.

Refer to the ARM9 Platform MCU Memory Controller specification for more detail on the RAM controller design. Refer to Table 9-4 for the RAM's location within the platform's memory map.

9.2.5.2 ROM

The rom_connect input on the ARM9 Platform must be tied high if ROM exists on the MCTL ROM interface. The MCTL module supports a minimum of 1 Kbyte of ROM and a maximum of 4 Mbyte. Non power-of-two sizes between 1 Kbyte and 4 Mbyte are supported by strapping the rom_max_addr[11:0] inputs, which correspond to the primary AHB's haddr[21:10]. The rom_wait input should be tied high at integration time if a wait state is required to make read data timing on ROM accesses. A configurable BIST engine is provided.

9.2.5.2.1 ROM Addressing

The first 16 Kbytes of ROM will always be mapped starting at haddr[31:0]=32'h0000_0000. Any ROM larger than 16Kbyte will have the remainder of its space mapped starting at haddr[31:0]=32'h0040_4000. Any ROM size smaller than 16 Kbyte, will be aliased throughout the 16 Kbyte region. Accesses to the "hole" between these two regions will be terminated with an ERROR response by the MCTL.

Refer to the ARM9 Platform MCU Memory Controller specification for more detail on the ROM controller design.

9.2.6 AHB IP Bus Interface (AIPI)

There are two AHB IP Bus Interface (AIPI) modules that interface the primary AHB to two external IP Bus interfaces. The IP Bus interfaces and their peripherals will conform to the IP Bus Rev 2.0/3.0 specification. Each AIPI module supports up to 31 peripherals (the AIPI configuration registers consume the slot 0), however the ARM9 Platform will only support 48 external peripherals as shown in Table 9-1.

Table 9-1. AIPI ARM9 Platform IP Bus Support

AIPI #	Module Slot(s)	Use
1	0	AIPI1 Configuration Registers
1	1–31	Off platform IP Bus module support
2	0	AIPI2 Configuration Registers
2	1–17	Off platform IP Bus module support
2	18–26	Reserved
2	27	On platform ETB Register Interface
2	28	On platform ETB RAM Interface
2	29	On platform ETB RAM Interface
2	30	On platform JAM Interface
2	31	On platform MAX Interface

Refer to the ARM9 Platform AIPI specification for more details on the operation of the AIPI.

9.2.7 PAHBMUX–Primary AHB Mux

The PAHBMUX module is responsible for address decoding for the primary AHB module selects. In addition, the PAHBMUX module will perform the primary AHB read data muxing, the primary AHB watchdog, and other miscellaneous functions.

Refer to the ARM9 Platform AHBMUX design specification for more detail.

9.2.8 ROMPATCH

The ROMPATCH will sit on the ARM926EJ-S I-AHB and D-AHB interfaces which are connected to MAX Master Ports 0 and 1. This location will allow for patching of both internal and external memory addresses on both ARM926EJ-S processor buses. The registers of the ROMPATCH will be programmed via the Primary AHB. The ROMPATCH can be used to patch source code or data tables. The ROMPATCH supports 32 patches.

9.2.8.1 External Boot

An external boot feature exists in the ROMPATCH module which allows patching of the reset vector fetch (address = 32’h0000_0000) if the boot_int signal is negated. This mechanism will cause the ARM926EJ-S to, in effect, fetch the reset vector from the address indicated by the ext_boot_addr[31:2] inputs.

Refer to the ARM9 Platform ROMPATCH design specification for more details.

9.2.9 Clock Control Module (CLKCTL)

The CLKCTL module performs block level clock gating, ARM926EJ-S JTAG synchronization requirements, as well as other miscellaneous clock control for the platform. Refer to the ARM9 Platform CLKCTL design specification for more detail on design implementation.

9.2.10 JAM

The JAM ("Just Another Module") implements miscellaneous logic with the platform. Functionality within the JAM includes IP Bus #2 read data muxing, gating of the AHB debug signals in order to save power, and an IP Bus interface for accessing ARM9 Platform general purpose registers. The IP Bus interface on the JAM populates AIPI2, slot 30. The IP bus registers implemented in the JAM are shown in Table 9-2.

Table 9-2. JAM IP Bus General Purpose Registers

Primary haddr	Register Name	Type	Implementation
32'h1003_E000	ARM9P_GPR0	Write/Read	{30'h0, etb_reg_clken, ahb_dbg_en}
32'h1003_E010	ARM9P_GPR4	Read Only	tapid[31:0]

The registers may be aliased throughout the AIPI2 slot 30 location; however, the registers should only be accessed at the above listed addresses. Attempts to access the registers at aliased locations may result in an error response. Additionally, attempts to access the registers in user mode or in non-word sizes will result in an error response. Writes to the read only ARM9P_GPR4 register are ignored and will not cause an AHB transfer error.

ARM9P_GPR0 bit [1], etb_reg_clken, is reset to zero on power-up and disables the clocks to the ETB for non debug purposes. When set to one, etb_reg_clken will enable clocks to the ETB such that the memory in the ETB can be used as general scratch-pad memory.

ARM9P_GPR0 bit [0], ahb_dbg_en, is reset to zero on power-up and disables the AHB debug related signals from toggling in order to save power. When set to a one, ahb_dbg_en enables the following top-level platform AHB related signals to toggle for platform debug:

- I-AHB: dbg_iahb_hready, dbg_iahb_htrans1, dbg_iahb_haddr[31:29]
- D-AHB: dbg_dahb_hready, dbg_dahb_htrans1, dbg_dahb_haddr[31:29]
- P-AHB: dbg_dahb_hready, dbg_dahb_htrans1, dbg_dahb_hmaster

These signals, along with signals available on the alternate bus master and secondary AHB ports, can be used to gain insight into the functionality of the MAX.

ARM9P_GPR4 is provided for software to determine the version of the platform. These bits correspond to the static state of the tapid[31:0] signals which include the tapid_ver[3:0] platform inputs. See Section 9.4, "JTAG ID Register" for more details.

9.2.11 Test Wrapper

The ARM9 Platform's test architecture is composed primarily of two functions: scan and BIST. The test module (ARM926P_TEST) includes a test control unit which decodes primary test mode input signals and places the platform into various test modes including scan, ac path testing, BIST, and safe state. These test modes support the ability to test a deeply embedded platform.

Refer to the ARM9 Platform DFT specification for more information.

9.3 ARM9 Platform Hierarchy

The first two levels of the ARM9 Platform design hierarchy are shown in Figure 9-2.

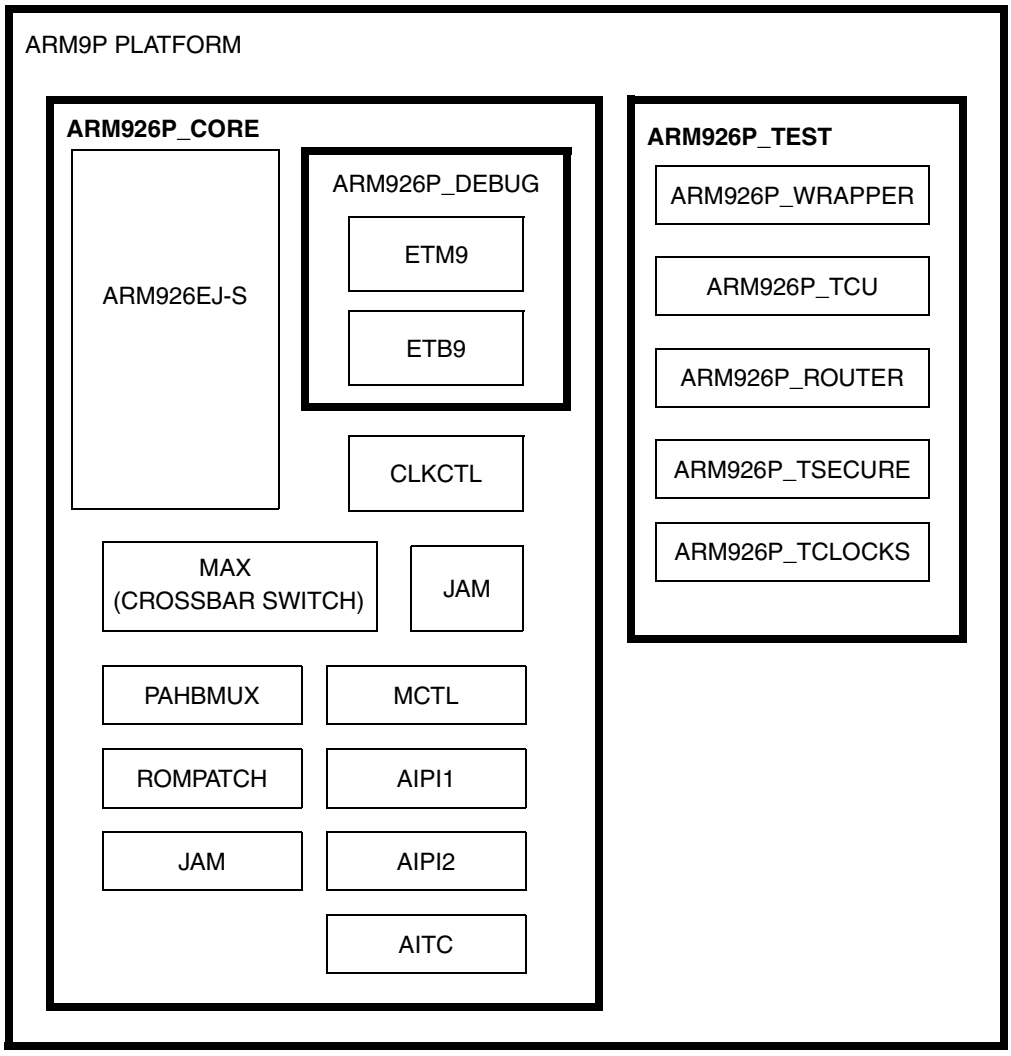


Figure 9-2. ARM9 Platform Hierarchy

9.4 JTAG ID Register

The ARM926EJ-S processor has a 32-bit input bus which corresponds to the JTAG ID register. This 32-bit register is defined fields as shown in Table 9-3. ARM requires bits [31:12] be set in accordance with their general rules such that Multi-ICE can auto-detect the device type.

Table 9-3. ARM926EJ-S JTAG ID Register Definition

tapid[31:28]	tapid[27:12]	tapid[11:8]	tapid[7:1]	tapid[0]
Version	Part Number	Manufacturing ID		1'b1
4'b0	16'h7926	tapid_ver[3:0]	7'b001_0000	1'b1

9.5 System Memory Map

haddr[31:29] on the MAX master ports are decoded determine which slave port has been selected. Only three bits are used in order to keep output port decode time to a minimum. Table 9-4 shows a simplified breakdown of the eight 512 Mbyte regions decoded within the 4 Gbyte address space.

Table 9-4. Upper Address Bit Decode

haddr[31:29]	Size	Use
3'b000	512 Mbyte	Primary AHB—AIPI1, AIPI2, AITC, MCTL (ROM), ROMPATCH
3'b001	512 Mbyte	Reserved
3'b010	512 Mbyte	Reserved
3'b011	512 Mbyte	Reserved
3'b100	512 Mbyte	Secondary AHB Slave Port 1
3'b101	1 Gbyte	Secondary AHB Slave Port 2
3'b110		
3'b111	512 Mbyte	Primary AHB–MCTL (RAM)

9.5.1 ARM9 Platform Memory Map

Table 9-5 shows the complete ARM9 Platform memory map.

Table 9-5. ARM9 Platform Memory Map

Address Range	Size	Use
0000_0000–0000_3FFF	16 Kbyte	ROM: First 16Kb (Primary AHB)
0000_4000–0040_3FFF	4 Mbyte	Reserved
0040_4000–007F_FFFF	4 Mbyte–16 Kbyte	ROM: Exceeding 16Kbyte (Primary AHB)
0080_0000–0FFF_FFFF	256 Mbyte–8 Mbyte	Reserved
1000_0000–1000_0FFF	4 Kbyte	AIPI1 Control Registers (Primary AHB)
1000_1000–1001_FFFF	124 Kbyte	AIPI1 Peripheral Space (Primary AHB)

Table 9-5. ARM9 Platform Memory Map (continued)

Address Range	Size	Use
1002_0000–1002_0FFF	4 Kbyte	AIPI2 Control Registers (Primary AHB)
1002_1000–1003_1FFF	68 Kbyte	AIPI2 Peripheral Space (Primary AHB)
1003_2000–1003_AFFF	36 Kbyte	Reserved
1003_B000–1003_BFFF	4 Kbyte	AIPI2—ETB Registers (Primary AHB)
1003_C000–1003_CFFF	4 Kbyte	AIPI2—ETB RAM (Primary AHB)
1003_D000–1003_DFFF	4 Kbyte	AIPI2—ETB RAM (Primary AHB)
1003_E000–1003_EFFF	4 Kbyte	AIPI2—JAM (Primary AHB)
1003_F000–1003_FFFF	4 Kbyte	AIPI2—MAX (Primary AHB)
1004_000–1004_0FFF	4 Kbyte	AITC (Primary AHB)
1004_1000–1004_1FFF	4 Kbyte	ROMPATCH (Primary AHB)
1004_2000–1FFF_FFFF	256 Mbyte–280 Kbyte	Reserved
2000_0000–7FFF_FFFF	1.5Gbyte	Reserved
8000_0000–9FFF_FFFF	512 Mbyte	Secondary AHB Slave Port 1
A000_0000–DFFF_FFFF	1 Gbyte	Secondary AHB Slave Port 2
E000_0000–FFEF_FFFF	511 Mbyte	Reserved (Aliased RAM Space)
FFF0_0000–FFFF_FFFF	1 Mbyte	RAM (Primary AHB)

9.5.2 External Peripheral Space

AIPI1 supports 31 external peripherals starting at 32’h1000_1000. AIPI2 has three slots used internal to the ARM9 Platform and supports 17 external peripherals from 1002_1000–1003_EFFF.

NOTE

Care should be taken when programming the PSR of AIPI2 since slot 31 (MAX), slot 30 (JAM), slot 29 (ETB Registers), slot 28 (ETB RAM) and slot 27 (ETB RAM) will always be occupied and slots 26:18 will always be unoccupied.

9.5.3 External Boot

When the boot_int input signal is asserted, the ARM926EJ-S will boot internal from ROM on the Primary AHB. When boot_int is negated, the ARM926EJ-S reset vector fetch will essentially be routed to an address indicated by the ext_boot_addr[31:2] input pins. This vectoring is done by the ROMPATCH module, which monitors the I-AHB of the ARM926EJ-S and over-rides the reset vector fetch.

Refer to the ROMPATCH design specification for more detail on the external boot mechanism.

NOTE

When boot_int is negated and the ARM926EJ-S boots externally the ARM9 Platform is placed in an insecure state.

9.5.4 Memory Map Considerations

- Accesses to “Reserved” locations in Table 9-5 other than aliased RAM space will result in an AHB error response.
- Accesses to unsupported address locations through the MAX will result in an AHB error response and the access will not pass through the MAX.
- Accesses to address locations on the Primary AHB bus which do not map to a specific module will time-out in accordance with the bmon_timeout[2:0] inputs.
- Accesses to unimplemented locations within the AITC and ROMPATCH register space will be terminated without a bus-error. Writes will have no effect and reads will return all zeros.

9.6 Platform Clocking

This section will describe some of the clocking considerations within the ARM9 Platform. The circuits contained in the ARM9 Platform to address most of these issues will be implemented within the Clock Control Module (CLKCTL). However, there are some external clock control issues that will be discussed. Refer to the ARM9 Platform CLKCTL Module design specification for more detail.

9.6.1 ARM926EJ-S Clock Considerations

The ARM926EJ-S processor design uses a single clock, clk. In many systems, it will be desirable for the ARM926EJ-S processor to run at a higher frequency than the AHB system bus (which runs on hclk). To support this, ARM926EJ-S requires a separate AHB clock enable for each of the two bus masters. dhclken is used to signify the rising edge of hclk for the system in which the data BIU is the bus master. ihclken is used to signify the rising edge of hclk for the system in which the instruction BIU is the bus master.

Figure 9-3 shows the relationship between clk, hclk and dhclken/ihclken. The ARM9 Platform will provide a single hclken input pin that will be fed to both the dhclken and ihclken inputs on the ARM926EJ-S. If hclk and clk are the same frequency, the hclken input to the platform must be tied HIGH.

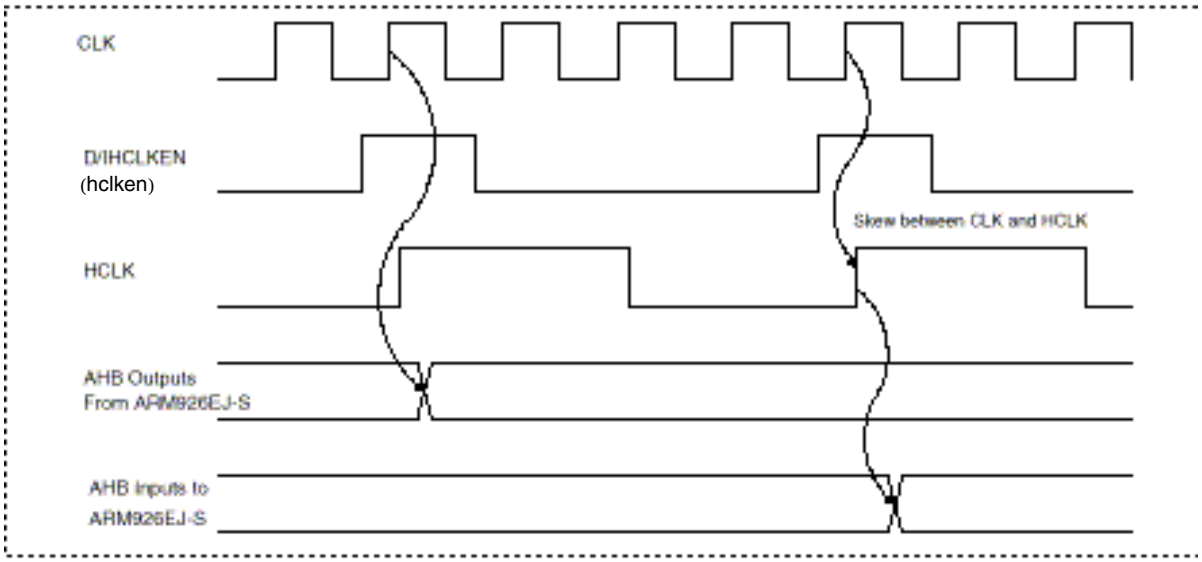


Figure 9-3. AHB Clock Relationship

clk and hclk must be synchronous and the skew between clk and hclk to the ARM9 Platform should be minimized. This will require some synchronization inside the chip clock control module. An example of this is provided in Figure 9-4. In the example, clk and hclk are completely asynchronous and clk must be much faster to sample the slower clock, otherwise a different scheme will be needed. Also, if clk and hclk are synchronous to each other by design, then a synchronizer may not be needed, but care must still be taken in aligning the rising edges of both clocks to the ARM9 Platform.

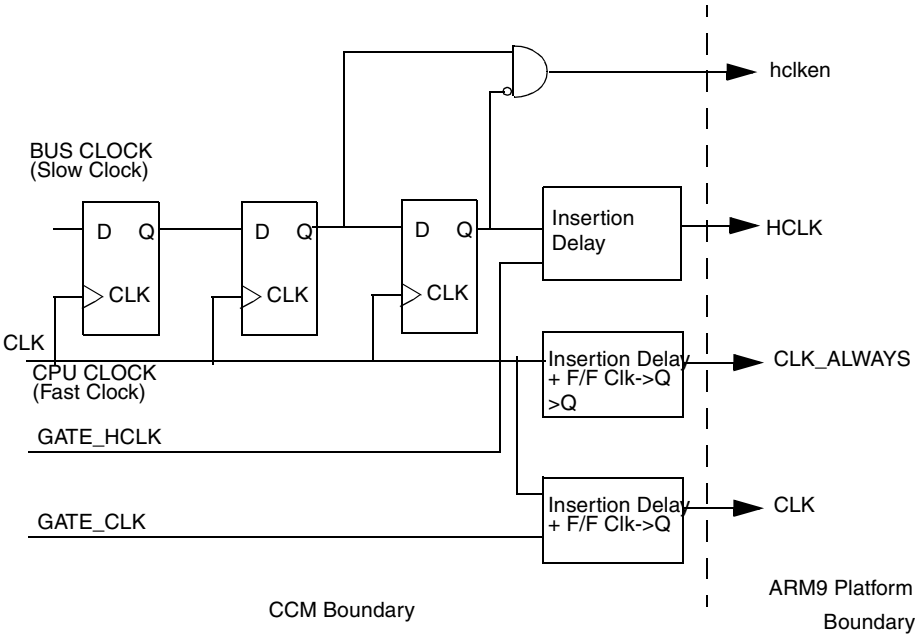


Figure 9-4. Example hclk to clk Synchronization When clk Is Faster

9.6.2 ARM926EJ-S JTAG Port Clocking Considerations

The ARM926EJ-S does not support direct connection to the JTAG interface. The JTAG interface must be synchronized to the clk domain. This synchronization will take place within the platform's CLKCTL module. Refer to the ARM9 Platform CLKCTL design specification for more detail.

9.6.2.1 JTAG_TCK

The jtag_tck clock must be less than 1/8 the frequency of the clk input in order for the JTAG port and synchronizer to function properly. Note that the frequency of clk can vary when executing low-power code. Therefore, care must be taken such that jtag_tck is less than 1/8 the lowest possible frequency of clk.

9.6.3 External Alternate Bus Master Interfaces

All four alternate bus master ports on the ARM9 Platform MUST have the AHB synchronized to hclk external to the platform. All alternate bus master AHB inputs and outputs to/from the ARM9 Platform will be synchronous to hclk.

9.6.4 External Secondary AHB Ports

Both secondary AHB ports inputs and outputs to and from the ARM9 Platform must be synchronous to the hclk and will run at the hclk frequency.

9.7 Platform Resets

This section will describe the various ARM9 Platform reset inputs. Figure 9-5 shows the reset paths within the ARM9 Platform.

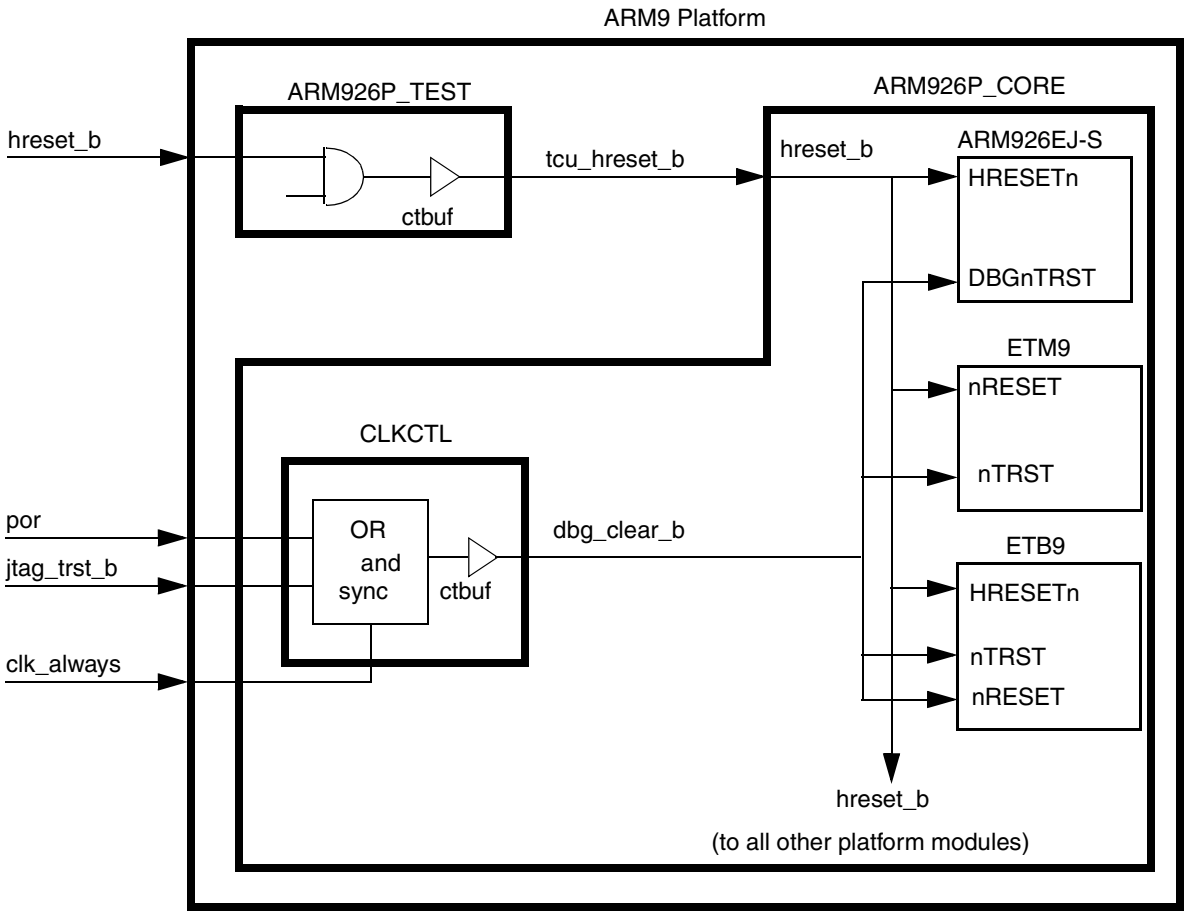


Figure 9-5. ARM9 Platform Resets

9.7.1 $\overline{\text{HRESET}}$

The hreset_b input is the asynchronous system reset for both the clk and hclk domains. It is gated with a test mode signal in the scan wrapper, and is then buffered for distribution throughout the platform. The hreset_b signal must satisfy the setup and hold time requirements relative to both clk and hclk rising edges.

9.7.2 POR and $\overline{\text{JTAG\_TRST}}$

The power-on-reset (POR) and the JTAG reset (jtag_trst_b) will be combined in the CLKCTL module to drive the dbg_clear_b signal to the ARM926EJ-S and ETM9 modules. The dbg_clear_b output of the CLKCTL module can be considered as the JTAG or debug reset of the platform. The dbg_clear_b signal will assert asynchronously when either POR or jtag_trst_b asserts, and will negate synchronously to clk (through a synchronizer). Refer to the CLKCTL module design specification for more detailed information.

9.8 Power Management

9.8.1 Register Level Clock Gating

A Synopsys power compiler will be used to implement clock-gating on all components of the ARM9 Platform. That is, under normal operating conditions, clocks internal to the platform will only be issued to registers or banks of flops that need a rising edge for proper functionality. Otherwise, the clocks will be held low.

9.8.2 Block Level Clock Gating

Clocks to individual modules within the platform will be enabled only when necessary. On the Primary AHB for example, the CLKCTL module will only enable hclk to a slave module when the current AHB access is addressed to that module. Slaves can also drive a signal to the CLKCTL if it requires its hclk to run for any other reason. See the CLKCTL module design specification for more detail.

9.8.3 External Clock Gating

The ARM926EJ-S processor may be put into a low-power state by the wait-for-interrupt instruction. This instruction switches the ARM926EJ-S into a low-power state until either an interrupt (nIRQ/nFIQ) or a debug request occurs. The switch into the low-power state is indicated by the assertion of the arm_standbywfi output signal. If arm_standbywfi is asserted then it is guaranteed that all ARM926EJ-S external interfaces will be in an IDLE state. The arm_standbywfi signal is intended to be used to shut down clocks to the other parts of the system, such as external coprocessors, which do not need to be clocked if the ARM926EJ-S is idle. NOTE: The ARM926EJ-S clk must NOT be stopped during wait-for-interrupt mode if an external debugger is connected to the JTAG port. An active clk is required to be able to write values into the ARM9EJ-S debug control register, which is required for a debugger to be able to force wait-for-interrupt mode to be exited. It should also be noted that the ARM926EJ-S needs clk to run in order for an interrupt to cause the negation of arm_standbywfi.

The JTAG synchronizer in the CLKCTL module needs to have an “always” clock running to it in order to, at any time, detect JTAG activity and thereby determine that a debugger is connected to the JTAG port. The presence of an active JTAG debugger will be detected by monitoring the JTAG TMS signal. After POR (or trst_b) assertion, a low state on TMS coincident with a rising-edge on TCK will transition the JTAG tap-controller from the test-logic-reset state to the run-test-idle state. The dbgen signal will be asserted, and held asserted, whenever the tap-controller is not in the test-logic-reset state. Once dbgen asserts, an active trst_b or POR is required to clear it (that is, once a debugger is detected to be connected, it is assumed to stay connected).

When asserted, the a9p_clk_off output of the platform will indicate to an external clock control module that clk and hclken should be turned off at the earliest opportunity. However, in order to assure that no alternate bus masters are in the middle of a transaction, the external clock control module must assert the ccm_br input (bus request) of the crossbar switch. This will request ownership of all AHB Output Ports. Once the bus grant is asserted (ccm_bg), the external clock control module is then free to gate off hclk as all transactions on both the primary and secondary AHBs will have completed.

Figure 9-6 shows a block diagram of how the clocks to the platform might be handled in a typical implementation.

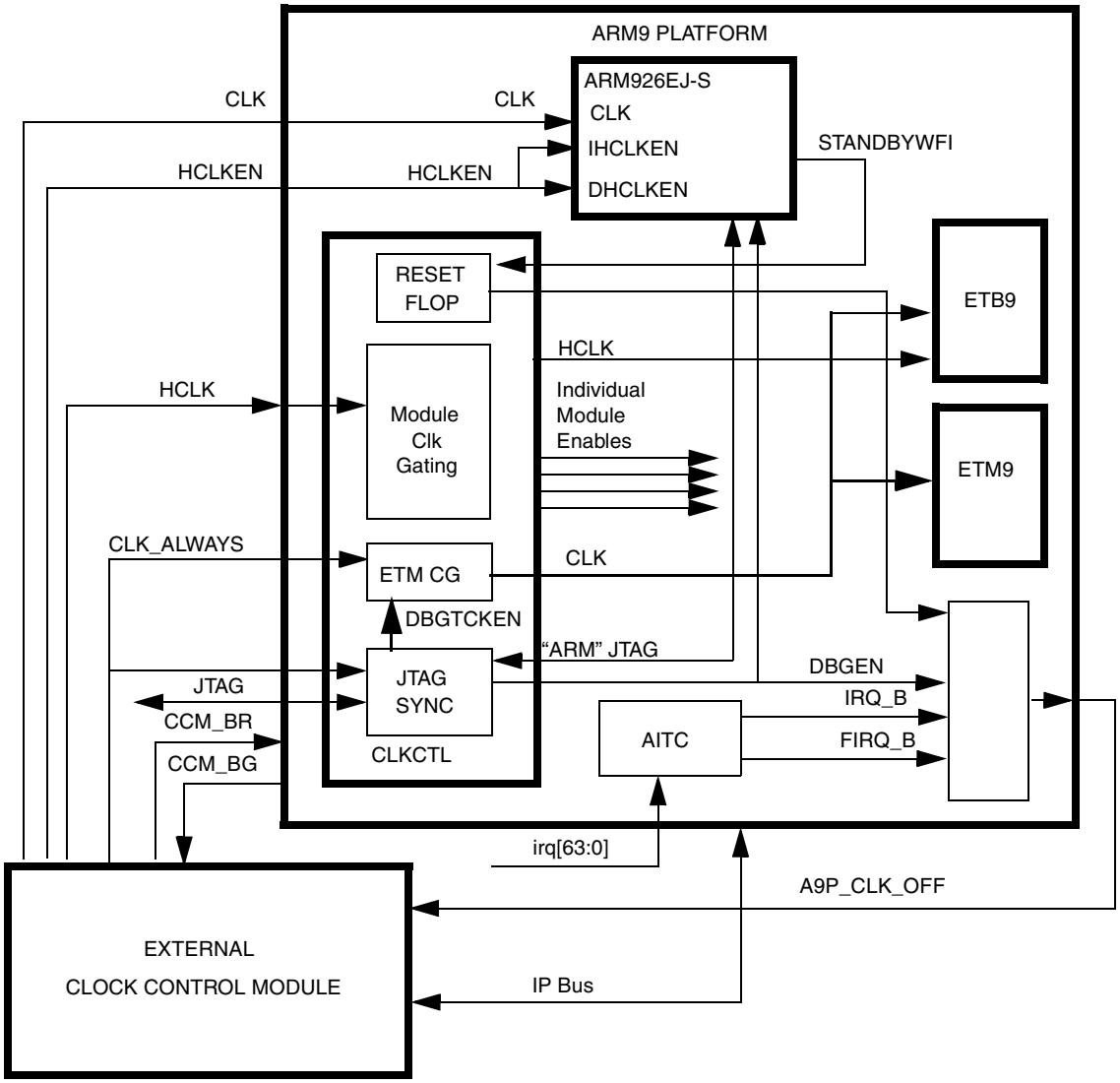


Figure 9-6. ARM9 Platform Clocking Strategy

9.8.4 Well Biasing

A well bias clamp enable input, wt_en, will be driven by an external clock control module to the ARM9 Platform. When asserted, V_{BB+} will be shorted to VDD and V_{BB-} will be shorted to GND.

9.9 Platform AHB Interfaces

This section will describe the major bus interfaces of the ARM9 Platform and the crossbar switch. A simple block diagram of the bus connections to the platform is shown in Table 9-1. A definition of

AHB-Lite, a functional description of the alternate bus master ports, and finally a description of the multi-layer crossbar switch slave ports follows.

9.9.1 Definition of AHB-Lite

All master and slave ports of the Multi-Layer AHB Crossbar switch are AHB-Lite compliant. Therefore, all AHBs connected externally to the ARM9 Platform must be AHB-Lite compliant.

The definition of “AHB-Lite” for the ARM9 Platform is as follows:

- AHB split and retry protocols are not supported within the ARM9 Platform. This means that all slaves connected to AHB-Lite ports (input or output) are prohibited from requesting a split or a retry. This also means there is only one response signal, hresp0.
- AMBA bus request and bus grant are not supported on the AHB-Lite interfaces.
- Bursts are supported. The default configuration of the Crossbar Switch (MAX) insures no early fixed length burst terminations due to the switch arbiter.

9.9.2 Alternate Bus Master Ports

There are four alternate bus master ports (ABM) on the ARM9 Platform which are connected directly to the Multi-Layer AHB Crossbar Switch. These four ABM interfaces are AHB-Lite compliant. Table 9-6 lists the ABM interface signals (“x” is equal to 2 through 5). All signals function as documented in the *AMBA Specification, Rev 2.0,* AMBA AHB chapter. It is assumed that the alternate bus masters are using the same hclk and hreset_b as the ARM9 Platform.

Table 9-6. Alternate Bus Master Interface Signal List

Pin List	Direction[1]	Description
mx_haddr[31:0]	Input	AHB Address Bus
mx_hmaster[3:0]	Input	AHB Master ID
mx_htrans[1:0]	Input	AHB Transfer Type
mx_hprot[3:0]	Input	AHB Access Protection Indicator
mx_hlock	Input	AHB Master Lock Indicator
mx_hmastlock	Input	AHB-Lite Master Lock Indicator
mx_hwrite	Input	AHB Access Write Indicator
mx_hsize[1:0]	Input	AHB Transfer Size
mx_hburst[2:0]	Input	AHB Access Burst Type
mx_hwdata[31:0]	Input	AHB Write Data
mx_hready_out	Output	AHB Termination/Take Indicator
mx_hrdata[31:0]	Output	AHB Read Data
mx_hresp0	Output	AHB Error Indicator

1 Direction is relative to the ARM9 Platform.

9.9.3 Single Master Seamless Connection to ABM Port

A single external master can connect seamlessly (no logic) to any of the four alternate bus master interfaces (all four ABM interfaces are identical). In this configuration, Table 9-7 lists the AHB signals which deserve special consideration.

Table 9-7. Single External Master Connections to an Alternate Bus Master Interface

AHB Signal	Connection
Master's hbusreq Output	If present leave unconnected
Master's hgrant Input	If present tie asserted (high)
Master's hready Input	Connect to ABM hready_out Output
Master's hlock Output	If present connect to ABM hlock Input[1]
Master's hmastlock Output	If present connect to ABM hmastlock Input[2]

1 If the Master does not have an hlock output, tie the ABM hlock input negated (low).

2 If the Master does not have an hmastlock output, tie the ABM hmastlock input negated (low).

NOTE

The alternate bus master must drive htrans = IDLE when not requesting the bus as the arbiter may be parked on that input port.

9.9.4 Multiple External Masters Connection to ABM Port

The four alternate bus master interfaces of the ARM9 Platform have been designed to support connection of multiple external AHB-Lite masters and slaves directly to the interface. Figure 9-7 shows the connection of two external masters to a ARM9 Platform alternate bus master interface. Note the location of the ARM9 Platform in the figure below and that only one of four ABM ports is shown.

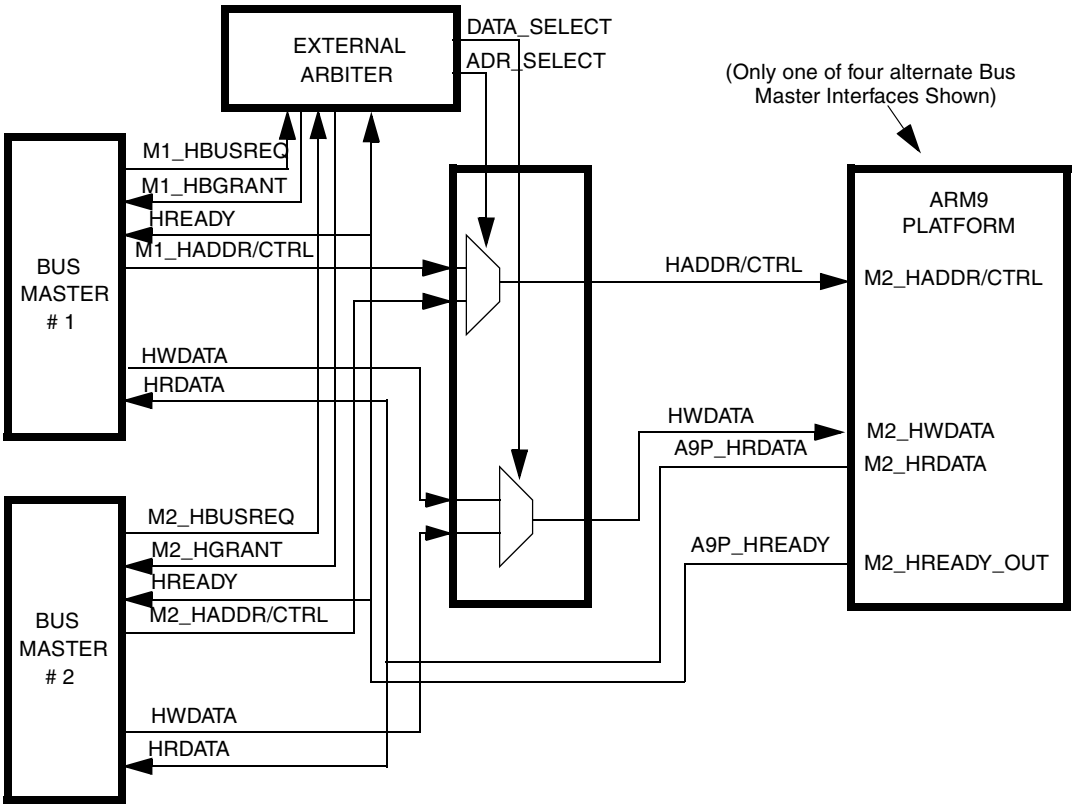


Figure 9-7. Example of Two External Masters Connected to an ABM Port

In the figure, two external masters, #1 and #2, arbitrate for control of the ABM interface on the ARM9 Platform. An external arbiter is required. The arbiter accepts the bus master's hbusreq signals, and responds to the masters with hgrants. The arbiter also controls the address/control and data muxing in the external AHBMUX module.

9.9.5 Alternate Bus Master Design Considerations

This section will discuss various issues which should be taken into account by engineers designing AHB masters to connect to the ARM9 Platform's alternate bus master interfaces.

9.9.5.1 Edge-Based Design

All alternate bus masters should be edge-based designs in order to meet the stringent timing imposed. Specifically, an AHB master's address and control information should be driven directly from the output of a flip-flop. Similarly, an AHB master's read data should go directly to a D-input of a flip-flop.

9.9.5.2 htrans [1:0]

Some important issues to remember about the AHB htrans signals:

- It is important that alternate bus masters drive htrans = IDLE when not requesting the bus. This is critical because the arbiter can grant the bus to a master even when the master is not requesting it (for example, a “parked” condition).
- Although the AMBA AHB specification does not require it, it is suggested that alternate bus masters assert htrans = NSEQ with the initial assertion of hbusreq. In those systems where only a single master is connected to an input port, the hgrant signal will tied high, and improved performance may result.
- It is highly recommended that alternate bus masters insert an IDLE cycle after any locked sequence to provide an opportunity for the arbitration to change before commencing further transfers.

9.9.5.3 hlock/hmastlock

The mx_hlock and mx_hmastlock ABM interface signal connections are dependent on whether there is a single external master or connection to an external arbiter. The following notes specify the connections:

- For single masters only, the mx_hlock input should be connected directly to the master’s hlock output. In this case, the mx_hmastlock input should be tied LOW. If the single master produces an hmastlock instead the mx_hmastlock input should be connected directly to the master’s hmastlock output. In this case, the mx_hlock input should be tied LOW.
- For multiple master connections on an ABM port, the mx_hmastlock input signal should be connected to the external arbiter’s hmastlock output. In this case, mx_hlock should be tied LOW.

In either case above, logic within the crossbar switch will insure the locked cycles’ functionality.

9.9.5.4 hmaster

Alternate bus masters external to the ARM9 Platform should be aware that four values of the hmaster field are used by bus masters internal to the platform. The reserved and available hmaster encodings are shown in Table 9-8.

Table 9-8. hmaster Encodings

hmaster	Use
4’h0	Reserved: MAX default
4’h1	Reserved
4’h2	Reserved: ARM926EJ-S I-AHB
4’h3	Reserved: ARM926EJ-S D-AHB
4’h4–4’hF	Available to External Bus Masters

9.9.5.5 hresp0—Bus Error

A slave two cycle ERROR response (hresp0 = HIGH) allows for a bus master to cancel the remaining transfers in a burst. However, this is not an AHB requirement, and it is acceptable for the master to continue the remaining transfers of the burst.

AHB error responses generated on accesses to cacheable or bufferable memory address on the I-AHB and D-AHB interfaces of the ARM926EJ-S are normally ignored by the processor. In the ROMPATCH module, a feature can be enabled which will, on the above described accesses, gate 0's onto hrdata[31:0] on data reads, and SWI opcodes onto hrdata[31:0] for instruction prefetches. At the same time, the ROMPATCH module will generate an abort which will guarantee entry into the ARM926EJ-S platform's abort exception handler. See the ARM9 Platform ROMPATCH design specification for more detail.

9.9.5.6 Unaligned Transfers

Alternate bus masters should not request unaligned transfers. That is, a word access to a non-word aligned address; and, a halfword access to a non-halfword aligned address should not be requested as neither transactions are supported by this platform. The transfers will complete as normal, however the lower order address bits will be ignored according to Figure 9-4 and Figure 9-5.

9.9.5.7 Alternate Bus Master Throttle Control

Alternate bus masters should be designed with programmable maximum burst lengths as well as programmable bus request interval timers. This will allow software to effectively "tune" the overall system for maximum throughput and efficiency.

9.9.5.8 Halt Request (ccm_br)

Care must be taken to ensure that the Halt Low Priority bit is not changing as the Clock Control Module's Halt request is asserted. This will result in unpredictable behavior. This can be avoided by not modifying this bit in the Slave General Purpose Control Register or in the Alternate Slave General Purpose Control register in software where Halt could be requested. Also, the Halt Low Priority bit should be programmed the same in both the Slave General Purpose Control Register and the Alternate Slave General Purpose Control Register, if it is likely the MAX can change between the General Purpose and Alternate registers during the time Halt could be requested.

Care should also be taken to ensure that the Clock Control Module's Halt request is not asserted until at least two clock cycles after the last locked access performed by any master connected to the MAX.

9.9.6 MAX AHB Slave Ports

Each slave port of the Multi-Layer Crossbar Switch is an AHB-Lite compliant bus. Brief functional descriptions and attributes of each output port are provided.

9.9.6.1 Slave Port 0—Primary AHB (Internal)

Slave Port 0 of the MAX is connected to the “Primary” AHB of the ARM9 Platform. The primary AHB is completely contained within the ARM9 Platform. A simplified block diagram of the Primary AHB components is shown in Figure 9-8.

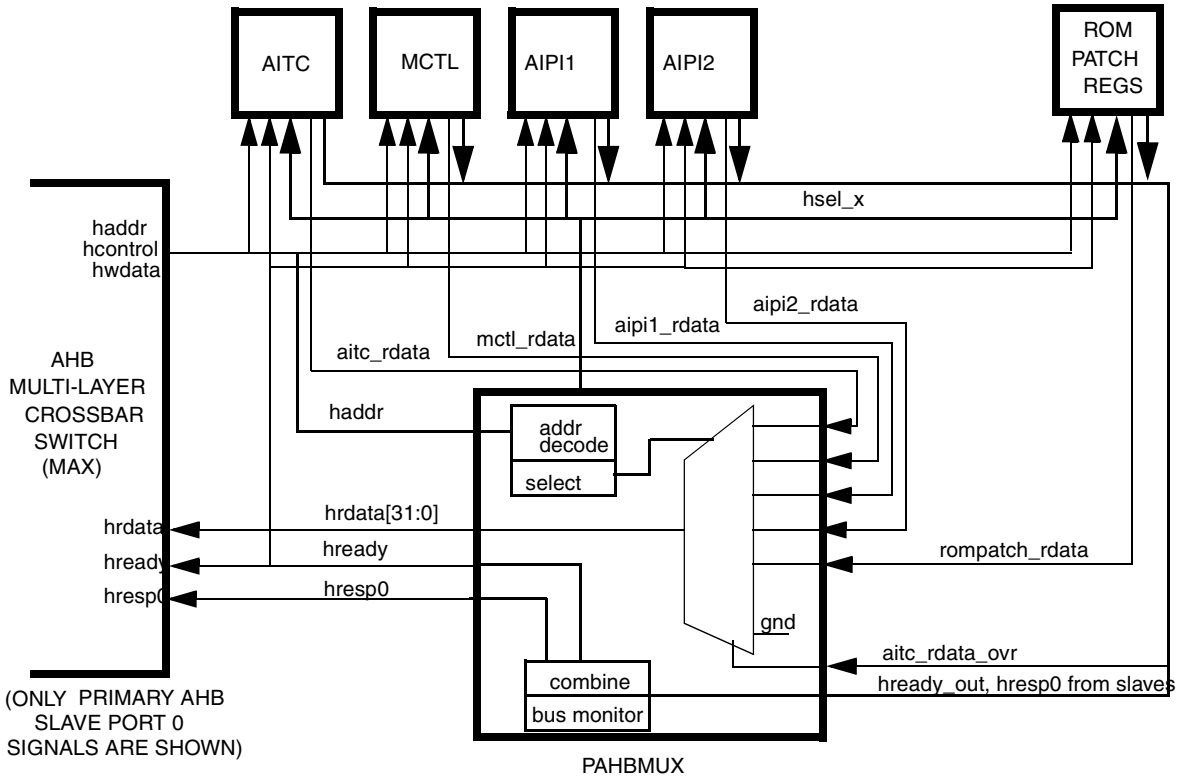


Figure 9-8. The Primary AHB

The primary AHB will have the AITC, AIPI (2), MCTL and ROMPATCH modules connected as slave devices. The PAHBMUX (Primary AHB mux) module is the glue that pulls the primary AHB and its components together. It will decode the primary AHB haddr lines and issue module selects to the slaves, combine the slave hready and hresp0 signals from the slaves into the single bus hready and hresp0, and mux the read data from the currently selected slave onto the bus hrdata lines. A bus monitor module inside of PAHBMUX will terminate timed-out bus transactions. In addition, the PAHBMUX will contain logic to terminate any IDLE cycles and issue the required assertion of hready following negation of the hreset_b.

9.9.6.1.1 Primary AHB Device Latencies

The latency of each slave device on the primary AHB can be found in Table 9-1. The clock latency number does not take into account the possible one clock arbitration delay of the MAX.

Table 9-1. Primary AHB Slave Device Latencies

Slave Device	Transaction Type	Latency
AITC	Register Access	1 clock
AIPI[1]	Writes Reads	3 clocks 2 clocks
MCTL	Memory Access	1 clock[2]
ROMPATCH	Register Access	1 clock

1 The latency listed for the AIPIs are best case and based on a zero wait state response from the IP bus target device. Each wait state the IP Bus target device adds will add one extra clock to the listed latency value.

2 Assumes ram_wait and rom_wait are negated.

9.9.6.2 Secondary AHB Slave Ports 1 and 2

Each of the secondary AHB slave ports are identical AHB-Lite compliant buses. It is envisioned that these ports will interface predominately to internal and external memory. However, it is possible to connect an external AIPI interface along with associated peripherals to these ports. The secondary port slave signals are list in Table 9-2 where “x” is equal to 1 or 2.

Table 9-2. Secondary AHB Interface Signal List

Pin List	Direction[1]	Description
sx_haddr[31:0]	Output	AHB Address Bus
sx_hmaster[3:0]	Output	AHB Master ID
sx_htrans[1:0]	Output	AHB Transfer Type
sx_hprot[3:0]	Output	AHB Access Protection Indicator
sx_hmastlock	Output	AHB-Lite Master Lock Indicator
sx_hwrite	Output	AHB Access Write Indicator
sx_hsize[1:0]	Output	AHB Transfer Size
sx_hburst[2:0]	Output	AHB Access Burst Type
sx_hwdata[31:0]	Output	AHB Write Data
sx_hready	Input	AHB Termination/Take Indicator
sx_hrdata[31:0]	Input	AHB Read Data
sx_hresp0	Input	AHB Error Indicator

1 Direction is relative to the ARM9 Platform.

9.9.7 Endian Modes

The ARM9 Platform will support both Big and Little Endian modes. The relevant signals to/from the ARM926EJ-S processor are shown in Table 9-3 below along with brief descriptions.

Table 9-3. ARM926EJ-S Endian Related Signals

Signal	Direction	Description
BIGENDINIT	Input	Determines the setting of the BIGEND bit held in the CP15 control register after system reset. When HIGH, the reset state of the BIGEND bit will be 1 (Big Endian). When LOW, the reset state of the BIGEND bit will be 0 (Little Endian).
CFGBIGEND	Output	ARM926EJ-S BIGEND configuration indicator. This signal reflects the value of the BIGEND bit held in the CP15 control register, which is used to determine the behavior of the ARM926EJ-S WRT endianness. When HIGH, the ARM926EJ-S treats bytes in memory as being in Big Endian format. When LOW, memory is treated as Little Endian.

The bigendinit platform input is connected directly to the BIGENDINIT input of the ARM926EJ-S and determines the processor and platform Endian mode of operation upon exiting system reset. However, the endian mode of operation of the processor (and therefore the platform and associated memory systems) may be changed according to the BIGEND bit in the CP15 control register. This output, cfg_bigend, reflects the BIGEND bit and is used to indicate the current Endian mode of operation for the platform as well as all external bus masters and slaves. The relevant Endian signals are shown in Figure 9-9.

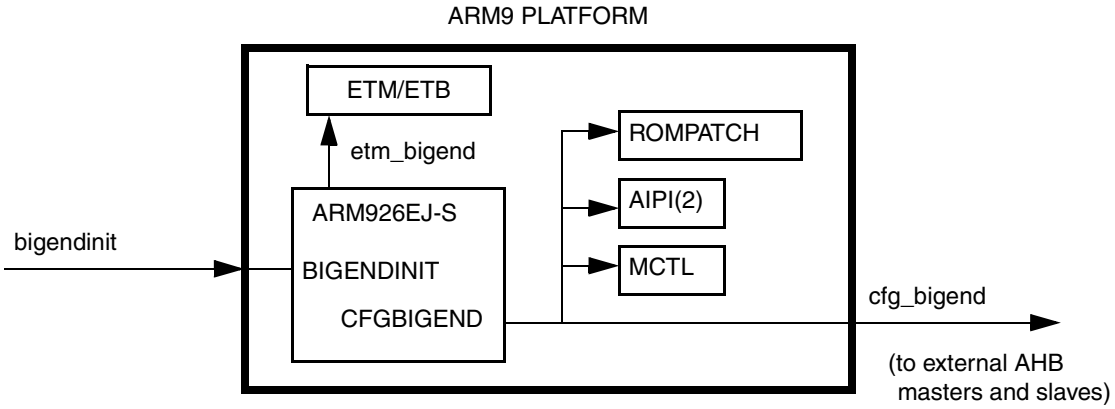


Figure 9-9. Endian Configuration Routing

The DHBL signals of the ARM926EJ-S will not be used within the platform. Instead, all modules affected by the Endian mode will use the cfg_bigend signal in conjunction with hsize and haddr[1:0] in order to handle non-word transfers correctly.

It is not envisioned that applications will need to dynamically change Endianness. However, this is still under investigation, and should be possible since the platform will support both Endian modes in hardware. It then becomes a software issue to insure a graceful mode change. For example, the write buffers should be drained prior to changing Endian modes.

9.9.7.1 Affected Modules

Only the AIPI (2), MCTL (RAM and ROM) and ROMPATCH modules are affected by the cfg_bigend signal within the platform. ETM9/ETB is affected by the Endian mode, and instead is connected to the etm_bigend signal internally.

9.9.7.2 Unaffected Modules

The AITC module is 32-bit write only. The registers within the MAX are also 32-bit access only, and the Endian mode is transparent from the AHB switch perspective. The PAHBMUX module has no registers associated with it and the Endian mode is transparent to its data muxing.

9.9.7.3 Un-Aligned Transfers

Unaligned transfers are not supported by the ARM9 Platform and therefore should not be attempted by alternate bus masters connected to it. That is, alternate bus masters should not attempt a 32-bit access to a non-word aligned address, nor a 16-bit access to a non half-word aligned address. The transfers will complete as normal, however the lower order address bits will be ignored according to Figure 9-4 and Figure 9-5.

9.9.7.4 Endian Mode and Alternate Bus Masters

Alternate bus masters must be cognizant of the Endian mode if they are capable of performing non-word accesses. Non-word register and memory transactions will be performed according to the state of the cfg_bigend output signal. The manner in which memory is accessed in the two Endian modes is described in the following two sections.

9.9.7.5 Little Endian Operation

A Little Endian configured ARM9 Platform (cfg_bigend = 0) should have memory connected to its secondary AHB ports as follows:

- Byte 0 of the memory connected to D[7:0]
- Byte 1 of the memory connected to D[15:8]
- Byte 2 of the memory connected to D[23:16]
- Byte 3 of the memory connected to D[31:24]

The byte write enables should be decoded by the AHB slaves as in Table 9-4.

Table 9-4. Little Endian Byte Write Enable Decoding

hwrite	hsize[1:0]	haddr[1:0]	we[31:24]	we[23:16]	we[15:8]	we[7:0]
0	x	x	0	0	0	0
1	00	00	0	0	0	1
1	00	01	0	0	1	0
1	00	10	0	1	0	0
1	00	11	1	0	0	0
1	01	0x	0	0	1	1
1	01	1x	1	1	0	0
1	10	xx	1	1	1	1
1	11	Reserved				

9.9.7.6 Big Endian Operation

A Big Endian configured ARM9 Platform (cfg_bigend=1) should have memory connected to its secondary AHB ports as follows:

- Byte 0 of the memory connected to D[31:24]
- Byte 0 of the memory connected to D[23:16]
- Byte 0 of the memory connected to D[15:8]
- Byte 0 of the memory connected to D[7:0]

The byte write enables should be decoded by the slaves as in Table 9-5.

Table 9-5. Big Endian Byte Write Enable Decoding

hwrite	hsize[1:0]	haddr[1:0]	we[31:24]	we[23:16]	we[15:8]	we[7:0]
0	x	x	0	0	0	0
1	00	00	1	0	0	0
1	00	01	0	1	0	0
1	00	10	0	0	1	0
1	00	11	0	0	0	1
1	01	0x	1	1	0	0
1	01	1x	0	0	1	1
1	10	xx	1	1	1	1
1	11	Reserved				

9.10 Preliminary Size Estimate

Table 9-6 show preliminary size estimates for the ARM9 Platform. Note that the area estimates correspond to C90LP, WCS, 1.1 V, 105C. clk = 266 MHz, hclk = 133 MHz. Gate equivalents are scaled to the area of the C90LP NAND2_2 cell.

Table 9-6. ARM9 Platform Size Estimates

Block	Number Included in Platform	Area in μm^2	Gate Count Total (NAND2_2)
I-Cache Data Memory (1024x32)	4	259,512	
I-Cache Tag Memory (128x22)	4	74,990	
I-Cache Valid Memory (32x24)	1	16,122*	
D-Cache Data Memory (1024x32)	4	259,512	
D-Cache Tag Memory (256x22)	4	149,980*	
D-Cache Valid Memory (32x24)	1	16,122*	
D-Cache Dirty Memory (128x8)	1	8,242	
MMU RAM (32x64)	2	73,286	

Table 9-6. ARM9 Platform Size Estimates (continued)

Block	Number Included in Platform	Area in μm^2	Gate Count Total (NAND2_2)
ETB RAM (1024x32)	2	129,756	
Memories Total	23	987,522	350K
ARM926 Core	1		TBD
ETM9 (Medium +)	1		TBD
ETB11	1		TBD
AITC	1		TBD
MCTL + ROM BIST	1		TBD
AIPI	2		TBD
AHBMUX	1		TBD
MAX	1		TBD
Scan Wrapper	1		TBD
ROMPATCH	1		TBD
BIST for Memories	4		TBD
IP to AHB (for ETB11)	1		TBD
Clock and Sync Control	1		TBD
JAM	1		TBD
Secure ROM monitor	1		TBD
Clock Tree	2		TBD
Logic Total		TBD	TBD
55% routing efficiency (logic only)		TBD	TBD
Platform Total		TBD	TBD

9.11 Power Consumption

Table 9-7 summarizes preliminary power estimates for the various ARM9 Platform operating modes. Power numbers will be measured off of the final freeze post-route netlist. There is no padding or margin included in these numbers.

Table 9-7. ARM9 Platform Power Estimates

Mode of Operation	BCS Corner (1.3 V –20C)	TYP Corner (1.2 V 25C)	WCS Corner (1.1 V, 105C)
Run Mode	TBD	TBD	TBD
Doze Mode	TBD	TBD	TBD

Table 9-7. ARM9 Platform Power Estimates (continued)

Mode of Operation	BCS Corner (1.3 V –20C)	TYP Corner (1.2 V 25C)	WCS Corner (1.1 V, 105C)
Sleep Mode without Well Bias Active	TBD	TBD	TBD
Sleep Mode with Well Bias Active	TBD	TBD	TBD

The operating modes are described below:

- Run Mode

clk, clk_always = 266MHz, hclk = 133MHz. Code running out of cache, once instructions loaded into cache. Exercising cache/MMU memories. Core busy with arithmetic operations. Activity on all alternate master ports and all slave ports concurrently. Compiled memory models pessimistic when memories not being accessed. Estimated loads on all platform outputs ranging from 0.5pf to 1.5pf.

- Doze Mode

clk, clk_always, and hclken stopped. hclk = 117MHz. No alternate bus master activity. Estimated loads on all platform outputs ranging from 0.5pf to 1.5pf.

- Sleep Mode

All clocks stopped. Includes: clk, clk_always, hclken, and hclk. Basically represents platform leakage current. No Dynamic or Static power in Sleep Mode. Well bias active power TYP very crude estimate of 10x reduction + compiled memories. Power due to charge pump not included since charge pump is external to the platform. WCS measured with well bias standard cell library.

9.12 ARM9 Platform I/O Signal List

The complete list of inputs and outputs for the ARM9 Platform are listed in Table 9-8.

Table 9-8. ARM9 Platform Signal List

Signal	Type	Description
Clocks and Resets		
clk	Input	Processor and Nexus reference clock
clk_always	Input	CLK that always runs
hclk	Input	AHB domain reference clock
hclken	Input	Controls ARM926EJ-S sampling of HCLK domain
a9p_clk_off	Output	To External Clock Control Module: The ARM9 Platform CLK may be turned off
por	Input	Power-On Reset
hreset_b	Input	System reset (ARM926EJ-S and AHB reset)
Platform Configuration		

Table 9-8. ARM9 Platform Signal List (continued)

Signal	Type	Description
bigendinit	Input	1 = Big Endian, 0 = Little Endian Determines initial Endian mode out of reset
cfg_bigend	Output	1 = Big Endian, 0 = Little Endian Reflects the value of BIGEND bit in ARM926EJ-S CP15 register. Determines endianness of platform slaves and external AHBs.
boot_int	Input	Internal Boot Indicator
ext_boot_adr[31:2]	Input	External Boot Address
bmon_timeout[2:0]	Input	Bus monitor timeout
JTAG Interface and Related I/O		
jtag_tck	Input	JTAG Test Clock
jtag_trst_b	Input	JTAG Test Reset
jtag_tms	Input	JTAG Test Mode Select
jtag_tdi	Input	JTAG Test Data Input
jtag_tdo	Output	JTAG Test Data Output
jtag_tdoen_b	Output	JTAG Test Data Output Tri-state Control
tapid_ver[3:0]	Input	Platform Version Number (JTAG ID register bits [11:8])
dbgrtck	Output	TCK “return clock” from JTAG synchronization
ARM926 Debug Related Signals		
dbgrq	Input	To ARM926: Debug request (connected to EDBGRQ)
dbgack	Output	From ARM926: Debug Acknowledge
dbgext[1:0]	Input	To ICE: External breakpoints/watchpoints
dbgiebkpt	Input	To ARM926: Instruction breakpoint
dbgdewpt	Input	To ARM926: Data watchpoint
arm_dbgrng[1:0]	Output	From ARM926: Embedded ICE-RT range out
arm_standbywfi	Output	From ARM926: Processor is in wait for interrupt mode
arm_java_mode	Output	From ARM926: Processor is in JAVA mode
arm_thumb_mode	Output	From ARM926: Processor is in THUMB mode
arm_fiq_b	Output	From AITC: Fast interrupt request to processor
arm_irq_b	Output	From AITC: Interrupt request to processor
arm_fiq_disable	Output	From ARM926: Processor has disabled FIQ interrupts
arm_irq_disable	Output	From ARM926: Processor has disabled IRQ interrupts
arm_cpsr_mode[4:0]	Output	From ARM926: Processor CPSR mode bits
Platform Debug Related Signals		

Table 9-8. ARM9 Platform Signal List (continued)

Signal	Type	Description
dbg_iahb_hready	Output	ARM926EJ-S I-AHB hready
dbg_iahb_htrans1	Output	ARM926EJ-S I-AHB htrans[1]
dbg_iahb_haddr[31:29]	Output	ARM926EJ-S I-AHB requested address (top 3 bits)
dbg_dahb_hready	Output	ARM926EJ-S D-AHB hready
dbg_dahb_htrans1	Output	ARM926EJ-S D-AHB htrans[1]
dbg_dahb_haddr[31:29]	Output	ARM926EJ-S D-AHB requested address (top 3 bits)
dbg_pahb_hready	Output	Primary AHB hready
dbg_pahb_htrans1	Output	Primary AHB htrans[1]
dbg_pahb_hmaster[3:0]	Output	Primary AHB hmaster ownership
dbg_a9p_ahb_en	Output	Enable output used for GPIO muxing of these debug signals
MCTL ROM Memory Interface		
rom_connect	Input	Indicates ROM exists on the MCTL interface.
rom_max_addr[11:0]	Input	Indicates ROM size. Corresponds to HADDR[21:10]. Smallest size supported is 1Kbyte, largest 4 Mbyte.
rom_wait	Input	ROM wait-state indicator 0 = No wait-state required 1 = One wait-state required
mctl_ce_rom_b	Output	MCU ROM chip enable
mctl_addr_rom[19:0]	Output	MCU ROM address.
mem_q_rom[31:0]	Input	ROM read data
MCTL RAM Memory Interface		
ram_connect	Input	Indicates RAM exists on the MCTL interface.
ram_max_addr[9:0]	Input	Indicates RAM size. Corresponds to HADDR[19:10]. Smallest size supported is 1Kbyte, largest 1 Mbyte.
ram_wait	Input	RAM read cycle wait-state indicator 0 = No wait-state required 1 = One wait-state required
mctl_mbist_sddtm	Output	MCU RAM mbist SDD test mode output
extram_oe	Output	Testmode control of external memories' output enable. This output should be connected to the OEN ports of all memories external to the ARM9 Platform (SRAM and TCM).
mctl_ce_ram_b	Output	MCU RAM chip enable
mctl_wr_ram_b	Output	MCU RAM access type: read=0, write=1
mctl_addr_ram[17:0]	Output	MCU RAM address
mctl_ben_ram_7_0	Output	MCU RAM byte enables

Table 9-8. ARM9 Platform Signal List (continued)

Signal	Type	Description
mctl_ben_ram_15_8	Output	MCU RAM byte enables
mctl_ben_ram_23_16	Output	MCU RAM byte enables
mctl_ben_ram_31_24	Output	MCU RAM byte enables
mctl_d_ram[31:0]	Output	RAM write data
mem_q_ram[31:0]	Input	RAM read data
Multi-Layer AHB Master Port 2		
m2_hlock	Input	AHB Locked Cycle Indicator (bus request timing)
m2_hmastlock	Input	AHB Locked Cycle Indicator (address timing)
m2_hmaster[3:0]	Input	AHB Master
m2_htrans[1:0]	Input	AHB Transfer Type
m2_hprot[3:0]	Input	AHB Protection Control
m2_hwrite	Input	AHB Write/Read Indicator
m2_hsize[1:0]	Input	AHB Transfer Size
m2_hburst[2:0]	Input	AHB Burst Length
m2_haddr[31:0]	Input	AHB Address
m2_hwdata[31:0]	Input	AHB Write Data
m2_hready_out	Output	AHB Transfer Done Out
m2_hrdata[31:0]	Output	AHB Read Data
m2_hresp0	Output	AHB Transfer Response
Multi-Layer AHB—Master Port 3		
m3_hlock	Input	AHB Locked Cycle Indicator (bus request timing)
m3_hmastlock	Input	AHB Locked Cycle Indicator (address timing)
m3_hmaster[3:0]	Input	AHB Master
m3_htrans[1:0]	Input	AHB Transfer Type
m3_hprot[3:0]	Input	AHB Protection Control
m3_hwrite	Input	AHB Write/Read Indicator
m3_hsize[1:0]	Input	AHB Transfer Size
m3_hburst[2:0]	Input	AHB Burst Length
m3_haddr[31:0]	Input	AHB Address
m3_hwdata[31:0]	Input	AHB Write Data
m3_hready_out	Output	AHB Transfer Done Out
m3_hrdata[31:0]	Output	AHB Read Data

Table 9-8. ARM9 Platform Signal List (continued)

Signal	Type	Description
m3_hresp0	Output	AHB Transfer Response
Multi-Layer AHB Master Port 4		
m4_hlock	Input	AHB Locked Cycle Indicator (bus request timing)
m4_hmastlock	Input	AHB Locked Cycle Indicator (address timing)
m4_hmaster[3:0]	Input	AHB Master
m4_htrans[1:0]	Input	AHB Transfer Type
m4_hprot[3:0]	Input	AHB Protection Control
m4_hwrite	Input	AHB Write/Read Indicator
m4_hsize[1:0]	Input	AHB Transfer Size
m4_hburst[2:0]	Input	AHB Burst Length
m4_haddr[31:0]	Input	AHB Address
m4_hwdata[31:0]	Input	AHB Write Data
m4_hready_out	Output	AHB Transfer Done Out
m4_hrdata[31:0]	Output	AHB Read Data
m4_hresp0	Output	AHB Transfer Response
Multi-Layer AHB Master Port 5		
m5_hlock	Input	AHB Locked Cycle Indicator (bus request timing)
m5_hmastlock	Input	AHB Locked Cycle Indicator (address timing)
m5_hmaster[3:0]	Input	AHB Master
m5_htrans[1:0]	Input	AHB Transfer Type
m5_hprot[3:0]	Input	AHB Protection Control
m5_hwrite	Input	AHB Write/Read Indicator
m5_hsize[1:0]	Input	AHB Transfer Size
m5_hburst[2:0]	Input	AHB Burst Length
m5_haddr[31:0]	Input	AHB Address
m5_hwdata[31:0]	Input	AHB Write Data
m5_hready_out	Output	AHB Transfer Done Out
m5_hrdata[31:0]	Output	AHB Read Data
m5_hresp0	Output	AHB Transfer Response
Multi-Layer AHB Slave Port 1		
s1_hmastlock	Output	AHB Locked Transfer
s1_hmaster[3:0]	Output	AHB Master

Table 9-8. ARM9 Platform Signal List (continued)

Signal	Type	Description
s1_htrans[1:0]	Output	AHB Transfer Type
s1_hprot[3:0]	Output	AHB Protection Control
s1_hwrite	Output	AHB Write/Read Indicator
s1_hsize[1:0]	Output	AHB Transfer Size
s1_hburst[2:0]	Output	AHB Burst Length
s1_haddr[31:0]	Output	AHB Address
s1_hwdata[31:0]	Output	AHB Write Data
s1_hrdata[31:0]	Input	AHB Read Data
s1_hready	Input	Transfer Done
s1_hresp0	Input	Transfer Response
Multi-Layer AHB Slave Port 2		
s2_hmastlock	Output	AHB Locked Transfer
s2_hmaster[3:0]	Output	AHB Master
s2_htrans[1:0]	Output	AHB Transfer Type
s2_hprot[3:0]	Output	AHB Protection Control
s2_hwrite	Output	AHB Write/Read Indicator
s2_hsize[1:0]	Output	AHB Transfer Size
s2_hburst[2:0]	Output	AHB Burst Length
s2_haddr[31:0]	Output	AHB Address
s2_hwdata[31:0]	Output	AHB Write Data
s2_hrdata[31:0]	Input	AHB Read Data
s2_hready	Input	Transfer Done
s2_hresp0	Input	Transfer Response
MAX Specific (Crossbar Switch)		
ccm_hbusreq	Input	External Clock Control Module Low-power Bus Request
ccm_hgrant	Output	Low-power Mode Bus Grant
s0_ampr_sel	Input	Slave port 0 alternate master priority register select.
s1_ampr_sel	Input	Slave port 1 alternate master priority register select.
s2_ampr_sel	Input	Slave port 2 alternate master priority register select.
IP Bus #1 (A)		
ipsa_module_en[31:1]	Output	IP Bus “A” Module Select
ipsa_addr[11:0]	Output	IP Bus “A” Address

Table 9-8. ARM9 Platform Signal List (continued)

Signal	Type	Description
ipsa_wdata[31:0]	Output	IP Bus “A” Write Data
ipsa_byte_31_24	Output	IP Bus “A” Byte Select
ipsa_byte_23_16	Output	IP Bus “A” Byte Select
ipsa_byte_15:8	Output	IP Bus “A” Byte Select
ipsa_byte_7_0	Output	IP Bus “A” Byte Select
ipsa_rwb	Output	IP Bus “A” Read/Write Indicator
ipsa_supervisor_access	Output	IP Bus “A” Supervisor Mode Access Control
ipsa_rdata[31:0]	Input	IP Bus “A” Read Data
ipsa_xfr_wait	Input	IP Bus “A” Transfer Wait State Indicator
ipsa_xfr_err	Input	IP Bus “A” Transfer Error Indicator
IP Bus #2 (B)		
ipsb_module_en[17:1]	Output	IP Bus “B” Module Select
ipsb_addr[11:0]	Output	IP Bus “B” Address
ipsb_wdata[31:0]	Output	IP Bus “B” Write Data
ipsb_byte_31_24	Output	IP Bus “B” Byte Select
ipsb_byte_23_16	Output	IP Bus “B” Byte Select
ipsb_byte_15:8	Output	IP Bus “B” Byte Select
ipsb_byte_7_0	Output	IP Bus “B” Byte Select
ipsb_rwb	Output	IP Bus “B” Read/Write Indicator
ipsb_supervisor_access	Output	IP Bus “B” Supervisor Mode Access Control
ipsb_rdata[31:0]	Input	IP Bus “B” Read Data
ipsb_xfr_wait	Input	IP Bus “B” Transfer Wait State Indicator
ipsb_xfr_err	Input	IP Bus “B” Transfer Error Indicator
ETM/ETB		
etm_traceclk	Output	ETM Trace Clock
etm_clkdivtwoen	Output	ETM half rate clocking mode
etm_dbgrq	Output	Debug Request.
etm_etmen	Output	ETM Enabled
etm_pipestat[2:0]	Output	Pipeline Status
etm_tracepkt[15:0]	Output	ETM Trace Packet
etm_tracesync	Output	Trace synchronization.
etm_portsize[2:0]	Output	ETM Port Size.

Table 9-8. ARM9 Platform Signal List (continued)

Signal	Type	Description
etm_portmode[1:0]	Output	Normal, Multiplexed, or Demultiplexed mode of operation
etb_full	Output	ETB Overflow Indicator
etb_acqcomp	Output	ETB Trace Acquisition Complete
etm_extout	Output	External ETM Outputs
ect_dbgrq	Input	Debug Request.
etm_extin[3:0]	Input	External ETM Inputs.
Miscellaneous		
aitc_rise_arb	Output	Interrupt pending, raise arbitration priority if desired
a9p_mem_on	Input	Used with a9p_mem_pwr_dn to power off ARM926 ICACHE, DCACHE, and MMU memories
a9p_mem_pwr_dn	Input	Used with a9p_mem_on to power off ARM926 ICACHE, DCACHE, and MMU memories
a9p_int_b[63:0]	Input	External Interrupts
a9p_dsm_int_holdoff	Input	Deep Sleep Module Interrupt Disable
wt_en	Input	Well Tie Input (Physical connection only)
wt_en_dnw	Input	Well Tie Input for deep n-wells (Physical connection only)
Platform Scan Test Interface		
ipt_mode[3:0]	Input	Test Mode Control
ipt_clk_se	Input	Clock Gating Cell Scan Enable
ipt_memory_read_inhibit_int	Input	Disables memory read operations from internal memories (caches, ETB) during scan testing. Read data is forced to zeros when asserted. When negated, memories function normally.
ipt_scan_size[1:0]	Input	Scan Chain Length Control
ipt_scan_enable	Input	Scan Shift Enable
ipt_scan_in[66:0]	Input	Platform Test Serial In
ipt_scan_out[66:0]	Output	Platform Test Serial Out
Scan Wrapper Test Interface		
ipt_wrapper_clk_in[1:0]	Input	Platform Wrapper Clocks [0] = CLK Domain [1] = HCLK Domain
ipt_wrapper_se	Input	Scan Shift Enable
ipt_wrapper_scan_size[1:0]	Input	Scan Wrapper Chain Length
ipt_wrapper_scan_in[23:0]	Input	Scan Wrapper Test Serial In [2:0] = CLK [11:3] = HCLK

Table 9-8. ARM9 Platform Signal List (continued)

Signal	Type	Description
ipt_wrapper_scan_out[23:0]	Output	Scan Wrapper Test Serial Out [2:0] = CLK [11:3] = HCLK
Memory BIST Interface		
ipt_bist_fail	Output	Aggregate Memory BIST Fail Status
ipt_bist_done	Output	Aggregate Memory BIST Execution Status
ipt_bist_bitmap[15:0]	Output	Memory BIST Bitmap Data Out
ipt_bist_sdo	Output	Memory BIST Bitmap Serial Data Out
ipt_bist_addr_out[17:0]	Output	Memory BIST Address Out
ipt_bist_bmdata_avail	Output	Memory BIST Bitmap Data Strobe
ipt_bist_done_dcache	Output	Data Cache Memory BIST Done
ipt_bist_done_etb	Output	ETB Memory BIST Done
ipt_bist_done_icache	Output	Instruction Cache Memory BIST Done
ipt_bist_done_mmu	Output	MMU Memory BIST Done
ipt_bist_done_mram	Output	MCTL RAM Memory BIST Done
ipt_bist_done_mrom	Output	MCTL ROM Memory BIST Done
ipt_bist_fail_dcache	Output	Data Cache Memory BIST Fail
ipt_bist_fail_etb	Output	ETB Memory BIST Fail
ipt_bist_fail_icache	Output	Instruction Cache Memory BIST Fail
ipt_bist_fail_mmu	Output	MMU Memory BIST Fail
ipt_bist_fail_mram	Output	MCTL RAM Memory BIST Fail
ipt_bist_config_addr_mode[2:0]	Input	Memory BIST Address Mode Selection
ipt_bist_config_alt_al_en	Input	Memory BIST Alternate Algorithm Enable
ipt_bist_config_aftest_en	Input	Memory BIST Address Fault Test Enable
ipt_bist_config_dpat_en[7:0]	Input	Memory BIST Data Pattern Enable
ipt_bist_config_dret_en	Input	Memory BIST Data Retention Test Enable
ipt_bist_config_dsof	Input	Memory BIST Disable “Stop on Fail”
ipt_bist_config_marchc_en	Input	Memory BIST Marching Pattern Test Enable
ipt_bist_config_sdd_en	Input	Memory BIST SDD Test Enable
ipt_bist_config_sel_dcache	Input	Memory BIST Data Cache Engine Select
ipt_bist_config_sel_etb	Input	Memory BIST ETB Engine Select
ipt_bist_config_sel_icache	Input	Memory BIST Instruction Cache Engine Select
ipt_bist_config_sel_mmu	Input	Memory BIST MMU Engine Select

Table 9-8. ARM9 Platform Signal List (continued)

Signal	Type	Description
ipt_bist_config_sel_mram	Input	Memory BIST MCTL RAM Engine Select
ipt_bist_config_sel_mrom	Input	Memory BIST MCTL ROM Engine Select
ipt_bist_config_usrctrl_bm	Input	Memory BIST User Controlled Parallel Bitmap Output Rate
ipt_bist_invoke	Input	Memory BIST Invoke
ipt_bist_mode[2:0]	Input	Memory BIST Mode Select
ipt_bist_release	Input	Memory BIST Pause State Release
ipt_bist_repdata_out_en	Input	Memory BIST Repair Data Output Enable
ipt_bist_reset	Input	Memory BIST Reset
ipt_bist_retention_en	Input	Memory BIST Retention Enable
ipt_bist_sdi	Input	Memory BIST Serial Data In
ipt_bist_serial_data_en	Input	Memory BIST Serial Data Enable
ipt_bist_shift_clk	Input	Memory BIST Shift Clock

9.13 Electrical Specifications

This section will present timing information for all major AHBs (both internal and external) to the platform. Timing information on all other signals on the platform periphery will be grouped by functionality and presented after the AHB timings.

9.13.1 Conditions

The timing presented in this section were derived from an ARM926EJ-S synthesis run using the C90LP library, worst case process, 105^{o}C, 1.10 V with clk running at 266 MHz. In this case, the hclk domain (all AHBs) will run at half the clk speed or 133 MHz.

9.13.2 Well Bias Mode

The timing specifications in this section do not cover the well bias mode of operation. At the present time, well bias mode is planned to be used in Sleep Mode only. That is, clk and hclk will be stopped and the platform buses will be inactive. However, the negation of the a9p_clock_off output when an interrupt is asserted is still required in order for the external clock control module to exit Sleep Mode and turn on the clocks. The delay for the a9p_int_b[61:0] to a9p_clock_off path will be affected by well bias mode, but not significantly so.

9.13.2.1 Functional Operation in Well Bias Mode

Programmable options should be used to support laboratory testing of the platform in well bias mode. The platform's AC performance will be impacted (slower) when well biasing is enabled and is TBD. Care

should be taken to identify external interfaces which may not be running in well bias mode as clock insertion and clock skew differences may prevent proper operation.

9.13.3 clk and jtag_tck Relationship

The jtag_tck clock input must always be less than 1/8 the frequency of the clk clock input. This constraint is due to the JTAG synchronization logic in the CLKCTL module. During the execution of low-power code, the frequency of clk is dynamic, and therefore care should be taken that jtag_tck is always less than 1/8 the frequency of clk at any given instant.

To maximize throughput via the JTAG port when uploading/downloading code or memory images, it is suggested the debugger enter debug mode directly out of reset with clk and jtag_tck running as fast as possible. However, once normal mode low-power code execution begins, the jtag_tck frequency should be set to be 1/8 the frequency of the lowest possible clk frequency.

9.13.4 Clocks and Reset Timing

Table 9-9 and Figure 9-10 are valid for all AHB interfaces on the ARM9 Platform. The same clock insertion delay and hreset_b negation timing will be used for all modules on the ARM9 Platform.

Table 9-9. ARM9 Platform AHB Clock and Reset Timing Constraints

Description	Delay
CLK_ROOT Period	3.75 ns (266 MHz)
CLK_ROOT jitter (3% rounded up)	115.0 ps
HCLK_ROOT Period	7.5 ns (133 MHz)
HCLK_ROOT jitter (3% rounded up)	230.0 ps
CLK_ROOT to CLK and HCLK_ROOT to HCLK Insertion Delay (T_{insert})	1.60 +/- 0.100 ns
CLK and HCLK Uncertainty	200 ps
HRESET_B hold time to HCLK_LEAF (T_{ihrst})	1.80 ns
HRESET_B setup time to HCLK_LEAF (T_{isrst})	1.60 ns
HCLKEN setup time to CLK_LEAF ($T_{isclken}$)	2.00 ns
HCLKEN hold time to CLK_LEAF ($T_{ihclken}$)	0.00 ns

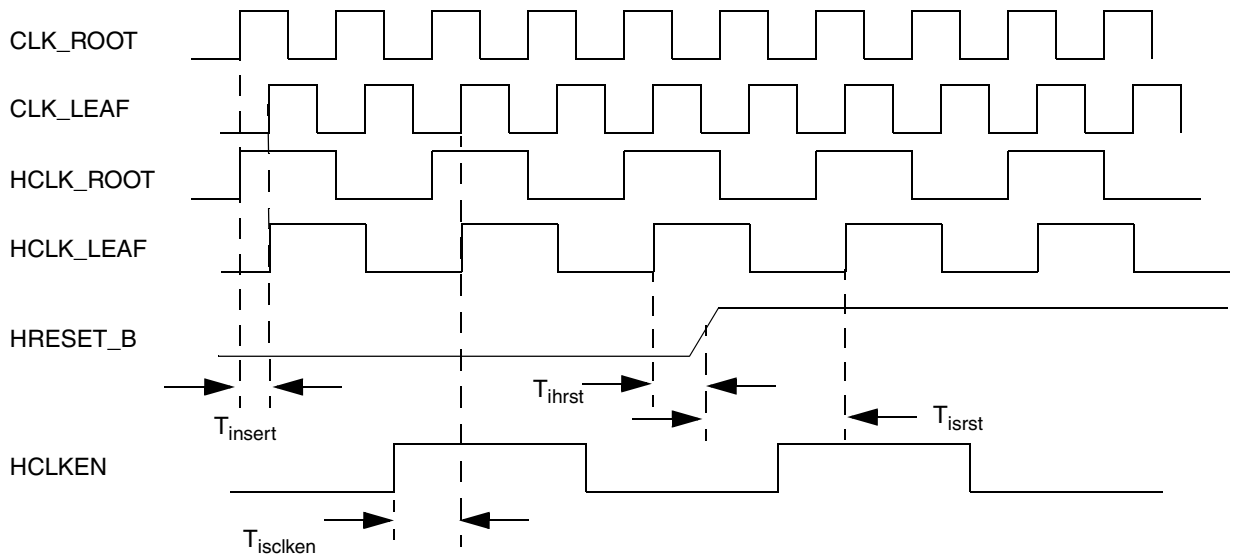


Figure 9-10. ARM9 Platform AHB Clock and Reset Timing Relationship

9.13.5 Alternate Bus Master (ABM) Interface Timing

Table 9-10 shows the loading constraints used on all ARM9 Platform Alternate Bus Master bus interfaces. The timing parameters in Figure 9-11 reflect these constraints. The Alternate Bus Master signals are designated by the “MX_” prefix attached to the normal AHB naming convention.

Table 9-10. Alternate Bus Master Constraints

Description	Value
All Output Loading	0.50 pf
Input Transition Time (platform boundary)	0.750 ns (20/80)

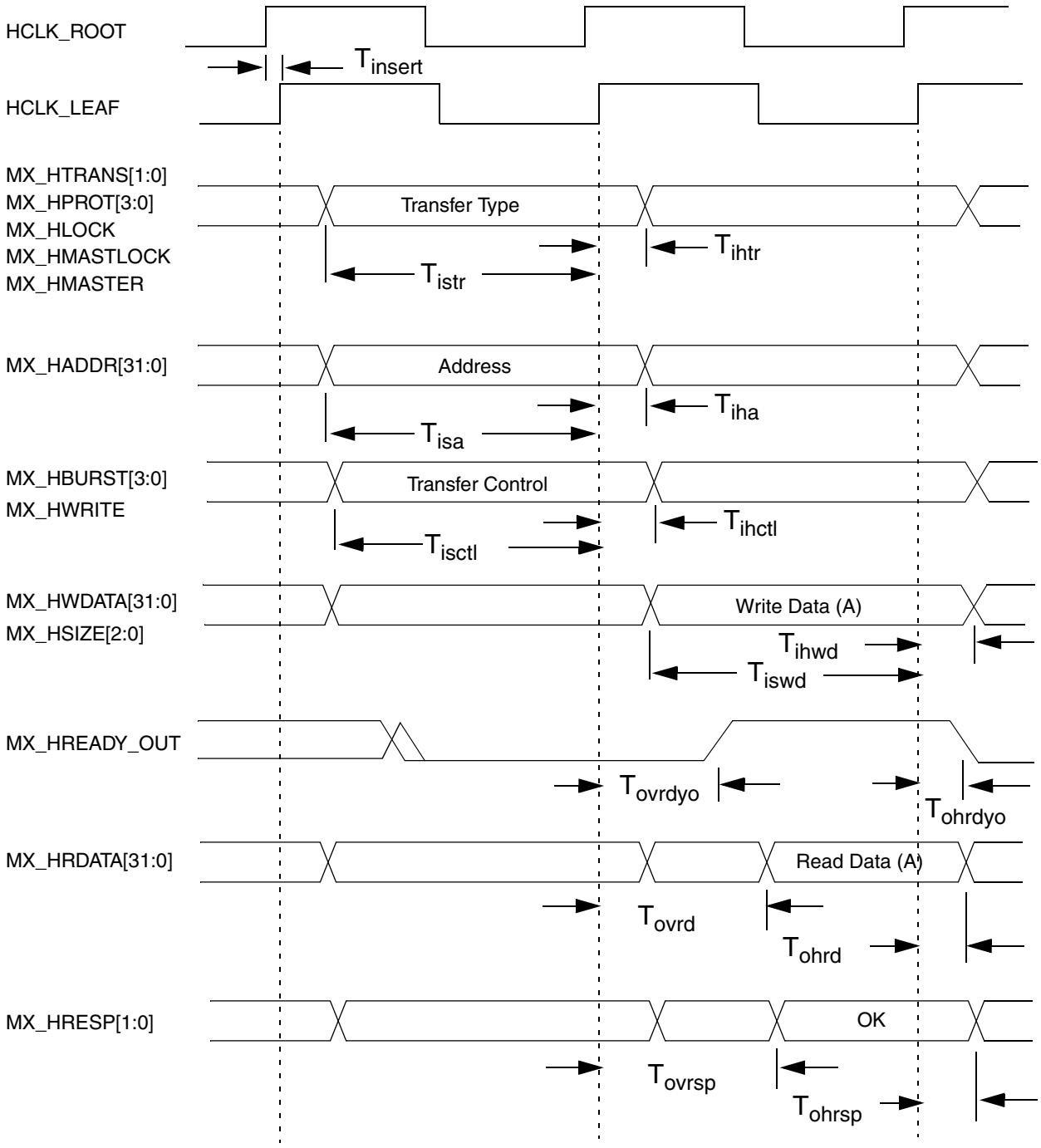


Figure 9-11. Alternate Bus Master Timing Parameters

Table 9-11. Alternate Bus Master Interface AC Timing Parameters

Description	Parameter	Timing (ns)
HCLK_LEAF minimum clock period including jitter	T_{clk}	7.27
MX_HMASTER/MX_HTRANS/MX_HPROT/MX_HLOCK/MX_HMASTLOCK/MX_HMASTER Transfer Type setup time before HCLK_LEAF	T_{istr}	6.23
MX_HMASTER/MX_HTRANS/MX_HPROT/MX_HLOCK/MX_HMASTLOCK/MX_HMASTER Transfer Type hold time after HCLK_LEAF	T_{ihtr}	>0
MX_HADDR[31:0] Address setup time before HCLK_LEAF	T_{isa}	6.23
MX_HADDR[31:0] Address hold time after HCLK_LEAF	T_{iha}	>0
MX_HWRITE/MX_HSIZE/MX_HBURST control signal setup time before HCLK_LEAF	T_{isctl}	6.23
MX_HWRITE/MX_HSIZE/MX_HBURST control signal hold time after HCLK_LEAF	T_{ihctl}	>0
MX_HWDATA Write Data setup time before HCLK_LEAF	T_{iswd}	6.00
MX_HWDATA Write Data hold time after HCLK_LEAF	T_{ihwd}	>0
MX_HREADY_OUT Ready Out valid time after HCLK_LEAF	T_{ovrdyo}	4.80
MX_HREADY_OUT Ready Out hold time after HCLK_LEAF	T_{ohrdyo}	>0
MX_HRDATA Read Data valid time after HCLK_LEAF	T_{ovrd}	6.00
MX_HRDATA Read Data hold time after HCLK_LEAF	T_{ohrd}	>0
MX_HRESP0 valid time after HCLK_LEAF	T_{ovrsp}	6.00
MX_HRESP0 hold time after HCLK_LEAF	T_{ohrsp}	>0

9.13.6 Secondary AHB Timing

Table 9-12 shows the loading constraints used on all ARM9 Platform Secondary AHB interfaces. The timing parameters in Figure 9-13 reflect these constraints. The constraints and AC parameters are valid for all 3 of the platform's secondary AHBs. The Secondary AHB signals are designated by the "SX_" prefix attached to the normal AHB naming convention.

Table 9-12. Secondary AHB Constraints

Description	Value
SX_HADDR, SX_HWDATA Loading	0.50 pf
All Other Output Loading	0.50 pf
Input Transition Time (at platform boundary)	0.75 ns (20/80)

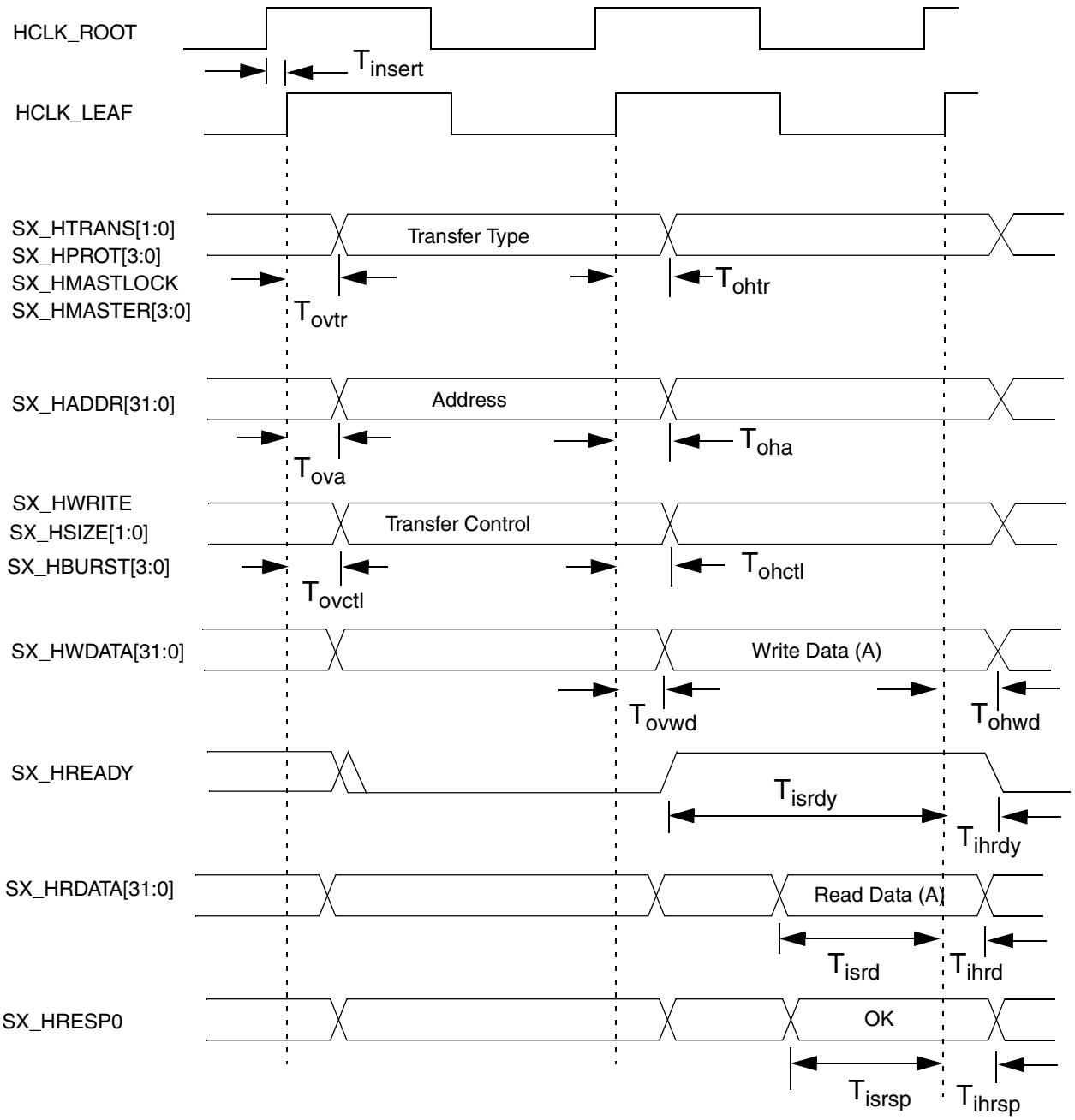


Figure 9-12. Secondary AHB AC Timing Parameters

Table 9-13. Secondary AHB AC Timing Parameters

Description	Parameter	Timing (ns)
HCLK_LEAF minimum clock period including jitter	T_{clk}	7.27
SX_HTRANS/SX_HPROT/SX_HMASTLOCK/SX_HMASTER Transfer Type valid time after HCLK_LEAF	T_{ovtr}	5.00

Table 9-13. Secondary AHB AC Timing Parameters (continued)

Description	Parameter	Timing (ns)
SX_HTRANS/SX_HPROT/SX_HMASTLOCK/SX_HMASTER Transfer Type hold time after HCLK_LEAF	T_{ohtr}	>0
SX_HADDR[31:0] Address valid time after HCLK_LEAF	T_{ova}	4.30
SX_HADDR[31:0] Address hold time after HCLK_LEAF	T_{oha}	>0
SX_HWRITE/SX_HSIZE/SX_HBURST control signal valid time after HCLK_LEAF	T_{ovctl}	5.70
SX_HWRITE/SX_HSIZE/SX_HBURST control signal hold time after HCLK_LEAF	T_{ohctl}	>0
SX_HWDATA Write Data valid time after HCLK_LEAF	T_{ovwd}	5.70
SX_HWDATA Write Data hold time after HCLK_LEAF	T_{ohwd}	>0
SX_HREADY setup time before HCLK_LEAF (input to slaves)	T_{isrdy}	5.60
SX_HREADY hold time after HCLK_LEAF	T_{ihrdy}	>0
SX_HRDATA setup time before HCLK_LEAF	T_{isrd}	4.10
SX_HRDATA hold time after HCLK_LEAF	T_{ihrd}	>0
SX_HRESP0 setup time before HCLK_LEAF	T_{isrsp}	4.10
SX_HRESP0 hold time after HCLK_LEAF	T_{ihrsp}	>0

9.13.7 RAM and ROM Interface Timing

Table 9-14 shows the loading constraints used when generating timing parameters on the ARM9 Platform's RAM and ROM interfaces. External RAM and ROM interface signals not shown in the table below are either static or test related.

Table 9-14. RAM and ROM Interface Loading Constraints

Signal	Type	Constraint
RAM		
MCTL_OEN_RAM	Output	0.50 pf
MCTL_CE_RAM_B	Output	0.25 pf
MCTL_WR_RAM_B	Output	0.50 pf
MCTL_ADDR_RAM[17:0]	Output	0.50 pf
MCTL_BEN_RAM_*_*	Output	0.50 pf
MCTL_D_RAM[31:0]	Output	0.50 pf
MEM_Q_RAM[31:0]	Input	0.75 ns (Input Transition Time)
ROM		
MCTL_CE_ROM_B	Output	0.25 pf

Table 9-14. RAM and ROM Interface Loading Constraints (continued)

Signal	Type	Constraint
MCTL_ADDR_ROM[19:0]	Output	0.50 pf
MEM_Q_ROM[31:0]	Input	0.75 ns (Input Transition Time)

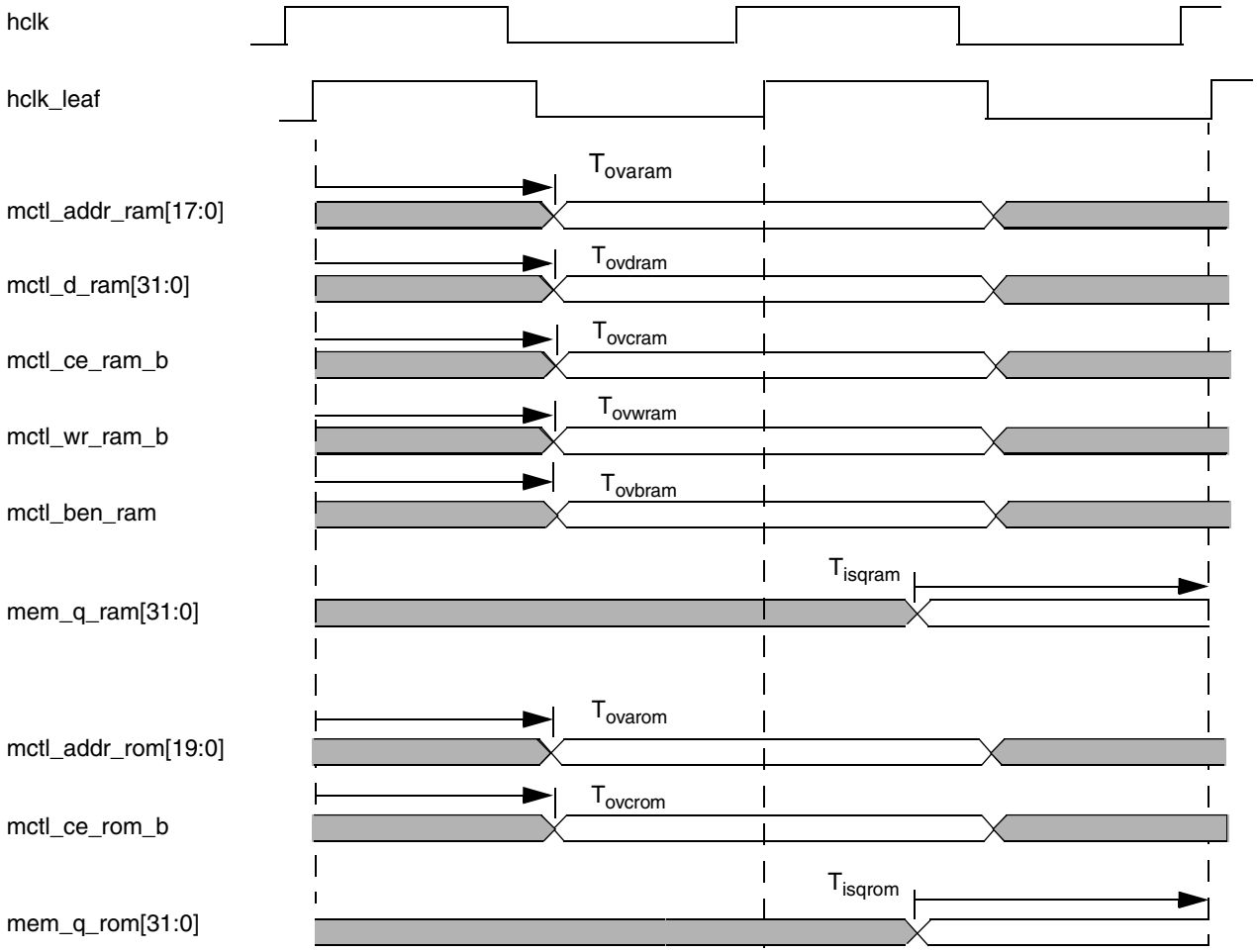


Figure 9-13. RAM and ROM Interface AC Timing Parameters

Table 9-15. RAM and ROM Interface AC Timing Parameters

Description	Parameter	Timing (ns)
HCLK_LEAF minimum clock period	T_{clk}	8.40
RAM		
MCTL_ADDR_RAM valid time after HCLK_LEAF	T_{ovaram}	6.65

Table 9-15. RAM and ROM Interface AC Timing Parameters (continued)

Description	Parameter	Timing (ns)
MCTL_D_RAM valid time after HCLK_LEAF	T_{ovdram}	6.60
MCTL_CE_RAM_B valid time after HCLK_LEAF	T_{ovcram}	6.70
MCTL_WR_RAM_B valid time after HCLK_LEAF	T_{ovwram}	6.55
MCTL_BEN_RAM_*_* valid time after HCLK_LEAF	T_{ovbram}	6.6
MEM_Q_RAM setup time before HCLK_LEAF	T_{isqram}	4.65
MEM_Q_RAM hold time after HCLK_LEAF	T_{ihqram}	>0
ROM		
MCTL_ADDR_ROM valid time after HCLK_LEAF	T_{ovarom}	6.55
MCTL_CE_ROM_B valid time after HCLK_LEAF	T_{ovcrom}	6.80
MEM_Q_ROM setup time before HCLK_LEAF	T_{isqrom}	3.85
MEM_Q_ROM hold time after HCLK_LEAF	T_{ihqrom}	>0

Chapter 10
ARM926EJ-S Interrupt Controller (AITC)

The ARM926EJ-S Interrupt Controller (AITC) is a 32-bit peripheral that collects interrupt requests from up to 64 sources, and provides an interface to the ARM926EJ-S core. The AITC includes software-controlled priority levels for normal interrupts. Figure 10-1 shows the simplified block diagram of the AITC.

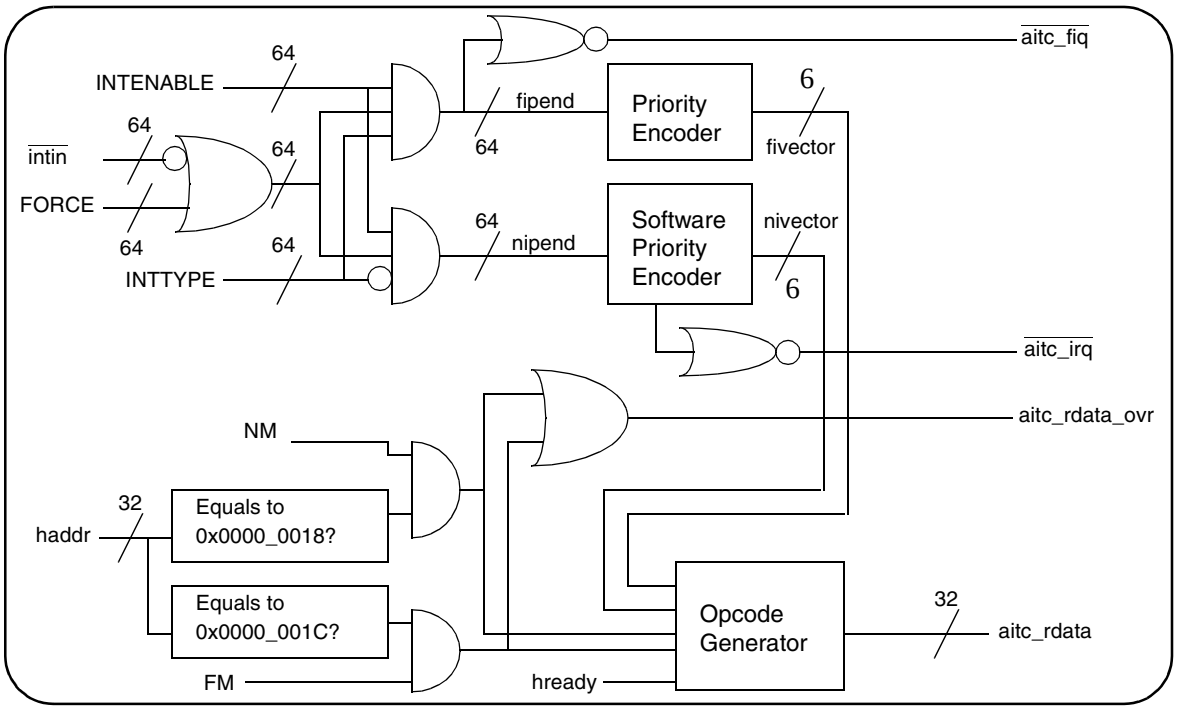


Figure 10-1. AITC Block Diagram

10.1 Overview

The AITC consists of a set of control registers and associated logic to perform interrupt masking, and priority support of normal interrupts. Interrupt source registers (INTSRCH/INTSRCL) are a pair of 32-bit status registers with a single interrupt source associated with each of the 64 bits. An interrupt line or set of interrupt lines are routed from each interrupt source to the INTSRCH or INTSRCL register. This allows up to 64 distinct interrupt sources in an implementation.

10.1.1 Features

10.1.2 Modes of Operation

Interrupt requests may be forced to be asserted by way of the interrupt force registers (INTFRCH/INTFRCL). Each bit in this register is logically “OR-ed” with the corresponding hardware request line prior to feeding the INTSRCH or INTSRCL register inputs. There is a corresponding set of interrupt enable registers (INTENABLEH/INTENABLEL), also 32-bits wide which allow individual bit masking of the INTSRCH/INTSRCL registers. There is also a corresponding set of interrupt type register (INTTYPEH/INTTYPEL), which selects whether an interrupt source will generate a normal or fast interrupt to the ARM926EJ-S core.

There is a corresponding set of normal interrupt pending registers (NIPNDH/NIPNDL) which indicate pending normal interrupt requests. These registers are equivalent to the logical AND of the interrupt source registers (INTSRCH/INTSRCL), the interrupt enable registers (INTENABLEH/INTENABLEL), and the NOT of the interrupt type registers (INTTYPEH/INTTYPEL). (Refer to Figure 10-1) The NIPNDH/NIPNDL register bits are bit-wise “NOR-ed” together to form the nIRQ signal routed to the ARM926EJ-S core. This core input signal is maskable by the normal interrupt disable bit (I bit) in the processor status register (CPSR). The normal interrupt vector register (NIVECSR) indicates the vector index of highest priority pending normal interrupt.

There is a corresponding set of fast interrupt pending registers (FIPNDH/FIPNDL) which indicate pending fast interrupt requests. These registers are equivalent to logical AND of interrupt source registers (INTSRCH/INTSRCL), interrupt enable registers (INTENABLEH/INTENABLEL), and interrupt type registers (INTTYPEH/INTTYPEL). (Refer to Figure 10-1) FIPNDH/FIPNDL register bits are bit-wise “NOR-ed” together to form the nFIQ signal routed to the ARM926EJ-S core. This core input signal is maskable by the fast interrupt disable bit (F bit) in the CPSR. The fast interrupt vector register (FIVECSR) indicates the vector index of highest priority pending fast interrupt.

AITC supports two vector table modes: high memory and low memory. If AITC is in high memory vector table mode, opcode is “LDR PC, [PC,#-(288-4*(vector index)]”. This causes ARM926-ES core to load the Program Counter (PC) with a vector from a table of 64 vectors located at 0xFFFF_FF00 to 0xFFFF_FFFF; more specifically the PC is loaded with the vector located at 0xFFFF_FF00 + 4*(vector index). If AITC is in low memory vector table mode, this opcode is “LDR PC, [PC, #((table pointer)+4*(vector index) –32]”. This causes the ARM926-EJS core to load the PC with a vector from a table of 64 vectors beginning at (table pointer) and ending at (table pointer)+0xFF; more specifically the PC is loaded with the vector located at (table pointer) + 4*(vector index). This hardware mechanism alleviates the need for software to determine which interrupt source caused the interrupt to be asserted. All interrupt controller registers can be read and written during privileged mode only. Writes attempted to read-only registers will be ignored. These registers can be only modified using 32-bit writes. INTFRCH/INTFRCL registers are provided for software generation of interrupts. By enabling interrupts for these bit positions, software can force an interrupt request. This register can also be used to debug hardware interrupt service routines by providing an alternate method of interrupt assertion. The interrupt requests are prioritized in the following sequence:

1. Fast interrupt requests, in order of highest number
2. Normal interrupt requests, in order of highest priority level, then highest source number with the same priority

AITC provides 16 software controlled priority levels for normal interrupts. Every interrupt can be placed in any priority level. The AITC also provides a normal interrupt priority level mask (NIMASK) which disables any interrupt with a priority level lower than or equal to the mask. If a level 0 normal interrupt and a level 1 normal interrupt are asserted at the same time, the level 1 normal interrupt will be selected assuming that NIMASK has not disabled level 1 normal interrupts. If two level 1 normal interrupts are asserted at the same time, the level 1 normal interrupt with the highest source number will be selected, also assuming that NIMASK has not disabled level 1 normal interrupts.

10.2 Memory Map and Register Definition

AITC module has 26 registers. All of these registers are single cycle access as the AITC sits on the native bus of the ARM926EJ-S core. This section provides the detailed descriptions for all of the AITC registers.

10.2.1 Memory Map

Table 10-1 shows the AITC memory map.

Table 10-1. AITC Memory Map

Address	Register	Access	Reset Value	Section/Page
General Registers				
0x1004_0000 (INTCNTL)	Interrupt Control Register	R/W	0x0000_0000	10.2.3/10-8
0x1004_0004 (NIMASK)	Normal Interrupt Mask Register	R/W	0x0000_001F	10.2.4/10-10
0x1004_0008 (INTENNUM)	Interrupt Enable Number Register	R/W	0x0000_0000	10.2.5/10-11
0x1004_000C (INTDISNUM)	Interrupt Disable Number Register	R/W	0x0000_0000	10.2.6/10-11
0x1004_0010 (INTENABLEH)	Interrupt Enable Register High	R/W	0x0000_0000	10.2.7/10-12
0x1004_0014 (INTENABLEL)	Interrupt Enable Register Low	R/W	0x0000_0000	10.2.7/10-12
0x1004_0018 (INTTYPEH)	Interrupt Type Register High	R/W	0x0000_0000	10.2.8/10-13
0x1004_001C (INTTYPEL)	Interrupt Type Register Low	R/W	0x0000_0000	10.2.8/10-13
0x1004_0020 (NIPRIORITY7)	Normal Interrupt Priority Level Register 7	R/W	0x0000_0000	10.2.9/10-14
0x1004_0024 (NIPRIORITY6)	Normal Interrupt Priority Level Register 6	R/W	0x0000_0000	10.2.9/10-14
0x1004_0028 (NIPRIORITY5)	Normal Interrupt Priority Level Register 5	R/W	0x0000_0000	10.2.9/10-14
0x1004_002C (NIPRIORITY4)	Normal Interrupt Priority Level Register 4	R/W	0x0000_0000	10.2.9/10-14

Table 10-1. AITC Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x1004_0030 (NIPRIORITY3)	Normal Interrupt Priority Level Register 3	R/W	0x0000_0000	10.2.9/10-14
0x1004_0034 (NIPRIORITY2)	Normal Interrupt Priority Level Register 2	R/W	0x0000_0000	10.2.9/10-14
0x1004_0038 (NIPRIORITY1)	Normal Interrupt Priority Level Register 1	R/W	0x0000_0000	10.2.9/10-14
0x1004_003C (NIPRIORITY0)	Normal Interrupt Priority Level Register 0	R/W	0x0000_0000	10.2.9/10-14
0x1004_0040 (NIVECSR)	Normal Interrupt Vector and Status Register	R	0xFFFF_FFFF	10.2.10/10-22
0x1004_0044 (FIVECSR)	Fast Interrupt Vector and Status Register	R	0xFFFF_FFFF	10.2.11/10-23
0x1004_0048 (INTSRCH)	Interrupt Source Register High	R	0x0000_0000	10.2.12/10-24
0x1004_004C (INTSRCL)	Interrupt Source Register Low	R	0x0000_0000	10.2.12/10-24
0x1004_0050 (INTFRCH)	Interrupt Force Register High	R/W	0x0000_0000	10.2.13/10-27
0x1004_0054 (INTFRCL)	Interrupt Force Register Low	R/W	0x0000_0000	10.2.13/10-27
0x1004_0058 (NIPNDH)	Normal Interrupt Pending Register High	R	0x0000_0000	10.2.14/10-28
0x1004_005C (NIPNDL)	Normal Interrupt Pending Register High	R	0x0000_0000	10.2.14/10-28
0x1004_0060 (FIPNDH)	Fast Interrupt Pending Register High	R	0x0000_0000	10.2.15/10-29
0x1004_0064 (FIPNDL)	Fast Interrupt Pending Register Low	R	0x0000_0000	10.2.15/10-29

10.2.2 Register Summary

Figure 10-2 shows the key to the register fields and Table 10-2 shows the register figure conventions.

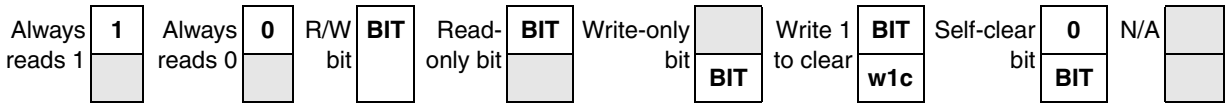


Figure 10-2. Key to Register Fields

Table 10-2. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 10-3 shows the AITC register summary.

Table 10-3. AITC Register Summary

Name		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
0x1004_0000 (INTCNTL)	R	0	0	0	0	0	0	0	0	0	NIDIS	FIDIS	NIAD	FIAD	0	0	MD
	W																
	R	0	0	0	0	POINTER										0	0
	W																
0x1004_0004 (NIMASK)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	NIMASK				
	W																
0x1004_0008 (INTENNUM)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W											ENNUM					

Table 10-3. AITC Register Summary (continued)

Name		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
0x1004_000C (INTDISNUM)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W											DISNUM					
0x1004_0010 (INTENABLEH)	R	INTENABLE[63:48]															
	W																
	R	INTENABLE[47:32]															
	W																
0x1004_0014 (INTENABLEL)	R	INTENABLE[31:16]															
	W																
	R	INTENABLE[15:0]]															
	W																
0x1004_0018 (INTTYPEH)	R	INTTYPE[63:48]															
	W																
	R	INTTYPE[47:32]															
	W																
0x1004_001C (INTTYPEL)	R	INTTYPE[31:16]															
	W																
	R	INTTYPE[16:0]															
	W																
0x1004_0020 (NIPRIORITY7)	R	NIPR63					NIPR62				NIPR61				NIPR60		
	W																
	R	NIPR59					NIPR58				NIPR57				NIPR56		
	W																
0x1004_0024 (NIPRIORITY6)	R	NIPR55					NIPR54				NIPR53				NIPR52		
	W																
	R	NIPR51					NIPR50				NIPR49				NIPR48		
	W																
0x1004_0028 (NIPRIORITY5)	R	NIPR47					NIPR46				NIPR45				NIPR44		
	W																
	R	NIPR43					NIPR42				NIPR41				NIPR40		
	W																

Table 10-3. AITC Register Summary (continued)

Name		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	
0x1004_002C (NIPRIORITY4)	R	NIPR39	NIPR38	NIPR37	NIPR36
	W				
	R	NIPR35	NIPR34	NIPR33	NIPR32
	W				
0x1004_0030 (NIPRIORITY3)	R	NIPR31	NIPR30	NIPR29	NIPR28
	W				
	R	NIPR27	NIPR26	NIPR25	NIPR24
	W				
0x1004_0034 (NIPRIORITY2)	R	NIPR23	NIPR22	NIPR21	NIPR20
	W				
	R	NIPR19	NIPR18	NIPR17	NIPR16
	W				
0x1004_0038 (NIPRIORITY1)	R	NIPR15	NIPR14	NIPR13	NIPR12
	W				
	R	NIPR11	NIPR10	NIPR9	NIPR8
	W				
0x1004_003C (NIPRIORITY0)	R	NIPR7	NIPR6	NIPR5	NIPR4
	W				
	R	NIPR3	NIPR2	NIPR1	NIPR0
	W				
0x1004_0040 (NIVECSR)	R	NIVECTOR			
	W				
	R	NIPRILVL			
	W				
0x1004_0044 (FIVECSR)	R	FIVECTOR			
	W				
	R	FIVECTOR			
	W				
0x1004_0048 (INTSRCH)	R	INTIN[63:48]			
	W				
	R	INTIN[48:32]			
	W				

Table 10-3. AITC Register Summary (continued)

Name		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
0x1004_004C (INTSRCL)	R	INTIN[31:16]
	W	
	R	INTIN[15:0]
	W	
0x1004_0050 (INTFRCH)	R	FORCE[63:48]
	W	
	R	FORCE[47:32]
	W	
0x1004_0054 (INTFRCL)	R	FORCE[31:16]
	W	
	R	FORCE[15:0]
	W	
0x1004_0058 (NIPNDH)	R	NIPEND[63:48]
	W	
	R	NIPEND[47:32]
	W	
0x1004_005C (NIPNDL)	R	NIPEND[31:16]
	W	
	R	NIPEND[15:0]
	W	
0x1004_0060 (FIPNDH)	R	FIPEND[63:48]
	W	
	R	FIPEND[47:32]
	W	
0x1004_0064 (FIPNDL)	R	FIPEND[31:16]
	W	
	R	FIPEND[15:0]
	W	

10.2.3 Interrupt Control Register (INTCNTL)

INTCNTL controls the interrupts in AITC. Both normal and fast interrupts can be enabled to jump directly to the interrupt service routine. For fast interrupts, it may be faster to begin to fast interrupt routine at

0x0000_001C instead of jumping to a service routine. The vector table can be sourced in high memory, 0xFFFF_FF00 to 0xFFFF_FFFF, or in low memory. If the vector table is located in low memory (MD=1), a register has been provided to control where the vector table is located. This register is located on the ARM926EJ-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. This register can be only modified using 32-bit writes.

0x1004_0000 (INTCNTL) — Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	NIDIS	FIDIS	NIAD	FIAD	0	0	MD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 10-3. Interrupt Control Register Format

Table 10-4. Interrupt Control Register Field Description

Field	Description
31–23	Reserved. These bits are reserved and should read 0.
22 NIDIS	Normal Interrupt Disable. This bit, when set, disables the generation of the normal interrupt signal. This bit is similar to the I bit of the ARM926EJ-S core. This bit along with the FIDIS bit is used to enable secure operations. 0 Does not affect the normal interrupt generation 1 Disable all normal interrupts
21 FIDIS	Fast Interrupt Disable. This bit, when set, disables the generation of the fast interrupt signal. This bit is similar to the F bit of the ARM926EJ-S core. This bit along with the NIDIS bit is used to enable secure operations. 0 Does not affect the fast interrupt generation 1 Disable all fast interrupts
20 NIAD	Normal Interrupt Arbiter Rise ARM Level. This bit, when asserted, increases bus arbitration priority of ARM core when normal interrupt signal (nIRQ) is asserted. If an alternate master has ownership of the bus when a normal interrupt occurs, bus will be given back to the processor core *after* the DMA device has completed its accesses. NIAD bit does not affect alternate master accesses that are in progress. To prevent an alternate master from accessing the bus during an interrupt service routine, the interrupt flag must not be cleared until the end of the service routine. Another option is to use the ABFEN and ABFLAG bits. 0 Disregard the normal interrupt flag when evaluating bus requests 1 Normal interrupt flag increases bus arbitration priority of the ARM core to decrease the latency of Interrupt service routine
19 FIAD	Fast Interrupt Arbiter Rise ARM Level. This bit functions same as NIAD bit except for the fast interrupts (nFIQ). 0 Disregard the fast interrupt flag when evaluating bus requests 1 Fast interrupt flag increases bus arbitration priority of the ARM core to decrease the latency of interrupt service routine.
18–17	Reserved. These bits are reserved and should read 0.
16 MD	Interrupt Vector Table Mode. Indicates whether the interrupt vector is located in high memory or low memory. 0 Interrupt vector table located in high memory from 0xFFFF_FF00 to 0xFFFF_FFFF 1 Interrupt vector table located in low memory from POINTER to POINTER+0xFF

Table 10-4. Interrupt Control Register Field Description (continued)

Field	Description
15–12	Reserved. These bits are reserved and should read 0.
11–2 POINTER	Interrupt Vector Table Pointer. Indicates start of vector table when in low memory (MD=1). Only word-aligned tables are allowed, and 2 zeros are added in the LSBs when this value is used by AITC. The value stored here is left shifted by 2 bits, so the actual table vector can be directly written into the appropriate bits. The value stored in 10 bits, times 4, must be set greater than or equal to 0x0000_0024 and less than or equal to 0x0000_0F00.
1–0	Reserved. These bits are reserved and should read 0.

10.2.4 Normal Interrupt Mask Register (NIMASK)

NIMASK controls the normal interrupt mask level. All normal interrupts with a priority level lower than or equal to NIMASK are disabled. The priority level of normal interrupts are determined by the normal interrupt priority level registers (NIPRIORITY7–0). Reset state of this register does not disable any normal interrupts. Writing all 1's, or –1, to NIMASK sets normal interrupt mask to –1 which does not disable any normal interrupt priority levels. This hardware mechanism can be used to create reentrant normal interrupt routines by disabling lower priority normal interrupts. Refer Section 10.3.6 for more details on use of NIMASK register. This register is located on the ARM926EJ-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. This register can be only modified using 32-bit writes.

Address 0x1004_0004 (NIMASK) Access: Supervisor read/write

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

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

Figure 10-4. Normal Interrupt Mask Register Format

Table 10-5. Normal Interrupt Mask Register Field Description

Field	Description
31–5	Reserved. These bits are reserved and should read 0.
4–0 NIMASK	Normal Interrupt Mask. Controls normal interrupt mask level. All normal interrupts of priority level lower than or equal to the NIMASK will be disabled. 0 Disable priority level 0 normal interrupts 1 Disable priority level 1 and lower normal interrupts ... 0xE (14)Disable priority level 14 and lower normal interrupts 0xF (15)Disable all normal interrupts 0x10–0x1FDo not disable any normal interrupts

10.2.5 Interrupt Enable Number Register (INTENNUM)

The Interrupt Enable Number Register provides hardware accelerated enabling of interrupts. Any write to this register enables an interrupt source. If 6 LSBs are 000000, then interrupt source 0 is enabled. If 6 LSBs are 000001, then interrupt source 1 is enabled. And so forth. This register is decoded into a one hot mask that is logically OR-ed with INTENABLEH/INTENABLEL register. This hardware mechanism alleviates the need for an atomic read/modify/write sequence to enable an interrupt source. To enable interrupts 10 and 20, software only preforms two writes to AITC: first write 10 to INTENNUM register, then write 20 to INTENNUM register (order of writes is irrelevant). This register is located on the ARM926EJ-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. This register can be only modified using 32-bit writes. This register always reads back all 0s.

Address 0x1004_0008 (INTENNUM) Access: Supervisor read/write

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

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

Figure 10-5. Interrupt Enable Number Register Format

Table 10-6. Interrupt Enable Number Register Description

Field	Description
31–6	Reserved. These bits are reserved and should read 0.
5–0 ENNUM	Interrupt Enable Number. Writing to this register will enable the interrupt source associated with this value. 0 Enable interrupt source 0 1 Enable interrupt source 1 ... 63 Enable interrupt source 63

10.2.6 Interrupt Disable Number Register (INTDISNUM)

The Interrupt Disable Number Register provides hardware accelerated disabling of interrupts. Any write to this register disables one interrupt source. If the 6 LSBs are equal 000000, then interrupt source 0 is disabled. If the 6 LSBs equal 000001, then interrupt source 1 is disabled, and so on. This register is decoded into a one hot mask which is inverted and logically AND-ed with the INTENABLEH/INTENABLEL register. The hardware mechanism alleviates the need for an atomic read/modify/write sequence to disable an interrupt source. To disable interrupts 10 and 20, the software need only preform two writes to the AITC: first write 10 to INTDISNUM register, then write 20 to INTDISNUM register (the order of the writes is irrelevant). This register is located on the ARM926EJ-S

native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. This register can be only modified using 32-bit writes. This register always reads back all 0s.

Address 0x1004_000C (INTDISNUM) Access: Supervisor read/write

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

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

Figure 10-6. Interrupt Enable Number Register Format

Table 10-7. Interrupt Disable Number Register Field Description

Field	Description
31–6	Reserved. These bits are reserved and should read 0.
5–0 DISNUM	Interrupt Disable Number. Writing to this register will disable the interrupt source associated with this value. 0 Disable interrupt source 0 1 Disable interrupt source 1 ... 63 Disable interrupt source 63

10.2.7 Interrupt Enable Register High (INTENABLEH) and Low (INTENABLEL)

The INTENABLEH and INTENABLEL registers are used to enable pending interrupt requests to the ARM9 core. Each bit in these registers corresponds to an interrupt source available in the system. The reset state of these registers are to have all interrupts masked. These registers can be updated by various methods: writing directly to INTENABLEH/INTENABLEL registers, setting bits in the INTENNUM register, or clearing bits in the INTDISNUM register. These registers are located on the ARM926EJ-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. These registers can be only modified using 32-bit writes.

Address 0x1004_0010 (INTENABLEH) Access: Supervisor read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INTENABLE[63:48]															
W	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTENABLE[47:32]															
W	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-7. Interrupt Enable Register High Format

Address 0x1004_0014 (INTENABLEL) Access: Supervisor read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INTENABLE[31:16]															
W	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTENABLE[15:0]															
W	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-8. Interrupt Enable Register Low Format

Table 10-8. Interrupt Enable Register Low and High Field Descriptions

Field	Description
31–0 INTENABLE	Interrupt Enable. This bit enables the corresponding interrupt source to request a normal interrupt or a fast interrupt. A reset operation clears this bit. If an enable bit is set and the corresponding interrupt source is asserted, the interrupt controller will assert a normal or a fast interrupt request depending on associated INTTYPEH/INTTYPEL setting. 0 Interrupt disabled 1 Interrupt enabled and will generate a normal or fast interrupt upon assertion

10.2.8 Interrupt Type Register High (INTTYPEH) and Low (INTTYPEL)

The INTTYPEH and INTTYPEL registers are used to select whether a pending interrupt source, when enabled with the INTENABLEH/INTENABLEL, will create a normal interrupt or a fast interrupt to the ARM9 core. Each bit in these registers corresponds to an interrupt source available in the system. The reset state of these registers will cause all enabled interrupt sources to generate a normal interrupt. These registers are located on the ARM926EJ-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. These registers can be only modified using 32-bit writes.

Address 0x1004_0018 (INTTYPEH) Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INTTYPE[63:48]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INTTYPE[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-9. Interrupt Type Register High Format

Address 0x1004_001C (INTTYPEL) Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INTTYPE[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INTTYPE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-10. Interrupt Type Register Low Format

Table 10-9. Interrupt Type Register High and Low Register Description

Field	Description
31–0 INTTYPE	Interrupt Type. This bit indicates whether the corresponding interrupt source will request a normal interrupt or a fast interrupt. If INTTYPE bit is set and the corresponding interrupt source is asserted, the interrupt controller will assert a fast interrupt request. 0 Interrupt source will generate a normal interrupt (nIRQ). 1 Interrupt source will generate a fast interrupt (nFIQ).

10.2.9 Normal Interrupt Priority Level Registers (NIPRIORITY*n*)

The Normal Interrupt Priority Level Registers (NIPRIORITY7–0) provide a software controllable prioritization of normal interrupts. Normal interrupts with a higher priority level will preempt normal interrupts with a lower priority. The reset state of these registers forces all normal interrupts to the lowest priority level. If a level 0 normal interrupt and a level 1 normal interrupt are asserted at the same time, the level 1 normal interrupt will be selected assuming that NIMASK has not disabled level 1 normal interrupts. If two level 1 normal interrupts are asserted at the same time, the level 1 normal interrupt with the highest source number will be selected, also assuming that NIMASK has not disabled level 1 normal interrupts. These registers can only be accessed to in privileged mode using 32-bit writes.

Address 0x1004_0020 (NIPRIORITY7) Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR63				NIPR62				NIPR61				NIPR60			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR59				NIPR58				NIPR57				NIPR56			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-11. Normal Interrupt Priority Level 7 Register Format

Table 10-10. Normal Interrupt Priority Level Register 7 Field Description

Bits	Field	Description
31–28	NIPR63	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt ... 15 Highest priority normal interrupt
27–24	NIPR62	
23–20	NIPR61	
19–16	NIPR60	
15–12	NIPR59	
11–8	NIPR58	
7–4	NIPR57	
3–0	NIPR56	

Address 0x1004_0024 (NIPRIORITY6) Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR55				NIPR54				NIPR53				NIPR52			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR51				NIPR50				NIPR49				NIPR48			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-12. Normal Interrupt Priority Level 6 Register Format

Table 10-11. Normal Interrupt Priority Level Register 6 Field Description

Field	Description
31–28 NIPR55	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt ... 15 Highest priority normal interrupt
27–24 NIPR54	
23–20 NIPR53	
19–16 NIPR52	
15–12 NIPR51	
11–8 NIPR50	
7–4 NIPR49	
3–0 NIPR48	

Address 0x1004_0028 (NIPRIORITY5) Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR47				NIPR46				NIPR45				NIPR44			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR43				NIPR42				NIPR41				NIPR40			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-13. Normal Interrupt Priority Level 5 Register Format

Table 10-12. Normal Interrupt Priority Level Register 5 Field Description

Field		Description
31–28	NIPR47	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt ... 15 Highest priority normal interrupt
27–24	NIPR46	
23–20	NIPR45	
19–16	NIPR44	
15–12	NIPR43	
11–8	NIPR42	
7–4	NIPR41	
3–0	NIPR40	

Address 0x1004_002C (NIPRIORITY4) Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR39				NIPR38				NIPR37				NIPR36			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR35				NIPR34				NIPR33				NIPR32			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-14. Normal Interrupt Priority Level 4 Register Format

Table 10-13. Normal Interrupt Priority Level Register 4 Field Description

Field	Description
31–28 NIPR39	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt ... 15 Highest priority normal interrupt
27–24 NIPR38	
23–20 NIPR37	
19–16 NIPR36	
15–12 NIPR35	
11–8 NIPR34	
7–4 NIPR33	
3–0 NIPR32	

Address 0x1004_0030 (NIPRIORITY3) Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR31				NIPR30				NIPR29				NIPR28			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR27				NIPR26				NIPR25				NIPR24			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-15. Normal Interrupt Priority Level 3 Register Format

Table 10-14. Normal Interrupt Priority Level Register 3 Field Description

Field	Description
31–28 NIPR31	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt ... 15 Highest priority normal interrupt
27–24 NIPR30	
23–20 NIPR29	
19–16 NIPR28	
15–12 NIPR27	
11–8 NIPR26	
7–4 NIPR25	
3–0 NIPR24	

Address 0x1004_0034 (NIPRIORITY2) Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR23				NIPR22				NIPR21				NIPR20			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR19				NIPR18				NIPR17				NIPR16			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-16. Normal Interrupt Priority Level 2 Register Format

Table 10-15. Normal Interrupt Priority Level Register 2 Field Description

Bits	Field	Description
31–28	NIPR23	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt ... 15 Highest priority normal interrupt
27–24	NIPR22	
23–20	NIPR21	
19–16	NIPR20	
15–12	NIPR19	
11–8	NIPR18	
7–4	NIPR17	
3–0	NIPR16	

Address 0x1004_0038 (NIPRIORITY1) Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR15				NIPR14				NIPR13				NIPR12			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR11				NIPR10				NIPR9				NIPR8			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-17. Normal Interrupt Priority Level 1 Register Format

Table 10-16. Normal Interrupt Priority Level Register 1 Field Description

Field	Description
31–28 NIPR15	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt ... 15 Highest priority normal interrupt
27–24 NIPR14	
23–20 NIPR13	
19–16 NIPR12	
15–12 NIPR11	
11–8 NIPR10	
7–4 NIPR9	
3–0 NIPR8	

Address 0x1004_003C (NIPRIORITY0) Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR7				NIPR6				NIPR5				NIPR4			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR3				NIPR2				NIPR1				NIPR0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-18. Normal Interrupt Priority Level 1 Register Format

Table 10-17. Normal Interrupt Priority Level Register 0 Field Description

Bits	Field	Description
31–28	NIPR7	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt ... 15 Highest priority normal interrupt
27–24	NIPR6	
23–20	NIPR5	
19–16	NIPR4	
15–12	NIPR3	
11–8	NIPR2	
7–4	NIPR1	
3–0	NIPR0	

10.2.10 Normal Interrupt Vector and Status Register (NIVECSR)

The NIVECSR register displays the priority of the highest pending normal interrupt and also provides vector index of the interrupt's service routine. This number can be used directly as an index into a vector table to select the highest pending normal interrupt source. This read-only register can only be accessed to in privileged mode.

Address 0x1004_0040 (NIVECSR) Access: Supervisor read

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

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

Figure 10-19. Normal Interrupt Vector and Status Register Format

Table 10-18. Normal Interrupt Vector and Status Register Field Description

Field	Description
31–16 NIVECTOR	Normal Interrupt Vector. Indicates vector index for the highest pending normal interrupt. –1 No normal interrupt request pending 0 Interrupt 0 highest priority pending normal interrupt 1 Interrupt 1 highest priority pending normal interrupt ... 63 Interrupt 63 highest priority pending normal interrupt 64+ (not –1)unused, will not occur
15–0 NIPRILVL	Normal Interrupt Priority Level. Indicates priority level of highest priority normal interrupt. This number can be written to NIMASK to disable current priority normal interrupts to build a reentrant normal interrupt system. –1 No normal interrupt request pending 0 Highest priority normal interrupt is level 0 1 Highest priority normal interrupt is level 1 ... 15 Highest priority normal interrupt is level 15 16+ (not –1)unused, will not occur

10.2.11 Fast Interrupt Vector and Status Register (FIVECSR)

FIVECSR provides the vector index for highest priority active fast interrupt's service routine (higher the source number of fast interrupt, higher will be the priority level). This hardware mechanism replaces the previous necessity for core support of the FF1 command. This number can be directly used as an index into a vector table to select the highest pending fast interrupt source. This read-only register is located on the ARM926EJ-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode.

Address 0x1004_0044 (FIVECSR) Access: Supervisor read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FIVECTOR[31:16]															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FIVECTOR[15:0]															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 10-20. Fast Interrupt Vector and Status Register Format

Table 10-19. Fast Interrupt Vector and Status Register Description

Field	Description
31–0 FIVECTOR	Fast Interrupt Vector. Indicates vector index for the highest pending fast interrupt. –1 No fast interrupt request pending (–1 is defined as all bits in the field are set to 1.) 0 Interrupt 0 highest pending fast interrupt 1 Interrupt 1 highest pending fast interrupt ... 63 Interrupt 63 highest pending fast interrupt 64+ (not –1)unused, will not occur

10.2.12 Interrupt Source Register High (INTSRCH) and Low (INTSRCL)

INTSRCH and INTSRCL are both 32-bits wide. INTSRCH and INTSRCL reflect the status of all interrupt request inputs into the interrupt controller. Unused bit positions always read zero (no request pending). The state of this register out of reset is determined by the peripheral circuits generating the requests; normally, the requests would be inactive. These read-only registers can only be accessed in privileged mode and can only accessed with 32-bit reads.

Address 0x1004_0048 (INTSRCH) Access: Supervisor read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INTIN[63:48]															
W																
Reset[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTIN[47:32]															
W																
Reset[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-21. Interrupt Source Register High Format

[1] The state of this register out of reset is determined by the peripheral circuits generating the requests; normally, the requests would be inactive.

Address 0x1004_004C (INTSRCL) Access: Supervisor read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INTIN[31:16]															
W																
Reset[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTIN[15:0]															
W																
Reset[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-22. Interrupt Source Register High Format

[1] The state of this register out of reset is determined by the peripheral circuits generating the requests; normally, the requests would be inactive.

Table 10-20. Interrupt Source Register High and Low Description

Field	Description
31–0 INTIN	Interrupt Source. Indicates the state of the corresponding hardware interrupt source. 0 Interrupt source negated 1 Interrupt source asserted

10.2.12.1 Interrupt Assignments High

Table 10-21. Interrupt Source High (INTSRCH) Assignment

Name	Bit	Interrupt Source Module	Notes
INT_DPTC	Bit 31	Dynamic Process Temperature Compensate (DPTC)	
INT_IIM	Bit 30	IC Identify Module (IIM)	
INT_LCDC	Bit 29	LCD Controller (LCDC)	
INT_SLCDC	Bit 28	Smart LCD Controller (SLCDC)	
INT_SAHARA	Bit 27	Symmetric/Asymmetric Hashing and Random Accelerator	
INT_SCM	Bit 26	SCC SCM	
INT_SMN	Bit 25	SCC SMN	
INT_USBOTG	Bit 24	USB OTG	
INT_USBHS2	Bit 23	USB HOST2	
INT_USBHS1	Bit 22	USB HOST1	
INT_H264	Bit 21	H264	
INT_EMMAPP	Bit 20	eMMA Post Processor	
INT_EMMAPRP	Bit 19	eMMA Pre Processor	
INT_FEC	Bit 18	Fast Ethernet Controller	
INT_UART5	Bit 17	UART5	
INT_UART6	Bit 16	UART6	
INT_DMACH15	Bit 15	DMA Channel 15	
INT_DMACH14	Bit 14	DMA Channel 14	
INT_DMACH13	Bit 13	DMA Channel 13	
INT_DMACH12	Bit 12	DMA Channel 12	
INT_DMACH11	Bit 11	DMA Channel 11	
INT_DMACH10	Bit 10	DMA Channel 10	
INT_DMACH9	Bit 9	DMA Channel 9	
INT_DMACH8	Bit 8	DMA Channel 8	

Table 10-21. Interrupt Source High (INTSRCH) Assignment (continued)

Name	Bit	Interrupt Source Module	Notes
INT_DMACH7	Bit 7	DMA Channel 7	
INT_DMACH6	Bit 6	DMA Channel 6	
INT_DMACH5	Bit 5	DMA Channel 5	
INT_DMACH4	Bit 4	DMA Channel 4	
INT_DMACH3	Bit 3	DMA Channel 3	
INT_DMACH2	Bit 2	DMA Channel 2	
INT_DMACH1	Bit 1	DMA Channel 1	
INT_DMACH0	Bit 0	DMA Channel 0	

10.2.12.2 Interrupt Assignments Low

Table 10-22. Interrupt Source Low (INTSRCL) Assignment

Name	Bit	Interrupt Source Module	Notes
INT_CSI	Bit 31	CMOS Sensor Interface (CSI)	
INT_ATA	Bit 30	Advanced Technology Attachment (ATA)	Hard Disk
INT_NFC	Bit 29	NAND Flash Controller (NFC)	
INT_PCMCIA	Bit 28	PCMCIA/CF Host Controller (PCMCIA)	
INT_WDOG	Bit 27	Watchdog (WDOG)	
INT_GPT1	Bit 26	General Purpose Timer (GPT1)	
INT_GPT2	Bit 25	General Purpose Timer (GPT2)	
INT_GPT3	Bit 24	General Purpose Timer (GPT3)	
INT_PWM	Bit 23	Pulse Width Modulator (PWM)	
INT_RTC	Bit 22	Real-Time Clock (RTC)	
INT_KPP	Bit 21	Key Pad Port (KPP)	
INT_UART1	Bit 20	UART1	
INT_UART2	Bit 19	UART2	
INT_UART3	Bit 18	UART3	
INT_UART4	Bit 17	UART4	
INT_CSPI1	Bit 16	Configurable SPI (CSPI1)	
INT_CSPI2	Bit 15	Configurable SPI (CSPI2)	
INT_SSI1	Bit 14	Synchronous Serial Interface (SSI1)	
INT_SSI2	Bit 13	Synchronous Serial Interface (SSI2)	
INT_I2C1	Bit 12	I^2C Bus Controller (I^2C1)	

Table 10-22. Interrupt Source Low (INTSRCL) Assignment (continued)

Name	Bit	Interrupt Source Module	Notes
INT_SDHC1	Bit 11	Secure Digital Host Controller (SDHC1)	
INT_SDHC2	Bit 10	Secure Digital Host Controller (SDHC2)	
INT_SDHC3	Bit 9	Secure Digital Host Controller (SDHC3)	
INT_GPIO	Bit 8	General Purpose Input/Output (GPIO)	
INT_MSHC	Bit 7	Memory Stick Host Controller (MSHC)	
INT_CSPI3	Bit 6	Configurable SPI (CSPI3)	
INT_RTIC	Bit 5	Real Time Integrity Checker (RTIC)	
INT_GPT4	Bit 4	General Purpose Timer (GPT4)	
INT_GPT5	Bit 3	General Purpose Timer (GPT5)	
INT_GPT6	Bit 2	General Purpose Timer (GPT6)	
INT_I2C2	Bit 1	I^2C Bus Controller (I^2C2)	
Reserved	Bit 0	Reserved	

10.2.13 Interrupt Force Register High (INTFRCH) and Low (INTFRCL)

INTFRCH and INTFRCL are both 32-bits wide. They allow software generation of interrupts for each of the possible interrupt sources for functional or debug purposes. System level design may reserve one or more sources for software purposes to allow software to self-schedule interrupts by forcing one or more of these “sources” in appropriate interrupt force register(s). These registers can only be accessed to in privileged mode. These registers can be only modified using 32-bit writes.

Address 0x1004_0050 (INTFRCH) Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FORCE[63:48]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FORCE[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address 0x1004_0054 (INTFRCL) Access: Supervisor read/write

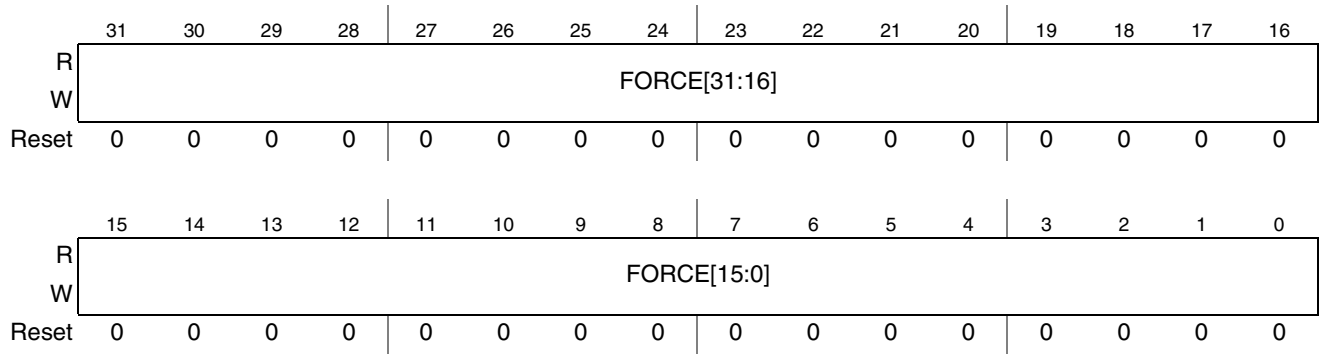


Figure 10-23. Interrupt Force Register Format

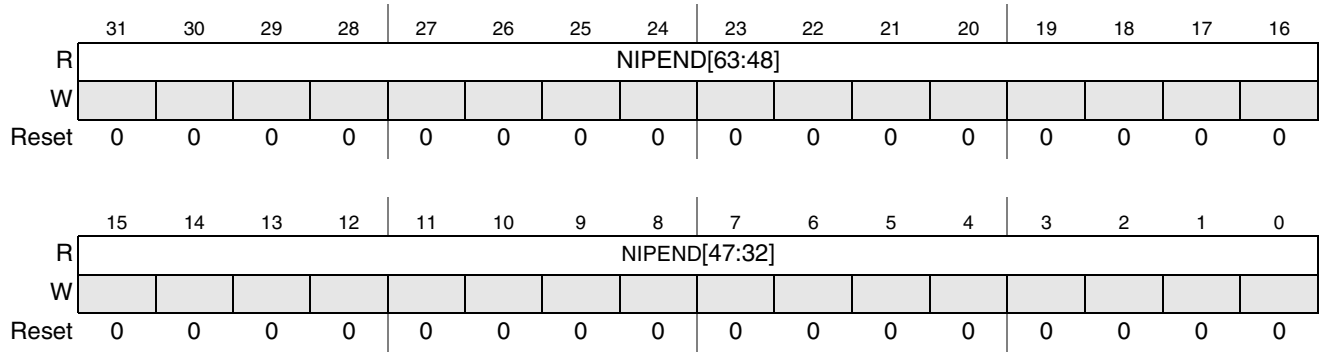
Table 10-23. Interrupt Force Register High and Low Field Description

Field	Description
31–0 FORCE	Interrupt Source Force Request. Used to force a request for the corresponding interrupt source. 0 Standard interrupt operation 1 Interrupt forced asserted

10.2.14 Normal Interrupt Pending Register High (NIPNDH) and Low (NIPNDL)

NIPNDH and NIPNDL are both 32-bits wide registers used to monitor the outputs of the enable and masking operations. These registers are actually a set of buffers; therefore, reset state of these registers are determined by normal interrupt enable registers, interrupt mask register and interrupt source registers. The value reflected in these registers is unaffected by the value of NIMASK register. These read-only registers are located on ARM926EJ-S native bus, accessible in 1 cycle, and can only be accessed in privileged mode

Address 0x1004_0058 (NIPNDH) Access: Supervisor read



Address 0x1004_005C (NIPNDL) Access: Supervisor read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NIPEND[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NIPEND[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-24. Normal Interrupt Pending Register High and Low Description

Field	Description
31–0 NIPEND	Normal Interrupt Pending Bit. If a normal interrupt enable bit is set and the corresponding interrupt source is asserted, the interrupt controller will assert a normal interrupt request. The normal interrupt pending bits reflect the interrupt input lines which are asserted and are currently enabled to generate a normal interrupt. 0 No normal interrupt request 1 Normal interrupt request pending

10.2.15 Fast Interrupt Pending Register High (FIPNDH) and Low (FIPNDL)

FIPNDH and FIPNDL are both 32-bits wide registers used to monitor the outputs of enable and masking operations. These registers are actually a set of buffers; therefore, reset state of these registers are determined by fast interrupt enable registers, interrupt mask register and interrupt source registers. These read-only registers are located on the ARM926EJ-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode.

Address 0x1004_0060 (FIPNDH) Access: Supervisor read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FIPEND[63:48]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FIPEND[47:32]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address 0x1004_0064 (FIPNDL) Access: Supervisor read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FIPEND[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FIPEND[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-24. Fast Interrupt Pending Register High and Low Format

Table 10-25. Fast Interrupt Pending Register High and Low Field Description

Field	Description
31–0 FIPEND	Fast Interrupt Pending Bit. If fast interrupt enable bit is set and the corresponding interrupt source is asserted, interrupt controller will assert fast interrupt request. Fast interrupt pending bits reflect interrupt input lines which are asserted and are currently enabled to generate a fast interrupt. 0 No fast interrupt request 1 Fast interrupt request pending

10.3 ARM926EJ-S Interrupt Controller Operation

10.3.1 ARM926EJ-S Prioritization of Exception Sources

The ARM926EJ-S core imposes the following priority among the various exceptions:

- Reset (highest priority)
- Data Abort
- Fast Interrupt
- Normal Interrupt
- Prefetch Abort
- Undefined Instruction and SWI (lowest priority)

10.3.2 AITC Prioritization of Interrupt Sources

AITC module prioritizes various interrupt sources by source number where higher source numbers have higher priority. Fast interrupt always have higher priority over normal interrupts. Interrupt requests are prioritized in the following sequence:

1. Fast interrupt requests, in order of highest source number
2. Normal interrupt requests, in order of highest priority level, then in order of highest source number with the same priority level

10.3.3 Assigning and Enabling Interrupt Sources

Interrupt controller provides for flexible assignment of any interrupt source to one of the two core interrupt request inputs. This is done by setting the appropriate bits in INTENABLEH/INTENABLEL registers and INTTYPEH/INTTYPEL registers. Usually, interrupt assignment is done once during system initialization and does not affect interrupt latency. Interrupt assignment is the first of three steps required to enable an interrupt source, and this is done at chip integration. The second step is to program the source to generate interrupt requests. The final step is to enable the interrupt inputs in the core by clearing the normal interrupt disable (I) and/or the fast interrupt disable (F) bits in the program status register (CPSR).

10.3.4 Enabling Interrupt Sources

There are two methods of enabling or disabling interrupts in the AITC. The first method is directly reading INTENABLEH/INTENABLEL registers, logically OR or BIT CLEAR these registers with a generated masks, then writing back to INTENABLEH/INTENABLEL registers. The second method is performing an atomic write to source number in INTENNUM register. AITC will decode this 6-bit register and enable one of the 64 interrupt sources. AITC will automatically generate a “one hot” enable mask and logically OR this mask to the correct INTENABLEH or INTENABLEL register. To disable interrupts is the same except the source number is written to the INTDISNUM register.

10.3.5 Typical Interrupt Entry Sequences

Table 10-26 shows a typical pipeline sequence for ARM926EJ-S core when a normal interrupt occurs, assuming single cycle memories, it approximately takes 6 clocks from normal interrupt acknowledgment within ARM926EJ-S to fetch first opcode of interrupt routine. Table 10-27 shows a typical pipeline sequence for ARM926EJ-S core when a fast interrupt occurs, assuming that FIQ service routine begins at 0x0000_001C and single cycle memories.

Table 10-26. Typical Hardware Accelerated Normal Interrupt Entry Sequence

ADDR	TIME										
	–2	–1	0	1	2	3	4	5	6	7	8
	nIRQ assert		nIRQ ack.								
Last ADDR before nIRQ	Fetch	Dec	Exec	Link	Adjust						
+4 / +2		Fetch	Dec								
+8 / +4			Fetch								
0x0000_0018				Fetch	Dec	Exec	Data	Wrbk			
+4					Fetch	Dec					
+8						Fetch					
Vector Table							Vector				
n/a											
nIRQ Routine									Fetch	Dec	Exec

Table 10-26. Typical Hardware Accelerated Normal Interrupt Entry Sequence (continued)

ADDR	TIME										
	–2	–1	0	1	2	3	4	5	6	7	8
+4										Fetch	Dec
+8											Fetch

Table 10-27. Typical Fast Interrupt Entry Sequence

ADDR	TIME					
	–2	–1	0	1	2	3
	nFIQ assert		nFIQ ack.			
Last ADDR before nFIQ	Fetch	Dec	Exec	Link	Adjust	
+4 / +2		Fetch	Dec			
+8 / +4			Fetch			
0x0000_001c				Fetch	Dec	Exec
+4					Fetch	Dec
+8						Fetch

10.3.6 Writing Reentrant Normal Interrupt Routines

AITC can be used to create a reentrant normal interrupt system. This enables preempting of lower priority level interrupts by higher priority level interrupts. This requires a small amount of software support and overhead.

1. Push the link register (LR_irq) on to the stack (SP_irq)
2. Push the saved status register (SPSR_irq) on to the stack
3. Read the current value of NIMASK and push this value on to the stack
4. Read current priority level via NIVECSR
5. Interrupts of the equal or lesser priority than the current priority level must be masked via the NIMASK register by writing value from NIVECSR
6. Clear I bit in ARM926EJ-S core by a MSR or MRS command sequence (now a higher priority normal interrupt can preempt a lower priority one). Also change operating mode of the core to System Mode from IRQ mode
7. Push System Mode link register (LR) on to the stack (SP_user)
8. The traditional interrupt service routine is now included
9. Pop System Mode link register (LR) from the stack (SP_user)
10. Set I bit in ARM926EJ-S core by MSR or MRS command sequence (disables all normal interrupts)
11. Also change the operating mode of the core to IRQ Mode from System mode
12. Pop the original value of normal interrupt mask and write to the NIMASK register

13. The saved status register must be popped from the stack (SP_irq)
14. The link register must be popped from the stack into the PC
15. Return from nIRQ

NOTE

Steps 1, 2, 13, and 14 are automatically done by most C compilers and are included for completeness.

10.3.7 AHB Interface of AITC

AITC is AHB compliant. This means, IDLE or BUSY cycles which are presented to AITC will receive an aitc_hready (as required by specification).

Book II, Part 2: Security

Introduction

This part provides an overview of the modules that make up the i.MX27 security systems.

Security Controller (SCC)

The Security Controller (SCC) is a hardware component composed of two sub-blocks, the Secure RAM and the Security Monitor.

The primary functionality of the SCC is associated with establishing the following:

- A centralized security state controller and hardware security state with a hardware configured, unalterable security policy
- An uninterruptable hardware mechanism to detect and respond to threat detection signals (specifically platform test access signals)
- A device-unique data protection/encryption resource to enable off chip storage of security sensitive data
- An internal storage resource that automatically and irrevocably destroys plain text security sensitive data upon threat detection

Symmetric/Asymmetric Hashing and Random Accelerator (SAHARA2)

Symmetric/Asymmetric Hashing and Random Accelerator (SAHARA2) is a security co-processor that can be used on cell phone baseband processors or wireless PDAs. It implements block encryption algorithms, (AES, DES, and 3DES), hashing algorithms (MD5, SHA-1, SHA-224, and SHA-256), stream cipher algorithm (ARC4), and a hardware random number generator. It has a slave IP bus interface for the host to write configuration and command information, and to read status information. It also has a DMA controller, with an AHB bus interface, to reduce the burden on the host to move the required data to and from memory.

Run Time Integrity Checker (RTIC)

The Run Time Integrity Checker (RTIC) ensures the integrity of the peripheral memory contents and assist with boot authentication. The RTIC has the ability to verify the memory contents during system boot and during run time execution. If the memory contents at runtime fail to match the hash signature, an error in the security monitor is triggered.

IC Identification (IIM)

The IC Identification Module (IIM) provides an interface for reading and in some cases programming and/or overriding identification and control information stored in on-chip fuse elements. The module supports electrically-programmable poly fuses (e-Fuses).

The IIM also provides a set of volatile software-accessible signals which can be used for software control of hardware elements, not requiring non-volatility.

Chapter 11
Security Controller (SCC)

The Security Controller (SCC) is composed of two sub-blocks, the Secure RAM and the Security Monitor (see Figure 11-1).

The primary functionality of the SCC is associated with establishing the following:

- A centralized security state controller and a hardware security state with a hardware configured, unalterable security policy
- An uninterruptible hardware mechanism that detects and responds to threat detection signals (specifically, platform test access signals)
- A device-unique data protection/encryption resource that enables off-chip storage of security-sensitive data

An internal storage resource that automatically and irrevocably destroys plain text security-sensitive data upon threat detection.

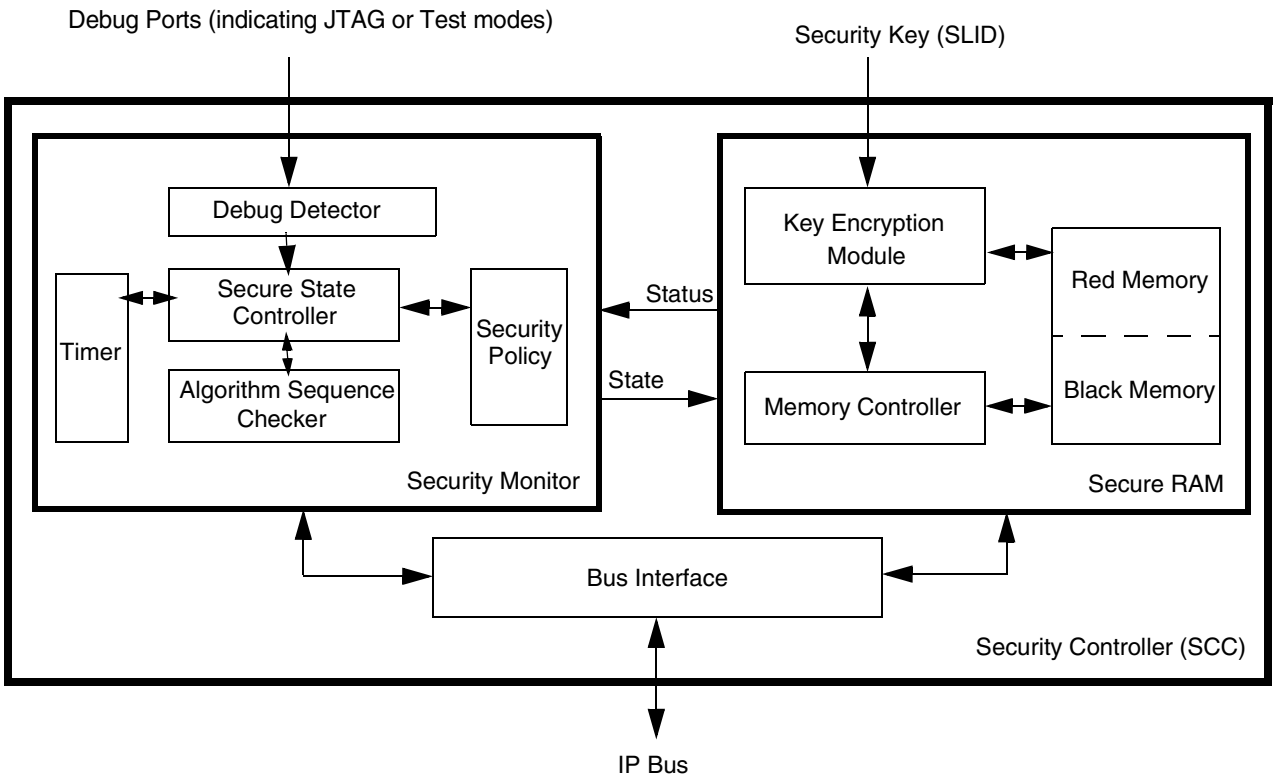


Figure 11-1. Security Controller Block Diagram

11.1 Overview

Security and security services, in an embedded or data processing platform, refer to the platform's ability to provide mandatory and optional information protection services. Information in this context refers to all embedded data, both to program store and data load. Therefore, a secure platform is intended to protect information and data from unauthorized access in the form of inspection (read), modification (write), or execution (use).

11.2 External Signal Description

The SCC has no external signals.

NOTE

Contact your Freescale Semiconductor sales office or distributor for additional information on SCC.

Chapter 12
Symmetric/Asymmetric Hashing and Random Accelerator (SAHARA2)

The Symmetric/Asymmetric Hashing and Random Accelerator (SAHARA2) is a security co-processor that can be used on cell phone baseband processors or wireless PDAs. It implements block encryption algorithms, (AES, DES, and 3DES), hashing algorithms (MD5, SHA-1, SHA-224, and SHA-256), stream cipher algorithm (ARC4), and a hardware random number generator. It has a slave IP bus interface for the host to write configuration and command information, and to read status information. It also has a DMA controller, with an AHB bus interface, to reduce the burden on the host to move the required data to and from memory.

12.1 Features

SAHARA2 accelerates the following security functions:

- AES encryption/decryption
  - — ECB, CBC, CTR, and CCM modes
  - — 128 bit key
- DES/3DES
  - — EBC, CBC and CTR modes
  - — 56-bit key with parity (DES)
  - — 112-bit or 168-bit key with parity (3DES)
- ARC4 (RC4-compatible cipher)
  - — 5-16 byte key
  - — Host accessible S-box
- MD5, SHA-1, SHA-224 and SHA-256 hashing algorithms.
  - — Messages lengths which are multiples of bytes.
  - — Autopadding supported.
  - — HMAC (support for IPAD and OPAD via descriptors).
  - — Up to 2^{32} byte message length.
- Random number generator (based NIST Approved PRNG - FIPS 186-2).
  - — Entropy is generated via an independent free running ring oscillators

SAHARA2 also provides the following enhanced features:

- Descriptor based processing to reduce communication between host processor and SAHARA2
- Low power design

  - Automatic power down of individual blocks when not in use
  - Clock gating on registers
  - RNG sleep mode
- Restricted access to potentially sensitive information
  - Internal registers are cleared after descriptor chain has completed processing in BATCH mode
  - Security Monitor can cause data to be cleared.
  - Scan reset and scan exit signals prevent data being scanned out.
- Mixed Endianness support.

NOTE

Contact your Freescale Semiconductor sales office or distributor for additional information on SAHARA2.

Chapter 13
Run-Time Integrity Checker (RTIC)

The Run-Time Integrity Checker (RTIC) function is to ensure the integrity of the peripheral memory contents, and assist with boot authentication. The RTIC has the ability to verify the memory contents during system boot and during run-time execution. If the memory contents at run-time fail to match the hash signature, an error in the security monitor is triggered. Figure 13-1 is a block diagram of the RTIC.

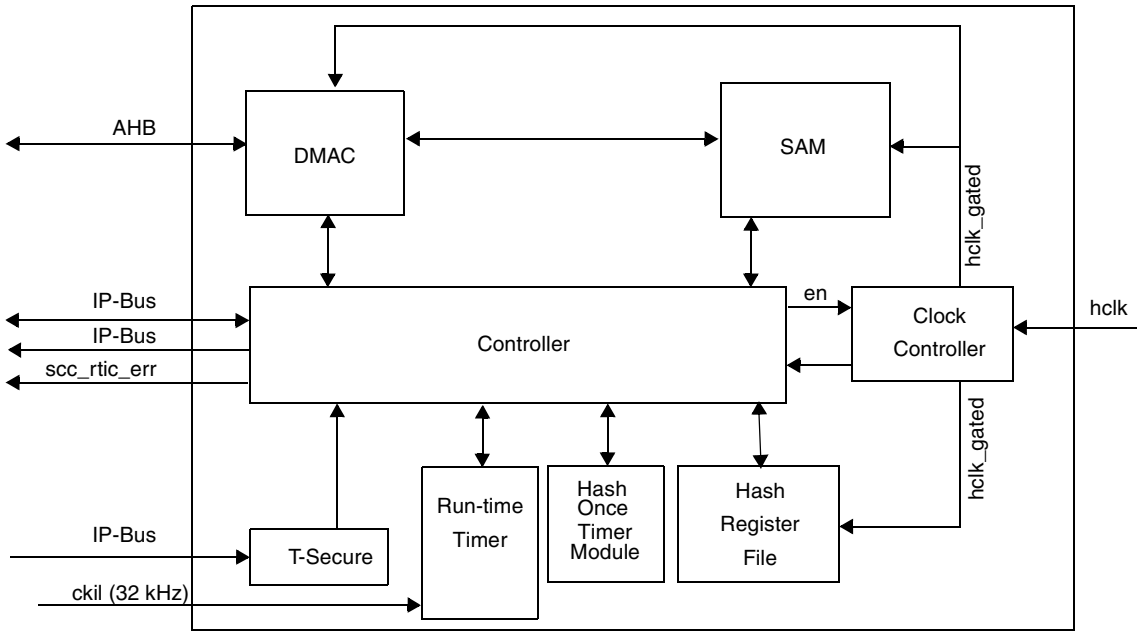


Figure 13-1. RTIC Block Diagram

13.1 Features

The RTIC offers the following features:

- SHA-1 message authentication
- Input DMA interface
- Segmented data gathering to support non-contiguous data blocks in memory (up to two segments per block)
- Works with high assurance boot process
- Support for up to four independent memory blocks
- Programmable DMA bus duty cycle timer and watchdog timer
- Power-saving clock gating logic

- Hardware configurable Big/Little-Endian data format
- Full word memory reads (word-aligned addresses, multiple of 32-bit lengths)

13.1.1 Modes of Operation

The RTIC operates in two primary modes:

- One-time hash mode
  - Is used during high assurance boot for code authentication or one time integrity checking
  - Stores hash result internally and signals interrupt to host
- Continuous hash mode
  - Is used at run-time to continuously to verify integrity of memory contents
  - Checks re-generated hash against internally stored values and interrupts host only if error occurs

13.2 Initialization/Application Information

13.2.1 System Application

The RTIC is intended to serve as a single-use hash accelerator to assist with code authentication and other services at boot time, and as an autonomous/passive memory integrity checker during run-time. It is programmed through the IP-slave interface, and scans the peripheral memory contents over the AHB interface using direct memory access. A typical system configuration using the RTIC is shown in Figure 13-2.

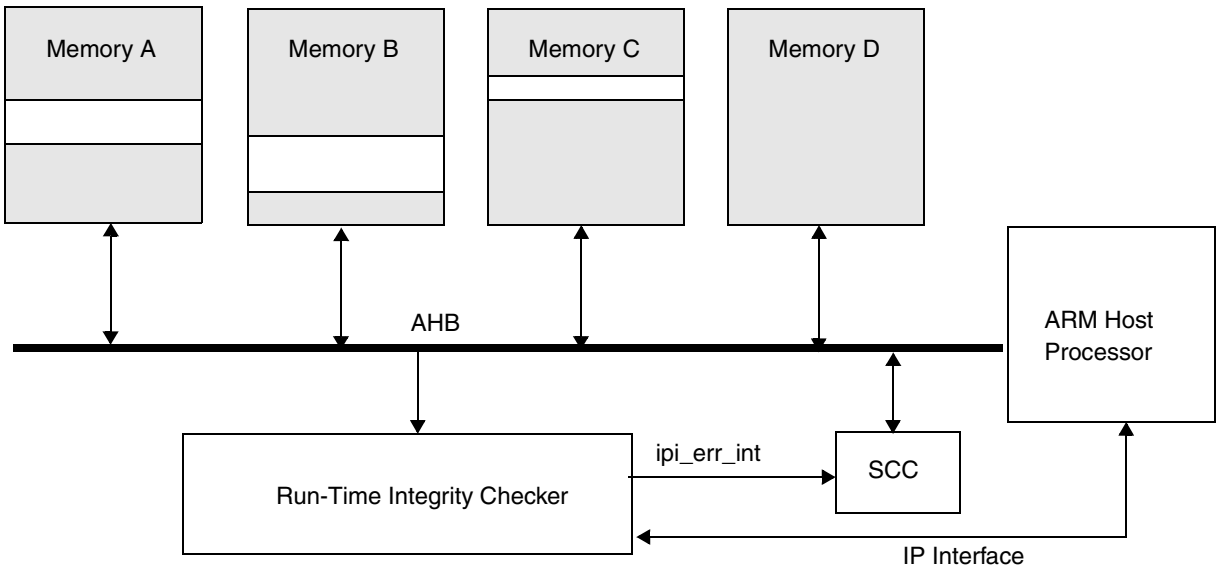


Figure 13-2. System Diagram

In this example, there are four independent memory blocks that can be checked by the RTIC. Memories A,B, and C have their contents partitioned over non-contiguous spaces. Memory D does not contain any

physical partitioning. The host would program the RTIC with the starting address and length of each partition inside memory A, B, and C. For memory D, only one starting address and length would be specified, with the second start address/length fields for memory D being set to 0.

After setting the A/B/C/D hash once memory enable bits in the RTIC control register and hash once bit in the RTIC command register, the RTIC hashes each memory and stores the result in its hash register file to be read by the host. If the RTIC is used to verify that the memories are not corrupted during run-time, the A/B/C/D run-time memory enable bits in the control register must be set, followed by the tun time check bit in the RTIC command register. The RTIC re-hashes each enabled memory in a continuous loop until either an error occurs or the RTIC is reset.

NOTE

Contact your Freescale Semiconductor sales office or distributor for additional information on RTIC.

Chapter 14
IC Identification (IIM)

The IC Identification Module (IIM) provides an interface for reading and, in some cases, programming and/or overriding identification and control information stored in on-chip fuse elements.

The IIM also provides a set of volatile software-accessible signals that can be used for software control of hardware elements, not requiring non-volatility.

14.1 Overview

The IIM provides the primary user-visible mechanism for interfacing with on-chip fuse elements. Among the uses for the fuses are unique chip identifiers, mask revision numbers, cryptographic keys, and various control signals requiring permanent non-volatility. The IIM also provides up to 28 volatile control signals.

The IIM consists of a master controller and a set of registers to hold the values of signals visible outside the module. Up to eight arrays of fuses (e-Fuses) are associated with the IIM.

The IIM is accessible using an 8-bit IP bus interface. An 8-bit interface is used because it matches the natural width of the fuse arrays.

14.1.1 Features

- Up to eight independent fuse banks (number of fuse banks and size of the bank are parameterized)
- Maximum usable fuse bank size is 2048 bits
- Laser- and e-Fuse banks may be intermixed on a per bank basis
- Support for driving secure JTAG challenge and response values to the SJC (size of each field configurable using RTL parameter; challenge default size is 64 bits, response default size is 56 bits)
- Up to 28 externally visible software-controlled volatile signals (driving SoC-level nets for feature enabling) lockable in groups of 7
- Ability to provide up to two distinct 168-bit 3DES keys from a single set of fuses
- Ability to override fuse values in software (does not affect the fuse element); override capability can be permanently disabled on a per-bank basis
- Ability to write-protect e-Fuses on a per-bank basis
- Ability to scan-protect (read and program) on a per-bank basis
- Fuses may be programmed by software, directly by JTAG, or indirectly by JTAG using a processor
- Recommended signal assignments to maximize software re-use

NOTE

Contact your Freescale Semiconductor sales office or distributor for additional information on IIM.

Book II, Part 3: External Interfaces

Introduction

The i.MX27 processor contains the following external interfaces:

External Memory Interface (EMI)

The External Memory Interface (M3IF) controls all IC external memory accesses (read/write/erase/program) from all the masters in the system, through two Master Port Gaskets (MPG)—(interface [AHB 32-bit] and MPG64 [AHB 64-bit])—to different external memories. All accesses are arbitrated by the Multi Master Memory Interface (M3IF) module and controlled by the respective memory controller.

The EMI contains different external memory controllers to support several memory devices:

- M3IF—Multi Master Memory Interface
- ESDCTL/MDDRC—Enhanced SDRAM/LPDDR memory controller
- PCMCIA—PCMCIA memory controller
- NFC—NAND Flash memory controller
- WEIM—SRAM/PSRAM/FLASH memory controller

Multi-Master Memory Interface (M3IF)

The Multi-Master Memory Interface (M3IF) controls memory accesses (read/write/erase/program) from one or more masters through different port interfaces to different external memory controllers ESDCTL/MDDRC, PCMCIA, NANDFLASH, and WEIM.

Wireless External Interface Module (WEIM)

The Wireless External Interface Module (WEIM) provides the capability to the system for accessing external Flash and RAM memories connected to either of its six chip selects. It has the ability to provide a single-cycle burst access for external flashes and support for multiple burst devices when not using its smart burst feature. Other features include Big/Endian mode support, Cellular RAM support, and support for multiplexed address/data bus.

Enhanced SDRAM Controller (ESDCTL)

The Enhanced Synchronous Dynamic RAM Controller (ESDCTL) provides interface and control for synchronous DRAM memories for the system. SDRAM memories use a synchronous interface with all signals registered on a clock edge. A command protocol is used for initialization, read, write, and refresh operations to the SDRAM and is generated on the signals by the controller when required due to external or internal requests. It has support for both single data rate RAMs and double data rate SDRAMs. It supports 64, 128, 256, and 512-Mbit, 4 bank synchronous DRAM by two independent chip selects and with up to 64 Mbytes addressable memory per chip select.

NAND Flash Controller (NFC)

The NAND Flash Controller (NFC) device is a type of flash memory that is optimized for data storage applications with its unique cell structure, providing significant cost advantages over conventional NOR flash memory. The NAND Flash Controller integrates the functionality necessary for access to these devices. NAND Flash has a smaller bit cell but has a fast sequential access as compared to a NOR flash which has a large bit cell but a fast random access. This makes NAND Flash perfect as a data memory for storing audio/video files in NAND Flash because typically these files are stored in sequential manner while access to processor code is quite random.

Personal Computer Memory Card International Association (PCMCIA) Controller

The PCMCIA host adapter module provides the control logic for PCMCIA socket interfaces, and requires some additional external analog power switching logic and buffering. The additional external buffers allow the PCMCIA host adapter module to support one PCMCIA socket.

Chapter 15
External Memory Interface (EMI)

The Master Memory Interface (M3IF) is an External Memory Interface that controls all IC external memory accesses (read/write/erase/program) from all the masters in the system, through two port interfaces (MPG (AHB 32 bit) and MPG64 (AHB 64-bit) to different external memories. All accesses are arbitrated by the M3IF module and controlled by the respective memory controller. EMI contains different external memory controllers in order to support several memory devices:

- M3IF—Multi Master Memory Interface
- ESDRAMC/MDDRC—Enhanced SDRAM/LPDDR memory controller
- PCMCIA—PCMCIA memory controller
- NFC—NAND Flash memory controller
- WEIM—SRAM/PSRAM/Flash memory controller

15.1 Overview

The M3IF-ESDCTL/MDDRC interface is optimized and designed to reduce access latency by generating multiple accesses through dedicated ESDCTL/MDDRC arbitration (MAB) module, which controls access to/from the Enhanced SDRAM/MDDR memory controller. For other memory interfaces, M3IF only arbitrates and forwards the masters requests received through the Master Port Gasket (MPG/MPG64) interface (and M3IF arbitration) to the respective memory controller.

When a master request a memory access, the access will immediately be taken by the M3IF if no other access is in progress. The M3IF will forward the access to the respective memory controller (slave), and depending on the respective memory controller state a command to the memory will be generated. If the access can't be started due to a previous active access, the master request pends ("HREADY" held negated) until it will be executed by the memory controller. When the access execution is completed the HREADY will be asserted and a new request can be processed.

EMI provides the ability to connect to a wide variety of memory devices. This chapter contains technical information about the operation and configuration of the EMI modules in the chip to allow the designer to quickly integrate external memory devices into new and existing designs. Several of the modules in the EMI portion of the chip share pins with the PCMCIA, EIM, SDRAMC, and NAND controllers.

The chip contains interfaces for the following types of memory devices:

- PCMCIA
- Flash Memory Devices
- SDRAM/Low Power DDR (LPDDR)

Figure 15-1 shows the M3IF block diagram.

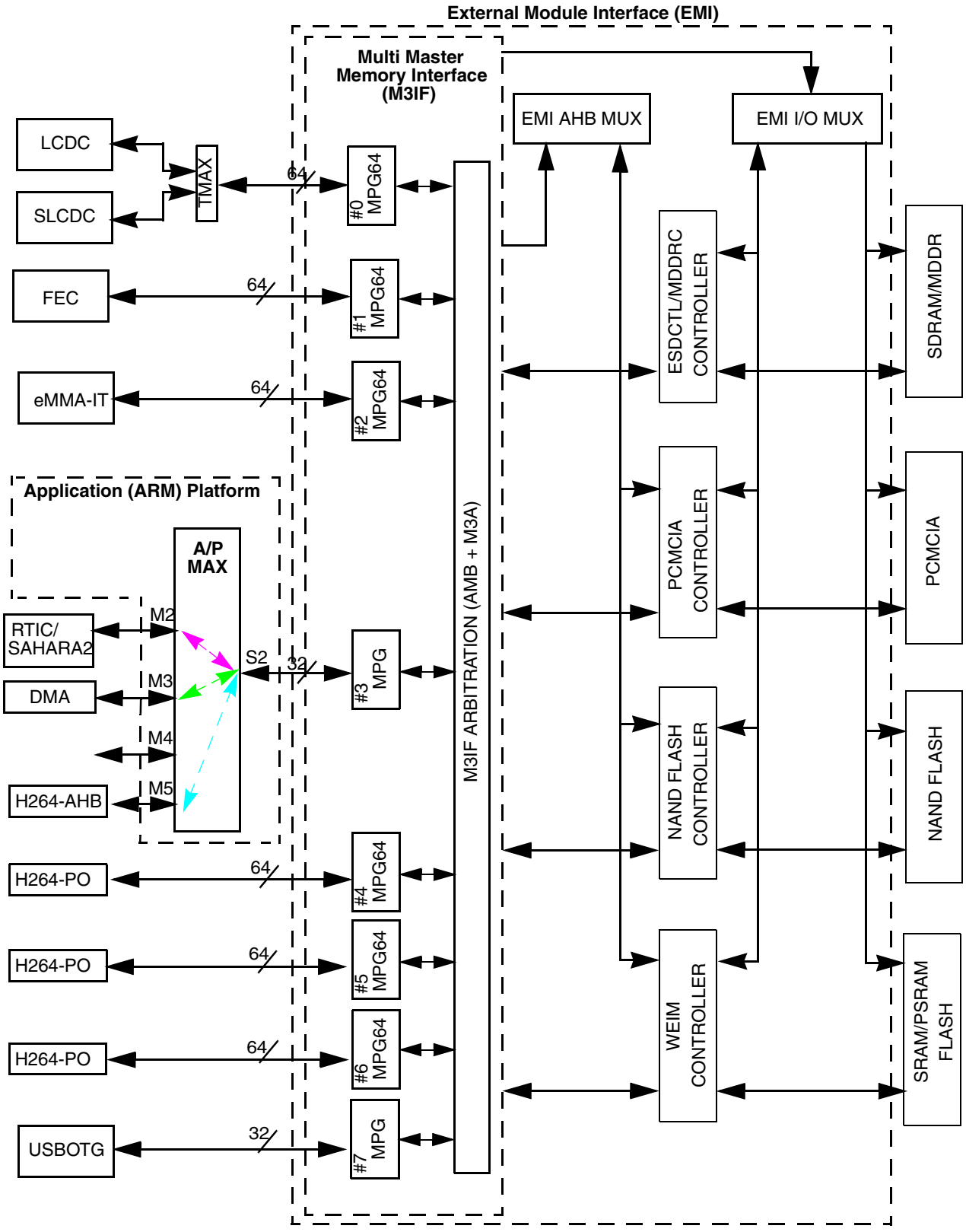


Figure 15-1. M3IF System Block Diagram

15.2 Features

M3IF includes these distinctive features:

- Multi Master Memory Interface (M3IF)
  - Supports multiple requests from 8 masters through two different input ports interfaces:
    Master Port Gasket (MPG)—ARM9 AMBA AHB-Lite bus protocol.
    Master Port Gasket (MPG64)—AMBA AHB access with 64 bits data bus width.
  - Supports memory "snooping," which monitors a region (from 2 Kbytes up to 16 Mbytes) in external memory for write accesses.
- Enables AHB accesses to four different memory controllers (that share some of their I/O pads, through the EMI AHB MUX and EMI I/O MUX)
- Enhanced SDRAM Controller (ESDCTL) or MDDR Controller (MDDRC)
  - Up to two chip selects (due to PADS sharing all 2 chip selects are supported only in case that WEIM CS2 and CS3 are not is use).
  - Supports 32 bit SDR SDRAM (up to 2 Gbytes @133 MHz)
  - Supports 32 bit MDDR SDRAM (up to 2 Gbytes @ 266 MHz)
- NAND Flash Controller (NFC)
  - 8/16 bit NAND Flash (up to 2 Gbyte address space)
  - 2-Kbyte RAM Internal Buffer
- Personal Computer Memory Card International Association Controller (PCMCIA)
  - Support PCMCIA Rel 2.1
  - Compact Flash
  - PC Card
  - TrueID Mode
- Wireless External Interface Memory Controller (WEIM)
  - Up to 6 chip selects (due to PADS sharing all 6 chip selects are supported only in case that both ESDCTL/MDDRC chip selects are not is use).
  - Supports 16-bit SRAM memories
  - Supports 16-bit PSRAM (up to 133 MHz) memories
  - Support s16-bit (NOR) Flash memories

15.3 PCMCIA Host Adapter

The Personal Computer Memory Card International Association (PCMCIA) interface provides a glueless interface to devices that comply with the PCMCIA association standard PCMCIA 2.1, which defines usage of memory and I/O devices as insertable and exchangeable peripherals for personal computers or PDAs. Examples of these types of devices include Compact Flash and WLAN adapters. Figure 15-2 shows a simplified block diagram of the PCMCIA controller.

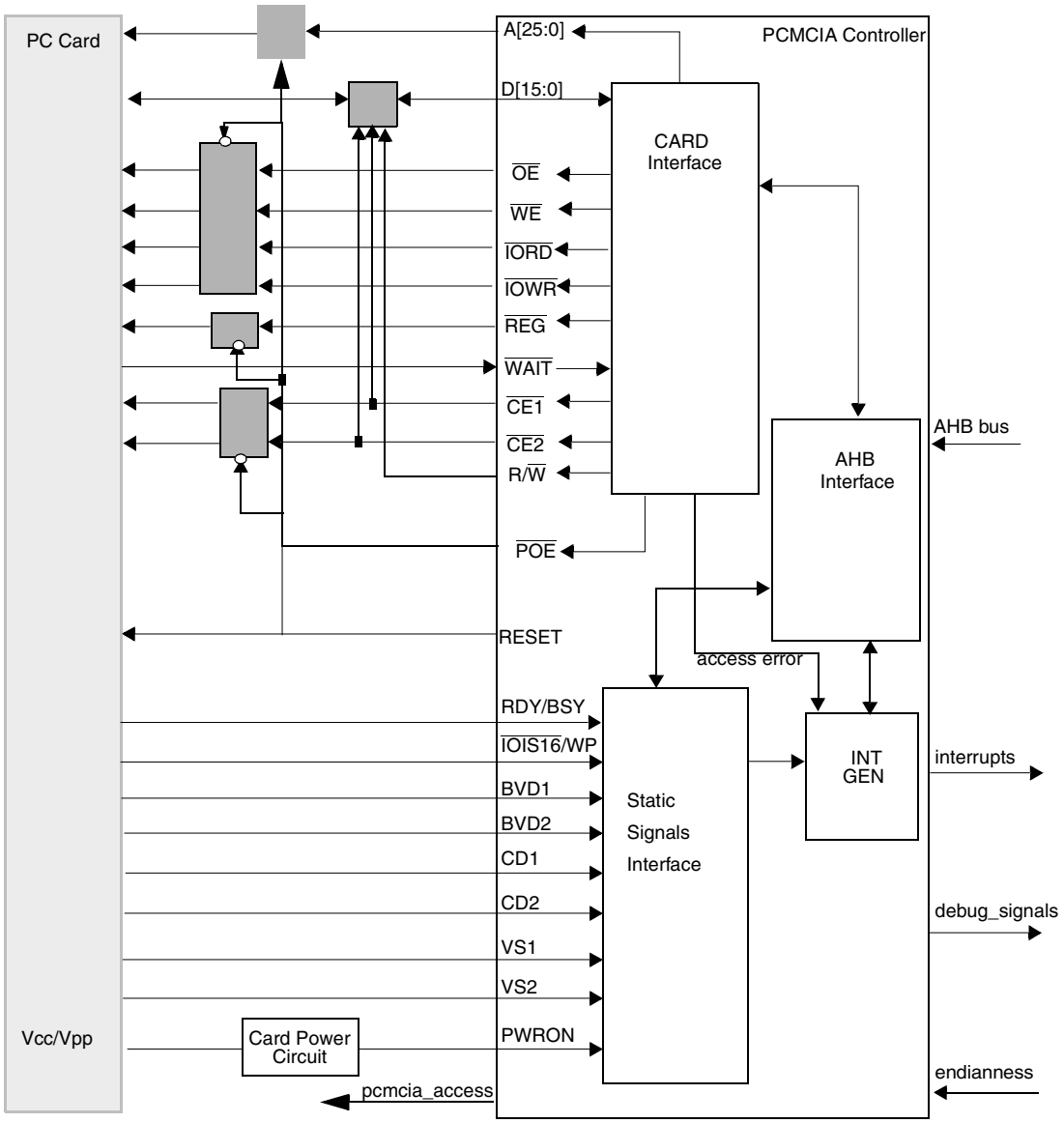


Figure 15-2. PCMCIA Host Adapter Simplified Block Diagram

PCMCIA host adapter module provides the all the necessary control logic for a single PCMCIA socket and only requires some additional external analog power switching logic and buffering for PC card operations. PCMCIA host adapter module can support one PCMCIA socket. PCMCIA controller shares its pins with other modules in the EIM area of the chip. The modules that share pins with PCMCIA are EIM, SDRAMC and NAND controllers.

15.3.1 Interrupt Generation

There are 14 interrupt sources in PCMCIA controller. In addition, PCMCIA generates a signal which is a locator of all the possible interrupts. It is up to the system's integrator to decide which signal(s) to connect to the system's interrupt controller module. PCMCIA Input Pins Register (PIPR) reports any change of inputs from the PCMCIA card to the host (BVD,CD,RDY,VS). PCMCIA Controller Status Changed

Register (PSCR) contents are logically ANDed with PCMCIA Controller Enable Register (PER) to generate a PCMCIA controller interrupt. The interrupt level is user programmable and the PCMCIA controller can generate an additional interrupt for RDY/IREQ that can trigger upon a level (low or high) change or edge (fall or rise) of the input signal.

15.3.2 Card Extraction

When a PC card is extracted the PCMCIA controller's registers are not reset. The registers settings remain the same as before the card's extraction. This allows the host software to quickly activate the card once the CIS indicates that it's the same card on reinsertion.

15.3.3 TrueIDE Support

The ATA standard specifies an AT attachment interface between the host systems and storage devices. PCMCIA controller can be dynamically configured to support a PCMCIA-compatible ATA disk interface (commonly known as IDE) instead of the standard PCMCIA card interface. Using the TrueIDE interface on the PCMCIA controller changes the function of some card socket signals to support the needs of ATA disk interface.

15.4 NAND Flash Controller (NFC)

The NFC module interfaces standard NAND Flash devices to the IC and hides the complexities of accessing a NAND Flash memory device. It provides a glueless interface to both 8-bits and 16-bits NAND Flash parts with page sizes of 512 Bytes or 2 Kilobytes and densities up to 2 Gbit. Figure 15-3 shows a simplified block diagram of the NAND Flash controller.

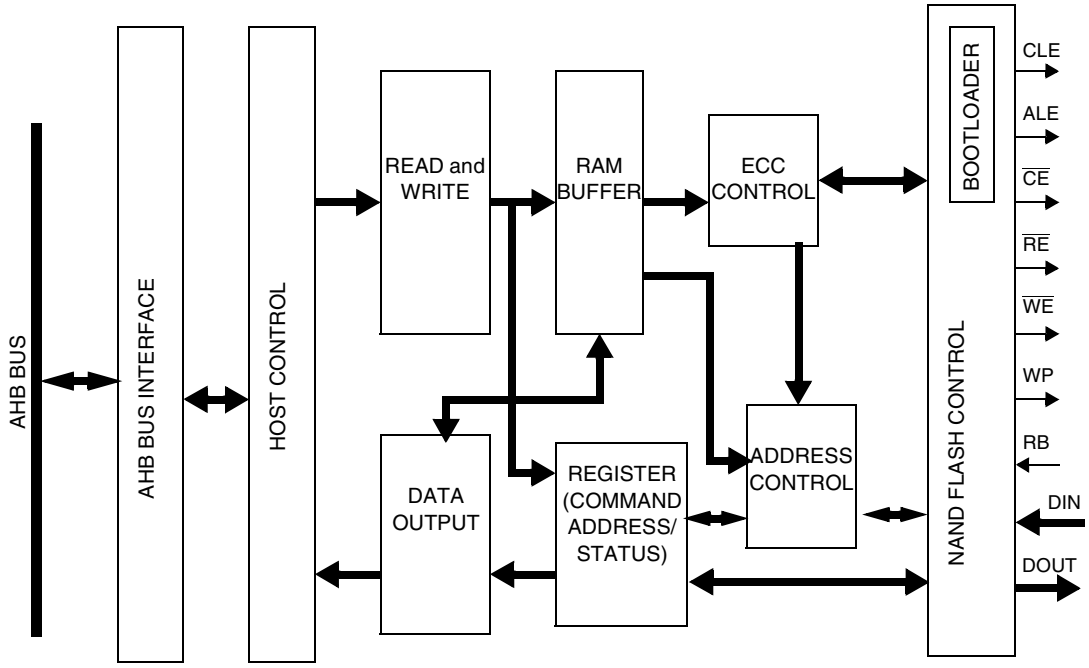


Figure 15-3. NAND Flash Controller Simplified Block Diagram

15.4.1 Operation

Communicating with a Flash memory device begins by the AHB host initiating a read from the NAND Flash controller (NFC). This is accomplished by configuring the NFC and then waiting for an interrupt from the Flash memory device to be generated. When the NFC receives the interrupt, it inputs a page from the Flash memory device, and upon completion, generates an interrupt to the AHB Host. When AHB host receives the NFC interrupt, it reads the content from the internal RAM buffer of the NFC. To complete the operation the AHB host checks the status of operation by reading the NFC status registers.

Data that is exchanged with the Flash memory device is temporarily maintained in a 2 kilobyte RAM buffer. This buffer is used as the boot RAM during a cold reset (if the IC is configured for a boot to be carried out from the NAND Flash device). After the boot load completes, the RAM is available as buffer RAM for normal Flash memory operations.

15.4.1.1 Internal and External Communications

To ensure the greatest degree of flexibility, NFC provides an internal X16 bit and X32 bit interface to the AHB bus allowing, 16-bit or 32-bit bus transfers, and a pin selectable X8 or X16 interface to the external NAND Flash memory device.

All communications between the NFC and the ARM9 platform is accomplished through the AHB host. Configuration and control of the NFC by the host is done using 14 16-bit registers. NFC generates all the control signals that controls the NAND Flash: $\overline{CE}$ (Flash Chip Enable), $\overline{RE}$ (Read Enable for read operations), $\overline{WE}$ (Flash Write Enable), CLE (Flash Command Latch Enable), ALE (Flash Address Latch Enable). It also monitors the R/nB (Flash Ready/Busy indication) signal to check if the NAND Flash memory device is currently in the middle of an operation. Flash memory data's integrity is monitored by automatic generation of ECC code of data during NFCs data loading from NAND Flash memory devices.

15.4.1.2 Sharing of I/O Pins

The NFC provides necessary logic to share I/O pins with pins of another memory controller. NFC state machine halts when a request to free the pins is asserted. NAND Flash signals when it finishes the existing transfer allowing other memory controller to control them. Since NAND Flash memory accesses are typically long and relatively slow, priority is given to other memory controller sharing the pins. NFC must wait until other memory controller is finished with its operation and the pins are free before it can continue with its accesses. One example for this pin muxing is sharing the 16 I/O pins of the NAND Flash Controller with the Data pins of the Wireless External Interface Module (WEIM) when interfacing to a PSRAM.

15.5 Enhanced SDRAM Controller (ESDRAMC)

The ESDRAMC module provides interface, configuration and control for many different types synchronous SDRAM and Low Power Mobile DDR (LPDDR) memories. Figure 18-1 is the Enhanced SDRAM Controller top-level diagram that shows the functional organization of the block. Enhanced SDRAM Controller consists of 9 major blocks, including the SDRAM command state machine controller, bank register (page and bank address comparators), Row/Column Address Multiplexer, configuration registers, refresh request counter, command sequencer, size logic (splitting access), data path (data aligner/multiplexer), LPDDR interface and the Power Down timer.

15.6 M3IF AHB MUX

The M3IF AHB MUX module controls traffic on the AHB bus (address and controls) between the memory controllers and the IC. M3IF uses several muxes/glue logic to control the traffic on the AHB bus. Only several AHB signals/busses are routed through the EMI AHB MUX toward the memory controllers. Most of the AHB busses (data, address and controls) are directly routed from the M3IF to the memory controllers.

15.6.1 Overview of EMI AHB MUX Operation

Figure 15-4 illustrates EMI AHB MUX block diagram. The interface is ARM's 11 AMBA-AHB-Lite compliant (does not support RETRY and SPLIT transfers). All AHB signals that are not shown in Figure 15-4 are directly routed between the M3IF and the relevant memory controllers. For the entire list of AHB signals refer to Table 15-3 and to the relevant memory controller specification document. EMI AHB MUX generates HSEL signals for all memory controllers except ESDCTL (generated within the M3IF due to latency hiding logic).

15.7 M3IF I/O MUX

M3IF I/O MUX controls the traffic (data, address and controls) between the memory controllers and the external devices (via the IC I/O MUX/PADS), and vice versa, for example, from the external devices to the memory controllers. The M3IF uses several muxes/glue logic to control the traffic. Refer Figure 15-4 and Figure 15-5 for a top level diagram of the EMIAHB and the EMI I/O MUX.

Only shared IC PADS signals/busses are routed through the EMI I/O MUX toward the external devices. Signals (mainly controls) that have dedicated PADS are directly routed from the memory controllers to the external devices.

15.7.1 Overview of EMI I/O MUX Operation

The signals not shown in Figure 15-5 are directly routed between the memory controllers and the respective external device. For entire list of signals, refer Table 15-3 and the relevant memory controller section. The select for muxes is CHOOSEN_SLAVE[1:0] bus driven by M3IF. CHOOSEN_SLAVE encoding is listed in Table 15-1 which summarizes EMI outputs to IC pads (dedicated and shared among all memory controllers).

Table 15-1. CHOOSEN_SLAVE Encoding

CHOOSEN_SLAVE Value	Selected Memory Controller
00	ESDCTL/MDDRC
01	WEIM
10	PCMCIA
11	NFC

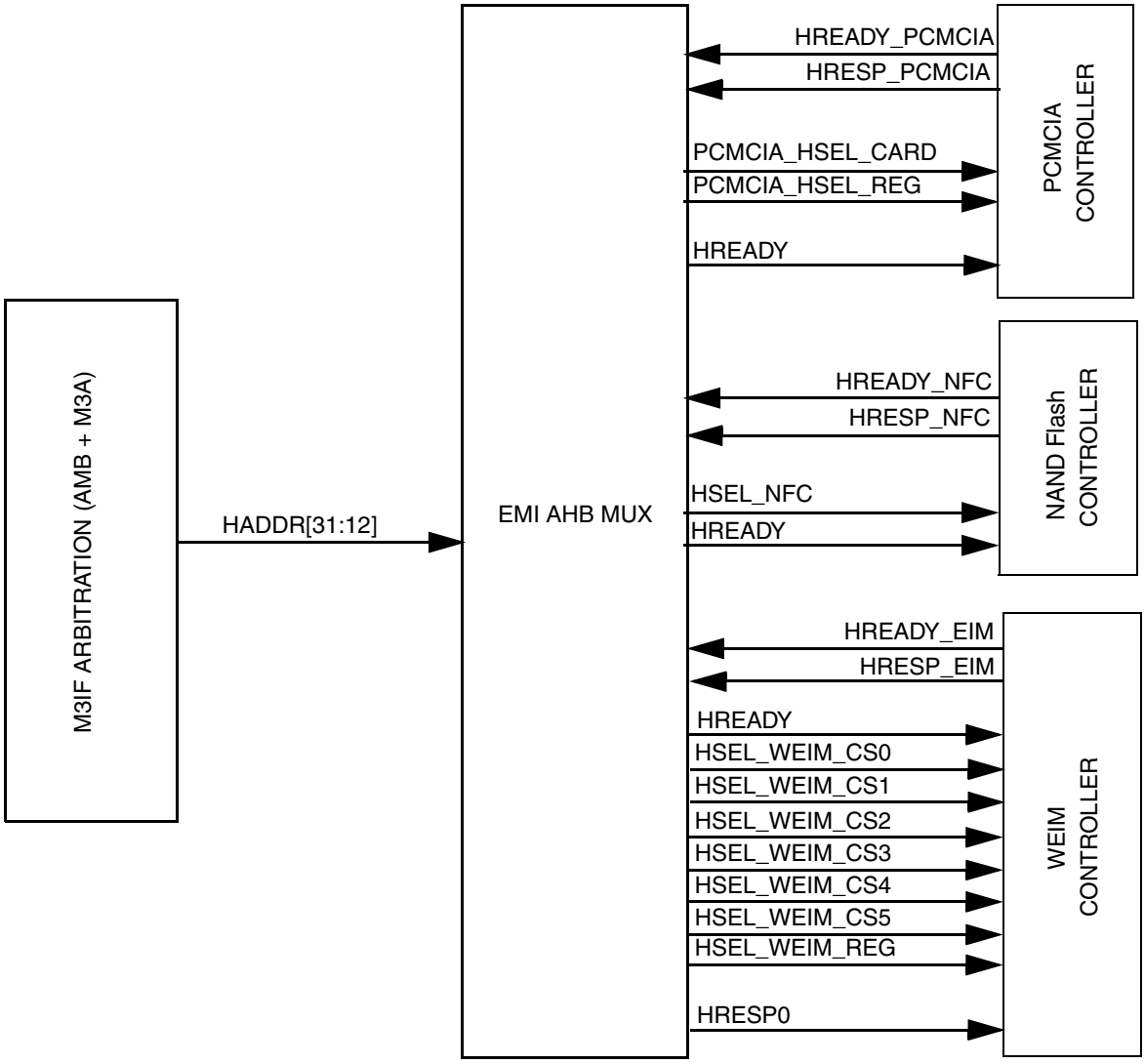


Figure 15-4. EMI AHB MUX Interface Diagram

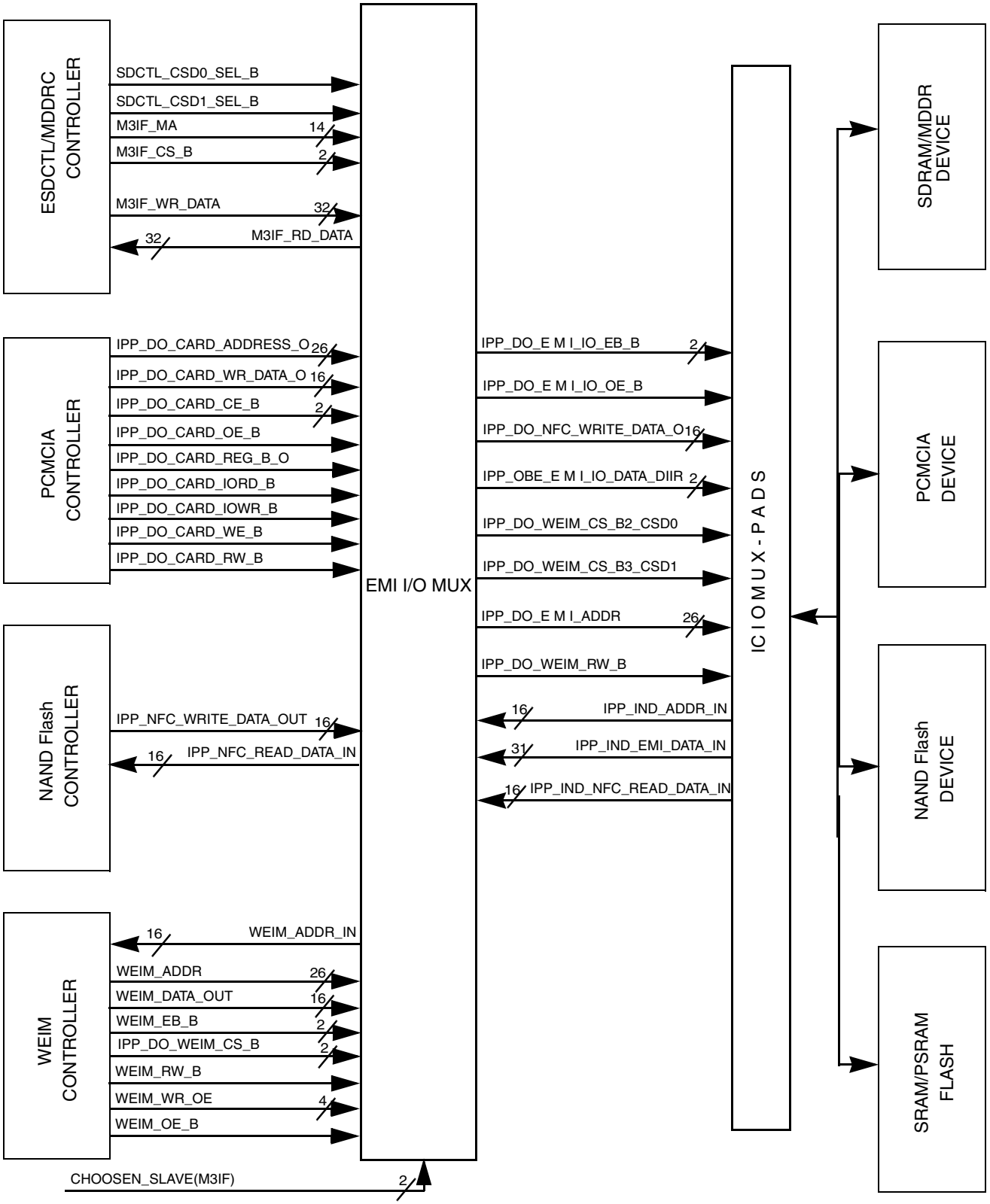


Figure 15-5. EMI I/O MUX Interface Diagram

Table 15-2. External Memory Interface I/O MUX Description

MEMORY Controller Outputs					EMI Output	IC Pin Name
SDRAMC	MDDRC	PCMCIA	WEIM	NFC		
WEIM/ESDCTL/PCMCIA Address MUXING/WEIM Muxed Mode Data[15:0]						
M3IF_MA[0]		IPP_DO_CARD_ADDRESS_O[0]	WEIM_ADDR_DATA_OUT[0]	—	IPP_DO_EMI_ADDR [0]	A0
—		—	WEIM_ADDR_DATA_IN[0]		IPP_IND_ADDR_IN [0]	
M3IF_MA[1]		IPP_DO_CARD_ADDRESS_O[1]	WEIM_ADDR_DATA_OUT[1]	—	IPP_DO_EMI_ADDR [1]	A1
—		—	WEIM_ADDR_DATA_IN[1]		IPP_IND_ADDR_IN [1]	
M3IF_MA[2]		IPP_DO_CARD_ADDRESS_O[2]	WEIM_ADDR_DATA_OUT[2]	—	IPP_DO_EMI_ADDR [2]	A2
—		—	WEIM_ADDR_DATA_IN[2]		IPP_IND_ADDR_IN [2]	
M3IF_MA[3]		IPP_DO_CARD_ADDRESS_O[3]	WEIM_ADDR_DATA_OUT[3]	—	IPP_DO_EMI_ADDR [3]	A3
—		—	WEIM_ADDR_DATA_IN[3]		IPP_IND_ADDR_IN [3]	
M3IF_MA[4]		IPP_DO_CARD_ADDRESS_O[4]	WEIM_ADDR_DATA_OUT[4]	—	IPP_DO_EMI_ADDR [4]	A4
—		—	WEIM_ADDR_DATA_IN[4]		IPP_IND_ADDR_IN [4]	
M3IF_MA[5]		IPP_DO_CARD_ADDRESS_O[5]	WEIM_ADDR_DATA_OUT[5]	—	IPP_DO_EMI_ADDR [5]	A5
—		—	WEIM_ADDR_DATA_IN[5]		IPP_IND_ADDR_IN [5]	
M3IF_MA[6]		IPP_DO_CARD_ADDRESS_O[6]	WEIM_ADDR_DATA_OUT[6]	—	IPP_DO_EMI_ADDR [6]	A6
—		—	WEIM_ADDR_DATA_IN[6]		IPP_IND_ADDR_IN [6]	
M3IF_MA[7]		IPP_DO_CARD_ADDRESS_O[7]	WEIM_ADDR_DATA_OUT[7]	—	IPP_DO_EMI_ADDR [7]	A7
—		—	WEIM_ADDR_DATA_IN[7]		IPP_IND_ADDR_IN [7]	
M3IF_MA[8]		IPP_DO_CARD_ADDRESS_O[8]	WEIM_ADDR_DATA_OUT[8]	—	IPP_DO_EMI_ADDR [8]	A8
—		—	WEIM_ADDR_DATA_IN[8]		IPP_IND_ADDR_IN [8]	

Table 15-2. External Memory Interface I/O MUX Description (continued)

MEMORY Controller Outputs					EMI Output	IC Pin Name
SDRAMC	MDDRC	PCMCIA	WEIM	NFC		
M3IF_MA[9]		IPP_DO_CARD_ADDRESS_O[9]	WEIM_ADDR_DATA_OUT[9]	—	IPP_DO_E M I _ADDR [9]	A9
—		—	WEIM_ADDR_DATA_IN[9]		IPP_IND_ADDR_IN [9]	
—		IPP_DO_CARD_ADDRESS_O[10]	WEIM_ADDR_DATA_OUT[10]	—	IPP_DO_E M I _ADDR [10]	A10
—		—	WEIM_ADDR_DATA_IN[10]		IPP_IND_ADDR_IN [10]	
M3IF_MA[11]		IPP_DO_CARD_ADDRESS_O[11]	WEIM_ADDR_DATA_OUT[11]	—	IPP_DO_E M I _ADDR [11]	A11
—		—	WEIM_ADDR_DATA_IN[11]		IPP_IND_ADDR_IN [11]	
M3IF_MA[12]		IPP_DO_CARD_ADDRESS_O[12]	WEIM_ADDR_DATA_OUT[12]	—	IPP_DO_E M I _ADDR [12]	A12
—		—	WEIM_ADDR_DATA_IN[12]		IPP_IND_ADDR_IN [12]	
M3IF_MA[13]		IPP_DO_CARD_ADDRESS_O[13]	WEIM_ADDR_DATA_OUT[13]	—	IPP_DO_E M I _ADDR [13]	A13
—		—	WEIM_ADDR_DATA_IN[13]		IPP_IND_ADDR_IN [13]	
—		IPP_DO_CARD_ADDRESS_O[14]	WEIM_ADDR_DATA_OUT[14]	—	IPP_DO_E M I _ADDR [14]	A14
			WEIM_ADDR_DATA_IN[14]		IPP_IND_ADDR_IN [14]	
—		IPP_DO_CARD_ADDRESS_O[15]	WEIM_ADDR_DATA_OUT[15]	—	IPP_DO_E M I _ADDR [15]	A15
			WEIM_ADDR_DATA_IN[15]		IPP_IND_ADDR_IN [15]	
—		IPP_DO_CARD_ADDRESS_O[16]	WEIM_ADDR_OUT[16]	—	IPP_DO_E M I _ADDR [16]	A16
—		IPP_DO_CARD_ADDRESS_O[17]	WEIM_ADDR_OUT[17]	—	IPP_DO_E M I _ADDR [17]	A17
—		IPP_DO_CARD_ADDRESS_O[18]	WEIM_ADDR_OUT[18]	—	IPP_DO_E M I _ADDR [18]	A18
—		IPP_DO_CARD_ADDRESS_O[19]	WEIM_ADDR_OUT[19]	—	IPP_DO_E M I _ADDR [19]	A19
—		IPP_DO_CARD_ADDRESS_O[20]	WEIM_ADDR_OUT[20]	—	IPP_DO_E M I _ADDR [20]	A20

Table 15-2. External Memory Interface I/O MUX Description (continued)

MEMORY Controller Outputs					EMI Output	IC Pin Name
SDRAMC	MDDRC	PCMCIA	WEIM	NFC		
—		IPP_DO_CARD_ADDRESS_O[21]	WEIM_ADDR_OUT[21]	—	IPP_DO_E M I _ADDR [21]	A21
—		IPP_DO_CARD_ADDRESS_O[22]	WEIM_ADDR_OUT[22]	—	IPP_DO_E M I _ADDR [22]	A22
—		IPP_DO_CARD_ADDRESS_O[23]	WEIM_ADDR_OUT[23]	—	IPP_DO_E M I _ADDR [23]	A23
—		IPP_DO_CARD_ADDRESS_O[24]	WEIM_ADDR_OUT[24]	—	IPP_DO_E M I _ADDR [24]	A24
—		IPP_DO_CARD_ADDRESS_O[25]	WEIM_ADDR_OUT[25]	—	IPP_DO_E M I _ADDR [25]	A25
ESDCTL address bit M3IF_MA[10] has a dedicated pad MA10 (required due to PRECHARGE ALL during AUTO REFRESH commands). ESDCTL BANK address bits have dedicated PADS due to PRECHARGE BANK during PRECHARGE TIMER time-out WEIM CRE signal is driven on A23 in MUXED MODE OPERATION.						
M3IF_MA[10]		—	—	—	IPP_DO_E M I _MA10	MA10
M3IF_BA[0]		IPP_DO_CARD_CE_B[2]	—	—	IPP_DO_SDBA[1:0]	SDBA0
M3IF_BA[1]		IPP_DO_CARD_CE_B[1]	—	—		SDBA1
Since SDBA PADS are shared between SDR/DDR SDRAM BANK ADDRESS and PCMCIA CE‘, ESDCTL PRECHARGE TIMER cannot be used. During precharge timer, after selected inactivity period of time expires, ESDCTL issue a PRECHARGE command to a specific bank during OFF LINE period. It means that PRECHARGE command can be issued during the time when EMI BUS is not possessed by ESDCTL.						
SDRAM/MDDR Dedicated Data Pads						
M3IF_WR_DATA[0]		—	—	—	IPP_DO_EMII_DATA [0]	SD0
M3IF_RD_DATA[0]					IPP_IND_EMII_DATA_IN [0]	
M3IF_WR_DATA[1]		—	—	—	IPP_DO_EMII_DATA [1]	SD1
M3IF_RD_DATA[1]					IPP_IND_EMII_DATA_IN [1]	
M3IF_WR_DATA[2]		—	—	—	IPP_DO_EMII_DATA [2]	SD2
M3IF_RD_DATA[2]					IPP_IND_EMII_DATA_IN [2]	

Table 15-2. External Memory Interface I/O MUX Description (continued)

MEMORY Controller Outputs					EMI Output	IC Pin Name
SDRAMC	MDDRC	PCMCIA	WEIM	NFC		
M3IF_WR_DATA[3]		—	—	—	IPP_DO_EMII_DATA [3]	SD3
M3IF_RD_DATA[3]					IPP_IND_EMII_DATA _IN [3]	
M3IF_WR_DATA[4]		—	—	—	IPP_DO_EMII_DATA [4]	SD4
M3IF_RD_DATA[4]					IPP_IND_EMII_DATA _IN [4]	
M3IF_WR_DATA[5]		—	—	—	IPP_DO_EMII_DATA [5]	SD5
M3IF_RD_DATA[5]					IPP_IND_EMII_DATA _IN [5]	
M3IF_WR_DATA[6]		—	—	—	IPP_DO_EMII_DATA [6]	SD6
M3IF_RD_DATA[6]					IPP_IND_EMII_DATA _IN [6]	
M3IF_WR_DATA[7]		—	—	—	IPP_DO_EMII_DATA [7]	SD7
M3IF_RD_DATA[7]					IPP_IND_EMII_DATA _IN [7]	
M3IF_WR_DATA[8]		—	—	—	IPP_DO_EMII_DATA [8]	SD8
M3IF_RD_DATA[8]					IPP_IND_EMII_DATA _IN [8]	
M3IF_WR_DATA[9]		—	—	—	IPP_DO_EMII_DATA [9]	SD9
M3IF_RD_DATA[9]					IPP_IND_EMII_DATA _IN [9]	
M3IF_WR_DATA[10]		—	—	—	IPP_DO_EMII_DATA [10]	SD10
M3IF_RD_DATA[10]					IPP_IND_EMII_DATA _IN [10]	
M3IF_WR_DATA[11]		—	—	—	IPP_DO_EMII_DATA [11]	SD11
M3IF_RD_DATA[11]					IPP_IND_EMII_DATA _IN [11]	

Table 15-2. External Memory Interface I/O MUX Description (continued)

MEMORY Controller Outputs					EMI Output	IC Pin Name
SDRAMC	MDDRC	PCMCIA	WEIM	NFC		
M3IF_WR_DATA[12]		—	—	—	IPP_DO_EMII_DATA [12]	SD12
M3IF_RD_DATA[12]					IPP_IND_EMII_DATA _IN [12]	
M3IF_WR_DATA[13]		—	—	—	IPP_DO_EMII_DATA [13]	SD13
M3IF_RD_DATA[13]					IPP_IND_EMII_DATA _IN [13]	
M3IF_WR_DATA[14]		—	—	—	IPP_DO_EMII_DATA [14]	SD14
M3IF_RD_DATA[14]					IPP_IND_EMII_DATA _IN [14]	
M3IF_WR_DATA[15]		—	—	—	IPP_DO_EMII_DATA [15]	SD15
M3IF_RD_DATA[15]					IPP_IND_EMII_DATA _IN [15]	
M3IF_WR_DATA[16]		—	—	—	IPP_DO_EMII_DATA [16]	SD16
M3IF_RD_DATA[16]					IPP_IND_EMII_DATA _IN [16]	
M3IF_WR_DATA[17]		—	—	—	IPP_DO_EMII_DATA [17]	SD17
M3IF_RD_DATA[17]					IPP_IND_EMII_DATA _IN [17]	
M3IF_WR_DATA[18]		—	—	—	IPP_DO_EMII_DATA [18]	SD18
M3IF_RD_DATA[18]					IPP_IND_EMII_DATA _IN [18]	
M3IF_WR_DATA[19]		—	—	—	IPP_DO_EMII_DATA [19]	SD19
M3IF_RD_DATA[19]					IPP_IND_EMII_DATA _IN [19]	
M3IF_WR_DATA[20]		—	—	—	IPP_DO_EMII_DATA [20]	SD20
M3IF_RD_DATA[20]					IPP_IND_EMII_DATA _IN [20]	

Table 15-2. External Memory Interface I/O MUX Description (continued)

MEMORY Controller Outputs					EMI Output	IC Pin Name
SDRAMC	MDDRC	PCMCIA	WEIM	NFC		
M3IF_WR_DATA[21]		—	—	—	IPP_DO_EMII_DATA [21]	SD21
M3IF_RD_DATA[21]					IPP_IND_EMII_DATA _IN [21]	
M3IF_WR_DATA[22]		—	—	—	IPP_DO_EMII_DATA [22]	SD22
M3IF_RD_DATA[22]					IPP_IND_EMII_DATA _IN [22]	
M3IF_WR_DATA[23]		—	—	—	IPP_DO_EMII_DATA [23]	SD23
M3IF_RD_DATA[23]					IPP_IND_EMII_DATA _IN [23]	
M3IF_WR_DATA[24]		—	—	—	IPP_DO_EMII_DATA [24]	SD24
M3IF_RD_DATA[24]					IPP_IND_EMII_DATA _IN [24]	
M3IF_WR_DATA[25]		—	—	—	IPP_DO_EMII_DATA [25]	SD25
M3IF_RD_DATA[25]					IPP_IND_EMII_DATA _IN [25]	
M3IF_WR_DATA[26]		—	—	—	IPP_DO_EMII_DATA [26]	SD26
M3IF_RD_DATA[26]					IPP_IND_EMII_DATA _IN [26]	
M3IF_WR_DATA[27]		—	—	—	IPP_DO_EMII_DATA [27]	SD27
M3IF_RD_DATA[27]					IPP_IND_EMII_DATA _IN [27]	
M3IF_WR_DATA[28]		—	—	—	IPP_DO_EMII_DATA [28]	SD28
M3IF_RD_DATA[28]					IPP_IND_EMII_DATA _IN [28]	
M3IF_WR_DATA[29]		—	—	—	IPP_DO_EMII_DATA [29]	SD29
M3IF_RD_DATA[29]					IPP_IND_EMII_DATA _IN [29]	

Table 15-2. External Memory Interface I/O MUX Description (continued)

MEMORY Controller Outputs					EMI Output	IC Pin Name
SDRAMC	MDDRC	PCMCIA	WEIM	NFC		
M3IF_WR_DATA[30]		—	—	—	IPP_DO_EMII_DATA [30]	SD30
M3IF_RD_DATA[30]					IPP_IND_EMII_DATA _IN [30]	
M3IF_WR_DATA[31]		—	—	—	IPP_DO_EMII_DATA [31]	SD31
M3IF_RD_DATA[31]					IPP_IND_EMII_DATA _IN [31]	
WEIM/NFC/PCMCIA Data Muxing						
—		IPP_DO_CARD_WR _DATA_O[0]	WEIM_DATA_OUT[0]	IPP_NFC_WRITE_D ATA_OUT[0]	IPP_DO_NFC_WRIT E_ DATA_OUT [0]	D0
		IPP_IND_CARD_RD _DATA_I[0]	WEIM_DATA_IN[0]	IPP_NFC_READ_DA TA_IN[0]	IPP_IND_NFC_REA D_ DATA_IN [0]	
—		IPP_DO_CARD_WR _DATA_O[1]	WEIM_DATA_OUT[1]	IPP_NFC_WRITE_D ATA_OUT[1]	IPP_DO_NFC_WRIT E_ DATA_OUT [1]	D1
		IPP_IND_CARD_RD _DATA_I[1]	WEIM_DATA_IN[1]	IPP_NFC_READ_DA TA_IN[1]	IPP_IND_NFC_REA D_ DATA_IN [1]	
—		IPP_DO_CARD_WR _DATA_O[2]	WEIM_DATA_OUT[2]	IPP_NFC_WRITE_D ATA_OUT[2]	IPP_DO_NFC_WRIT E_ DATA_OUT [2]	D2
		IPP_IND_CARD_RD _DATA_I[2]	WEIM_DATA_IN[2]	IPP_NFC_READ_DA TA_IN[2]	IPP_IND_NFC_REA D_ DATA_IN [2]	
—		IPP_DO_CARD_WR _DATA_O[3]	WEIM_DATA_OUT[3]	IPP_NFC_WRITE_D ATA_OUT[3]	IPP_DO_NFC_WRIT E_ DATA_OUT [3]	D3
		IPP_IND_CARD_RD _DATA_I[3]	WEIM_DATA_IN[3]	IPP_NFC_READ_DA TA_IN[3]	IPP_IND_NFC_REA D_ DATA_IN [3]	
—		IPP_DO_CARD_WR _DATA_O[4]	WEIM_DATA_OUT[4]	IPP_NFC_WRITE_D ATA_OUT[4]	IPP_DO_NFC_WRIT E_ DATA_OUT [4]	D4
		IPP_IND_CARD_RD _DATA_I[4]	WEIM_DATA_IN[4]	IPP_NFC_READ_DA TA_IN[4]	IPP_IND_NFC_REA D_ DATA_IN [4]	
—		IPP_DO_CARD_WR _DATA_O[5]	WEIM_DATA_OUT[5]	IPP_NFC_WRITE_D ATA_OUT[5]	IPP_DO_NFC_WRIT E_ DATA_OUT [5]	D5
		IPP_IND_CARD_RD _DATA_I[5]	WEIM_DATA_IN[5]	IPP_NFC_READ_DA TA_IN[5]	IPP_IND_NFC_REA D_ DATA_IN [5]	
—		IPP_DO_CARD_WR _DATA_O[6]	WEIM_DATA_OUT[6]	IPP_NFC_WRITE_D ATA_OUT[6]	IPP_DO_NFC_WRIT E_ DATA_OUT [6]	D6
		IPP_IND_CARD_RD _DATA_I[6]	WEIM_DATA_IN[6]	IPP_NFC_READ_DA TA_IN[6]	IPP_IND_NFC_REA D_ DATA_IN [6]	

Table 15-2. External Memory Interface I/O MUX Description (continued)

MEMORY Controller Outputs					EMI Output	IC Pin Name
SDRAMC	MDDRC	PCMCIA	WEIM	NFC		
—		IPP_DO_CARD_WR_DATA_O[7]	WEIM_DATA_OUT[7]	IPP_NFC_WRITE_DATA_OUT[7]	IPP_DO_NFC_WRITE_ DATA_OUT [7]	D7
		IPP_IND_CARD_RD_DATA_I[7]	WEIM_DATA_IN[7]	IPP_NFC_READ_DATA_IN[7]	IPP_IND_NFC_READ_ DATA_IN [7]	
—		IPP_DO_CARD_WR_DATA_O[8]	WEIM_DATA_OUT[8]	IPP_NFC_WRITE_DATA_OUT[8]	IPP_DO_NFC_WRITE_ DATA_OUT [8]	D8
		IPP_IND_CARD_RD_DATA_I[8]	WEIM_DATA_IN[8]	IPP_NFC_READ_DATA_IN[8]	IPP_IND_NFC_READ_ DATA_IN [8]	
—		IPP_DO_CARD_WR_DATA_O[9]	WEIM_DATA_OUT[9]	IPP_NFC_WRITE_DATA_OUT[9]	IPP_DO_NFC_WRITE_ DATA_OUT [9]	D9
		IPP_IND_CARD_RD_DATA_I[9]	WEIM_DATA_IN[9]	IPP_NFC_READ_DATA_IN[9]	IPP_IND_NFC_READ_ DATA_IN [9]	
—		IPP_DO_CARD_WR_DATA_O[10]	WEIM_DATA_OUT[10]	IPP_NFC_WRITE_DATA_OUT[10]	IPP_DO_NFC_WRITE_ DATA_OUT [10]	D10
		IPP_IND_CARD_RD_DATA_I[10]	WEIM_DATA_IN[10]	IPP_NFC_READ_DATA_IN[10]	IPP_IND_NFC_READ_ DATA_IN [10]	
—		IPP_DO_CARD_WR_DATA_O[11]	WEIM_DATA_OUT[11]	IPP_NFC_WRITE_DATA_OUT[11]	IPP_DO_NFC_WRITE_ DATA_OUT [11]	D11
		IPP_IND_CARD_RD_DATA_I[11]	WEIM_DATA_IN[11]	IPP_NFC_READ_DATA_IN[11]	IPP_IND_NFC_READ_ DATA_IN [11]	
—		IPP_DO_CARD_WR_DATA_O[12]	WEIM_DATA_OUT[12]	IPP_NFC_WRITE_DATA_OUT[12]	IPP_DO_NFC_WRITE_ DATA_OUT [12]	D12
		IPP_IND_CARD_RD_DATA_I[12]	WEIM_DATA_IN[12]	IPP_NFC_READ_DATA_IN[12]	IPP_IND_NFC_READ_ DATA_IN [12]	
—		IPP_DO_CARD_WR_DATA_O[13]	WEIM_DATA_OUT[13]	IPP_NFC_WRITE_DATA_OUT[13]	IPP_DO_NFC_WRITE_ DATA_OUT [13]	D13
		IPP_IND_CARD_RD_DATA_I[13]	WEIM_DATA_IN[13]	IPP_NFC_READ_DATA_IN[13]	IPP_IND_NFC_READ_ DATA_IN [13]	
—		IPP_DO_CARD_WR_DATA_O[14]	WEIM_DATA_OUT[14]	IPP_NFC_WRITE_DATA_OUT[14]	IPP_DO_NFC_WRITE_ DATA_OUT [14]	D14
		IPP_IND_CARD_RD_DATA_I[14]	WEIM_DATA_IN[14]	IPP_NFC_READ_DATA_IN[14]	IPP_IND_NFC_READ_ DATA_IN [14]	
—		IPP_DO_CARD_WR_DATA_O[15]	WEIM_DATA_OUT[15]	IPP_NFC_WRITE_DATA_OUT[15]	IPP_DO_NFC_WRITE_ DATA_OUT [15]	D15
		IPP_IND_CARD_RD_DATA_I[15]	WEIM_DATA_IN[15]	IPP_NFC_READ_DATA_IN[15]	IPP_IND_NFC_READ_ DATA_IN [15]	
MASK (Byte Enable) Muxing						

Table 15-2. External Memory Interface I/O MUX Description (continued)

MEMORY Controller Outputs					EMI Output	IC Pin Name
SDRAMC	MDDRC	PCMCIA	WEIM	NFC		
—	—	IPP_DO_CARD_REG_B_O	WEIM_EB_B[0]	—	IPP_DO_E M I _IO_EB_B[1:0]	EB0
—	—	IPP_DO_CARD_IORD_B	WEIM_EB_B[1]	—		EB1
EB_B[2] and EB_B[3] WILL BE DRIVEN ON A24 and A25 RESPECTIVELLY DURING MUXED MODE (IN ORDER TO USE 32 BIT MEMORY DEVICE.						
SDRAM/MDDR Mask (Byte Enable)						
M3IF_DQM[0]		—	—	—	IPP_DO_DQM [3:0]	DQM0
M3IF_DQM[1]		—	—	—		DQM1
M3IF_DQM[2]		—	—	—		DQM2
M3IF_DQM[3]		—	—	—		DQM3
Output Enable Muxing						
—		IPP_DO_CARD_IOWR_B	WEIM_WR_OE	—	IPP_DO_EMI _OE_B	OE
Chip Select Muxing						
—	—	—	IPP_DO_WEIM_CS_B[0]	—	IPP_DO_WEIM_CS_B0	CS0
—	—	—	IPP_DO_WEIM_CS_B[1]	—	IPP_DO_WEIM_CS_B1	CS1
M3IF_CS_B[0]		—	IPP_DO_WEIM_CS_B[2]	—	IPP_DO_WEIM_CS_B2_CSD0	CS2
M3IF_CS_B[1]		—	IPP_DO_WEIM_CS_B[3]	—	IPP_DO_WEIM_CS_B3_CSD1	CS3
—	—	—	IPP_DO_WEIM_CS_B[4]	—	IPP_DO_WEIM_CS_B4	CS4
—	—	—	IPP_DO_WEIM_CS_B[5]	—	IPP_DO_WEIM_CS_B5	CS5
CHIP SELECT are SYSTEM CONTROL REGISTER bits SDCTL_CSD0_SEL and SDCTL_CSD1_SEL respectively. Default select for both CHIP SELECTS are for the ESDCTL/MDDRC.						
WRITE Enable Muxing						
—	—	IPP_DO_CARD_WE_B	WEIM_RW_B	—	IPP_DO_WEIM_RW_B	RW
SDRAM/MDDR Command Dedicated Pads						
RAS_B		—	—	—	IPP_DO_M3IF_RAS_B	RAS

Table 15-2. External Memory Interface I/O MUX Description (continued)

MEMORY Controller Outputs					EMI Output	IC Pin Name
SDRAMC	MDDRC	PCMCIA	WEIM	NFC		
CAS_B		—	—	—	IPP_DO_M3IF_CAS_B	CAS
WE_B		—	—	—	IPP_DO_SDRC_SDWE	SDWE
CKE[1]		—	—	—	IPP_DO_SDRC_SDCKE[1]	SDCKE [1]
CKE[0]		—	—	—	IPP_DO_SDRC_SDCKE[0]	SDCKE [0]
SDCLK_OUT		—	—	—	IPP_DO_SDRC_SDCLK	SDCLK
—	MDDR_SDCLK_B	—	—	—	IPP_DO_MDDR_SDCLK_B	SDCLK_B
—	DQS_OUT [3]	—	—	—	IPP_DO_DQS[3]	DQS[3]
	DQS_IN [3]				IPP_DIN_DQS[3]	
—	DQS_OUT [2]	—	—	—	IPP_DO_DQS[2]	DQS[2]
—	DQS_IN [2]	—	—	—	IPP_DIN_DQS[2]	
—	DQS_OUT [1]	—	—	—	IPP_DO_DQS[1]	DQS[1]
—	DQS_IN [1]	—	—	—	IPP_DIN_DQS[1]	
—	DQS_OUT [0]	—	—	—	IPP_DO_DQS[0]	DQS[0]
—	DQS_IN [0]	—	—	—	IPP_DIN_DQS[0]	
—	—	—	—	—	M_REQUEST	M_REQUEST
—	—	—	—	—	M_GRANT	M_GRANT
NFC Command Dedicated Pads						
—	—	—	—	IPP_NFC_WE_OUT	IPP_NFC_WE_OUT	NFWE_B
—	—	—	—	IPP_NFC_WP_OUT	IPP_NFC_WP_OUT	NFWP_B

Table 15-2. External Memory Interface I/O MUX Description (continued)

MEMORY Controller Outputs					EMI Output	IC Pin Name
SDRAMC	MDDRC	PCMCIA	WEIM	NFC		
—	—	—	—	IPP_NFC_RE_OUT	IPP_NFC_RE_OUT	NFRE_B
—	—	—	—	IPP_NFC_ALE_OUT	IPP_NFC_ALE_OUT	NFALE
—	—	—	—	IPP_NFC_CLE_OUT	IPP_NFC_CLE_OUT	NFCLE
—	—	—	—	IPP_NFC_CE_OUT	IPP_NFC_CE_OUT	NFCE_B
—	—	—	—	IPP_NFC_RB_IN	IPP_NFC_RB_IN	NFRB
WEIM Command Dedicated Pads						
—	—	IPP_DO_CARD_OE_B	IPP_DO_WEIM_LBA_B	—	IPP_DO_WEIM_LBA_B	LBA_B
—	—	—	IPP_DO_WEIM_BCLK	—	IPP_DO_WEIM_BCLK	BCLK
—	—	—	IPP_IND_WEIM_ECB_B	—	IPP_IND_WEIM_ECB_B	ECB
PCMCIA Command Dedicated Pads						
—	—	IPP_INT_CD1_B	—	—	IPP_INT_CD1_B	PC_CD1_B
—	—	IPP_INT_CD2_B	—	—	IPP_INT_CD2_B	PC_CD2_B
—	—	IPP_IND_WAIT_B_I	—	—	IPP_IND_WAIT_B_I	PC_WAIT_B
—	—	IPP_IND_PWR_ON_I	—	—	IPP_IND_PWR_ON_I	PC_PWRON
—	—	IPP_IND_RDY_IRQ_B_I	—	—	IPP_IND_RDY_IRQ_B_I	PC_READY
—	—	IPP_IND_VS1_B	—	—	IPP_IND_VS1_B	PC_VS1
—	—	IPP_IND_VS2_B	—	—	IPP_IND_VS2_B	PC_VS2
—	—	IPP_IND_BVD1_STSCH_B_I	—	—	IPP_IND_BVD1_STSCH_B_I	PC_BVD1
—	—	IPP_IND_BVD2_SPKR_I	—	—	IPP_IND_BVD2_SPKR_I	PC_BVD2
—	—	IPP_DO_CARD_RESET	—	—	IPP_DO_CARD_RESET	PC_RST
—	—	IPP_DO_CARD_POE_O_B	—	—	IPP_DO_CARD_POE_O_B	PC_POE

Table 15-2. External Memory Interface I/O MUX Description (continued)

MEMORY Controller Outputs					EMI Output	IC Pin Name
SDRAMC	MDDRC	PCMCIA	WEIM	NFC		
—	—	IPP_DO_CARD_RW_B	—	—	IPP_DO_CARD_RW_B	PC_RW_B
—	—	IPP_IND_WP_I	—	—	IPP_IND_WP_I	IOIS16

15.7.2 EMI Input/Output Signals

This section lists all input and output signals for the entire EMI module. Table 15-3 summarizes interface signals. For detailed descriptions of each signal function, refer to the relevant module chapters in this manual.

Table 15-3. EMI Signal Properties

Name	Port	Function	Reset State
AHB Interface Outputs			
M3IF_HREADY_M0	O	AHB access completion strobe to master #0 — LCDC	1
M3IF_HREADY_M1	O	AHB access completion strobe to master #1 — FEC	1
M3IF_HREADY_M2	O	AHB access completion strobe to master #2 eMMA	1
M3IF_HREADY_M3	O	AHB access completion strobe to master #3 MAX	1
M3IF_HREADY_M4	O	AHB access completion strobe to master #4 — H264	1
M3IF_HREADY_M5	O	AHB access completion strobe to master #5 — H264	1
M3IF_HREADY_M6	O	AHB access completion strobe to master #6 — H264	1
M3IF_HREADY_M7	O	AHB access completion strobe to master #7 — USBOTG	1
M3IF_HRESP_M0	O	AHB error response to master #0 — LCDC	0
M3IF_HRESP_M1	O	AHB error response to master #1 — FEC	0
M3IF_HRESP_M2	O	AHB error response to master #2 eMMA	0
M3IF_HRESP_M3	O	AHB error response to master #3 MAX	0
M3IF_HRESP_M4	O	AHB error response to master #4 — H264	0
M3IF_HRESP_M5	O	AHB error response to master #5 — H264	0
M3IF_HRESP_M6	O	AHB error response to master #6 — H264	0
M3IF_HRESP_M7	O	AHB error response to master #7 — USBOTG	0
M3IF_HRDATA_M0[63:0]	O	AHB read data bus to master #0 — LCDC	0
M3IF_HRDATA_M1[63:0]	O	AHB read data bus to master #1 — FEC	0
M3IF_HRDATA_M2[63:0]	O	AHB read data bus to master #2 eMMA	0
M3IF_HRDATA_M3[31:0]	O	AHB read data bus to master #3 MAX	0
M3IF_HRDATA_M4[63:0]	O	AHB read data bus to master #4 — H264	0

Table 15-3. EMI Signal Properties (continued)

Name	Port	Function	Reset State
M3IF_HRDATA_M5[63:0]	O	AHB read data bus to master #5 — H264	0
M3IF_HRDATA_M6[63:0]	O	AHB read data bus to master #6 — H264	0
M3IF_HRDATA_M7[31:0]	O	AHB read data bus to master #7 — USBOTG	0
M3IF and ESDCTL/MDDRC Outputs			
IPP_DO_SDRC_SDCKE[1:0]	O	SDRAM/MDDR clock enable	0
M3IF_DQM[3:0]	O	SDRAM data mask strobes. DQM0 corresponds to DQ0–DQ7, DQM1 corresponds to DQ8–DQ15, DQM2 corresponds to DQ16–DQ23 and DQM3 corresponds to DQ24–DQ31.	0
DQS_OUT[3:0]	O	MDDR data sample strobes for write accesses. DQS0 corresponds to DQ0–DQ7, DQS1 corresponds to DQ8–DQ15, DQS2 corresponds to DQ16–DQ23 and DQS3 corresponds to DQ24–DQ31.	0
DQS_OUT_EN_X	O	DQS output enable strobe	0
M3IF_MA[13:0]	O	SDRAM/MDDR address bits	0
M3IF_BA[1:0]	O	SDRAM/MDDR bank address bits	0
IPP_DO_M3IF_MA10	O	SDRAM/MDDR address bit A10	0
M3IF_CS_B[1:0]	O	SDRAM/MDDR chip select strobe	3
CAS_B	O	SDRAM/MDDR CAS strobe	1
RAS_B	O	SDRAM/MDDR RAS strobe	1
WE_B	O	SDRAM/MDDR WE strobe	1
M3IF_CHOOSEN_MASTER[2:0]	O	M3IF arbitration chosen master (for debug)	3
IPP_DO_SDRC_SDCLK	O	SDRAM/MDDR clock (up to 133MHz)	0
IPP_DO_MDDR_SD_CLK_B	O	MDDR clock (up to 133MHz)	0
LPACK	O	Low power mode acknowledge — toward CCM	1
SDRC_SF_WACK	O	Memory wakeup acknowledge indication to WDOG	0
NFC Outputs			
IPP_DO_NFC_WRITE_DATA_OUT[15:0]	O	NFC write data out toward I/O MUX/PADS.	0
IPI_INT_NFC_B	0	NFC interrupt (indicating an access completion)	1
IPP_NFC_ALE_OUT	O	NFC out NF_ALE	0
IPP_NFC_CE_OUT	O	NFC out NF_CE	0
IPP_NFC_CLE_OUT	O	NFC out NF_CLE	0
IPP_NFC_RE_OUT	O	NFC out NF_RE	0
IPP_NFC_WE_OUT	O	NFC out NF_WE	0
IPP_NFC_WP_OUT	O	NFC out NF_WP	0

Table 15-3. EMI Signal Properties (continued)

Name	Port	Function	Reset State
WEIM Outputs			
IPP_DO_WEIM_CS_B0	O	WEIM CS0 chip select toward I/O MUX/PADS	1
IPP_DO_WEIM_CS_B1	O	WEIM CS1 chip select toward I/O MUX/PADS	1
IPP_DO_WEIM_CS_B2_CSD0	O	WEIM CS2 or ESDCTL/MDDRC CSD0 chip select toward I/O MUX/PADS	1
IPP_DO_WEIM_CS_B3_CSD1	O	WEIM CS2 or ESDCTL/MDDRC CSD1 chip select toward I/O MUX/PADS	1
IPP_DO_WEIM_CS_B4	O	WEIM CS4 chip select toward I/O MUX/PADS	1
IPP_DO_WEIM_CS_B5	O	WEIM CS5 chip select toward I/O MUX/PADS	1
IPP_DO_WEIM_BCLK	O	WEIM Burst Clock	0
IPP_DO_LBA_B	O	WEIM Load Burst Address (LBA)	1
IPP_DO_WEIM_RW_B	O	WEIM read/write strobe	1
PCMCIA Outputs			
IPI_INT_BVD1_B	O	BVD1 changed interrupt	1
IPI_INT_BVD2_B	O	BVD2 changed interrupt	1
IPI_INT_CD1_B	O	CD1 changed interrupt	1
IPI_INT_CD2_B	O	CD2 changed interrupt	1
IPI_INT_ERR_B	O	Access error interrupt	1
IPI_INT_IRQ_B	O	or of all the ready interrupts	1
IPI_INT_NFC_B	O	NFC interrupt (indicating an action completed)	1
IPI_INT_PCMCIA_B	O	or of all the interrupts (status+access+ready)	1
IPI_INT_POWERON_B	O	POWER_ON changed interrupt	1
IPI_INT_RDY_F_B	O	RDY negedge interrupt	1
IPI_INT_RDY_H_B	O	RDY is high interrupt	1
IPI_INT_RDY_L_B	O	RDY is low level sensitive interrupt	1
IPI_INT_RDY_R_B	O	RDY posedge sensitive interrupt	1
IPI_INT_STS_B	O	or of all the status interrupts	1
IPI_INT_VS1_B	O	VS1 changed interrupt	1
IPI_INT_VS2_B	O	VS2 changed interrupt	1
IPI_INT_WP_B	O	WP changed interrupt	1
IPP_DO_CARD_POE_O	O	POE signal to transceivers	0
IPP_DO_CARD_RESET_O	O	RESET signal to card	0
IPP_DO_CARD_RW_B	O	RW_B Signal to data transceiver	1

Table 15-3. EMI Signal Properties (continued)

Name	Port	Function	Reset State
IPP_DO_SPKR_OUT_O	O	speaker out	0
Global Outputs			
M3IF_DMA_ACCESS	O	Snooping detection indication toward IPU module	0
IPP_OBE_DDR_EN	O	MDDR active indication to ESDCTL/MDDRC DATA PADS	0
IPP_DO_EMI_ADDR[25:0]	O	EMI address out toward I/O MUX/PADS	0
IPP_DO_NFC_WRITE_DATA_OUT[15:0]	O	EMI data (NFC, WEIM) out toward I/O MUX/PADS	0
IPP_DO_EMI_DATA[31:0]	O	EMI SDRAM/DDR data out toward I/O MUX/PADS	0
IPP_OBE_EMI_DATA_DIR	O	EMI SDRAM/DDR data direction toward I/O MUX/PADS	0
IPP_OBE_NFC_DIR_HIGH	O	EMI (NFC, WEIM, PCMCIA) data direction toward I/O MUX/PADS	0
IPP_OBE_NFC_DIR_LOW	O	EMI (NFC, WEIM, PCMCIA) data direction toward I/O MUX/PADS	0
IPP_DO_SDBA[1:0]	O	SDRAM/MDDR bank address toward I/O MUX/PADS	0
IPP_DO_M3IF_MA10	O	SDRAM/MDDR address bit MA10 toward I/O MUX/PADS	0
IPP_DO_EMI_IO_DQM[3:0]	O	SDRAM/MDDR enable bytes toward I/O MUX/PADS	0
IPP_DO_EMI_OE_B	O	EMI output enable toward I/O MUX/PADS	0
IPP_OBE_IO_ADDR_DIR[1:0]	O	EMI output enable (dir) toward I/O ADDR/WEIM MUXED DATA MUX/PADS	0
AHB Interface Inputs			
M3IF_HADDR_M0[31:0]	I	AHB address bus from master #0 — LCDC	0
M3IF_HADDR_M1[31:0]	I	AHB address bus from master #1 — FEC	0
M3IF_HADDR_M2[31:0]	I	AHB address bus from master #2 eMMA	0
M3IF_HADDR_M3[31:0]	I	AHB address bus from master #3 MAX	0
M3IF_HADDR_M4[31:0]	I	AHB address bus from master #4 — H264	0
M3IF_HADDR_M5[31:0]	I	AHB address bus from master #5 — H264	0
M3IF_HADDR_M6[31:0]	I	AHB address bus from master #6 — H264	0
M3IF_HADDR_M7[31:0]	I	AHB address bus from master #7 — USBOTG	0
M3IF_HWDATA_M0[63:0]	I	AHB write data bus (bit) from master #0 — LCDC	0
M3IF_HWDATA_M1[63:0]	I	AHB write data bus (bit) from master #1 — FEC	0
M3IF_HWDATA_M2[63:0]	I	AHB write data bus (32 bit) from master #2 eMMA	0
M3IF_HWDATA_M3[31:0]	I	AHB write data bus (32 bit) from master #3 MAX	0
M3IF_HWDATA_M4[63:0]	I	AHB write data bus (32 bit) from master #4 — H264	0
M3IF_HWDATA_M5[63:0]	I	AHB write data bus (32 bit) from master #5 — H264	0
M3IF_HWDATA_M6[63:0]	I	AHB write data bus (32 bit) from master #6 — H264	0

Table 15-3. EMI Signal Properties (continued)

Name	Port	Function	Reset State
M3IF_HWDATA_M7[31:0]	I	AHB write data bus (32 bit) from master #7 — USBOTG	0
M3IF_HBURST_M0[2:0]	I	AHB burst size bus from master #0 — LCDC	0
M3IF_HBURST_M1[2:0]	I	AHB burst size bus from master #1 — FEC	0
M3IF_HBURST_M2[2:0]	I	AHB burst size bus from master #2 eMMA	0
M3IF_HBURST_M3[2:0]	I	AHB burst size bus from master #3 MAX	0
M3IF_HBURST_M4[2:0]	I	AHB burst size bus from master #4 —H264	0
M3IF_HBURST_M5[2:0]	I	AHB burst size bus from master #5 — H264	0
M3IF_HBURST_M6[2:0]	I	AHB burst size bus from master #6 — H264	0
M3IF_HBURST_M7[2:0]	I	AHB burst size bus from master #7 — USBOTG	0
M3IF_HSIZE_M0[1:0]	I	AHB data transfer width bus from master #0 — LCDC	0
M3IF_HSIZE_M1[1:0]	I	AHB data transfer width bus from master #1 — FEC	0
M3IF_HSIZE_M2[1:0]	I	AHB data transfer width bus from master #2 eMMA	0
M3IF_HSIZE_M3[1:0]	I	AHB data transfer width bus from master #3 MAX	0
M3IF_HSIZE_M4[1:0]	I	AHB data transfer width bus from master #4 — H264	0
M3IF_HSIZE_M5[1:0]	I	AHB data transfer width bus from master #5 — H264	0
M3IF_HSIZE_M6[1:0]	I	AHB data transfer width bus from master #6 —H264	0
M3IF_HSIZE_M7[1:0]	I	AHB data transfer width bus from master #7 —USBOTG	0
M3IF_HBSTRB_M0[7:0]	I	byte lane (8) bus from master #0 — LCDC	0
M3IF_HBSTRB_M1[7:0]	I	byte lane (8) bus from master #1 — FEC	0
M3IF_HBSTRB_M2[7:0]	I	byte lane (4) bus from master #2 eMMA	0
M3IF_HBSTRB_M3[3:0]	I	byte lane (4) bus from master #3 MAX	0
M3IF_HBSTRB_M4[7:0]	I	byte lane (4) bus from master #4 — H264	0
M3IF_HBSTRB_M5[7:0]	I	byte lane (4) bus from master #5 — H264	0
M3IF_HBSTRB_M6[7:0]	I	byte lane (4) bus from master #6 — H264	0
M3IF_HBSTRB_M7[3:0]	I	byte lane (4) bus from master #7 — USBOTG	0
M3IF_HTRANS_M0[1:0]	I	AHB transfer state bus from master #0 — LCDC	0
M3IF_HTRANS_M1[1:0]	I	AHB transfer state bus from master #1 — FEC	0
M3IF_HTRANS_M2[1:0]	I	AHB transfer state bus from master #2 eMMA	0
M3IF_HTRANS_M3[1:0]	I	AHB transfer state bus from master #3 MAX	0
M3IF_HTRANS_M4[1:0]	I	AHB transfer state bus from master #4 — H264	0
M3IF_HTRANS_M5[1:0]	I	AHB transfer state bus from master #5 — H264	0
M3IF_HTRANS_M6[1:0]	I	AHB transfer state bus from master #6 — H264	0
M3IF_HTRANS_M7[1:0]	I	AHB transfer state bus from master #7 — USBOTG	0

Table 15-3. EMI Signal Properties (continued)

Name	Port	Function	Reset State
M3IF_HWRITE_M0	I	AHB read/write signal from master #0 — LCDC	0
M3IF_HWRITE_M1	I	AHB read/write signal from master #1 — FEC	0
M3IF_HWRITE_M2	I	AHB read/write signal from master #2 eMMA	0
M3IF_HWRITE_M3	I	AHB read/write signal from master #3 MAX	0
M3IF_HWRITE_M4	I	AHB read/write signal from master #4 — H264	0
M3IF_HWRITE_M5	I	AHB read/write signal from master #5 — H264	0
M3IF_HWRITE_M6	I	AHB read/write signal from master #6 — H264	0
M3IF_HWRITE_M7	I	AHB read/write signal from master #7 — USBOTG	0
M3IF_HPROT_M0	I	AHB protection mode signal from master #0 — LCDC	0
M3IF_HPROT_M1	I	AHB protection mode signal from master #1 — FEC	0
M3IF_HPROT_M2	I	AHB protection mode signal from master #2 eMMA	0
M3IF_HPROT_M3	I	AHB protection mode signal from master #3 MAX	0
M3IF_HPROT_M4	I	AHB protection mode signal from master #4 — H264	0
M3IF_HPROT_M5	I	AHB protection mode signal from master #5 — H264	0
M3IF_HPROT_M6	I	AHB protection mode signal from master #6 —H264	0
M3IF_HPROT_M7	I	AHB protection mode signal from master #7 —USBOTG	0
M3IF_HUNALIGN_M0	I	Unalign access signal from master #0 — LCDC	0
M3IF_HUNALIGN_M1	I	Unalign access signal from master #1 — FEC	0
M3IF_HUNALIGN_M2	I	Unalign access signal from master #2 eMMA	0
M3IF_HUNALIGN_M3	I	Unalign access signal from master #3 MAX	0
M3IF_HUNALIGN_M4	I	Unalign access signal from master #4 — H264	0
M3IF_HUNALIGN_M5	I	Unalign access signal from master #5 — H264	0
M3IF_HUNALIGN_M6	I	Unalign access signal from master #6 — H264	0
M3IF_HUNALIGN_M7	I	Unalign access signal from master #7 — USBOTG	0
M3IF and ESDCTL/MDDRC Inputs			
M3IF_HCLK	I	M3IF AHB system clock—up to 133 MHz	0
HCLK32	I	32 KHz clock for ESDCTL refresh counter	0
IPP_IND_SDRC_SDCLK_FB	I	SDRAM/MDDR feedback clock (up to 133MHz)	0
LPMD	I	Low power mode indication signal, “0”=STOP, “1”=RUN.	1
DQS_IN[3:0]	I	MDDR data sample strobes for read accesses. DQS0 corresponds to DQ0–DQ7, DQS1 corresponds to DQ8–DQ15, DQS2 corresponds to DQ16–DQ23 and DQS3 corresponds to DQ24–DQ31.	0

Table 15-3. EMI Signal Properties (continued)

Name	Port	Function	Reset State
SDCTL_CSD0_SEL_B	I	SDRAM/MDDR CSD0 select multiplexed with CS2 (configurable via the system control register, FMCR)	0
SDCTL_CSD1_SEL_B	I	SDRAM/MDDR CSD1 select multiplexed with CS3 (configurable via the system control register, FMCR)	0
NFC Inputs			
NF16_BOOT_B	I	Boot mode source is 16 bit NAND Flash memory.	application dependent
NF8_BOOT_B	I	Boot mode source is 8 bit NAND Flash memory.	application dependent
NF_16BIT_SEL	I	16 bit NAND Flash memory is use indication.	0
NFC_HCLK	I	NFC AHB input clock	0
NFC_RD_OE	I	NFC read output enable — controls the direction of data bus	0
NFC_WR_OE	I	NFC write output enable — controls the direction of data bus	0
IPP_IND_FLASH_CLK	I	NAND Flash side clock with period of 40nS	0
IPP_IND_NFC_RB_IN	I	NFC in NF_RB	0
WEIM Inputs			
WEIM_BOOT_CFG[3:0]	I	WEIM BootMode select (from CCM) “101” - 16 bit CS0 at D[15:0]	application dependent
IPP_IND_WEIM_ECB_B	I	WEIM End Current Burst	1
WEIM_HCLK	I	WEIM AHB input clock	0
IPP_IND_WEIM_DTACK_B	I	External DTACK acknowledge	1
PCMCIA Inputs			
IPP_IND_CD_B_I[1:0]	I	CD[1:0] signals from card	3
IPP_IND_VS_I[1:0]	I	VS[1:0] signals from card	0
IPP_IND_BVD1_STSCH_B_I	I	BVD1/STSCHG signals from card	0
IPP_IND_BVD2_SPKR_I	I	BVD2/SPKR signals from card	0
IPP_IND_PWR_ON_I	I	POWER_ON signals from card/board	0
IPP_IND_RDY_IRQ_B_I	I	READY/IRQ_B signals from card	1
IPP_IND_WP_I	I	WP signals from card	1
Global Inputs			
IPP_IND_RESETB	I	Reset signal	1
HRESET_HCLK_B	I	Reset signal	1
IPP_IND_NFC_READ_DATA_IN[15:0]	I	External memories (non SDRAM) read data in from I/O MUX/PADS	0
IPP_IND_EMI_DATA_IN[31:0]	I	EMI SDRAM/DDR data in from I/O MUX/PADS	0

Table 15-3. EMI Signal Properties (continued)

Name	Port	Function	Reset State
IPP_IND_ADDR_IN[15:0]	I	EMI WEIM muxed data in from I/O MUX/PADS. WEIM/NFC/PCMCIA address out to I/O MUX/PADS.	0
IPP_IND_EMI_DATA_IN[31:0]	I	SDRAM/MDDR read data in from I/O MUX/PADS	0
M3IF_BIGEND_M0	I	Endian mode signal from master #0 — LCDC	master dependent
M3IF_BIGEND_M1	I	Endian mode signal from master #1 — FEC	master dependent
M3IF_BIGEND_M2	I	Endian mode signal from master #2 eMMA	master dependent
M3IF_BIGEND_M3	I	Endian mode signal from master #3 MAX	master dependent
M3IF_BIGEND_M4	I	Endian mode signal from master #4 — H264	master dependent
M3IF_BIGEND_M5	I	Endian mode signal from master #5 — H264	master dependent
M3IF_BIGEND_M6	I	Endian mode signal from master #6 — H264	master dependent
M3IF_BIGEND_M7	I	Endian mode signal from master #7 — USBOTG	master dependent

15.8 Memory Map and Register Definitions

The EMI supports four memory controllers. Table 15-4 and Table 15-5 illustrates the EMI registers and memory map (mapped by all memory controllers). For detailed descriptions regarding registers definitions and memory map, refer to the relevant module chapters in this manual.

Table 15-4. EMI Registers Definition

Address	Use	Access
M3IF Registers Space		
0xD800_3000–0xD800_3FFF	M3IF registers space (4K)	Read/Write
ESDCTL/MDDRC Registers Space		
0xD800_1000–0xD800_1FFF	ESDCTL/MDDRC registers space (4K)	Read/Write
PCMCIA Registers Space		
0xD800_4000–0xD800_4FFF	PCMCIA registers space (4K)	Read/Write
WEIM Registers Space		
0xD800_2000–0xD800_2FFF	WEIM registers space (4K)	Read/Write
NFC Memory Space		
0xD800_0E00–0xD800_0FFF	NFC registers space	Read/Write

Table 15-5. EMI Memory Map

Address	Use	Access
ESDCTL/MDDRC Memory Space		
0xA000_0000–0xAFFF_FFFF	CSD0 SDRAM/MDDR memory region (256 Mbytes)	Read/Write
0xB000_0000–0xBFFF_FFFF	CSD1 SDRAM/MDDR memory region (256 Mbytes)	Read/Write
PCMCIA Memory Space		
0xDC00_0000–0xDCFF_FFFF	PCMCIA/CF memory region (64 Mbytes)	Read/Write
WEIM Memory Space		
0xC000_0000–0xC7FF_FFFF	WEIM CS0 memory region[1] (128 Mbytes)	Read/Write
0xC800_0000–0xCFFF_FFFF	WEIM CS1 memory region (128 Mbytes)	Read/Write
0xD000_0000–0xD1FF_FFFF	WEIM CS2 memory region (32 Mbytes)	Read/Write
0xD200_0000–0xD3FF_FFFF	WEIM CS3 memory region (32 Mbytes)	Read/Write
0xD400_0000–0xD5FF_FFFF	WEIM CS4 memory region (32 Mbytes)	Read/Write
0xD600_0000–0xD7FF_FFFF	WEIM CS5 memory region (32 Mbytes)	Read/Write
NFC Memory Space		
0xD800_0000–0xD800_0FFF	NFC memory region[1] (4K, NAND Flash)	Read/Write

1. Can be used as a boot memory region.

Chapter 16
Multi-Master Memory Interface (M3IF)

The M3IF controls memory accesses (read/write/erase/program) from one or more masters through different port interfaces to different external memory controllers ESDCTL/MDDRC, PCMCIA, NAND Flash, and WEIM. Figure 16-1 provides top-level diagram that shows the functional organization of the block.

16.1 Overview

The M3IF-ESDCTL/MDDRC interface is optimized and designed to reduce access latency by generating multiple accesses through the dedicated ESDCTL/MDDRC arbitration (MAB) module, which controls the access to/from the Enhanced SDRAM/MDDR memory controller. For the other port interfaces, the M3IF only arbitrates and forwards the master requests received through the Master Port Gasket (MPG) interface and M3IF Arbitration (M3A) module toward the respective memory controller. The masters that interface with the M3IF include the ARM Platform, SDMA, MPEG-4 encoder, and the IPU. The controllers are the ESDCTL/MDDRC, PCMCIA, NAND Flash, and WEIM.

When a master requests a memory access, the access is immediately taken by the M3IF if no other access is in progress. The M3IF forwards the access to the respective memory controller (slave), and depending on the state of the respective memory controller, a command to the memory is generated. If the access cannot be started due to a previous active access, the master request remains pending (HREADY held negated) until it is executed by the memory controller. When the access execution is complete, the HREADY is asserted and a new request can be processed.

Accesses to SDRAM or MDDR external devices are optimized through command anticipation (MIF2 strategy). For example, the next access control phase (memory address and command) is driven during the previous access data phase (data flow to/from the memory), thus an overlap between accesses is created and latency is partially or fully hidden.

16.1.1 M3IF Interfaces

The interface between M3IF and the controllers can be divided into two different types: M3IF-ESDCTL, and M3IF-all others. The M3IF-ESDCTL/MDDRC interface reduces access latency by generating multiple accesses using the dedicated ESDCTL/MDDRC arbitration (MAB) module. For other port interfaces, M3IF arbitrates and forwards the masters' requests received through the Master Port Gasket (MPG) interfaces and the M3IF arbitration (M3A) module toward the respective memory controller. To support multiple accesses to the ESDCTL/MDDRC, the MAB includes a FIFO which controls the access traffic from/to the ESDCTL/MDDRC.

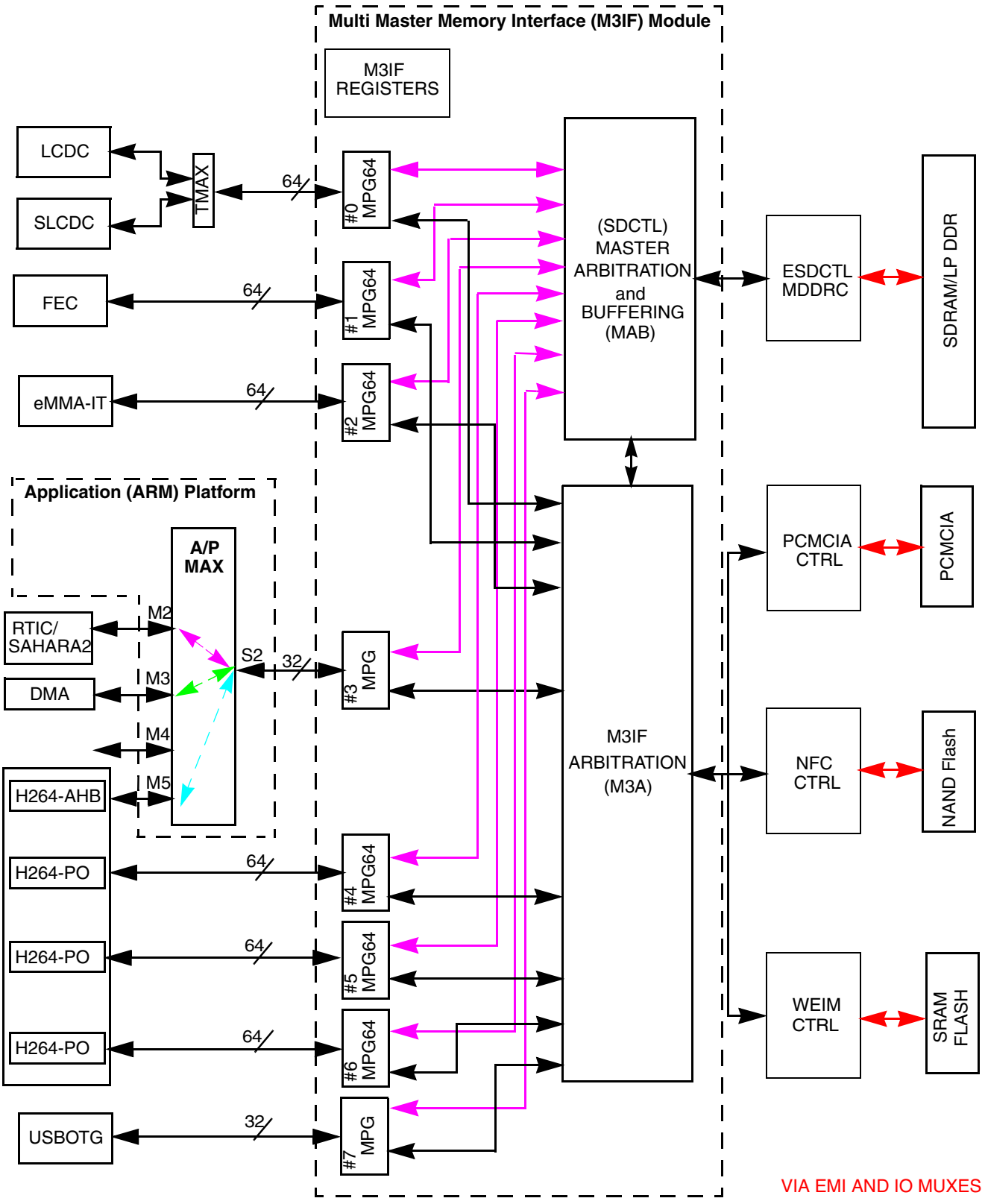


Figure 16-1. M3IF Block Diagram—System Overview

M3IF can be viewed as a device that has multiple SDRAM/MDDR controllers and one controller per other memory type, therefore, the M3A arbitrates the requests as follows:

- A round robin chooses the next master that is going to grant the bus:
  - — If this master is requesting access to non-SDRAM/MDDR memory controller, M3A waits until the previous access finishes and only then passes the request.
  - — If the master is requesting access to ESDCTL/MDDRC, M3IF arbitration passes the access to MAB in two cases:
    - – After previous non-ESDCTL access is accomplished.
    - – If previous access was to ESDCTL, M3IF arbitration will pass the request immediately without waiting for the previous access to accomplished.

M3IF Arbitration (M3A) and the ESDCTL/MDDRC Master Arbitration and Buffering (MAB) supports a round-robin arbitration scheme (which can be programed to non-equal probability). If two masters request access to the memory port on the same cycle the master with the token (see Section 16.4.3.2, “M3A–Find First 1 (FF1) Algorithm” for more details) gains control on the bus to the slave. To support multiple accesses to the ESDCTL/MDDRC, the MAB includes a FIFO which controls the access traffic from/to the ESDCTL/MDDRC. Once a master grants the bus, the Memory Controller gaining the access converts the access to a command to the specified memory.

16.1.2 Features

M3IF Master Port Gasket (MPG) converts the master request (data write, data read, address, and controls) to a set of bus/signals that the M3IF arbitration, the SDCTL/MDDRC arbitration, and other memory controllers need. The MPG is also responsible to give the right response to the master after getting the response from the relevant memory controller. M3IF support 2 port interfaces (the number and types of gasket ports used depends on the system requirements):

MPG—Master Port Gasket for ARM9 AMBA-AHB lite with 32 bit data bus.MPG64—Master Port Gasket for AMBA-AHB lite with 64 bit data bus.The M3IF includes these distinctive features:

- Supports multiple requests from masters through 2 different input port interfaces:
  - — Master Port Gasket (MPG)—ARM9 AMBA AHB lite bus protocol.
- Master Port Gasket (MPG64)—AMBA AHB access with 64 bits data bus width.Arbitrates requests to four different memory controllers (that share some of their I/O pads)
  - — Enhanced SDRAM Controller (ESDCTL) or MDDR Controller (MDDRC)
  - — NAND Flash Controller—(NFC)
  - — PCMCIA Controller—Wireless External Interface Memory (WEIM) Controller
- Multiple requests capabilities to ESDCTL through a dedicated arbitration mechanism.
- Flexible round robin access arbitration, with equal priority or 50% priority to selective masters.
- Programmable master that controls (lock) accesses to SDRAM/DDR and programmable master that controls (lock) accesses to other memories (= general: NFC, WEIM, PCMCIA).
- Multi-endianness support to all memory controllers.
- Supports memory snooping, an example of which would be monitoring a region in external memory for write accesses:

— The region's location is specified by a base address (from 2 KB up to 16 MB), which is divided into 64 equal segments.
— Each segment has an access status and enable bit in the M3IF register definition.
— M3IF generates a one cycle DMA_ACCESS for each snooping detection.

16.2 External Signal Description

16.2.1 Overview

This section discusses input and output signals between the M3IF, masters, and the memory controllers. Table 16-1 summarizes the interface signals, and is followed by a detailed description of signal functions. Interconnect and timing diagrams are included as part of the detailed discussion on controller operation in Section 16.4, "Functional Description." Detailed M3IF sub-block diagrams are shown in Figure 16-11 and Figure 16-19.

Table 16-1. M3IF Signal Properties

Name	Port	Function	Reset State
M3IF_HRDATA_M#[63:0]	O	read data to master	0
M3IF_HREADY_M#	O	access completion strobe to master	1
M3IF_HRESP_M#	O	error response to master	0
HTRANS[1:0]	O	transfer state bus to memory controllers	0
HPROT	O	protection mode signal to memory controllers	0
HWDATA[31:0]	O	write data bus to memory controllers	0
HADDR[31:0]	O	address bus to memory controllers	0
HBURST[2:0]	O	burst size bus to memory controllers	0
HSIZE[1:0]	O	data transfer width bus to memory controllers	0
HWRITE	O	read/write signal to memory controllers	0
HBSTRB[3:0]	O	byte lane bus to memory controllers	0
HUNALIGN	O	unalign signal to memory controllers	0
BIGENDIAN	O	big/little endian signal (internal) to memory controllers	system dependent
DVFS_GRANT	O	M3IF acknowledge to CCM, indicating that M3IF is ready for frequency changes (DVFS activation)	0
MA10_SHARE	O	MA10 share indication toward EMI	0
M3IF_GUARD_2_PCMCIA	O	assert high during active ESDCTL/MDDRC/NFC/WEIM request execution	0
M3IF_GUARD_2_EIM	O	assert high during active ESDCTL/MDDRC/NFC/PCMCIA request execution	0
EIM_PCMCIA_ACTIVE	O	assert high during request/active WEIM/PCMCIA access execution	0

Table 16-1. M3IF Signal Properties (continued)

Name	Port	Function	Reset State
CHOSEN_SLAVE[1:0]	O	num of slave to be activated	0
DMA_ACCESS	O	snooping detected strobe	0
M3IF_CHOSEN_MASTER[2:0]	O	Reflects the current master number that has ownership on the external DATA bus (to/from memories).	0
HCLK	I	AHB system clock (up to 133 MHz)	0
HCLK32	I	32 KHz clock (use for SDRAM/MDDR refresh)	0
M3IF_HADDR_M#[1][31:0]	I	address bus from master	0
M3IF_HWDATA_M#[63:0]	I	write data bus from master	0
M3IF_HBURST_M#[2:0]	I	burst size bus from master	0
M3IF_HSIZE_M#[1:0]	I	data transfer width bus from master	0
M3IF_HBSTRB_M#[7:0]	I	byte lane bus from master	0
M3IF_HTRANS_M#[1:0]	I	transfer state bus from master	0
M3IF_HWRITE_M#	I	read/write signal from master	0
M3IF_HPROT_M#	I	protection mode signal from master	0
M3IF_HUNALIGN_M#	I	unalign access signal from master	0
M3IF_HMASTLOCK_M#	I	hmasterlock access indication from master	0
M3IF_BIGEND_M#	I	big/little endian signal specific/dedicated from each master connected to M3IF MPG	system dependent
HREADY_EIM	I	EIM controller ready signal	1
HREADY_PCMCIA	I	PCMCIA controller ready signal	1
HREADY_NF	I	NF controller ready signal	1
HRESP_EIM	I	EIM controller error response signal	0
HRESP_PCMCIA	I	PCMCIA controller error response signal	0
HRESP_NF	I	NF controller error response signal	0
HRDATA_EIM	I	read data bus from EIM controller	0
HRDATA_PCMCIA	I	read data bus from PCMCIA controller	0
HRDATA_NF	I	read data bus from NF controller	0
RESET	I	reset signal	1
EIM_GUARD	I	assert high during WEIM write burst access to PSRAM external memory	0
NF_ACTIVE	I	assert high during NFC page fetch	0
DVFS_REQ	I	CCM request signal to M3IF, indicating CCM frequency change is pending for acknowledge (DVFS algorithm)	0

[1]Signals names with suffix "_M#", states for master number. The number of masters in use is system architecture dependent.

16.3 Memory Map and Register Definition

M3IF programming model consists of two classes of registers, M3IF control and lock registers and snooping configuration and status registers as shown in Table 16-2. The control and master lock general register defines the M3IF configurable logic functionality. The configuration and status registers set and monitor snooping activity. All M3IF registers are 32-bits in length with bit fields defined in Figure 16-3 to Figure 16-10. All implemented bits are fully readable and writable in supervisor mode only (an error response will be generated in case of user mode access to M3IF registers). All M3IF (and ESDCTL) registers can be accessed only by a SINGLE word (32-bit) access, through the AHB bus protocol. Accesses of any other size or type will cause an undetermined behavior.

All registers can be accessed by only one master at a time. Multi access to M3IF register causes undetermined behavior. The only exception is M3IF Master Lock General register can be accessed by more than one master at a time. The reset state of each bit is shown underneath the bit field name. An asterisk indicates that the value is dependent on the operating mode selected during reset. Details are provided in the following bit field descriptions.

16.3.1 Memory Map

M3IF supports four different memory controllers. Each memory controller defines a specific memory address mapped as shown in Table 16-2. Table 16-3 shows the M3IF Memory Space Summary.

Table 16-2. M3IF Memory Map

Address	Register	Access	Reset Value	Section/Page
0xD800_3000 (M3IFCTL)	M3IF Control Register	R/W	0x0000_0000	16.3.3.1/16-10
0xD800_3028 (M3IFSCFG0)	M3IF Snooping Configuration Register 0	R/W	0x0000_0000	16.3.3.2/16-12
0xD800_302C (M3IFSCFG1)	M3IF Snooping Configuration Register 1	R/W	0x0000_0000	16.3.3.3/16-13
0xD800_3030 (M3IFSCFG2)	M3IF Snooping Configuration Register 2	R/W	0x0000_0000	16.3.3.3/16-13
0xD800_3034 (M3IFSSR0)	M3IF Snooping Status Register 0	R/W	0x0000_0000	16.3.3.4/16-14
0xD800_3038 (M3IFSSR1)	M3IF Snooping Status Register 1	R/W	0x0000_0000	16.3.3.4/16-14
0xD800_3040 (M3IFMLWE0)	M3IF Master Lock WEIM CS0 Register	R/W	0x0000_0000	16.3.3.5/16-16
0xD800_3044 (M3IFMLWE1)	M3IF Master Lock WEIM CS1 Register	R/W	0x0000_0000	16.3.3.5/16-16
0xD800_3048 (M3IFMLWE2)	M3IF Master Lock WEIM CS2 Register	R/W	0x0000_0000	16.3.3.5/16-16
0xD800_304C (M3IFMLWE3)	M3IF Master Lock WEIM CS3 Register	R/W	0x0000_0000	16.3.3.5/16-16
0xD800_3050 (M3IFMLWE4)	M3IF Master Lock WEIM CS4 Register	R/W	0x0000_0000	16.3.3.5/16-16
0xD800_3054 (M3IFMLWE5)	M3IF Master Lock WEIM CS5 Register	R/W	0x0000_0000	16.3.3.5/16-16

Table 16-3. M3IF Memory Space Summary

Address	Use	Access
ESDCTL/MDDRC Memory Space		
0xA000_0000–0xAFFF_FFFF	CSD0 SDRAM or MDDR memory region (256 Mbyte)	READ/WRITE

Table 16-3. M3IF Memory Space Summary (continued)

Address	Use	Access
0xB000_0000–0xBFFF_FFFF	CSD1 SDRAM or MDDR memory region (256 Mbyte)	READ/WRITE
WEIM Memory Space		
0xC000_0000–0xC7FF_FFFF	WEIM CS0 memory region (128 Mbyte)	READ/WRITE
0xC800_0000–0xCFFF_FFFF	WEIM CS1 memory region (128 Mbyte)	READ/WRITE
0xD000_0000–0xD1FF_FFFF	WEIM CS2 memory region (32 Mbyte)	READ/WRITE
0xD200_0000–0xD3FF_FFFF	WEIM CS3 memory region (32 Mbyte)	READ/WRITE
0xD400_0000–0xD5FF_FFFF	WEIM CS4 memory region (32 Mbyte)	READ/WRITE
0xD600_0000–0xD7FF_FFFF	WEIM CS5 memory region (32 Mbyte)	READ/WRITE
NFC Memory Space		
0xD800_0000–0xD800_0FFF	NAND Flash memory region[1] (4 Kbyte)	READ/WRITE
PCMCIA Memory Space		
0xDC00_0000–0xDFFF_FFFF	PCMCIA memory region (64 Mbyte)	READ/WRITE

1 Can be used as a boot memory region.

16.3.2 Register Summary

Figure 16-2 shows the key to the register fields and Table 16-4 shows the register figure conventions.

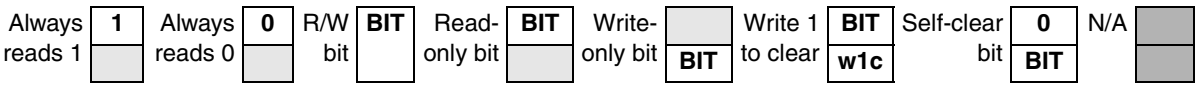


Figure 16-2. Key to Register Fields

Table 16-4. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.

Table 16-4. Register Figure Conventions (continued)

Convention	Description
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 16-5 shows the M3IF register summary.

Table 16-5. M3IF Register Summary

Name		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
0xD800_3000 (M3IFCTL)	R	SDA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	MLSD_EN	MLSD			MRRP							
	W																
0xD800_3028 (M3IFSCFG0)	R	SWBA															
	W																
	R	SWBA					0	0	0	0	0	0	SWSZ				SE
	W																
0xD800_302C (M3IFSCFG1)	R	SSE0															
	W																
	R	SSE0															
	W																
0xD800_3030 (M3IFSCFG2)	R	SSE1															
	W																
	R	SSE1															
	W																
0xD800_3034 (M3IFSSR0)	R	SSS0															
	W																
	R	SSS0															
	W																

Table 16-5. M3IF Register Summary (continued)

Name		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
0xD800_3038 (M3IFSSR1)	R	SSS1															
	W																
	R	SSS1															
	W																
0xD800_3040 (M3IFMLWE0)	R	WEMA0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE0_EN	MLWE0		
	W																
0xD800_3048 (M3IFMLWE2)	R	WEMA2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE2_EN	MLWE2		
	W																
0xD800_3048 (M3IFMLWE2)	R	WEMA2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE2_EN	MLWE2		
	W																
0xD800_304C (M3IFMLWE3)	R	WEMA3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE3_EN	MLWE3		
	W																
0xD800_3050 (M3IFMLWE4)	R	WEMA4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE4_EN	MLWE4		
	W																

Table 16-5. M3IF Register Summary (continued)

Name		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
0xD800_3054 (M3IFMLWE5)	R	WEMA5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE5_EN	MLWE5		
	W																

16.3.3 Register Descriptions

This section contains detailed register descriptions for M3IF registers.

16.3.3.1 M3IF Control Register (M3IFCTL)

M3IFCTL contains access status, provides access control to SDRAM/MDDR memory devices and arbitration priority for M3IF port masters. The field assignments for this register are shown in Figure 16-3 and the field descriptions are listed in Table 16-6.

0xD800_3000 (M3IFCTL) Access: User read-write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	MLSD_EN	MLSD			MRRP							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-3. M3IF Control Register

Table 16-6. M3IF Control Register Field Descriptions

Field	Description
31 SDA	SDRAM/MDDR Memory Active. This is a read-only status bit, that if set, indicates that an active/pending access to SDRAM/MDDR memory exists. The SDA bit will be set on one of the following conditions: • MLSD_EN cleared—Any active/pending access to SDRAM/MDDR memory space will set the bit (until the access is completed). • MLSD_EN is set—Any accesses to SDRAM/MDDR memory space initiated previously to MLSD_EN assertion, will keep the SDA status bit set. The bit will clear after all pending/active accesses execution is completed. Access from master number equal to MLSD field will not assert the status bit. **Note:** When MLSD_EN is set, any new accesses (initiated after MLSD_EN assertion) to SDRAM/MDDR not from MLSD master will be pending without setting SDA to 1. Only the SDRAM/MDDR memory space region will be lock to the MLSD port. Accesses to M3IF/ESDCTL registers are available to all masters in the system and its system/software responsibility not to access those registers during lock period. 0 No active/pending access to SDRAM/MDDR memory exists. 1 Indicates an active/pending access to SDRAM/MDDR memory exists.
30–12	Reserved
11 MLSD_EN	Master Lock SDRAM/MDDR Access. This bit enables the Master Control SDRAM/MDDR access (MLSD). The reset value of this bit is “0”. 0 Master Control SDRAM/MDDR access (MLSD) disabled. 1 Master Control SDRAM/MDDR access (MLSD) enabled.
10–8 MLSD	Master Lock SDRAM/MDDR Access. This 3-bit field defines the master port number (MPG) that will be the only master in the system that will be served by the SDRAM/MDDR controller. All accesses toward the SDRAM/MDDR from the other masters will be postponed, until the MLSD master will clear MLSD_EN bit. The reset value of the MLSD is “0”. **Note:** Accesses to ESDCTL registers are not effected by the MLSD field. For example, they can be accessed by any master even if MLSD_EN is set.
10–8 MLSD	Prior to lock accesses, the MLSD master should perform the following steps: 1. Set MLSD_EN bit and MLSD field (with the desired value) in the M3IFCTL register. 2. Read M3IFCTL register and check: 3. SDA status bit is cleared (no pending/active access to SDRAM/MDDR memory space exists). 4. MLSD_EN bit is set. 5. MLSD (value) points to the required port number (master port number that requires lock access). 000 Master Port Gasket 0 001 Master Port Gasket 1 010 Master Port Gasket 2 011 Master Port Gasket 3 100 Master Port Gasket 4 101 Master Port Gasket 5 110 Master Port Gasket 6 111 Master Port Gasket 7
7–0 MRRP	Master Round Robin Priority. MRRP field is an 8-bit field with one bit per master (bit #i to master #i). Masters with their MRRP bit set are added to a priority arbitration “list” so that together they will have 50% probability to gain access through both M3A and MAB arbitration processes (50% probability for each one of the arbitration separately). Assertion of MRRP bit for an unused master is forbidden. If all MRRP bits are cleared the masters will have equal probability to pass the arbitration processes. For more details about the M3IF arbitration see Section 16.4.3.2, “M3A–Find First 1 (FF1) Algorithm.” 0 The respective master is not on the priority arbitration “list”. 1 Add respective master to priority arbitration “list” with a 50% probability to pass the arbitration processes.

16.3.3.2 M3IF Snooping Configuration Register 0 (M3IFSCFG0)

M3IFSCFG0 register contains the Snooping window base address, the size of snooping window and the snooping control bit fields which are used by the M3IF to monitor the write access. The Snooping feature is described in detail in Section 16.4.5, "Snooping Logic." The field assignments for this register are shown in Figure 16-5 and the field descriptions are listed in Table 16-7.

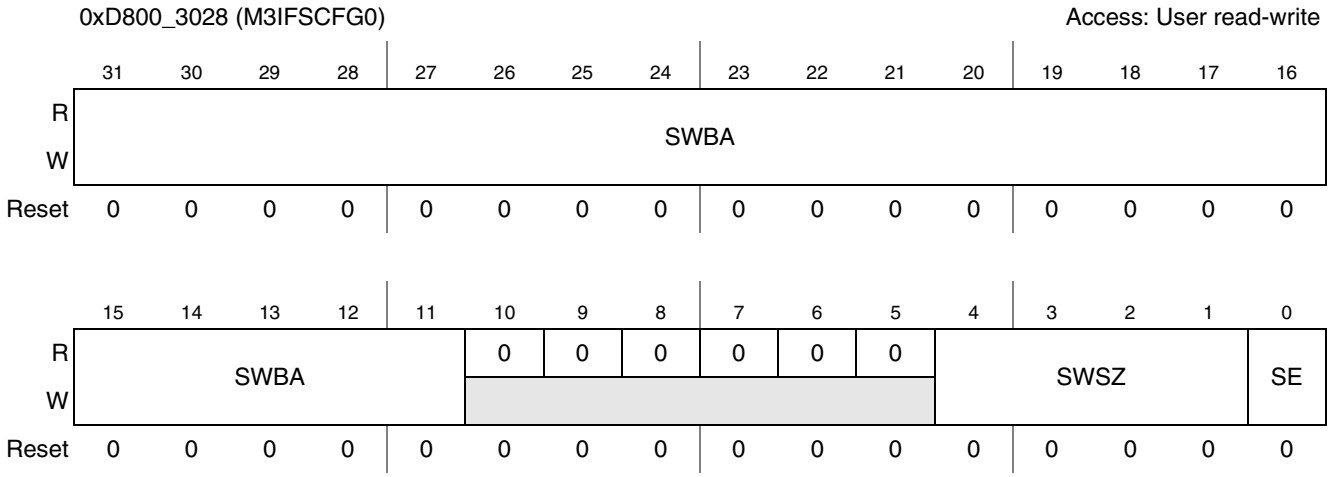


Figure 16-5. M3IF Snooping Configuration Register 0 (M3IFSCFG0)

Table 16-7. M3IF Snooping Configuration Register 0 Field Descriptions

Field	Description
31–11 SWBA	Snooping Window Base Address. This field defines the snooping window base address to be monitored by the M3IF. M3IF monitors write accesses to the memory region above the base address window.
10–5	Reserved
4–1 SWSZ	Snooping Window Size. This field define the snooping window size as described in Table 16-8.
0 SE	Snooping Enable. This bit enables snooping detection. The M3IF monitors and detects write accesses to the snooping window. 0 Snooping feature is disabled. 1 Snooping feature is enabled.

Table 16-8. SWSZ Field Descriptions

SWSZ	Snooping Window Size	Window Base Address Bits	Window Address Bits in Use
0000	2 KByte	[31:11]	[10:0]
0001	4 KByte	[31:12]	[11:0]
0010	8 KByte	[31:13]	[12:0]
0011	16 KByte	[31:14]	[13:0]
0100	32 KByte	[31:15]	[14:0]

Table 16-8. SWSZ Field Descriptions (continued)

SWSZ	Snooping Window Size	Window Base Address Bits	Window Address Bits in Use
0101	64 KByte	[31:16]	[15:0]
0110	128 KByte	[31:17]	[16:0]
0111	256 KByte	[31:18]	[17:0]
1000	512 KByte	[31:19]	[18:0]
1001	1 MByte	[31:20]	[19:0]
1010	2 MByte	[31:21]	[20:0]
1011	4 MByte	[31:22]	[21:0]
1100	8 MByte	[31:23]	[22:0]
1101	16 MByte	[31:24]	[23:0]
1110	Reserved	—	—
1111	Reserved	—	—

16.3.3.3 M3IF Snooping Configuration Register 1–2 (M3IFSCFG1–2)

M3IFSCFG1 register contains enable bits for lower 32 segments [31:0] in M3IFSCFG0 register. M3IFSCFG2 register contains enable bits for upper 32 segments [63:32] in M3IFSCFG0 register. Snooping feature is described in detail in Section 16.4.5, “Snooping Logic.” The field assignments for these register are shown in Figure 16-6 and Figure 16-7 and the field descriptions are listed in Table 16-9.

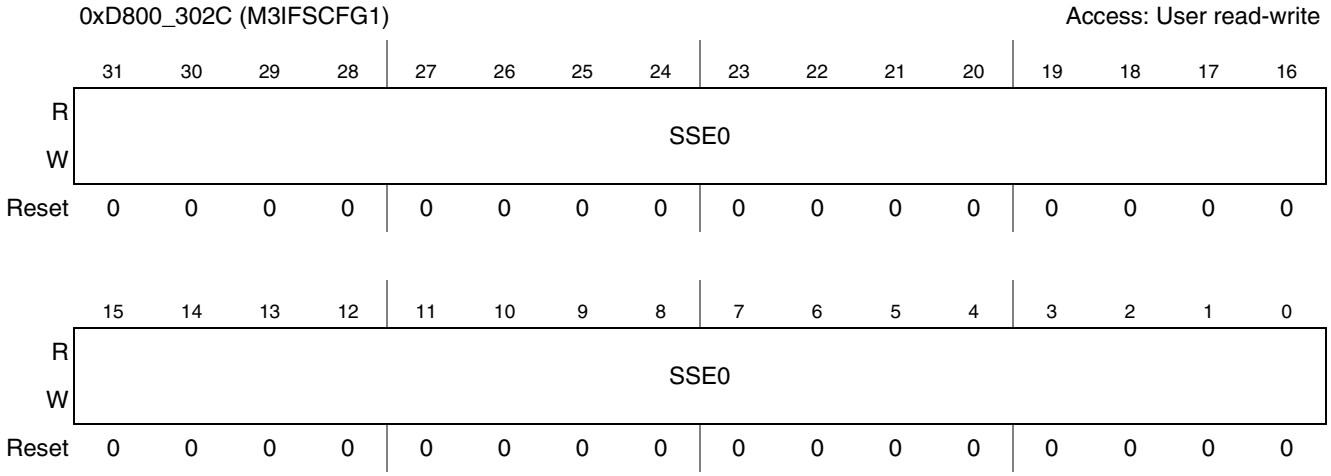


Figure 16-6. M3IF Snooping Configuration Register 1 (M3IFSCFG1)

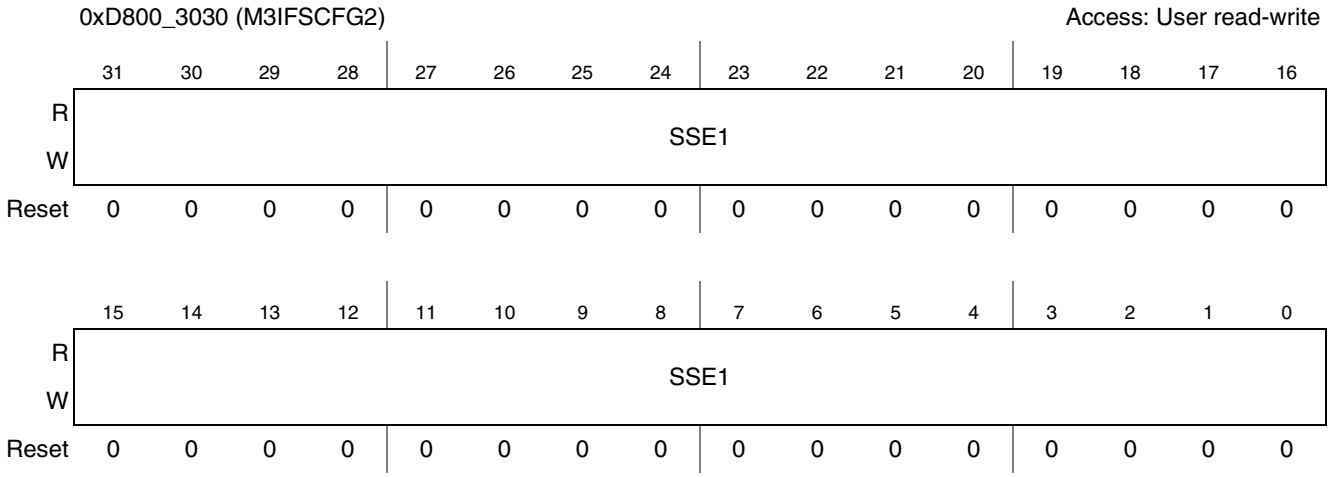


Figure 16-7. M3IF Snooping Configuration Register 2 (M3IFSCFG2)

Table 16-9. M3IF Snooping Configuration Register 1–2 Field Descriptions

Field	Description
31–0 SSE0	Snooping Segment Enable 0. This register contains the enable bits for the lower 32 segments [31:0] in the snooping window (defined by the M3IFSCFG0 register). If snooping is enabled for segment #x (respective SSE0 bit is high), than any write access detected to that segment will set the DMA_ACCESS for one cycle and the respective snooping status bit will be set. If the SSE0 bit is low, and a write access to the respective segment is detected by the M3IF, only the relevant status bit in the snooping status register will be set but the DMA_ACCESS will not be generated. 0 Snooping segment #x is disabled. 1 Snooping segment #x is enabled.
31–0 SSE1	Snooping Segment Enable 1. This register contains the enable bits for the higher 32 segments [63:32] in the snooping window (defined by the M3IFSCFG1 register). If snooping is enabled for segment #x (respective SSE1 bit is high), than any write access detected to that segment will set the DMA_ACCESS for one cycle and the respective snooping status bit will be set. If the SSE1 bit is low, and a write access to the respective segment is detected by the M3IF, only the relevant status bit in the snooping status register will be set but the DMA_ACCESS will not be generated. 0 Snooping segment #x is disabled. 1 Snooping segment #x is enabled.

16.3.3.4 M3IF Snooping Status Register 0–1 (M3IFSSR0–1)

M3IFSSR0 register contains the snooping status bits for the lower 32 segments. M3IFSSR1 register contains the snooping status bits for the higher 32 segments.The Snooping feature is described in detail in Section 16.4.5, “Snooping Logic.” The field assignments for these registers are shown in Figure 16-8 and Figure 16-9 and the field descriptions are listed in Table 16-10 and Table 16-11.

0xD800_3034 (M3IFSSR0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SSS0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SSS0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-8. M3IF Snooping Status Register 0 (M3IFSSR0)

Table 16-10. M3IF Snooping Status Register 0 Field Descriptions

Field	Description
31–0 SSS0	Snooping Segment Status 0. This register contains the snooping status bits for the lower 32 segments [31:0] in the snooping window (defined by the M3IFSCFG0 register). A bit in the SSS0 register is asserted if snooping to the respective segment occurred. **Note:** If snooping occurred the status bit will be updated regardless of the respective snooping segment enable bit SSE0[x]. The DMA_ACCESS will be asserted only if the respective snooping segment enable bit SSE0[x] is enabled. 0 Snooping for segment #x did not occur. 1 Snooping for segment #x occurred.

0xD800_3038 (M3IFSSR1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SSS1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SSS1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-9. M3IF Snooping Status Register 1 (M3IFSSR1)

Table 16-11. M3IF Snooping Status Register 1 Field Descriptions

Field	Description
31–0 SSS1	Snooping Segment Status 1. This register contains the snooping status bits for the higher 32 segments [63:32] in the snooping window (defined by the M3IFSCFG1 register). A bit in the SSS1 register is asserted if snooping to the respective segment occurred. **Note:** If snooping occurred the status bit will be updated regardless of the respective snooping segment enable bit SSE0[x]. The DMA_ACCESS will be asserted only if the respective snooping segment enable bit SSE1[x] is enabled. 0 Snooping for segment #x did not occur. 1 Snooping for segment #x has occurred.

16.3.3.5 M3IF Master Lock WEIM CSx Register (M3IFMLWEx)

The field assignments for this register are shown in Figure 16-10 and the field descriptions are listed in Table 16-12.

0xD800_3040 (M3IFMLWE0)
0xD800_3044 (M3IFMLWE1)
0xD800_3048 (M3IFMLWE2)
0xD800_304C (M3IFMLWE3)
0xD800_3050 (M3IFMLWE4)
0xD800_3054 (M3IFMLWE5)

Access: User read-write

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

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

Figure 16-10. M3IF Lock General Register (M3IFMLGE)

Table 16-12. M3IF Lock General Register Field Descriptions

Field	Description
31 WEMAx	WEIM CSx (0-) Memory Active. This is a read-only status bit, that if set indicates that an active/pending access to a WEIM CSx memory exists. The WEIMx bit is set on one of the following conditions: • MLWEx_EN cleared—Any active/pending access to WEIM CSx memory space will set the bit (until the access is completed). • MLWEx_EN is set—Any accesses to WEIM CSx memory space initiated previously to MLWEx_EN assertion, will keep the WEMAx status bit set. The bit clears after all pending/active accesses execution is completed. Access from master number equal to MLWEx field does not assert the status bit. **Note:** When MLWEx_EN is set, any new accesses (initiated after MLWEx_EN assertion) to WEIM CSx memories (or to M3IFMLWEx register) not from MLWEx master will be pending without setting WEMAx to 1. Both the M3IFMLWEx register and the WEIM CSx space region will be lock to the MLWEx port. 0 No active/pending access to WEIM CSx memory exists. 1 Indicates an active/pending access to WEIM CSx memory exists.
30–4	Reserved
3 MLWEx_EN	Master Lock WEIM CSx Access Enable. This bit enables the Master Lock WEIM CSx access (MLWEx). The reset value of this bit is 0. **Note:** After MLWEx master does not need the lock any more, the master should clear MLWEx_EN bit, so WEIM CSx memory region is open to all masters. 0 Master Lock WEIM CSx access (MLWEx) is disabled. 1 Master Lock WEIM CSx access (MLWEx) is enabled.
2–0 MLWEx	Master Lock WEIM CSxl Access. This 3 bits field defines the master port number (MPG) that will be the only one in the system served by the WEIM controller. All accesses to the WEIM CSx memory space from the other masters will be postponed. The reset value of the MLWEx is 0. 1. Prior to lock accesses, the MLGE master should perform the following steps: 2. Set the MLWEx_EN bit and the MLWEx field (with the desired value) in the M3IFMLWEx register. 3. Read M3IFMLWEx register and check: WEMAx status bit is cleared (no pending/active accesses to WEIM CSx memory space exists). MLWEx_EN bit is set. MLWEx (value) points to the required port number (master port number that requires lock accesses). 000 Master Port Gasket 0 001 Master Port Gasket 1 010 Master Port Gasket 2 011 Master Port Gasket 3 100 Master Port Gasket 4 101 Master Port Gasket 5 110 Master Port Gasket 6 111 Master Port Gasket 7

16.4 Functional Description

This section provides the functional description for the M3IF module.

16.4.1 Master Port Gasket (MPG)

MPG is a flexible port gasket. Up to 8 masters can be connected to the M3IF with any combination of MPG type, which support the following different port types:

- MPG—Master port gasket for ARM9 AMBA-AHB lite 32 bits data bus.
- MPG64—Master port gasket AMBA-AHB lite 64 bits data bus

The number and type of ports in use is system dependent, and the unused ports will be unconnected at the system level. Each one of the MPG gaskets communicates with a single master, through one of the (two) defined port interfaces/protocols.

16.4.1.1 Overview of MPG Operation

MPG port gasket is used for those system masters that are 32-bit ARM9 AHB lite bus compliant. MPG port gasket appears as another slave to any master it connects to. Table 16-13 lists the access types supported by the MPG.

Table 16-13. MPG Supported Burst Accesses

HBURST	TYPE	M3IF SLAVES			
		ESDCTL 32-bit	EIM 32-bit	NFC 16/32-bit	PCMCIA 8/16-bit
000	SINGLE	YES	YES	YES	YES
001	INCR	YES	YES	YES	YES
010	WRAP 4	YES	YES	NO	YES
011	INCR 4	YES	YES	YES	YES
100	WRAP 8	YES	YES	NO	YES
101	INCR 8	YES	YES	YES	YES
110	WRAP 16	NO	YES	NO	YES
111	INCR 16	NO	YES	YES	YES

NOTE

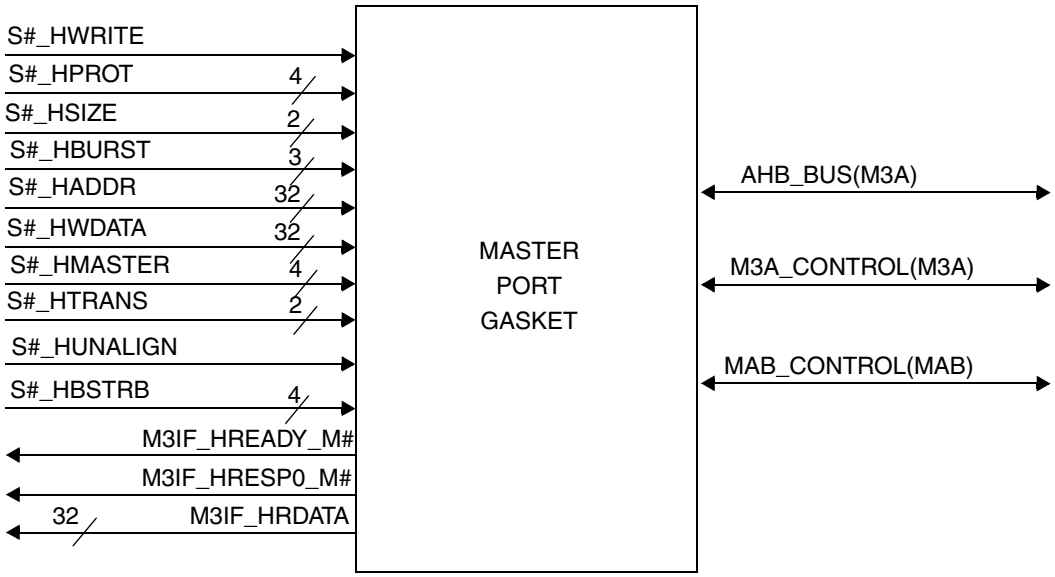
Unsupported access types will produce undefined behavior but an error response will not be generated.

Figure 16-11 shows MPG port interface diagram. The interface is ARM 11 AMBA-AHB lite compliant (does not support RETRY and SPLIT transfers). MPG works with both M3A and MAB, and output AHB_BUS and CONTROL signals to/from M3A (including request to ESDCTL/MDDRC signal and request to non-ESDCTL/MDDRC = general signal). MPG decodes AHB bus inputs and convert them to MAB_CONTROL bus, which includes ADDR, DATA and CONTROL signals (like suspend and abort commands).

Once an accesses is initiated by one of the M3IF masters, the access reaches the respective MPG. The MPG asserts the request signal toward the M3A which starts the arbitration process. Once the arbitration is completed and the request can gain access to the bus, the request is accepted by the MPG and the handshake between the MPG and the M3A is completed for that access. If the access was not targeted toward the ESDCTL, the master can start the access (by passing the master AHB bus) toward the respective slave (NFC, EIM).

If initiated access is targeted to the ESDCTL after M3A arbitration process is completed, the request is transferred toward the MAB, which arbitrates and schedules the access toward the ESDCTL as a function of ESDCTL state. An internal handshake between the MAB and ESDCTL is used to schedule the new

access, and once the handshake is completed, the MAB asserts the request accept signal toward the MPG. All ESDCTL related AHB signals are transferred from the master to the MPG which convert them to an internal protocol between the MPG and the MAB.



M#—M3IF Master port number (from 0 to 8)
S#—Slave port number
MAB—Master Arbitrator and Buffering

Figure 16-11. Master Port Gasket (MPG) Interface Diagram

MPG also converts MAB or M3A outputs to the AHB standard interface. Table 16-14 presents the signals name in both modules.

Table 16-14. MPG MAX Signals

AHB Master—Signal Name	MPG—Signal Name	Description
S#_HWRITE (o)	M3IF_HWRITE_M# (I)	HWRITE is HIGH—Indicates a write transfer. HWRITE is LOW—Indicates a read transfer.
S#_HPROT[3:0] (O)	M3IF_HPROT_M#[3:0] (I)	The protection control signals provide additional information about a bus access. For more information on this signal see Protection discussion in the AHB document. M3IF is using only HPROT[1] signal—user/supervisor access. This signal is used to protect both registers and restricted memory regions. An error response will be generated in case of protection violation, for example, access supervisor registers/memory regions in user mode.
S#_HSIZE[1:0] (O)	M3IF_HSIZE_M#[1:0] (I)	Indicates the size of the transfer. 00 8-bits (Byte) 01 16-bits (Half-word) 10 32-bits (Word) 11 Not define for MPG
S#_HMASTER[3:0] (O)	not defined	Not used by M3IF.

Table 16-14. MPG MAX Signals (continued)

AHB Master—Signal Name	MPG—Signal Name	Description
S#_HBURST[2:0] (O)	M3IF_HBURST_M#[2:0] (I)	Burst information is provided using HBURST signal, and the 8 possible types are: 000 SINGLE (Single transfer) 001 INCR (Incrementing burst of unspecified length) 010 WRAP4 (4-beat wrapping burst) 011 INCR4 (4-beat incrementing burst) 100 WRAP8 (8-beat wrapping burst) 101 INCR8 (8-beat incrementing burst) 110 WRAP16 (16-beat wrapping burst)[1] 111 INCR16 (16-beat incrementing burst)[1]
S#_HADDR[31:0] (O)	M3IF_HADDR_M#[31:0] (I)	Indicates the 32 bits memory ADDRESS bus.
S#_HWDATA[31:0] (O)	M3IF_HWDATA_M#[31:0] (I)	The write data bus is driven by the master during write transfers (on data phase). If the transfer is extended then the bus master hold the data valid until the transfer completes, as indicated by HREADY HIGH.
S#_HTRANS[1:0] (O)	M3IF_HTRANS_M#[1:0] (I)	Each transfer can be classified into one of four different types, as indicated by the HTRANS[1:0] signals: 00 IDLE. Indicates that no data transfer is required. The IDLE transfer type is used when a bus master is granted the bus, but does not wish to perform a data transfer. M3IF will provide a zero wait state OKAY response to IDLE transfers. 01 BUSY. BUSY transfer type allows bus masters to insert IDLE cycles in the middle of bursts of transfers. This transfer type indicates that the bus master is continuing with a burst of transfers, but the next transfer cannot take place immediately. M3IF will provide a zero wait state OKAY response to IDLE transfers. When a master uses the BUSY transfer type the address and control signals reflects the next transfer in the burst. 10 NONSEQ. Indicates the first transfer of a burst or a single transfer. Single transfers on the bus are treated as bursts of one and therefore transfer type is NONSEQUENTIAL. 11 SEQ. The remaining transfers in a burst are SEQUENTIAL and the address and control are related to the previous transfer. In the case of a wrapping burst the address of the transfer wraps at the boundary equal to the size (in bytes) multiplied by the number of beats in the transfer (4,8 or 16).
S#_HBSTRB[3:0] (O)	M3IF_HBSTRB_M#[3:0] (I)	Indicates which byte lanes are valid for each word transfer.[2]
S#_HUNALIGN (O)	M3IF_HUNALIGN (I)	Signal to indicate an unalign access requiring HBSTRB information.[2]
S#_HMASTLOCK (O)	M3IF_HMASTLOCK (I)	Indicates that the current master is performing a locked sequence of transfers.
S#_HREADY (I)	M3IF_HREADY_M# (O)	M3IF uses HREADY signal to insert the appropriate number of wait states in to the transfer (the M3IF adds wait states as long as the HREADY in signal is deasserted). The transfer completes with HREADY HIGH (and an OKAY response, which indicates the successful completion of the transfer). One wait state will be added for every cycle that has HREADY diasserted.

Table 16-14. MPG MAX Signals (continued)

AHB Master—Signal Name	MPG—Signal Name	Description
S#_HRESP0 (I)	M3IF_HRESP0_M# (O)	HRESP0 response is used by M3IF to indicate some form of error condition with the associated transfer. Since M3IF is AHB Lite compliant (AHB SPLIT and RETRY protocols are not supported) means that only one response signal is needed. HREPS0 encoding is: 0 OKAY. When HREADY is HIGH this shows the transfer has completed successfully. OKAY response is also used for any additional cycles that are inserted, with HREADY LOW. 1 ERROR. This (two cycle) response shows an error has occurred. The error condition is signalled to the bus master so it is aware the transfer has been unsuccessful. M3IF response with Error on cases as specified in Section 16.4.1.4, "MPG Transfer Response."
S#_HRDATA[31:0] (I)	M3IF_HRDATA[31:0] (O)	The read data bus is driven by the M3IF during read transfers. If M3IF extends the read transfer by holding HREADY LOW then M3IF will provide valid data at the end of the final cycle of the transfer, as indicated by HREADY HIGH.

[1]INCR16/WRAP16 are supported only for accesses addressed to the EIM or PCMCIA.
[2]HUANLIGN and HBSTRB are supported only by ESDCTL and WEIM.

A granted bus master starts an AMBA AHB transfer by driving the address and control signals. These signals provides information on the address, direction and width of the transfer, as well as indication if the transfer forms parts of a burst. Two different forms of burst transfers are allowed:

- Incrementing bursts, which do not wrap at address boundaries.
- Wrapping bursts, which wrap at particular address boundaries.

A write data bus is used to move data from the master to M3IF, while read data bus is used to move data from M3IF to the master.

Every transfer consists of:

- An address and control cycle (address phase)
- One or more cycles for the data (data phase)

Since the first address phase cannot be extended (since it will always get HREADY asserted high) M3IF samples all control bus during first address phase, so if the master does not gain access immediately, the address phase information will be saved. The data, however, can be extended by using M3IF_HREADY_MX signal. When LOW this signal causes wait states to be inserted into the transfer and allows extra time for M3IF (ESDCTL/MDDRC or memories) to provide or sample data. In this way, back to back access between different/same slave can be performed and MPG will store all needed bus/signals so that when the master gains access, all the needed bus/signals will be available. During a transfer, M3IF shows the status using only one response signal HRESP0 (since M3IF is only AHB Lite compliant).

- 0-OKAY—The OKAY response is used to indicate that the transfer is progressing normally and when M3IF_HREADY_MX goes high this shows the transfer has completed successfully.
- 1-ERROR—The ERROR response indicates that a transfer error has occurred and the transfer has been unsuccessful.

16.4.1.2 MPG Basic Transfer

An AMBA AHB transfer consists of two distinct sections:

- Address phase.
- Data phase that may require several cycles. This is achieved using M3IF_HREADY_MX signal.

Figure 16-12 shows the simplest transfer, one data with no wait states.

- The AHB lite bus compliant master drives the address and control signals onto the bus after the rising edge of the clock.
- M3IF then samples the address and control information in the next rising edge of the clock and access starts (memory is not busy).
- After M3IF has sampled the address and control (and derived the appropriate command to the memory) it can start to drive the appropriate response and this is sampled by the bus master on the third rising edge of the clock.

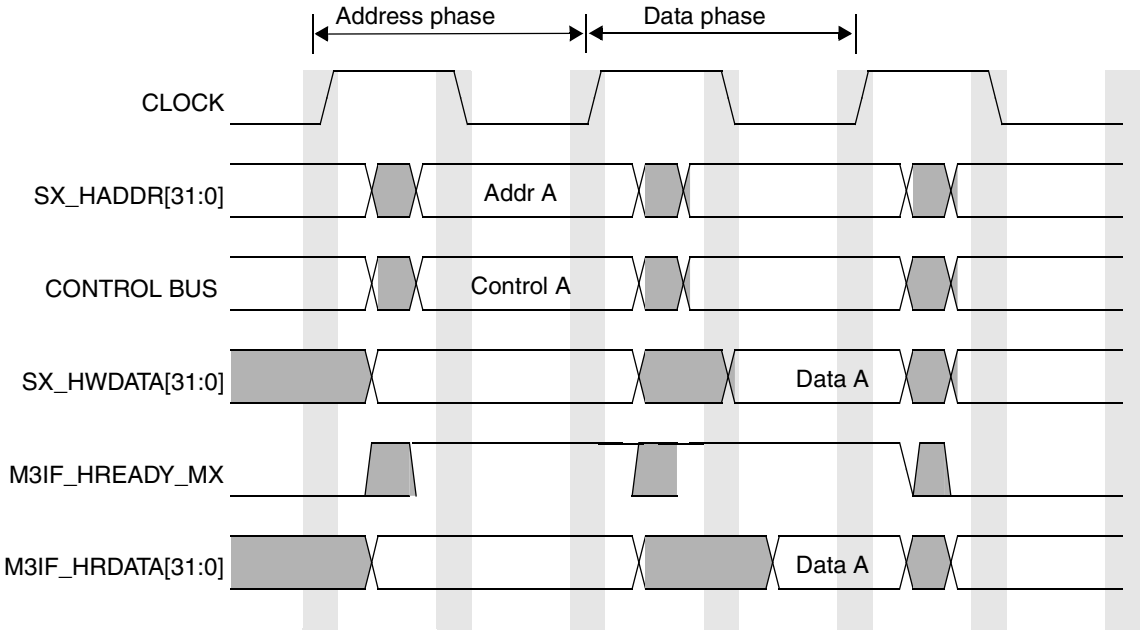


Figure 16-12. MPG Simple Transfer

The address phase of any transfer occurs during the data phase of the previous transfer. This overlapping of address and data is at the pipelined nature of the AHB bus and allows for high performance operation. M3IF may insert wait states into any transfer, as shown in Figure 16-13, which extends the transfer allowing additional time for completion.

- For write operations the bus master will hold the data stable throughout the extended cycles.
- For read transfer, M3IF does not have to provide valid data until the transfer is about to complete.

When a transfer is extended in this way, it will have side effect to extend address phase for the next transfer. This is shown in Figure 16-14, which shows three transfers to unrelated addresses, A, B, and C.

- The transfers to addresses A and C are both zero state.
- The transfer to address B is one wait state.

Extending the data phase of the transfer to address B has the effect of extending the address phase of the transfer to address C.

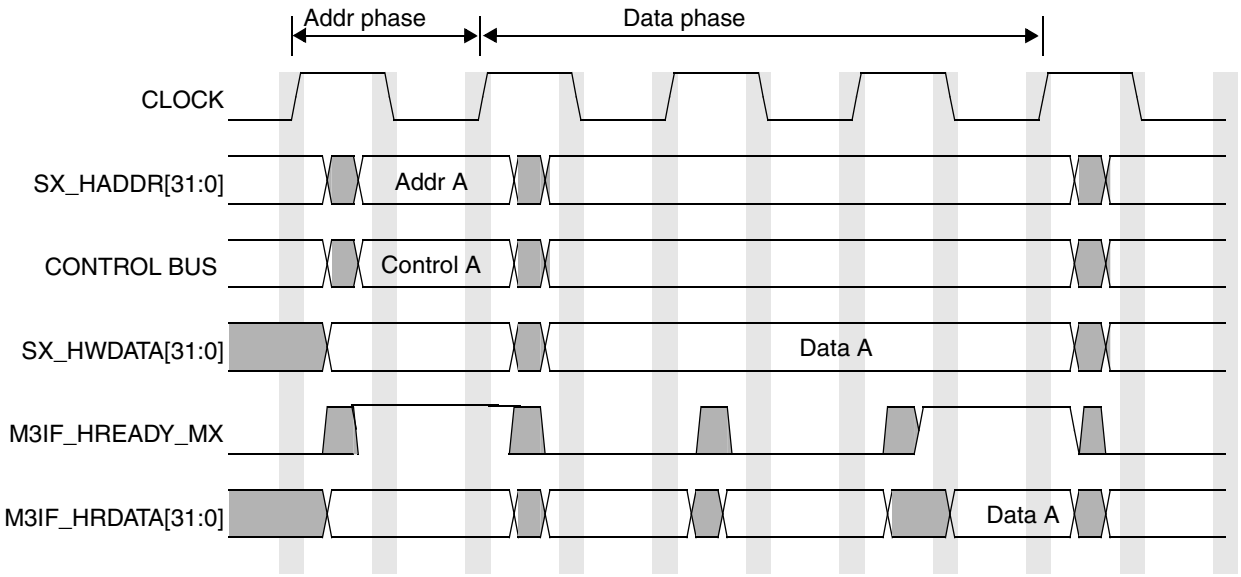


Figure 16-13. MPG with Wait States

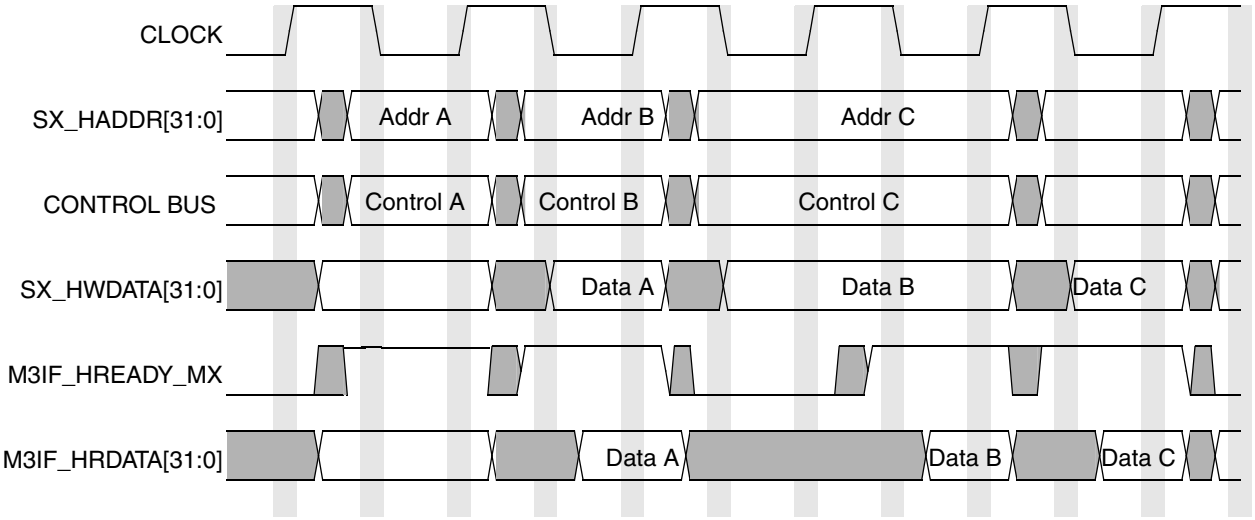


Figure 16-14. MPG Multiple Transfers

16.4.1.3 MPG Transfer Type

Every transfer can be classified into one of four different types, as indicated by SX_HTRANS[1:0] signals as described in Table 16-14. Figure 16-15 shows a number of different transfer types being used.

- The first transfer is the start of a burst and therefore is NON-SEQUENTIAL.
- The master is unable to perform the second transfer of the burst immediately and therefore the master uses BUSY transfer to delay the start of the next transfer (after M3IF sees BUSY with HREADY high it continues to give HREADY high until HTRANS bus changes from BUSY and

then HREADY will act as usual). In this example the master requires only one cycle before it is ready to start the next transfer in the burst, which completes with no wait states.

- The master performs the third transfer of the burst immediately, but this time the M3IF is unable to complete and uses M3IF_HREADY_MX to insert a single wait state.
- The final transfer of the burst completes with zero wait states.

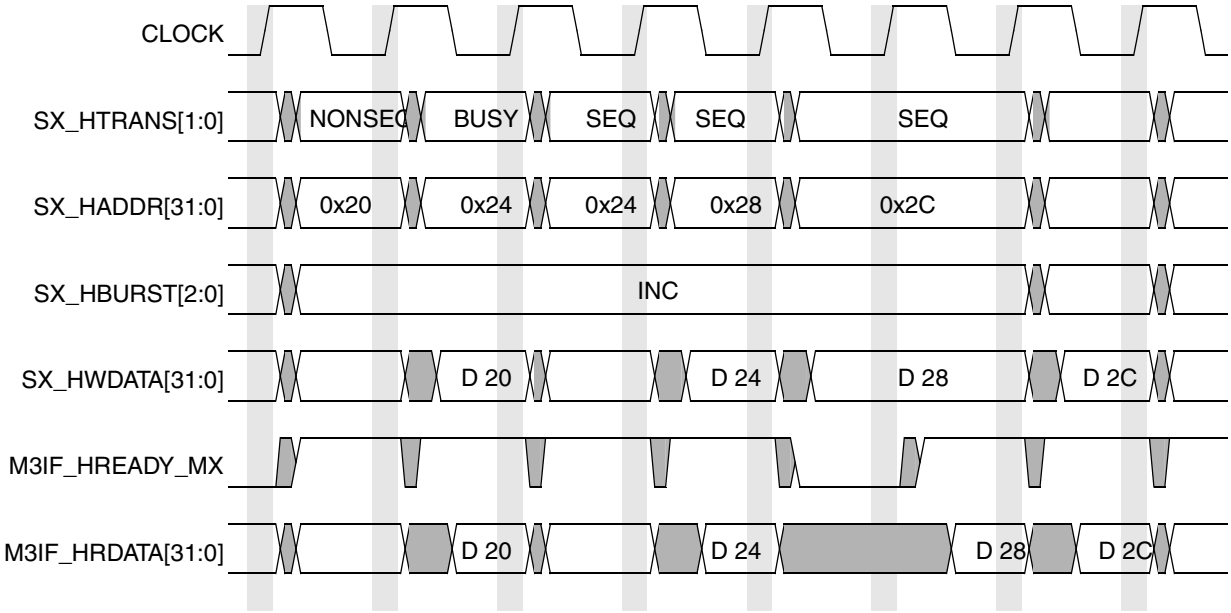


Figure 16-15. MPG—Transfer Type Examples

16.4.1.4 MPG Transfer Response

Whenever M3IF is accessed it provides a response which indicates the status of the transfer. The M3IF_HREADY_MX signal is used to extend the transfer and this works in combination with the response signals, M3IF_HRESP_MX, which provide the status of the transfer. M3IF can complete the transfer in a number of ways:

- Complete the transfer immediately.
- Insert one or more wait states to allow time to complete the transfer.
- Signal error to indicate that the transfer has failed.

The M3IF_HREADY_MX signal is used to extend the data portion/phase of a transfer. When LOW the M3IF_HREADY_MX indicates the transfer is to be extended and when HIGH indicates that data transfer had completed. Both M3IF_HREADY_MX and M3IF_HRESP0 encoding is described in Figure 16-12. It should be noted that M3IF does not support AMBA AHB, SPLIT and RETRY transfer response.

A transfer will complete successfully (as defined by the AHB bus protocol) with M3IF_HREADY_MX HIGH and an OKAY response (M3IF_HRESP[1] LOW). A transfer will complete unsuccessfully (ERROR response) with two consecutive cycles of M3IF_HRESP[1] HIGH, while during the first cycle M3IF_HREADY_MX is LOW and during the second cycle M3IF_HREADY_MX is HIGH (as defined by the AHB bus protocol). The ERROR response is used by the M3IF to indicate one of the following error types (which can be associated with the transfer):

- Master is trying to access a disabled CSD in the ESDCTL/MDDRC system register.
- Master in user mode is trying to access a CSD that is configured to SUPERVISOR access only.
- System gave software reset command to ESDCTL/MDDRC while access to ESDCTL/MDDRC is in progress.
- Error response coming from all other memory controllers (except ESDCTL/MDDRC).
- Access to ESDCTL registers during an active access to SDRAM memory.

NOTE

M3IF controls/handles ESDCTL error response logic, and only transfer the error response signal from all other memory controllers (PCMCIA, NFC, and WEIM). For more details regarding the error response generation from other memory controllers, consult the respective memory controller chapter available in the respective system architecture.

If an error response is generated by the MPG on the beginning of an access, the access will not be executed and none of the data that is supposed to be read/write will get transferred; however, if the error response has been given after few data transfers (in a burst access), the status of the first data transfer before the error response, for write access, is unknown (data maybe written or not) and the master should treat the data of the whole access as unknown data. In the case that this access was a read access the data that has been transferred until the error response is valid data and master can use it. If an error occurs during a burst access, the M3IF will generate an (AHB) error response for all remaining beats from the burst.

16.4.1.5 MPG Burst Operation

Four, eight and sixteen-beat bursts are defined in the AMBA AHB protocol, as well as incremental undefined length bursts and single transfers. Both incrementing and wrapping bursts are supported in the protocol. A detailed description of the supported access type by the MPG is shown at Table 16-13.

Burst information is provided using SX_HBURST[2:0] signal and the eight possible types are defined in Figure 16-12. It is acceptable to perform single transfers using an unspecified length incrementing burst which only has a burst length of one.

The burst size indicates the number of beats in the burst, not the number of bytes transferred. The total amount of data transferred in a burst is calculated by multiplying the number of beats by the amount of data in each beat, as indicated by SX_HSIZE[1:0]. SX_HSIZE[1:0] encoding is shown in Figure 16-12. The size is used in conjunction with the SX_HBURST[2:0] signals to determine the address boundary for wrapping bursts.

All transfers within a burst must be aligned to the address boundary equal to the size of the transfer (that must be a word as mentioned). For example, word transfers must be aligned to word address boundaries (that is A[1:0]=00). If an unalign access is being perform HUNALIGN signal must be asserted high and the respective HBSTRB bus must be given by the master.

NOTE

Unaligned burst crossing bus width boundary is supported only if eventual number of transfers on the bus is not higher than the value implied by HBURST.

Four beat wrapping and incrementing burst are shown in Figure 16-16 and Figure 16-17, respectively.

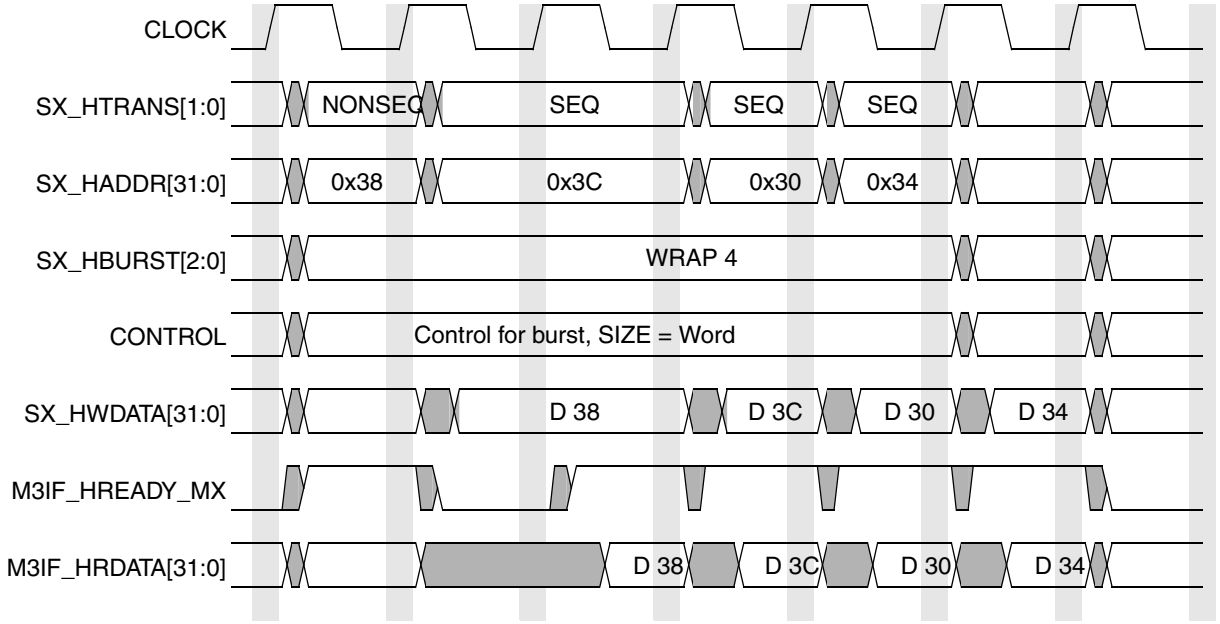


Figure 16-16. MPG Four Beat Wrapping Burst

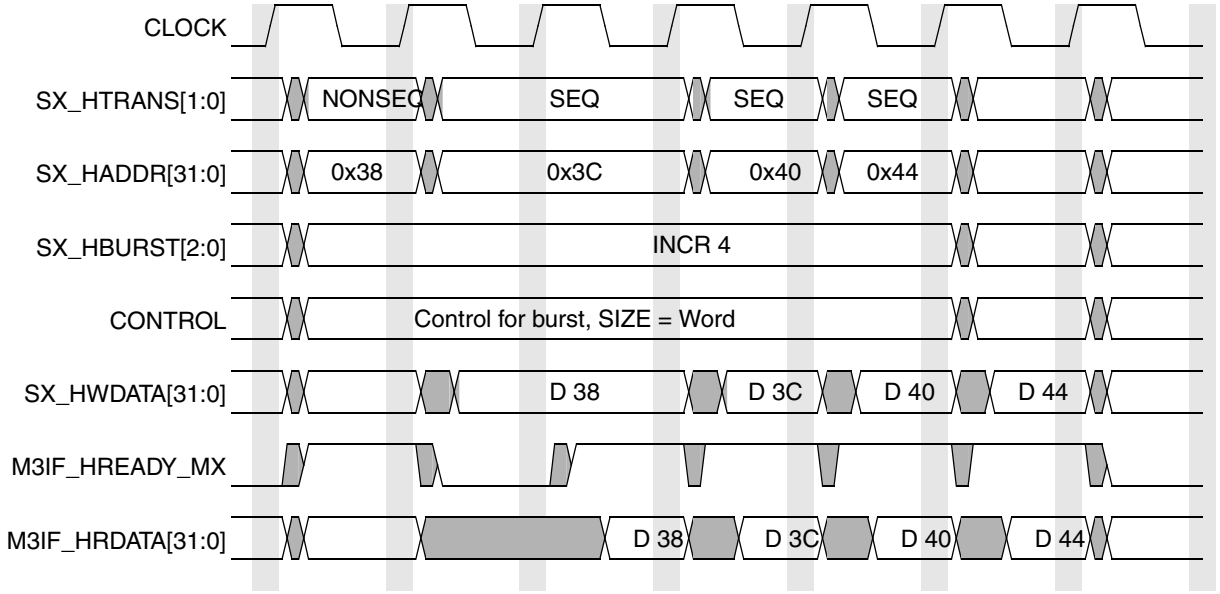


Figure 16-17. MPG Four Beat Incrementing Burst

16.4.1.6 MPG Early Burst Termination

M3IF can determine when a burst has terminated early by monitoring the SX_HTRANS[1:0] signals and ensuring that after the start of the burst every transfer is labelled as SEQUENTIAL or BUSY. If a NON-SEQUENTIAL transfer occurs in middle of a burst it indicates that a new burst has started and therefore the previous one must be terminated immediately. If an IDLE transfer occurs in middle of a burst, it indicates the burst should be terminated immediately.

If a master cannot complete a burst because it loses ownership of the bus (for example, MAX slave port SX_HTRANS[1:0] is IDLE during a burst access due to MAX internal arbitration logic, means that the served MAX master port loses ownership of the bus) then it must rebuild the burst appropriately when it re-gains access to the bus. For example, if a master has only completed one beat of a four-beat burst then it must use an undefined-length burst to perform the remaining three transfers. Figure 16-18 shows incrementing bursts of undefined length that starts after aborting previous INCR 4 burst access.

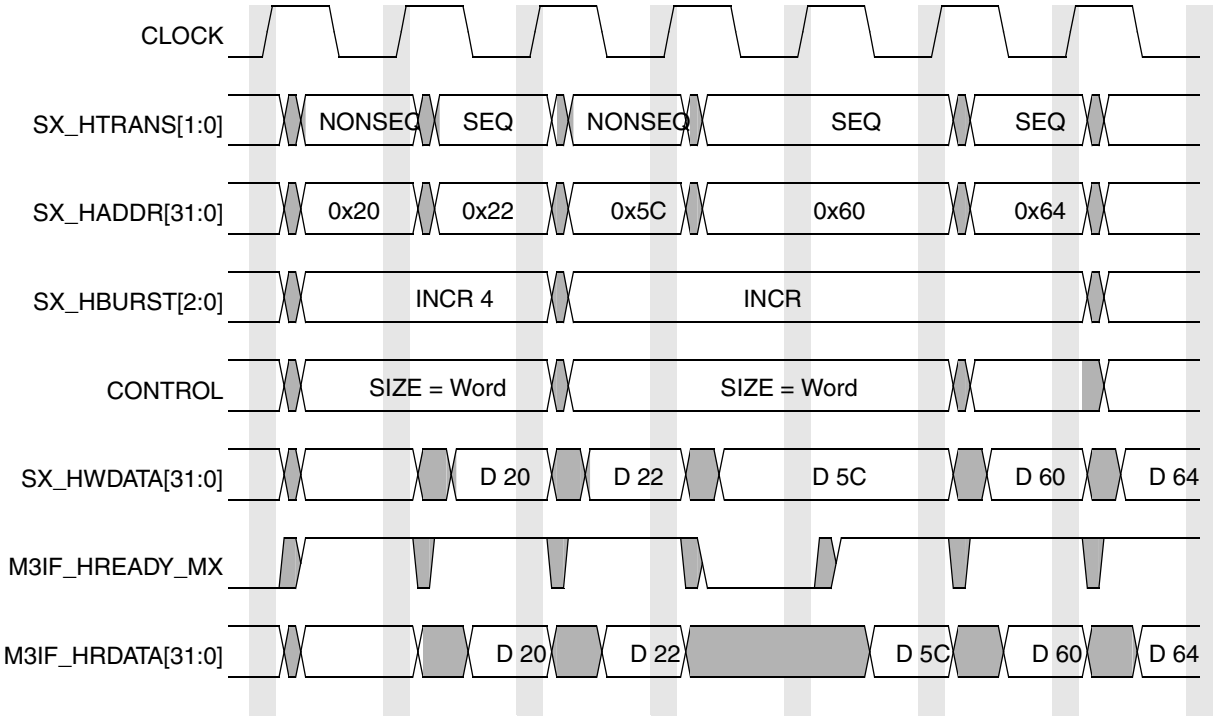


Figure 16-18. MPG Undefined Length Bursts

NOTE

To perform burst access length not equal to 4 or 8 words, it is possible to start INCR access of undefined length and to abort it after the desired words, or to start 4/8 burst length access and to abort it after the desired word., Both ways are supported by the M3IF and it is the master's decision which way to choose.

16.4.1.7 Multi-Endianness

M3IF supports multi-endianness, an example of this is there is an endian signal input to each one of the MPGs. The endianness signal from each master should be static after reset, means that all accesses from each master will have the same endianness type. If in a given system, there are masters connected to the M3IF that does not drive endian signal (means they support only one endian type, big or little) the respective MPGs big end signal should be static, meaning connected to 0 or 1 (depends on the endianness supported by the master connected to it). M3IF does NOT support shared external memory area for masters with different endianness mode. This feature should be handled by software or other additional hardware in the system.

16.4.2 Master Port Gasket 64 (MPG64)

16.4.2.1 Overview

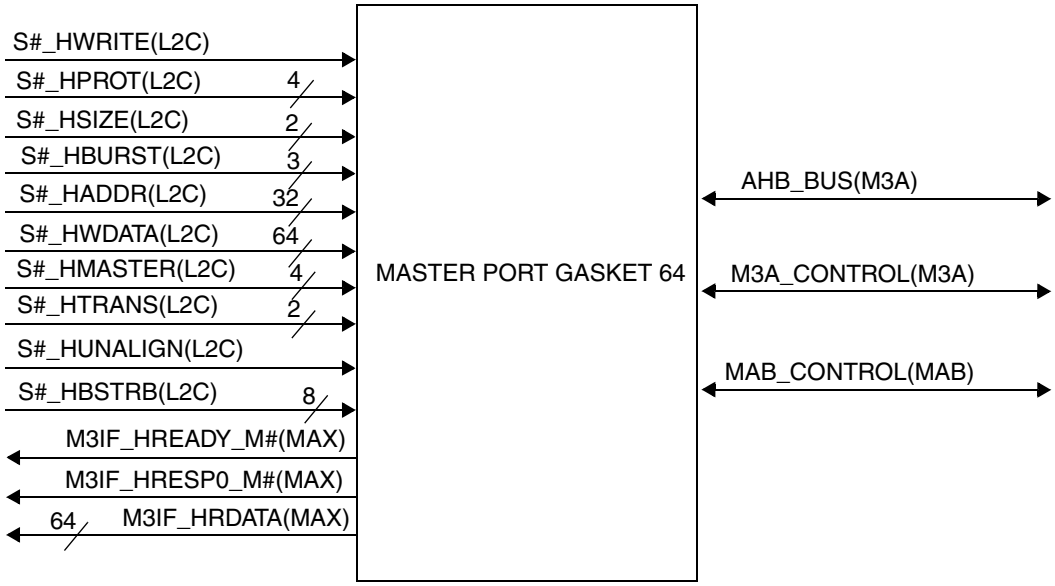
MPG64 port gasket is used for those system masters that have 64 bits data bus.Table 16-15 presents the access types supported by the MPG. Figure 16-19 shows the MPG64 port interface diagram. MPG64 does not support RETRY and SPLIT transfers.

Table 16-15. MPG64 Supported Burst Accesses

HBURST	TYPE	M3IF slaves							
		ESDCTL		EIM		NFC		PCMCIA	
		32 Bit	64 Bit	32 Bit	64 Bit	16/32 Bit	64 Bit	8/16 Bit	64 Bit
000	SINGLE	YES	YES	YES[3]	YES	YES	YES	YES	NO
001	INCR	YES	YES	YES	YES	YES	YES	YES	NO
010	WRAP 4	YES	YES	YES	YES	NO	NO	YES	NO
011	INCR 4	YES	YES	YES	YES	YES	YES	YES	NO
100	WRAP 8	YES	NO	YES	NO	NO	NO	YES	NO
101	INCR 8	YES	YES	YES	YES	YES	YES	YES	NO
110	WRAP 16	NO	NO	YES	NO	NO	NO	YES	NO
111	INCR 16	NO	NO	YES	NO	YES	NO	YES	NO

1. NFC does not support accesses of 8 bit data width.
2. PCMCIA does not support accesses of 32 or 64 bit data width.
3. M3IF MPG64 supports only double word (64 bits) or word (32 bits) size bursts. Since SINGLE access is not a burst type access, byte (8 bits) or half word (16 bits) is supported as well.

For MPG64 brief overview description, see Section 16.4.1.1, “Overview of MPG Operation.” Table 16-17 only shows the buses that have different widths, all other signals are the same as described in Table 16-15.



L2C—Layer 2 Cache
MAB—Master Arbitrator and Buffering
S#—L2C port number
M#—M3IF Master port number (from 0 to 8)

Figure 16-19. MPG64 Port Interface Diagram

Table 16-17. MPG64 Additional Signals

64-Bit Master—Signal Name	MPG64—Signal Name	Description
S#_HBSTRB[7:0] (O)	M3IF_HBSTRB_M#[7:0] (I)	Indicates which byte lanes are valid (each bit for each byte)—extended to 8 bits.
S#_HWDATA[63:0] (o)	M3IF_HWDATA_M#[63:0] (I)	The write data bus extended to 64 bit width
S#_HSIZE[1:0] (O)	M3IF_HSIZE_M#[1:0] (I)	Indicates the size of the transfer 00 8 bits (Byte) 01 16 bits (Halfword) 10 32 bits (Word) 11 64 bits (Double-Word) (defined only for MPG64)
S#_HBURST[2:0] (O)	M3IF_HBURST_M#[2:0] (I)	Burst information is provided using HBURST signal, and the 8 possible types are; 000 SINGLE (Single transfer) 001 INCR (Incrementing burst of unspecified length) 010 WRAP4 (4-beat wrapping burst) 011 INCR4 (4-beat incrementing burst) 100 WRAP8 (8-beat wrapping burst) 101 INCR8 (8-beat incrementing burst) 110 WRAP16 (16-beat wrapping burst) 111 INCR16 (16-beat incrementing burst)
S#_HRDATA[63:0] (I)	M3IF_HRDATA_M#[63:0] (O)	The read data bus extended to 64 bit width

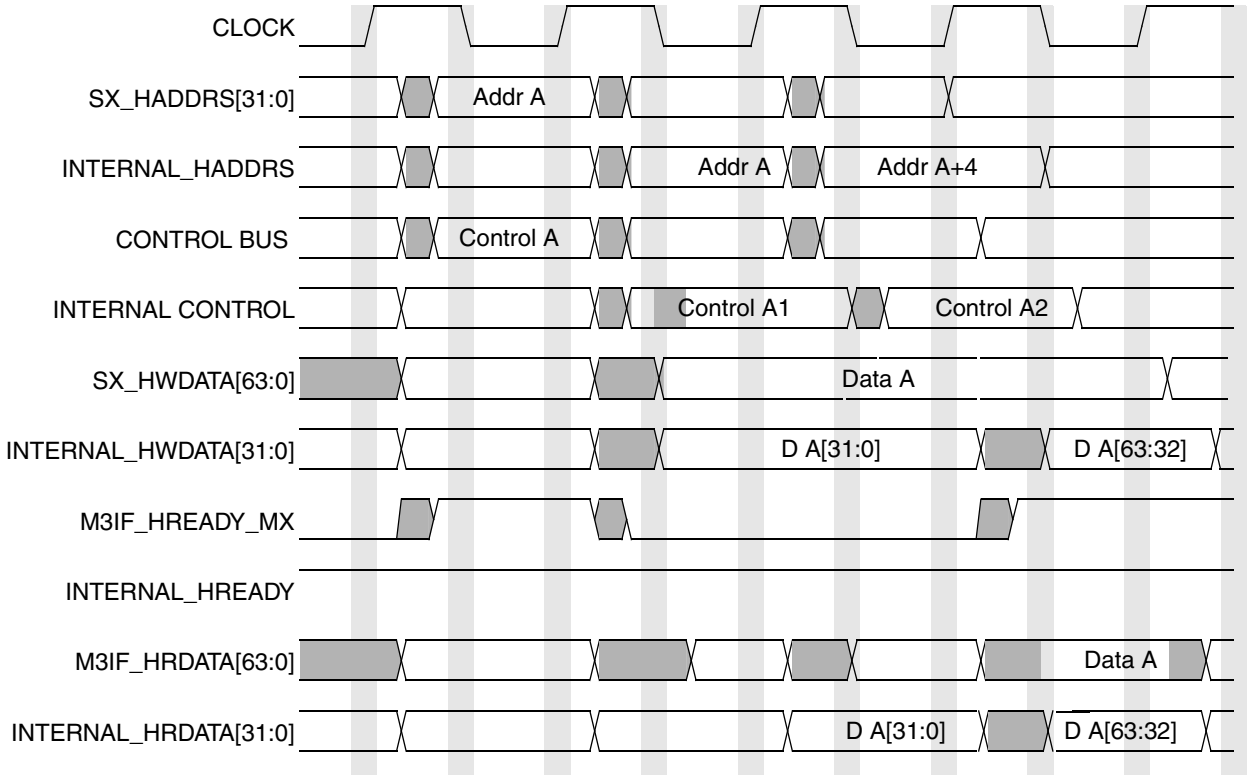
16.4.2.2 MPG64 Basic Transfer

All Basic transfer for 32 bits access are perform as AMBA-AHB usual access as described in 16.4.1.2, “MPG Basic Transfer.” All 64 bits access are performed differently. Since the output data port is 32 bits wide, each 64 bit (double word) access is translated into 2 separated access, so a single read/write access of 64 bits is translated by the MPG64 into two single read/write 32 bits access. Burst length of 4 double words is being translated into 8 words (32 bits) burst length and 8 double words burst length is translated into 2 bursts of 8 words (32 bits) length. For 32 bit MDDR there is no need for the MPG64 gasket to translate the access since 64 bits can be transferred by the MDDR each cycle.

Figure 16-20 shows the simplest transfer of a single double word, one data with two wait states (one cycle translation and one cycle per two single access of 32 bits access should be perform).

- The master drives the address and control signals onto the bus after the rising edge of the clock.
- M3IF then samples the address and control information and translate it to 2 separated single access of 32 bits.
- First and then second access is performed on the SDRAM.
- Each single access is treated as described in Section 16.4.1.2, “MPG Basic Transfer.”
- The response that the 64 bit master sees is similar to AMBA-AHB response.

Figure 16-20. MPG64 Simple Double Word Transfer



16.4.2.3 MPG64 Transfer Type

See Section 16.4.1.3, “MPG Transfer Type.”

16.4.2.4 Mpg64 Transfer Response

See Section 16.4.1.4, “MPG Transfer Response.”

16.4.2.5 MPG64 Burst Operation

There are few things different than what is described in Section 16.4.1.5, “MPG Burst Operation.” The size of each beat can be either 8, 16, 32 or 64-bits (byte/half word/word/double word), as shown in Table 16-15. If a burst access of double word is issued, WRAP8, INCR16 and WRAP16 are not supported by the M3IF.

Eight beat incremental burst of double word beat size is translated into 2 burst of 8 words length each, as shown in Figure 16-21 (for 32 bit MDDR there is no need for the MPG64 gasket to translate the access).

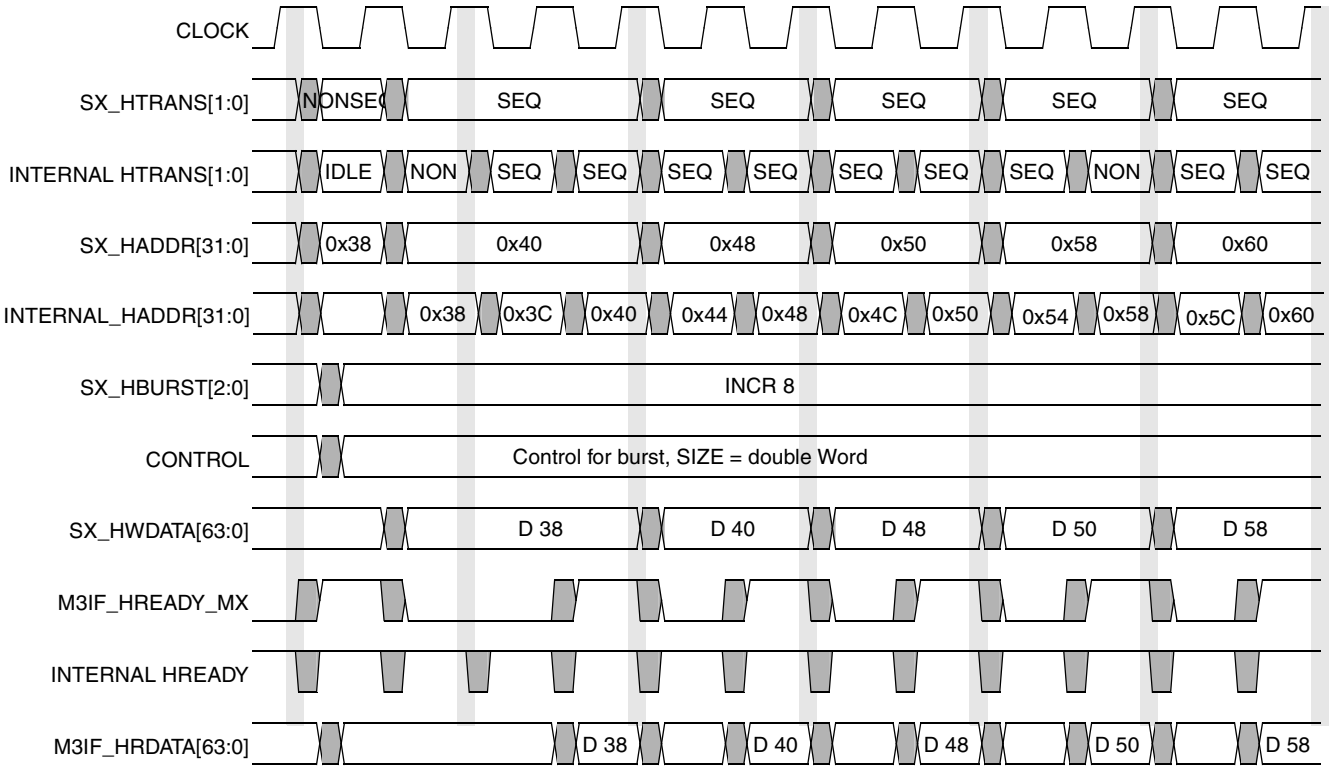


Figure 16-21. MPG64—8 Beat Incremental Burst of Double Words to 32-bit SDRAM

16.4.2.6 MPG64 Early Burst Termination

See Section 16.4.1.6, “MPG Early Burst Termination.”

16.4.2.7 Multi Endianness

See Section 16.4.1.7, “Multi-Endianness.”

16.4.3 M3IF Arbitration (M3A)

16.4.3.1 Overview

M3A is a programmable arbiter. All incoming requests from the different masters are on hold until access is granted by the arbiter. The arbitration is performed by a round robin algorithm which grants the access to the master that holds the token. In the case that a master holds the token but does not request access (to one of the M3IF slaves), the bus is granted to the nearest requesting master with a higher round robin number.

The internal signal bus free indicates the FF1 algorithm to choose a new master. The bus_free signal is asserted high by the M3A by monitoring HTRANS bus of the active master (the master that grants access). As soon as the M3A notices that the access has been accomplished (HTRANS equal to NONSEQ or IDLE with HREADY asserted high) it allows to a new master to gain access according to the round robin value. When a new master gain an access the M3A will transfer the AHB bus coming from this master to all memory controllers with an hsel signal to the specific memory controller, (all other will get low hsel).

Because MAB can get multiple access to the ESDCTL and pass accesses to the ESDCTL according to internal handshake between the ESDCTL and the MAB, the M3A will allow multiple access to pass to the MAB without waiting for previous accesses to be completed.

The M3A will pass the request to the MAB if the access is to the ESDCTL, and will pass a new access to the MAB (before the previous/active master access is completed) if the master with the token is accessing the ESDCTL If the request is for a different memory controller the M3A will hold the access until all pending transfers in the MAB (ESDCTL accesses) are finished. In this case (multiple access to MAB are in progress) the bus_free signal will assert high when the MAB completes all incoming requests.

There two different access paths:

1. Access request to SDRAM/MDDR—involves ESDCTL/MDDRC memory controller. The access path is as follows:
   a) Master #x initiates access to SDRAM/MDDR memory —> M#_ESDCTL_REQ signal is high.
   b) M3A arbitrates master request.
   c) After successful arbitration M3A passes the request to MAB (by MASTER_REQ_EN signal set to 1).
   d) MAB and the respective MPG (the one that initiated the access) are handling the access directly from/to the ESDCTL/MDDRC (M3A is not involved).
   e) All data transfer is accomplished by the ESDCTL/MDDRC and the SDRAM/MDDR external memory.
2. Access request not to SDRAM/MDDR memory—involves the respective memory controller. The access path is as follows:
   a) Master #x initiates access not to SDRAM/MDDR memory —> M#_GENERAL_REQ signal is high.
   b) M3A arbitrates master request.
   c) After successful arbitration M3A passes the AHB bus (address, data, control signals) of the respective master to the relevant memory controller.

d) M3A and the respective MPG (the one that initiated the access) are handling the access from/to the relevant memory controller (MAB is not involved).

e) All data transfer is accomplished by the relevant memory controller and the external memory.

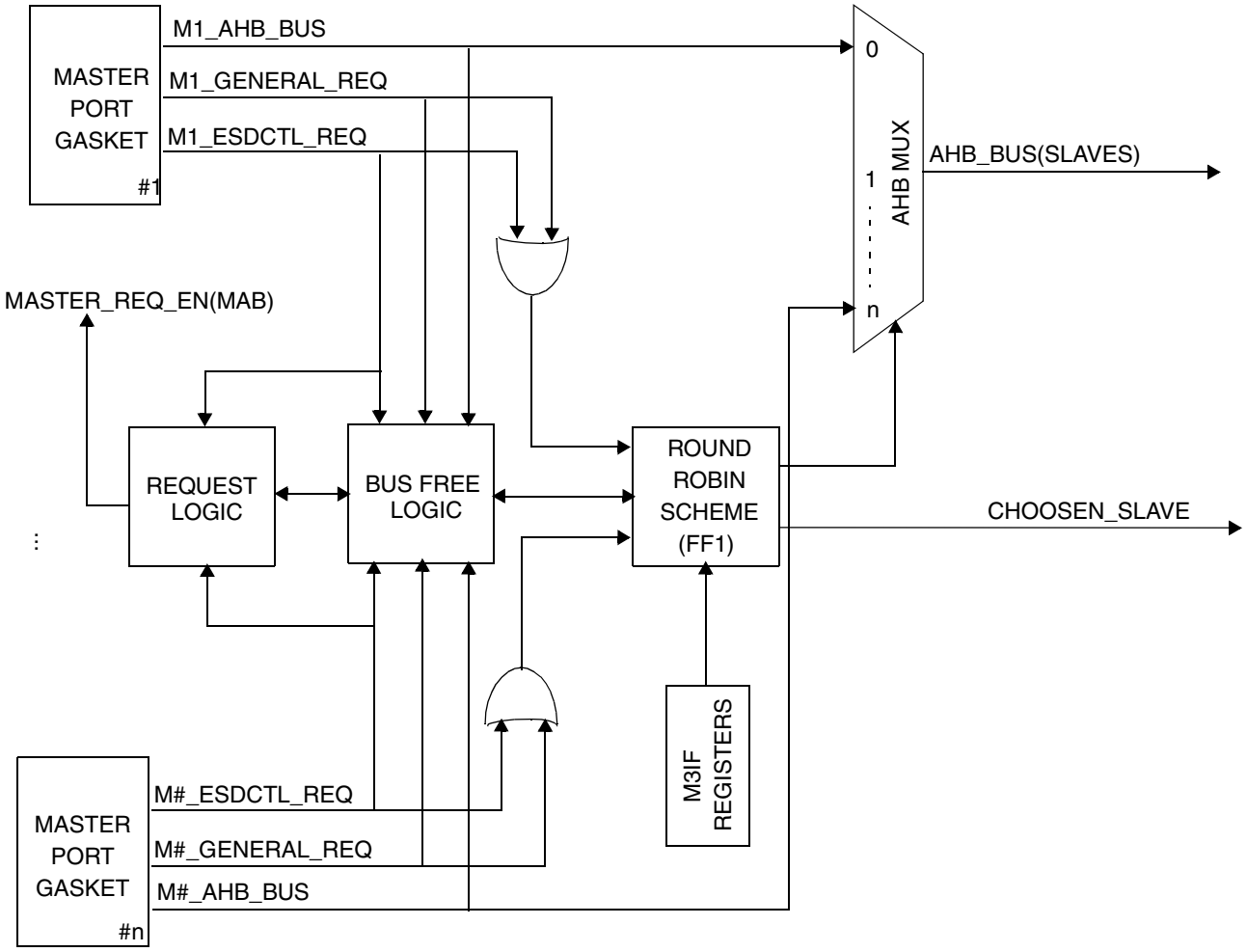


Figure 16-22. M3A Block Diagram

NOTE

Although AHB bus indicates only one direction, the round robin scheme DEMUX the AHB response signals (hready, hresp, and hrdata) from all slaves (except from ESDCTL that goes directly to the MPG).

Figure 16-23 illustrates a simple transfer (all previous accesses are completed, and there are no other pending requests) between one of the M3IF masters and the EIM module. There is one cycle penalty at the beginning of the access. The MPG samples the access relevant signals and de-asserts HREADY signal toward the master, until the target slave confirms the access (EIM_HREADY high with EIM_NONSEQ cycle). After the target slave (EIM in this example) confirms the request (HREADY high with EIM_NONSEQ cycle) the access traffic (control and data) is direct between the master and the target slave (EIM).

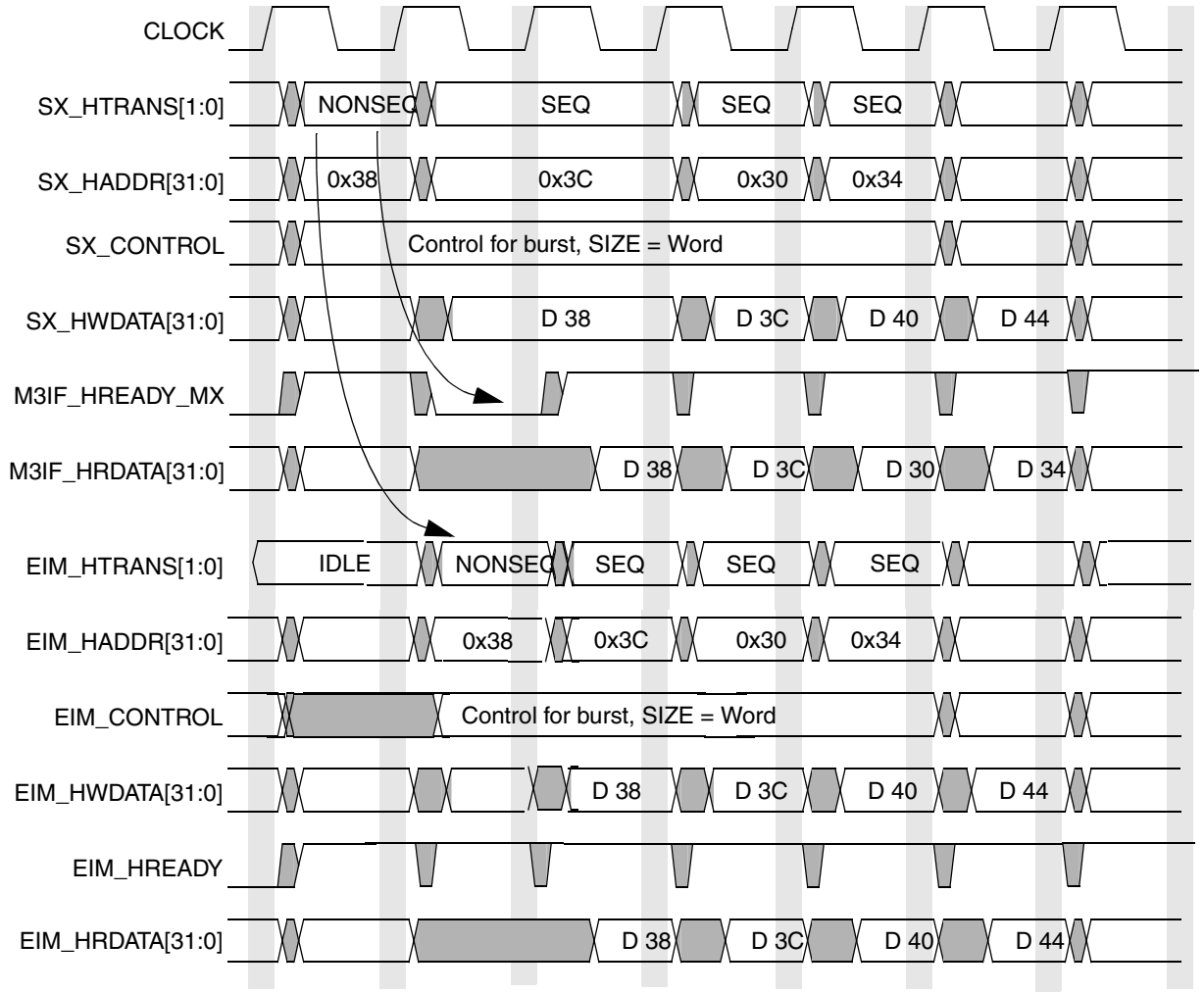


Figure 16-23. M3A Simple Transfer Timing Diagram

16.4.3.2 M3A–Find First 1 (FF1) Algorithm

Arbitration between the various requests is done by Find First 1 algorithm based on round robin algorithm (see Figure 16-24). If two or more masters request access to the M3IF the master with the token, or the master that will be the first to receive the token (in case the master with the token does not request access) will gain control over the AHB bus or will enable his request signal to the MAB. For example, if masters 0, 1, and 6 requesting access at the same time and the token is at master 2, then master 6 will gain control over the AHB bus or his request signal to the MAB will be enabled, since it will be the first to receive the token (this is done to reduce arbitration time, in this example 4 clock cycles are saved). The Round Robin pointer increase its value in two cases, if the master with the token does not request access or if the master with the token requests access and gains the AHB bus/enable request—on the cycle that the master gains the AHB bus/enable request the Round Robin pointer will increase its value. If a master requests an access and has the token but does not gain access because no new access can pass on, the Round Robin will not increase until the master gains the AHB bus/enable request.

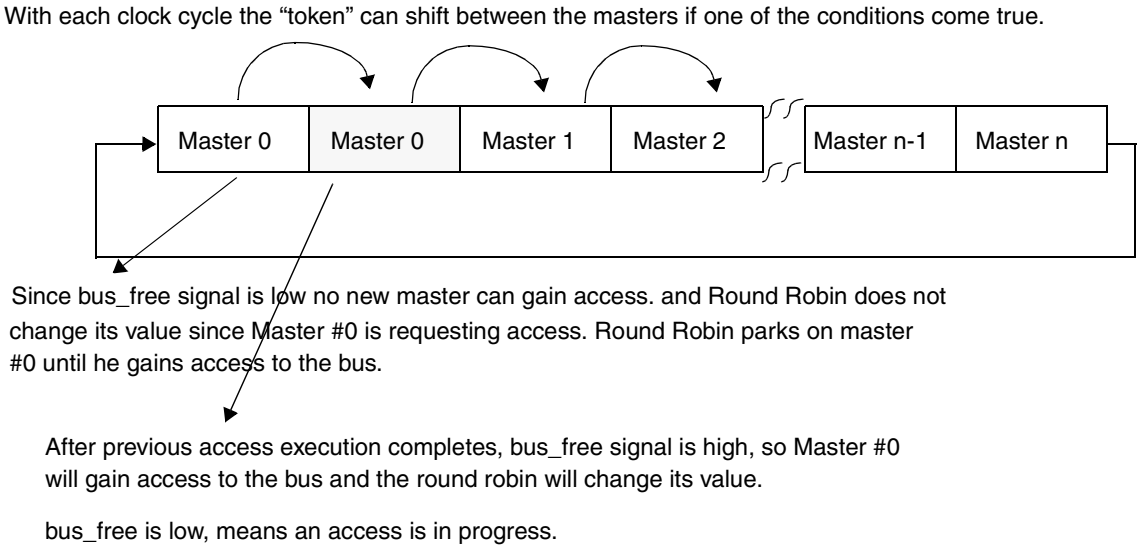


Figure 16-24. M3A—Round Robin Token Chain—Equal Priority

There is an option to program the Round Robin to work with different priority using MRRP field of the M3IF control register. When the MRRP equal to 0 no priority is given to any master so the probability to gain access is equal for each one of the masters. If one or more bits of the MRRP are set to 1, the priority changes, and all master with their bit set to 1, will get together 50% priority of gaining access. For example, if both master 1 and master 4 bits set to 1 in the MRRP field, the priority to gain access of master 1 and 4 together will be 50%. If only one bit is set the respective master (that his bit is set) will alone have 50% priority to gain access.

NOTE

By default (after reset) all MRRP bits are cleared, means that all M3IF masters have the same priority.

The 50% priority refers only to Round Robin mechanism. If a non priority master (MRRP respective bit is not set) with the token is already waiting to gain the bus the new coming request from the priority master (MRRP respective bit is set) will not gain access before the previous master request is completed. Additionally, priority master cannot and will not terminate an on going access of any other masters. Figure 16-25 shows Round Robin chain in case that MRRP configured to 8’b00010010–master 1 and 4 set to 1.

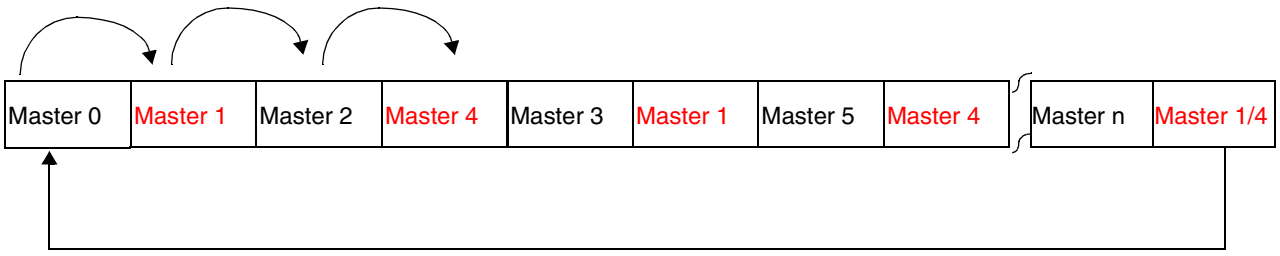


Figure 16-25. M3A—Round Robin Token Chain—Masters 1 and 4 has 50% Priority

16.4.3.3 Bus_free Signal Algorithm

When the bus_free signal asserts high, it indicates the new master can gain access. The bus_free asserts high in the following cases:

- When the previous access was a NON ESDCTL access and the HTRANS bus of the previous master that gained the access was equal to NON SEQUENTIAL or IDLE and HREADY asserted high.
- When the previous access was an ESDCTL access and the new master that gained access was also an ESDCTL access.
- When the previous access or accesses were ESDCTL accesses and all previous accesses finished.

NOTE

To avoid contention between the memory controllers/memories, there is a special signal coming from the MPG and from the MAB to the M3A and going to the bus_free algorithm indicating which kind of slave is still using shared I/O pins so no new access to a different memory begins.

16.4.4 Master Arbitration and Buffering (MAB)

16.4.4.1 Overview of MAB Operation

MAB arbiter uses the same programmable arbiter as the M3A (Section 16.4.3.2, "M3A–Find First 1 (FF1) Algorithm"). All incoming requests from the different masters are put on hold until access is granted by the arbiter. The MAB communicates with the ESDCTL and grants access at the earliest possible time, for example when the ESDCTL is ready to handle a new memory request. Each time ESDCTL can get a new access, the new access is sampled into the CONTROL and DATA buffers so ESDCTL receives stable inputs during the time the access is in progress. The DATA buffer is sampled according to ESDCTL write acknowledge response. Figure 16-26 shows MAB operation block diagram.

16.4.4.2 M3B–Find First 1 (FF1) Algorithm

This operation of the arbiter algorithm is identical to the M3A arbiter algorithm described in Section 16.4.3.2, "M3A–Find First 1 (FF1) Algorithm."

Each time NEW_ACCESS goes HIGH it indicates that the ESDCTL is ready to handle a new memory request, meaning that the previous request is either completed or controlled by the SDRAM memory. During the same cycle the memory port is granted to the master with the token. Figure 16-27 shows the arbitration process for 4 master requests. At the first clock cycle masters M0, M1, and M2 simultaneously request the memory port. At the rising edge of the clock (while NEW_ACCESS is HIGH) master M0 has the token, so the memory port is controlled by M0 (see MASTER_CONTROL signals). During that time all M3IF_HREADY_M# signals are low, besides M3IF_HREADY_M3 which was the previous served master.

The same arbitration process occurs for the following requests. Master M1 has the token and grant access to the ESDCTL (see MASTER_CONTROL). HREADY of master M0 asserted high and accomplished the previous access. After several cycles M0 assert the REQUEST signal again due to a new access.

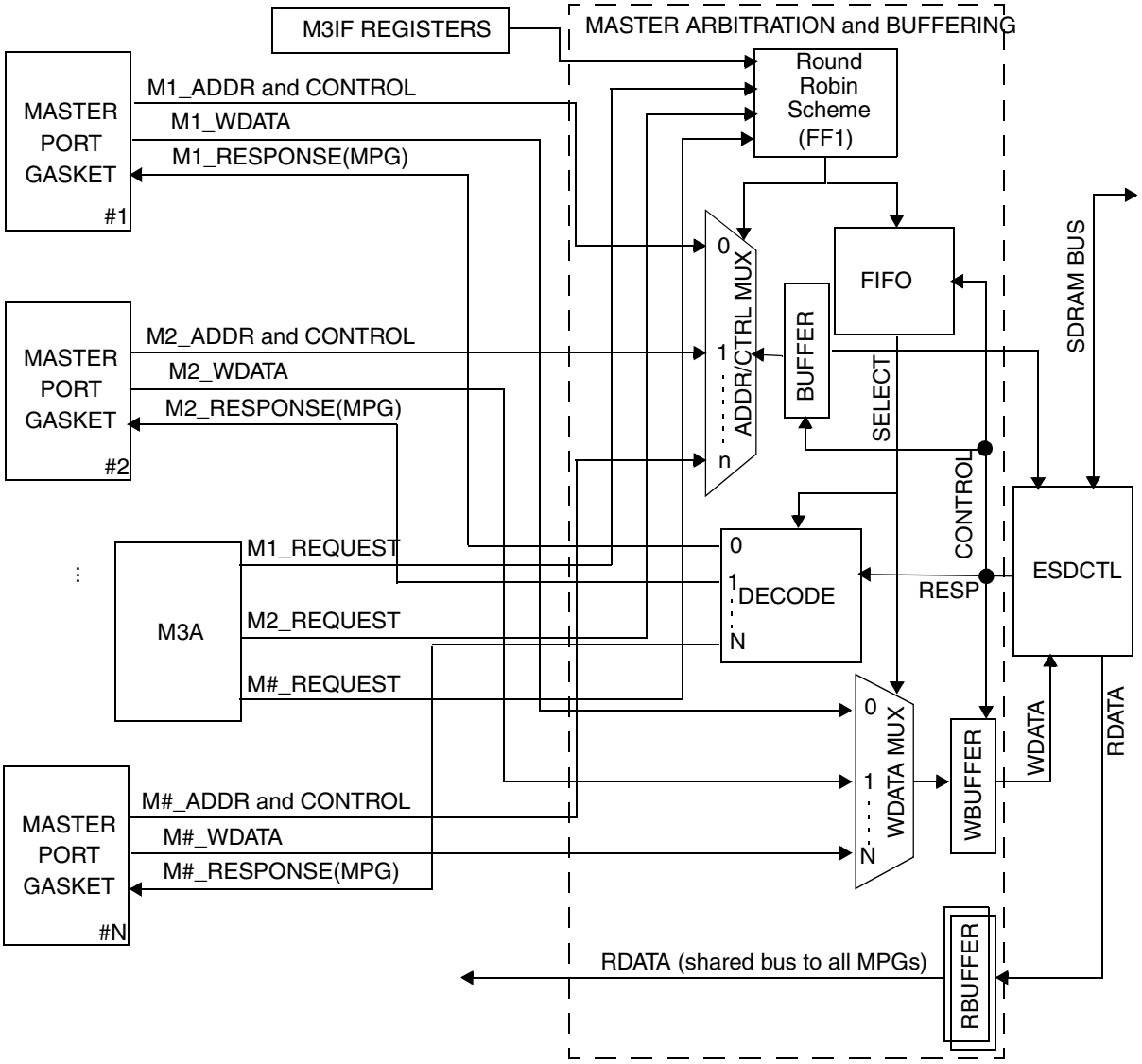


Figure 16-26. MAB Overview Block Diagram

Once a master has control over the port, the other requests remain on hold, meaning that their M3IF_HREADY_M# is LOW. The current master has control over the memory port until it completes the requested transfer. The new master gaining access to the memory port is the master with the new token.

The MAB ADDR/CTRL MUX and WDATA MUX connect (using the round robin algorithm) the selected master and when NEW_ACCESS arrives the MUX output is sampled by the MAB so that a stable bus is provided to ESDCTL. After a new access is detected by the ESDCTL, data transfer between the memory and the masters can start. In order for the MAB to serve more than one master at a time, a cyclic 4 entry FIFO with two pointers (read/write) is used. The FIFO read pointer is used (by the Decode block) as the selector for the memory RESPONSE signals and for the WDATA MUX, while the FIFO write pointer is used to add a new master to the FIFO entries. (since ESDCTL hides latency a new access can start before previous access ended. This why this MUX control should work separately for information to the ESDCTL and ESDCTL response. Figure 16-27 shows the MAB arbitration process timing diagram.

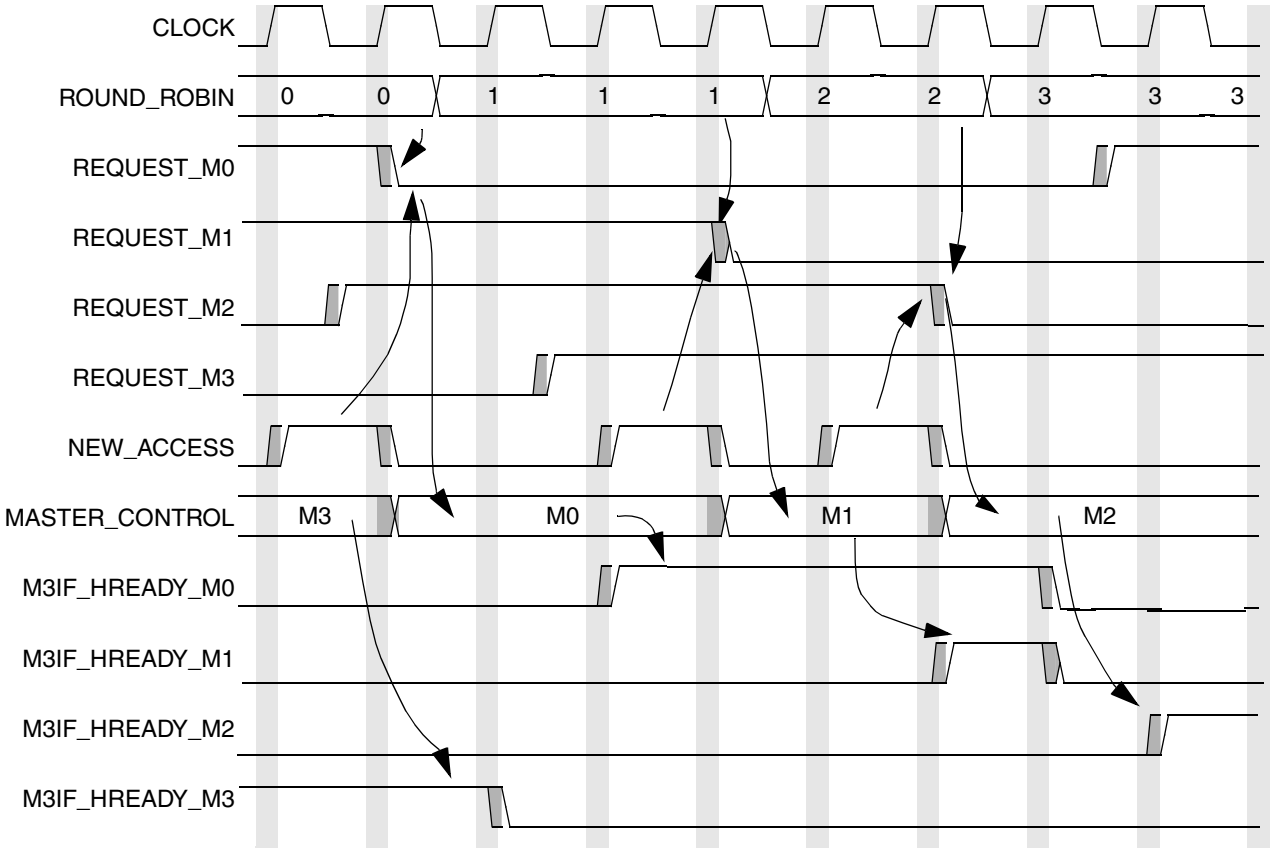


Figure 16-27. MAB Arbitration Process Timing Diagram

Figure 16-28 shows a detailed multi master memory request time diagram. The diagram shows two masters presenting memory request (AMBA AHB) signals and their conversion by the MPG and MAB to the ESDCTL. The HRDATA timing is also shown with the respective M3IF_HREADY_M# signal. The HRDATA bus from the memory (during READ transfers) is shared with all present masters (except 64-bit masters that become a 64-bit bus after decoding by MPG64) while the arbitration is completed by the use of the M3IF_HREADY_M# signals at the master level.

16.4.4.3 M3IF Operation During HMASTLOCK Accesses

If a HMASTLOCK access to any memory controller (memory space) is initiated by one of the M3IF masters, the request need to pass the arbitration like a regular access. After the access passes the arbitration it “locks” the arbitration and all other accesses (regardless to memory space destination) will remain pending until the HAMSTLOCK signal (from the master that initiated the HMASTLOCK access) de-asserts. While the HMASTLOCK is high all accesses initiated by the locking master will be executed without arbitration, while all other masters accesses will remain pending.

NOTE

During HMASTLOCK high, the locking master is NOT ALLOWED to change the memory space destination, from SDR/DDR SDRAM space to non SDR/DDR SDRAM, or vice versa.

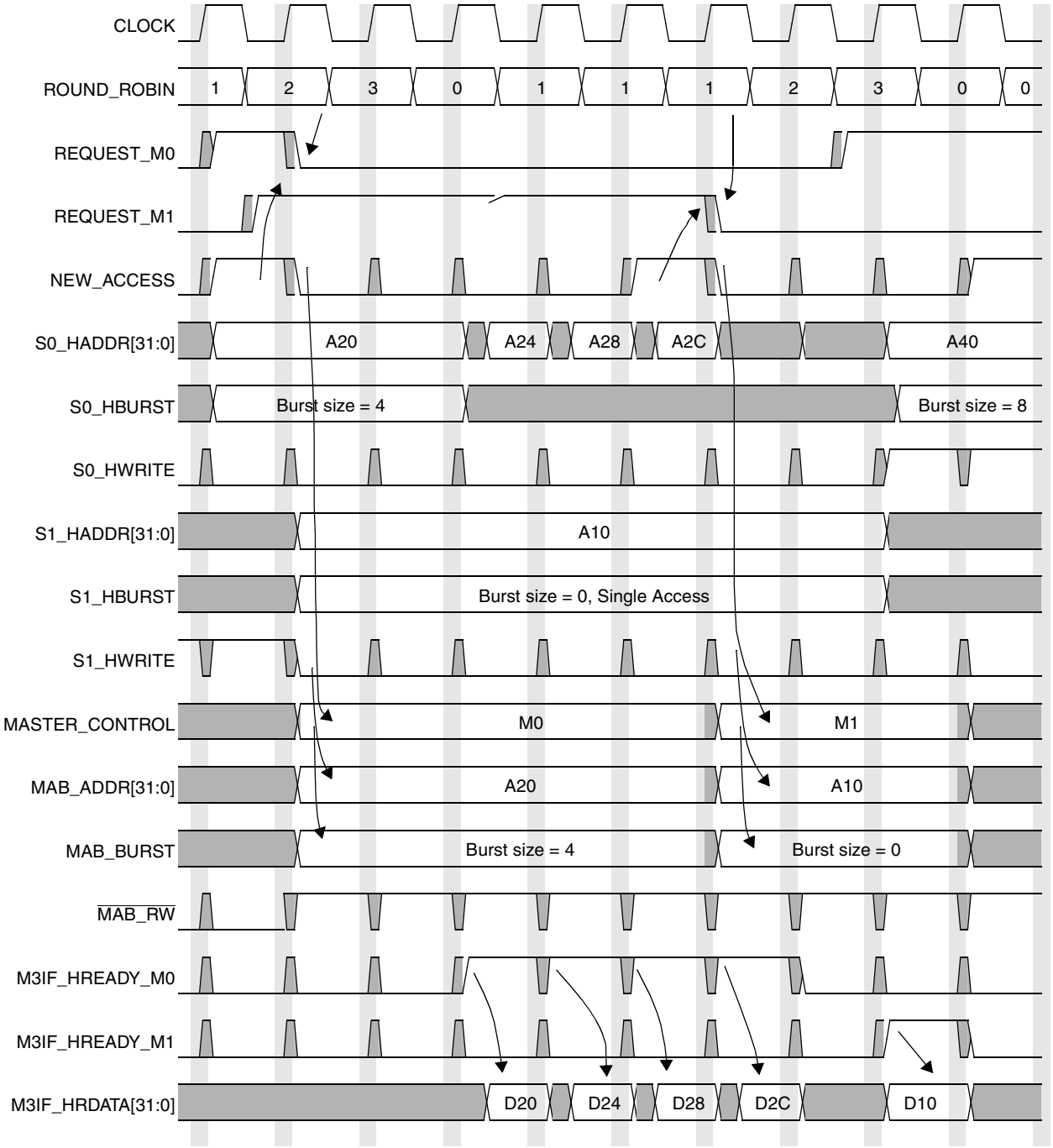


Figure 16-28. MAB Multi-Master Request Time Diagram

16.4.5 Snooping Logic

M3IF snooping feature (used by the Image Processor Unit, IPU), monitors and detects write accesses to a configurable window (snooping window) in one of the memory regions mapped by the M3IF. The snooping window base address, memory region, and window size are configurable parameters through the M3IFSCFG0 register (register details at Section 16.3.3.2, “M3IF Snooping Configuration Register 0

(M3IFSCFG0)”). Snooping window is further divided into 64 equally sized segments. A detected write access to the snooping window results in:

- The respective segment status bit in M3IFSSR0 and/or M3IFSSR1 registers is set.
- DMA_ACCESS strobe is asserted for 1 clock cycle if the snooping segment enable bit is set for the snooped segment. The snooping segment enable bit is configured via 2 snooping configuration registers, M3IFSCFG1 and M3IFSCFG2.

It is the software’s responsibility to clear the snooped segment status bits, but the snooped segment status bit will be set for each snooping detection regardless of its value.

16.5 Initialization/Application Information

16.5.1 M3IF in a System

This section provides an example of M3IF initialization, integration and configuration in a given system. The system requirements are listed below:

- Several masters with external memories access capabilities.
  - Several masters access the M3IF via AP MAX crossbar switch.
  - ARM I-Cache—32-bit data bus
  - ARM D-Cache—32-bit data bus
- The system uses 2 SDRAM memory devices (32 bits), a 32-bit Flash (via WEIM CS0) and one SRAM (via WEIM CS1).

Figure 16-29 presents M3IF integration in the system. All the above masters are connected to the M3IF through ports.

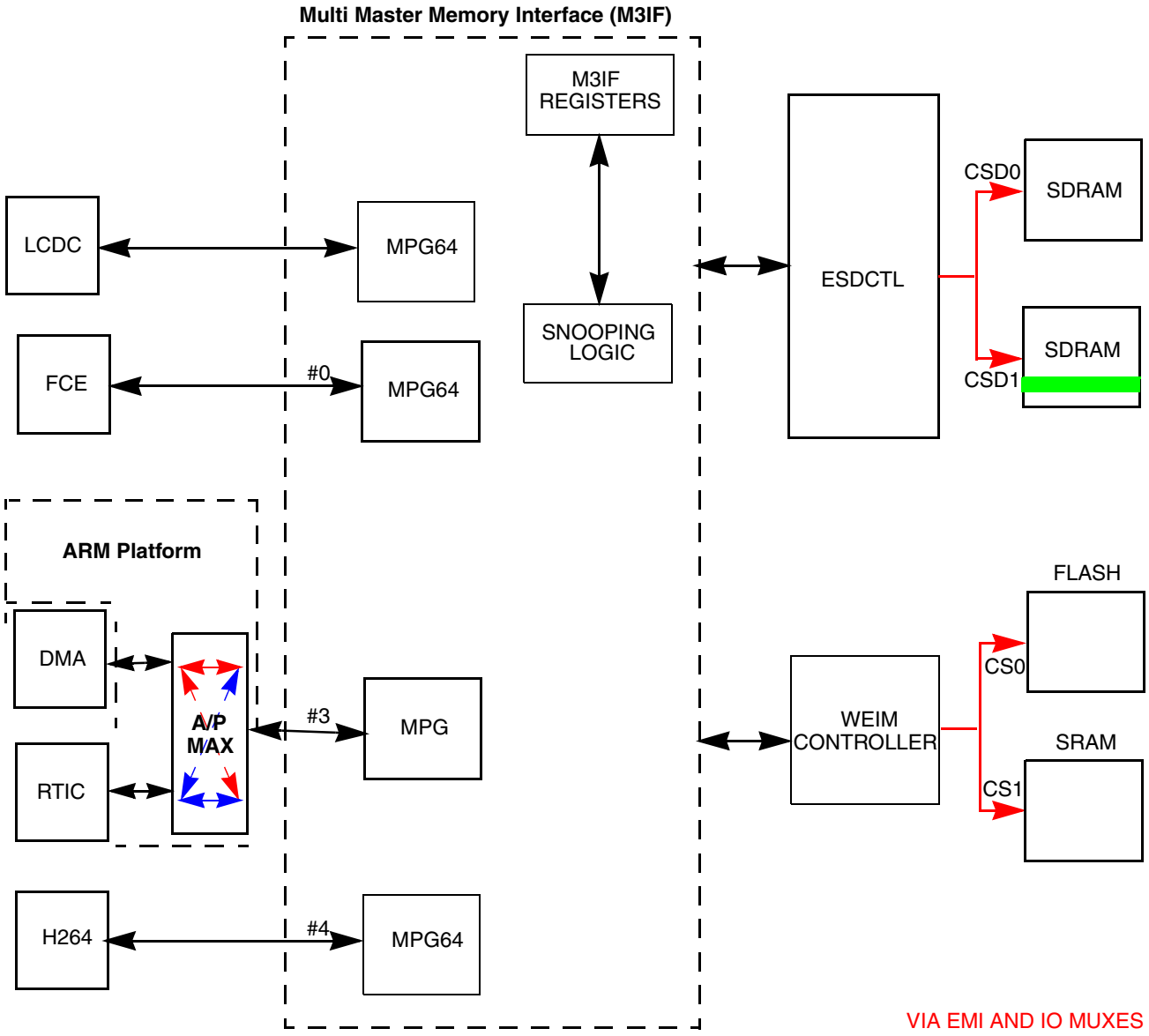


Figure 16-29. M3IF System Example

Chapter 17
Wireless External Interface Module (WEIM)

The Wireless External Interface Module (WEIM) handles interface to devices external to the chip, including generation of chip selects, clock and control for external peripherals and memory. It provides asynchronous and synchronous access to devices with SRAM-like interface. Figure 17-1 shows a top-level WEIM block diagram. All block signals are shown in Table 17-1 and described in Table 17-2.

17.1 Features

- Six Chip Selects for external devices, with $\overline{CS0}$ and $\overline{CS1}$ each covering a range of 128 Mbytes, and $\overline{CS2}$–$\overline{CS5}$, each covering a range of 32 Mbytes
- $\overline{CS0}$ range can be increased to 256 Mbytes when collapsed with $\overline{CS1}$
- Selectable Protection for each Chip Select
- Programmable Data Port Size for each Chip Select
- Asynchronous accesses with programmable setup and hold times for control signals
- Synchronous Memory Burst Read Mode support for AMD, Intel, and Micron burst flash memory
- Synchronous Memory Burst Write Mode support for PSRAM (CellularRAM™ from Micron, Infineon, and Cypress)
- Support for multiplexed address/data bus operation
- External cycle termination/postpone with $\overline{DTACK}$ signal
- Programmable Wait-State generator for each Chip Select
- Support for Big Endian and Little Endian modes of operation per access
- ARM AHB slave interface

17.2 Overview

WEIM has six modes of operation. WEIM does not require a dedicated low-power mode because most of clocks are gated anytime when there are no accesses to WEIM providing maximum energy conservation.

- Asynchronous mode. This is a non-burst mode is used for SRAM access. In this mode a single data is read/written with each access (asserted address). All controls timings are controlled by preset values in Chip Select control registers.
- Synchronous read mode. This is a burst mode is used for reading Flash memory devices. In this mode after address assertion, a burst of sequential data can be read. Data exchange is carried out according to BCLK clock that is generated by WEIM. An access may be delayed by external $\overline{ECB}$ signal assertion after first word of data.

- Page mode. This mode is used for memory burst read, but address is asserted for each data in the burst as $\overline{\text{LBA}}$ and BCLK operate asynchronously. In this mode, setup time is greater for first data burst than for the remaining data burst (in the same page).

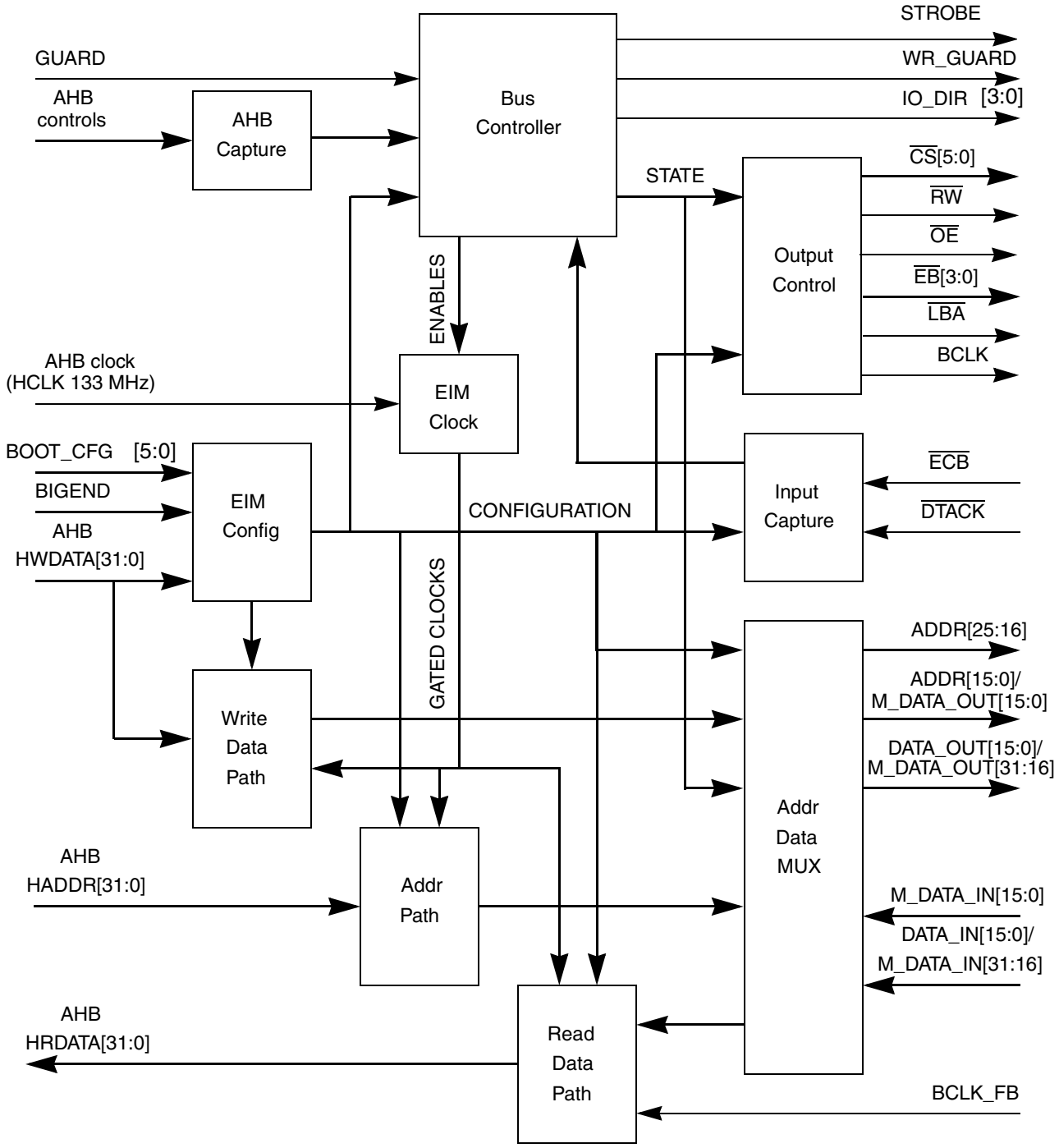


Figure 17-1. WEIM Block Diagram

- Synchronous read/write mode (PSRAM synchronous mode). In this mode, read and write are synchronous. Access may be additionally delayed according to $\overline{\text{ECB}}$ state before first piece of data arrives (refresh wait enable).
- DTACK mode. This is a non-burst mode used for PCMCIA access. In this mode, WEIM waits for $\overline{\text{DTACK}}$ acknowledge until 1024 counts of AHB clock have passed. In this mode, $\overline{\text{DTACK}}$ can be used as posedge or level sensitive according to WSC field and EW bit settings.
- Multiplexed Address/Data mode. In this mode, multiplexing addresses and data bits on same pins is supported for synchronous/asynchronous accesses to the 32-bit data width memory devices.

17.3 External Signal Description

This section discusses input and output signals (see Table 17-1) between WEIM and external devices.

Table 17-1. Signal Properties

Name	Port	Function	Direction	Reset State
ADDR[25:16]	—	Address Bus MSB/$\overline{\text{EB}}$[3:2], CRE	Out	Low
ADDR[15:0]/ M_DATA_OUT[15:0]	—	Address Bus LSB/Output Data Multiplexed Bus LSB	Out	Low
BCLK	—	Burst Clock	Out	Low
BCLK_FB	—	Feedback Burst Clock	In	—
BIGEND	—	Data Endian	In	—
BOOT_CFG[5:0]	—	Boot Configuration	In	—
$\overline{\text{CS}}$[5:0]	—	Chip Selects	Out	High
DATA_IN[15:0]/ M_DATA_IN[31:16]	—	Input Data Bus LSB/Input Data Multiplexed Bus MSB	In	—
DATA_OUT[15:0]/ M_DATA_OUT[31:16]	—	Output Data Bus LSB/Output Data Multiplexed Bus MSB	Out	Low
$\overline{\text{DTACK}}$	—	Data Transfer Acknowledge/Wait	In	—
$\overline{\text{EB}}$[3:0]	—	Enable Byte	Out	High
$\overline{\text{ECB}}$	—	End Current Burst/Wait	In	—
GUARD	—	Input Guard	In	—
IO_DIR[3:0]	—	IO Direction	Out	Low
$\overline{\text{LBA}}$	—	Load Burst Address	Out	High
M_DATA_IN[15:0]	—	Input Data Multiplexed Bus LSB	In	—
$\overline{\text{OE}}$	—	Output Enable	Out	High
$\overline{\text{RW}}$	—	Read/Write	Out	High
WR_GUARD	—	Write Guard	Out	High

17.4 Detailed Signal Descriptions

Table 17-2. WEIM Detailed Signal Descriptions

Signal	I/O	Description
$\overline{\text{ADDR}}$[25:16]	IO	Address Bus MSB. These pins are used as address bits [25:16]. In the multiplexed mode these pins do not change their state. If corresponding AUSx bit is set, those pins reflect [25:16] AHB address bits. If AUSx bit is not set, these pins represent [27:18] AHB address bits for word-width memory, [26:17] bits for halfword width memory and [25:16] bits for byte-width memory. ADDR[25:24] also are used as $\overline{\text{EB}}$[3:2] in the multiplexed mode (MUM=1). ADDR[23] also is used as CRE in PSRAM mode (PSR=1).
ADDR[15:0]/ M_DATA_OUT[15:0]	IO	Multiplexed Address Bus LSB/Output Data Bus LSB. In non-multiplexed mode those pins are used as address bits [15:0]. If corresponding AUSx bit is set those pins reflect [15:0] AHB address bits. If AUSx bit is not set, those pins reflect [17:2] AHB address bits for word width memory, [16:1] bits for halfword width memory and [15:0] bits for byte width memory. In multiplexed Address/Data mode those bits are multiplexed between address and output data [15:0] bits. In the multiplexed Address/Data mode its behavior is affected by the LBA, LBN and LAH fields in the Chip Select control registers. In synchronous multiplexed mode its behavior is affected by the BCS, BCD and LAH fields. Bidirectional LSB address/data bus are made in IO PAD from M_DATA_IN[15:0] and ADDR[15:0]/M_DATA_OUT[15:0].
BCLK	O	Burst Clock. This active-high output signal BCLK is used to clock external, burst-capable devices to synchronize the loading and incrementing of addresses and delivery of burst read and write data to/from the WEIM. Its behavior is affected by the BCM bit in the WEIM configuration register and the SYNC bit and BCD and BCS fields in the Chip Select control registers. BCLK can start on both rising and falling edge of HCLK.
BCLK_FB	I	Burst Clock Feedback. This input is used to provide input data sampling clock in high data speed. It is a feedback from the IO PAD of the BCLK output pin that intend to align the clock used by the memory, and the one that is used to sample the read data.
BIGEND	I	Big/Little Endian. This input is used to provide big/little endian support in the WEIM. WEIM supports big and little endian accesses according Table 17-4. WEIM supports mixing big and little endian accesses.
BOOT_CFG[5:0]	O	Boot Configuration. These input pins determine the state of some WEIM configuration bits after hardware reset. See Table 17-3 for settings.
$\overline{\text{CS}}$[5:0]	O	Chip Selects. The $\overline{\text{CS}}$[5:0] signals are chip selects active-low output pins. Its behavior is affected by the CSA and CSN fields in the Chip Select control registers. $\overline{\text{CS}}$[0] address space range can be increased to 256 Mbytes by merging the $\overline{\text{CS}}$[0] and $\overline{\text{CS}}$[1] ranges. In this case the Merged Address Space (MAS) bit is set in the WEIM Configuration register, and the $\overline{\text{CS}}$[1] pin is used as address line A26. As shown in Table 17-6, the chip select signals are asserted based on a decode of address lines [31:24] (when enabled).
DATA_IN[15:0]/M_DATA_IN[31:16]	IO	Input Data Bus LSB/Input Data Multiplexed Bus MSB. These signals are the input data bus used to transfer data from external devices. In a non multiplexed mode it is LSB, in a multiplexed mode it is MSB. Bidirectional LSB data bus are made in IO PAD from DATA_IN[15:0]/M_DATA_IN[31:16] and DATA_OUT[15:0]/M_DATA_OUT[31:16].

Table 17-2. WEIM Detailed Signal Descriptions (continued)

Signal	I/O	Description
DATA_OUT[15:0]/ M_DATA_OUT[31:16]	0	Output Data Bus LSB/Output Data Multiplexed Bus MSB. These signals are the output data bus used to transfer data to an external devices. In a non-multiplexed mode, it is LSB and in a multiplexed mode, it is MSB. Bidirectional LSB data bus are made in IO PAD from DATA_IN[15:0]/M_DATA_IN[31:16] and DATA_OUT[15:0]/M_DATA_OUT[31:16].
$\overline{\text{DTACK}}$	I	Data Transfer Acknowledge. This input signal is used to externally terminate a data transfer when enabled. For $\overline{\text{DTACK}}$ enabled cycles, the bus time-out monitor generates a bus error if $\overline{\text{DTACK}}$ after has been asserted is not deasserted before 1024 clocks have elapsed. This signal is used in two modes: with rising edge detection or with level detection with an insensitiveness time. Edge detection mode is used for devices like PCMCI card. Level detection mode is used for asynchronous devices like ATI graphic controller. $\overline{\text{DTACK}}$ control keeps backward compatibility with previous architectures that used it in the Application processors.
$\overline{\text{EB}}$[3:0]	O	Enable Byte. Those active-low output pins indicate active data bytes for the current access. They may be configured to assert for read and write cycles or for write cycles only as programmed in the Chip Select control registers. $\overline{\text{EB}}$[0] corresponds to DATA_OUT[7:0] and M_DATA_OUT[7:0]. $\overline{\text{EB}}$[1] corresponds to DATA_OUT[15:8] and M_DATA_OUT[15:8]. $\overline{\text{EB}}$[2] corresponds to M_DATA_OUT[23:16]. $\overline{\text{EB}}$[3] corresponds to M_DATA_OUT[31:24]. $\overline{\text{EB}}$[3:2] also are multiplexed to the ADDR[25:24] bits in the multiplexed mode (MUM = 1). In the write accesses its behavior is affected by the EBWA and EBWN fields and EBC bit in the Chip Select control registers. In the read accesses its behavior is affected by the EBRA and EBRN fields in the Chip Select control registers.
$\overline{\text{ECB}}$	I	End Current Burst (WAIT). This active-low input signal $\overline{\text{ECB}}$ is asserted by external burst capable devices. It is serviced in synchronous mode only (SYNC=1). This signal can be used in two different modes depending on the EW bit in the Chip Select Control Register. In the ECB mode (EW=0) $\overline{\text{ECB}}$ indicates the end of the current (continuous) burst sequence. Following assertion, the WEIM terminates the current burst sequence and initiate a new one. In the WAIT mode (EW=1) the memory device asserts this signal to insert wait states during refresh collisions or during a row boundary crossing. Following assertion, the WEIM does not terminate the current burst sequence and continues it once WAIT is negated. $\overline{\text{ECB}}$ will have a pull up resistor in IO. For burst devices $\overline{\text{ECB}}$/WAIT output should be configured to change one cycle before data is ready (before delay).
GUARD	I	Guard. This active-high input signal indicates that IO is locked by another module and current access should be postponed till IO unlocked. GUARD and WR_GUARD together are used in back to back accesses between memory controllers to avoid contention on the shared pins.
IO_DIR[3:0]	O	IO Direction. These active-high output signal indicates IO direction (0 for input and “1” for output). Bidirectional buses are made in IO module from ADDR[15:0]/M_DATA_OUT[15:0] and M_DATA_IN[15:0] from DATA_OUT[15:0]/M_DATA_OUT[31:16] and DATA_IN[15:0]/M_DATA_IN[31:16]. Bit IO_DIR[0] corresponds to ADDR[7:0]/M_DATA_OUT[7:0]/M_DATA_IN[7:0], Bit IO_DIR[1] corresponds to ADDR[15:8]/M_DATA_OUT[15:8]/M_DATA_IN[15:8], Bit IO_DIR[2] corresponds to DATA_OUT[7:0]/M_DATA_OUT[23:16]/DATA_IN[7:0]/M_DATA_IN[23:16], and Bit IO_DIR[3] corresponds to DATA_OUT[15:8]/M_DATA_OUT[31:24]/DATA_IN[15:8]/M_DATA_IN[31:24]
$\overline{\text{LBA}}$	O	Load Burst Address. This active-low output signal is asserted during burst mode accesses to cause the external burst capable device to load a new starting burst address. Assertion of $\overline{\text{LBA}}$ indicates that a valid address is present on the address bus. Its behavior is affected by the SYNC bit, BCD, BCS, LBA and LBN fields in the Chip Select control registers. In asynchronous mode (SYNC=0) $\overline{\text{LBA}}$ length decreased by LBA and LBN fields. In the synchronous mode (SYNC=1) $\overline{\text{LBA}}$ length is equal BCD + BCS + LBN + 1 half AHB clock cycles.

Table 17-2. WEIM Detailed Signal Descriptions (continued)

Signal	I/O	Description
M_DATA_IN[15:0]	I	MUX Data Input Bus. These signals are LSB input data bus used to transfer data from an external devices in multiplexed mode. Bidirectional LSB address/data bus are made in IO PAD from M_DATA_IN[15:0] and ADDR[15:0]/M_DATA_OUT[15:0].
$\overline{OE}$	O	Output Enable. This active-low output signal $\overline{OE}$ indicates the bus access is a read and enables slave devices to drive the data bus with read data. Its behavior is affected by the OEA and OEN fields in the Chip Select control registers.
$\overline{RW}$	O	Read/Write. $\overline{RW}$ output signal indicates if the current bus access is a read or write cycle. A high (logic one) level indicates a read cycle and a low (logic zero) level indicates a write cycle. Its behavior is affected by the RWA and RWN fields in the Chip Select control registers.
STROBE	O	Strobe. This signal allows capture current access controls, address and data on logic analyzer. sync. and async. accesses are supported.
WR_GUARD	O	WR_GUARD. This active-high output signal indicates that IO is locked by WEIM. (It is asserted also in Extra Dead Cycles time). WR_GUARD and GUARD together are used in back to back accesses between memory controllers to avoid contention on the shared pins.

Table 17-3. Boot Configuration Settings

BOOT_CFG Bits	Configured Bits	Place
5	AUS0	WCR
4	MUM	CSCR0A
3	MAS	WCR
2:0	DSZ[2:0]	CSCR0U

Table 17-4. WEIM Out/in Data in Case AHB Out/in Data is 0xB3B2B1B0

Endian Mode	AHB Access	AHB Address [1:0]	Port Size and Used Bits								
			Word Port				Halfword Port			Byte Port	
			[31:24]	[23:16]	[15:8]	[7:0]	External address [0]	[31:24] ([15:8])	[23:16] ([7:0])	External address [1:0]	[31:24] ([15:8], [23:16], [7:0])
Big	Word	0	0xB3	0xB2	0xB1	0xB0	0	0xB3	0xB2	0	0xB3
										1	0xB2
							1	0xB1	0xB0	2	0xB1
										3	0xB0
	Half Word	0	0xB3	0xB2			0	0xB3	0xB2	0	0xB3
										1	0xB2
		2			0xB1	0xB0	1	0xB1	0xB0	2	0xB1
										3	0xB0
	Byte	0	0xB3				0	0xB3		0	0xB3
		1		0xB2					0xB2	1	0xB2
		2			0xB1		1	0xB1		2	0xB1
		3				0xB0			0xB0	3	0xB0
Little	Word	0	0xB3	0xB2	0xB1	0xB0	0	0xB1	0xB0	0	0xB0
										1	0xB1
							1	0xB3	0xB2	2	0xB2
										3	0xB3
	Half Word	0			0xB1	0xB0	0	0xB1	0xB0	0	0xB0
										1	0xB1
		2	0xB3	0xB2			1	0xB3	0xB2	2	0xB2
										3	0xB3
	Byte	0				0xB0	0		0xB0	0	0xB0
		1			0xB1			0xB1		1	0xB1
		2		0xB2			1		0xB2	2	0xB2
		3	0xB3					0xB3		3	0xB3

17.5 Memory Map and Register Definition

The WEIM module includes 19 user-accessible 32-bit registers. There is a common register called WEIM Configuration Register (WCR) that contains control bits to configure WEIM for certain operation modes. The other 18 registers: Chip Select Control Register 0–5 Upper, Lower, and Additional (CSCR0U,

CRCR0L, CRCR0A,..., CSCR5U, CSCR5L, CSCR5A) and six Chip Select Control Registers 0–5 (CSCR0–CSCR5) for each chip select. The layout of control register is slightly different for the CSCR0 register because the CSCR0 reset state depends on BOOT_CFG input. These registers are accessible only in supervisor mode with word (32-bit) reads and writes. Complete decoding is not performed, so shadowing can occur with these registers. The user should not attempt to address these registers at any other address location other than those listed in Table 17-5 and Table 17-6.

17.5.1 Memory Map

Memory map for WEIM is shown in Table 17-5 and memory map for Chip Select is shown in Table 17-6.

Table 17-5. WEIM Memory Map

Address	Definition	Access	Reset	Section/Page
0xD800_2000 (CSCR0U)	Chip Select 0 Upper Control Register	R/W	0x0000_1E00	17.5.3.1/17-12
0xD800_2004 (CSCR0L)	Chip Select 0 Lower Control Register	R/W	0x000_081[1]	17.5.3.2/17-16
0xD800_2008 (CSCR0A)	Chip Select 0 Addition Control Register	R/W	0x0000_000	17.5.3.3/17-20
0xD800_2010 (CSCR1U)	Chip Select 1 Upper Control Register	R/W	0x0000_0000	17.5.3.1/17-12
0xD800_2014 (CSCR1L)	Chip Select 1 Lower Control Register	R/W	0x0000_0000	17.5.3.2/17-16
0xD800_2018 (CSCR1A)	Chip Select 1 Addition Control Register	R/W	0x0000_0000	17.5.3.3/17-20
0xD800_2020 (CSCR2U)	Chip Select 2 Upper Control Register	R/W	0x0000_0000	17.5.3.1/17-12
0xD800_2024 (CSCR2L)	Chip Select 2 Lower Control Register	R/W	0x0000_0000	17.5.3.2/17-16
0xD800_2028 (CSCR2A)	Chip Select 2 Addition Control Register	R/W	0x0000_0000	17.5.3.3/17-20
0xD800_2030 (CSCR3U)	Chip Select 3 Upper Control Register	R/W	0x0000_0000	17.5.3.1/17-12
0xD800_2034 (CSCR3L)	Chip Select 3 Lower Control Register	R/W	0x0000_0000	17.5.3.2/17-16
0xD800_2038 (CSCR3A)	Chip Select 3 Addition Control Register	R/W	0x0000_0000	17.5.3.3/17-20
0xD800_2040 (CSCR4U)	Chip Select 4 Upper Control Register	R/W	0x0000_0000	17.5.3.1/17-12
0xD800_2044 (CSCR4L)	Chip Select 4 Lower Control Register	R/W	0x0000_0000	17.5.3.2/17-16
0xD800_2048 (CSCR4A)	Chip Select 4 Addition Control Register	R/W	0x0000_0000	17.5.3.3/17-20
0xD800_2050 (CSCR5U)	Chip Select 5 Upper Control Register	R/W	0x0000_0000	17.5.3.1/17-12
0xD800_2054 (CSCR5L)	Chip Select 5 Lower Control Register	R/W	0x0000_0000	17.5.3.2/17-16
0xD800_2058 (CSCR5A)	Chip Select 5 Addition Control Register	R/W	0x0000_0000	17.5.3.3/17-20
0xD800_2060 (WCR)	WEIM Configuration Register (WCR)	R/W	0x0000_0100	17.5.3.4/17-23

[1] Some bits are set according BOOT_CFG input.

Table 17-6. WEIM Chip Selection Memory Map

Address	Use	Access
0xC000_0000 ... 0xC7FF_FFFF	$\overline{\text{CS0}}$ memory region	R/W
0xC800_0000 ... 0xCFFF_FFFF	$\overline{\text{CS1}}$ memory region	R/W

Table 17-6. WEIM Chip Selection Memory Map (continued)

Address	Use	Access
0xD000_0000 ... 0xD1FF_FFFF	$\overline{CS2}$ memory region	R/W
0xD200_0000 ... 0xD3FF_FFFF	$\overline{CS3}$ memory region	R/W
0xD400_0000 ... 0xD5FF_FFFF	$\overline{CS4}$ memory region	R/W
0xD600_0000 ... 0xD7FF_FFFF	$\overline{CS5}$ memory region	R/W

17.5.2 Register Summary

Figure 17-2 shows the key to the register fields and Table 17-7 shows the register figure conventions.

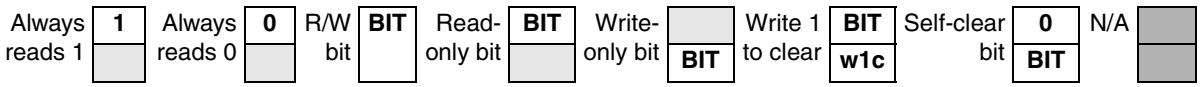


Figure 17-2. Key to Register Fields

Table 17-7. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read-only. Writing this bit has no effect.
W	Write-only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

17.5.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bits and field function follow the register diagrams, in bit order. Table 17-8 provides a summary of the registers in the WEIM module. The 96 bits

used to control Chip Select are divided into three registers: Chip Select x Upper Control Register (CSCRxU), Chip Select x Lower Control Register (CSCRxL) and Chip Select x Additional Control Register (CSCRxA).

- Bits [95:64] are located in Chip Select x Upper Control Register (see Figure 17-3).
- Bits [63:32] are located in Chip Select x Lower Control Register (see Figure 17-4).
- Bits [31:0] are located in Chip Select x Additional Control Register (see Figure 17-5).

Table 17-8. WEIM Register Summary

Name		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
0xD800_2000 (CSCR0U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0xD800_2004 (CSCR0L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CSEN
	W																
0xD800_2008 (CSCR0A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0xD800_2010 (CSCR1U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0xD800_2014 (CSCR1L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CSEN
	W																
0xD800_2018 (CSCR1A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																

Table 17-8. WEIM Register Summary (continued)

Name		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
0xD800_2020 (CSCR2U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0xD800_2024 (CSCR2L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CSEN
	W																
0xD800_2028 (CSCR2A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0xD800_2030 (CSCR3U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0xD800_2034 (CSCR3L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CSEN
	W																
0xD800_2038 (CSCR3A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0xD800_2040 (CSCR4U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0xD800_2044 (CSCR4L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CSEN
	W																

Table 17-8. WEIM Register Summary (continued)

Name		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
0xD800_2048 (CSCR4A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0xD800_2050 (CSCR5U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0xD800_2054 (CSCR5L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CSEN
	W																
0xD800_2058 (CSCR5A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0xD800_2060 (WCR)	R													ECP5	ECP4	ECP3	ECP2
	W																
	R	ECP1	ECP0	AUS5	AUS4	AUS3	AUS2	AUS1	AUS0						BCM		MAS
	W																

17.5.3.1 Chip Select x Upper Control Register (CSCRxU)

Figure 17-3 shows the register and Table 17-9 provides its field descriptions.

0xD800_2000 (CSCR0U)
0xD800_2010 (CSCR1U)
0xD800_2020 (CSCR2U)
0xD800_2030 (CSCR3U)
0xD800_2040 (CSCR4U)
0xD800_2050 (CSCR5U)

Access: User read-write

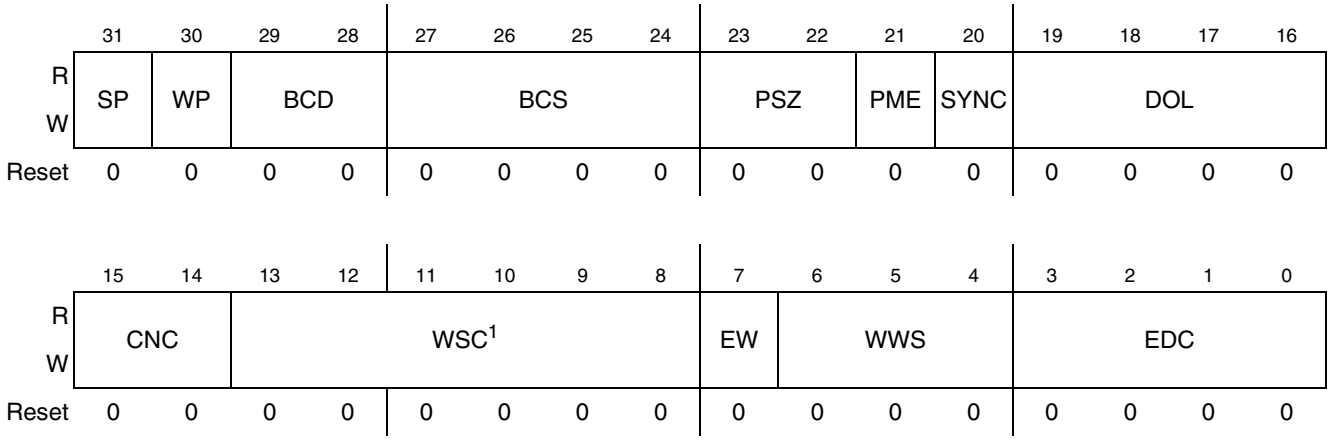


Figure 17-3. Chip Select x Upper Control Register

1 WSC field (bits 8–13) reset value is 011110 for CS0U register and is 0 for all others.

Table 17-9. Chip Select x Upper Control Register Field Descriptions

Field	Description
31 SP	Supervisor Protect. This bit prevents accesses to the address range defined by the corresponding chip select when the access is attempted in the User mode. SP is cleared by a hardware reset. 0 User mode accesses are allowed in the memory range defined by chip select. 1 User mode accesses are prohibited. All attempts to access an address mapped by this chip select in User mode results in a error response on the AHB and no assertion of the chip select output.
30 WP	Write Protect. This bit prevents writes to the address range defined by the corresponding chip select. WP is cleared by a hardware reset. 0 Writes are allowed in the memory range defined by chip. 1 Writes are prohibited. All attempts to write to an address mapped by this chip select result in a error response on the AHB and no assertion of the chip select output.
29–28 BCD	Burst Clock Divisor. This bit field contains the value used to program the burst clock divisor for BCLK generation. It is used to divide the internal AHB bus frequency (HCLK 133 MHz). See 17.6.4/17-26 for more information on the burst clock divisors. An example is shown in Figure 17-41. When the BCM bit is set in the WEIM configuration register, BCD is ignored. BCD is cleared by a hardware reset. 00 Divide AHB clock by 1 01 Divide AHB clock by 2 10 Divide AHB clock by 3 11 Divide AHB clock by 4
27–24 BCS	Burst Clock Start. If SYNC 1 this bit field determines the number of half cycles after address assertion before the first rising edge of BCLK is seen. See example on the Figure 17-41. A value of 0 results in a half clock delay, not an immediate assertion. When the BCM bit is set in the WEIM configuration register, this overrides the BCS bits. BCS is cleared by a hardware reset. 0000 1 half AHB clock cycle. 0001 2 half AHB clock cycles. 1111 16 half AHB clock cycles.

Table 17-9. Chip Select x Upper Control Register Field Descriptions (continued)

Field	Description
23–22 PSZ	Page Size. If PME is clear the PSZ bit field indicates memory burst length in words (where *word* is defined by the DSZ field) and should be properly initialized for mixed wrap/increment AHB accesses support. Continuous PSZ value corresponds to continuous burst length setting of the external memory device. If PME is set (set to 1) the PSZ bit field indicates number of words (where “word” is defined by the port size in DSZ field) in a page in memory. This ensures that the WEIM does not burst past a page boundary at increment access when the PME bit is set. PSZ is cleared by a hardware reset. See Table 17-10 for PSZ Bit Field Combinations.
21 PME	Page Mode Emulation. This bit enables page mode emulation in burst mode. When PME is set (and SYNC equals 1), the external address is asserted for each piece of data requested. Additionally, the $\overline{\text{LBA}}$ and BCLK signals behave in the same way when an asynchronous access is performed (see Figure 17-25). PME is cleared by a hardware reset. 0 Disables page mode emulation 1 Enables page mode emulation
20 SYNC	Synchronous Burst Mode Enable. This bit enables synchronous burst mode if PME is clear (see also PME and PSR description). When enabled, the WEIM is capable of interfacing to burst flash devices through additional burst control signals: BCLK, $\overline{\text{LBA}}$, and $\overline{\text{ECB}}$ (see example on the Figure 17-29). The sequencing of these additional I/Os is controlled by other WEIM configuration register bit settings as described in Section 17.6.3, “Burst Mode Memory Operation.” SYNC is cleared by a hardware reset. 0 Disables synchronous burst mode 1 Enables synchronous burst mode
19–16 DOL	Data Output Length. If SYNC is set (equals 1) the DOL bit field specifies number of wait states during the burst access after the delay of the first data according to the settings shown below (see examples on the Figure 17-25 and Figure 17-40). The reset value 0 specifies that burst data is held for a single AHB clock period. As AHB clock frequencies increase, it may become necessary to delay sampling the data for multiple AHB clock periods in order to meet burst memory setup and/or frequency specifications and/or WEIM data setup time requirements. DOL has no effect on burst data length when SYNC = 0. DOL is cleared by a hardware reset. 0000 1 AHB clock cycle data length. 0001 2 AHB clock cycles data length. — 1111 16 AHB clock cycles data length.
15–14 CNC	Chip Select Negation Clock Cycles. This bit field specifies the minimum number of clock cycles a chip select must remain negated after it is negated (but doesn’t guarantee negation for back-to-back accesses, it requires EDC using) according to the settings shown below. See examples on the Figure 17-12, and Figure 17-13. CNC has no effect on write accesses when any CSA bit is set. CNC is cleared by a hardware reset. The number of clock cycles of this field can be increased using the CNC2 bit in the appropriate Chip Select Addition Control Register. 00 0 Minimum number of AHB clock cycles $\overline{\text{CS}}$ must remain negated. 01 1 Minimum number of AHB clock cycles $\overline{\text{CS}}$ must remain negated. 10 2 Minimum number of AHB clock cycles $\overline{\text{CS}}$ must remain negated. 11 3 Minimum number of AHB clock cycles $\overline{\text{CS}}$ must remain negated.

Table 17-9. Chip Select x Upper Control Register Field Descriptions (continued)

Field	Description
13–8 WSC	Wait State Control. This bit field programs the number of wait-states for an access to the external device connected to the chip select (see Figure 17-7). For SYNC = 1 WSC programs the number of AHB clock cycles required for the initial access (see Figure 17-25, and Figure 17-32) of a memory burst sequence initiated by the WEIM to an external burst device. For EW = 1 after the wait cycle count expires $\overline{ECB}$ is sampled and the cycle terminates when the $\overline{ECB}$ is negated. On other case WEIM special watch dog counter check that $\overline{ECB}$ will not be asserted for more than 1024 AHB clocks. Additionally in this case WEIM suppresses $\overline{LBA}$ generation after next $\overline{ECB}$ assertion (during increment burst at page boundary crossing). For write accesses the number of wait-states is increased according WWS value or decreased by DWW (see Table 17-11). WSC = 11_1111 indicates operation in a positive edge-sensitive DTACK mode. It selects $\overline{DTACK}$ input as access length control sign (instead of default WSC counter). It means that access length is determined by $\overline{DTACK}$ length. WSC is set to 01_1110 by a hardware reset for CSCR0. WSC is cleared by a hardware reset for CSCR1 – CSCR5. **Note:** For SYNC=1 and DOL=0, WSC value should be at least 4. For SYNC=1, MUM =0, WSC≥ 2(BCD + 1) + (BCS + LBN + 2)/2. For SYNC=1, MUM =1, WSCŠ 2(BCD + 1) + (BCS + LBN + 2 + LBH) + 2/2. For PSR=1, the number of wait-states is increased by one for read access.
7 EW	$\overline{ECB}$/WAIT. This bit determines how WEIM supports the $\overline{ECB}$ input in the synchronous mode. In asynchronous mode this bit determines the operation of level-sensitive DTACK mode. (see Table 17-18 for EW effect on WEIM operation modes) 0 For SYNC = 1, if $\overline{ECB}$ goes to low state in the middle of memory burst access then the WEIM starts a new access by asserting the current AHB address to the ADDR pins and LBA assertion (ECB mode). If SYNC = 0 and WSC=111111, the WEIM waits for $\overline{DTACK}$ posedge for access termination. 1 If $\overline{ECB}$ goes to low state in the middle of a memory burst access then the WEIM waits until $\overline{ECB}$ goes high (WAIT mode) to continue current access; at the end of first access in burst it allows to wait $\overline{ECB}$ negation till 1024 clock. For SYNC = 0 WEIM begins access and after (2+DCT) clocks tests $\overline{DTACK}$ input. If $\overline{DTACK}$ is low WEIM waits $\overline{DTACK}$ high state then loads WSC value (see Figure 17-27 and Figure 17-28).
6–4 WWS	Write Wait State. This bit field determines whether additional wait-states are required for write cycles (see Table 17-11). This is useful for writing to memories that require additional data setup time (see example on Figure 17-9). The DWW field should be zero when this field is in use. WWS is cleared by a hardware reset. **Note:** To decrease write wait states use the DWW bit field.
3–0 EDC	Extra Dead Cycles. This bit field determines whether idle cycles are inserted before a new access (see example in Figure 17-10). If the currently accessed CS EDC field is not empty then idle cycles are inserted before next access except for two conditions: current access is an asynchronous write (its SYNC = 0) or the next access is an asynchronous read from the same chip select. This field is used in two cases: • Slow memory or peripherals that use long CS or $\overline{OE}$ to output data hold times to prevent data bus contention on back-to-back external transfers. • Synchronous accesses (SYNC = 1) to provide $\overline{CS}$ high minimum pulse width (even for back-to-back accesses). The EDC field is cleared by a hardware reset. **Note:** On some occasions, setting EDC field to 0000 may result in memory read/write mistakes. Therefore, the recommended configuration for EDC field is 2 or higher. 0000 No idle AHB clock cycles inserted. 0001 1 Idle AHB clock cycle inserted. — 1111 15 AHB clock cycles inserted.

Table 17-10. PSZ Bit Field Values

PSZ	PME=0 Memory Burst Length	PME=1 Number of Words in Page
00	4	4
01	8	8
10	16	16
11	continuous	32

Table 17-11. WSC Bit Field Values

WSC	Number of Wait-States					
	Read Access	Write Access				
		WWS = 0, DWW = 0	WWS = 1, DWW = 0	WWS = 7, DWW = 0	WWS = 0, DWW = 1	WWS = 0, DWW = 2
000000	1	1	1	7	1	1
000001	1	1	2	8	1	1
000010	2	2	3	9	1	1
000011	3	3	4	10	2	1
000100	4	4	5	11	3	2
...	—	—	—	—	—	—
110111	119	119	120	126	118	117
111000	120	120	121	127	119	118
111001	121	121	122	127	120	119
111010	122	122	123	127	121	120
111011	123	123	124	127	122	121
111100	124	124	125	127	123	122
111101	125	125	126	127	124	123
111110	126	126	127	127	125	124
111111	Posedge sensitive DTACK mode					

17.5.3.2 Chip Select x Lower Control Register (CSCRxL)

Figure 17-4 shows the register and Table 17-12 provides its field descriptions.

0xD800_2004 (CSCR0L)
0xD800_2014 (CSCR1L)
0xD800_2024 (CSCR2L)
0xD800_2034 (CSCR3L)
0xD800_2044 (CSCR4L)
0xD800_2054 (CSCR5L)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	OEA[1]				OEN				EBWA				EBWN			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CSA				EBC[2]	DSZ[3]			CSN[4]				PSR	CRE	WRAP	CSEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0[5]

Figure 17-4. Chip Select x Lower Control Register

1 OEA (bits 28–31) reset value is 1010 for CSCR0L register and 0 for other registers

2 EBC (bit 11) reset value is 1 for CSCR0L register and 0 for other registers

3 DSZ (bits 8–10) reset value is configurable for the CSCR0L register and 0 for other registers

4 CSN (bits 4–7) reset value is 0100 for CSCR0L register and 0 for other registers

5 Bit 0 reset value is 1 for CSCR0L register and 0 for other registers

Table 17-12. Chip Select x Lower Control Register Field Descriptions

Field	Description
31–28 OEA	$\overline{OE}$ Assert. This bit field determines when $\overline{OE}$ is asserted during a read cycle. For SYNC = 0, OEA determines number of half clocks before $\overline{OE}$ asserts during a read cycle. For SYNC = 1, after initial memory burst access, $\overline{OE}$ is asserted continuously for subsequent memory burst accesses, and is not affected by OEA (see memory burst read timing diagram for more detail); the behavior of $\overline{OE}$ on initial memory burst access is same as when SYNC = 0 (see example on Figure 17-25). OEA field do not affect the cycle length. OEA is set to 1010 by a hardware reset for CSCR0L register and is cleared for other registers. **Note:** Minimum time $\overline{OE}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{OE}$ assertion and end of access 0001 1 Half AHB clock cycle between $\overline{OE}$ assertion and end of access — 1111 15 Half AHB clock cycles between $\overline{OE}$ assertion and end of access
27–24 OEN	$\overline{OE}$ Negate. This bit field determines when $\overline{OE}$ is negated during a read cycle (see example on Figure 17-20). Setting the SYNC bit (SYNC = 1) overrides OEN and $\overline{OE}$ negates at the end of a read access and no sooner. OEN does not affect the cycle length, except in posedge sensitive DTACK mode. OEN is cleared by a hardware reset. **Note:** The term “end of a read access” is the nearest possible address, data or control signal change by another access (an own next access or an access from others pin shared controller). Minimum time $\overline{OE}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{OE}$ negation and end of access 0001 1 Half AHB clock cycle between $\overline{OE}$ negation and end of access — 1111 15 Half AHB clock cycles between $\overline{OE}$ negation and end of access
23–20 EBWA	$\overline{EB}$ Write Assert. This bit field determines when $\overline{EB}$ [3:0] is asserted during write cycles (see example on Figure 17-8). This is useful to meet data setup time requirements for slow memories. EBWA does not affect the cycle length. EBWA is cleared by a hardware reset. 0000 0 Half AHB clock cycles before $\overline{EB}$ is asserted. 0001 1 Half AHB clock cycle before $\overline{EB}$ is asserted. — 1111 15 Half AHB clock cycles before $\overline{EB}$ is asserted.
19–16 EBWN	$\overline{EB}$ Write Negate. This bit field determines when $\overline{EB}$ [3:0] outputs are negated during a write cycle (see example on Figure 17-8). This is useful to meet data hold time requirements for slow memories. EBWN does not affect the cycle length, except in posedge sensitive DTACK mode. Setting the SYNC bit (SYNC = 1) overrides EBWN and $\overline{EB}$ negates at the end of a write access and according AHB hbstrb[3:0]. EBWN is cleared by a hardware reset. 0000 0 Half AHB clock cycles between $\overline{EB}$ negation and end of access. 0001 1 Half AHB clock cycle between $\overline{EB}$ negation and end of access. — 1111 15 Half AHB clock cycles between $\overline{EB}$ negation and end of access.
15–12 CSA	$\overline{CS}$ Assert. This bit field determines when chip select is asserted for devices that require additional address setup time (see example on Figure 17-11). It does not affect the cycle length. CSA is cleared by a hardware reset. **Note:** CSA bit setting affects both reads and writes for all WEIM modes. 0000 0 Half AHB clock cycles before $\overline{CS}$ is asserted. 0001 1 Half AHB clock cycle before $\overline{CS}$ is asserted. — 1111 15 Half AHB clock cycles before $\overline{CS}$ is asserted.
11 EBC	Enable Byte Control. This bit indicates the types of access that assert Enable Byte outputs $\overline{EB}$[3:0] (see example on Figure 17-7). The $\overline{EB}$[3:0] outputs can be configured as byte write enables. EBC is set by a hardware reset for CSCR0L register and is cleared for other registers. 0 Both read and write accesses assert the $\overline{EB}$[3:0]. 1 Only write accesses assert the $\overline{EB}$[3:0], thus configuring as byte write enables.

Table 17-12. Chip Select x Lower Control Register Field Descriptions (continued)

Field	Description
10–8 DSZ	Data Port Size. This bit field defines the width of an external device's data port as shown in the Table 17-13. DSZ is mapped by a hardware reset for CSCR0L by the value of the BOOT_CFG [2:0] bits. BOOT_CFG [2] maps to DSZ [2], BOOT_CFG [1] maps to DSZ [1] and BOOT_CFG [0] maps to DSZ [0]. DSZ and MUM (multiplexed mode) affected on data port location as shown in Table 17-13. DSZ is cleared by a hardware reset of CSCR1L–CSCR5L.
7–4 CSN	$\overline{CS}$ Negate. This bit field determines when chip select is negated for devices that require additional address/data hold times (see example on Figure 17-11). CSN affects only asynchronous (read and write) access (SYNC=0), and is ignored on synchronous (SYNC=1). CSN does not affect cycle length, except in positive edge sensitive DTACK mode. CSN is set to 0100 by a hardware reset for CSCR0L register and is cleared by a hardware reset for other registers. 0000 0 Half AHB clock cycles between $\overline{CS}$ negation and end of access. 0001 1 Half AHB clock cycle between $\overline{CS}$ negation and end of access. — 1111 15 Half AHB clock cycles between $\overline{CS}$ negation and end of access.
3 PSR	Pseudo SRAM Enable (Burst Write Enable). This bit enables four function for Pseudo SRAM (for example,. CellularRAM™) or any other device that support these modes: burst write, write wrap disable, read wait state increase and memory control register accessibility. If PSR=1, then memory burst write is enable (with SYNC = 1). In this mode, WRAP bit is masked on write time, unless WWU bit is set in CSCRxA register, and wait state on read is automatically increased to WSC +1 (see Figure 17-40 and Figure 17-41). An asynchronous access (SYNC=0) should be used with PSR=1 and CRE=1 to write to the memory control register. PSR is cleared by a hardware reset. 0 PSRAM mode is disabled. 1 PSRAM mode is enabled.
2 CRE	Control Register Enable. This bit indicates CRE memory pin state while writing to CS address space, for PSRAM control register write. For PSR=1 the CRE bit will be driven on pin ADDR[23] in a write access time. CRE is cleared by a hardware reset. **Note:** SYNC = 0 should be used to access to PSRAM control register. 0 CRE pin 0 1 CRE pin 1
1 WRAP	Wrap Memory Mode. This bit indicates that memory is in wrap mode. Wrap size is set using the PSZ field. In case not matching wrap boundaries in both memory (PSZ field) and AHB access on current address, WEIM puts address on address bus and generates $\overline{LBA}$ signal (see example on Figure 17-37). WRAP is cleared by a hardware reset. 0 Memory is in not in wrap mode. 1 Memory is in wrap mode.
0 CSEN	$\overline{CS}$ Enable. This bit controls the operation of the chip select pin. CSEN is set by a hardware reset for CSCR0L to allow CSCR0L to select from an external boot ROM. CSEN is cleared by a hardware reset to CSCR1L–CSCR5L. 0 Chip select function is disabled; attempts to access an address mapped by this chip select results in a error respond on the AHB and no assertion of the chip select output. 1 Chip select is enabled, and is asserted when presented with a valid AHB access.

Table 17-13. DSZ Bit Field Values

DSZ	Data Port Size	
	MUM=0	MUM=1
000	Reserved	Reserved
001	Reserved	Reserved
010	8-bit port, resides on DATA_IN/OUT [15:8] pins	Reserved
011	8-bit port, resides on DATA_IN/OUT [7:0] pins	Reserved

Table 17-13. DSZ Bit Field Values (continued)

DSZ	Data Port Size	
	MUM=0	MUM=1
100	Reserved	16-bit port, resides on ADDR/M_DATA_IN/OUT [15:0] pins
101	Reserved	Reserved
110	Reserved	32-bit port, resides on ADDR/M_DATA_IN/OUT [15:0] and M_DATA_IN/OUT [31:16] pins
111	Reserved	The same

17.5.3.3 Chip Select x Additional Control Register (CSCRxA)

Figure 17-5 shows the register and Table 17-14 provides its field descriptions.

0xD800_2008 (CSCR0A)
0xD800_2018 (CSCR1A)
0xD800_2028 (CSCR2A)
0xD800_2038 (CSCR3A)
0xD800_2048 (CSCR4A)
0xD800_2058 (CSCR5A)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EBRA				EBRN				RWA				RWN			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MUM	LAH[1]			LBN[2]			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
Reset	1[3]0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-5. Chip Select x Addition Control Register

1 LAH (bits 13, 14) reset value is 10 for CSCR0A and 0 for other registers
2 LBN (bits 10 -12) reset value is 100 for CSCR0A and 0 for other registers
3 MUM (bit 15) reset value is determined by settings of BOOT_CFG inputs (see Table 17-3) for CSCR0A and 0 for other registers

Table 17-14. Chip Select x Addition Control Register Field Descriptions

Field	Description
31–28 EBRA	$\overline{EB}$ Read Assert. This bit field determines when $\overline{EB}$ [3:0] is asserted during read cycles (see example on Figure 17-25). EBRA does not affect the cycle length. EBRA is cleared by a hardware reset. **Note:** Minimum time $\overline{EB}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles before $\overline{EB}$ asserted. 0001 1 Half AHB clock cycle before $\overline{EB}$ asserted. — 1111 15 Half AHB clock cycles before $\overline{EB}$ asserted.
27–24 EBRN	$\overline{EB}$ Read Negate. This bit field determines when $\overline{EB}$ [3:0] outputs are negated during a read cycle (see example on Figure 17-20). EBRN does not affect the cycle length, except when in positive edge sensitive DTACK mode. Setting the SYNC bit (SYNC = 1) overrides EBRN and $\overline{EB}$ negates at the end of a read access but not sooner. EBRN is cleared by a hardware reset. **Note:** Minimum time $\overline{EB}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{EB}$ negation and end of access. 0001 1 Half AHB clock cycle between $\overline{EB}$ negation and end of access. — 1111 15 Half AHB clock cycles between $\overline{EB}$ negation and end of access.
23–20 RWA	$\overline{RW}$ Assertion. This bit field determines when $\overline{RW}$ is asserted during write cycles. (see example on Figure 17-28). RWA is cleared by a hardware reset. **Note:** Minimum time $\overline{RW}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles $\overline{RW}$ delay 0001 1 Half AHB clock cycle $\overline{RW}$ delay. — 1111 15 Half AHB clock cycles $\overline{RW}$ delay.
19–16 RWN	$\overline{RW}$ Negation. This bit field determines when $\overline{RW}$ is negated during a write cycle (see example on Figure 17-28). RWN does not affect the cycle length, except in posedge sensitive DTACK mode. Setting the SYNC bit (SYNC = 1) overrides RWN and $\overline{RW}$ negates at the end of a write access and no sooner. RWN is cleared by a hardware reset. **Note:** Minimum time $\overline{RW}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{RW}$ negation and end of access. 0001 1 Half AHB clock cycle between $\overline{RW}$ negation and end of access. — 1111 15 Half AHB clock cycles between $\overline{RW}$ negation and end of access.
15 MUM	Multiplexed Mode. This bit determines the address/data multiplexed mode for asynchronous and synchronous accesses (see examples in Figure 17-43–Figure 17-46). Port mapping is defined in the Table 17-13. MUM is cleared by a hardware reset for CSCR1A–CSCR5A. For CSCR0A MUM is configured at reset time with the BOOT_CFG[4] (see Table 17-3). 0 Non-multiplexed mode 1 Multiplexed mode
14–13 LAH	$\overline{LBA}$ to Address Hold. This bit field determines address hold time after LBA de-assertion for MUM = 1 only. See example on the Figure 17-43 and Figure 17-44. LAH is cleared by a hardware reset for CSCR1A–CSCR5A. For CSCR0A LAH is set to 10 by hardware reset. 00 0 AHB half clock cycles between $\overline{LBA}$ negation and address invalid. 01 1 AHB half clock cycle between $\overline{LBA}$ negation and address invalid. 10 2 AHB half clock cycles between $\overline{LBA}$ negation and address invalid. 11 3 AHB half clock cycles between $\overline{LBA}$ negation and address invalid.

Table 17-14. Chip Select x Addition Control Register Field Descriptions (continued)

Field	Description
12–10 LBN	$\overline{\text{LBA}}$ Negation. This bit field determines when $\overline{\text{LBA}}$ is negated. For SYNC=0 and MUM =0 LBN determines how many half AHB clock cycle will be between $\overline{\text{LBA}}$ negation and end of access (see example on the Figure 17-8). For SYNC=0 and MUM =1 this field determines $\overline{\text{LBA}}$ length (see example on the Figure 17-44). Negation time and $\overline{\text{LBA}}$ lengths are listed in Table 17-15. For SYNC=1 (MUM=0 and MUM=1) $\overline{\text{LBA}}$ negation occurs (LBN+BCD+1) half AHB clock cycles after first BCLK posedge detection. LBN does not affect the cycle length, except in positive edge sensitive DTACK mode. LBN is cleared by a hardware reset for CSCR1A–CSCR5A. For CSCR0A LBN is set to 100 by hardware reset. **Note:** Minimum of time $\overline{\text{LBA}}$ to be asserted is a one clock for MUM = 0 and two clocks for MUM = 1.
9–8 LBA	$\overline{\text{LBA}}$ Assertion. This bit field determines when $\overline{\text{LBA}}$ is asserted according the settings shown below (see example on the Figure 17-11). LBA is cleared by a hardware reset. **Note:** LBA field affects all modes. Minimum of time $\overline{\text{LBA}}$ to be asserted is a one clock for MUM = 0 and two clocks for MUM = 1. 00 0 AHB half clock cycles between beginning of access and $\overline{\text{LBA}}$ assertion. 01 1 AHB half clock cycle between beginning of access and $\overline{\text{LBA}}$ assertion. 10 2 AHB half clock cycles between beginning of access and $\overline{\text{LBA}}$ assertion. 11 3 AHB half clock cycles between beginning of access and $\overline{\text{LBA}}$ assertion.
7–6 DWW	Decrease Write Wait State. This bit field in combination with WWS determines whether write cycles are shorter than the read cycles (see Table 17-11). WWS field should be zero when this field is in use. DWW is cleared by a hardware reset.
5–4 DCT	$\overline{\text{DTACK}}$ Check Time. This bit field determines time of insensitivity at the beginning of access for SYNC=0 and EW=1 according to the settings shown below (see example on the Figure 17-27). DCT is a number of clock cycles between $\overline{\text{CS}}$ assertion and first $\overline{\text{DTACK}}$ check. DCT is cleared by a hardware reset. 00 2 AHB clock cycles between $\overline{\text{CS}}$ assertion and first $\overline{\text{DTACK}}$ check. 01 6 AHB clock cycles between $\overline{\text{CS}}$ assertion and first $\overline{\text{DTACK}}$ check. 10 8 AHB clock cycles between $\overline{\text{CS}}$ assertion and first $\overline{\text{DTACK}}$ check 11 12 AHB clock cycles between $\overline{\text{CS}}$ assertion and first $\overline{\text{DTACK}}$ check.
3 WWU	Write Wrap Unmask. This bit allow unmask WRAP bit in case PSR = 1 and write access. WWU is cleared by a hardware reset. 0 Prevents Wrap during write access 1 Allow wrap on write
2 AGE	Acknowledge Glue Enable. This bit is used to enable/disable glue logic between external $\overline{\text{DTACK}}$ and internal control logic. The glue logic is a flip-flop that is reset by $\overline{\text{CS}}$ assertion, it's data input is a constant 1 and $\overline{\text{DTACK}}$ goes to it's clock input. This glue logic is used to synchronize the posedge of the external $\overline{\text{DTACK}}$ in a worst noise or a slowly edge grown conditions. AGE is cleared by a hardware reset. 0 Disable glue logic 1 Enable glue logic
1 CNC2	Chip Select Negation Clock Cycles, Bit [2]. This bit is used to increase the CNC field to a 3-bit field. See CNC field description in the Chip Select x Upper Control Register. CNC2 is cleared by a hardware reset. The number of AHB clock cycles produced by both bit fields is shown in Table 17-16.
0 FCE	Feedback Clock Enable. This bit is used to enable/disable data capture by BCLK_FB. If FCE=1, WEIM used addition one clock to synchronize feedback clock captured data to AHB clock, so read access is slow then FCE=0. FCE is cleared by a hardware reset. 0 Data captured using AHB clock 1 Data captured using BCLK_FB

Table 17-15. LBN Bit Field Values

LBN	Half AHB clock cycle between $\overline{LBA}$ negation and end of access	$\overline{LBA}$ length, half AHB clock cycle
	MUM = 0	MUM = 1
000	0	2
001	1	3
...	—	—
111	7	9

Table 17-16. CNC/CNC2 Bit Values

CNC2	CNC	Minimum $\overline{CS}$ Negation, in AHB clock cycle
0	00	0
0	01	2
0	10	3
0	11	4
1	00	5
1	01	6
1	10	7
1	11	8

17.5.3.4 WEIM Configuration Register (WCR)

WCR contains control bits for the configuration and operation of WEIM. Figure 17-6 shows the register and Table 17-17 provides its field descriptions.

0xD800_2060 (WCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	ECP5	ECP4	ECP3	ECP2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ECP1	ECP0	AUS5	AUS4	AUS3	AUS2	AUS1	AUS0	0	0	0	0	0	BCM		MAS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1/0[1]

Figure 17-6. WCR Register

Table 17-17. WEIM Control Register Field Descriptions

Field	Description
31–20	Reserved
19-14 ECP5 ECP4 ECP3 ECP2 ECP1 ECP0	$\overline{ECB}$ Capture Phase. This bit indicates in which phase of HCLK, BCLK is generated to the memory for synchronous (CS5, CS4, CS3, CS2, CS1 or CS0) write accesses. This bit is XORed with BCS[0] bit in write accesses to determine the BCLK starting phase to memory and it also influence on capturing of ECB in WEIM design. 0 BCLK starting phase is as BCS[0] bit indicates. 1 BCLK starting phase is the opposite of BCS[0] bit indicates for write accesses only.
13 AUS5 AUS4 AUS3 AUS2 AUS1 AUS0	Address Unshifted for (CS5, CS4, CS3, CS2, CS1 or CS0). This bit indicates an unshifted mode for address assertion for (CS5, CS4, CS3, CS2, CS1 or CS0) accesses. This bit is cleared by a hardware reset. 0 Address shifted according (CS5, CS4, CS3, CS2, CS1 or CS0) port size. 1 Address unshifted
7–3	Reserved
2 BCM	Burst Clock Mode. This bit selects the burst clock mode of operation. It is mainly used for system debug mode. BCM is cleared by a hardware reset. 0 Burst clock runs only when accessing a chip select range with SYNC bit set; when burst clock is not running, it remains in a logic 0 state; when burst clock is running, it is configured by BCD and BCS fields in chip select control register. 1 Burst clock runs on every memory access (independent of chip select configuration)
1	Reserved
0 MAS	Merged Address Space. This bit indicates merged address space mode. If MAS is set the $\overline{CS1}$ address space is merged with $\overline{CS0}$ for a total of 256 (for halfword width port and 512 for word width port) Mbytes. $\overline{CS1}$ output is used as A26. This bit is configured at reset time with the BOOT_CFG[] pin. 0 Standard address space 1 Merged address space

17.6 Functional Description

17.6.1 Configurable Bus Sizing

WEIM supports byte, halfword, and word operands allowing access to 8-bit, 16-bit, and multiplexed 32-bit ports. Port size is programmable via the DSZ field in the corresponding Chip Select control register. In addition, portion of the data bus used for transfer to or from an 8-bit ports is programmable via the same DSZ field. An 8-bit port can reside on DATA_IN/OUT bus bits [15:8] or [7:0]. A 16-bit port reside on DATA_IN/OUT bus bits [15:0]. A 32-bit multiplexed port resides on M_DATA_IN/OUT bus bits [31:0].

NOTE

Misaligned transfers are not supported.

A word access to or from an 8-bit port requires four external bus cycles to complete the transfer. A word access to or from a 16-bit port requires two external bus cycles to complete the transfer. A halfword access to or from an 8-bit port requires two external bus cycles to complete the transfer. In case of a multi-cycle

transfer, the lower two address bits (ADDR[1:0]) are incremented appropriately. WEIM address bus is configured according to DSZ field and AUSx bits, too. There is either one or two bits right shift of AHB address bits for halfword or word width port accordingly. WEIM has a data multiplexer which takes the four bytes of the AHB interface data bus and routes them to their required positions to properly interface to memory and peripherals.

17.6.2 WEIM Operational Modes

WEIM has 9 main operational modes selected by control fields settings as described in the Table 17-18. For details see corresponding bit fields descriptions.

Table 17-18. WEIM Operation Modes Field Settings

Control Fields					Brief Mode Description
SYNC	PME	MUM	EW	WSC	
0	0	0	0	< 11_1111	Asynchronous
		1	0		Asynchronous multiplexed
		0	1		Asynchronous level sensitive DTACK mode
		0	0	11_1111	Asynchronous posedge sensitive DTACK mode
1	1	0	0	< 11_1111	Page mode emulation
	0	0	0		Synchronous burst with restart on $\overline{\text{ECB}}$ negation
		1	0		Synchronous multiplexed burst with restart on $\overline{\text{ECB}}$ negation
		0	1		Synchronous burst with wait on $\overline{\text{ECB}}$ negation
		1	1		Synchronous multiplexed burst with wait on $\overline{\text{ECB}}$ negation

17.6.3 Burst Mode Memory Operation

With memory burst mode enabled (SYNC = 1), WEIM attempts to translate AHB burst accesses to memory burst accesses, being limited by the memory burst length, predefined PSZ value, or memory and AHB WRAP/INCR boundary crossing non-matching. WEIM only displays the first address accessed in a memory burst sequence unless the page mode emulation (PME) bit is set. WEIM may translate from some AHB sequential accesses to one or few memory bursts, but not from two AHB nonsequential accesses to one memory burst.

For the first access in a memory burst sequence, WEIM asserts $\overline{\text{LBA}}$—causing the external burst device to latch the starting burst address—and then toggle the burst clock (BCLK) a predefined number of cycles in order to latch the first unit of data. Subsequent accessed data units can then be burst in fewer clock cycles, realizing an overall increase in bus bandwidth.

Memory burst accesses are terminated by WEIM whenever it detects that:

- The next AHB access is not sequential,
- The next sequential access crosses boundary with unequal condition (wrap/increment, burst length) on the AHB and memory,

- Current memory burst length reached,
- By external burst device request it needs additional cycles to retrieve the next requested memory location.

In last case, burst memory device provides an $\overline{ECB}$ (or WAIT) feedback signal to WEIM whenever it is necessary to terminate/postponed the on-going burst sequence. If EW = 0, WEIM initiate a new (with long first access) memory burst sequence, if EW = 1, WEIM only waits for $\overline{ECB}$ negation to continue current memory burst sequence. Additionally EW = 1 allows wait states insertion after wait state counter expires, but $\overline{ECB}$ still asserted. Over this a new memory burst sequence should be generated. Synchronous mode is also used for burst Cellular RAM, which supports memory burst writes, which is enabled by PSR = 1.

17.6.4 Burst Clock Divisor

In some cases it may be necessary to slow the external bus in relation to the internal bus to allow accesses to burst devices that have a maximum operating frequency which is less than the operating frequency of internal bus. Internal AHB bus frequency (HCLK 133 MHz) can be divided by two, three, or four for presentation on the external bus in burst mode operation. By programming the BCD field to various values, two signals on the external bus are affected; $\overline{LBA}$ and BCLK. $\overline{LBA}$ signal is asserted according to LBA field programming and remain asserted until the first falling edge of BCLK signal. BCLK signal runs with 50% duty cycle until a non-sequential internal request is received or an external $\overline{ECB}$ signal is recognized.

Caution should be exercised when programming these fields to ensure WSC and DOL fields are coordinated to provide the desired external bus waveforms. BCD and DOL fields should always get the same value when configured. For example, if BCD field is programmed to 01, DOL field should be programmed to 0001 and if BCD field is programmed to 10, DOL field should be programmed to 0010. BCM bit in WEIM configuration register has priority over the BCD field. If BCM = 1, BCLK runs at full frequency on every memory access (both with SYNC=1 and with SYNC=0). BCM bit is used mainly for system debug mode. It has no functional use of WEIM.

17.6.5 Burst Clock Start

In an effort to allow greater flexibility in achieving minimum number of wait states on bursted accesses, user can determine when they want the BCLK to start toggling. This allows BCLK to be skewed from point of data capture on the AHB clock by any number of AHB clock phases. Care must be exercised when setting BCS field in conjunction with the BCD, WSC, and DOL fields. See external timing diagrams from Section 17.8.4, “Burst Memory Accesses Timing Diagrams” for some examples of how to use the BCS, BCD, WSC, and DOL fields together.

17.6.6 Page Mode Emulation

Setting PME and SYNC bits causes WEIM to perform memory bursted accesses by emulating page mode operation. $\overline{LBA}$ signal remains asserted for entire access, burst clock does not send a signal, and the external address asserts when each access are made. The initial access timing is dictated by the WSC field, and the page mode access timing is dictated by the DOL field. See external timing diagrams from the Section 17.8.2, “Page Mode Timing Diagrams” for some examples. WEIM can take advantage of improved page timing for sequential accesses only. Accesses that are on the page but are not sequential in

nature have their timing dictated by the WSC field. The page size can be set via the PSZ field to 4, 8, 16, or 32 words (word size is determined by data width of the external memory, such as the DSZ field).

17.6.7 PSRAM Mode Operation

A control bit PSR is provided to enable PSRAM operation. For SYNC = 1, this bit enables a memory burst write. In this mode, WRAP bit on write time is automatically masked (CellularRAM™ SPEC wrap supports only for read accesses) unless WWU bit is set. Initial wait state value is automatically incremented on read access (see Figure 17-40 and Figure 17-41). Bit EW determines how WEIM supports $\overline{\text{ECB}}$ input. For SYNC = 1, if $\overline{\text{ECB}}$ goes to low state in middle of memory burst access then WEIM only waits; it'll go high (WAIT mode) to continue current access; at the end of first access in memory burst it allows to wait $\overline{\text{ECB}}$ negation during PSRAM refresh insertion. Bit CRE and an unused address line can be used to drive the control register enable (CRE) memory input to load the PSRAM configuration registers. For PSR = 1, CRE bit will be driven on pin ADDR[23] in a write access time.

NOTE

SYNC = 0 should be used to access the PSRAM control register.

17.6.8 Multiplexed Address/Data mode

A control bit MUM allows memory support with multiplexed address/data bus both in asynchronous and synchronous modes. LBN and LBH bit fields should be used for proper bus timing setup (see Figure 17-43–Figure 17-46).

17.6.9 Mixed AHB/Memory Burst Modes Support

To provide mixed sequential/wrap accesses with different length, WEIM interprets burst signal and generate additional $\overline{\text{LBA}}$ signals whenever unequal address or burst boundary crossing condition appears (see section 17.6.3/17-25). PSZ field and WRAP bit should be used to notify WEIM about the current memory burst and wrap condition for proper external address generation. In case of non matching boundaries in both the memory and AHB access, WEIM starts a new memory burst access by putting address from AHB on address bus and generating $\overline{\text{LBA}}$ signal. For example, Table 17-20 shows how WEIM interprets with various types of AHB access in the case when memory is configured as 8 beat burst with wrap.

17.6.10 AHB Bus Cycles Support

WEIM uses an ARM AHB slave interface. It has a 32-bit bus and supports four transfer types defined in the AHB specification (IDLE, BUSY, NONSEQ, and SEQ).

NOTE

Only 32-bit accesses are supported for SEQ mode.

It also supports AHB transfers shown in Table 17-19. These AHB cycles will be translated into necessary cycles on the memory side. For optimal operation, ARM cache is configured to 8 beat burst with wrap, a synchronous flash and cellular RAM memory should be configured in 16 word wrap burst mode when

using a 16-bit data port, and in 8 word wrap burst mode when using a 32-bit data port. WEIM uses WRAP bit and PSZ field for support different memory configurations. The controller splits the transaction when needed in some cases (see section Section 17.6.3, “Burst Mode Memory Operation” on page 17-25).

Table 17-19. AHB Burst Cycles Supported

HBURST	TYPE	Supported	Description
000	SINGLE	Yes	Single transfer
001	INCR	Yes	Incrementing burst
010	WRAP4	Yes	4-beat wrapping burst
011	INCR4	Yes	4-beat incrementing burst
100	WRAP8	Yes	8-beat wrapping burst
101	INCR8	Yes	8-beat incrementing burst
110	WRAP16	Yes	16-beat wrapping burst
111	INCR16	Yes	16-beat incrementing burst

For example, Table 17-20 shows AHB bus sequential accesses breaking in to external memory bursts for memory configured to 8 beat burst with wrap and for some different AHB burst types and start addresses. $\overline{\text{LBA}}$(X) means start memory burst access ($\overline{\text{LBA}}$ generation) from address X (all addresses in hex form).

Table 17-20. External Memory Bursts Start Addresses for Some AHB Burst Accesses

AHB Burst Type	Memory Data Port Width	AHB Burst Start Address							
		0	4	8	C	10	14	18	1C
WRAP8	16-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14)	$\overline{\text{LBA}}$(18)	$\overline{\text{LBA}}$(1C)
		$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(0)
			$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(0)		$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)
	32-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14)	$\overline{\text{LBA}}$(18)	$\overline{\text{LBA}}$(1C)
INCR8	16-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14)	$\overline{\text{LBA}}$(18)	$\overline{\text{LBA}}$(1C)
		$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)
			$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)		$\overline{\text{LBA}}$(30)	$\overline{\text{LBA}}$(30)	$\overline{\text{LBA}}$(30)
	32-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14)	$\overline{\text{LBA}}$(18)	$\overline{\text{LBA}}$(1C)
			$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)
WRAP4	16-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14)	$\overline{\text{LBA}}$(18)	$\overline{\text{LBA}}$(1C)
	32-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14)	$\overline{\text{LBA}}$(18)	$\overline{\text{LBA}}$(1C)
			$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(0)		$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)
INCR4	16-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14)	$\overline{\text{LBA}}$(18)	$\overline{\text{LBA}}$(1C)
			$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)		$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)
	32-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14)	$\overline{\text{LBA}}$(18)	$\overline{\text{LBA}}$(1C)
							$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(20)

Some examples are shown in Figure 17-32 to Figure 17-39.

17.6.11 DTACK Mode

It is a mode where WEIM timing depends on $\overline{\text{DTACK}}$ input signal. This signal may be used in two ways: by posedge sensitive or by level sensitive (with an initial insensitiveness time). Posedge sensitive mode is set by WSC=111111 (EW=0) and selects $\overline{\text{DTACK}}$ input as access length control sign (instead of default WSC counter). It means that access length is determined by $\overline{\text{DTACK}}$ length. WEIM begins deasserting control signals after approximately 1.5 clock (synchronization delay) in the sequence according to negation control fields.

NOTE

It may be required to program CSA and/or CSN fields for a correct word access to 16 or 8-bit port in this mode if corresponding module is CS sensitive. CSN maximum value is 6 for this case.

Level sensitive mode is set by EW=1 (WSC < 111111). The access length is controlled by WSC. In this case, WEIM begins access (by CS assertion) and after some clocks (according to DCT field) checks $\overline{\text{DTACK}}$ input. If $\overline{\text{DTACK}}$ is low, WEIM waits for $\overline{\text{DTACK}}$ high state and reload wait state counter (see Figure 17-27 and Figure 17-28). For sequential AHB accesses, where CS doesn't negates during burst, $\overline{\text{DTACK}}$ is being checked on the first access only. Glue logic Enabled by AGE bit of the CSCRxA register can be used for noisy or slowly rising $\overline{\text{DTACK}}$. Refer AGE bit description for more details.

17.6.12 Internal Input Data Capture

In typical case, input data is not sampled by WEIM but it is sampled by AHB master on the rising edge of HCLK when HREADY is high. WEIM assert HREADY signal to AHB master (according to WSC or DOL count). This allows better performance on the data path. There are 2 cases by which input data gets sampled inside the WEIM.

First one is when an access size is larger then a port size. In this case, WEIM samples all Data coming from the memory device except the last one. For example, if there is a word access to the byte wide memory, WEIM captures first three input bytes internally and drive them together with the last byte to AHB master (last byte is not sampled in WEIM). WEIM captures data by rising edge of HCLK when WSC (or DOL if it is a part of burst) time expires and (if it depends) suitable $\overline{\text{ECB}}$ or $\overline{\text{DTACK}}$ input condition.

Second case is when a feedback clock is used (FCE=1) for synchronous burst. Data will be sampled on the rising edge of the feedback clock (when WSC or DOL time expires and input condition kept) and then those captured data is again sampled by HCLK before it will be driven to the AHB master.

17.6.13 Error Conditions

The following conditions cause an error signal:

- Access to a disabled chip select (access to a mapped chip select address space where CSEN bit in the corresponding chip select control register is clear)
- Write access to a write-protected chip select address space (WP bit in the corresponding chip select control register is set)
- User access to a supervisor-protected chip select address space (the SP bit in the corresponding chip select control register is set)

- User read or write access to a chip select control register or the WEIM configuration register
- Byte or halfword access to a chip select control register or the WEIM configuration register
- $\overline{\text{DTACK}}$ acknowledge is absent more then 1024 clock
- WAIT deassertion more then 1024 clock.

17.7 Initialization/Application Information

WEIM is ready to work with $\overline{\text{CS0}}$ after hardware reset, but it has been configured for very slowly access (for boot purpose) without additional setup and hold time. Other $\overline{\text{CS}}$ are disabled by hardware reset. So any $\overline{\text{CS}}$ has to be properly initialized before using it by writing values to high and low configuration register. Example 17-1 shows how to prepare WEIM and 16-bit flash memory to work in the synchronous mode.

Example 17-1. WEIM and Flash Memory Initialization for Work in Synchronous Mode

```
@; config WEIM to Async access with EDC, OEA, RWA, RWN, EBC, 16 bit port and PSR
         WRITE WEIM_CSCR2U, 0x12020802
         WRITE WEIM_CSCR2L, 0x80330d03

@ ; config flash to WRAP 8 mode (by half word accesses)
         WRITE_H (CS2_BASE_ADDR+0x2384), 0x60    @ ; offset = 0x11c2 << 1 for 16 bit port
         WRITE_H (CS2_BASE_ADDR+0x2384), 0x03

         WRITE_H (CS2_BASE_ADDR+0x0), 0xff       @ ; flash to read mode

@ ; config to WEIM Sync access with WRAP8, 16 bit port
         WRITE WEIM_CSCR2U, 0x13510802
         WRITE WEIM_CSCR2L, 0x80330d03
```

17.8 External Bus Timing Diagrams

The following timing diagrams shows access timing to memory or a peripheral with different timing parameters. All examples are for $\overline{\text{CS0}}$, but they are same for any others chip select. $\overline{\text{EB}}$ means one from current used $\overline{\text{EB}}$[3:0]

For asynchronous mode:

- Figure 17-7 to Figure 17-13 shows halfword read and write accesses to halfword-width memory.
- Figure 17-14 to Figure 17-24 shows word read and write accesses to halfword-width memory.
- Figure 17-25 shows page mode timing diagram, Figure 17-26 to Figure 17-28 shows two kinds of $\overline{\text{DTACK}}$ accesses.
- Figure 17-43 and Figure 17-44 shows asynchronous data exchange in multiplexed address/data mode with word-width memory.

For synchronous (burst) mode:

- Figure 17-29 shows synchronous word read accesses to halfword-width memory.
- Figure 17-30 shows recommended parameter setting using for synchronous accesses: BCLK clock has a short pulse at the end of access.

- Different cases of wrap/increment states on AHB and memory are shown in the Figure 17-32 through Figure 17-39 for burs size 4. AHB increment/wrap accesses with another length are made like this.
- PSRAM read and write accesses are shown in the Figure 17-40 and Figure 17-41.
- Figure 17-45 and Figure 17-46 show synchronous data exchange in multiplexed address/data mode with word-width memory.

17.8.1 Asynchronous Memory Accesses Timing Diagrams

17.8.1.1 AHB Halfword Access to Halfword Width Memory

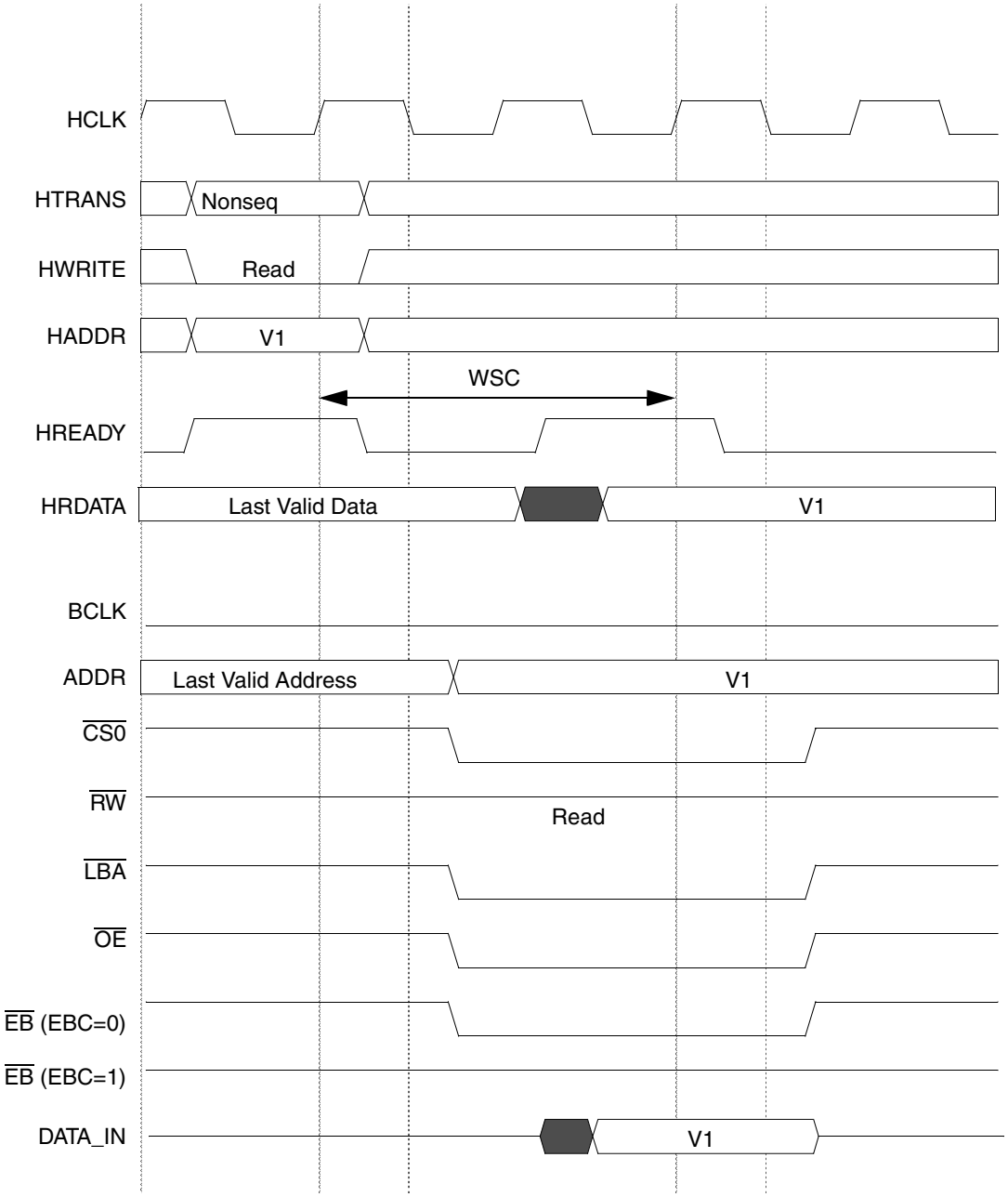


Figure 17-7. Read Access, WSC=1

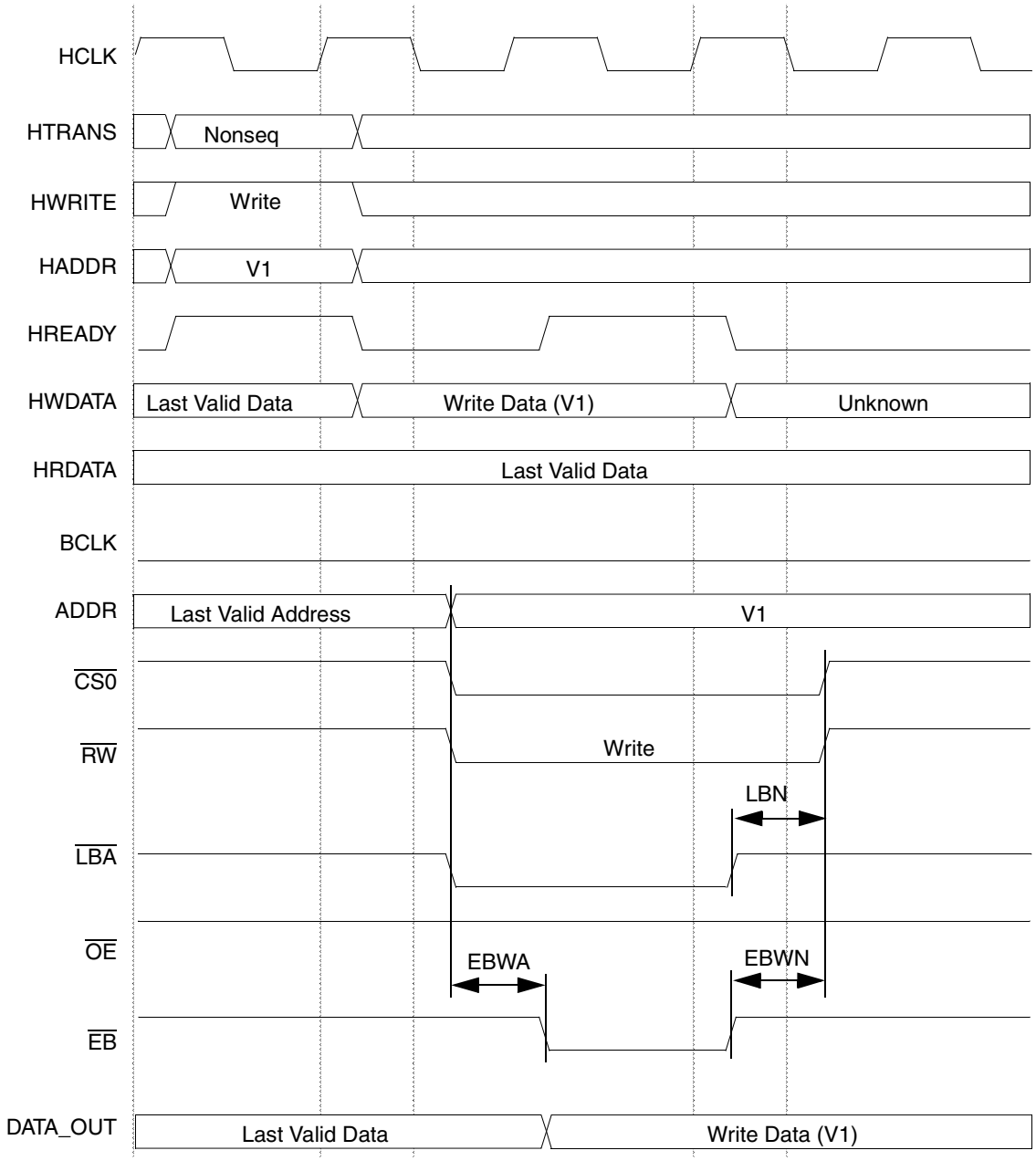


Figure 17-8. Write Access, WSC=1, EBWA=1, EBWN=1, LBN=1

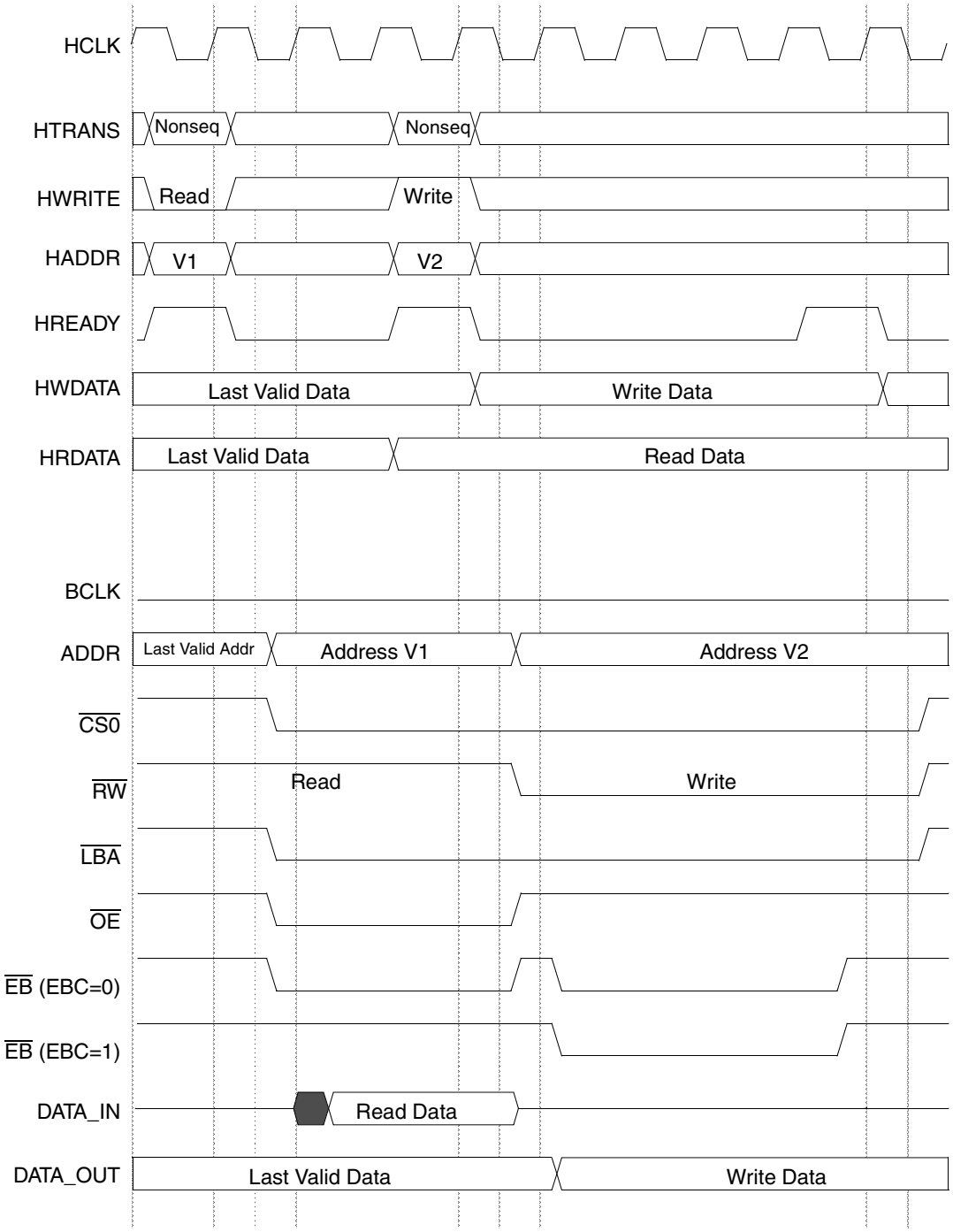


Figure 17-9. Read and Write Accesses, WSC=2, WWS=2, EBWA=1, EBWN=2

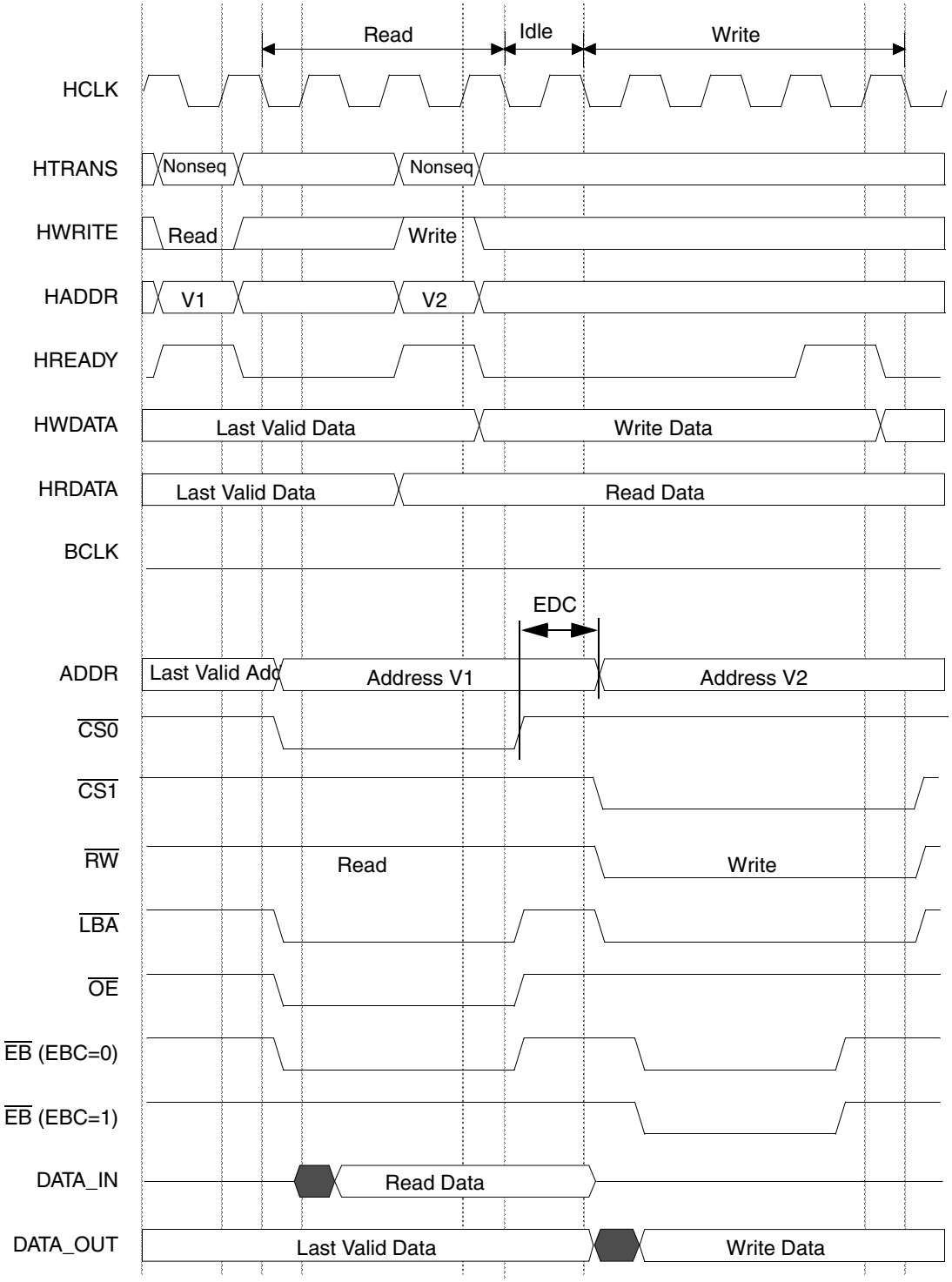


Figure 17-10. Read and Write Accesses, WSC=2, WWS=1, EBWA=1, EBWN=2, EDC=1

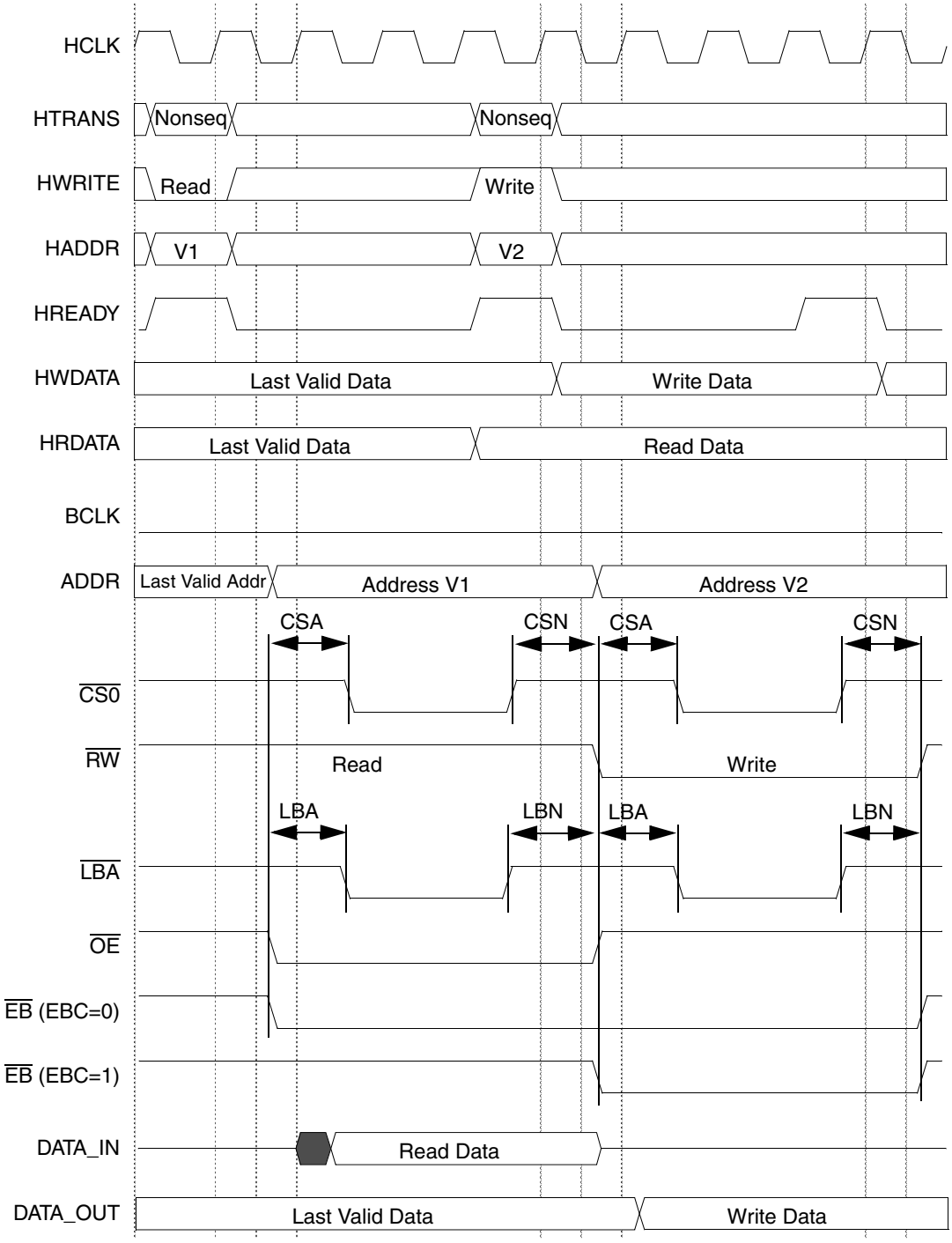


Figure 17-11. Read and Write Accesses, WSC=3, CSA=1, CSN=1, LBA=1, LBN=1

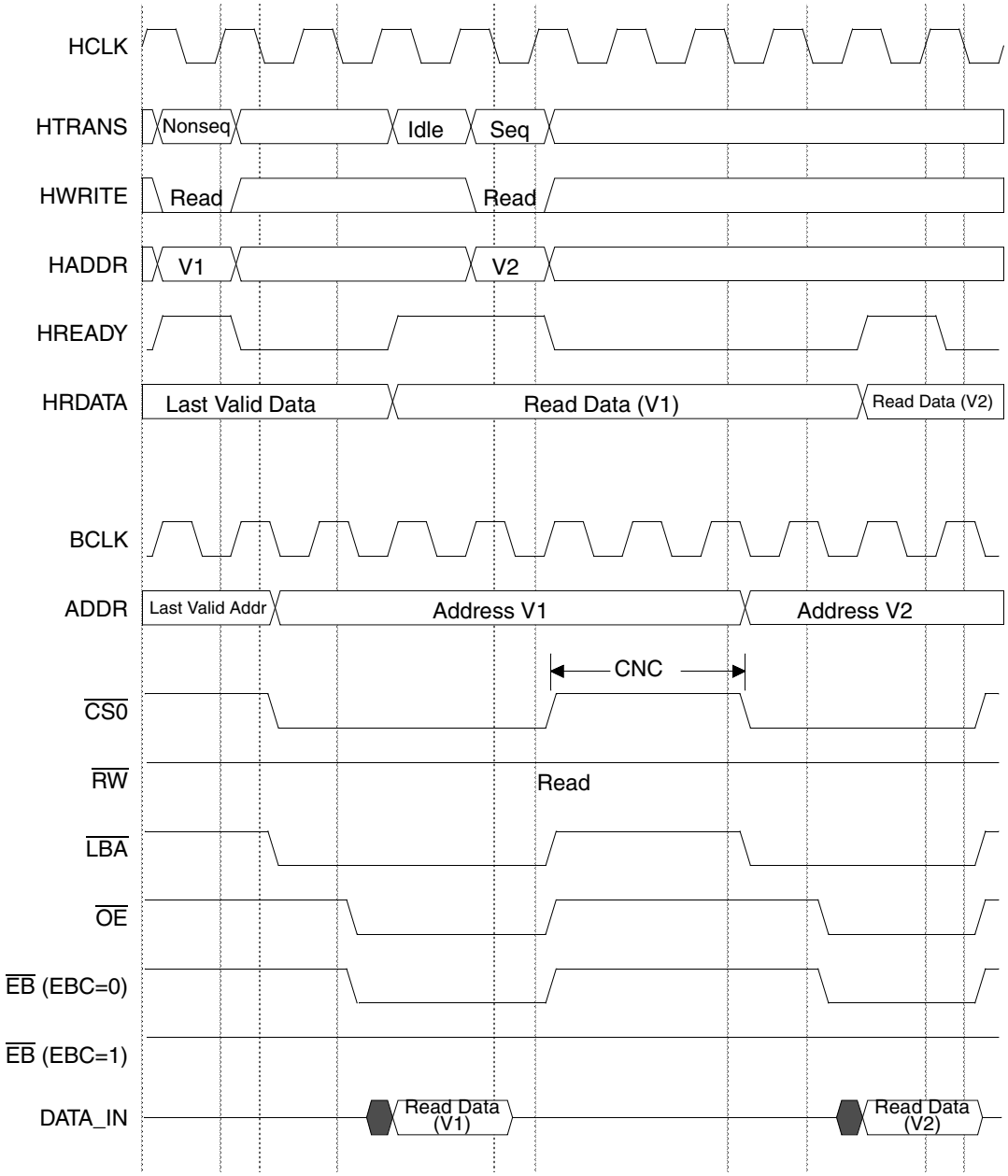


Figure 17-12. Read Accesses, WSC=2, OEA=2, CNC=2, BCM=1, EBRA=2

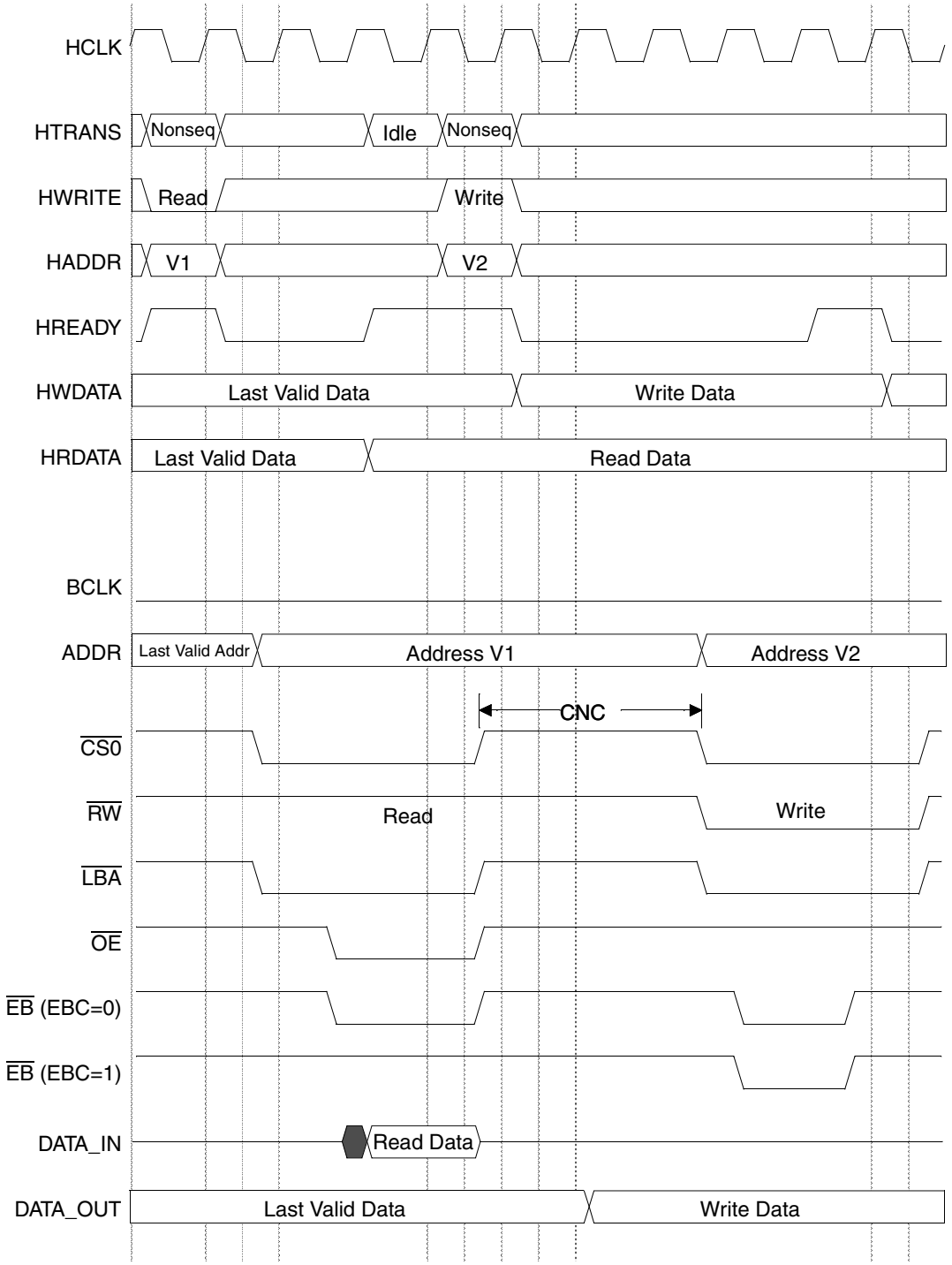


Figure 17-13. Read and Write Accesses, WSC=2, OEA=2, EBWA=1, EBWN=2, CNC=2, EBRA=2

17.8.1.2 AHB Word Access to Halfword Width Memory

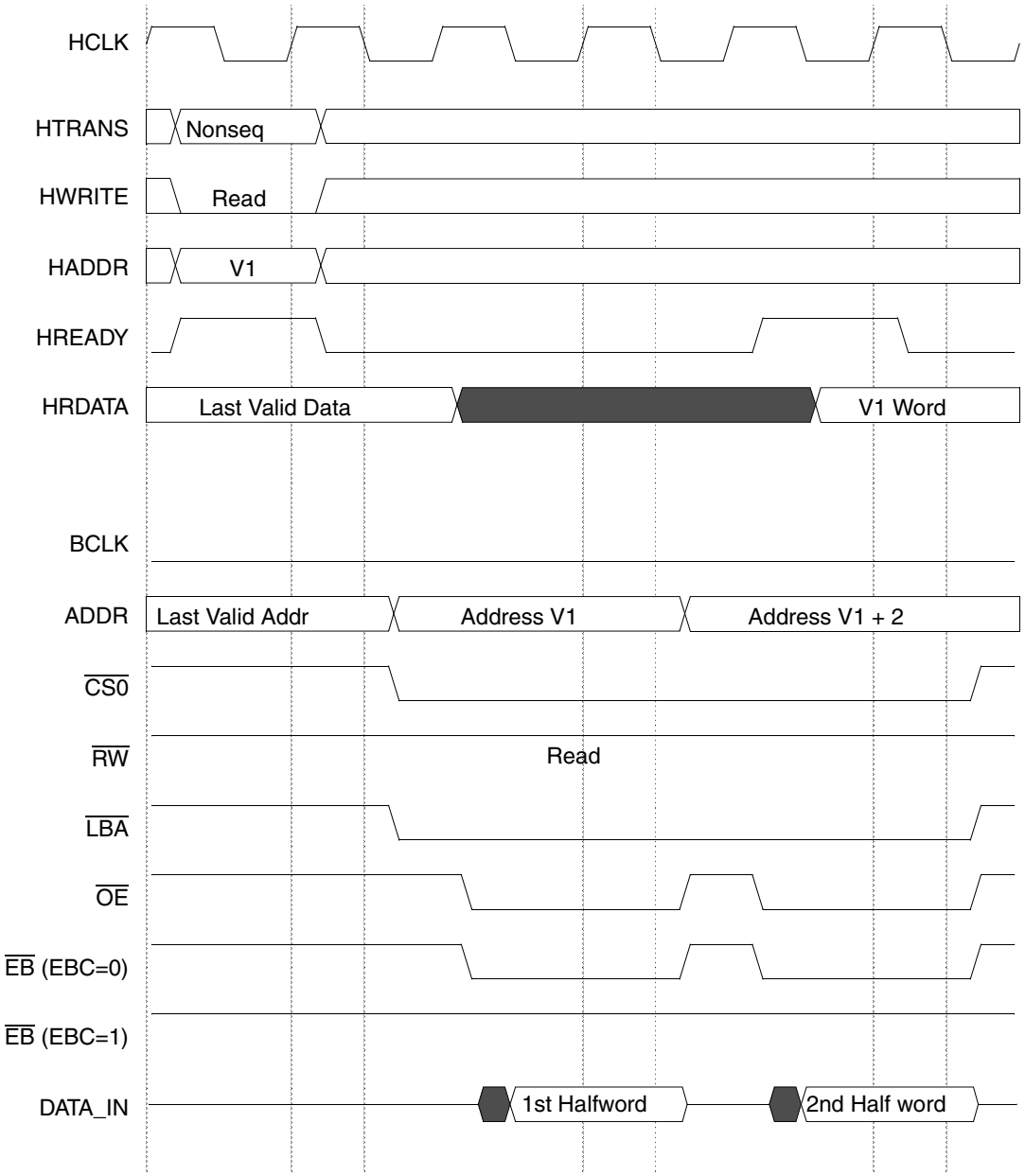


Figure 17-14. Read Access, WSC=1, OEA=1, EBRA=1

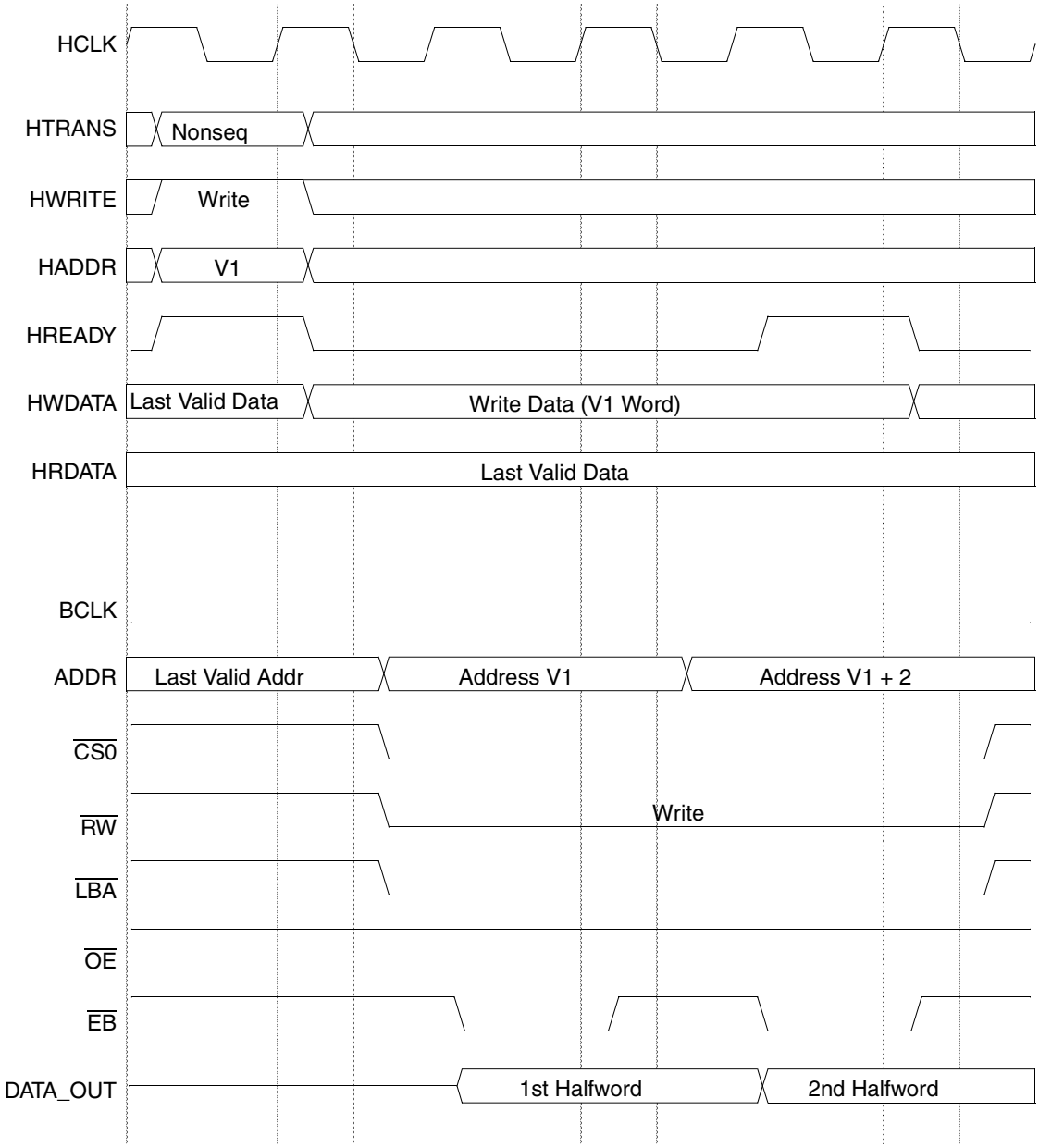


Figure 17-15. Write Access, WSC=1, EBWA=1, EBWN=1

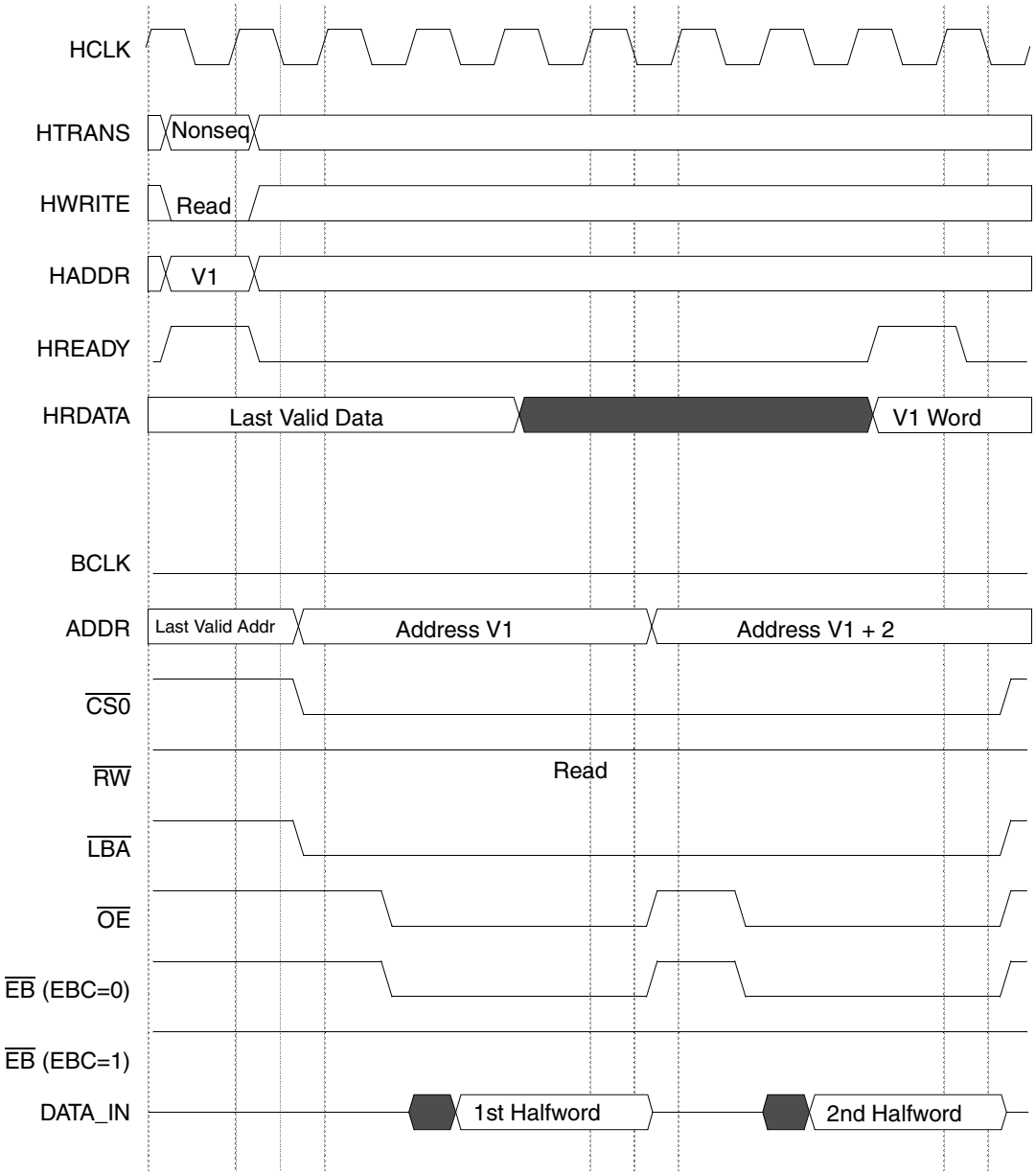


Figure 17-16. Read Access, WSC=3, OEA=2, EBRA=2

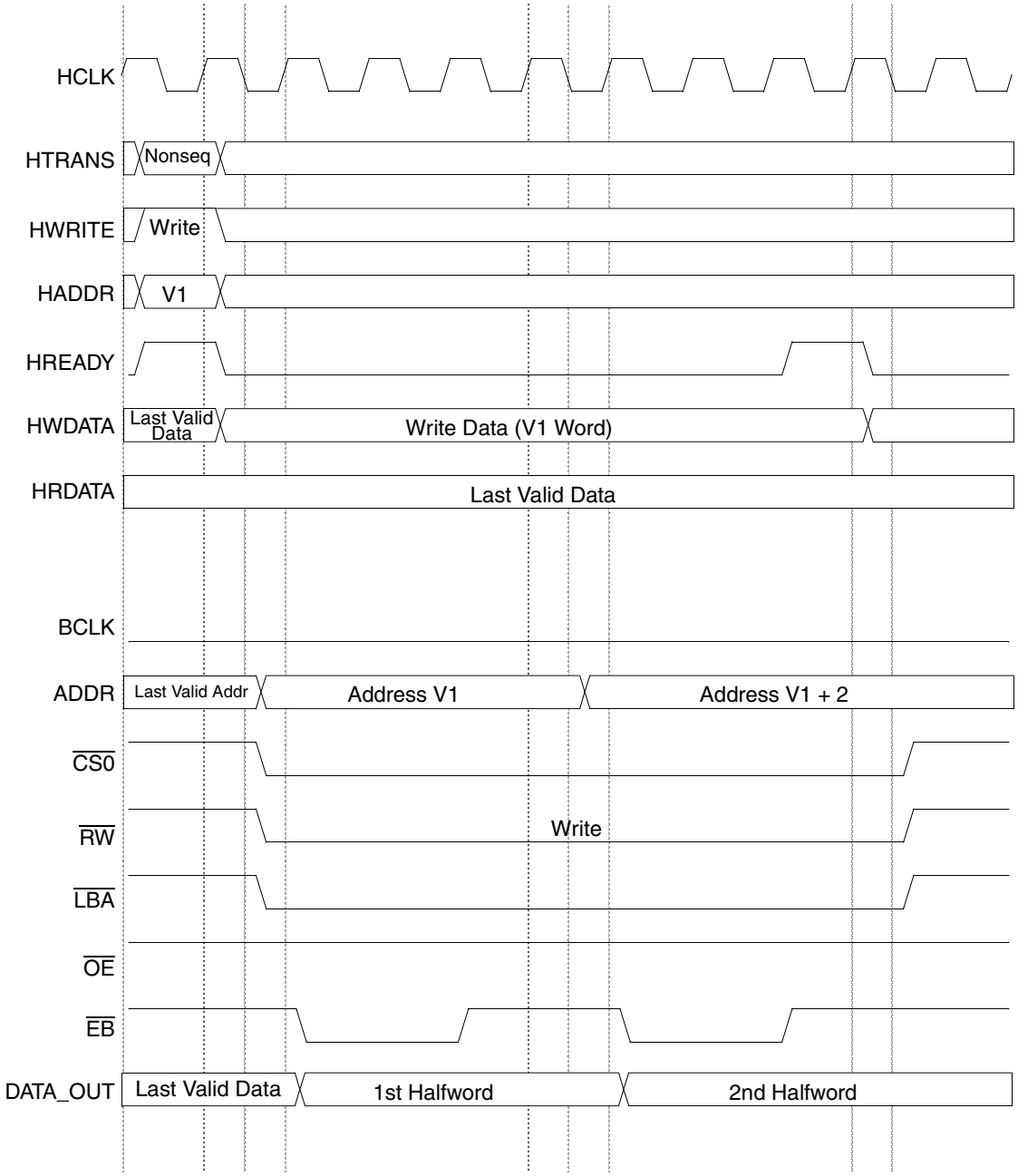


Figure 17-17. Write Access, WSC=3, EBWA=1, EBWN=3

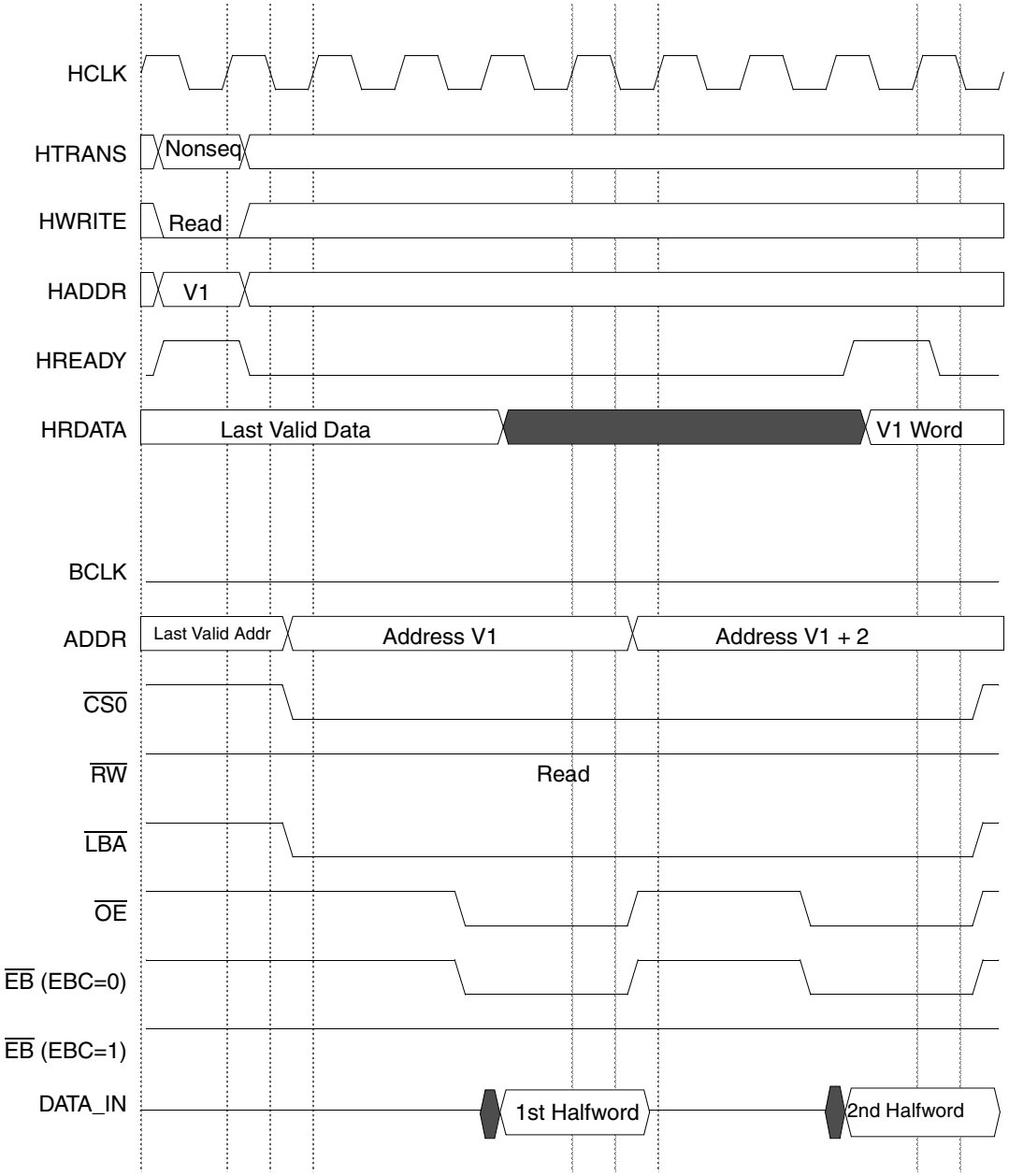


Figure 17-18. Read Access, WSC=3, OEA=4, EBRA=4

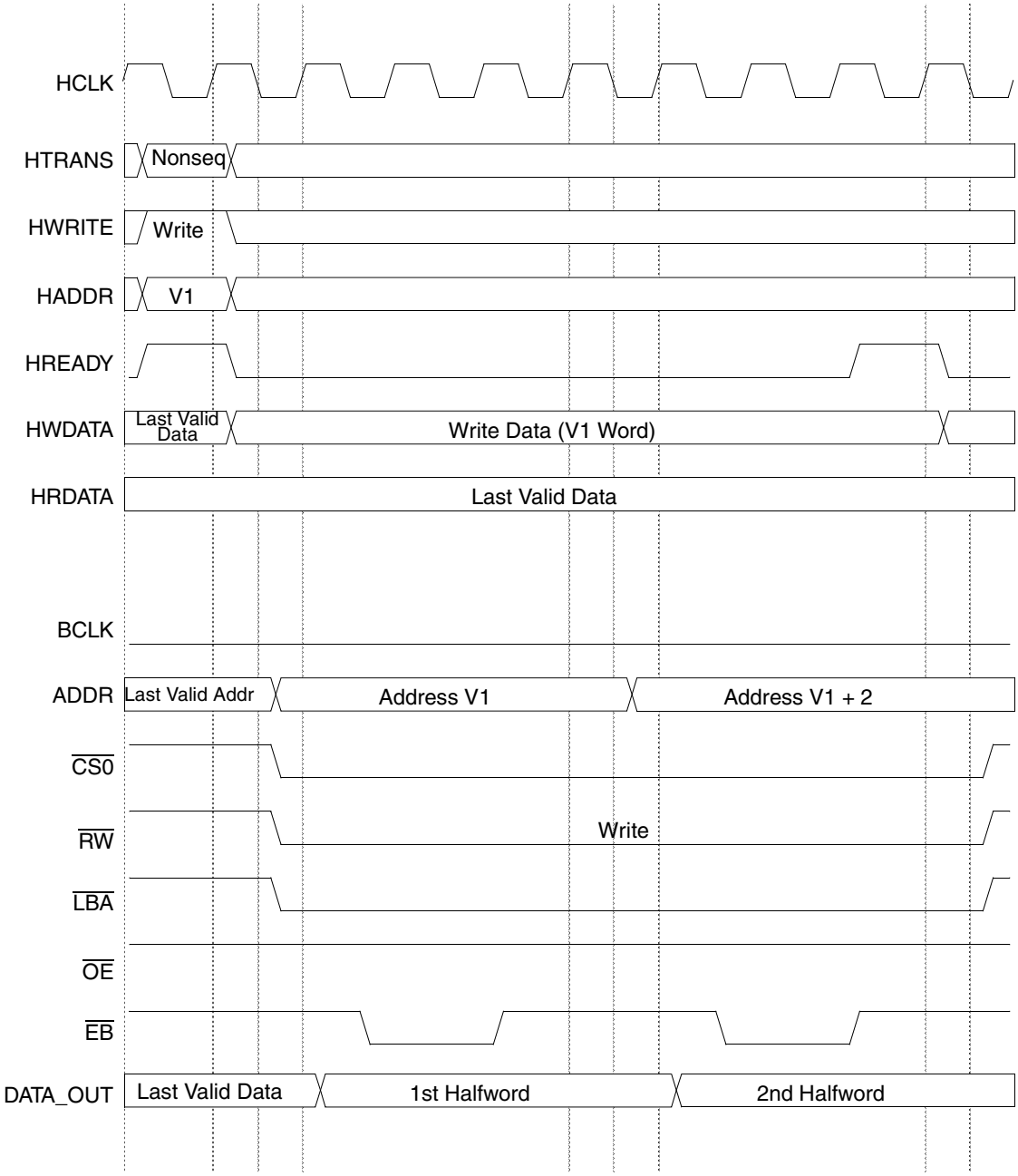


Figure 17-19. Write Access, WSC=3, EBWA2, EBWN=3

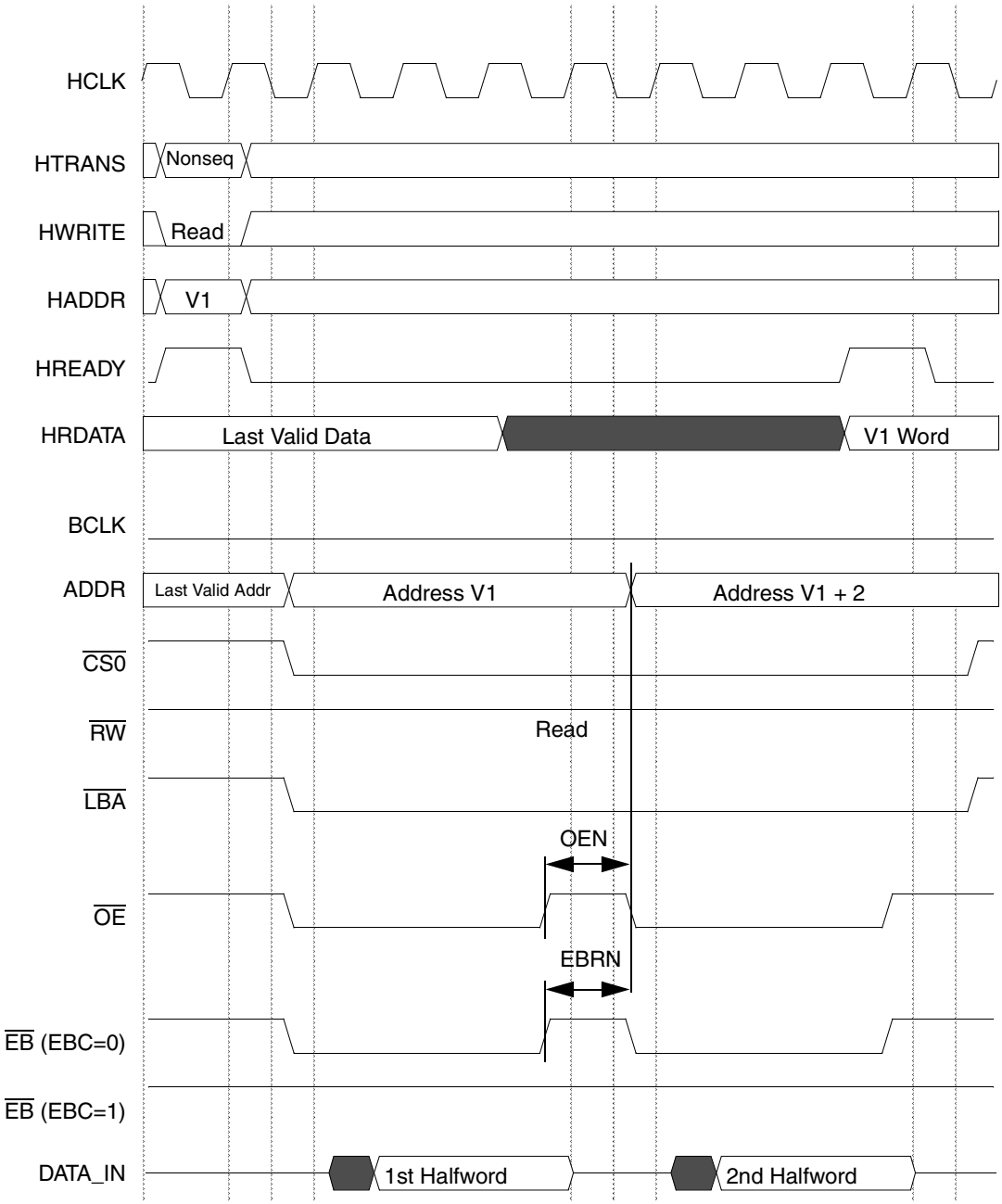


Figure 17-20. Read Access, WSC=3, OEN=2, EBRN=2

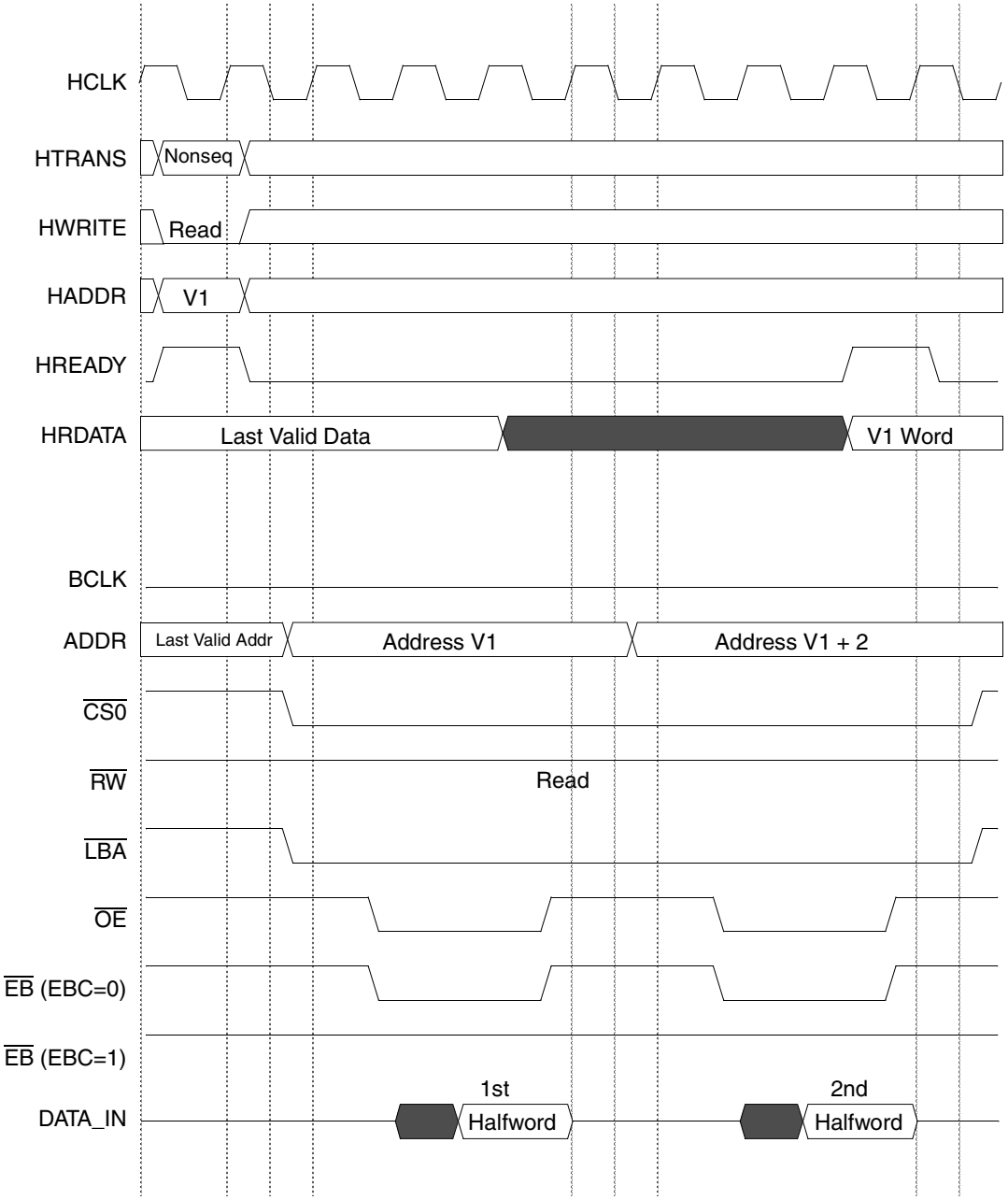


Figure 17-21. Read Access, WSC=3, OEA=2, OEN=2, EBRA=2, EBRN=2

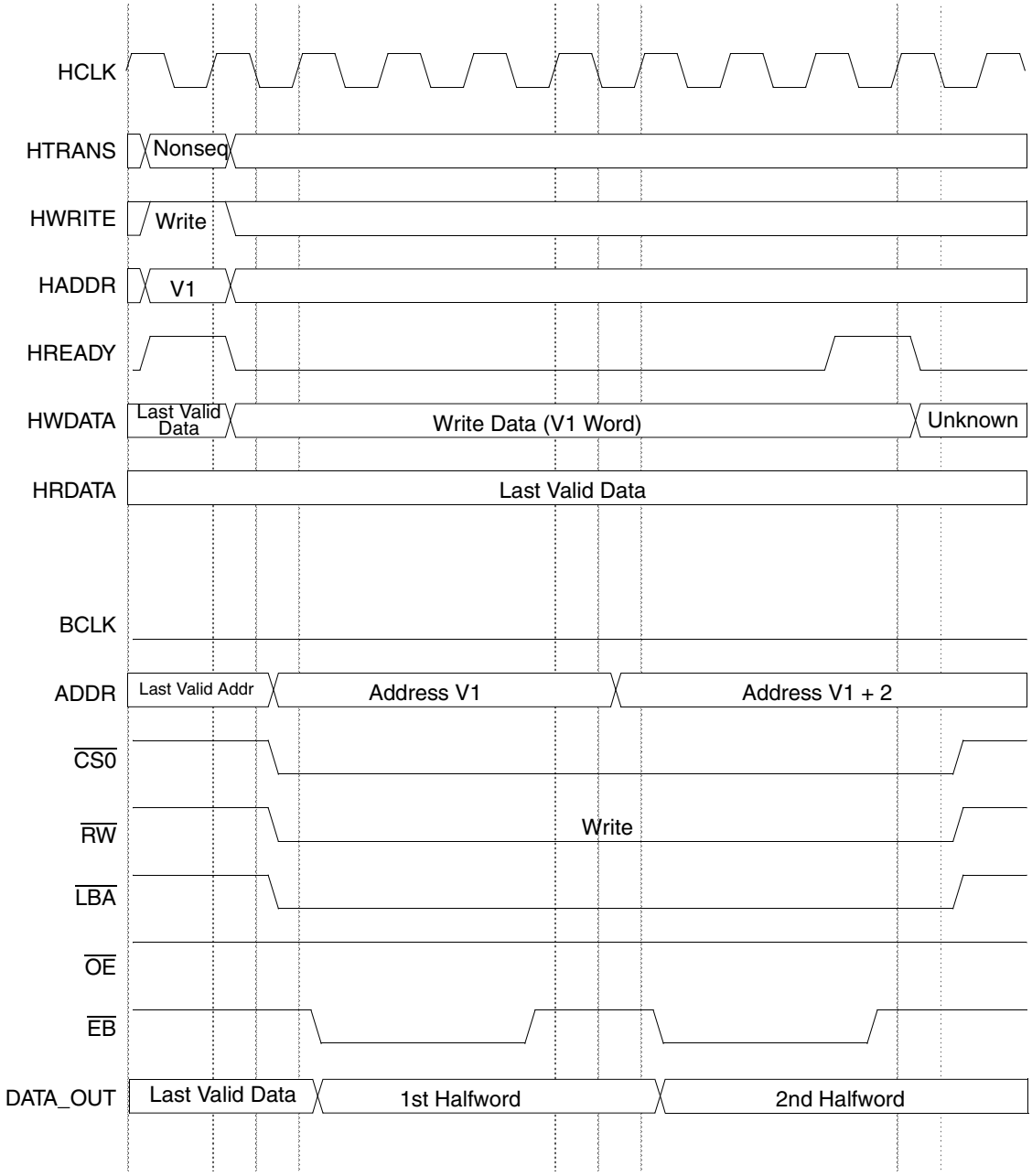


Figure 17-22. Write Access, WSC=2, WWS=1, EBWA=1, EBWN=2

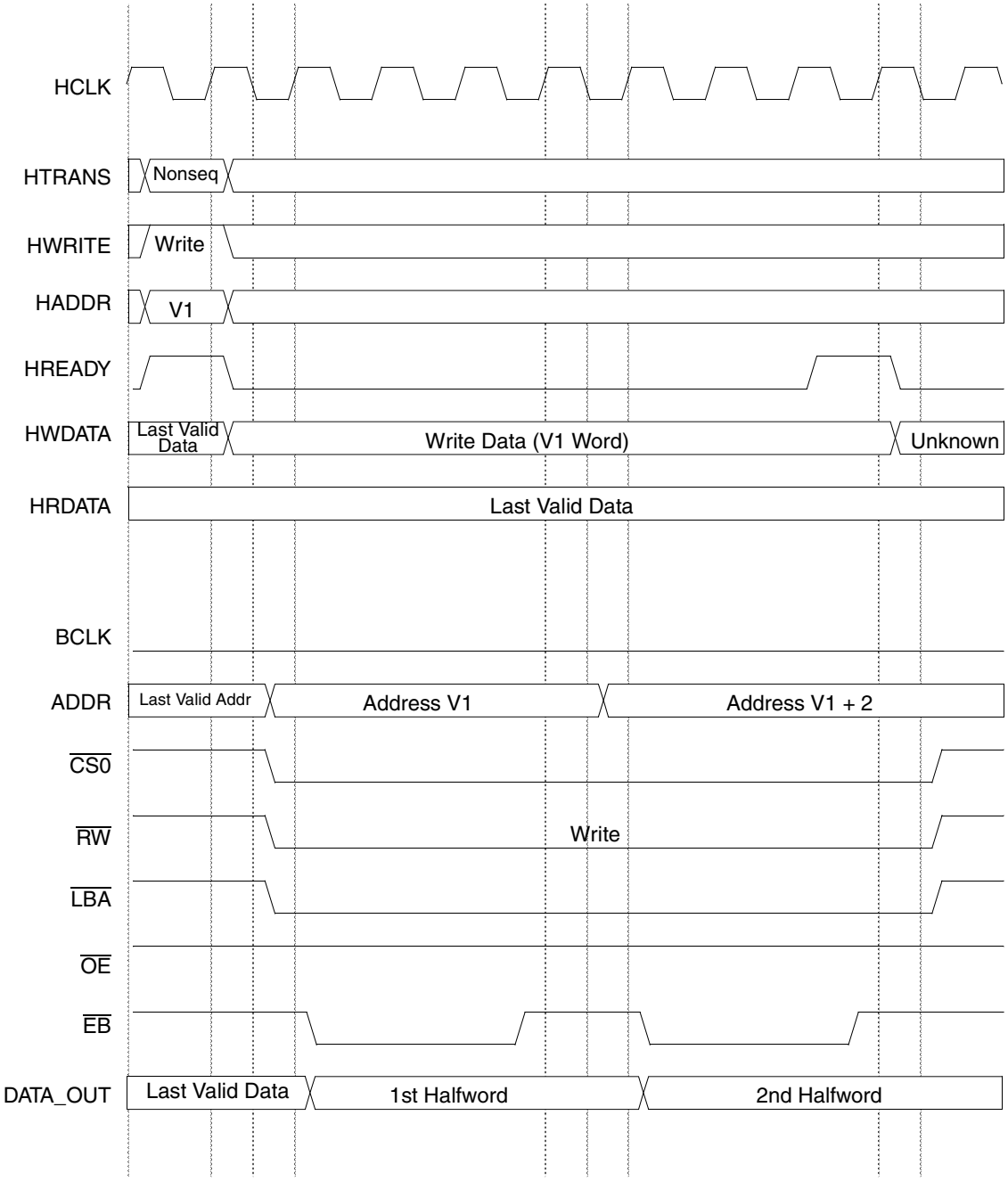


Figure 17-23. Write Access, WSC=1, WWS=2, EBWA=1, EBWN=2

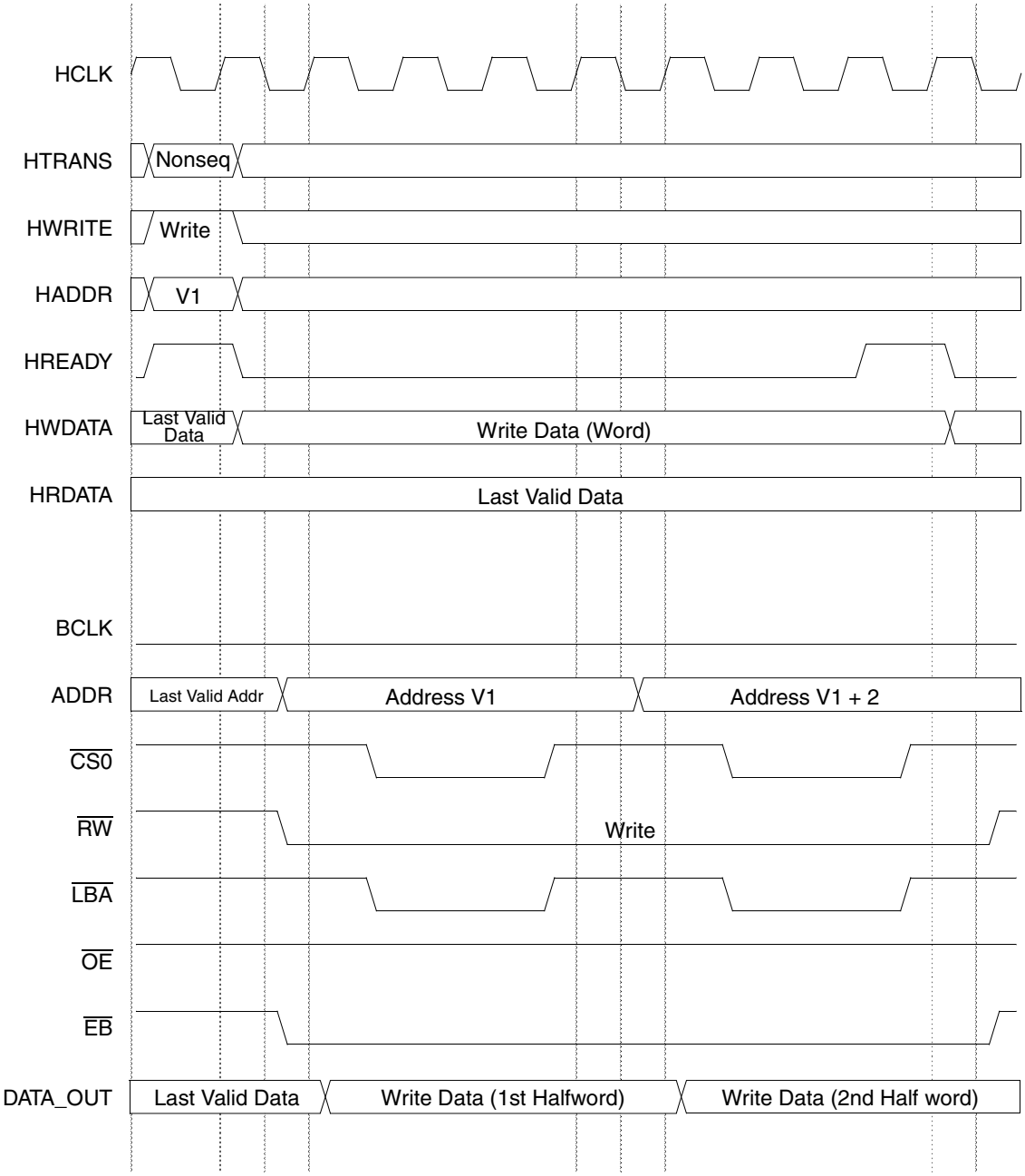


Figure 17-24. Write Access, WSC=2, CSA=1, WWS=1, CSN=1

17.8.2 Page Mode Timing Diagrams

17.8.2.1 AHB Word Accesses to Halfword Width Memory

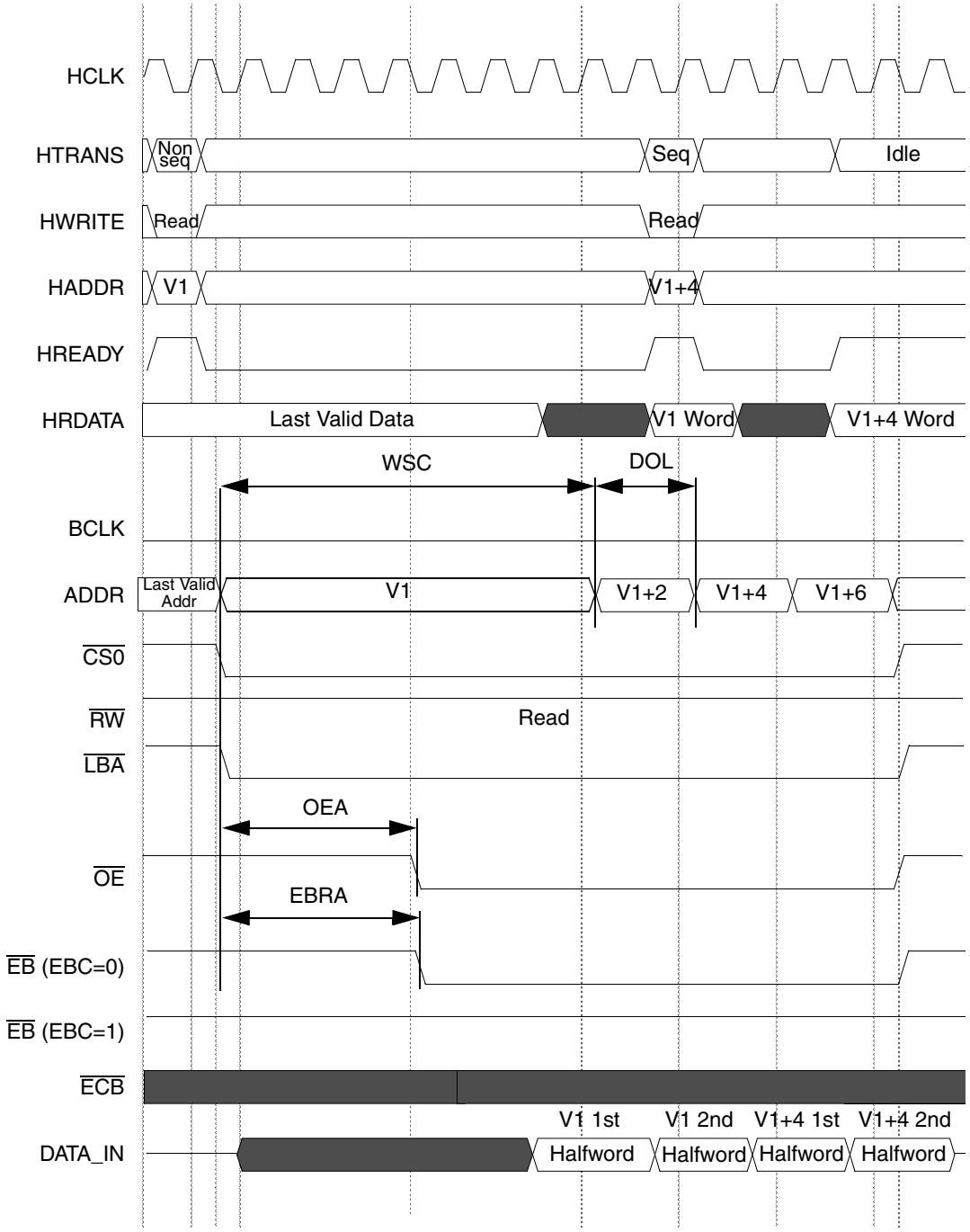


Figure 17-25. Sequential Read Access, WSC=7, OEA=8, PME=1, SYNC=1, DOL=1, EBRA=8

17.8.3 DTACK Mode Memory Accesses Timing Diagrams

17.8.3.1 AHB Word Accesses to Word-Width Memory

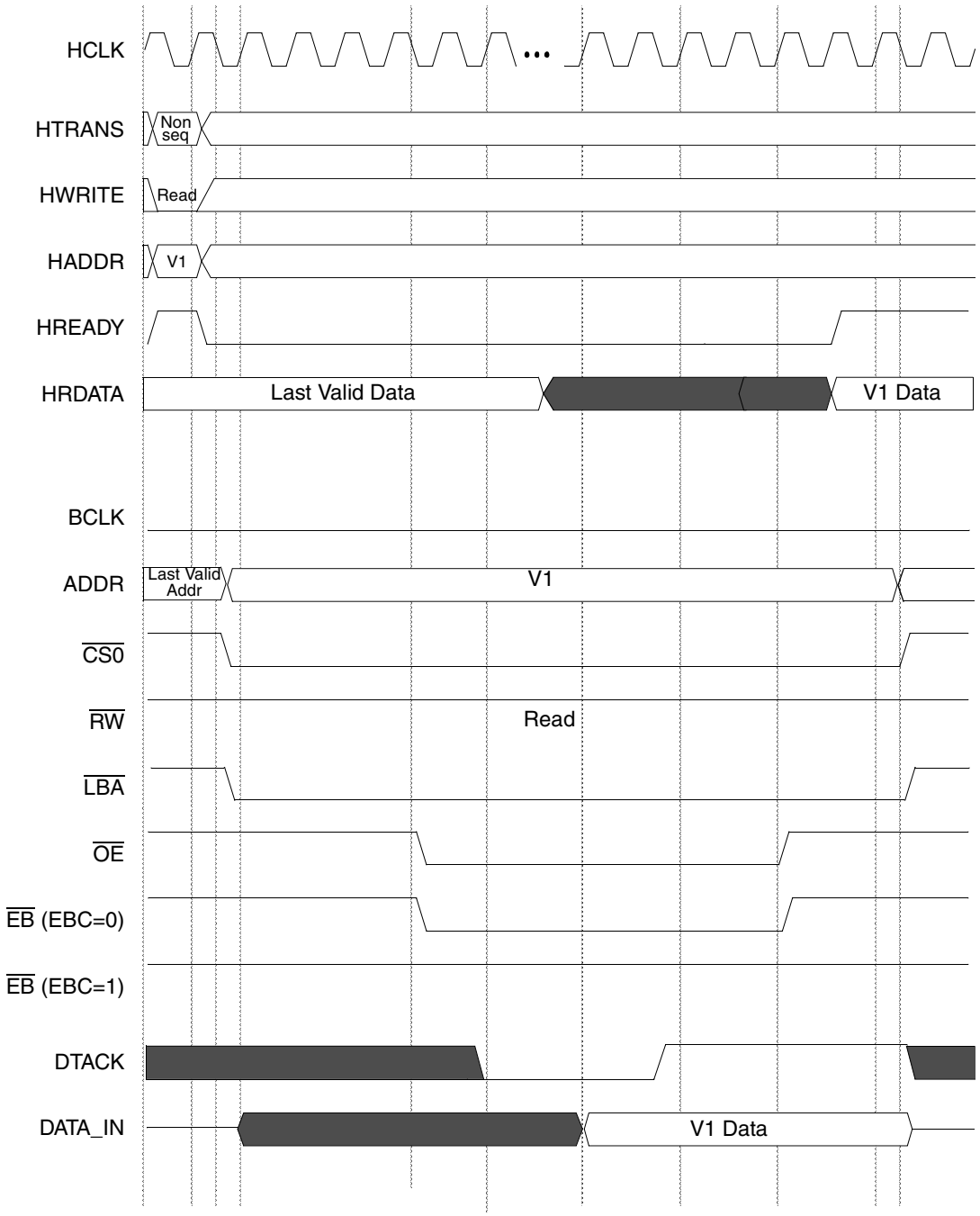


Figure 17-26. Read Access, WSC=3F, OEA=8, OEN=5, EBRA=8, EBRN=5

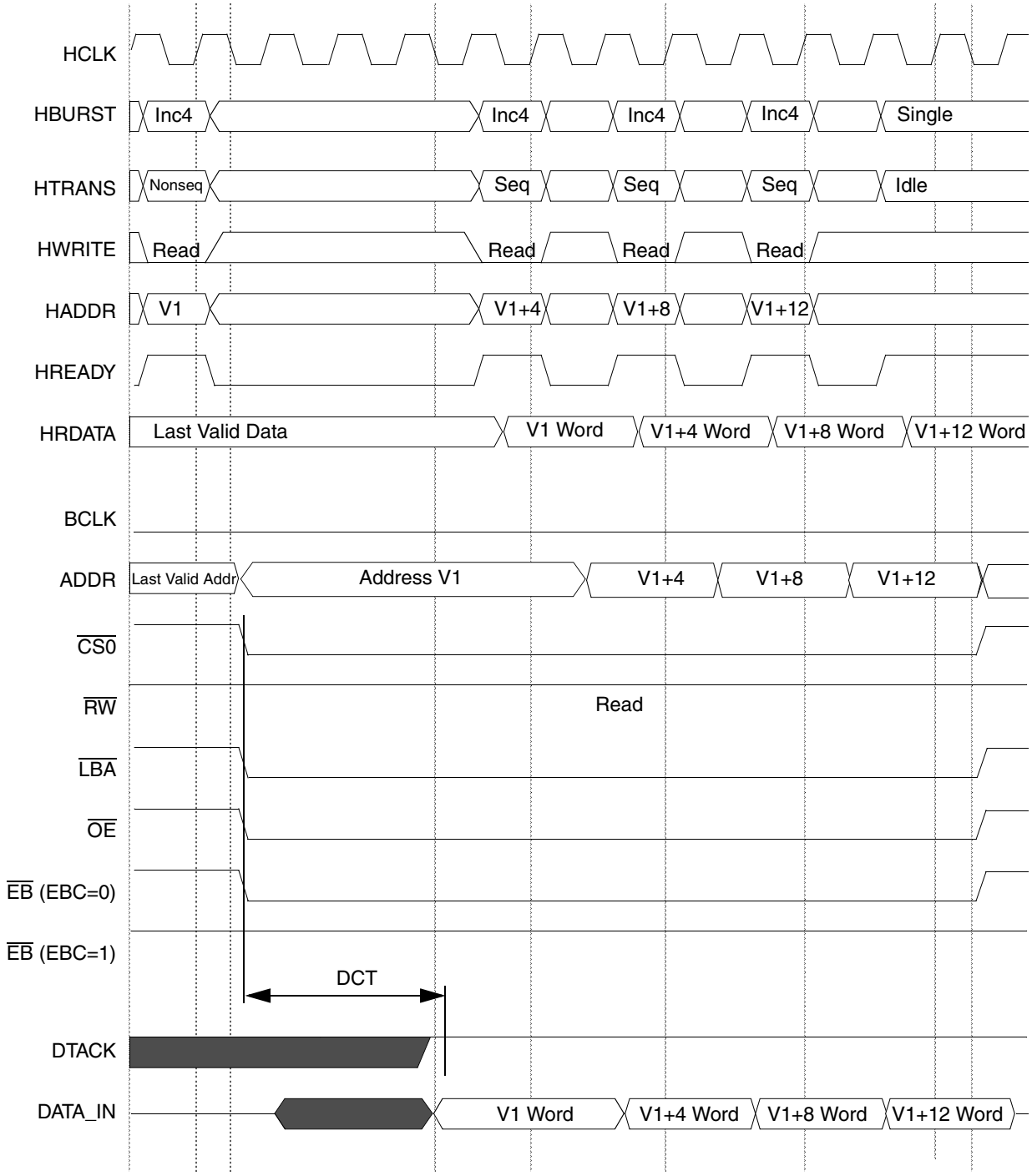


Figure 17-27. Sequential Read Accesses, WSC=1, EW=1, DCT=1

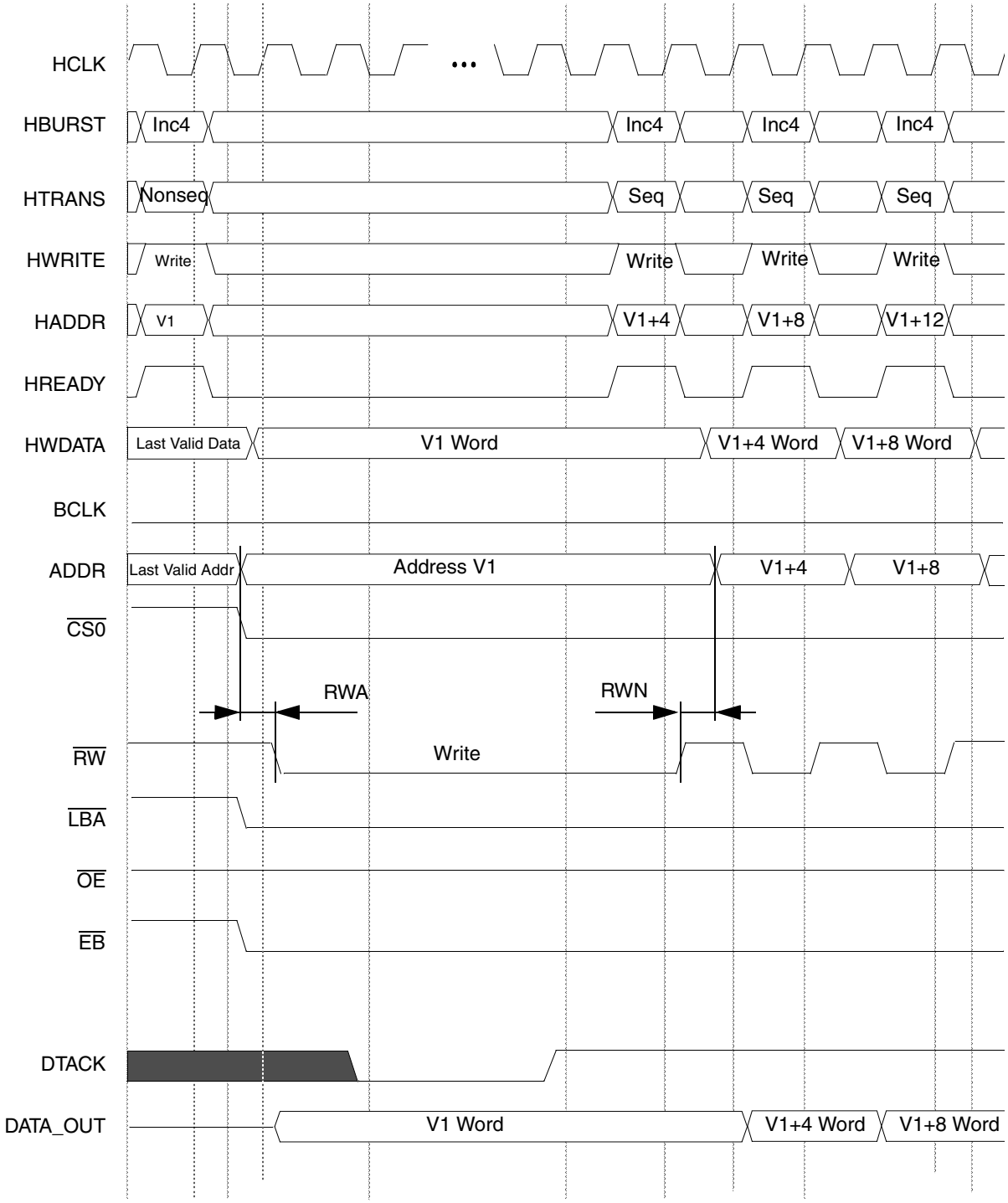


Figure 17-28. Sequential Write Accesses, WSC=1, EW=1, RWA=1, RWN=1

17.8.4 Burst Memory Accesses Timing Diagrams

17.8.4.1 AHB Word Accesses to Halfword Width Memory

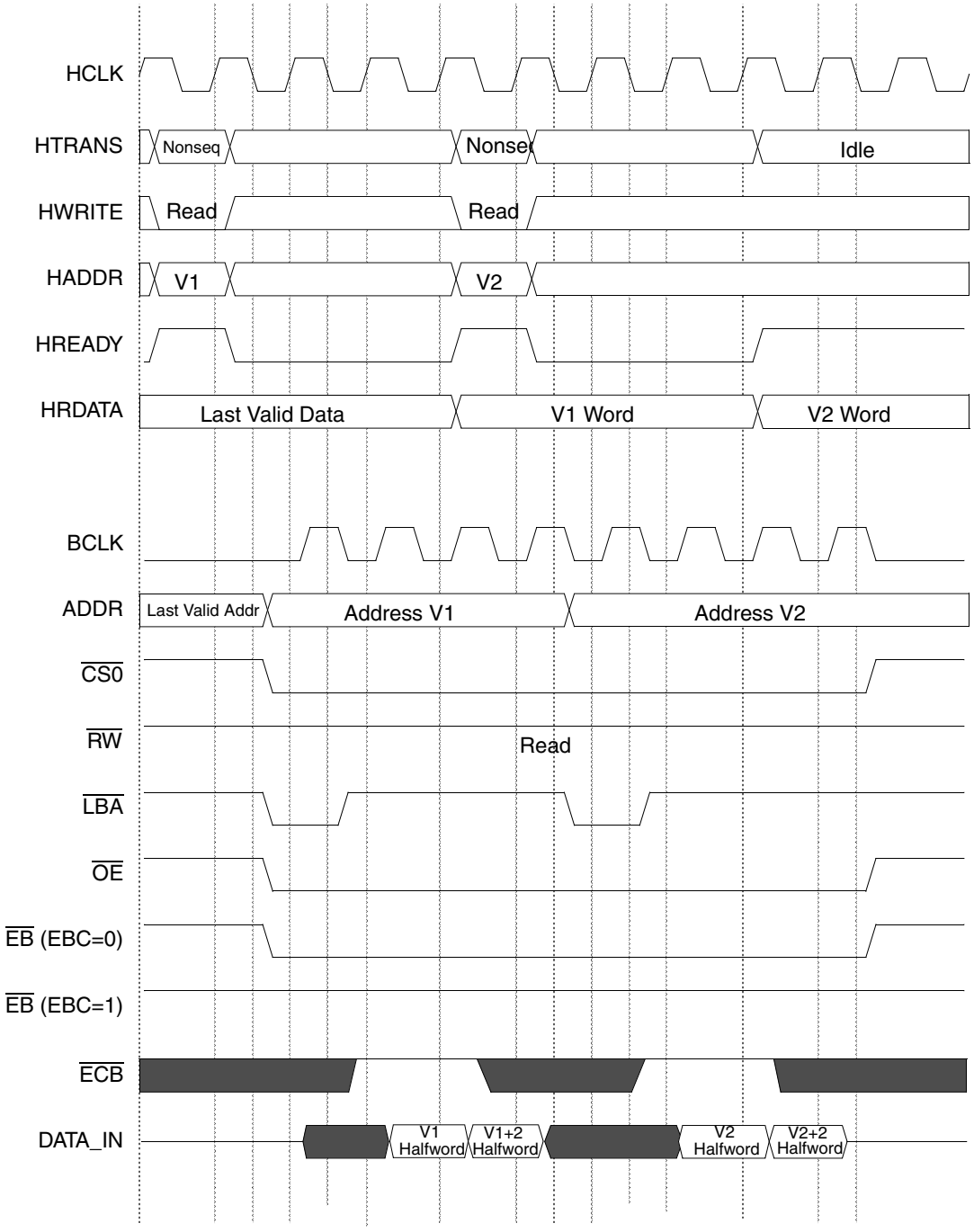


Figure 17-29. Non-sequential Read Accesses, WSC=2, SYNC=1, DOL=0

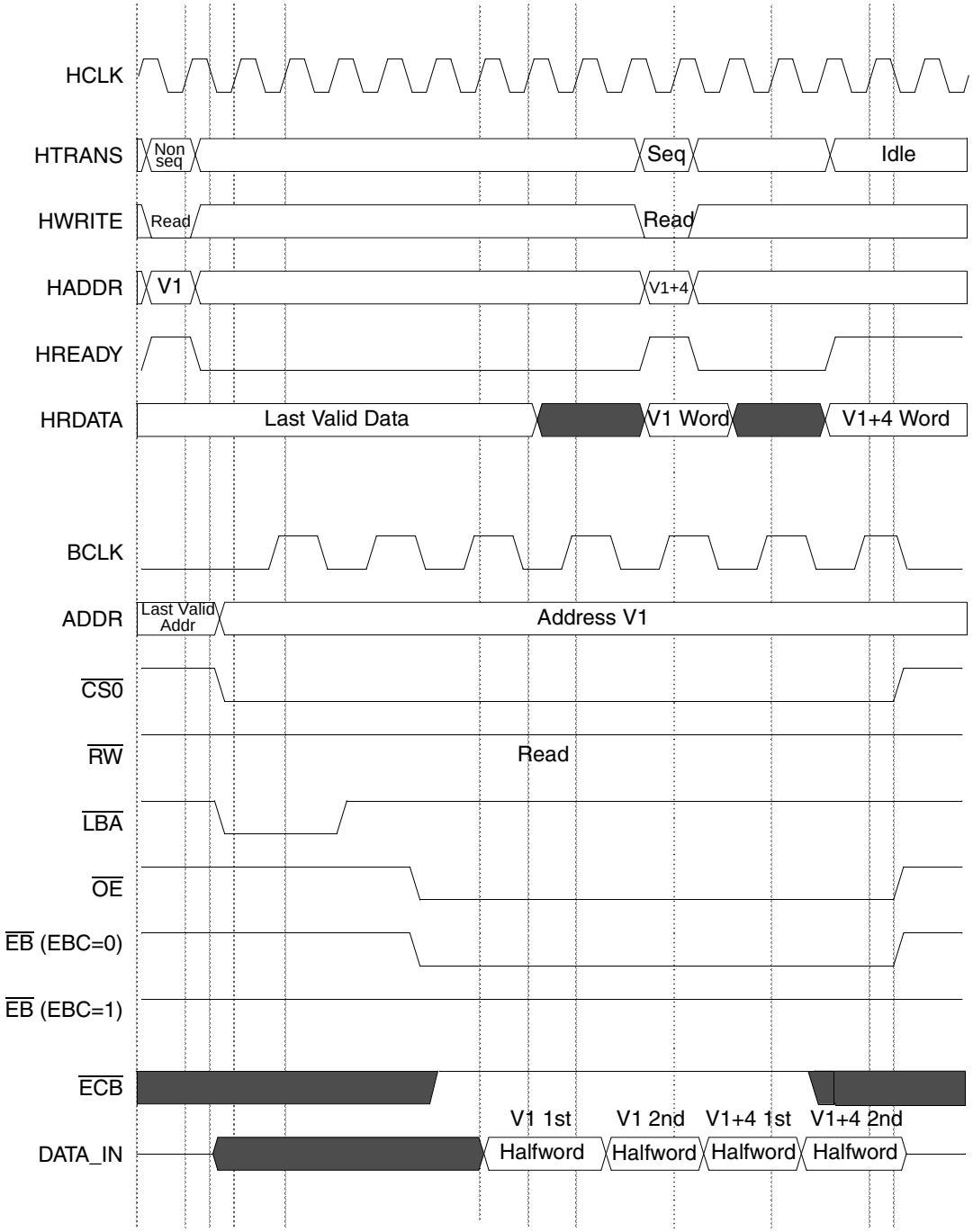


Figure 17-30. Sequential Read Access, WSC=7, OEA=8, SYNC=1, DOL=1, BCD=1, BCS=1, EBRA=8

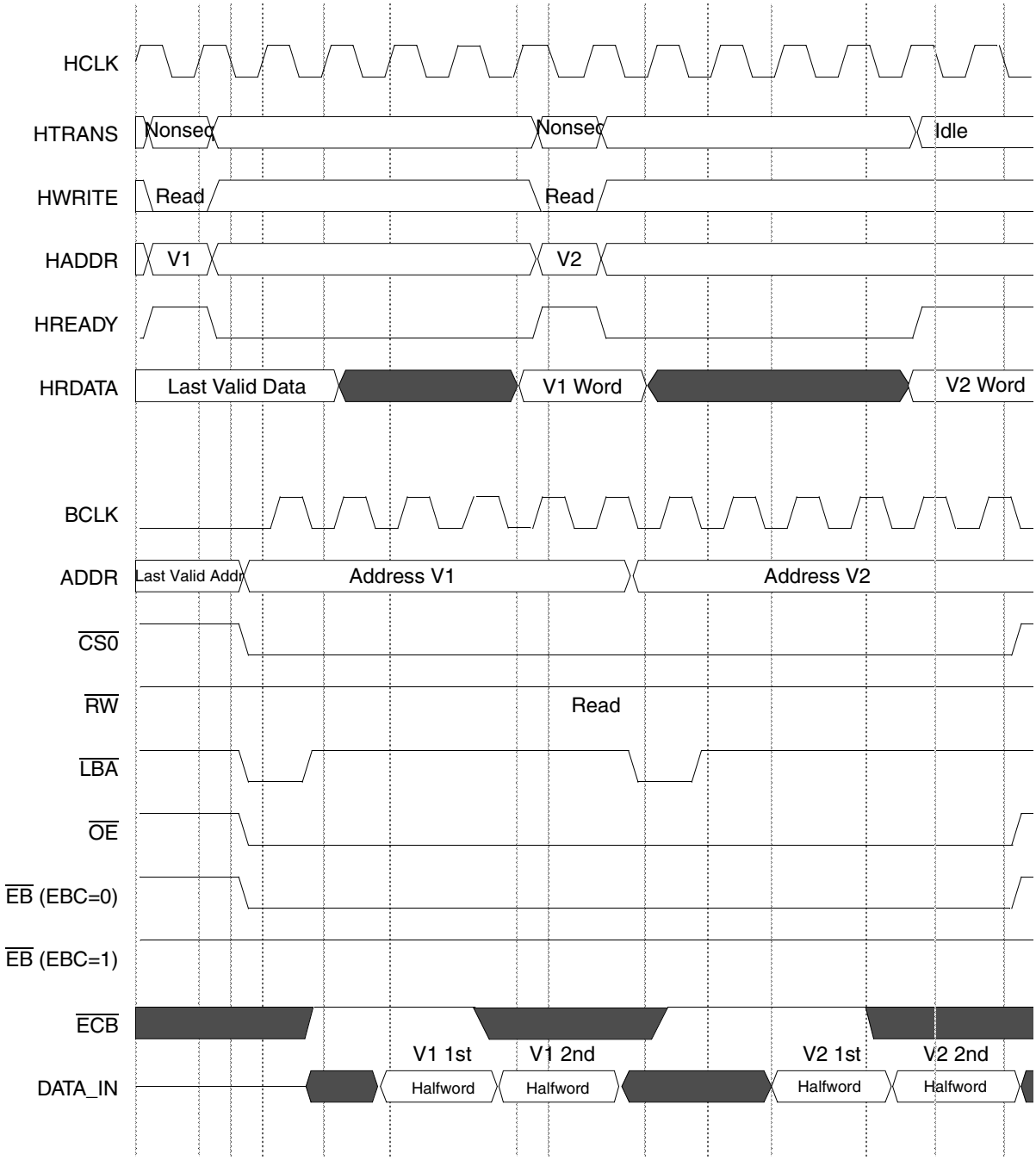


Figure 17-31. Non-sequential Read Accesses, WSC=3, SYNC=1, DOL=1

17.8.4.2 AHB Accesses to Word-Width Burst Memory

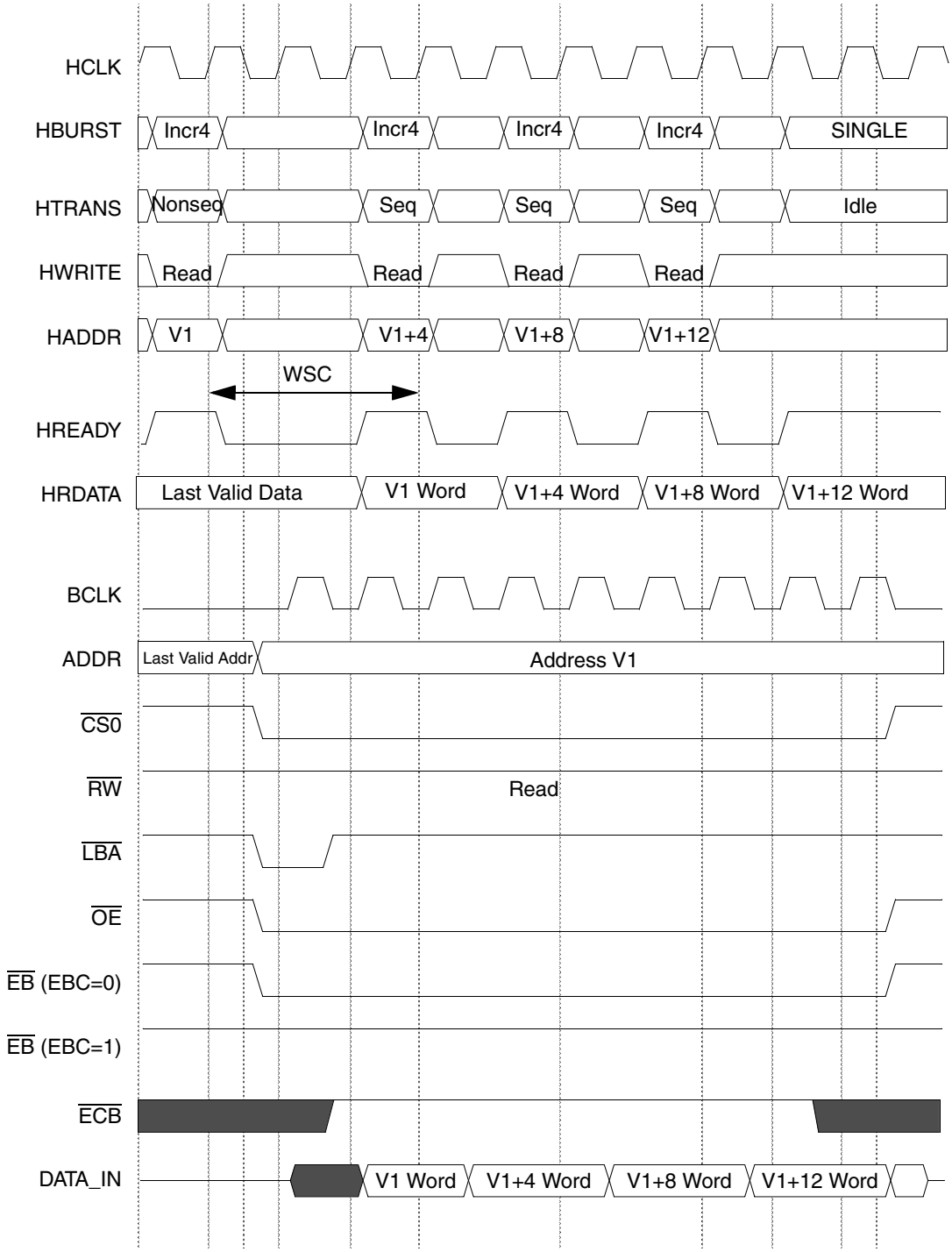


Figure 17-32. Increment 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=0

In Figure 17-32, any address may be a four word boundary address, but not a memory boundary address.

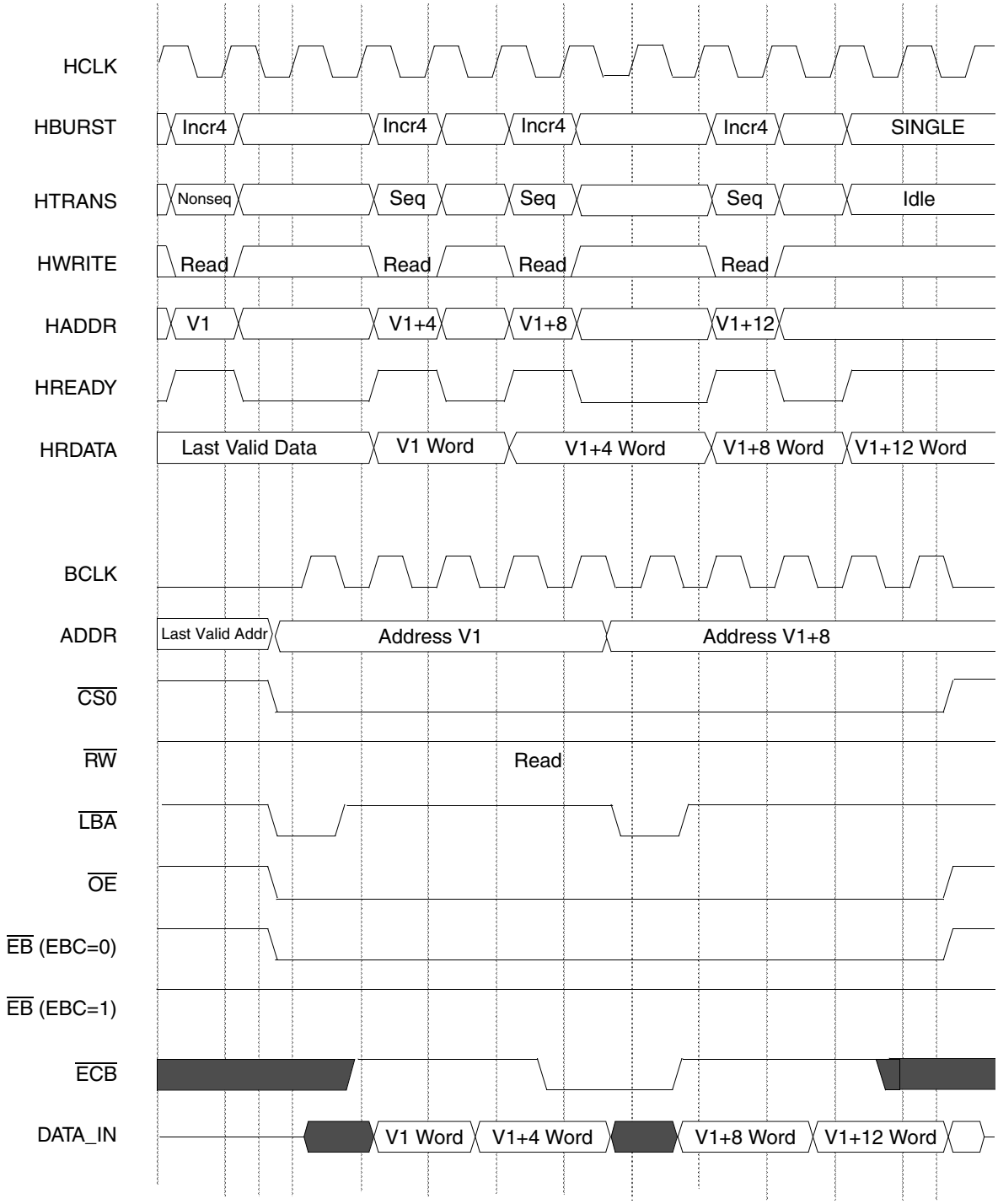


Figure 17-33. Increment 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=0

In Figure 17-33, (V1+8) is a memory boundary address and may be a four word boundary address.

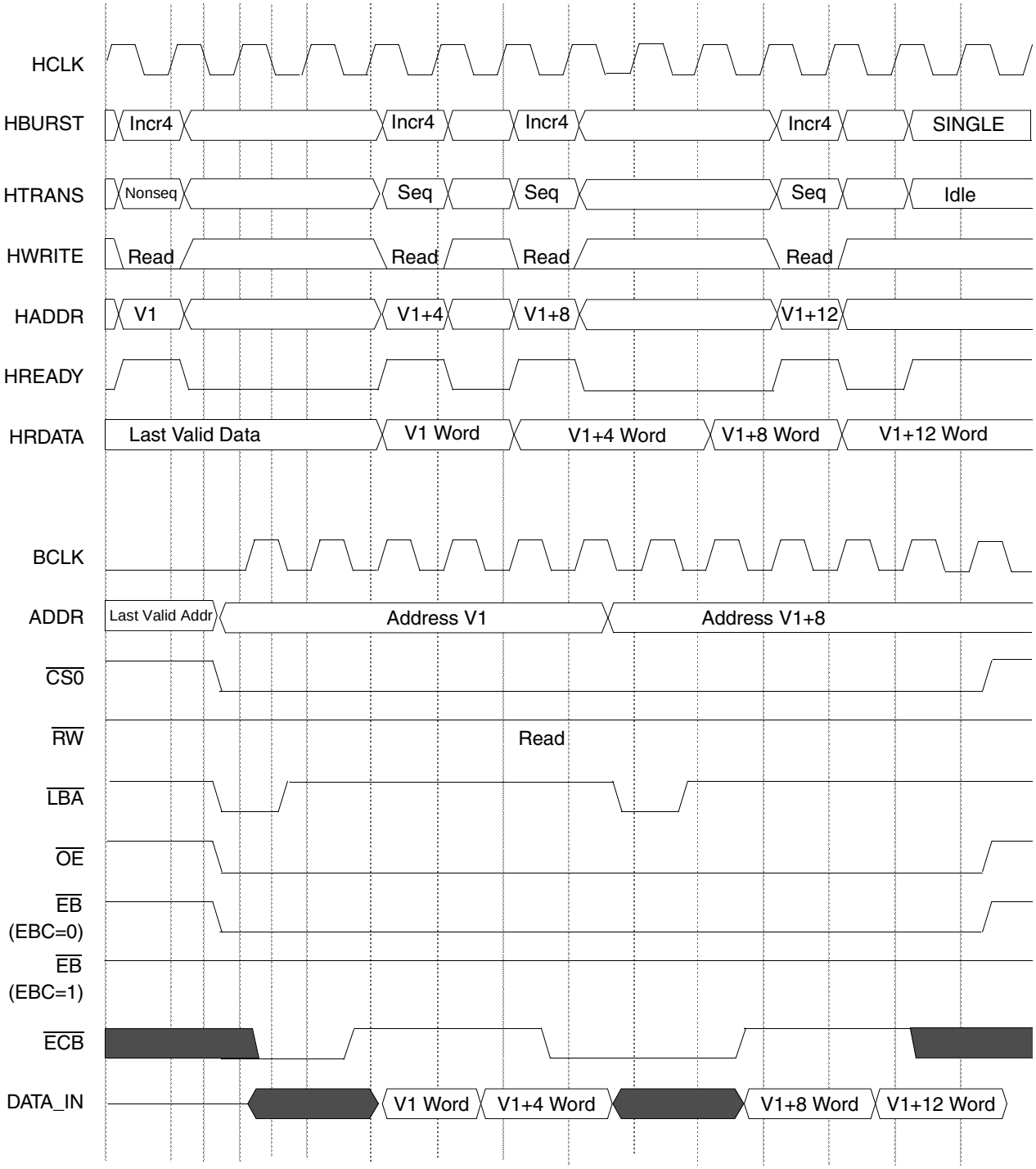


Figure 17-34. Increment 4 AHB Read Access, WSC=3, SYNC=1, DOL=1, WRAP=0, EW=0

In Figure 17-34, (V1+8) is a memory boundary address and may be a four word boundary address.

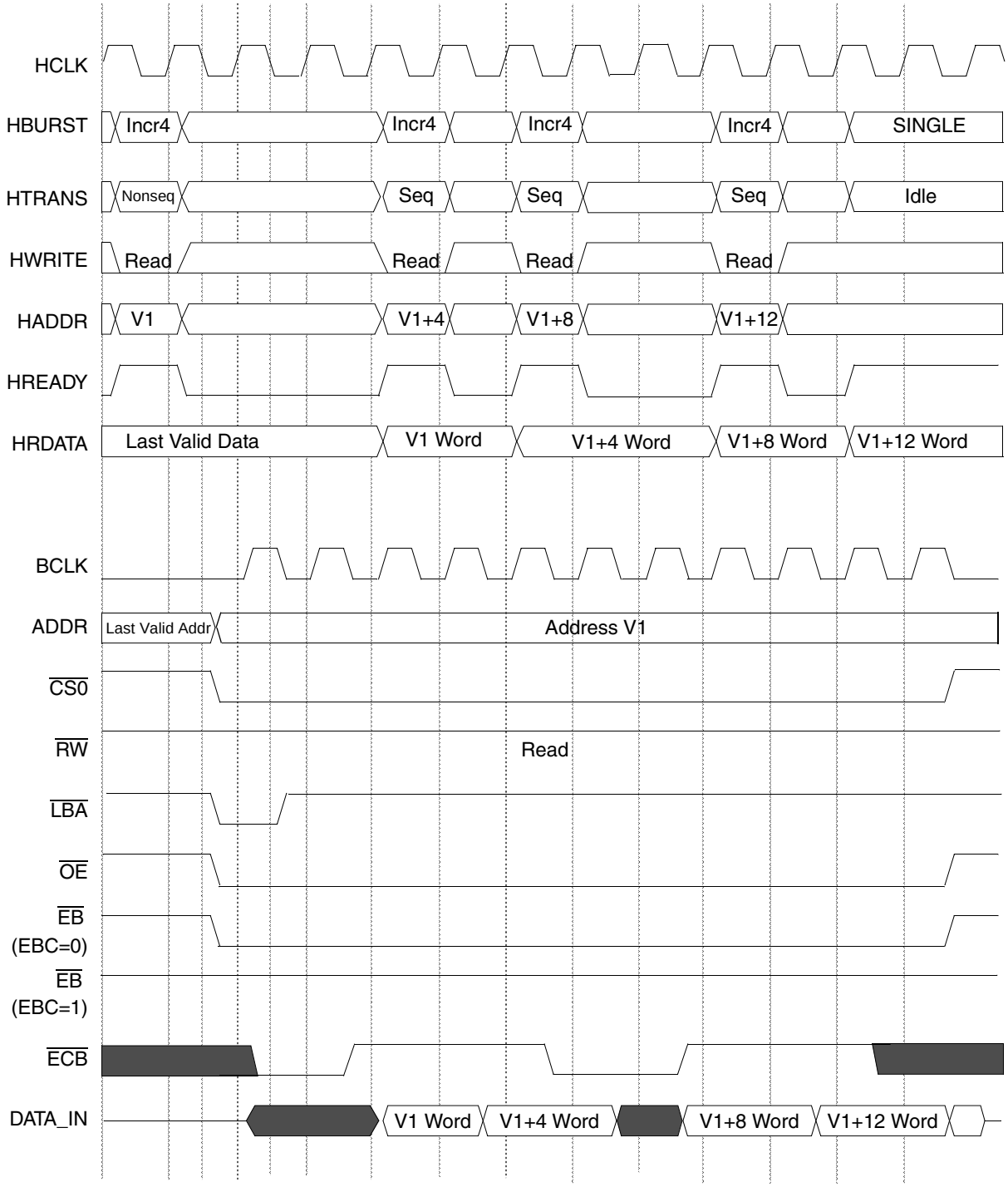


Figure 17-35. Increment 4 AHB Read Access, WSC=3, SYNC=1, DOL=1, WRAP=0, EW=1

In Figure 17-35 and Figure 17-36, (V1+8) is a memory boundary address and may be a four word boundary address.

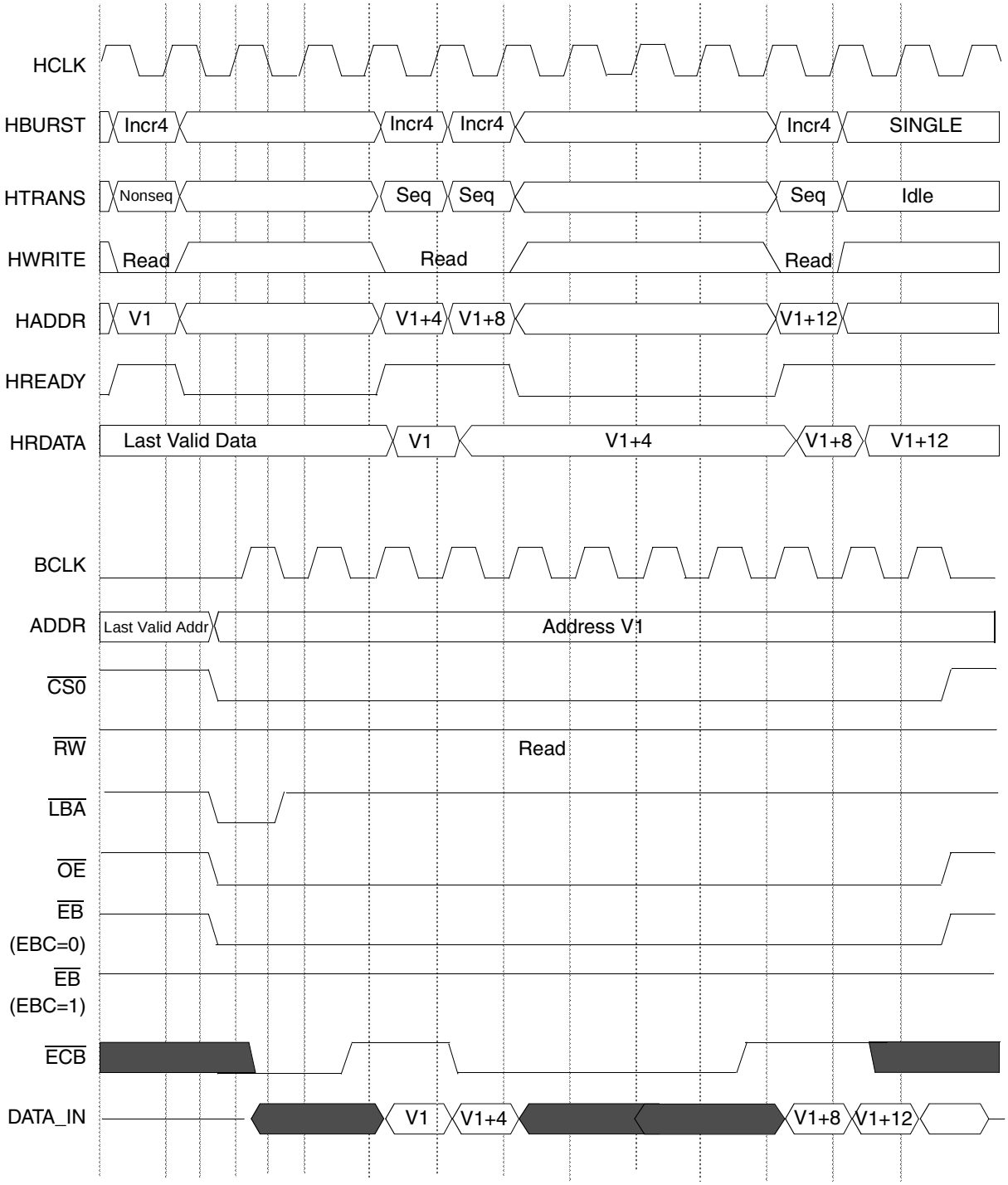


Figure 17-36. Increment 4 AHB Read Access, WSC=3, SYNC=1, WRAP=0, EW=1

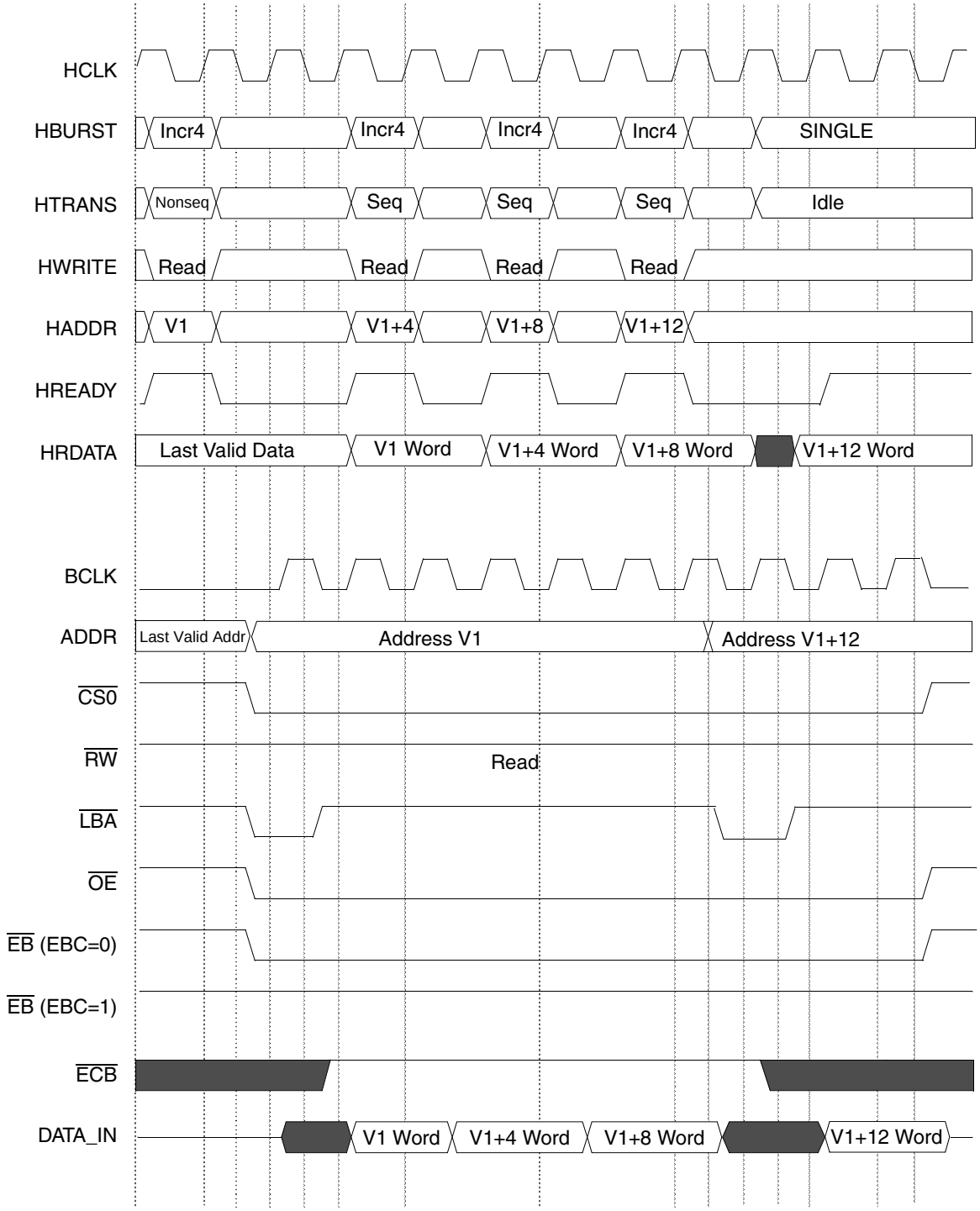


Figure 17-37. Increment 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=1, PSZ=0

In Figure 17-37, (V1+12) is a four-word boundary address.

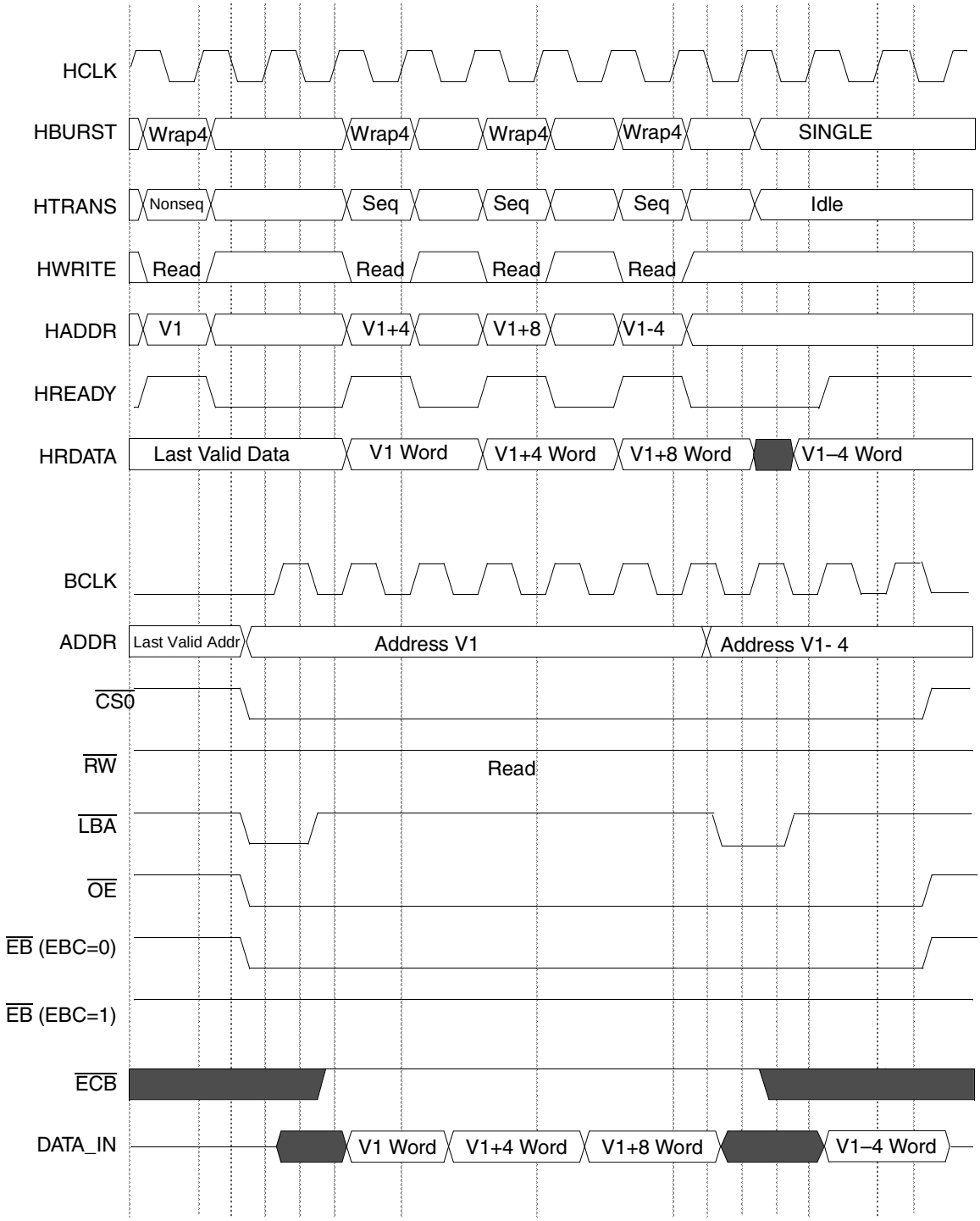


Figure 17-38. Wrap 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=0

In Figure 17-38, (V1-4) is a four-word boundary address.

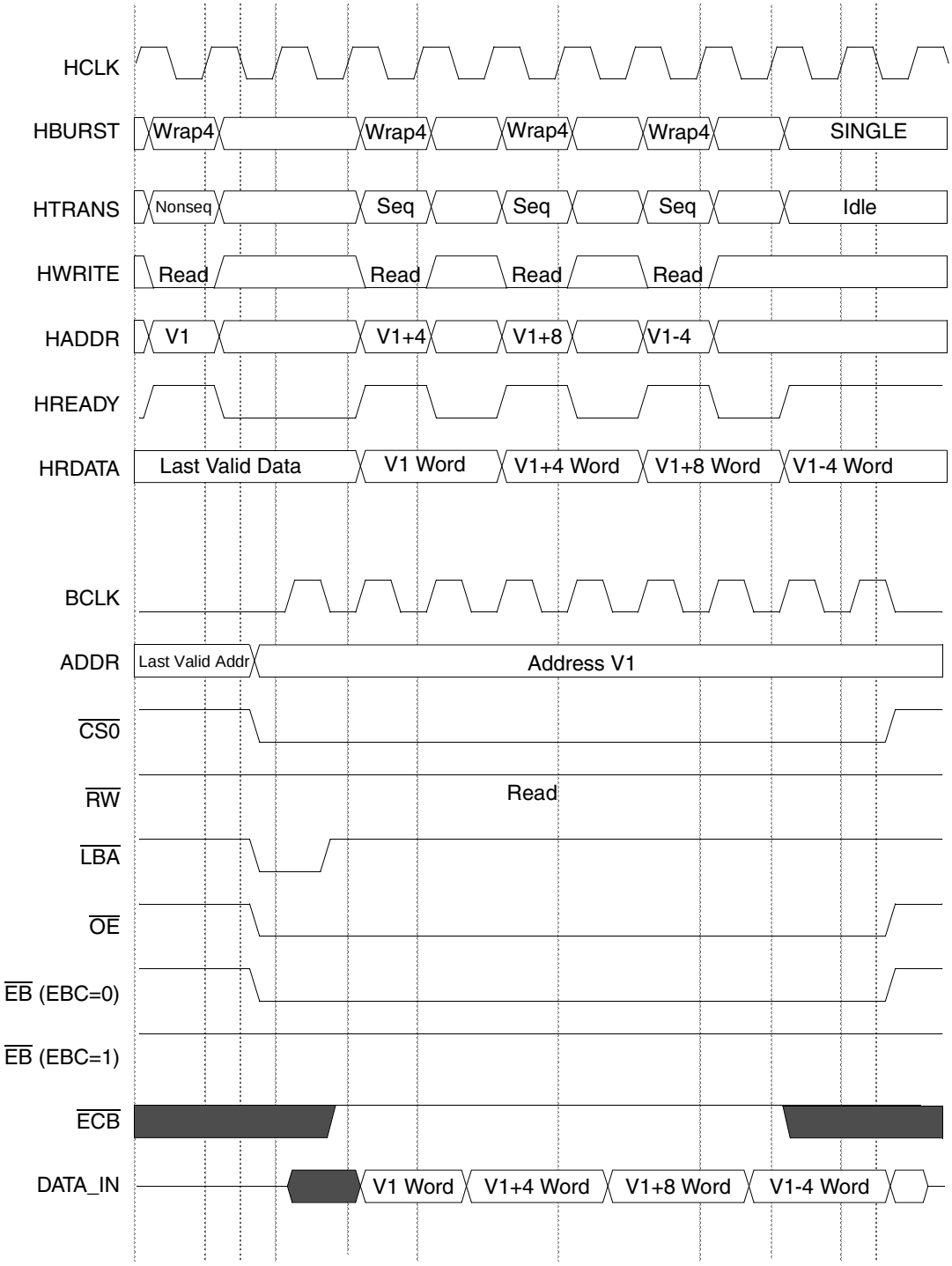


Figure 17-39. Wrap 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=1, PSZ=0

In Figure 17-39, (V1-4) is a four-word boundary address.

17.8.5 Synchronous Accesses Timing Diagrams with PSRAM

17.8.5.1 AHB Sequential Accesses to Halfword Width PSRAM Memory

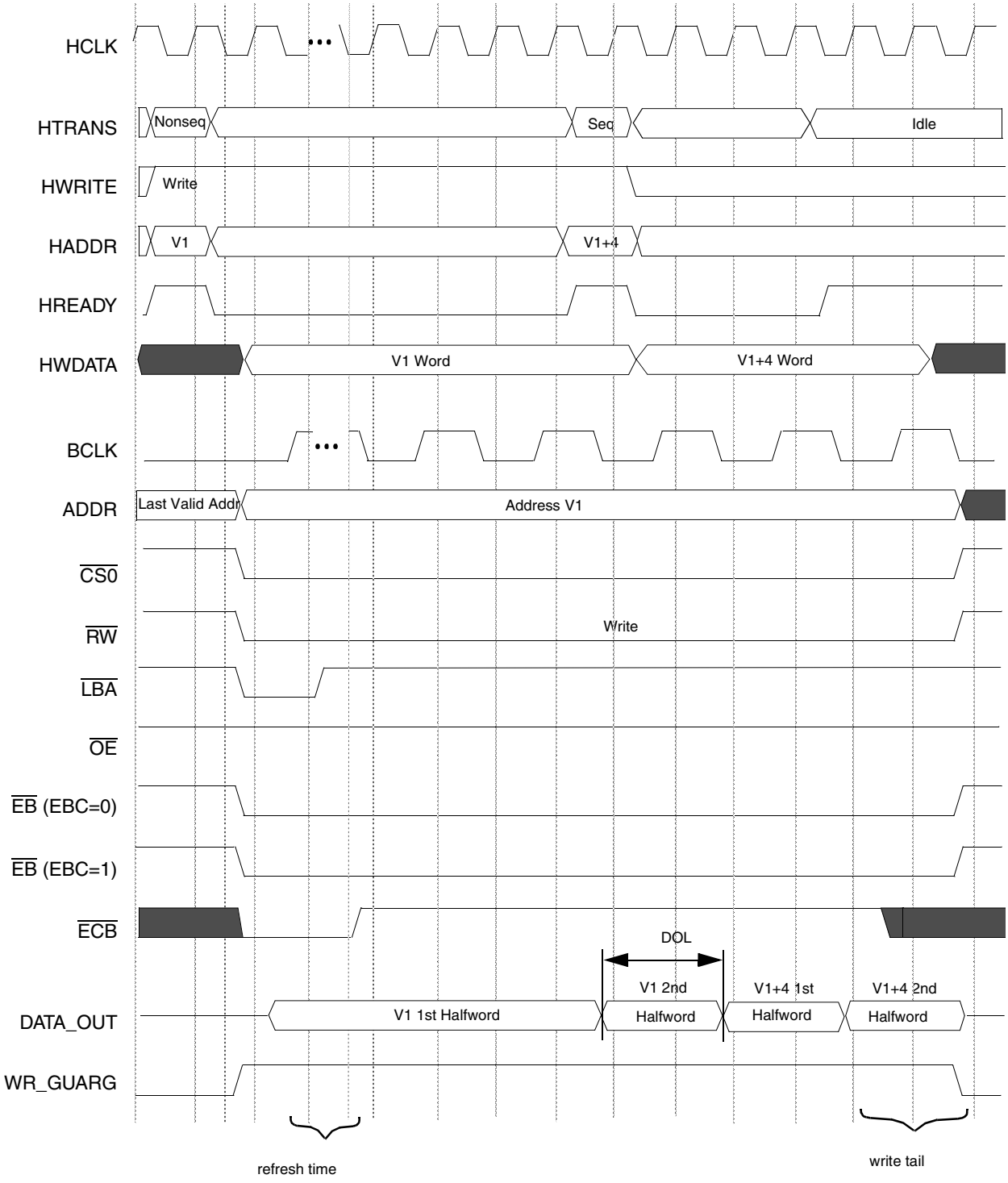


Figure 17-40. Write Access, BCD=1, BCS=1, WSC=5, SYNC=1, DOL=1, EW=1, PSR=1

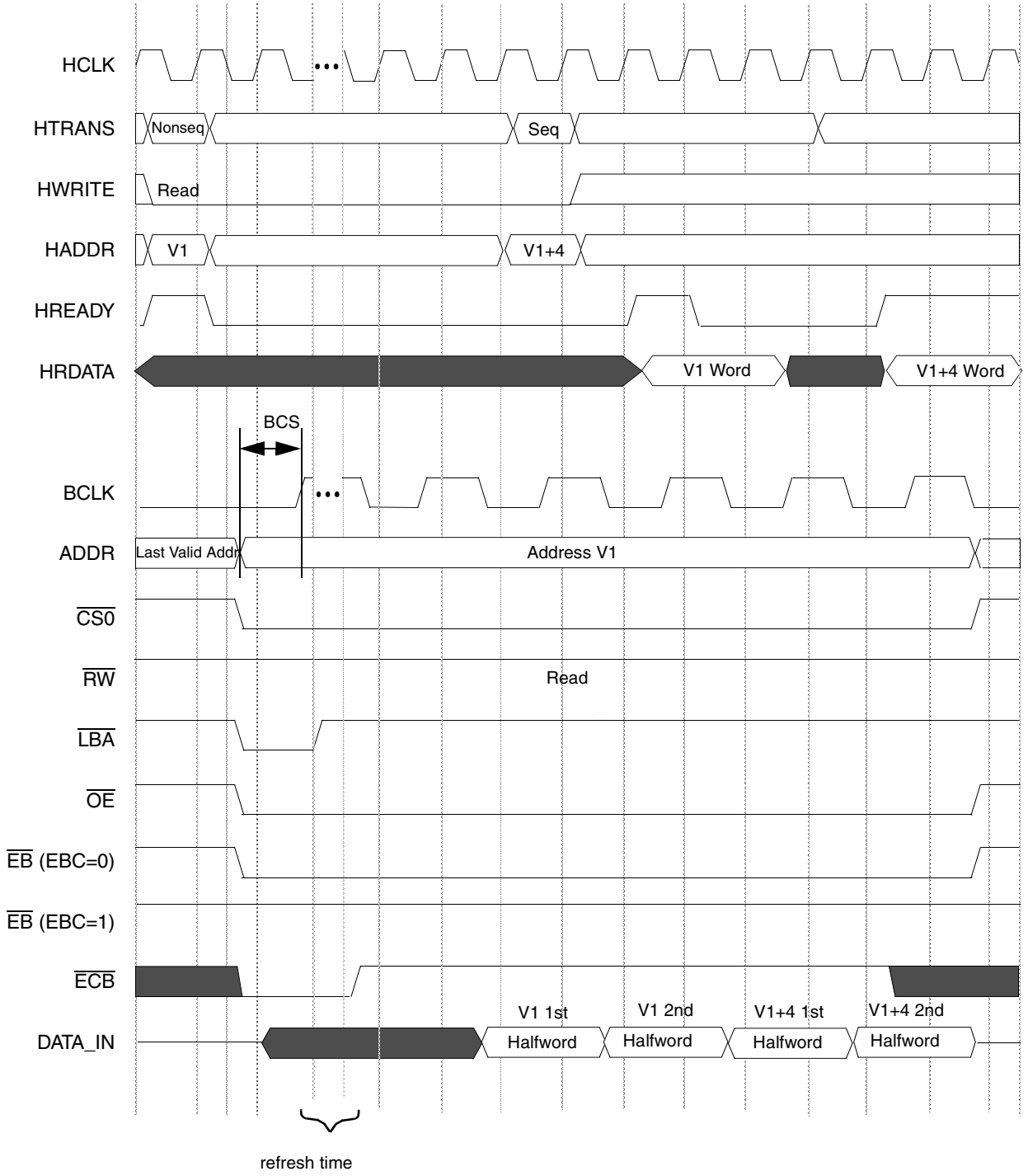


Figure 17-41. Read Access, BCD=1, BCS=1, WSC=5, SYNC=1, DOL=1, EW=1, PSR=1

17.8.5.2 AHB Sequential Accesses to Word-Width PSRAM Memory

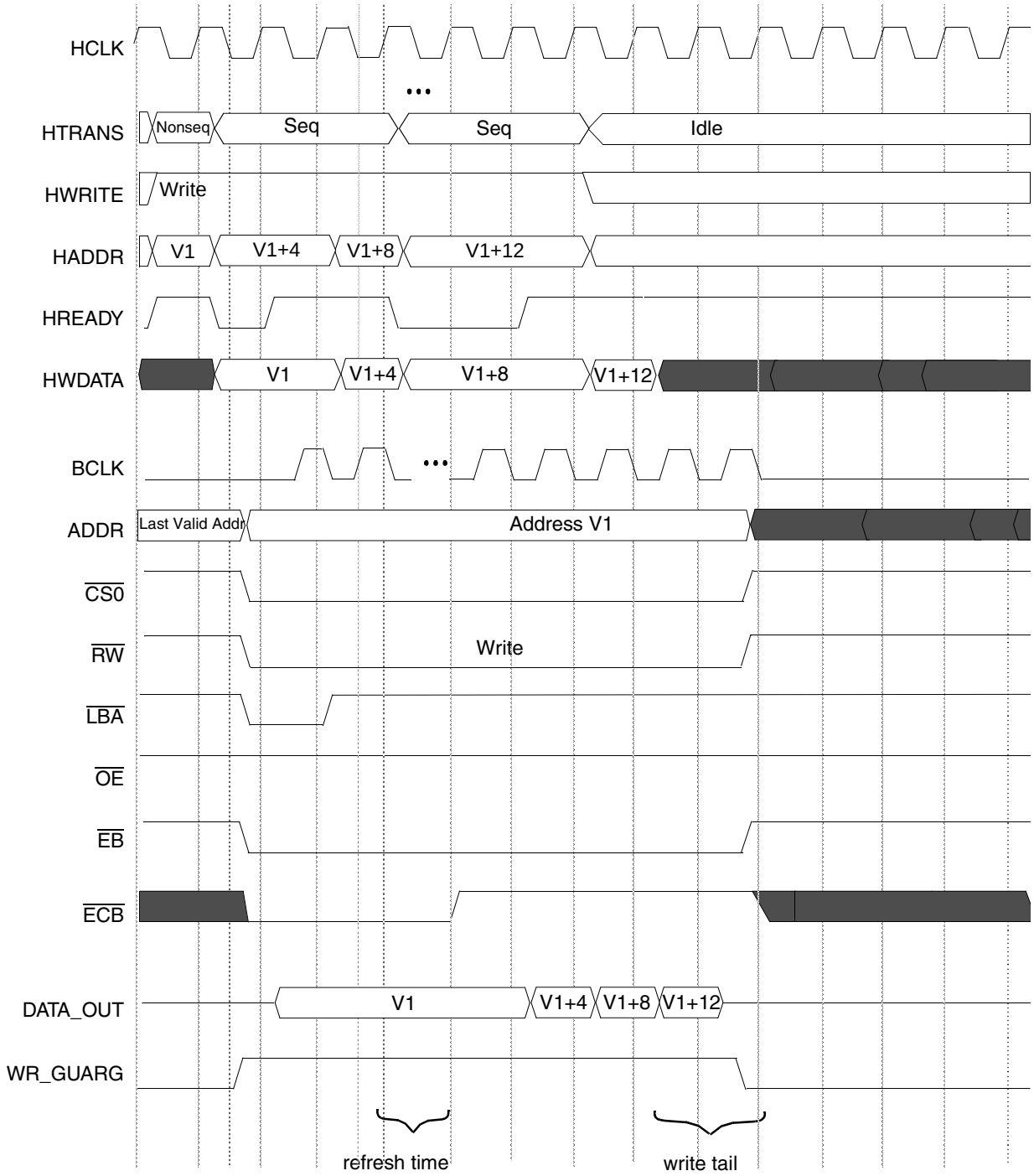


Figure 17-42. Write Access, BCS=1, WSC=4, SYNC=1, PSR=1

17.8.6 Multiplexed A/D Mode

17.8.6.1 Asynchronous Word Accesses to Word-Width Memory

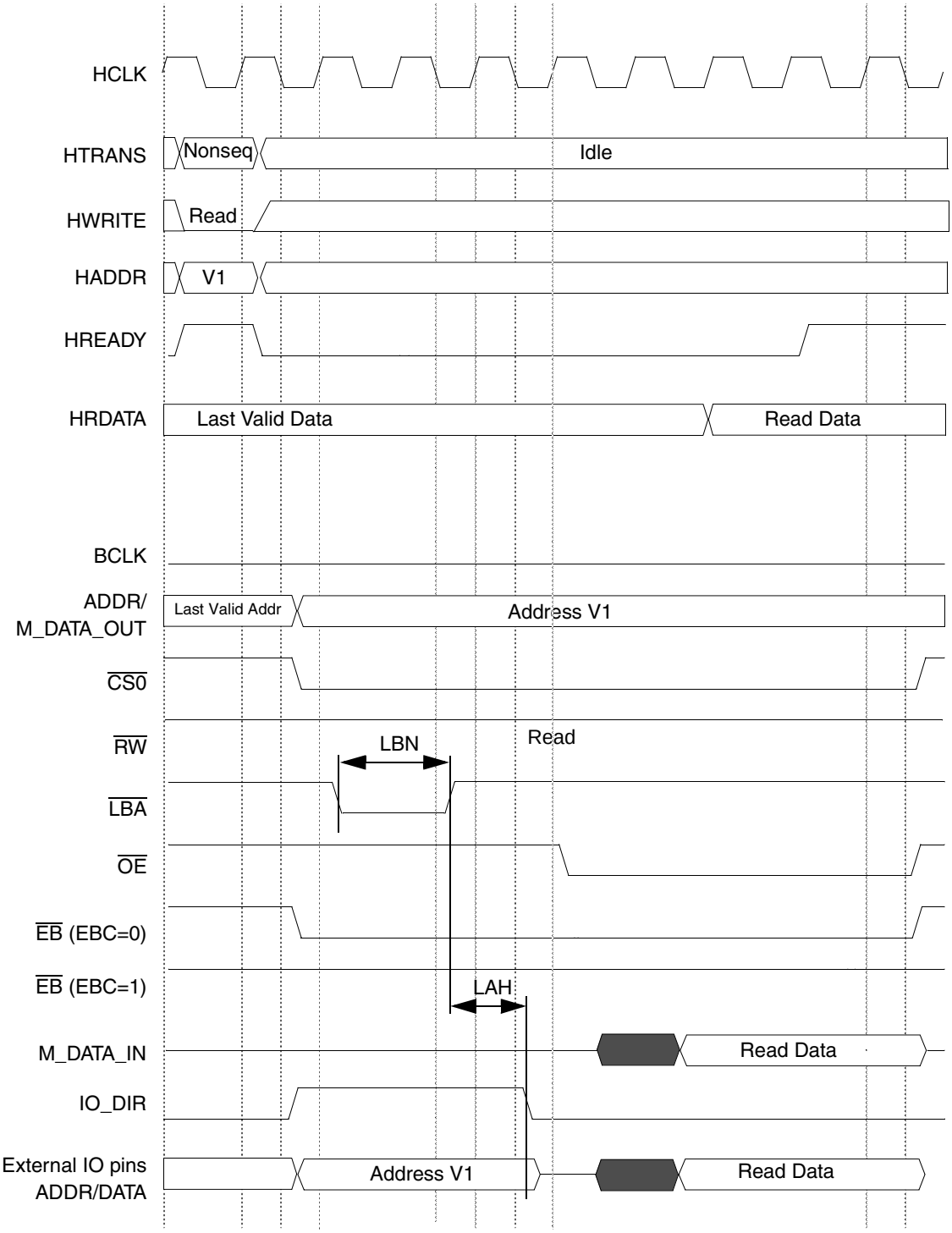


Figure 17-43. Read Access, WSC=7, LBA=1, LBN=1, LAH=1, OEA=7

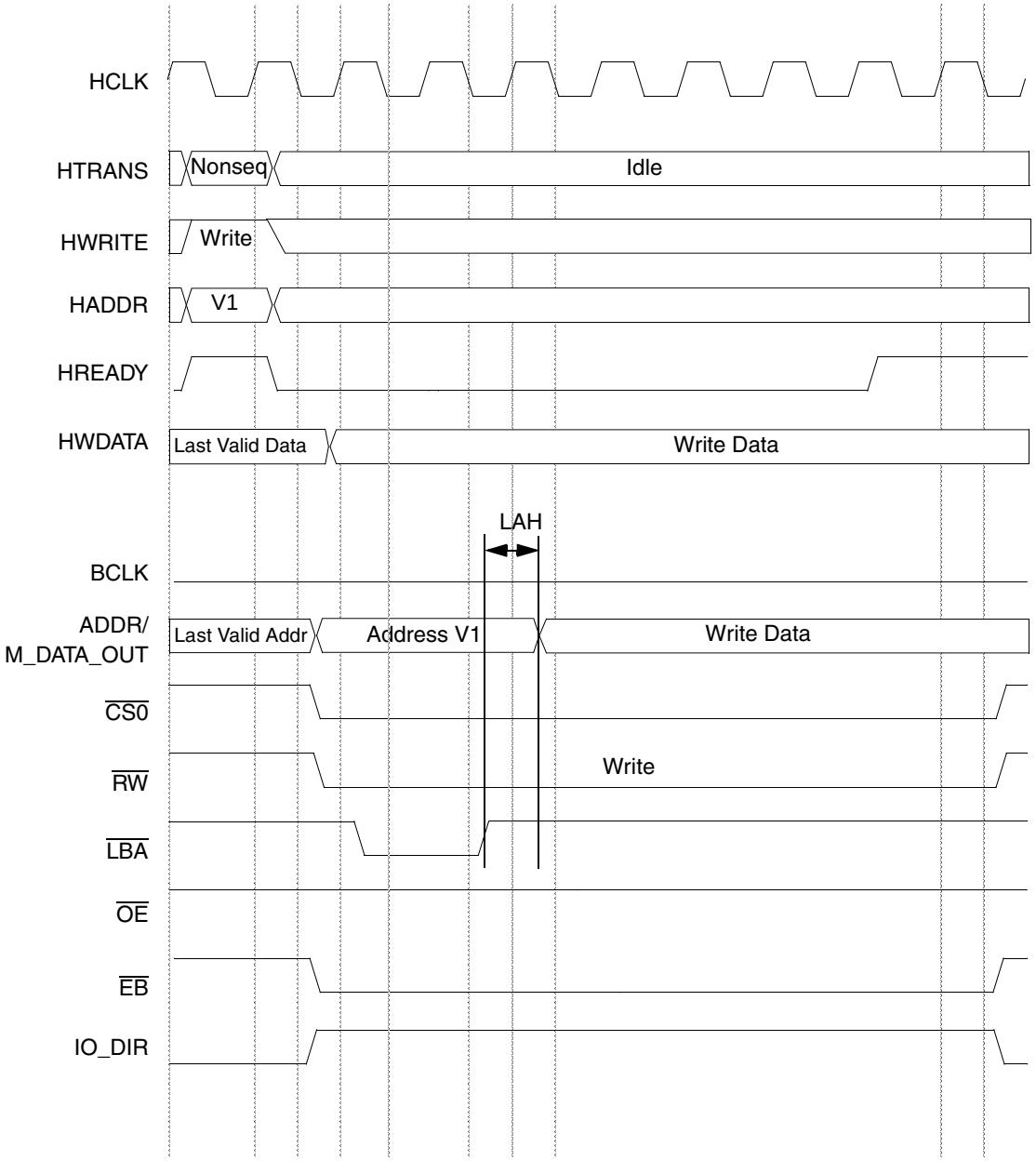


Figure 17-44. Write Access, WSC=7, LBA=1, LBN=1, LAH=1

17.8.6.2 Synchronous Accesses with Word-width Memory

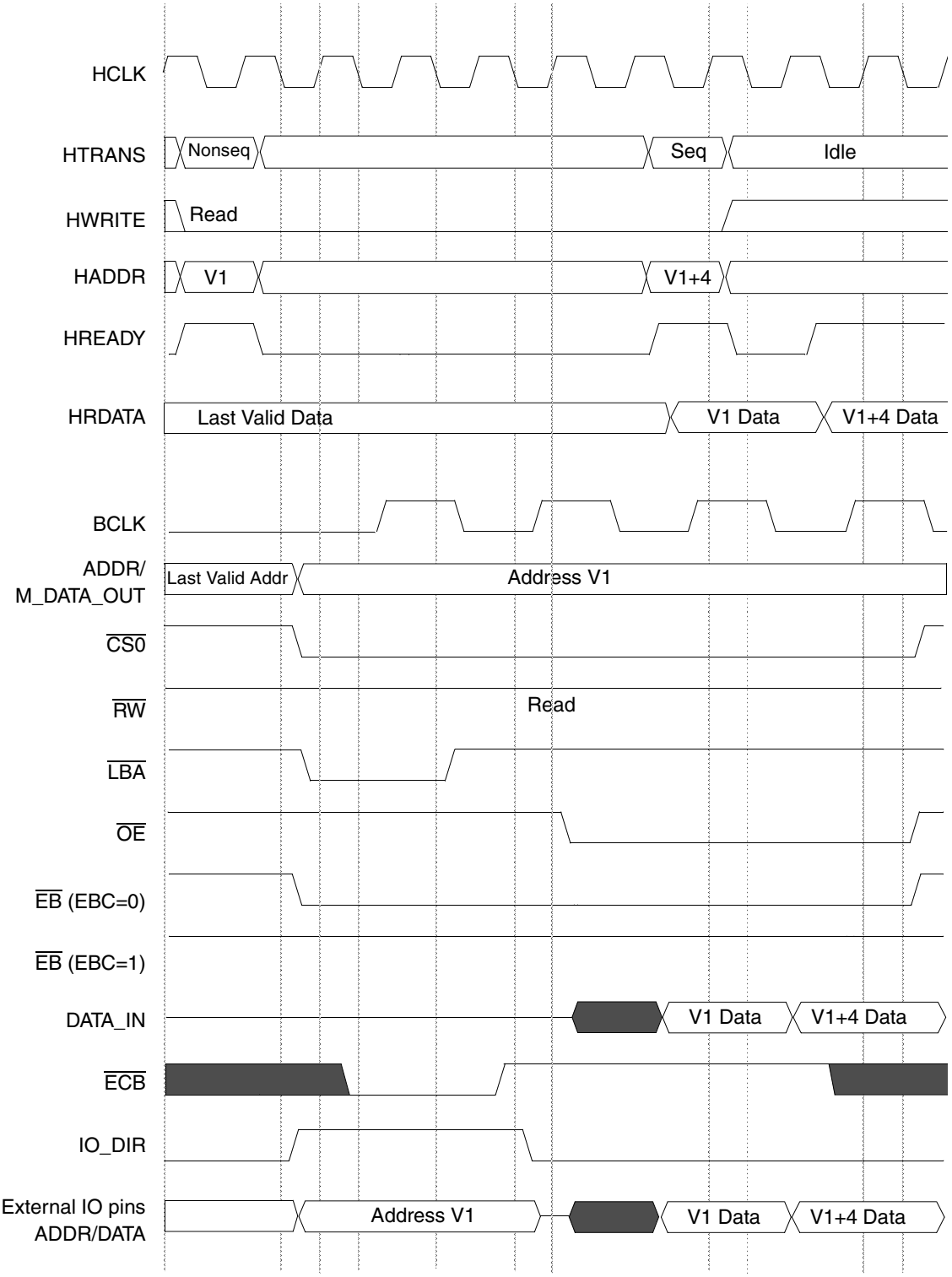


Figure 17-45. Read Access, BCD=1, SYNC=1, WCS=4, DOL=1, LBN=2, LAH=1, PSR=1

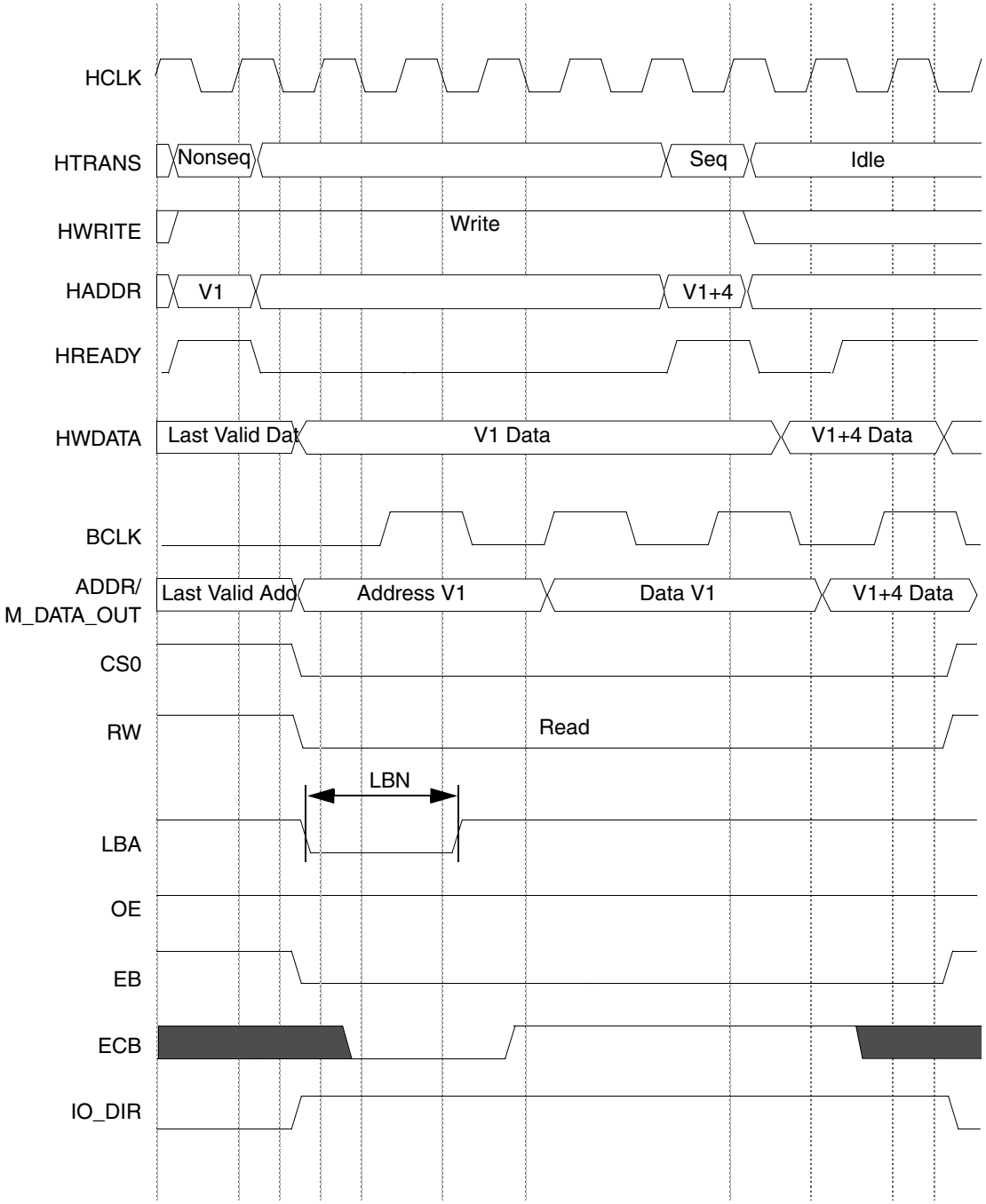


Figure 17-46. Write Access, BCD=1, SYNC=1, WCS=5, DOL=1, LBN=2, LAH=1, PSR=1

Chapter 18
Enhanced SDRAM Controller (ESDRAMC)

The Enhanced Synchronous Dynamic RAM Controller (ESDRAMC) provides interface and control for synchronous DRAM memories for the system. SDRAM memories use a synchronous interface with all signals registered on a clock edge. A command protocol is used for initialization, read, write, and refresh operations to the SDRAM and is generated on the signals by the controller when required due to external or internal requests. It has support for both single data rate RAMs and double data rate SDRAMs. It supports 64-Mbyte, 128-Mbyte, 256-Mbyte, and 512-Mbyte, 1-Gbyte, 2-Gbyte, 4-bank synchronous DRAM by two independent chip selects and with up to 64-Mbyte addressable memory per chip select. Figure 18-1 shows the Enhanced SDRAM Controller top-level diagram that shows the functional organization of the block.

18.1 Overview

Enhanced SDRAM Controller consists of nine major blocks, including the SDRAM command state machine controller, bank register (page and bank address comparators), Row/Column Address Multiplexer, configuration registers, refresh request counter, command sequencer, size logic (splitting access), data path (data aligner/multiplexer), LPDDR interface, and the Power Down timer.

18.1.1 SDRAM Command Controller

This functional block controls the majority of the actions within the Enhanced SDRAM controller, including 12 FF indicating if the bus to the memory is busy for the next 12 cycles, and all the command to the memory are executing through this block.

18.1.2 Bank Model

There are a total of 8 address comparators, one comparator for each of the 4 banks within a chip select region. The comparators are used to determine if a requested access falls within the address range of a currently active SDRAM page. The bank model includes also all timing parameters comparators.

18.1.3 Decoder and Address MUX

All synchronous SDRAMs incorporate a multiplexed address bus, although the address folding points vary according to memory density, number of data I/O, and the processor data bus width. The Enhanced SDRAM Controller takes these variables into account and provides the proper alignment of the multiplexed address through the combination of the row/column address multiplexer, non-multiplexed address pins, and the connections between the controller and the memory devices.

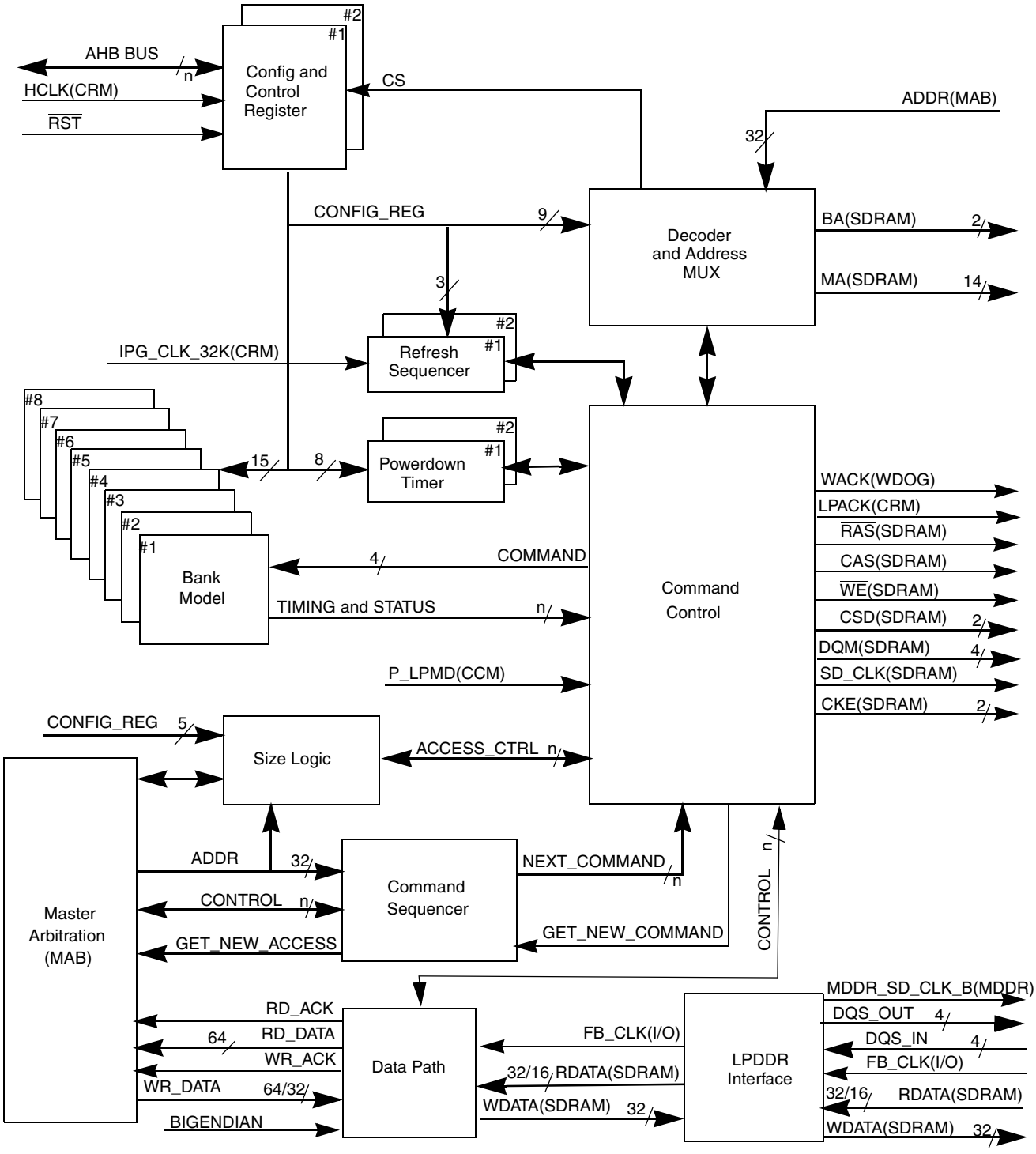


Figure 18-1. Enhanced SDR/LPDDR SDRAM Controller Block Diagram

18.1.4 ESDRAMC Control and Configuration Registers

Control and Configuration registers determine the operating mode of the M3IF. Memory device density and bus width, number of memory devices, CAS latency, row to column delay, and burst length, and others are all configurable. Enable bits are provided for refresh and the Power Down timer. Mode bits provide a mechanism for software initiated SDRAM initialization, setting the device mode register, precharge, and auto-refresh cycles.

18.1.5 Refresh Sequencer

SDRAM memories require periodic refresh to retain data. The refresh request counter generates requests to the SDRAM Command Controller to perform those refresh cycles. Requests are scheduled according to a 32-kHz clock input. One, 2, 4, 8, or 16 refresh cycles are generated per a 32-kHz clock.

18.1.6 Command Sequencer

The Command Sequencer block send the commands be executed (precharge (all/bank), active, read, write, and burst terminate) to the Command Controller, after taking in account the bank's state of the access, the command controller execute signal and the command busy situation.

18.1.7 Size Logic

The Size Logic block gets as inputs from one side the address, access size, and wrap/incr access, and from the other side the configuration values—burst length and DSIZ. In case of a misaligned access, the size logic splits the access into multiple access as a functions of all inputs.

18.1.8 Mobile/Low Power DDR (LPDDR) Interface

The LPDDR interface adds ability to interface with Low Power Double Data Rate SDRAMIt converts the double data rate which is synchronized to both positive and negative edges of the DQS signals to a double width of data bus synchronized to the positive edge of the controller internal clock (which is derived from HCLK).

The interface uses a read FIFO which samples the data with delayed DQS (data strobe) in read cycles. Two read FIFOs are used: one for positive clock edge and the other is for negative clock edge. Delay lines are used to generate a delayed version of DQS input signals in order to sample the data at the middle of the data valid window. In write cycles DQS are output signals that are generated using a delay lines. The data at the input to the interface has a double width from the memory so it is divided into the memory width but with double frequency. For this purpose a MUX is used to pass the upper half and lower half of the data. This way in read cycles we receive double rate data with a DQS signal which is Edge-aligned with read data and create double data rate and centered DQS in write cycles.

DQS delay lines are based on three units:

1. Measurement unit which measures one cycle time from positive edge to the next positive edge. The output of this unit is the number of small delay unit needed to delay a specific positive edge of the clock to overlap the following positive edge.

2. Dividing the result of the measurement by 4.
3. Delay unit take the result and selects the correct delay tap.

The delay unit is duplicated 5 times: 4 units for read (each byte has DQS signal) and one unit for write (to delay the data sampling).

18.1.8.1 Power Down Timer

A Power Down timer detects periods of inactivity to the SDRAM and disables the clock when the inactive period surpasses the selected time-out. Data is retained during the Power Down state. Subsequent requests to the SDRAM incur only a minimal added start-up delay (beyond the normal access time, as specified in Table 18-26). The Power Down timer may be programmed to expire anytime the controller is not actively reading/writing the memory, after 64 or 128 clocks of inactivity, or disabled entirely.

18.1.9 Features

The ESDRAMC includes the following features:

- Optimization of consecutive memory accesses through memory command anticipation (latency hiding)
  - Hiding latency by optimization the commands to both CS connected-command anticipation
  - Keeping track of open memory pages
  - Bank-wise memory address mapping
  - SDRAM burst length configuration of 4 or 8 (for 16-bit memory burst length 4 is not supported) or full page mode
  - LPDDR burst length configuration of 8
  - Support of different internal burst length (1/4/8 words) by using burst truncate commands
  - ARM AMBA AHB light compliant
  - Shared Address and command bus to SDRAM/LPDDR
- Supports 64-Mbyte, 128-Mbyte, 256-Mbyte, 512-Mbyte, 1-Gbyte, 2-Gbyte, 4-bank, single data rate, synchronous SDRAM, and LPDDR
  - Two independent chip selects
  - Up to 256-Mbyte per chip select
  - Up to four banks active simultaneously per chip select
  - JEDEC standard pinout/operation
- Support for 16-bit and 32-bit Mobile/Low Power DDR266 devices
- PC133-compliant Interface
  - 133 MHz system clock achievable with “-7” option PC133 compliant memories
  - Single fixed-length (4/8-word) burst or full page access
  - Access time of 9-1-1-1-1-1-1-1 at 133 MHz (for read access when memory bus is available, row is open and CAS latency configured to 3 cycles). The access time includes the M3IF delay (assuming no arbitration penalty).

- Software configurable for different system and memory devices requirements
  - 16- or 32-bit memory data bus width
  - number of row and column addresses
  - row cycle delay (t_{RC})
  - row precharge delay (t_{RP})
  - row to column delay (t_{RCD})
  - column to data delay (CAS Latency)
  - load mode register to active command (t_{MRD})
  - write to precharge (t_{WR})
  - write to read (t_{WTR}) for LPDDR memories only
  - LPDDR exit power down to next valid command delay (t_{XS})
  - active to precharge (t_{RAS})
  - active to active (t_{RRD})
- Built in auto-refresh timer and state machine
- Hardware and software supported self-refresh entry and exit
  - Keeps data valid during system reset and low power modes
  - Auto Power Down timer (one per Chip Select)
  - Auto Precharge timer (one per bank in each chip select)

18.1.10 Modes of Operation

Each of the Enhanced SDRAM Controller (ESDRAMC) operating modes are briefly described in this section. The ESDRAMC's different operating modes for each chip select ($\overline{CSD}0$ and $\overline{CSD}1$) are defined by the SMODE field (3 bits) in the ESDCTL0 and ESDCTL1 registers respectively. In addition to the normal operating mode, the controller is capable of operating in the alternate operating modes primarily used for SDRAM/LPDDR initialization.

Any access to the SDRAM/LPDDR memory space, while in one of the alternate modes, results in the corresponding special cycle being run. Moving from Normal to any other mode does not close (precharge) any banks that may be open (activated). Under most circumstances, software should run a precharge-all cycle when transitional out of Normal Read/Write mode. Reset initializes the operating mode to "Normal Read/Write". This is a high level description only, detailed descriptions of operating modes are contained in Section 18.4, "Functional Description."

- Normal Read/Write Mode: This is the normal operating mode used to READ/WRITE (single or burst accesses) from/to external SDRAM/LPDDR devices. ESDRAMC automatically drives the PRECHARGE/ACTIVE/BURST TERMINATE commands during the normal operating mode.
- Precharge Mode: The manual PRECHARGE command is used to manually deactivate the open row (ACTIVE) in a particular bank or the open row in all banks. The bank(s) will be available for a subsequent row access a specified time (tRP) after the PRECHARGE command is issued. External memory device input A10 determines whether one or all banks are to be precharged, and in the case that only one bank is to be precharged, inputs BA0 and BA1 select the bank. The manual

PRECHARGE command is used during SDRAM initialization, LOAD MODE REGISTER command and manual REFRESH cycles

- Auto Refresh Mode: The AUTO REFRESH command is used to retain data in the SDRAM memory devices. This command is non persistent, so it must be issued each time a refresh is required. The ESDRAMC has a REFRESH counter for each CSD_B (external memory region) and it automatically handles the REFRESH commands toward the memory. The manual AUTO REFRESH command is used during SDRAM initialization or in case that the REFRSH counters are not enabled.
- Load Mode Register Mode: The Mode Register is used to define the specific mode of operation of the SDRAM device. This definition includes the selection of a burst length, a burst type, a CAS latency, an operating mode and a write burst mode. The Mode Register is programmed via the LOAD MODE REGISTER command and will retain the stored information until it is programmed again or the external memory device loses power. The Mode Register must be loaded when all banks are idle (after PRECHARGE ALL). The Enhanced SDRAM Controller will wait a specified ($t_{MRD)}$ period of time as configured in ESDCFG0 or ESDCFG1 registers. Violating either of these requirements by an incorrect controller register configuration results in unspecified operation.

18.2 External Signal Description

This section discusses input and output signals between the Enhanced SDRAM Controller and the external memory devices. Other than the chip select outputs ($\overline{CSD}0$ and $\overline{CSD}1$) and clock enables (CKE0 and CKE1), all signals are shared between the two chip select regions. Table 18-1 summarizes the interface signals, and is followed by a detailed description of signal functions. Interconnect and timing diagrams are included as part of the detailed discussion on controller operation in Section 18.4, “Functional Description.”

Table 18-1. ESDRAMC Signal Properties

Name	Port	Function	Reset State	Direction
SD_CLK	—	Clock to SDRAM (up to 133MHz)	1	Output
FB_CLK_IN	—	Feedback CLock	0	Input
CKE0	—	Clock enable to SDRAM 0	0	Output
CKE1	—	Clock enable to SDRAM 1	0	Output
$\overline{CSD}$[0]	—	Chip select to SDRAM Array 0	1	Output
$\overline{CSD}$[1]	—	Chip select to SDRAM Array 1	1	Output
RDATA[31:0]	—	Read Data from memories	0	Input
WDATA[31:0]	—	Write data to memories	0	Output
MA[13:0]	—	Multiplexed Address	0	Output
BA[1:0]	—	Bank address	0	Output
DQM3	—	Data Qualifier Mask byte 3 (D[31:24])	0	Output
DQM2	—	Data Qualifier Mask byte 2 (D[23:16])	0	Output
DQM1	—	Data Qualifier Mask byte 1(D[15:8])	0	Output

Table 18-1. ESDRAMC Signal Properties (continued)

Name	Port	Function	Reset State	Direction
DQM0	—	Data Qualifier Mask byte 0 (D[7:0])	0	Output
$\overline{WE}$	—	Write Enable	1	Output
$\overline{RAS}$	—	Row Address Strobe	1	Output
$\overline{CAS}$	—	Column Address Strobe	1	Output
LPACK	—	Low Power Mode Acknowledge	1	Output
WACK	—	Memory wake up	0	Output
BIGEND	—	big endian signal	system dependent	Input
P_LPMD	—	Low Power Mode Entry Request	0	Input
Mobile LPDDR Signals				
MDDR_SD_CLK_B	—	Inverted Clock to LPDDR SDRAM	0	Output
DQS_OUT3	—	Data strobe byte 3 (D[31:24])	0	Output
DQS_OUT2	—	Data strobe byte 2 (D[23:16])	0	Output
DQS_OUT1	—	Data strobe byte 1 (D[15:8])	0	Output
DQS_OUT0	—	Data strobe byte 0 (D[7:0])	0	Output
DQS_IN3	—	Data strobe byte 3 (D[31:24])	0	Input
DQS_IN2	—	Data strobe byte 2 (D[23:16])	0	Input
DQS_IN1	—	Data strobe byte 1 (D[15:8])	0	Input
DQS_IN0	—	Data strobe byte 0 (D[7:0])	0	Input

18.2.1 Detailed Signal Descriptions

Table 18-2 lists the signals and descriptions of all of the I/O signals that interface with the ESDRAMC.

Table 18-2. ESDRAMC Detailed Signal Description

Signal	I/O	Description
SD_CLK	O	SDRAM CLock. SD_CLK output provides the timing reference for the memory devices. All other SDRAM interface signals are referenced to this clock. SD_CLK is synchronous to the system clock, but is gated off during low power operating modes when both CKE0 and CKE1 are negated.
FB_CLK_IN	I	Feedback Clock. The FB_CLK_IN signal is used by the Enhanced SDRAM controller to sample the data during READ cycles. The FB_CLK_IN is a delayed SD_CLK, and at a good proximity is identical to the external memory device clock. A delay between SD_CLK and FB_CLK_IN compensates the PAD input/output buffers, chip route.
CKEO CKE1	O	SDRAM/MMDR Clock Enables. Clock enable outputs to the SDRAM memory devices. CKE0 corresponds to SDRAM/LPDDR array 0 and CKE1 to SDRAM/LPDDR array 1. Activates the memory's clock input when high, indicating a stable clock is being supplied. Deactivates the memory's clock input when low. CKEx low initiates Power Down and Self Refresh modes to the SDRAM.

Table 18-2. ESDRAMC Detailed Signal Description (continued)

Signal	I/O	Description
$\overline{\text{CSD0}}$ $\overline{\text{CSD1}}$	O	SDRAM/LPDDR Chip Select. $\overline{\text{CSD0}}$ and $\overline{\text{CSD1}}$ are used to select SDRAM/LPDDR Array 0 and SDRAM/LPDDR Array 1, respectively. The chip select signals are used to indicate when a valid command is present on the other control signals and to which device the command is directed.
RDATA[31:0] WDATA[31:0]	I/O	Read/Write Data Bus. The 32 data pins are used to transfer data between the Enhanced SDRAM Controller and memory. Data bit 31 is the most significant bit. Bit 0 is the least significant. 16-bit memory alignment is selectable according to the programming of the DSIZ field in the ESDCTL register (see Section 18.3.3, “Register Descriptions”).
MA[13:0]	O	Multiplexed Address Bus specifies the SDRAM page and location within the page targeted by the current access. Connections between Enhanced SDRAM Controller and memory will vary depending on SDRAM device density, Consult Section 18.4.2, “Address Multiplexing” and specifically Table 18-23 and Table 18-24 for details on supported SDRAM configurations.
BA[1:0]	O	Bank Address Bus. The bank address pins specify to which bank the current command is targeted. Table 18-23 and Table 18-24 document which address pins are used as the bank address bus for given device configuration.
DQM3, DQM2, DQM1, DQM0	O	Data Qualifier Mask. During read cycles, DQMx controls the SDRAM data output buffers. DQMx asserted high disables the output buffers leaving them in a high-impedance state. DQMx low allows the data buffers to drive normally. During write cycles, DQMx controls which bytes are written in the SDRAM. DQMx driven low enables a write to the corresponding byte, while DQMx asserted high leaves the byte unchanged. DQM0 corresponds to D[7:0] and DQM3 to D[31:24]. Sixteen bit memories require only two DQM connections. Memories aligned to the upper data bus (D[31:16]) connect to DQM2 and DQM3, while memories aligned to the lower data bus (D[15:0]) connect to DQM0 and DQM1. The ESDRAMC takes care of the endian operating mode, and drives the respective DQM strobe required. **Note:** When the controller is used to interface Mobile/Low Power DDR, DQM will change twice each cycle (like the data) and will be aligned to DQS edges.
$\overline{\text{WE}}$	O	Write Enable is part of the three bit command field ($\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ make up the other two bits) used by the SDRAM. Generally, $\overline{\text{SDWE}}$ is asserted low if a command transfers data to the memory. A detailed summary of the supported SDRAM commands is provided in Table 18-27.
$\overline{\text{RAS}}$	O	Row Address Strobe is also part of SDRAM command field. It is generally used to indicate an operation affecting an entire bank or row. RAS is an active low signal. Table 18-27 provides details on SDRAM command encoding.
$\overline{\text{CAS}}$	O	Column Address Strobe is the third signal comprised in the command field. It generally signifies a column oriented command. CAS is an active low signal. Table 18-27 provides details on SDRAM command encoding.
LPACK	O	Low Power Mode Acknowledge. This signal indicates that the external memory devices operates in one of the low power mode. This signal is used by the system clock and reset module to enable the deactivation of the system clock during system low power operating mode. The default value of LPACK is high, not to affect in case that the memory devices are not used/disabled.
L_LPMD	I	Low Power Mode Entry Indication. This input signal (from the CRM) is used as a low power mode entry indication. During RUN mode P_LPMD value is 0. While there is a system request to enter low power mode (STOP) the P_LPMD value changes to 1. During Sleep Mode the ESDCTL clock is shut off, so the external memory devices need to enter self refresh. This change will cause the ESDCTL to complete all active/pending requests afterwards, the external memory devices will be placed in self refresh mode.

Table 18-2. ESDRAMC Detailed Signal Description (continued)

Signal	I/O	Description
WACK	O	External memory device WakeUp. This signal indicates that the external memory device power up period is over, so the initialization commands can be issued. This signals goes down during reset or low power operating mode. This signal can be used by the system WatchDog module to disable/mask WatchDog reset/interrupt during the time WACK is low.
SD_CLK/$\overline{\text{MDDR\_SD\_CLK}}$	O	LPDDR SDRAM Inverted Clock. SD_CLK and $\overline{\text{MDDR\_SD\_CLK}}$ are Mobile/Low Power DDR Differential Clocks. All LPDDRs address and control input signals are sampled on the crossing of the positive edge of SD_CLK and negative edge of $\overline{\text{MDDR\_SD\_CLK}}$. Internal clock signals are derived from SD_CLK/$\overline{\text{MDDR\_SD\_CLK}}$.
DQS_OUT3, DQS_OUT2, DQS_OUT1, DQS_OUT0	O	Data Strobes Outputs. Data strobes are used for data capture for LPDDR memory. During write cycles, they are generated by the SDRAMC controller and are centered with write data. DQS3 corresponds to the most significant byte and DQS0 to the least significant byte.
DQS_IN3, DQS_IN2, DQS_IN1, DQS_IN0	I	Data Strobes Inputs. During read cycles, DQS_INx are generated by the memory devices and are edge aligned with read data. For read data, the controller receives a DQS signal that is edge aligned with the read data. The DQS delay module is used to delay the DQS signal and center it in the data valid window.
BIGEND	I	Big Endian. Defines the byte ordering. For example, it controls how multi-byte objects are represented by the underlying architecture. This signal is driven by the ARM platform, and defines two endianess modes, little and big as illustrated in Figure 18-3.

18.3 Memory Map and Register Definition

The Enhanced SDRAM Controller programming model consists of control and configuration registers (32-bit length) for each chip select and a miscellaneous register, as shown in Table 18-3. The control register maintain system dependent information, while the configuration register maintain SDRAM/LPDDR memory device dependent information. ESDCFG0 defines the operating characteristics for the region selected by $\overline{\text{CSD0}}$, while ESDCFG1 does the same for the $\overline{\text{CSD1}}$ region. Bit field assignments within the registers are identical, so a single description will apply to both registers.

Both the control and configuration registers are 32 bits in length with bit fields defined in Figure 18-4 to Figure 18-7, respectively. All implemented bits are fully readable and writable. Reserved bit locations are unaffected by writes and always read back as zero. All register accesses must be SINGLE word (32-bit) operations through the AHB bus protocol. Accesses of any other size will have indeterminate results. All Enhanced SDRAM Controller registers can be access only by one master at a time. Multi access to the Enhanced SDRAM Controller register causes undetermined behavior.

Reset state of each bit is shown underneath the bit field name. An asterisk indicates that the value is dependent on the operating mode selected during reset. Details are provided in the following bit field descriptions.

18.3.1 Memory Map

The ESDRAMC supports 64-Mbyte, 128-Mbyte, 256-Mbyte, 512-Mbyte, 1-Gbyte and 2-Gbyte, 4 bank, single data rate, synchronous DRAMs on two independent chip selects. Each chip selects defines a specific

memory address mapped as shown in Table 18-4. Table 18-3 shows the ESDRAMC memory map and Table 18-4 shows the ESDRAMC register definition.

Table 18-3. ESDRAMC Memory Map

Address	Register	Access	Reset	Section/Page
0xD800_1000 (ESDCTL0)	Enhanced SDRAM Control Register 0	R/W	0x0111_0080	18.3.3.1/18-13
0xD800_1004 (ESDCFG0)	Enhanced SDRAM Configuration Register 0	R/W	0x0076_EB3A	18.3.3.2/18-18
0xD800_1008 (ESDCTL1)	Enhanced SDRAM Control Register 1	R/W	0x8112_0080	18.3.3.1/18-13
0xD800_100C (ESDCFG1)	Enhanced SDRAM Configuration Register 1	R/W	0x007A_C727	18.3.3.2/18-18
0xD800_1010 (ESDMISC)	Enhanced SDRAM Miscellaneous Register	R/W	0x0000_0000	18.3.3.3/18-31
0xD800_1020 (ESDCDLY1)	Enhanced MDDR Delay Line 1 Configuration Debug Register	R/W	0x001C_0000	18.3.3.4/18-33
0xD800_1024 (ESDCDLY2)	Enhanced MDDR Delay Line 2 Configuration Debug Register	R/W	0x001C_0000	18.3.3.4/18-33
0xD800_1028 (ESDCDLY3)	Enhanced MDDR Delay Line 3 Configuration Debug Register	R/W	0x001C_0000	18.3.3.4/18-33
0xD800_102C (ESDCDLY4)	Enhanced MDDR Delay Line 4 Configuration Debug Register	R/W	0x001C_0000	18.3.3.4/18-33
0xD800_1030 (ESDCDLY5)	Enhanced MDDR Delay Line 5 Configuration Debug Register	R/W	0x001C_0000	18.3.3.4/18-33
0xD800_1034 (ESDCDLYL)	Enhanced MDDR Delay Line Cycle Length Debug Register	R	N/A	18.3.3.5/18-35

Table 18-4. ESDRAMC Memory Map

Address	Use	Access
0xA000_0000–0xAFFF_FFFF	CSD0 SDRAM or LPDDR memory region (256 Mbyte)	Read/Write
0xB000_0000–0xBFFF_FFFF	CSD1 SDRAM or LPDDR memory region (256 Mbyte)	Read/Write

18.3.2 Register Summary

Figure 18-2 shows the key to the register fields and Table 18-5 shows the register figure conventions.

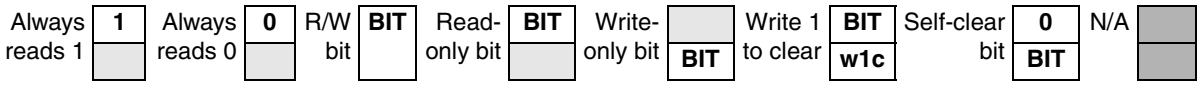


Figure 18-2. Key to Register Fields

Table 18-5. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read-only. Writing this bit has no effect.
W	Write-only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 18-6 shows the ESDRAMC register summary.

Table 18-6. ESDRAMC Register Summary

Name		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
0xD800_1000 (ESDCTL0)	R	SDE	SMODE			SP	ROW			0	0	COL		0	0	DSIZ	
	W																
	R	SREFR			0	PWDT		0	FP	BL	0	PRCT					
	W																
0xD800_1004 (ESDCFG0)	R	0	0	0	0	0	0	0	0	0	tXP		tWTR	tRP		tMRD	
	W																
	R	tWR	tRAS			tRRD		tCAS		0	tRCD			tRC			
	W																
0xD800_1008 (ESDCTL1)	R	SDE	SMODE			SP	ROW			0	0	COL		0	0	DSIZ	
	W																
	R	SREFR			0	PWDT		0	FP	BL	0	PRCT					
	W																

Table 18-6. ESDRAMC Register Summary (continued)

Name		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
0xD800_100C (ESDCFG1)	R	0	0	0	0	0	0	0	0	0	tXP		tWTR	tRP		tMRD	
	W																
	R	tWR	tRAS			tRRD		tCAS		0	tRCD			tRC			
	W																
0xD800_1010 (ESDMISC)	R	SDRAM_ RDY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	MA10_SHARE	LHD	MDDR_MDIS	0	MDDR EN	0	0
	W													MDDR_DL_RST		RST	
0xD800_1020 (ESDCDLY1)	R	SEL_DLY _REG_1	0	0	0	0	DLY_CORR_1										
	W																
	R	0	0	0	0	0	DLY_REG_1										
	W																
0xD800_1024 (ESDCDLY2)	R	SEL_DLY _REG_2	0	0	0	0	DLY_CORR_2										
	W																
	R	0	0	0	0	0	DLY_REG_2										
	W																
0xD800_1028 (ESDCDLY3)	R	SEL_DLY _REG_3	0	0	0	0	DLY_CORR_3										
	W																
	R	0	0	0	0	0	DLY_REG_3										
	W																
0xD800_102C (ESDCDLY4)	R	SEL_DLY _REG_4	0	0	0	0	DLY_CORR_4										
	W																
	R	0	0	0	0	0	DLY_REG_4										
	W																
0xD800_1030 (ESDCDLY5)	R	SEL_DLY _REG_5	0	0	0	0	DLY_CORR_5										
	W																
	R	0	0	0	0	0	DLY_REG_5										
	W																

Table 18-6. ESDRAMC Register Summary (continued)

Name		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
0xD800_1034 (ESDCDLYL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
	W																
	R	0	0	0	0	0	DLY_CYCLE_LENGTH										
	W																

18.3.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

NOTE

Memory may be viewed from either a Big Endian or Little Endian byte ordering perspective depending on the processor configuration (see Figure 18-3). In Big Endian mode (the typical default operating mode), the most significant byte (byte 0) of word 0 is located at address 0. For Little Endian mode, the most significant byte of word 0 is located at address 3. Within registers, bits are numbered within a word starting with bit 31 as the most significant bit. By convention, byte 0 of a register is the most significant byte regardless of Endian mode.

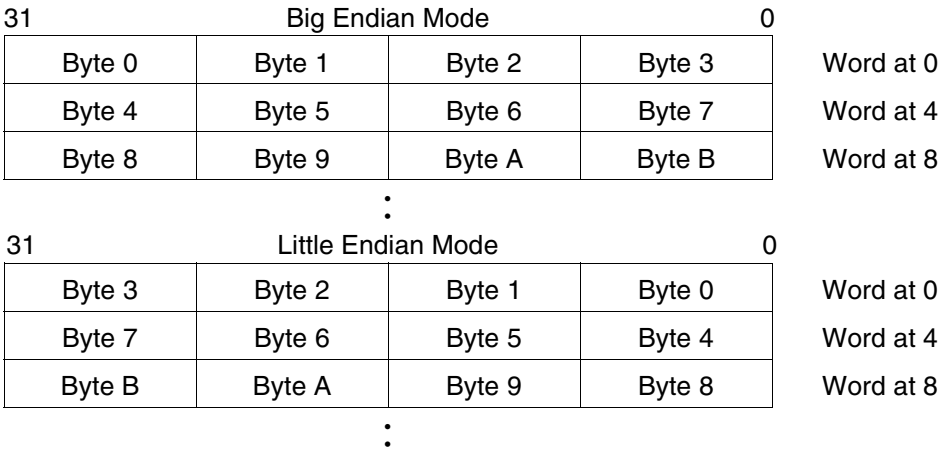


Figure 18-3. Data Organization in Memory

18.3.3.1 ESDCTL0 and ESDCTL1 Control Registers

This register contains the controlling various memory and control settings for the ESDRAMC. The bit assignments for the register are shown in Figure 18-4 and Figure 18-5. The field descriptions for the bit assignments are listed in Table 18-7.

0xD800_1000 (ESDCTL0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SDE	SMODE			SP	ROW			0	0	COL		0	0	DSIZ	
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SREFR			0	PWDT		0	FP	BL		PRCT					
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 18-4. Enhanced SDRAM Control Register (ESDCTL0)

0xD800_1008 (ESDCTL1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SDE	SMODE			SP	ROW			0	0	COL		0	0	DSIZ	
W																
Reset	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SREFR			0	PWDT		0	FP	BL		PRCT					
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 18-5. Enhanced SDRAM Control Register (ESDCTL1)

Table 18-7. Enhanced SDRAM Control Register (ESDCTL0/1) Field Descriptions

Field	Description
31 SDE	Enhanced SDRAM Controller Enable. This control bit enables/disables the Enhanced SDRAM controller. The reset value of the SDE0 bit is “0”, means the module is disabled for CSD0. Writing a one to those control bits enables the module, for both chip selects. Clearing those bits disables the module. By disabling both bits (SDE0 and SDE1) all clocks within the module shuts off (with the exception of register accesses). 0 Disabled 1 Enabled
30–28 SMODE	SDRAM Controller Operating Mode. This bit field determines the operating mode of the Enhanced SDRAM controller. In addition to the normal operating mode, the controller is capable of operating in the alternate operating modes listed below. These modes are primarily used for SDRAM initialization. Any access to the SDRAM memory space, while in one of the alternate modes, will result in the corresponding special cycle being run. Moving from Normal to any other mode does not close (precharge) any banks that may be open (activated). Under most circumstances, software should run a precharge-all cycle when transitional out of Normal Read/Write mode. Operating mode details are provided in Section 18.4, “Functional Description.” Reset initializes the operating mode to “Normal Read/Write”. 000 Normal Read/Write 001 Precharge Command 010 Auto-Refresh Command 011 Load Mode Register Command 100 Manual Self Refresh 101 – 111 Reserved
27 SP	Supervisor Protect. This control bit is used to restrict User accesses within the chip select region. The default at reset are that both user and supervisor accesses are allowed. 0 User mode accesses are allowed to this chip select region. 1 User mode accesses are prohibited. An attempted access to this chip select region while in User mode will result in a high HRESP[1] being returned back to the CPU. The chip select will not be asserted. ESDRAMC error response is generated by the M3IF arbitrator.
26–24 ROW	Row Address Width. This control field specifies the number of row addresses used by the memory array. This number does not include the bank, column, or data qualifier addresses. Parameters affected by the programming of this field include the page-hit address comparators and the bank address bit locations. 000 11 Row Addresses 001 12 Row Addresses 010 13 Row Addresses 011 14 Row Addresses 100 15 Row Addresses 101–111 Reserved
23–22	Reserved
21–20 COL	Column Address Width. This control field is used to specify the number of column addresses in the memory array and will determine the break point in the address multiplexer. Column width is the number of multiplexed column addresses and does not include bank, and row addresses, or addresses used to generate the DQM signals. The settings for the COL bit field encoding are shown in Table 18-8.
19–18	Reserved
17–16 DSIZ	SDRAM Memory Data Width. This field defines the width of the SDRAM memory and its alignment on the external data bus. 16-bit ports may be aligned to either the high or low half word to equalize capacitive loading on the bus. Data qualifier mask control outputs must be matched to the selected data bus alignment. Memories aligned to D[31:16] use DQM2 and DQM3. Memories aligned to D[15:0] use DQM0 and DQM1. 00 16-bit memory width aligned to D[31:16] 01 16-bit memory width aligned to D[15:0] (reset value for CSD0) 10 32-bit memory width (reset value for CSD1) 11 Reserved

Table 18-7. Enhanced SDRAM Control Register (ESDCTL0/1) Field Descriptions (continued)

Field	Description
15–13 SREFR	SDRAM Refresh Rate. This control bit field enables/disables SDRAM refresh cycles and controls the refresh rate. Refresh cycles are referenced to a 32 kHz clock. At each rising edge 1, 2, 4, 8 or 16 rows are refreshed as determined by this bit field. Multiple refresh cycles will be separated by the row cycle delay specified in the SRC control field. Refresh is disabled by hardware reset. SREFR Bit Field Encoding settings are listed in Table 18-9. Usage example see Table 18-20.
12	Reserved
11–10 PWDT	Power Down Timer. This field determines whether the SDRAM will be placed in a Power Down condition after a selectable delay from the last access. The Power Down time-out can be triggered on either the absence of an active bank (PWDT=01) or a clock (HCLK) count from the last access (PWDT=10 or 11). Count based time-outs do not force the SDRAM into an idle condition (for example., any active banks remain open). The Power Down timers feature is disabled by hardware reset. See Section 18.4.5.3, “Precharge Power Down Mode” and Section 18.4.5.4, “Active Power Down Mode” for a comprehensive description of this operating mode. A listing of the PWDT Bit Field Encoding is shown in Table 18-10.
9	Reserved
8 FP	Full Page. This bit should be set to 1 if the Burst Length of the SDRAM connected to the CSD has been configured to Full-Page mode. This bit is needed since ESDRAMC needs to induce a BURST TERMINATE (BT) command to terminate early all accesses that are less than Full Page. 0 Burst Length of the external memory device is not set to Full Page. 1 Burst Length of the external memory device is set to Full Page. **Note:** Full page mode is not supported when LPDDR external devices are used.
7 BL	Burst Length. This bit configures the access burst length. For proper operation the ESDRAMC burst length configuration must match the external SDRAM/LPDDR memory device (configured via special operating mode Load Mode Register). If this bit is set to 1 means that the external memory device connected to the CSD have been configured to Burst Length of 8. If this bit is cleared to 0, means that the external memory device connected to the CSD have been configured to Burst Length of 4. The settings for the Burst Length Bit Field Encoding is listed in Table 18-11.
6	Reserved
5–0 PRCT	Precharge Timer. Precharges a bank after 2xPRCT clocks (HCLK, up to 133 MHz) of no activity. Table 18-12 illustrates the PRCT bit field encoding. “Closing” (due to precharge command) the last used/open row in any non active bank within a chip select reduces the power consumption of the external memory device. The power saving is device dependent, and one should consult/examine the external memory device specification for more details on power consumption reduction. If PRCT is enabled, a PRECHARGE command is issued after approximately number of cycles (as shown in Table 18-12) of non activity to one of the SDRAM/LPDDR banks. The number of cycles before the PRECHARGE command is issued depends on command bus (WE, RAS, CAS and CSD) availability (means there is no active access to other bank) and the memory timing parameters.

Table 18-8. COL Bit Field Encoding

COL[1:0]	Number of Column Addresses
00	8
01	9
10	10
11	Reserved

Table 18-9. SREFR Bit Field Encoding

SREFR[2:0]	Rows Each Refresh Clock	# Rows/64 mS @ 32 kHz	Row Rate @ 32 kHz
000	Refresh Disabled (bit field reset value)		
001	1	2048	31.25 µs
010	2	4096	15.62 µs
011	4	8192	7.81 µs
100	8	16384	3.91µs
101	16	32768	1.95 µs
110	Reserved		
111	Reserved		

Table 18-10. PWDT Bit Field Encoding

PWDT[1:0]	Power Down Time-out	Memory Device Operating Mode
00	Disabled (bit field reset value)	Run Mode
01	Any time no banks are active	Precharge Power Down
10	64 clocks (HCLK) after completion of last access[1]	Active Power Down
11	128 clocks (HCLK) after completion of last access[2]	Active Power Down

1 This setting can't be used if the PRCT (precharge timer) is enabled.

Table 18-11. Burst Length Bit Field Encoding

BL	SDR SDRAM	LPDDR SDRAM
0	4[1]	Reserved
1[2]	8	8

1 For 16-bit SDRAM memory devices a BL setting of 4 is not supported.

2 Bit field reset value

Table 18-12. PRCT Bit Field Encoding

PRCT[5:0][1]	Precharge Timer[2]
000000	Disabled (Bit field reset value)
000001	2 clocks to precharge
000010	4 clocks to precharge
000011	6 clocks to precharge
000100	8 clocks to precharge
000101	10 clocks to precharge
000110	12 clocks to precharge

Table 18-12. PRCT Bit Field Encoding (continued)

PRCT[5:0][1]	Precharge Timer[2]
000111	14 clocks to precharge
001000	16 clocks to precharge
....	—
010000	32 clocks to precharge
....	.....
100000	64 clocks to precharge
....	.....
111111	126 clocks to precharge

1 PRCT can be used only if PWDT is disabled (“00”) or set to “any time no banks are active (“01”).” PRCT can’t be used with any other PWDT settings.

2 Number of clocks is approximate and it depends on external bus availability.

18.3.3.2 ESDRAMC Configuration Registers (ESDCFG0 /ESDCFG1)

The bit assignments for the Configuration registers are shown in Figure 18-6 and Figure 18-7. The field descriptions for the registers is listed in Table 18-13.

0xD800_1004 (ESDCFG0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	tXP		tWTR	tRP		tMRD	
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	tWR		tRAS		tRRD		tCAS		0	tRCD			tRC			
W																
Reset	1	1	1	0	1	0	1	1	0	0	1	1	1	0	1	0

Figure 18-6. Enhanced SDRAM Configuration Register 0 (ESDCFG0)

0xD800_100C (ESDCFG1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	tXP		tWTR	tRP		tMRD	
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	tWR	tRAS			tRRD		tCAS		0	tRCD			tRC			
W																
Reset	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	1

Figure 18-7. Enhanced SDRAM Configuration Register 1 (ESDCFG1)

Table 18-13. ESDCFG0/ESDCFG1 Field Descriptions

Field	Description
31–23	Reserved
22–21 t_{XP}	LPDDR exit power down to next valid command delay. This control field determines the minimum delay between a valid command is issued to the LPDDR after exiting power down mode. The value programmed in t_{XP} is the number of clocks inserted after exiting power down mode and any subsequent new valid command. An example timing diagram for t_{XP} can be found in Figure 18-19. 00 1 clock delay before new COMMAND issued to LPDDR after power down mode exit 01 2 clock delay before new COMMAND issued to LPDDR after power down mode exit 10 3 clock delay before new COMMAND issued to LPDDR after power down mode exit 11 4 clock delay before new COMMAND issued to LPDDR after power down mode exit
20 t_{WTR}	tLPDDR WRITE to READ Command Delay. Data for any WRITE burst may be followed by a subsequent READ command. To follow a WRITE without truncating the WRITE burst, t_{WTR} should be set as shown in Figure 18-9. The LPDDRC will automatically induce a t_{WTR} number of idle cycles between a WRITE followed by a READ command. The t_{WTR} should be configured according to the LPDDR device being used. The t_{WTR} is referenced from the first positive clock edge after the last data-in pair. 0 1 clock 1 2 clocks
19–18 t_{RP}	SDRAM Row Precharge Delay. This control bit determines the number of idle clocks inserted between a precharge command and the next row activate command to the same bank. Hardware reset initializes the controller to insert 3 clocks. Following a PRECHARGE command, a subsequent command to the same bank cannot be issued until t_{RP} is met. 00 1 clock 01 2 clocks 10 3 clocks[1] 11 4 clocks
17–16 t_{MRD}	tMRD. SDRAM Load Mode Register to ACTIVE Command. This control bits determines the minimum number of idle clocks required between a Load-Mode-Register (LMR) command to ACTIVE. Hardware reset initializes the controller to insert 2 clocks. 00 1 clock 01 2 clocks[2] 10 3 clocks 11 4 clocks

Table 18-13. ESDCFG0/ESDCFG1 Field Descriptions (continued)

Field	Description
15 t_{WR}	SDRAM WRITE to PRECHARGE Command. Data for a fixed length WRITE burst may be followed by, or truncated with, a PRECHARGE command to the same bank (provided that auto precharge was not activated), and a full-page WRITE burst may be truncated with a PRECHARGE command to the same bank. The PRECHARGE command should be issued t_{WR} after the clock edge at which the last desired input data element is registered. t_{WR} control bit determines the number of idle clocks inserted between the last desired input data element and the next PRECHARGE command, as shown in Figure 18-11. The t_{WR} Bit field encoding is listed on Table 18-14. **Note:** The auto precharge mode requires a t_{WR} of at least one clock plus time, regardless of frequency.
14–12 t_{RAS}	SDRAM ACTIVE to PRECHARGE Command. These control bits determine the minimum number of clocks required between a ACTIVE to PRECHARGE command to the same bank. Hardware reset initializes the controller to insert 6 clocks. Following a ACTIVE command, a subsequent PRECHARGE command to the same bank cannot be issued until t_{RAS} is met. Figure 18-12 presents an example of a single read (without auto precharge). It should be noticed that the PRECHARGE command is not allowed at T3 and at T4, since t_{RAS} will be violated (for t_{RAS}= 4 clock cycles). Note that t_{RAS} also defines the minimum period of time that the SDRAM must remain in self refresh mode, as it shown in Figure 18-13. 000 1 clock 001 2 clocks 010 3 clocks 011 4 clocks 100 5 clocks 101 6 clocks[2] 110 7 clocks 111 8 clocks
11–10 t_{RRD}	ACTIVE Bank A to ACTIVE Bank B Command. A subsequent ACTIVE command to a different row in the same bank can only be issued after the previous active row has been “closed” (precharged). A subsequent ACTIVE command to another bank can be issued while the first bank is being accessed, which results in a reduction of total row-access overhead. The minimum interval between successive ACTIVE commands to different banks is defined by t_{RRD} as shown in Figure 18-14 (for t_{RRD}=3). The t_{RRD} bits field encoding is listed below and it determines the number of idle clocks inserted between consecutive ACTIVE commands to different banks. 00 1 clock active to active (different banks) 01 2 clocks active to active (different banks)[2] 10 3 clocks active to active (different banks) 11 4 clocks active to active (different banks)
9–8 t_{CAS}	SDRAM CAS Latency. This field determines the latency between a read command and the availability of data on the bus, as shown in Figure 18-15. This field does not affect the second and subsequent data words in a burst. This control field has no effect on write cycles. CAS latency is initialized to 3 clocks following a hardware reset. 00 3 clocks only for LPDDR SDRAM CAS latency[2] 01 Reserved[3] 10 2 clocks SDR and LPDDR SDRAM CAS latency 11 3 clocks SDR and LPDDR SDRAM CAS latency[2]
7	Reserved

Table 18-13. ESDCFG0/ESDCFG1 Field Descriptions (continued)

Field	Description
6–4 t_{RCD}	SDRAM Row to Column Delay. This field determines the number of clocks inserted between a row activate command and a subsequent read or write command to the same bank. Hardware reset initializes the delay to 3 clocks. 000 1 clock row to column delay 001 2 clocks row to column delay 010 3 clocks row to column delay[4] 011 4 clocks row to column delay 100 5 clocks row to column delay 101 6 clocks row to column delay 110 7 clocks row to column delay 111 8 clocks row to column delay
3–0 t_{RC}	SDRAM Row Cycle Delay. This control field determines the minimum delay between a refresh and any subsequent refresh or read/write access. This delay corresponds to the minimum row cycle time captured in the t_{RC}/t_{RFC} memory timing specification. The value programmed in t_{RC} is the number of clocks inserted between the refresh and subsequent refresh/activate command. An example timing diagram for t_{RC} can be found in Figure 18-18. The bit field settings are listed on Table 18-16. **Note:** The t_{RC} control field is not used to enforce t_{RC} timing for row activate to row activate within the same bank as this is implicitly guaranteed by the sum of t_{RCD} + t_{CAS} + t_{RP}.Use regular paragraphs to summarize register function, then use the following special styles to define bit and field function.

1 Reset value for CSD0—optimal @ 133 MHz.

2 This is a LPDDR SDRAM only configuration. The external LPDDR SDRAM need to be configured to CAS latency 3, and ESDRAMC will delay the data (during READ cycles) toward the master by and additional HCLK cycle.

3 CAS1 is not supported due to shared DQM pads in a system

4 Reset value for ESDCFG1: optimal @ 133 MHz.

NOTE

ESDRAMC configuration registers reset value defines timing parameters to meet the memory device optimal timing requirements at 133 MHz. If the external memory device operates at a lower frequency the user should reconfigure the timing parameters (according to the device electrical characteristics and operating conditions) for optimal performance.

Table 18-15 lists and summarizes the Enhanced SDRAM Controller configurable set of timing parameters for the SDRAM and LPDDR devices.

Table 18-14. tWR Bit Field Encoding

tWR	Write to Precharge (SDRAM)	Write to Precharge (LPDDR)[1]
0	1 clock	2 clocks
1	2 clocks[2]	3 clocks[2]

1 Relevant in case that MDDR_EN bit is set, that is, the external memory device is a LPDDR.

2 Reset value for CSD0 and CSD1.

Table 18-15. Configurable SDRAM/LPDDR Timing Parameters

Symbol	Description	Relevancy	Optimal Values at 133 MHz
t_{MRD}	Load Mode Register command to ACTIVE/REFRESH command	Always	2 cycles
t_{WR}	Write recovery time (write to precharge)	Commands to Same Bank	2 cycles
t_{RAS}	ACTIVE to PRECHARGE command	Commands to Same Bank	6 cycles
t_{RRD}	ACTIVE bank A to ACTIVE bank B command	Commands to Different Banks	2 cycles
t_{CAS}	READ to DATA out period (known as CAS LATENCY)	Always	3 cycles
t_{RP}	PRECHARGE command period	Commands to Same Bank	3 cycles
t_{RCD}	ACTIVE to READ or WRITE delay	Commands to Same Bank	3 cycles
t_{RC}	ACTIVE to ACTIVE command period	Commands to Same Bank	10 cycles
t_{WTR}	LPDDR READ to WRITE command delay	Command to Same Bank	2 cycles
t_{XP}	LPDDR EXIT power down to next valid command delay	Always	4 cycles

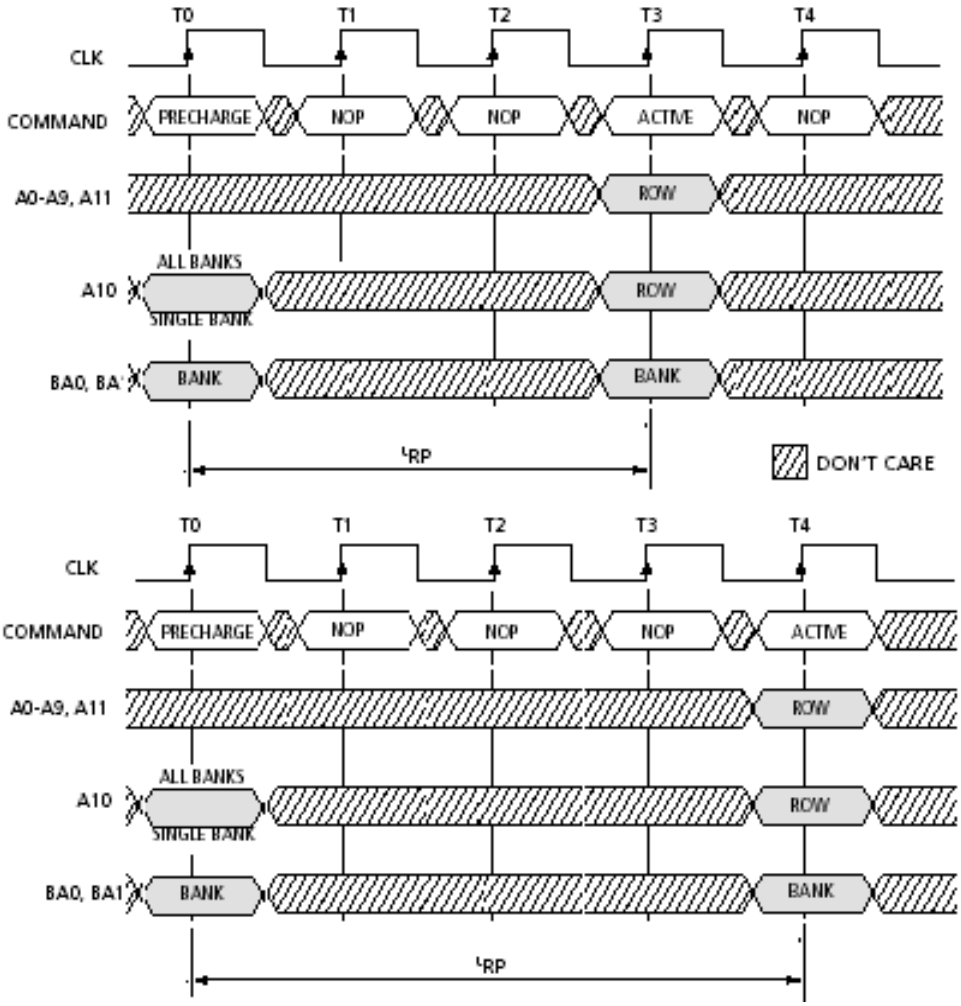


Figure 18-8. tRP—Precharge Delay Timing

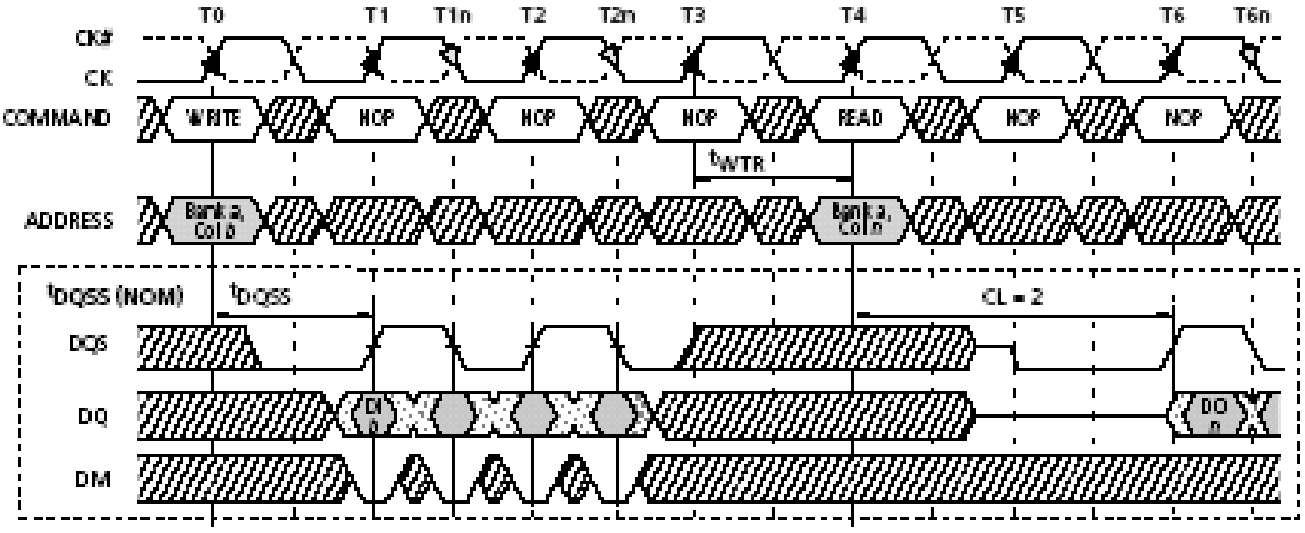


Figure 18-9. tWTRtRP Bit Field Encoding

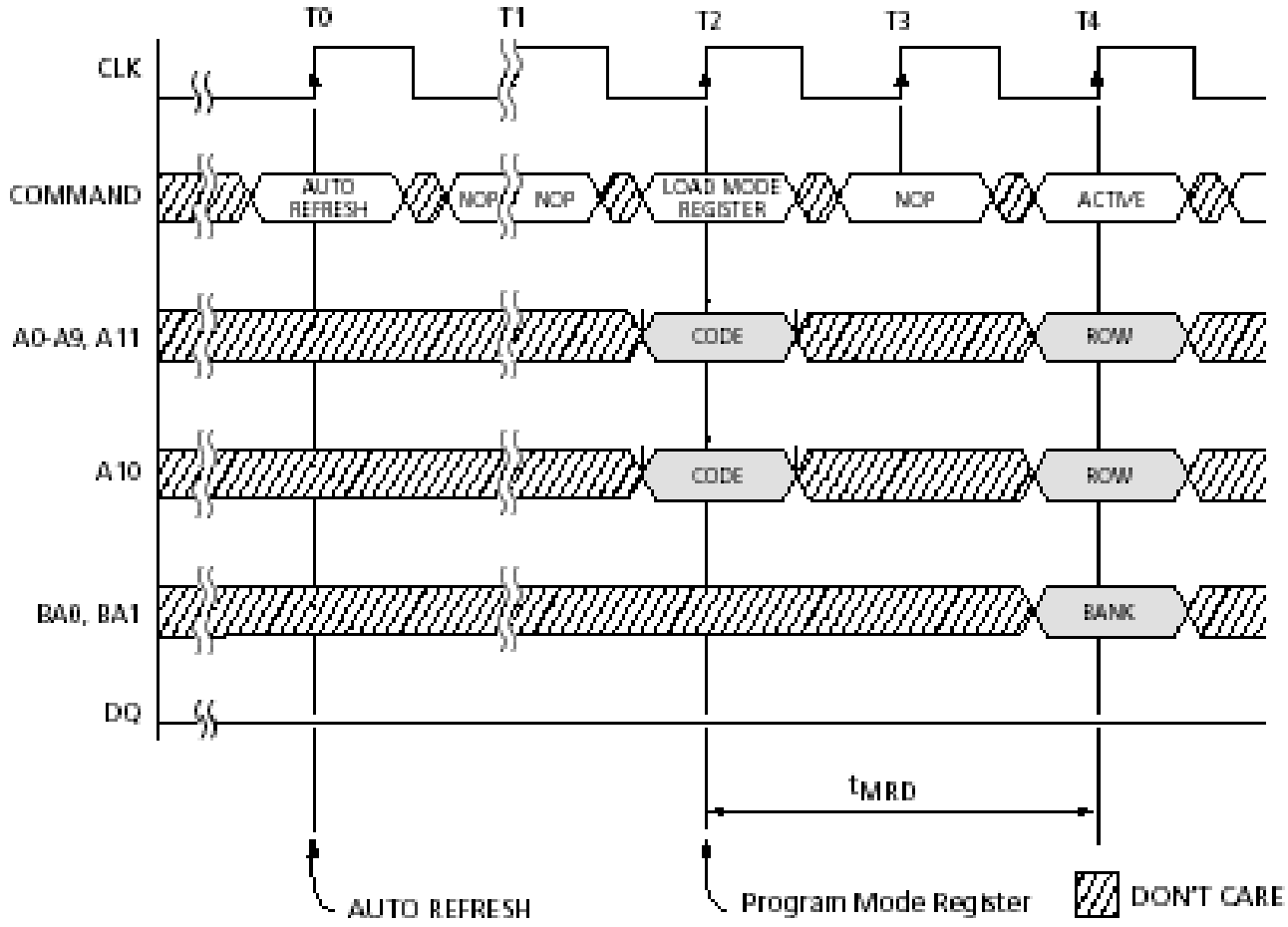


Figure 18-10. tMRD—SDRAM Load Mode Register to Active Command Timing Diagram

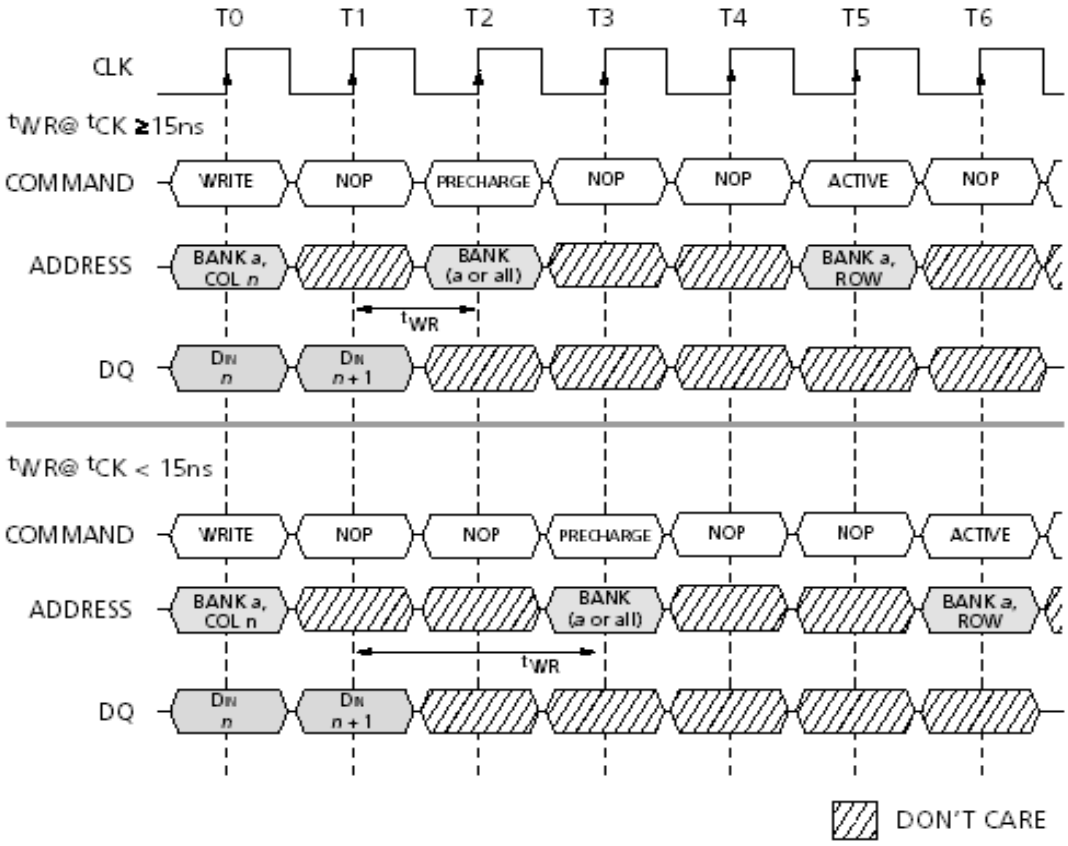


Figure 18-11. tWR—WRITE to PRECHARGE Timing Diagram

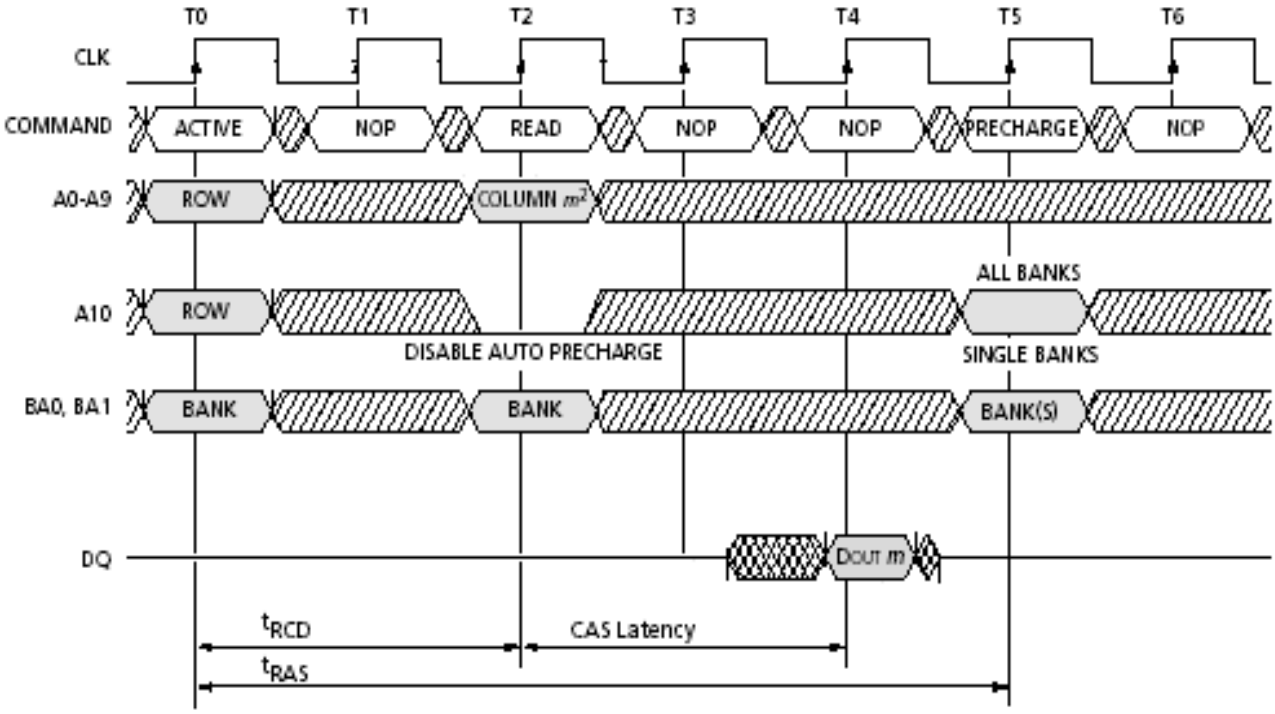


Figure 18-12. t_{RAS}—SDRAM ACTIVE to PRECHARGE Command Timing Diagram

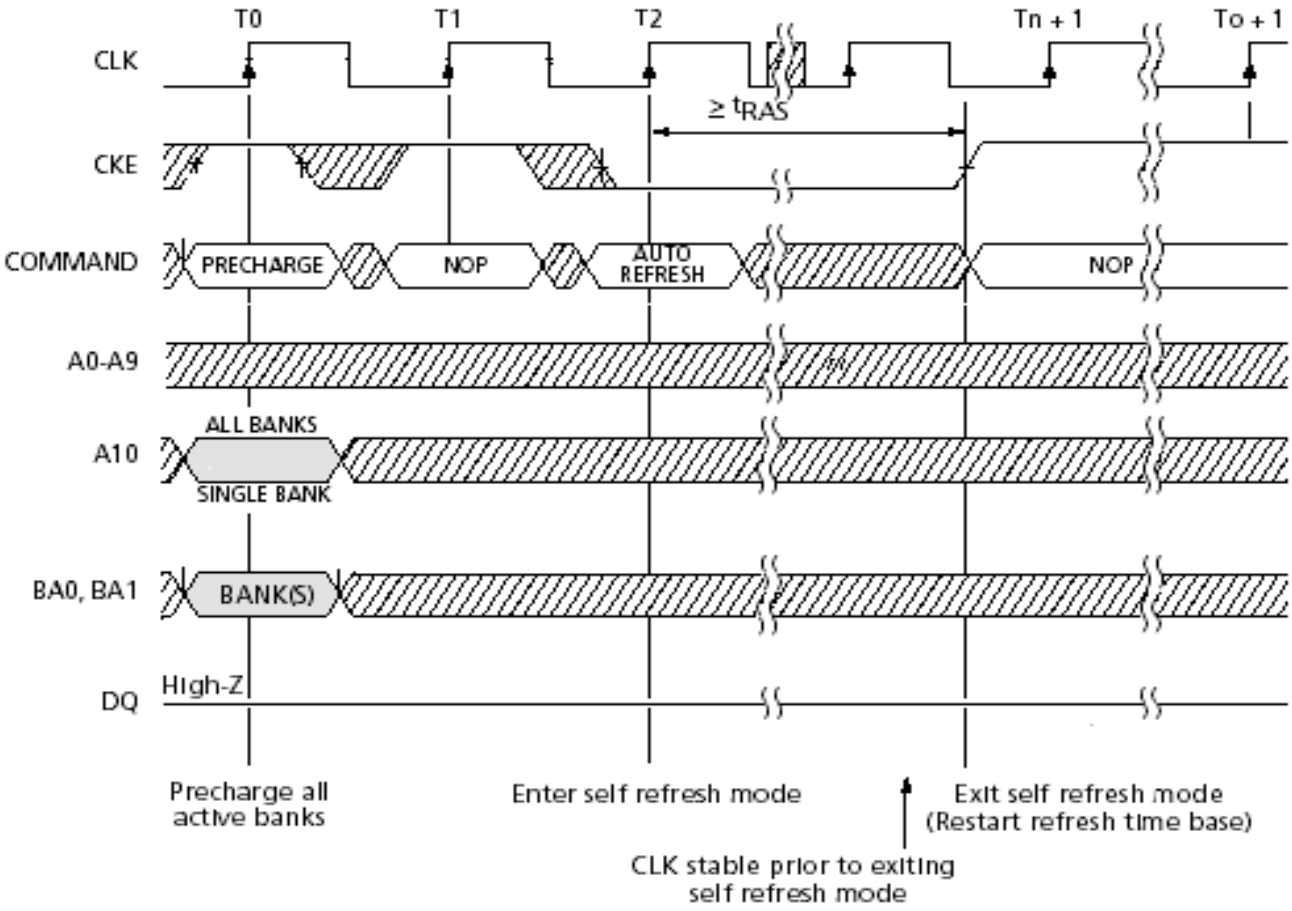


Figure 18-13. t_{RAS}—SELF REFRESH Mode Minimum Time Period

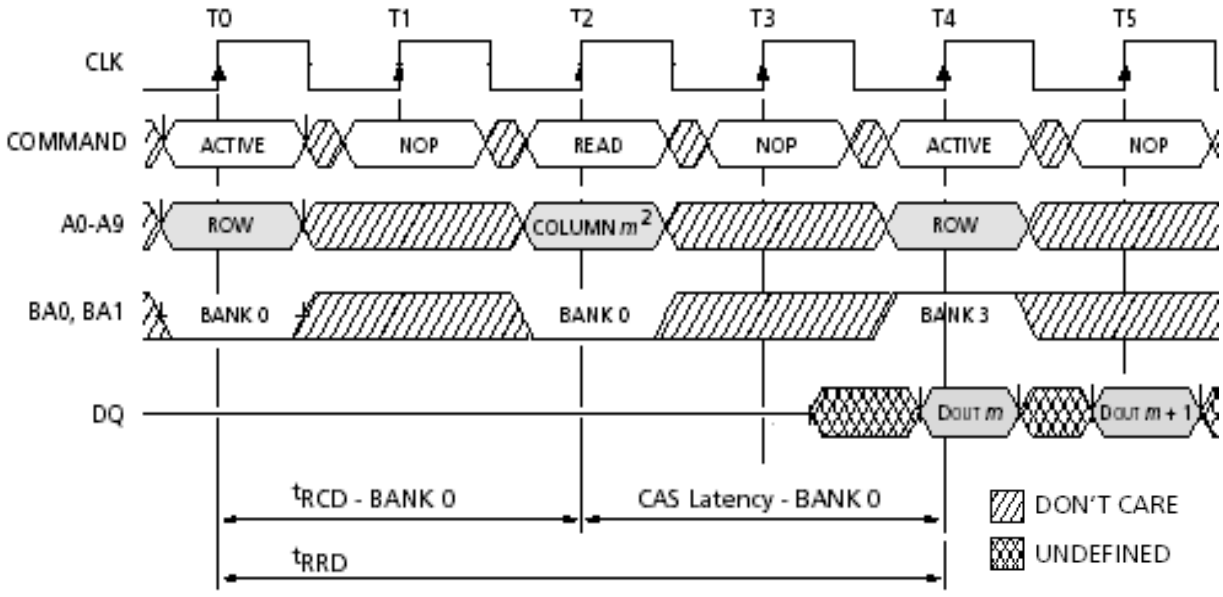


Figure 18-14. t_{RRD}—Alternating Bank Read Access

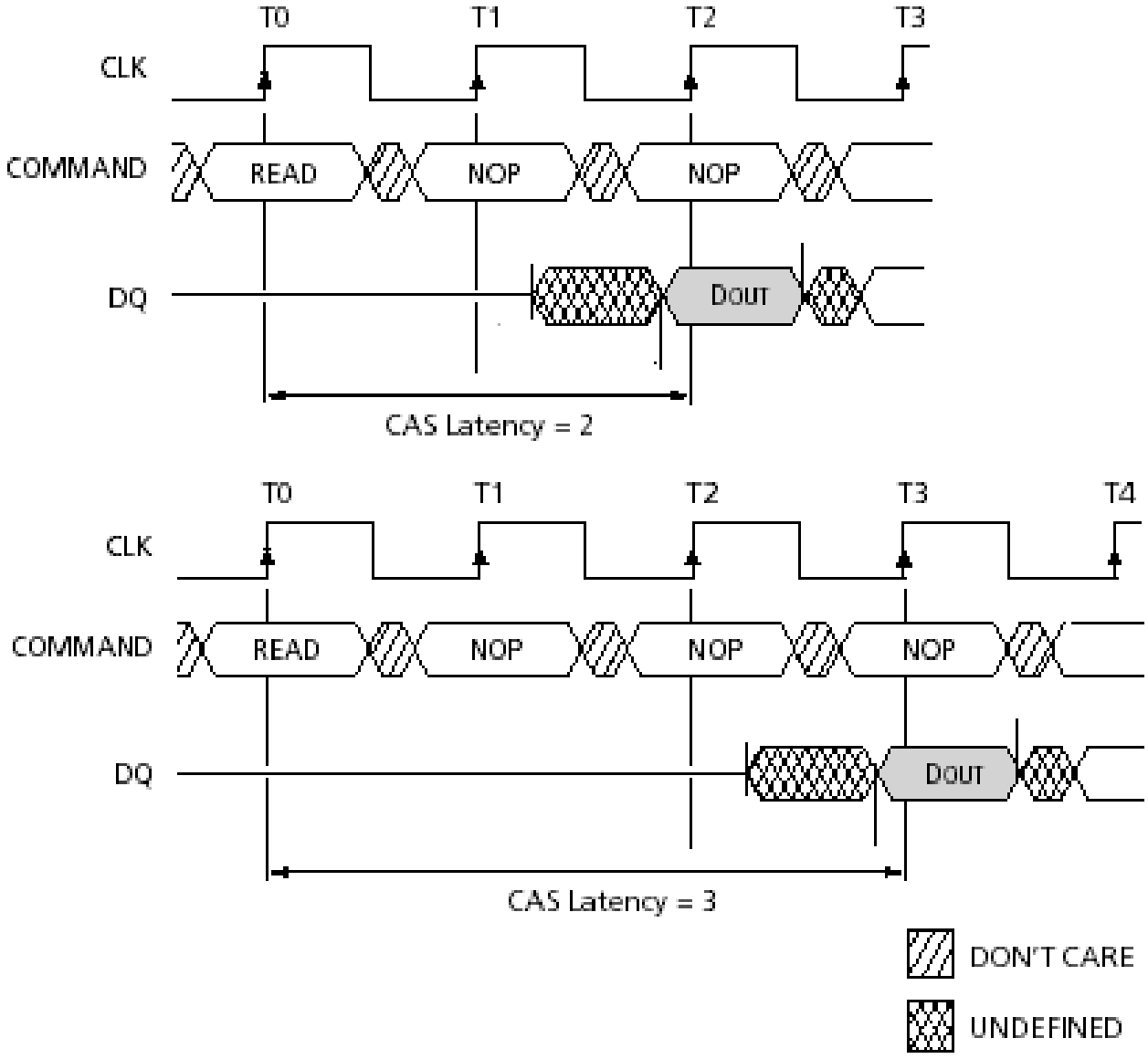


Figure 18-15. $_{SDR}$ CAS Latency Timing

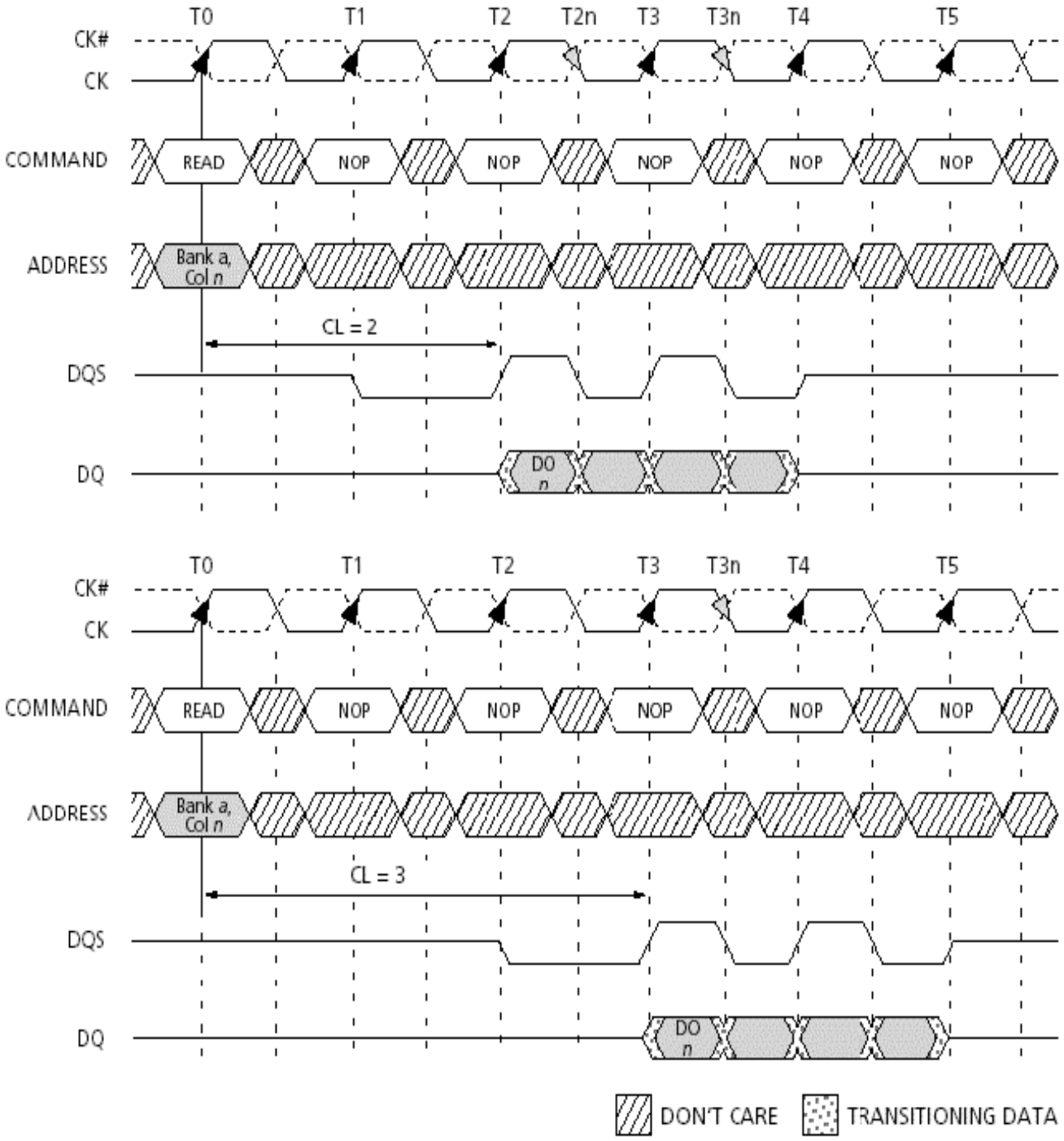


Figure 18-16. Mobile LPDDR CAS Latency Timing

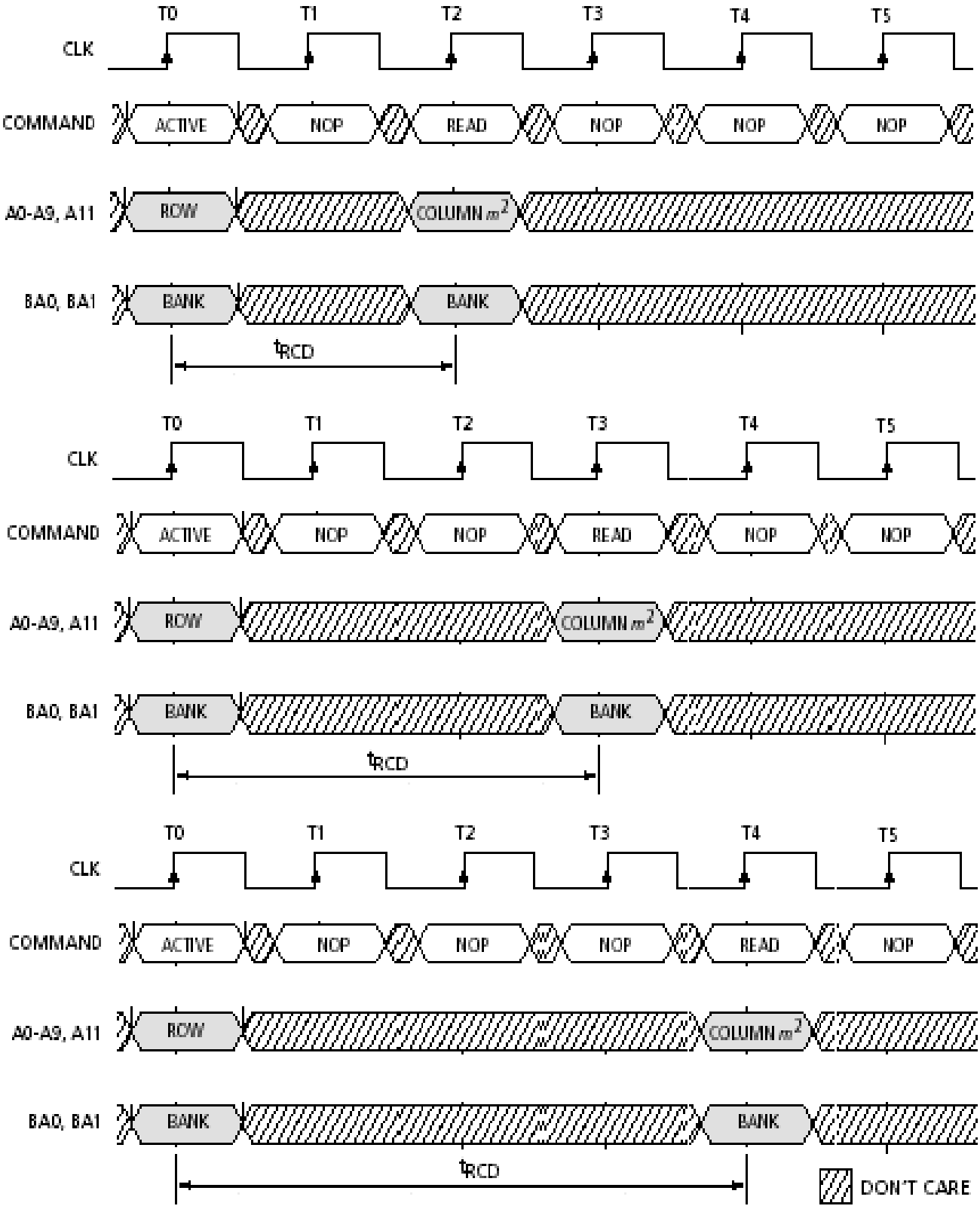


Figure 18-17. t_{RCD}—Row to Column Delay Timing

Table 18-16. t_{RC} Bit Field Encoding

t_{RC}[2:0]	Delay
0000	20 clocks
0001	2 clocks
0010	3 clocks
0011	4 clocks (reset value for CSD0)
0100	5 clocks
0101	6 clocks
0110	7 clocks
0111	8 clocks
1000	9 clocks
1001	10 clocks[1]
1010	11 clocks
1011	12 clocks
1100	13 clocks
1101	14 clocks
1110	14 clocks
1111	16 clocks

[1] Reset value for CSD0.

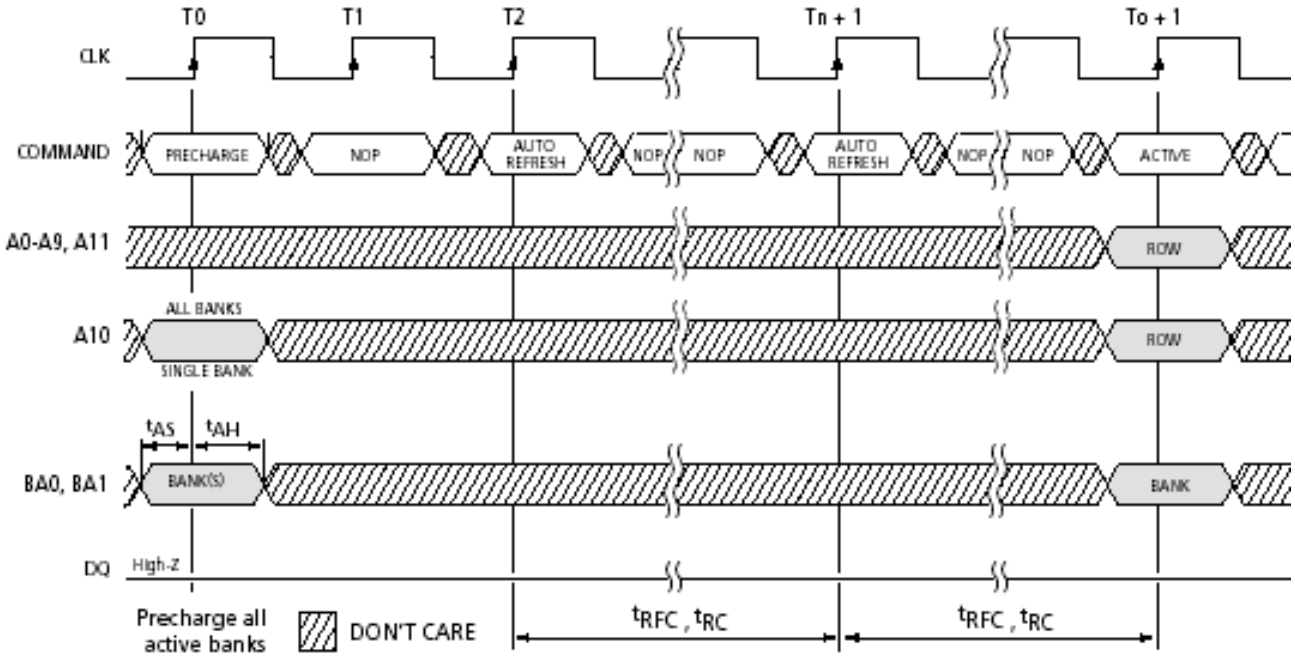


Figure 18-18. t_{RC}—Row Cycle Timing

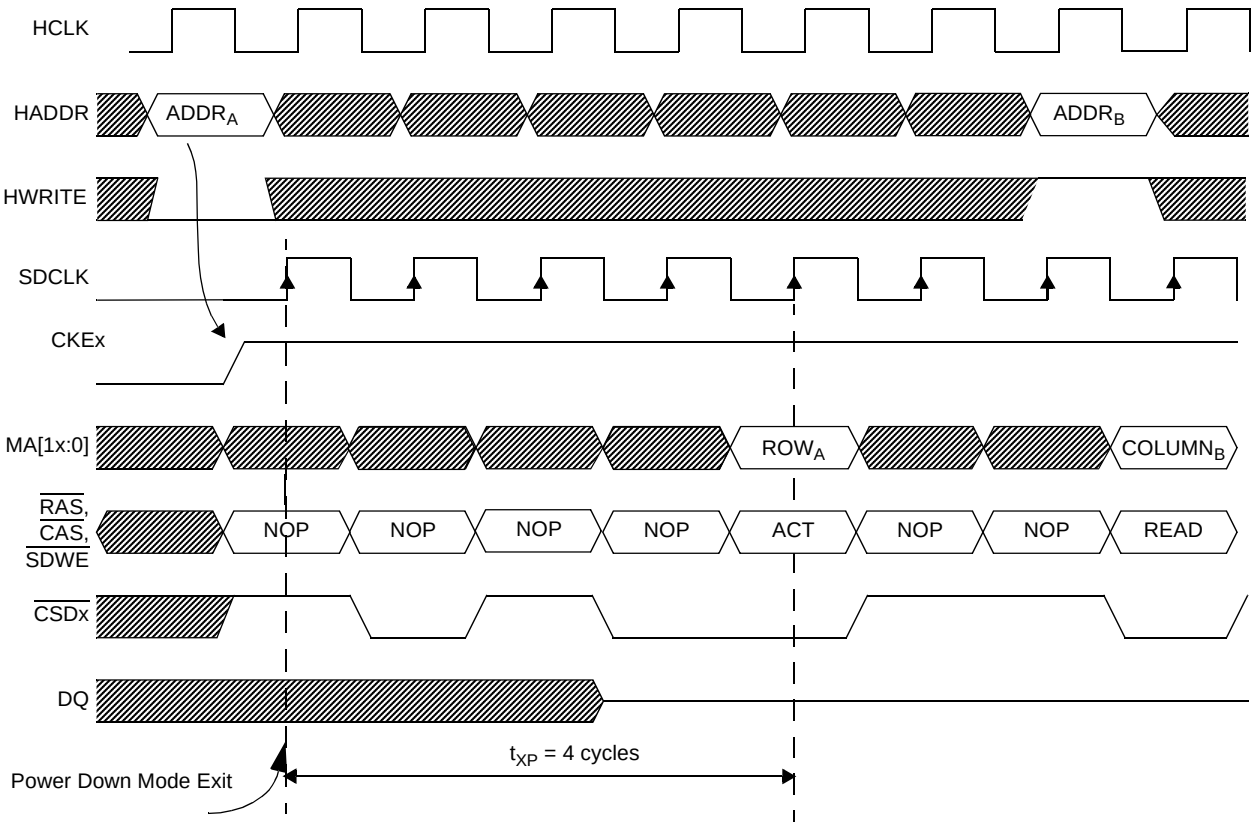


Figure 18-19. t_{XP}—New Command After Power Down Exit (4 Cycles)

18.3.3.3 ESDMISC Miscellaneous Register (ESDMISC)

This register controls various memory and settings for ESDRAMC. The bit assignments for this register is shown in Figure 18-20 and the field descriptions for the bit assignments are listed in Table 18-17.

0xD800_1010 (ESDMISC) Access: User read-write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	MA10 _SHA RE	LHD	MDD R_M DIS	0	MDD R EN	0	0
W													MDDR_ DL_RST		RST	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-20. ESDRAMC Miscellaneous Register (ESDMISC)

Table 18-17. Enhanced SDRAM Control Register (ESDCTL0/1) Field Descriptions

Field	Description
31 SDRAMRDY	External SDRAM/LPDDR Device Status. This is a read-only status bit, that indicates the state of the external memory device(s). This bit is cleared at reset. This bit is set after the 200 µs SDRAM/LPDDR external memory wakeup period. After the wakeup period the software can start the external memory initialization (as described in Section 18.5.4, "SDRAM/LPDDR Initialization Sequence"). 0 SDRAM/LPDDR external device is not ready for use. 1 SDRAM/MMDR external device is ready for use.
30–7	Reserved
6 MA10_SHARE	MA10 share. Need to be enabled if MA10 address line is shared with other memory controllers address line. If enabled, the ESDRAMC will request the address line from the M3IF before issuing the Precharge all command (during auto refresh cycles). After the Precharge all command is completed the ESDRAMC will remove the request. If MA10 share is disable, the ESDRAMC will execute the Precharge all command without requesting the MA10 address line, by assuming that MA10 address line is dedicated to ESDRAMC. 0 MA10 Share Disable 1 MA10 Share Enable
5 LHD	Latency Hiding Disable. This bit disables the command anticipation (latency hiding) mechanism. If this bit is set, the M3IF/ESDCTL will operate in MIF1 (non-optimized) mode as shown in Figure 18-29 and Figure 18-30. The first memory command of a new access is sent to the memory only after the previous access is completed, For example: the last data word of a burst has been read or written. The reset value of this bit is 0, meaning that the latency bidding is enabled (M3IF/ESDCTL works in MIF2 mode) as described in Section 18.4.1, "Enhanced SDRAM Controller Optimization Strategy"). 0 Latency Hiding Enable 1 Latency Hiding Disable
4 MDDR_MDIS	LPDDR Delay Line Measure Disable. This is a read/write bit, that, if set, disables the delay line measure unit. After reset, this bit is cleared, meaning the delay line measure unit is enabled. The measure time period is estimated to be around 2000 clock cycles of the AHB HCLK. 0 LPDDR delay line measure unit is enabled. 1 LPDDR delay line measure unit is disabled.
3 MDDR_DL_RST	LPDDR Delay Line Soft Reset. This is a write only bit, that if set the delay line unit is reset. After reset the delay unit will automatically (if LPDDR_MDIS is cleared) start a new measurement. 0 LPDDR Delay Line is not reset. 1 LPDDR Delay Line is reset.
2 MDDREN	Enable Mobile/Low Power DDR SDRAM. This bits activates the LPDDR Interface and enable the pipeline to work in LPDDR mode. 0 Enable Mobile SDR SDRAM operation. 1 Enable Mobile DDR SDRAM operation.
1 RST	Software Initiated Local Module Reset. This bit generate local module reset to the ESDRAMC. Writing a 1 to RST bit results in a one cycle reset pulse to the controller. This bit is always read as 0. All ESDRAMC registers are not affected by the software reset, in order to keep the REFRESH mechanism active as initially configured, so the SDRAM/LPDDR data is not violated. A burst terminate command is issued to the memory after the soft reset (to terminate any active bursts, in order to prevent potential contention on the DATA pads). During the software RESET an ERROR response (HRESP[1]=1) is broadcast to all masters with active access to the ESDRAMC by the Multi Master Memory Interface (M3IF) module and the M3IF arbitration pipeline is cleared. For detailed information on error response refer to M3IF module specification. **Note:** After soft reset, a Precharge all command must be issued prior to normal usage of the ESDRAMC. 0 Soft Reset is disabled. 1 Soft Reset is initiated.
0	Reserved

18.3.3.4 MDDR Delay Line 1–5 Configuration Debug Register

These debug registers controls delay line 1 to 5 functionality, that is, DQS[0] to DQS[4] delays are used during READ cycles for line 1 to 5 respectively. It allows to override/manually set the delay of DQS[0] to DQS[4] lines, that is used during READ cycles for BYTE[0] to BYTE[4] respectively. The delay line compensates for process variations, and produces a constant delay regardless of the process, temperature and voltage. The bit assignments for these registers are shown in Figure 18-21 to Figure 18-25 and the field descriptions for the bit assignments are listed in Table 18-18.

0xD800_1020 (ESDCDLY1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SEL_DLY_REG_1	0	0	0	0	DLY_CORR_1										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	DLY_REG_1										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-21. MDDR Delay Line 1 Configuration Debug Register

0xD800_1024 (ESDCDLY2) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SEL_DLY_REG_2	0	0	0	0	DLY_CORR_2										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	DLY_REG_2										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-22. MDDR Delay Line 2 Configuration Debug Register

0xD800_1028 (ESDCDLY3) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SEL_DLY_REG_3	0	0	0	0	DLY_CORR_3										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	DLY_REG_3										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-23. MDDR Delay Line 3 Configuration Debug Register

0xD800_102C (ESDCDLY4) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SEL_DLY_REG_4	0	0	0	0	DLY_CORR_4										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	DLY_REG_4										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-24. MDDR Delay Line 4 Configuration Debug Register

0xD800_1030 (ESDCDLY5) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SEL_DLY_REG_5	0	0	0	0	DLY_CORR_5										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	DLY_REG_5										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-25. MDDR Delay Line 5 Configuration Debug Register

Table 18-18. Enhanced MDDR Delay Line 5 Control Register (ESDCDLY5) Field Descriptions

Field	Description
31 SEL_DLY_REG_1–5	This bit selects the delay used by delay line 1–5. It selects between a quarter of a cycle (measured) minus delay line 1–5 correction factor field (DLY_CORR_1–5) and delay line 1–5 register value (DLY_REG_1–5). 0 Delay line 1value is, a quarter of a cycle (measured) minus the delay line 4 correction factor field. 1 Delay line 1value is, the value of delay line 4 register field (skipping the measurement).
30–27	Reserved
26–16 DLY_CORR_1–5	This field is the delay line 1–5 correction factor. The correction factor is used only if SEL_DLY_REG_1–5 is cleared. The correction factor value is used to compensate a minimum delay (in number of inverters units) from the measured delay (the minimum delay varies with the process, voltage and temperature changes).
15–11	Reserved
10–0 DLY_REG_1–5	This field is the delay (in number of inverters units) that will be used by delay line 1–5, if SEL_DLY_REG1–5 bit is set. Since the delay is process, temperature and voltage dependent, for a given value of this filed we get different delay values.

18.3.3.5 MDDR Delay Line Cycle Length Debug Register

MDDR Delay Line Cycle Length Debug Registers is a read only register that shows the number of inverters that “fit” in a cycle. The reset value is unknown, because after reset the register value is updated from the measured delay. The register value represents the number of inverters required to achieve a delay of one clock cycle, as a function of the IC conditions (temperature, voltage, frequency, process). The bit assignments for the register are shown in Figure 18-26 and the field descriptions for the bit assignments are listed in Table 18-19.

0xD800_1034 (ESDCDLYL) Access: User read-only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	DLY_CYCLE_LENGTH										
W																
Reset	0	0	0	0	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Figure 18-26. MDDR Delay Line Cycle Length Debug Register

Table 18-19. Enhanced MDDR Delay Line Cycle Length Debug Register (ESDCDLYL) Field Descriptions

Field	Description
31–11	Reserved
10–0 DLY_CYCLE_LENGTH	DLY_CYCLE_LENGTH shows the number of Inverters that “fit” in a cycle.

18.4 Functional Description

General Enhanced SDRAM Controller operating characteristics are addressed in this section. The discussion starts with the optimization strategy (latency hiding/command anticipation), continues with one of the most basic of all DRAM controller features, the address multiplexor. Following subsections explain operation out of reset, hardware refresh and the low power modes. More on, each of the Enhanced SDRAM Controller operating modes are described. The discussion includes details on basic operation, relationship to SDRAM/LPDDR operating modes, and any special precautions which need to be observed. State and timing diagrams are included where appropriate.

The Enhanced SDRAM Controller is designed to support a broad range of broad range of JEDEC standard SDRAM/LPDDR configurations including devices of 64-Mbyte, 128-Mbyte, 256-Mbyte, 512-Mbyte, 1-Gbyte and 2-Gbyte densities. Given the physical size constraints of the target applications, the design support memory devices with data widths of 16 and 32 bits. Table 18-20 summarizes the devices supported by the design. Only 4 bank devices are supported. 133-MHz system bus operation is possible with PC133 compliant Single or Double Data Rate memory devices.

Each of the Enhanced SDRAM Controller operating modes are described in this section. The discussion includes details on basic operation, relationship to SDRAM/LPDDR operating modes, and any special precautions which need to be observed. State and timing diagrams are included where appropriate.

Table 18-20. JEDEC Standard Single/Double Data Rate SDRAMs

	SDRAM Configurations—4-Bank Devices											
Size	64 MBit		128 MBit		256 MBit		512MBit[1]		1-GBit[1]		2-GBit[1]	
Bus size	16	32	16	32	16	32	16	32	16	32	16[2]	32
Depth	4M	2M	8M	4M	16M	8M	32M	16M	64M	32M	N/A	64M
Refresh Rows	4096	4096	4096	4096	8192	8192	8192	8192	16384	16384	N/A	16384
Refresh rate (us)	15.6	31.25	15.6	15.6	7.81	7.81	7.81	7.81	3.91	3.91	N/A	3.91
Refresh cycles	2	1	2	2	4	4	4	4	8	8	N/A	8
Row Address	12	11	12	12	13	13	13	13	14	14	N/A	14
Col. Address	8	8	9	8	9	8	10	9	10	9	N/A	10

1 Not ratified by JEDEC, row-column organization may change.

2 2-GB SDRAM/LPDDR (16-bit) are not supported/available.

18.4.1 Enhanced SDRAM Controller Optimization Strategy

SDRAM (SDR and LPDDR) memories provide high speed access by hiding the latency of consecutive memory accesses through a pipeline interface architecture. The resulting high bandwidth is achieved with a rather complex command interface and a large number of timing constraints that must be kept by the memory controller. SDRAM are organized in several independent banks. By issuing a row address and bank number to the memory device, the corresponding memory page is activated (opened). Consecutive read commands (together with the column address) into this memory page (same row address) have a low latency. Accessing another page in the same bank requires to close the open page by a PRECHARGE command, followed by the activation (ACTIVE command) of the new memory page (new row address).

In Figure 18-27 and Figure 18-28 examples for an SDR and LPDDR SDRAM read bursts are shown. Initially the cell in [COL a, ROW a] is active/open. Trying now to access/open the cell position [COL b, ROW b] requires first to close the old sell in [COL a, ROW a]. This is done with a precharge command (P on the figure). Now the access row address (ROW b) can be activated (A on the figure) before the column address (COL b) will be passed (read command R). The timing restrictions are:

- The ACTIVE command can be issued only tRP cycles after the PRECHARGE command.
- The READ command can be issued only tRCD cycles after the ACTIVE command.
- The first data is available only tCAS cycles after the READ command has been issued.

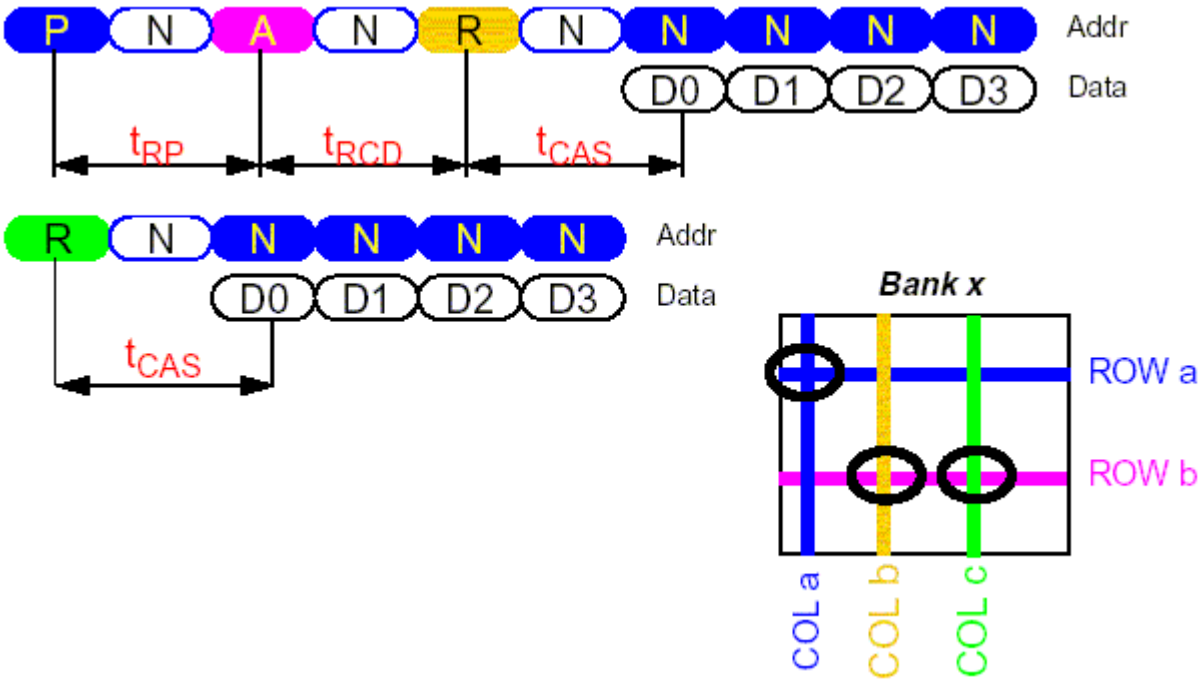


Figure 18-27. SDR SDRAM Read Burst Command Sequence Example

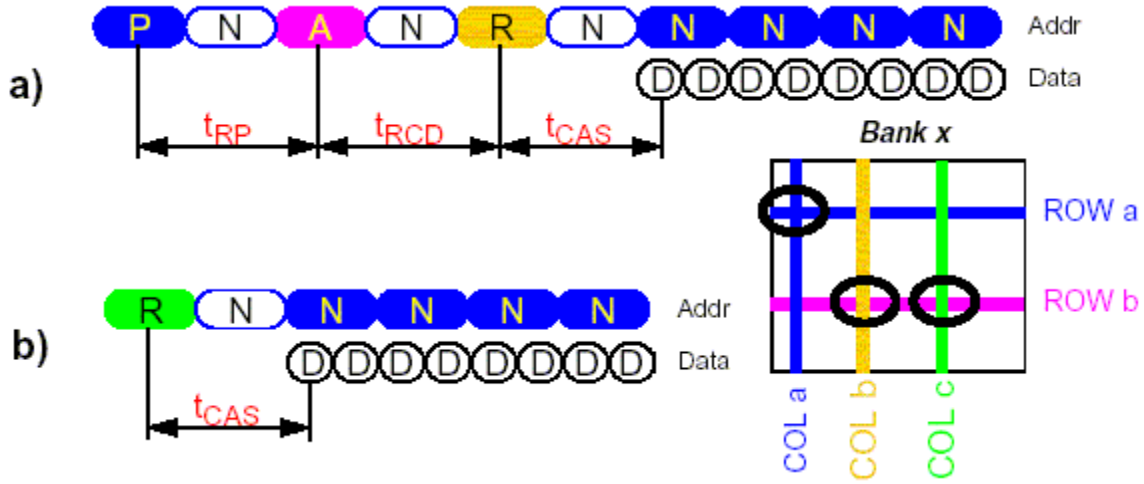
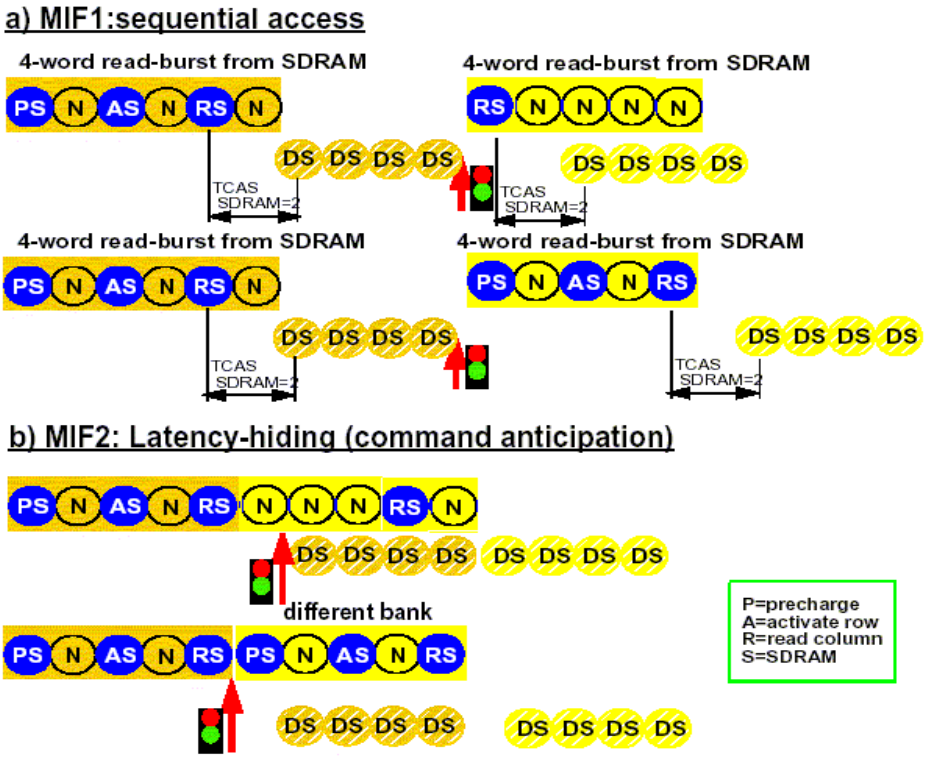


Figure 18-28. LPDDR SDRAM Read Burst Command Sequence Example

Hence, in those examples a 4-word (8-word in LPDDR will be converted to 4-word with twice data width) read burst requires 9 cycles (6 cycles latency from precharge command to the first word on the external bus) or 6-1-1-1. A read access from an already open row is shown as well, the read burst from target cell [ROW b, COL c] on takes only 5 cycles (2 cycles latency from the read command to the first word on the external bus) or 2-1-1-1.

Figure 18-29 (SDR) and Figure 18-30 (LPDDR) shows two different optimization strategies. First “no optimization” strategy (referred as MIF1) and second the “medium level optimization” (referred as MIF2). For SDR SDRAM each strategy, two examples for two consecutive read accesses are given:

- An 8-word burst from the SDRAM with CAS latency of 2 cycles followed by an 8-word burst from the same bank and row (different column) with CAS latency of 2 cycles.
- An 8-word burst from one bank in the SDRAM followed by an 8-word burst from a different bank to the same SDRAM.


a) MIF1:sequential access
4-word read-burst from SDRAM
PS N AS N RS N
DS DS DS DS
TCAS SDRAM=2
4-word read-burst from SDRAM
RS N N N N
DS DS DS DS
TCAS SDRAM=2
4-word read-burst from SDRAM
PS N AS N RS N
DS DS DS DS
TCAS SDRAM=2
4-word read-burst from SDRAM
PS N AS N RS
DS DS DS DS
TCAS SDRAM=2
b) MIF2: Latency-hiding (command anticipation)
PS N AS N RS N N N RS N
DS DS DS DS DS DS DS DS
different bank
PS N AS N RS PS N AS N RS
DS DS DS DS DS DS DS DS
P=precharge
A=activate row
R=read column
S=SDRAM

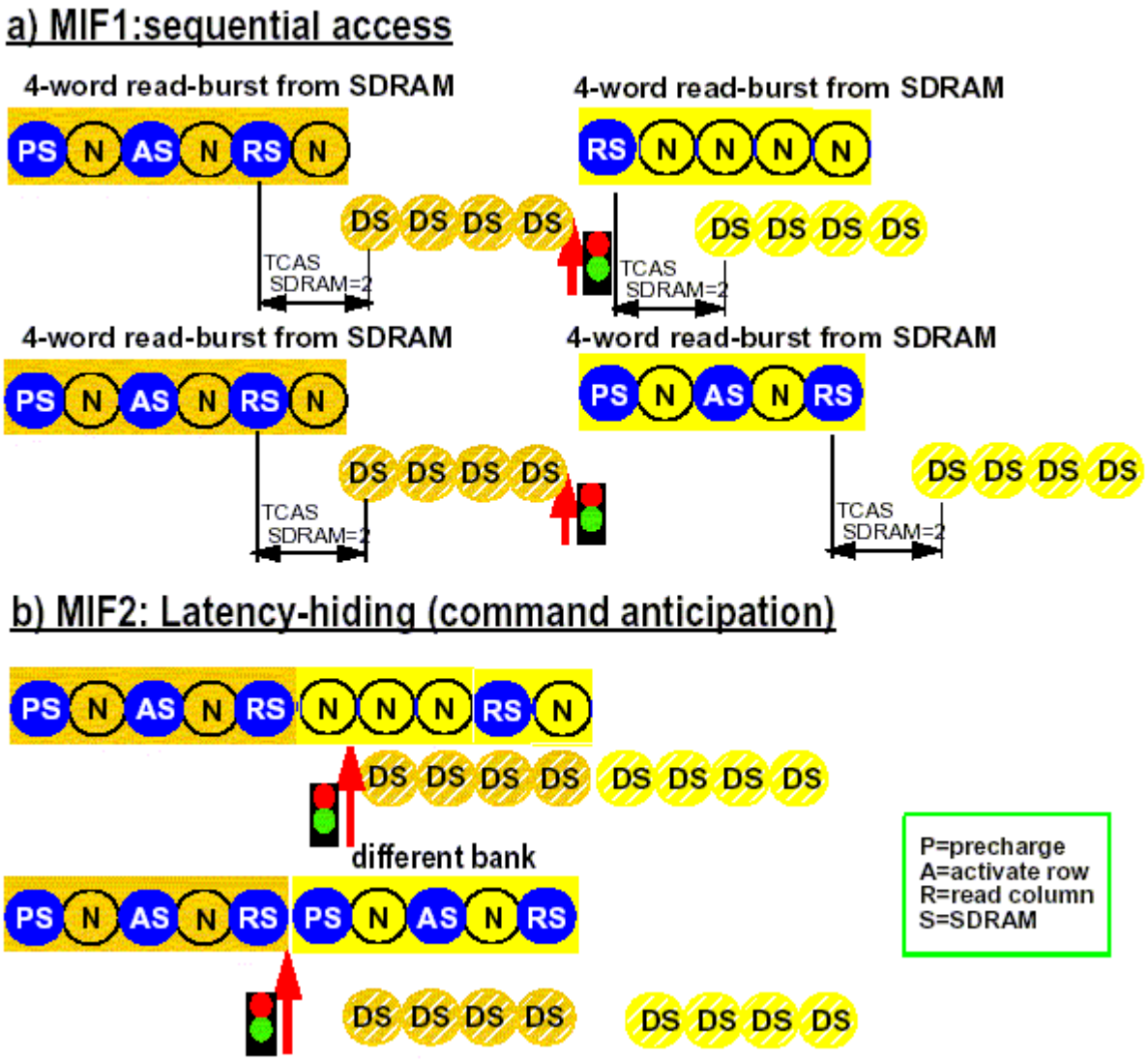


Figure 18-29. SDR SDRAM Optimization Strategies—MIF1 and MIF2 Examples

For LPDDR SDRAM each strategy, one example for two consecutive read accesses is given:

- a 8-word burst from one bank in the LPDDR SDRAM followed by a 8-word burst from a different bank to the same LPDDR SDRAM.

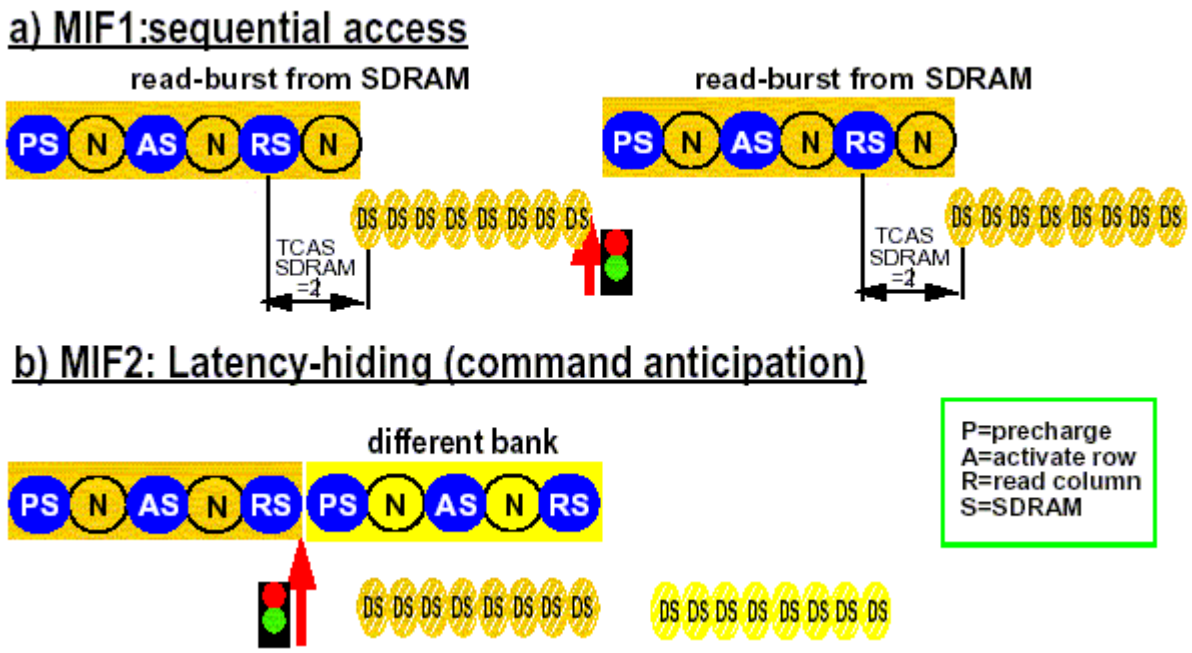


Figure 18-30. Mobile LPDDR SDRAM Optimization Strategies—MIF1 and MIF2 Examples

The fact that Enhanced SDRAM Controller handles two devices and that each of them features 4 independent memory banks opens the opportunity for the controller to optimize the access timing of consecutive memory accesses by trying to hide as much as possible the latency of the first data word in a burst. The latency hiding is possible in the cases that are summarized in Table 18-21.

Table 18-21. Possibilities for Latency Hiding

Current Burst Access	Next Burst Access
SDRAM bank x	SDRAM bank y
SDRAM bank x (row y, col z)	SDRAM bank x (row y, col w)
LPDDR SDRAM bank x	LPDDR SDRAM bank y

18.4.1.1 MIF1—No optimization/Sequential Accesses

This is the non-optimized case shown in Figure 18-29 and Figure 18-30. The first memory command of a new access is sent to the memory only after the previous access is completed, that is, the last data word of a burst has been read or written. It can be seen in this example that although the 2 memory accesses follow with no delay between them, the bandwidth usage for the command (address) and data busses is far from being optimal. This no optimization/ sequential accesses occurs in the following cases;

- Only one master active/present in a given system—in this case only sequential commands are possible since the given master need to receive/send (READ/WRITE) all data's for one access before it can proceed to the next access, that is, command anticipation is not possible.
- Low density accesses, such as non-overlapped/consecutive/continuos requests, means that the next SDRAM request starts after the previous request is completed. In this low SDRAM utilization only sequential accesses occurs.

- Large number of SINGLE or INCR (aborted after one data) accesses instead of burst type accesses. SINGLE/INCR refer to AMBA AHB bus protocol.
- The LHD (latency hiding disable) bit is set.

18.4.1.2 MIF2—Medium Level Optimization/Command Anticipation

This strategy is shown in Figure 18-29 and Figure 18-30. As soon as the address and command bus (refer as the SDRAM control bus) is no more used for issuing the previous memory access command, the controller can use this bus to start issuing the PRECHARGE/ACTIVE commands for the next scheduled memory access while the previous one is still active on the data bus. Two conditions limit the use of this optimization at a given time;

- Memory timing constraints must not be violated.
- The execution of the previous command should not be affected (that is, truncated).

This approach allows for hiding a part of the latency for the first data word or even the complete hiding of the latency in case that the burst length exceeds the maximal command sequence length.

18.4.1.3 Latency Hiding

Enhanced SDRAM Controller optimization is based on command anticipation (MIF2), that is, the next access control phase (memory address and command) is driven during the previous access data phase (data flow from/to the memory), thus an overlap between accesses is created and latency is partially or fully hidden. Additional optimization (not implemented in ESDRAMC) can be achieved by control phase interleaving, that is, during idle cycles in the control phase (caused by memory timing constraints like tRP, tRCD) two accesses control phases can be interleaved so the latency is fully hidden. Enhanced SDRAM Controller optimization can occur only in multi master system with high SDRAM utilization (high density accesses).

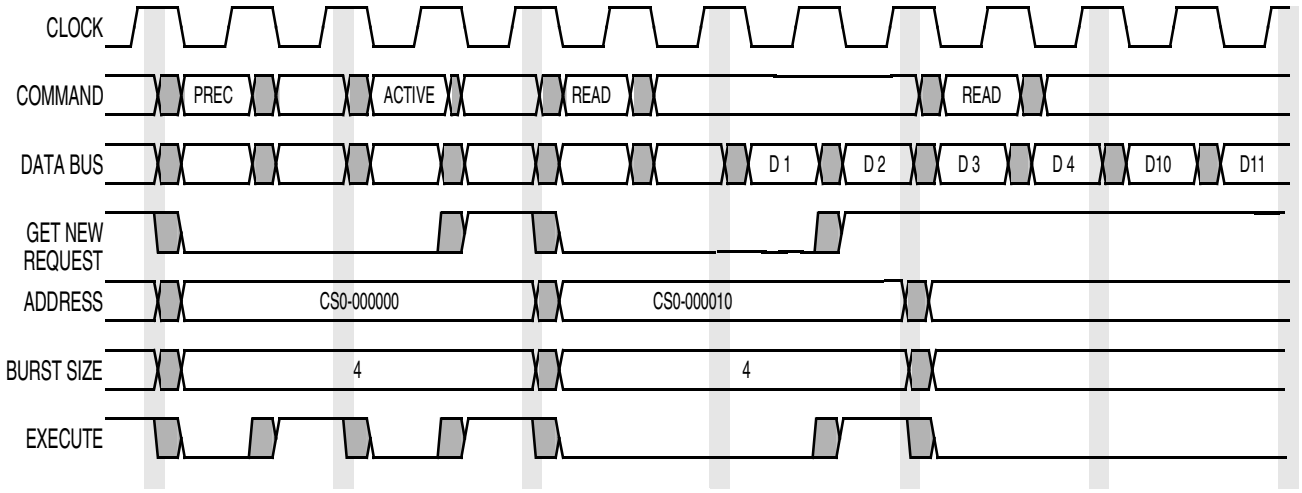


Figure 18-31. SDR Simple Read after Read Latency Hiding Timing Diagram

As mentioned earlier, ESDCTL optimize command sequence toward the memories in order to hide latency, so data bus will be used as much as possible under the access required and SDRAM/system initial

configuration. In Figure 18-31 and Figure 18-32, latency hiding timing diagram example is shown when miss burst read from bank A is followed by a hit burst read request from same chip select. The second read command is issued during the first access data phase, so first data of the second access (D10) is valid immediately after the last data of the first access (D4 for SDR and D8 for LPDDR). CAS latency (tCAS) is set to 2 cycles. The second access latency is fully hidden during the first access data phase. Mobile/Low Power DDR needs another cycle to prevent contention on the DQS signals when two different LPDDRs (two different chip selects) drive those data strobe signals (contention between the last two data cycles of the first transfer and the preamble of the second transfer).

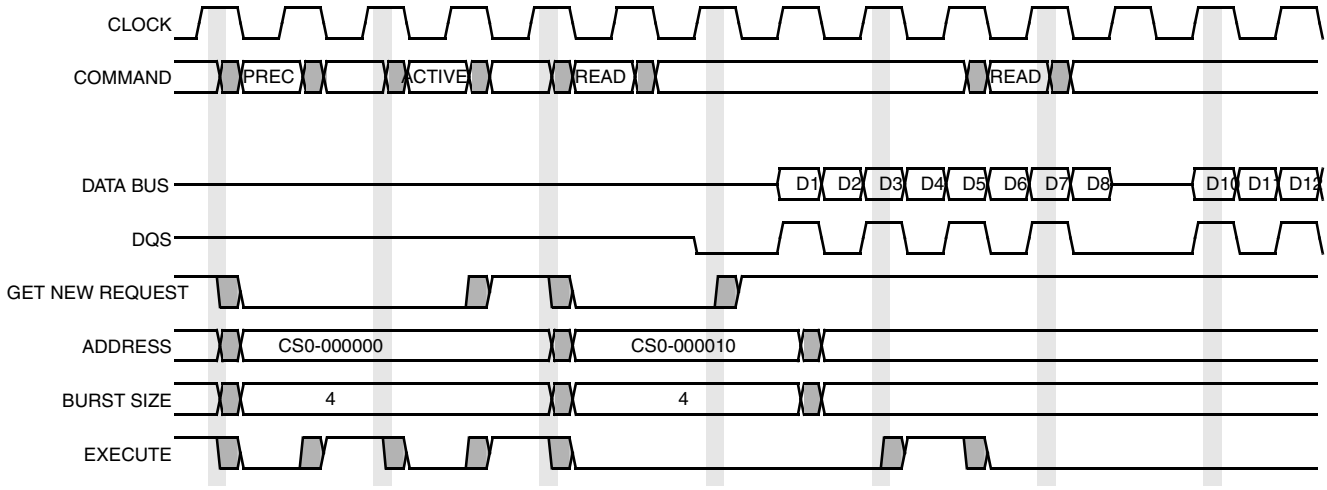


Figure 18-32. Mobile DDR Simple Read after Read Latency Hiding Timing Diagram

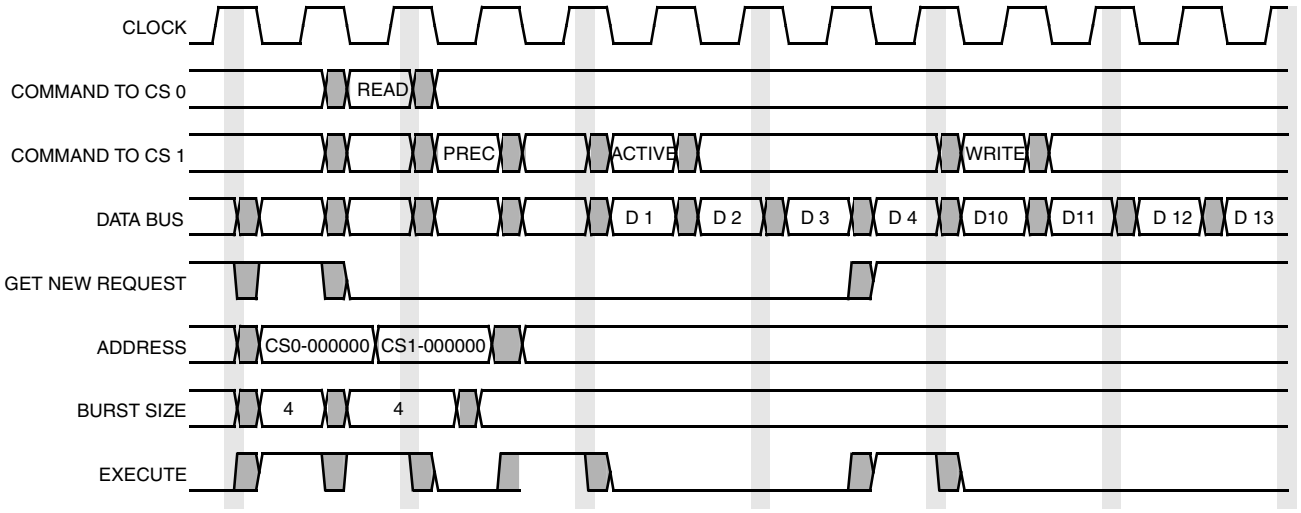


Figure 18-33. SDR Miss Write to CSD1 after Read from CSD0

At Figure 18-33 and Figure 18-34, a burst read to CSD0 is followed by a miss burst write access to CSD1. Due to command anticipation the control phase of the WRITE command overlap the data phase of the READ command, so again the second access latency is fully hidden during the first access data phase. The first WRITE command to CSD1 (D10) is issued immediately after the last data (D4 for SDR and D8 for LPDDR) from CSD0, although the access to CSD1 is a miss access (CSD0 CAS latency is set to 3 cycles,

and both CSD burst length is set to 4 words. LPDDR burst length is 8 but is considered from the system point of view as a burst of 4 with double bus width).

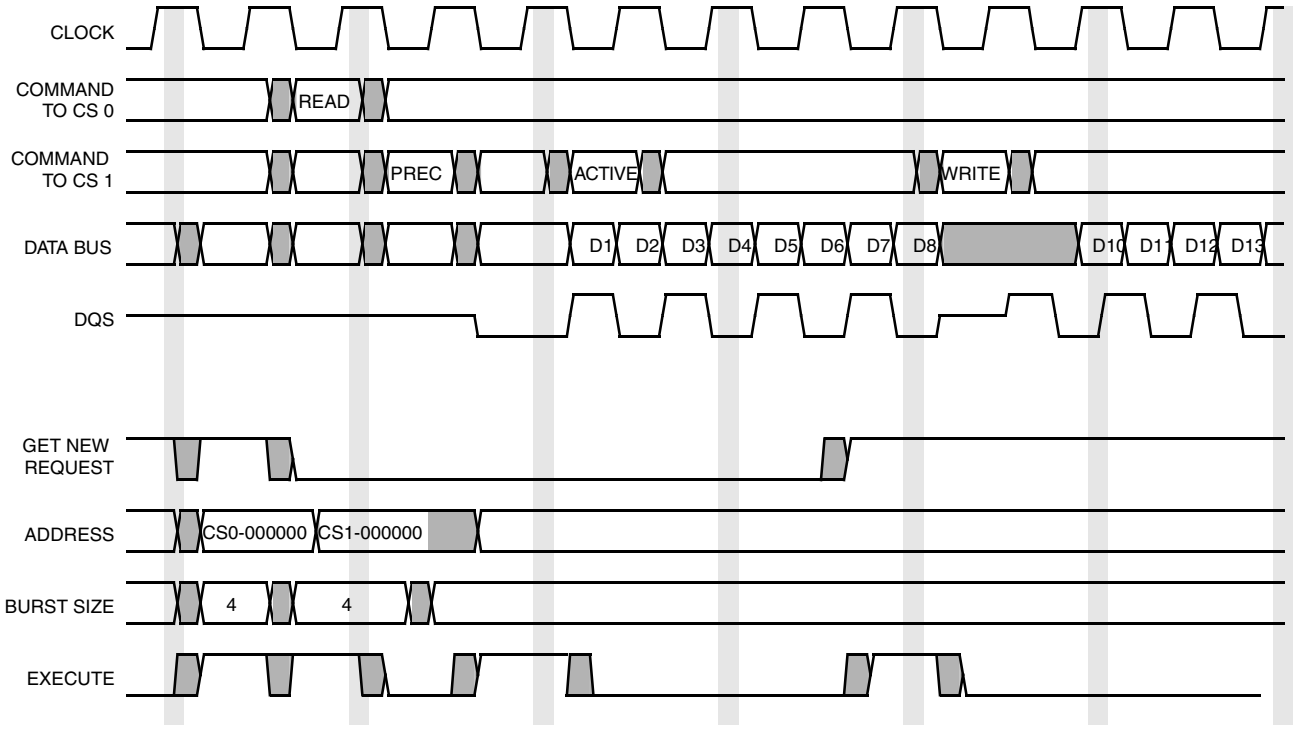


Figure 18-34. Mobile DDR Miss Write to CSD1 after Read from CSD0

18.4.2 Address Multiplexing

18.4.2.1 Multiplexed Address Bus

Table 18-22 illustrates several examples on how a CPU address is scrambled by Enhanced SDRAM Controller to implement a contiguous address space.

Table 18-22. CPU to SDRAM/LPDDR Translation

CPU ADDRESS	16-Bit SDRAM[1]	32-Bit SDRAM[1]	CPU ADDRESS	16-Bit SDRAM[1]	32-Bit SDRAM[1]
A25	—	BA1	A17	R6	R5
A24	BA1	BA0	A16	R5	R4
A23	BA0	R11	A15	R4	R3
A22	R11	R10	A14	R3	R2
A21	R10	R9	A13	R2	R1
A20	R9	R8	A12	R1	R0
A19	R8	R7	A11	R0	C9
A18	R7	R6	A10	C9	C8
A9	C8	C7	A4	C3	C2

Table 18-22. CPU to SDRAM/LPDDR Translation (continued)

CPU ADDRESS	16-Bit SDRAM[1]	32-Bit SDRAM[1]	CPU ADDRESS	16-Bit SDRAM[1]	32-Bit SDRAM[1]
A8	C7	C6	A3	C2	C1
A7	C6	C5	A2	C1	C0
A6	C5	C4	A1[2]	C0	—
A5	C4	C3	A0[3]	—	—

1 For SDRAM example a memory configuration with 10 columns and 12 rows is illustrated. The address translation is based on the following concept, COLUMN-ROW-BANK.

2 CPU A1 defines how the data masks are driven, that is, it is used as the byte enable for non word accesses. This bit has a regular/normal use only in case of 16-bit memory, while CPU A0 defines the 2 bytes (low and high) in the 16-bit word.

3 CPU A0 defines how the data masks are driven, that is, it is used as the byte enable for non word accesses to 32-bit memory device. Both CPU A0 and A1 defines the 4 bytes in the 32-bit word.

4. Legend: C=COLUMN, S=SEGMENT, R=ROW, BA=BANK

Enhanced SDRAM Controller multiplexed address bus is aligned to column addresses so that address line A1 and A2 always appears on pin MA0 for 16-bit and 32-bit memory devices respectively. With this alignment, “folding point” in multiplexor is driven solely by the number of column address bits. Column bus widths of 8 to 11 bits are supported. Table 18-23 and Table 18-24 summarizes controller multiplex options supported for 16 and 32-bit devices respectively. Column addresses through A10 are driven regardless of multiplexor configuration, although some of the lines will be unused for smaller page sizes.

Table 18-23. Address Multiplexing by Column/Row Width for 16-Bit Devices

Device Pins	ESDCTL Pins	16-bit SDR and LPDDR SDRAM Memory Device									
		64 Mb		128 Mb		256 Mb		512 Mb		1 Gb	
		8 Col 12 Row		9 Col 12 Row		9 Col 13 Row		10 Col 13 Row		10 Col 14 Row	
		Col	Row	Col	Row	Col	Row	Col	Row	Col	Row
BA1	BA1	A22	A22	A23	A23	A24	A24	A25	A25	A26	A26
BA0	BA0	A21	A21	A22	A22	A23	A23	A24	A24	A25	A25
MA13	MA13	—	—	—	—	—	—	—	—	—	A24
MA12	MA12	—	—	—	—	—	A22	—	A23	—	A23
MA11	MA11	—	A20	—	A21	—	A21	—	A22	—	A22
MA10	MA10	—	A19	—	A20	—	A20	—	A21	—	A21
MA9	MA9	—	A18	—	A19	—	A19	A10	A20	A10	A20
MA8	MA8	—	A17	A9	A18	A9	A18	A9	A19	A9	A19
MA7	MA7	A8	A16	A8	A17	A8	A17	A8	A18	A8	A18
MA6	MA6	A7	A15	A7	A16	A7	A16	A7	A17	A7	A17
MA5	MA5	A6	A14	A6	A15	A6	A15	A6	A16	A6	A16

Table 18-23. Address Multiplexing by Column/Row Width for 16-Bit Devices (continued)

Device Pins	ESDCTL Pins	16-bit SDR and LPDDR SDRAM Memory Device									
		64 Mb		128 Mb		256 Mb		512 Mb		1 Gb	
		8 Col 12 Row		9 Col 12 Row		9 Col 13 Row		10 Col 13 Row		10 Col 14 Row	
		Col	Row	Col	Row	Col	Row	Col	Row	Col	Row
MA4	MA4	A5	A13	A5	A14	A5	A14	A5	A15	A5	A15
MA3	MA3	A4	A12	A4	A13	A4	A13	A4	A14	A4	A14
MA2	MA2	A3	A11	A3	A12	A3	A12	A3	A13	A3	A13
MA1	MA1	A2	A10	A2	A11	A2	A11	A2	A12	A2	A12
MA0	MA0	A1	A9	A1	A10	A1	A10	A1	A11	A1	A11

Table 18-24. Address Multiplexing by Column/Row Width for 32-bit Devices

Device Pins	ESDCTL Pins	32-Bit SDR and LPDDR SDRAM Memory Device											
		64 Mb		128 Mb		256 Mb		512 Mb		1 Gbyte		2 Gbyte	
		8 Col 11 Row		8 Col 12 Row		8 Col 13 Row		9 Col 13 Row		9 Col 14 Row		10 Col 14 Row	
		Col	Row	Col	Row	Col	Row	Col	Row	Col	Row	Col	Row
BA1	BA1	A22	A22	A23	A23	A24	A24	A25	A25	A26	A26	A27	A27
BA0	BA0	A21	A21	A22	A22	A23	A23	A24	A24	A25	A25	A26	A26
MA13	MA13	—	—	—	—	—	—	—	—	—	A24	—	A25
MA12	MA12	—	—	—	—	—	A22	—	A23	—	A23	—	A24
MA11	MA11	—	—	—	A21	—	A21	—	A22	—	A22	—	A23
MA10	MA10	—	A20	—	A20	—	A20	—	A21	—	A21	—	A22
MA9	MA9	—	A19	—	A19	—	A19	—	A20	—	A20	A11	A21
MA8	MA8	—	A18	—	A18	—	A18	A10	A19	A10	A19	A10	A20
MA7	MA7	A9	A17	A9	A17	A9	A17	A9	A18	A9	A18	A9	A19
MA6	MA6	A8	A16	A8	A16	A8	A16	A8	A17	A8	A17	A8	A18
MA5	MA5	A7	A15	A7	A15	A7	A15	A7	A16	A7	A16	A7	A17
MA4	MA4	A6	A14	A6	A14	A6	A14	A6	A15	A6	A15	A6	A16
MA3	MA3	A5	A13	A5	A13	A5	A13	A5	A14	A5	A14	A5	A15
MA2	MA2	A4	A12	A4	A12	A4	A12	A4	A13	A4	A13	A4	A14
MA1	MA1	A3	A11	A3	A11	A3	A11	A3	A12	A3	A12	A3	A13
MA0	MA0	A2	A10	A2	A10	A2	A10	A2	A11	A2	A11	A2	A12

18.4.2.2 Bank Addresses

Bank address connections are summarized in Table 18-25. Bank addressing utilizes the most-significant addresses to specify active bank, actual bits being dependent on the density of memory system. Page size and density for a number of potential configurations are documented in Table 18-25. For undocumented configurations, Equation 18-1 and Equation 18-2 can be used to calculate page size and density.

$$\text{Page Size (Bytes)} = 2^{\#\text{ Column Address Bits}} \times (\text{Memory Width in bits} / 8) \qquad \textit{Eqn. 18-1}$$

$$\text{Density (bytes)} = 2^{(\#\text{Column Address Bits} + \#\text{ Row Address Bits})} \times (\text{Memory Width in bits} / 2) \qquad \textit{Eqn. 18-2}$$

Table 18-25. Bank Address Bit Assignment

Density	Page Size (Bytes)	BA1	BA0
8MB	X	A22	A21
16MB		A23	A22
32MB		A24	A23
64MB		A25	A24
128MB		A26	A25
256MB		A27	A26

18.4.3 Multiplexed Address Bus—During "Special" Mode (SMODE 1 or 3)

During "special" mode, that is, precharge mode (SMODE=1) or load mode registers (SMODE=3) there is no address shifting, means CPU address A0 is mapped on MA0 at all memory width. For example, in order to drive MA10 bit (for precharge all command), CPU A10 bit should be set (for both 16 or 32 bit external devices). Same logic is valid for load mode register command, as seen on initialization routine example in Section 18.5.4.1, "SDRAM Initialization."

NOTE

BYTE accesses are required (during precharge/load mode register modes) since address can be non-aligned, depending on the load mode register data.

18.4.4 Refresh

Enhanced SDRAM Controller hardware satisfies all SDRAM refresh requirements after an initial configuration by user software. 0, 1, 2, 4, 8 or 16 refresh cycles are scheduled at 31.25 us (nominal 32 kHz clock) intervals, providing 0, 2048, 4096, 8192, 16384, or 32768 refresh cycles every 64 ms. Refresh rate is programmed through REFR field in ESDCTL0 and ESDCTL1 registers. Each array can have a different rate, allowing a mix of SDRAM/LPDDR devices, or different SDRAMs density. Refresh is disabled by hardware reset. A refresh request is made pending at each rising edge on the 32 kHz clock. In response to this request, the hardware gains control of the SDRAM as soon as any in-process bus cycle completes. Once it has gained control of the memory, commands are issued to precharge all banks. Following a row precharge delay (t_{RP}), an auto-refresh command is issued. At t_{RC} intervals, additional auto-refresh cycles are issued until the specified number of cycles have been run.

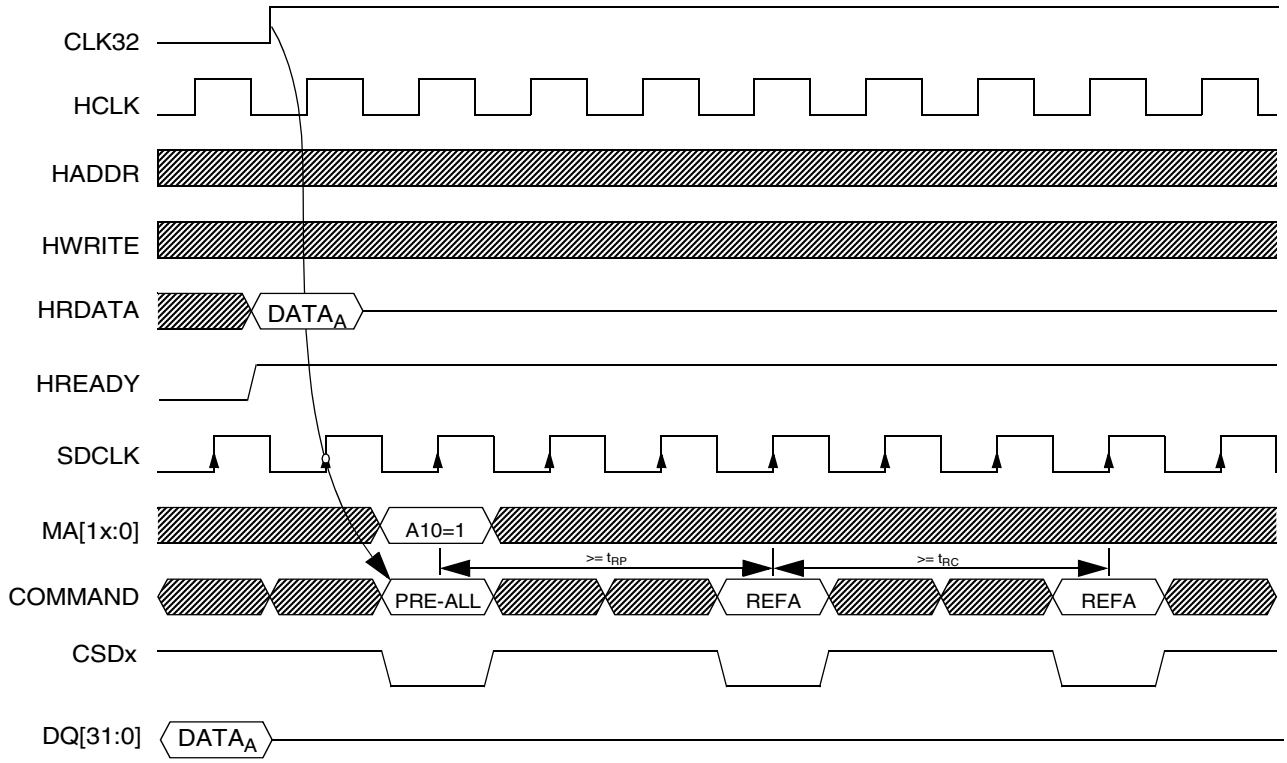


Figure 18-35. Hardware Refresh Timing Diagram

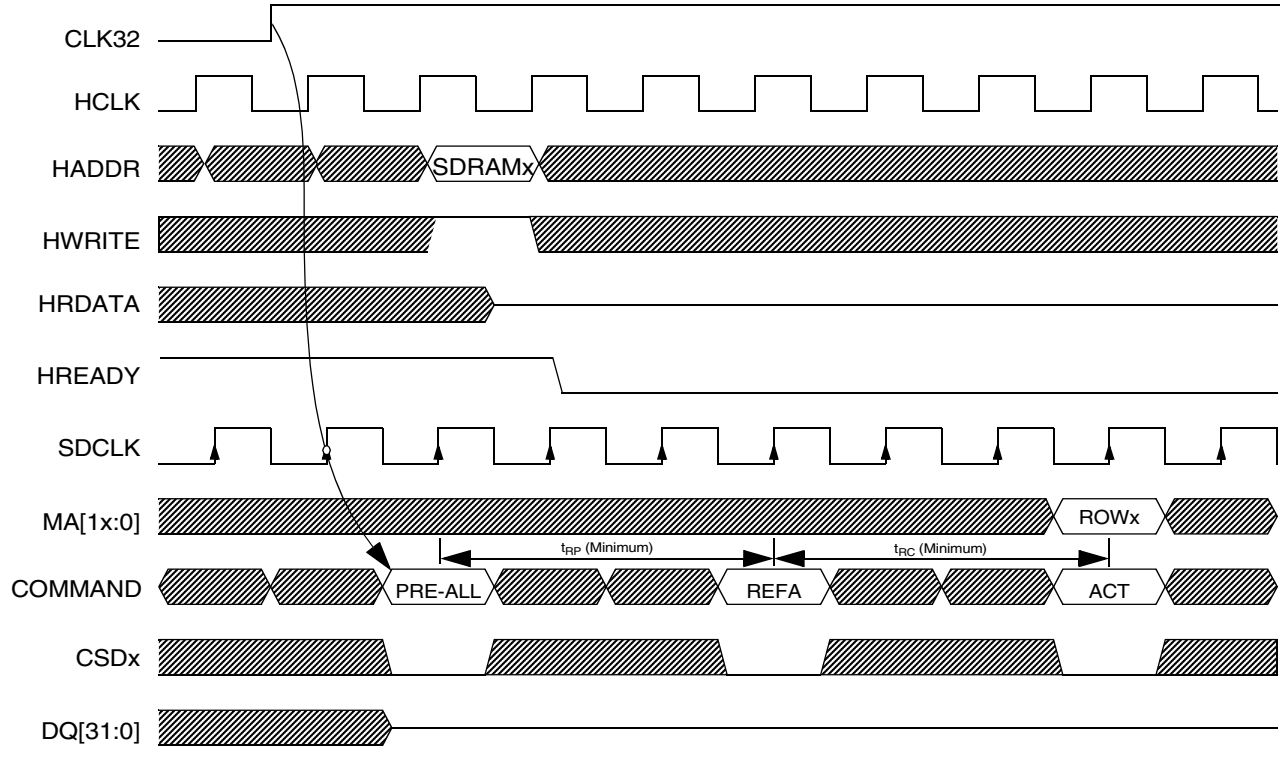


Figure 18-36. Hardware Refresh with Pending Bus Cycle Timing Diagram

Figure 18-35 illustrates 2 refresh sequence. Burst transfers in progress when the refresh request arrives are allowed to complete prior to the refresh operation. SDRAM bus accesses queued after the refresh request are held off until refresh completes. In Figure 18-36, an access is queued just as the refresh begins. This cycle is delayed until the precharge and single refresh (REFR=01) cycles are run. Bus cycles targeted to other memory or peripheral devices are allowed to progress normally while the refresh is in progress. None of the pins shared between the SDRAM and other devices are required for the refresh operation.

NOTE

Since REFRESH commands (requires all banks to be in IDLE state, achieved by PRECHARGE ALL) are issued automatically by Enhanced SDRAM Controller at each 32 kHz clock, address bits A10 (for both 16 and 32-bit devices) cannot be shared with other peripherals address bus in the system.

18.4.5 Low Power Operating Modes

This section describes low power operating modes of Enhanced SDRAM Controller as a function of various memory devices. Table 18-26 lists and summarizes low power modes supported by Enhanced SDRAM Controller.

Table 18-26. ESDRAMC Low Power Operating Modes

Memory Device	System Operating Mode	Memory Device Low Power Operating Mode	WakeUp Penalty
SDRAM	RUN	POWER DOWN MODE	1 clock cycle
	RUN	PRECHARGE BANK(s)	1 clock cycle
	RUN	MANUAL SELF REFRESH MODE[1] (SMODEx=100)	2 Refresh Period
	STOP	SELF REFRESH MODE	2 Refresh Period
LPDDR	RUN	POWER DOWN MODE	tXP
	RUN	PRECHARGE BANK(s)	1 clock cycle
	RUN	MANUAL SELF REFRESH MODE (SMODEx=100)	tXS + 2 Refresh Period
	STOP	SELF REFRESH MODE	tXS + 2 Refresh Period

1 SDCLK stops, only if both chip selects are in manual self refresh mode.

18.4.5.1 Self Refresh Mode for SDRAM/LPDDR Devices

This operating mode (see Figure 18-37 and Figure 18-38) allows the software/user to control a Self refresh mode entry of the external SDRAM/LPDDR device if refresh has been enabled, during system RUN mode. When this mode is selected (SMODE=100 in the respective CSD control register) and refresh is enabled the Enhanced SDRAM Controller will complete any active access and a self refresh command to the external device will be issued. No access is allowed to the respective CSD during manual self refresh mode. If refresh has not been enabled, the Enhanced SDRAM Controller places the memory in a low

power consumption mode known as power down. The LPACK signal (low power mode acknowledge) will not be asserted if only one CSD enters manual self refresh mode.

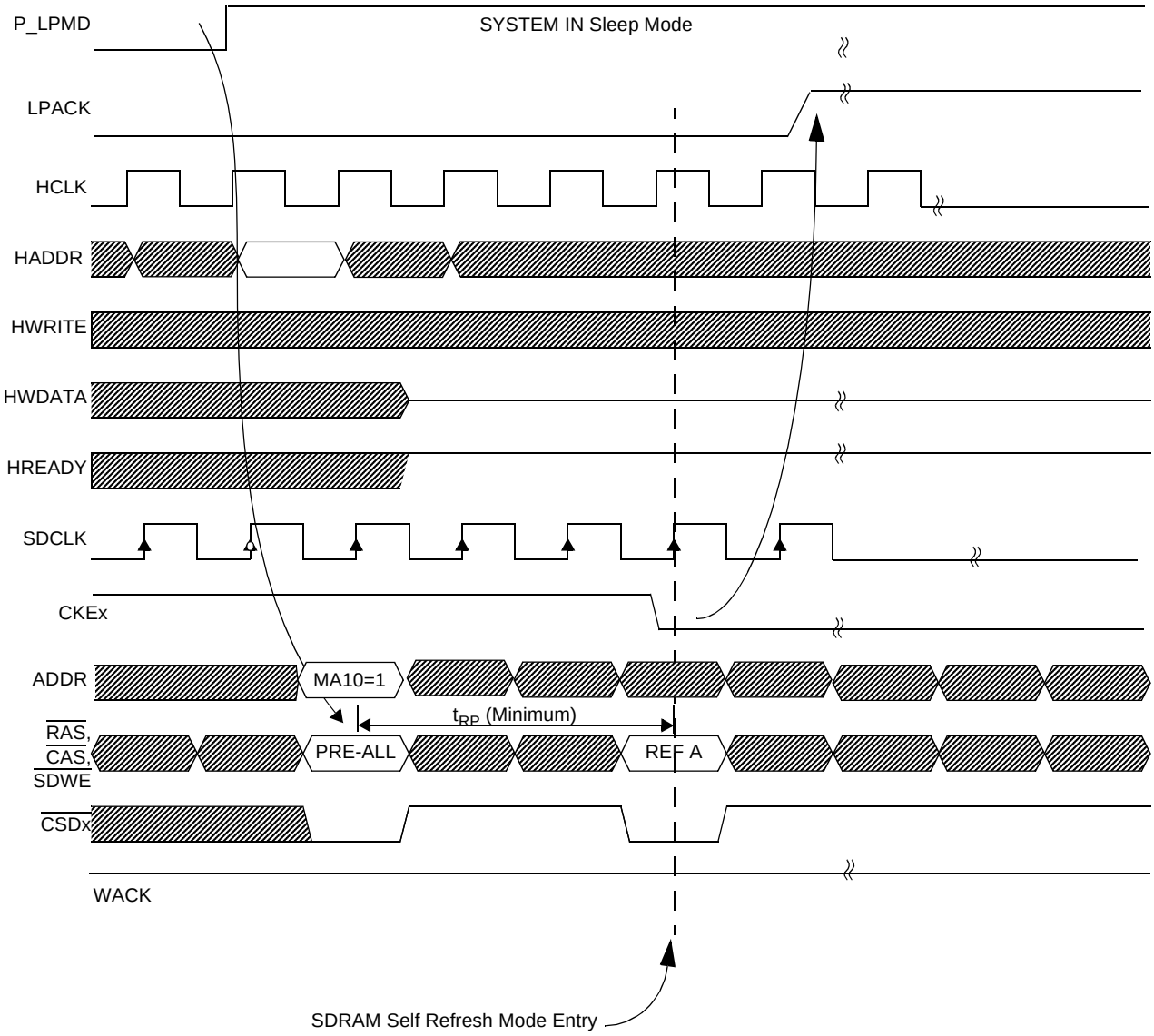


Figure 18-37. SDRAM/LPDDR Enter Self Refresh Mode During System Sleep Mode

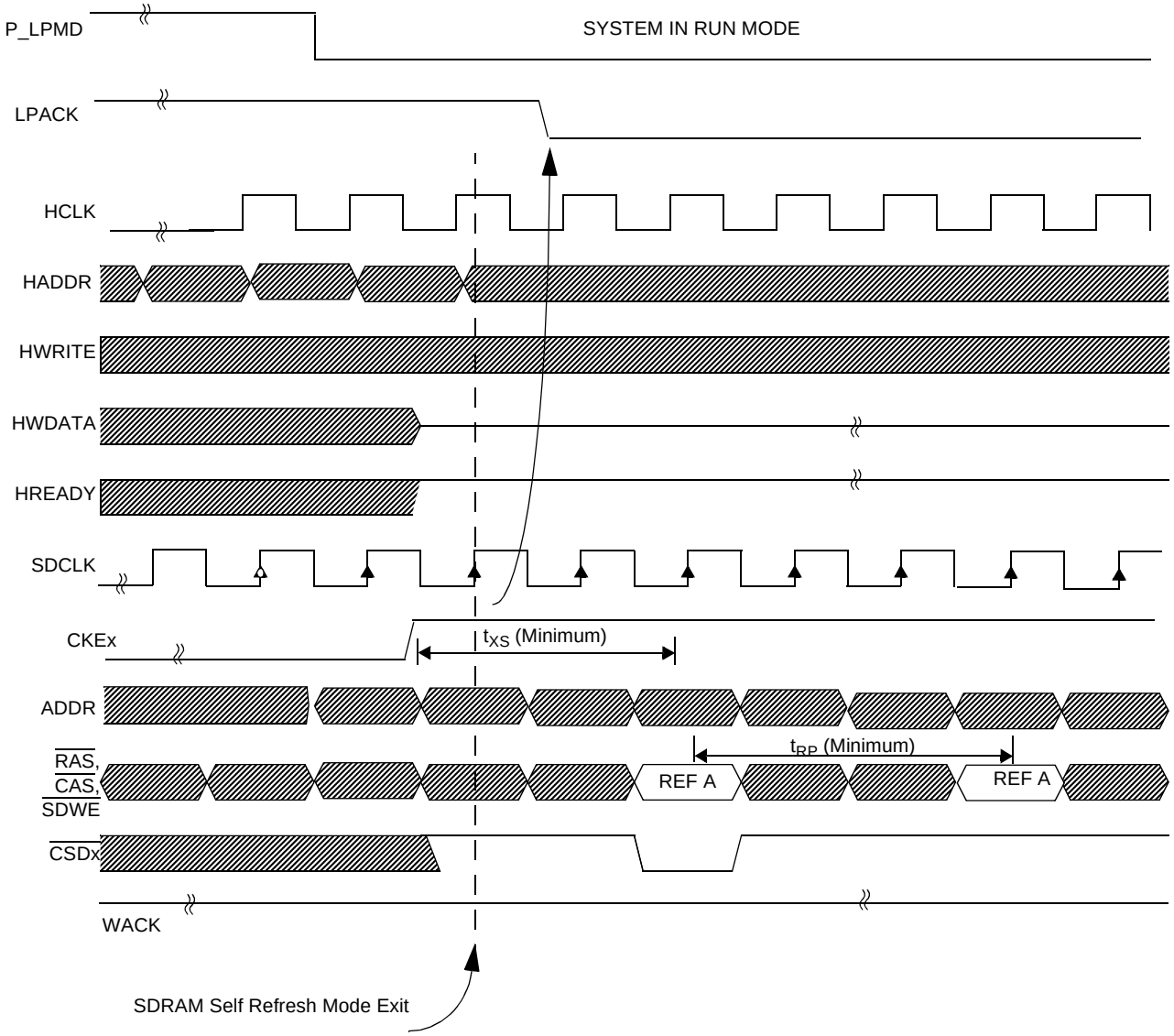


Figure 18-38. SDRAM/LPDDR Exit Self Refresh Mode During System Sleep Mode

18.4.5.2 Manual Self Refresh Mode for SDRAM/LPDDR Devices

This operating mode allows the software/user to control a Self refresh mode entry of the external SDRAM/LPDDR device if refresh has been enabled, during the system RUN mode. When this mode is selected (SMODE=100 in the respective CSD control register) and refresh is enabled the Enhanced SDRAM Controller will complete any active access and a self refresh command to the external device will be issued. No access is allowed to the respective CSD during manual self refresh mode. If refresh has not been enabled, the Enhanced SDRAM Controller places the memory in a low power consumption mode known as power down. The LPACK signal (low power mode acknowledge) will not be asserted if only one CSD enters manual self refresh mode.

NOTE

Manual precharge all should be initiated by user before manual self refresh.

To exit manual self refresh mode, a different operating mode need to be selected by changing SMODE bits in the respective chip select control register. When a different mode is selected, the controller will take the SDRAM device out of self refresh mode and will begin issuing auto refresh cycles (if the refresh has been enabled). illustrates the entry and exit from manual self refresh mode. See Figure 18-39 and Figure 18-40 for timing information.

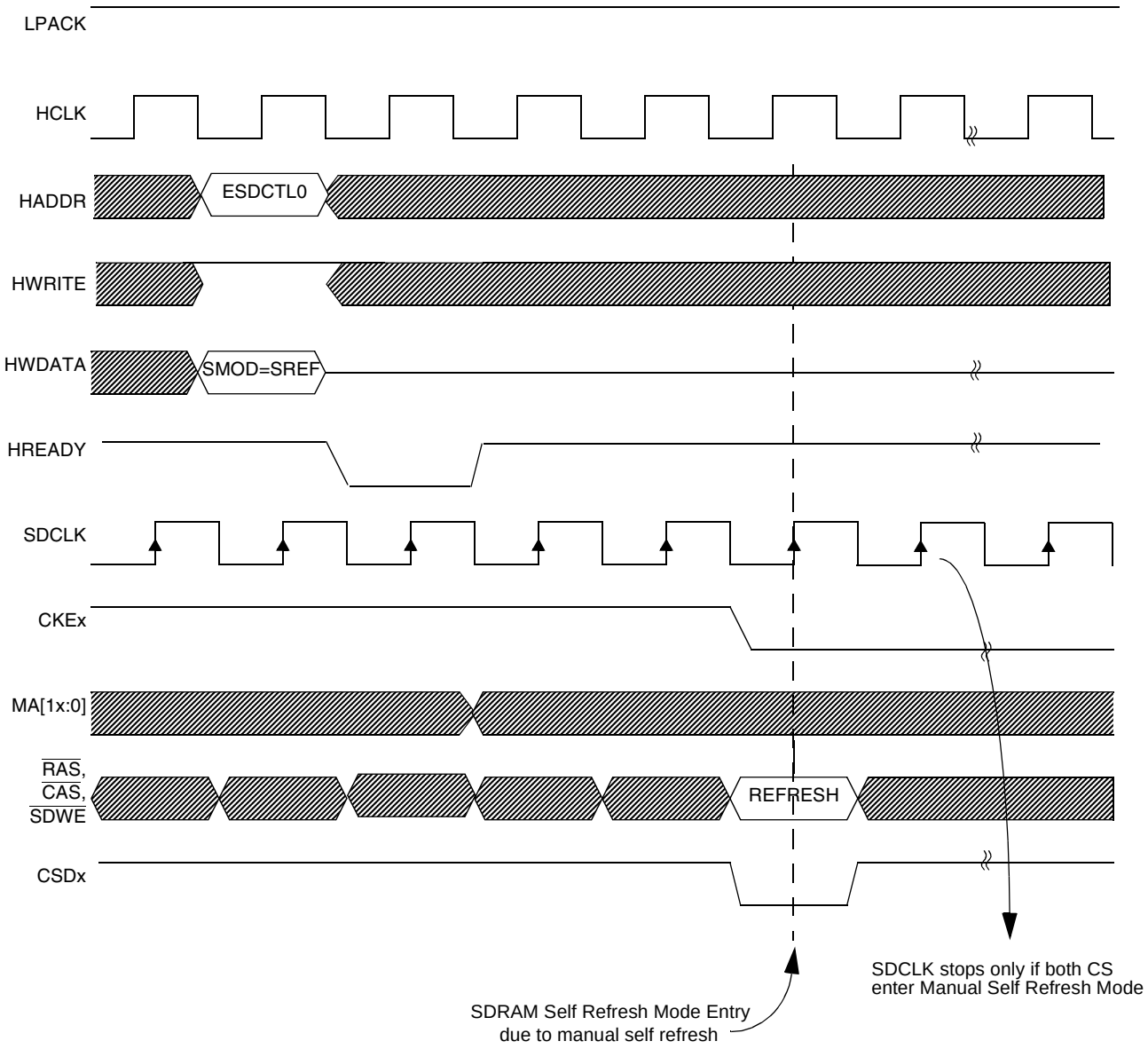


Figure 18-39. Manual Self Refresh Entry Timing Diagram

NOTE

SDCLK stops, only if both chip selects are in manual self refresh. This is in order to allow the usage of one chip select, while the other is in manual self refresh (in case that both chip selects are in use).

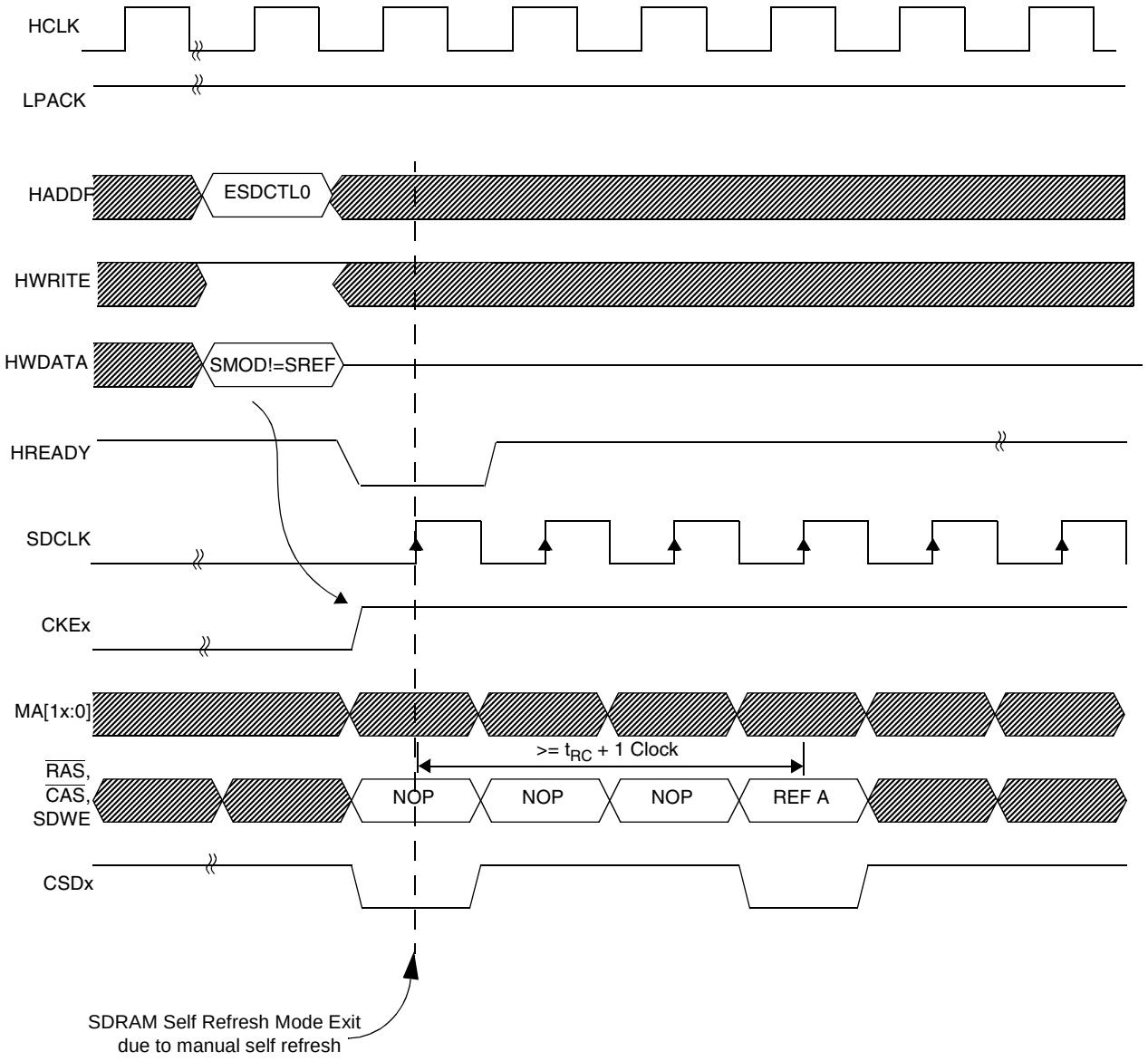


Figure 18-40. Manual Self Refresh Exit Timing Diagram

18.4.5.3 Precharge Power Down Mode

18.4.5.3.1 SDRAM Precharge Power Down Mode

All low power operating mode described in the above paragraphs will be activated only if the system enters low power operating mode, for example, Sleep Mode. Enhanced SDRAM Controller has the capability to reduce power consumption if SDRAM/LPDDR memory utilization is low, by setting SDRAM device in Power Down Mode. This mode is activated through the PWDT bits in ESDCTL0 and/or ESDCTL1 registers. During this operating mode, ESDRAMC automatically issues the REFRESH commands toward the SDRAM/LPDDR memories at the rate defined by SREFR bits in ESDCTL0 and/or ESDCTL1 registers. Programming PWDT[1:0] = 01 causes Enhanced SDRAM Controller to place the memories in power down mode at anytime the controller detects that no banks are active. This mode is useful in

applications where a memory array is accessed infrequently and chances of another access to same page are minimal. Reading or writing to memory activates a page within the addressed bank. Reset, software generated precharge, and hardware initiated refresh are three ways to close an active bank. The periodically occurring refresh will be the normal means that invokes the power down mode. At each refresh interval, all banks will be closed by a precharge-all command, followed by the refresh operation. The controller will then issue the power down command to the memories. A few cycle delay is incurred with the first read or write cycle in order to restart the clocks, but only on the first cycle. After that, the clocks will continue to run until the next refresh operation or until any active banks are manually precharged.

Page misses on read and write cycles cause the addressed bank to be closed (precharged) and a new page opened within the bank. This operation does not cause the clocks to stop, nor does manually precharging only a single bank within the memory. All banks within the memory must be inactive before the power down mode is invoked.

Power Down Mode occurs if CKE is registered LOW coincident with a NOP or COMMAND INHIBIT, when no accesses are in progress. Entering power down will deactivate the input and output buffers (excluding CKE) of the device. The Power Down Mode state is exited by registering a NOP or COMMAND INHIBIT and CKE HIGH at the desired clock edge. For SDR SDRAM, Figure 18-41 and Figure 18-42 illustrates the power down mode entry and exit respectively. For LPDDR SDRAM, Figure 18-43 and Figure 18-44 illustrates the power down mode entry and exit respectively.

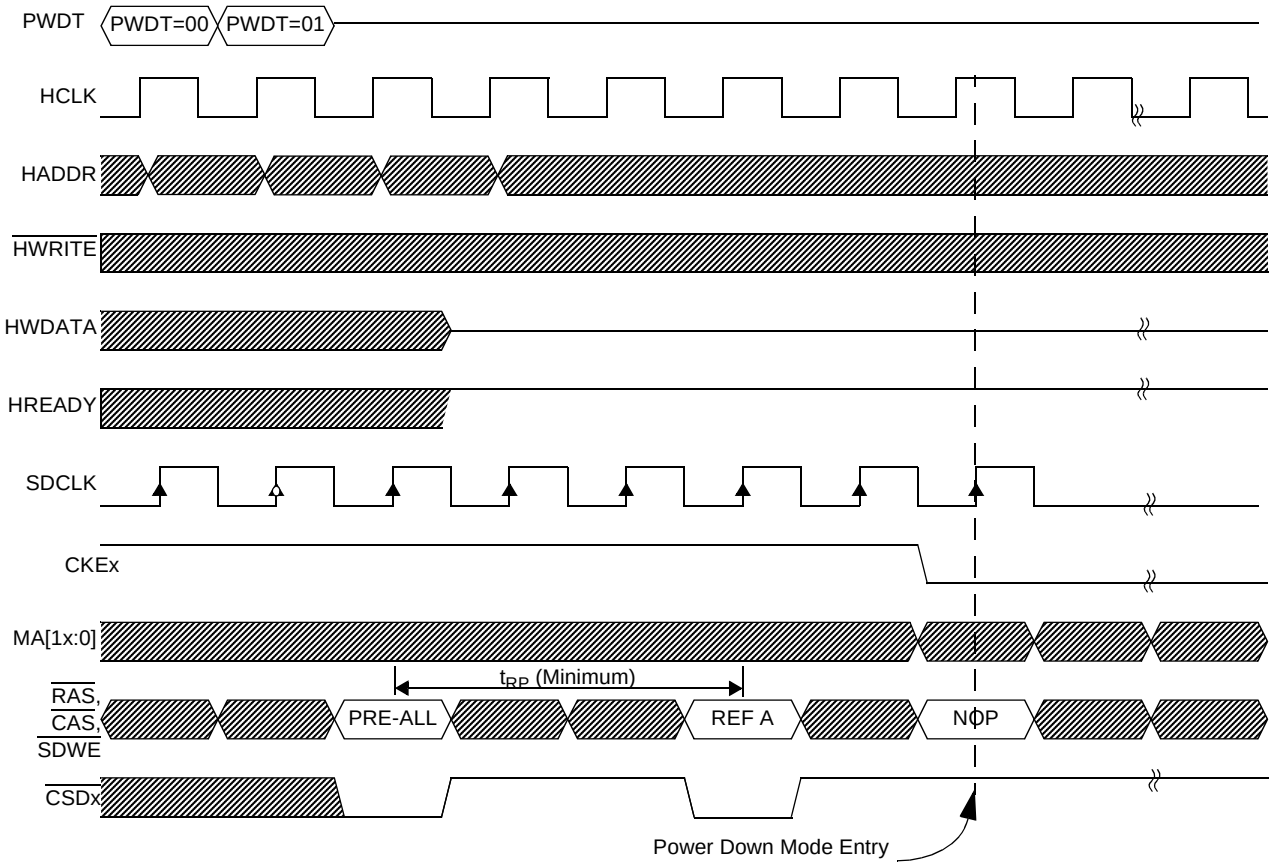


Figure 18-41. SDR SDRAM Precharge Power Down Mode Entry Timing Diagram

NOTE

Since ESDRAMC doesn't issue AUTO PRECHARGE commands toward the SDRAM, software will have to issue a PRECHARGE ALL command in order to enter Precharge Low Power Down Mode, to wait for PRECHARGE timer (PRCT) to close/precharge all active banks, or to wait for the next REFRESH cycle in order to enter this low power mode. (During the REFRESH cycle, the ESDRAMC automatically issue the PRECHARGE ALL command).

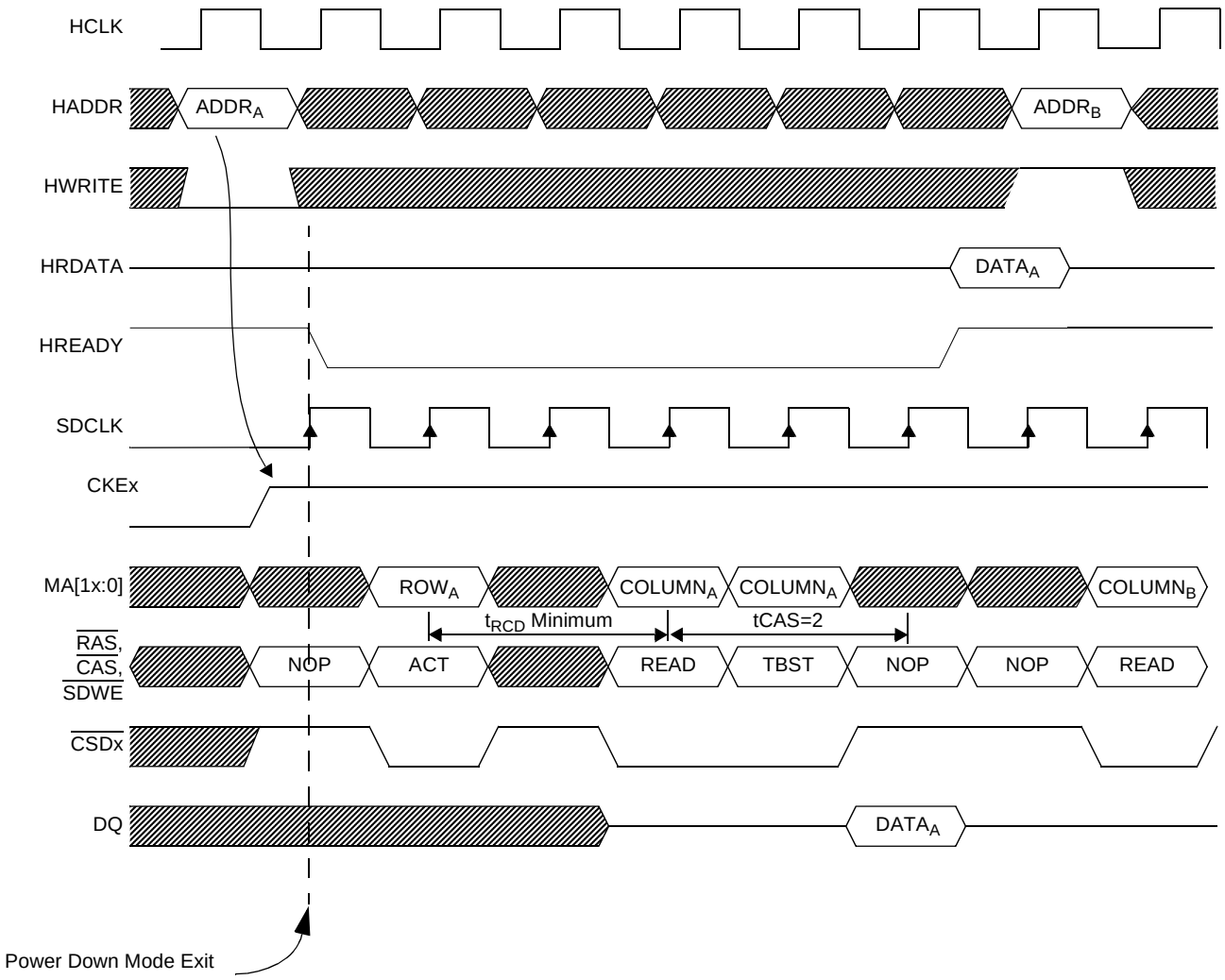


Figure 18-42. SDR SDRAM Precharge Power Down Mode Exit Timing Diagram

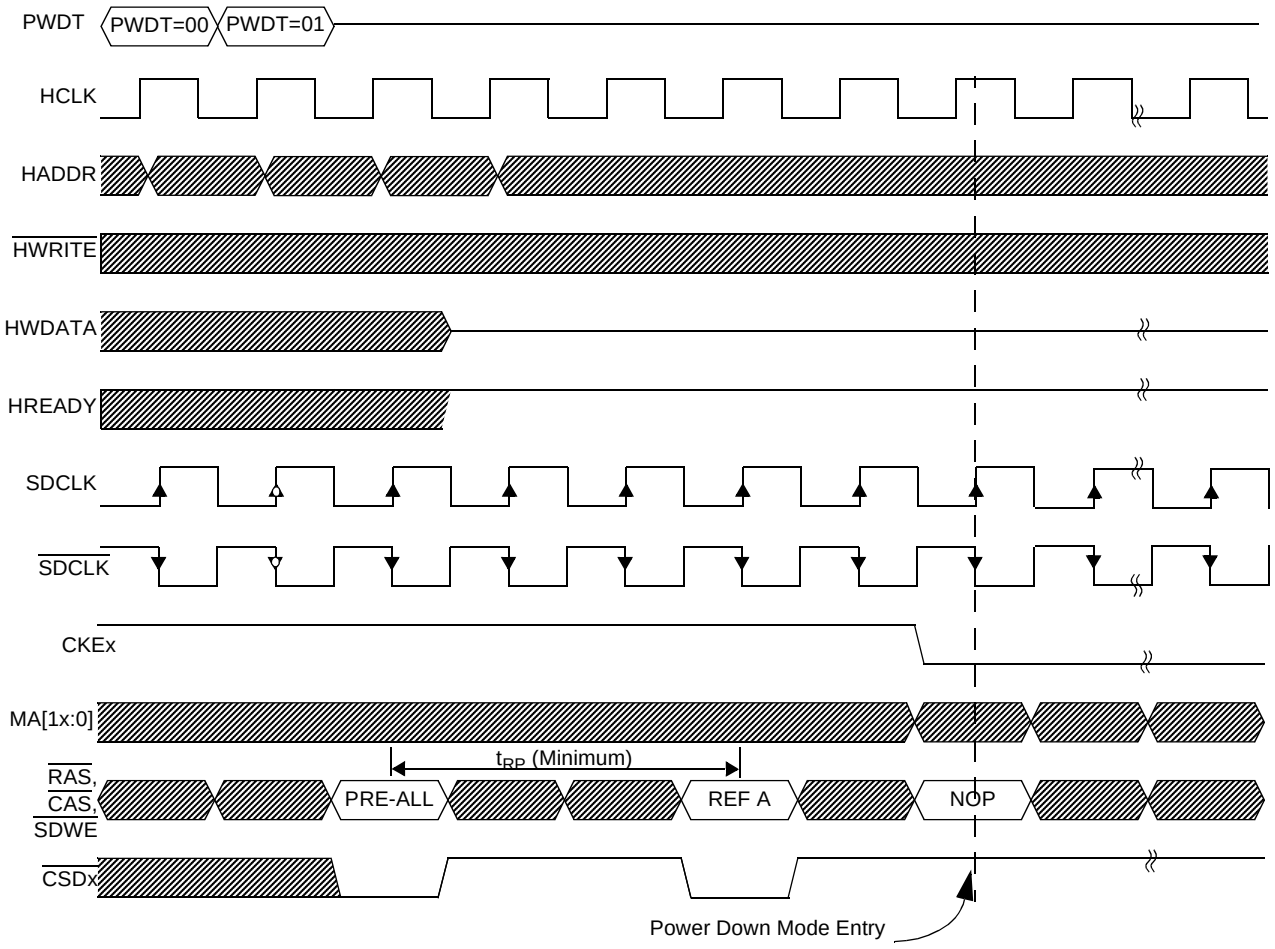


Figure 18-43. Mobile DDR SDRAM Precharge Power Down Mode Entry Timing Diagram

Power Down Mode for several Mobile/Low Power DDRs require the clock CK (and $\overline{CK}$) to continue running. (The PWR CK EN (Power Down Clock Enable for Mobile/Low Power DDR SDRAM) should be set to “1” in order to enable this option)

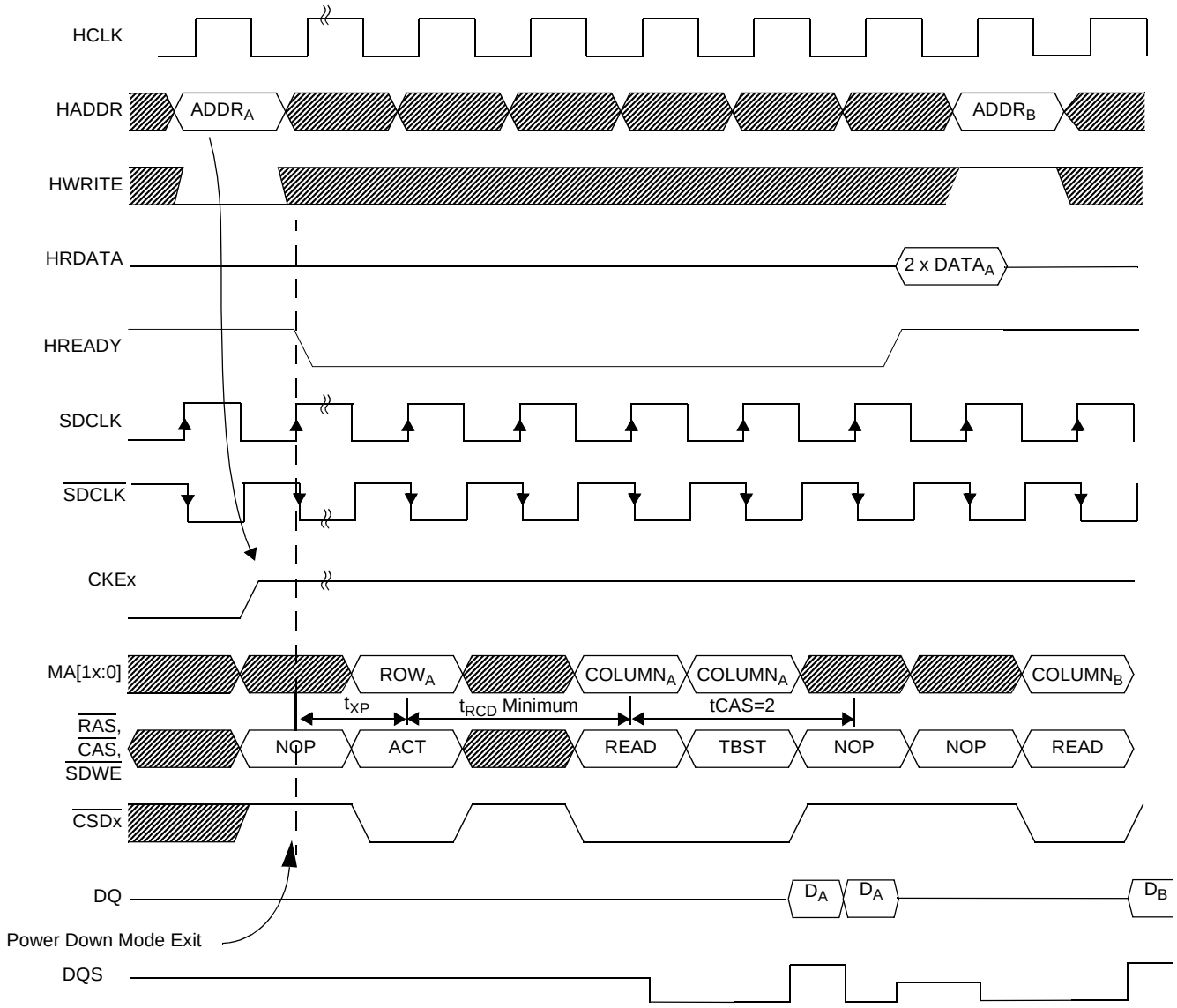


Figure 18-44. Mobile DDR SDRAM Precharge Power Down Mode Exit Timing Diagram

18.4.5.4 Active Power Down Mode

18.4.5.4.1 SDRAM/LPDDR Active Power Down Mode

The second clock suspend mode is selected whenever PWDT[1:0] = 1x. In this mode the SDCLK is stopped after a selectable delay from the last access to the array. Active banks are not closed prior to disabling the SDRAM/LPDDR clock. Either 64 (PWDT[1:0] = 10) or 128 (PWDT[1:0] = 11) cycle delays are possible. SDRAM/LPDDR clocks are counted from the end of the last read or write access. Subsequent read or write accesses, and self-refresh modes reset the counter. Auto-refresh cycles do not affect the counter; however, if the counter expires during a refresh operation the clock will be disabled immediately following the refresh.

The distinguishing factor between Precharge Power Down Mode and Active Power Down Mode is whether banks remain active while the clock is stopped. Active Power Down allows banks to remain activated, while Precharge Power down does not. Figure 18-45 and Figure 18-46 illustrates SDR and LPDDR SDRAM Active Power Down Mode entry and exit timing diagram.

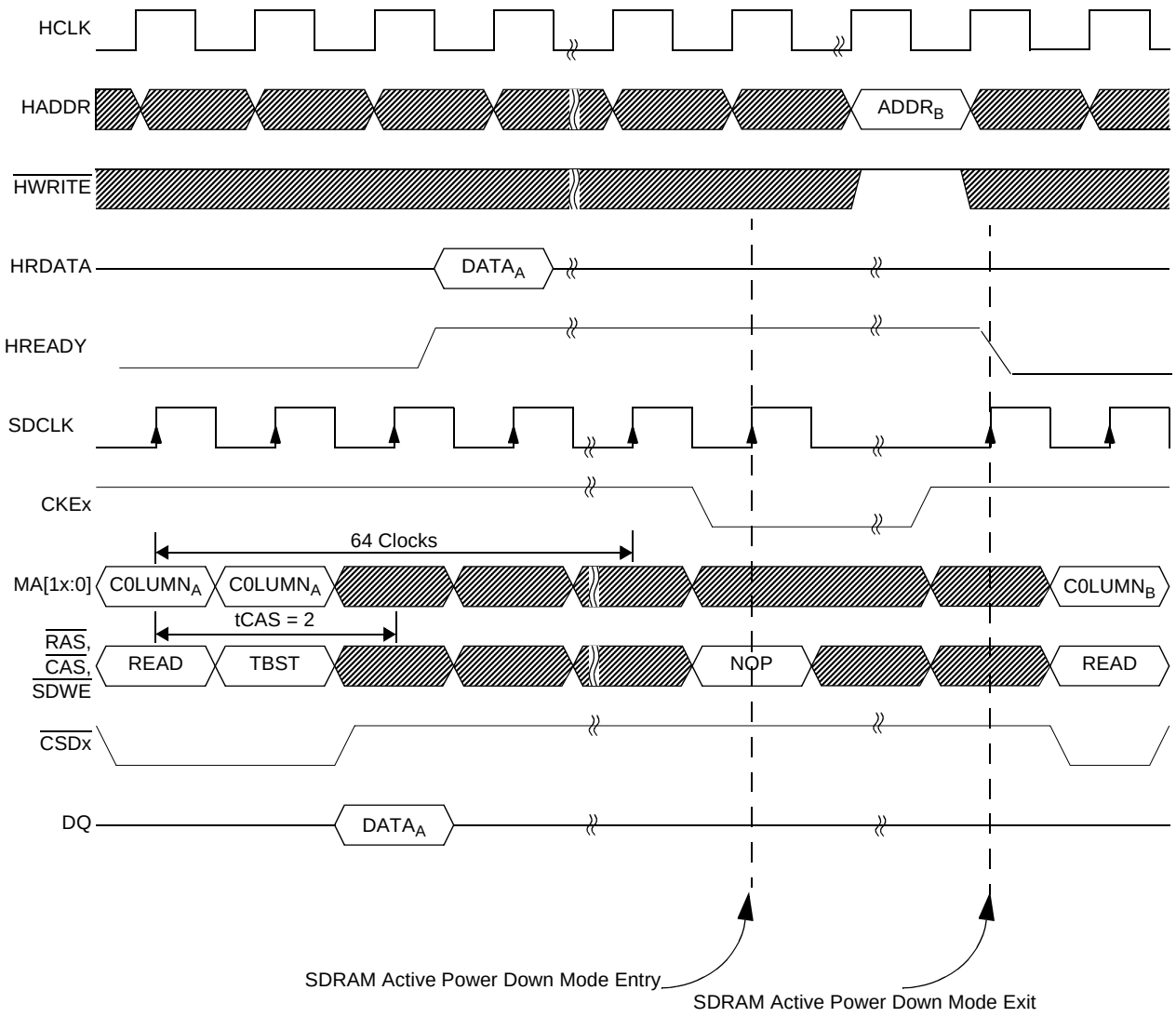


Figure 18-45. SDR SDRAM Active Power Down Mode Timing Diagram

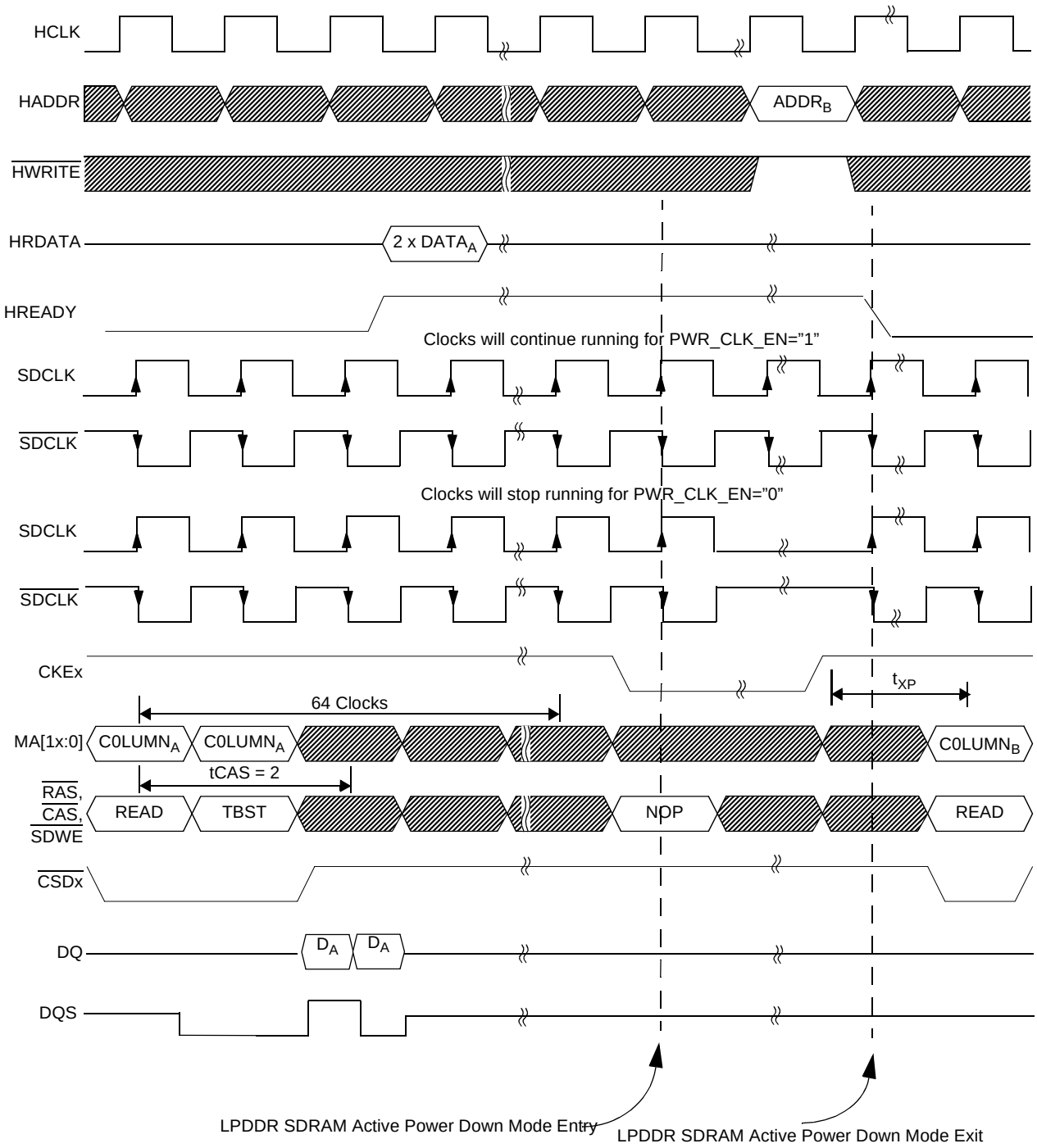


Figure 18-46. Mobile DDR SDRAM Active Power Down Mode Timing Diagram

Power Down Mode for several Mobile DDRs require the clock CK (and $\overline{CK}$) to continue running. (The PWR CK EN (Power Down Clock Enable for Mobile DDR SDRAM) should be set to “1” in order to enable this option)

18.4.5.5 Precharge Bank(s)—Low Power Mode

"Closing" (due to precharge command) the last used/open row in any non active bank within a chip select reduces the power consumption of the external memory device. The power saving is device dependent, and one should consult/examine the external memory device specification for more details on power consumption reduction. The precharge bank is activated if PRCT is enabled. A PRECHARGE command is issued after 2xPRCT clocks (HCLK, up to 133 MHz) of no activity (as shown in Table 18-12) to one of the SDRAM/LPDDR banks. The number of cycles before the PRECHARGE command is issued depends on command bus (WE, RAS, CAS and CSD) availability (means there is no active access to other bank) and the memory timing parameters.

18.4.5.6 LPDDR Frequency Change

The following steps need to be performed prior to a frequency change in a LPDDR based system, in order for the DDRC delay line re-calibration.locking.

1. Issue PRECHARGE_ALL command.
2. Enter the external memories in SELF_REFRESH operating mode.
3. Change system frequency.
4. Reset the delay line, by setting the MDDR_DL_RST bit in the ESDRAMC MISC register.
5. Wait ~4500 HCLK cycles in order for the delay line to lock on the new frequency.
6. Exit SELF_REFRESH mode.

After the above 6 steps are performed the LPDDR is ready for normal operation at the new frequency.

18.4.6 SDRAM (SDR and LPDDR) Command Encoding

Table 18-27 summarizes the command encoding utilized by this controller. These commands represent a subset of the commands defined by the JEDEC standard.

Table 18-27. SDRAM (SDR and LPDDR) Command Encoding

Function	Symbol	CKE n-1	CKE n	CS	RAS	CAS	WE	A11	A10	BA[1:0]	A[13:0]
Deselect	DSEL	H	X	H	X	X	X	X	X	X	X
No Operation	NOP	H	X	L	H	H	H	X	X	X	X
Read	READ	H	X	L	H	L	H	V	L	V	V
Write	WRIT	H	X	L	H	L	L	V	L	V	V
Bank Activate	ACT	H	X	L	L	H	H	V	V	V	V
Burst Terminate[1]	TBST	H	X	L	H	H	L	X	X	V	X
Precharge Select Bank	PRE	H	X	L	L	H	L	V	L	V	X
Precharge All Banks	PALL	H	X	L	L	H	L	X	H	X	X
Auto-Refresh	CBR	H	X	L	L	L	H	X	X	X	X
Self Refresh Entry	SLFRSH	H	L	L	L	L	H	X	X	X	X

Table 18-27. SDRAM (SDR and LPDDR) Command Encoding (continued)

Function	Symbol	CKE n-1	CKE n	CS	RAS	CAS	WE	A11	A10	BA[1:0]	A[13:0]
Self Refresh Exit	SLFRSHX	L	H	H	X	X	X	X	X	X	X
Power-Down Entry	PWRDN	H	L	X	X	X	X	X	X	X	X
Power-Down Exit	PWRDNX	L	H	H	X	X	X	X	X	X	X
Mode Register Set[2]	MRS	H	X	L	L	L	L	L	L	V	V

1 For Mobile DDR, applies only to read bursts (with auto precharge disabled).

2 BA0–BA1 select either the mode register, the extended mode register or the LOW POWER extended mode register.

18.4.6.1 Reset

Assertion of the $\overline{\text{RST}}$ signal initializes the controller into the idle state, and disables the module. While disabled, the controller remains in the idle state with the internal clocks stopped. The reset state of the control register allows for basic read/write operations sufficient to fetch the reset vector and execute the initialization code. A complete initialization of the controller should be performed as part of the start-up code sequence.

Read/write cycles, refresh and low-power mode requests, and Power Down time-outs will all trigger transitions out of the idle state. As shown in the simplified Enhanced SDRAM Controller state diagram pictured in Figure 18-47, state transitions due to a read or write request depend on the operating mode. Other transitions require the corresponding function to be enabled in the ESDCTL registers. Some state transitions have been removed from the figure to minimize complexity and allow an easier understanding of the basic controller operation.

The following subsections document the operation of each of the operating modes.

18.4.7 Normal READ/WRITE Mode

The Normal Read/Write mode (SMODE = 000) is used for general read and write accesses (AHB light compliant) to the SDRAM/LPDDR. Single and read burst accesses are supported for both SDRAM/LPDDR memories (although bursts requests are limited as shown in Table 18-28). For SDRAM/LPDDR memories single and burst write accesses are supported as well.

Read or write requests to Enhanced SDRAM Controller initiate a check to see whether the page is already open. This check consists of comparing request address against last row accessed within the corresponding bank. If the rows are different, a precharge has occurred since the last access, or there has never been an access to the bank, then the access must follow the “off-page” sequence. If the requested and last row match, the shorter “on-page” access is used. An off-page sequence must first activate the requested row, an operation which is analogous to a conventional DRAM RAS cycle. An activate cycle is the first operation depicted in Figure 18-48. During the activate cycle, the appropriate chip select is driven low, the row addresses are placed on the multiplexed address pins, the non-multiplexed addresses are driven to their respective values, write enable is driven high, CAS is driven high, and RAS is driven low. These latter three pins form the SDRAM command word. The data bus is unused during the activate command.

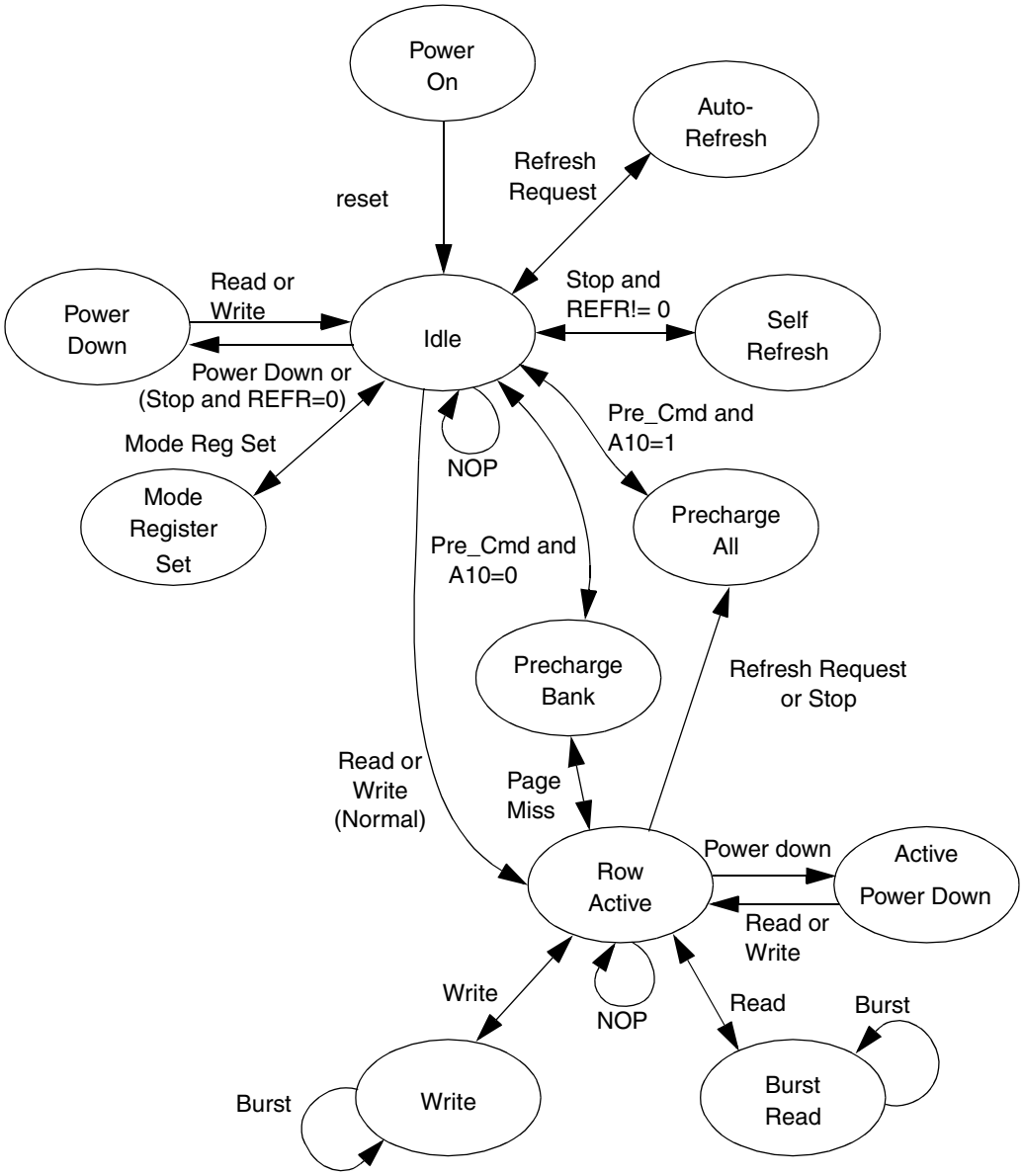


Figure 18-47. Simplified Enhanced SDRAM Controller State Diagram

Once the selected row has been activated, the read operation begins after the row to column delay (tRCD) has been met. This delay is either 2 or 3 clocks, as determined by the tRCD control field. During the read cycle, the chip select is once again asserted, the column addresses are driven onto the multiplexed address bus, the non-multiplexed addresses remain driven to the value presented during the activate cycle, the write enable is driven high (read), RAS is driven high, and CAS is driven low. After the CAS latency has expired, data is transferred across the data bus. CAS latency is programmable via the tCAS control field. As data is being returned across the AHB, transfer acknowledge is asserted back to the CPU indicating that the CPU should latch data. While data is still on the bus, the Enhanced SDRAM Controller must begin monitoring transfer request since the CPU is free to issue the next bus request on the same edge that data is being latched.

Table 18-28. SDRAM/LPDDR Burst Access Support

Internal AHB WORD Access (32bit)		External Memory Device[3]						Description
		16-Bit			32-Bit			
HBURST	TYPE	BL=4	BL=8	BL=FP	BL=4	BL=8	BL=FP[2]	
000	SINGLE[1,2]	No	Yes	Yes	Yes	Yes	Yes	Single transfer
001	INCR	No	Yes	Yes	Yes	Yes	Yes	Incrementing burst
010	WRAP4	No	Yes	Yes	Yes	Yes	Yes	4-beat wrapping burst
011	INCR4	No	Yes	Yes	Yes	Yes	Yes	4-beat incrementing burst
100	WRAP8	No	Yes	Yes	Yes	Yes	Yes	8-beat wrapping burst
101	INCR8	No	Yes	Yes	Yes	Yes	Yes	8-beat incrementing burst
110	WRAP16	No	No	No	No	No	No	16-beat wrapping burst
111	INCR16	No	No	No	No	No	No	16-beat incrementing burst

1 ESDCTL supports only burst of 32-bit (word size). Byte (8 bits) or half words (16 bits) accesses are supported only in case of single transfer (SINGLE). The ESDCTL automatically splits (see example in Table 18-29) the AHB bursts as a function of the external memory device, configured via the ESDCTL configuration registers (size and burst length), in such a way that a continuous flow of data is obtained (for both READ or WRITE bursts).

2 BL = Burst Length field in device mode register; FP = Full Page (wrap at external memory device ROW boundary). For both LPDDR 16 and 32-bit devices only BL=8 is supported.

Data transfers can be either single operand or a burst (WRAP or INCR) of up to a full page. Burst requests are designated as such by the HBURST bus indicating the length of the access, When HBURST equal to 0 a single access is required, otherwise the access is a burst of HBURST words.

SDRAM memories assume that all transfers are burst transfers unless terminated early. Burst transfers can be terminated by a variety of mechanisms: another read or write cycle, a precharge operation, or through a burst terminate command. Burst terminate commands are the general mechanism used by the ESDCTL for early burst termination, The burst terminate command is subject to the CAS latency and must be pipeline similar to the Read command, as shown in Figure 18-48 to Figure 18-68.

NOTE

The signals displayed in are internal signals, that is, interface signals between the M3IF and the ESDRAMC. All those figures are not cycle accurate and meant to show mainly the external pins activity. For cycle accurate diagrams, refer to Figure 18-70 to Figure 18-72.

SDRAM write cycles are different than read cycles in one important aspect. Whereas read data was delayed by the CAS latency, write data has no delay and is supplied at the same time as the Write command. Figure 18-56 illustrates an off-page write cycle followed by one that hits on-page. Note that the write data is driven during the same clock that the Write command is issued. A Burst Terminate command cancels the burst operation, but again without the CAS latency.

NOTE

ESDRAMC handles the HUNALIGN accesses in the following way. The M3IF module converts the original access address to a word align address toward the ESDRAMC. The HBSTRB are used by the ESDRAMC to drive the correct value on the DQM signals toward the external memory.

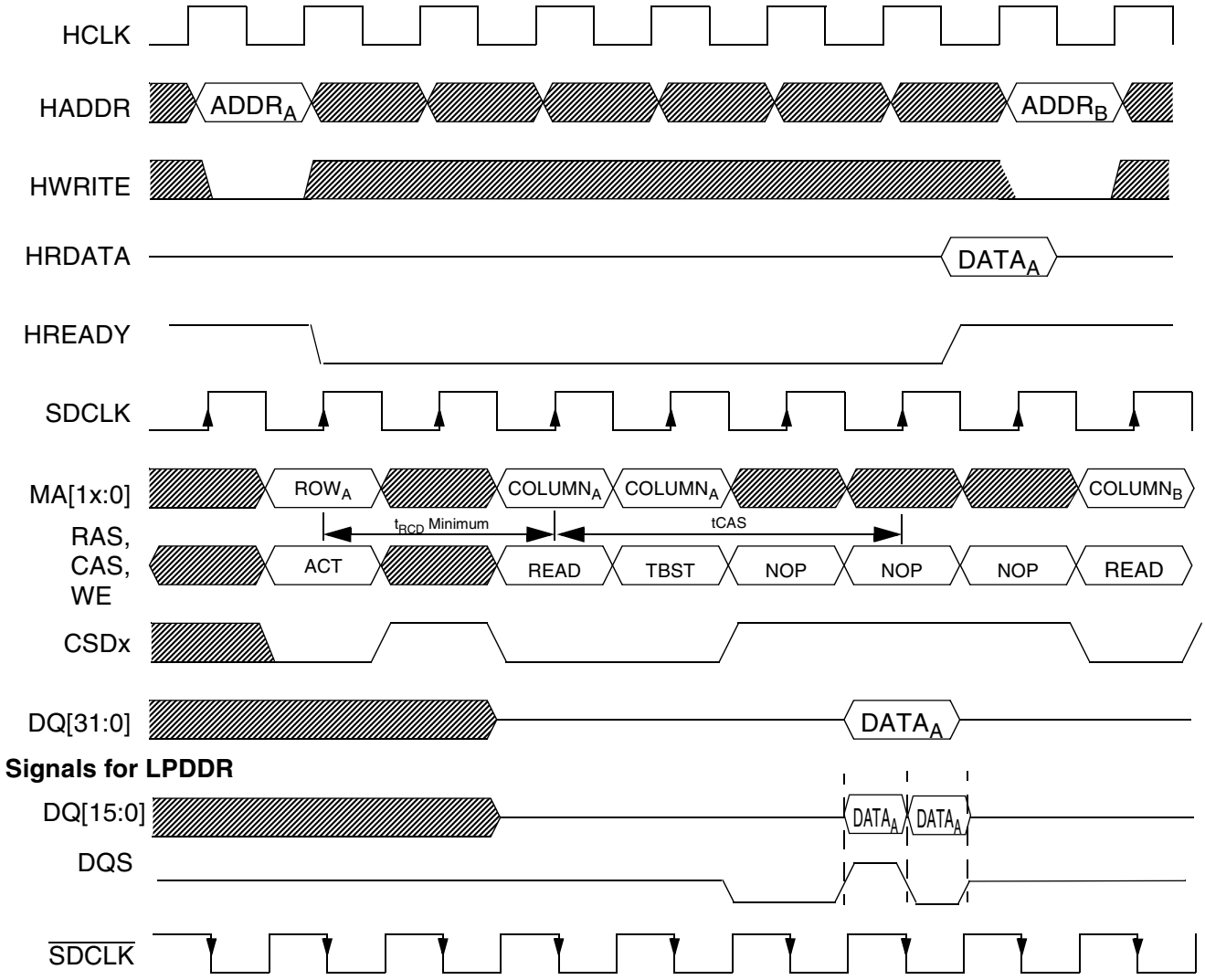


Figure 18-48. SDR and LPDDR Off-Page Single Read Timing Diagram (32-Bit Memory for SDR and 16-Bit for LPDDR)

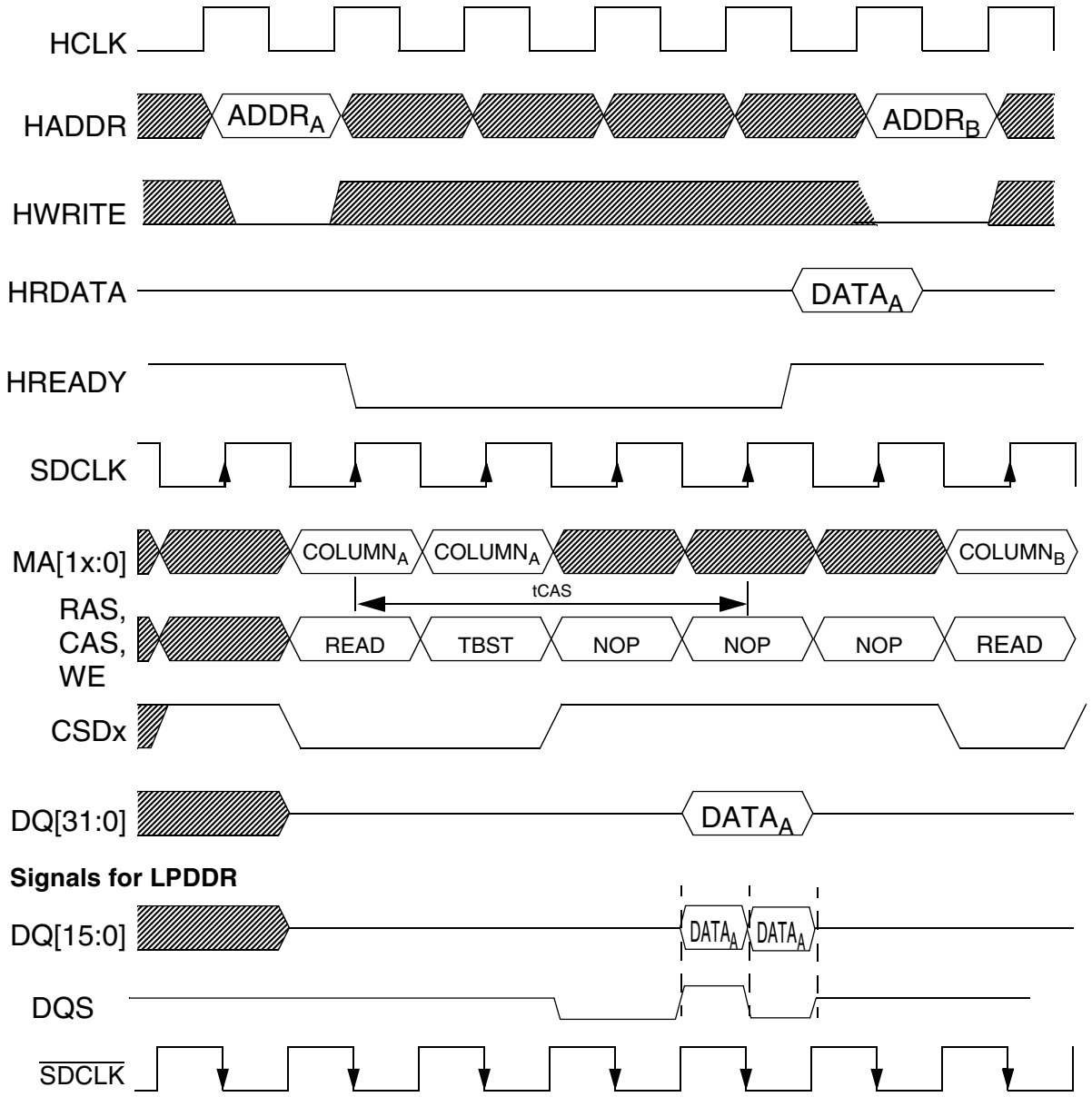


Figure 18-49. SDR and LPDDR On-Page Single Read Timing Diagram (32-Bit Memory for SDR, 16-Bit for LPDDR)

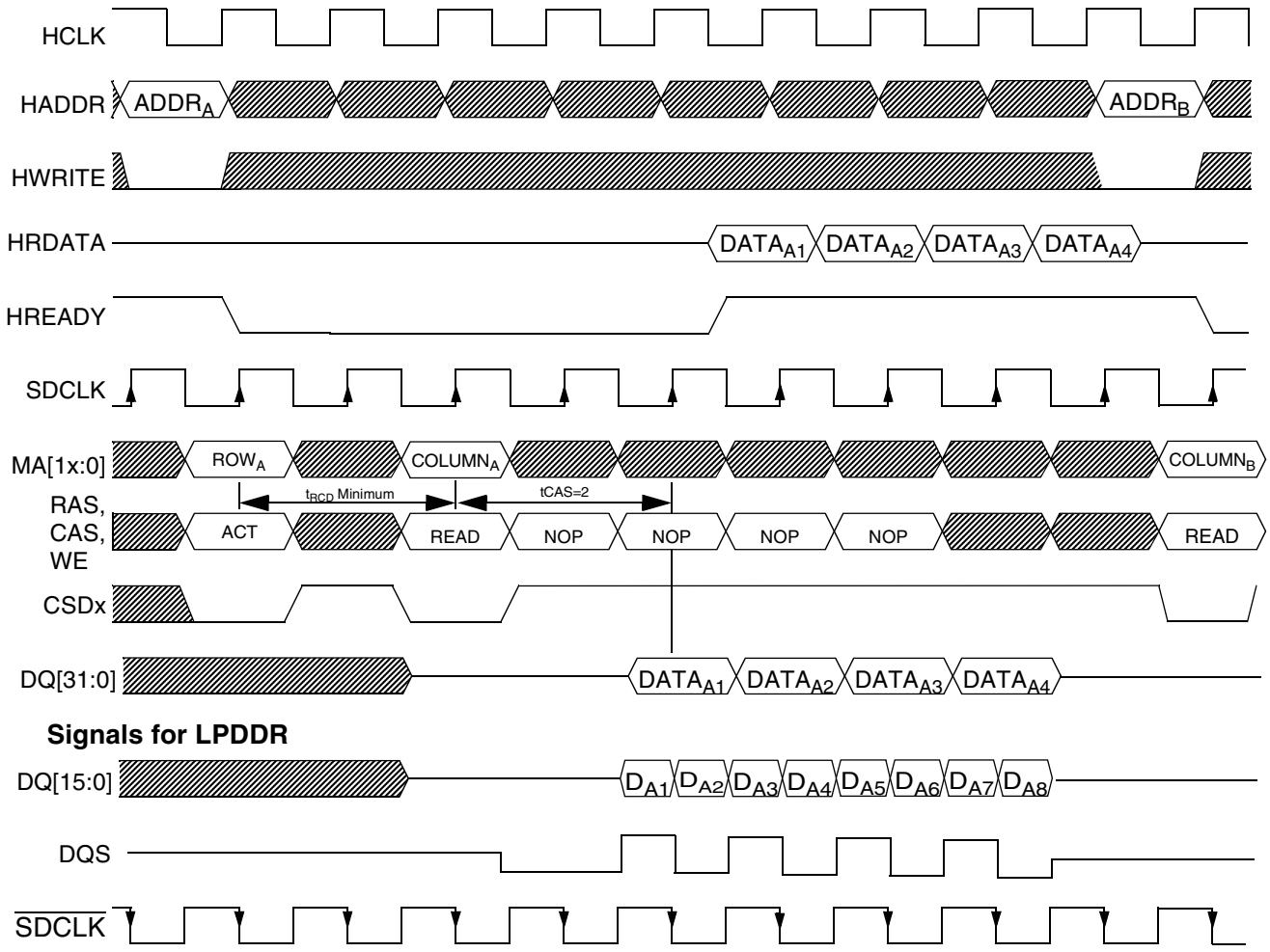


Figure 18-50. SDR and LPDDR Off-Page Burst Read Timing Diagram (32-Bit Memory for SDR or 16-Bit for LPDDR)

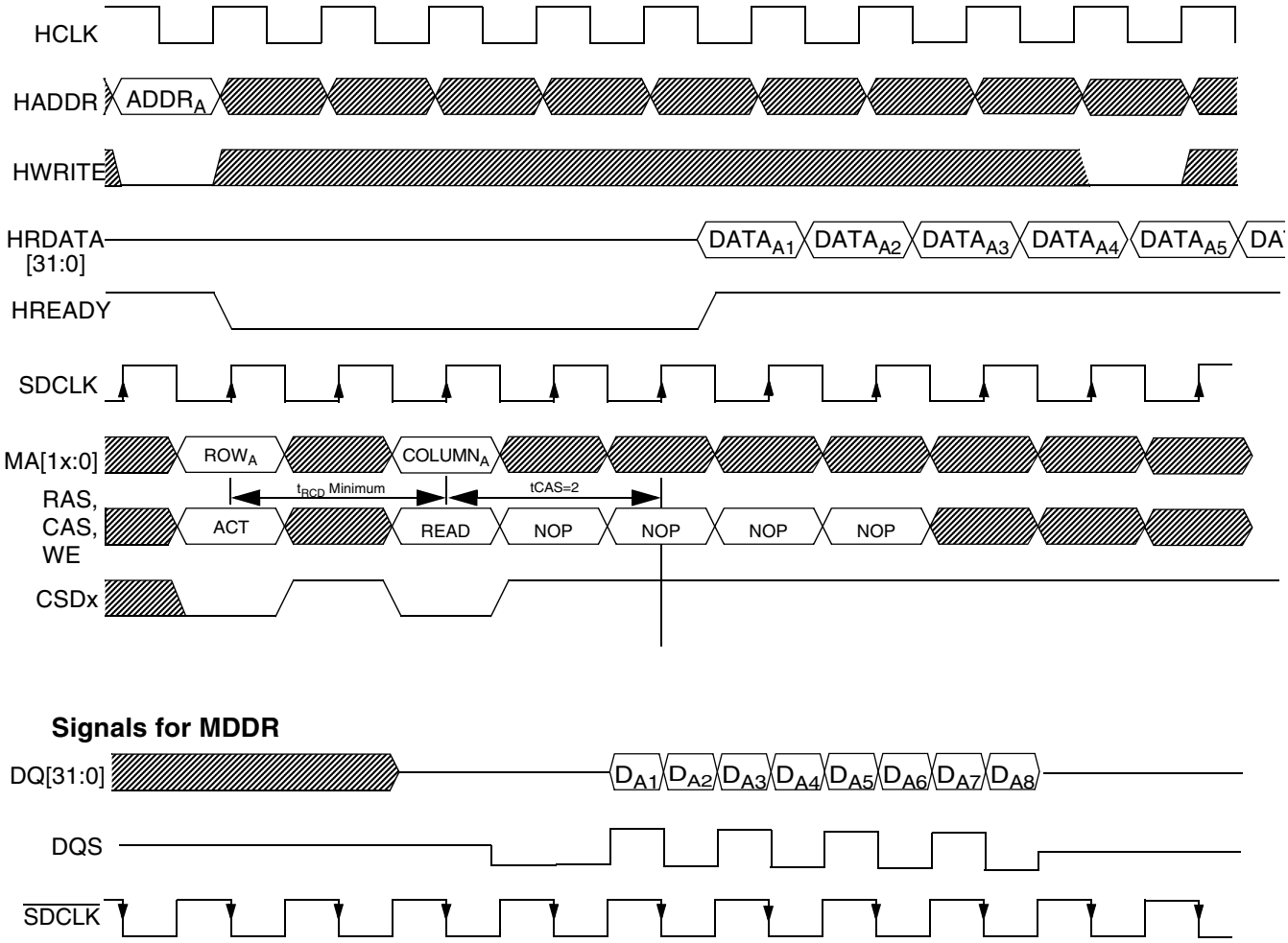


Figure 18-51. AHB 32-Bit read from a LPDDR: Off-Page Burst Read Timing Diagram (32-Bit)

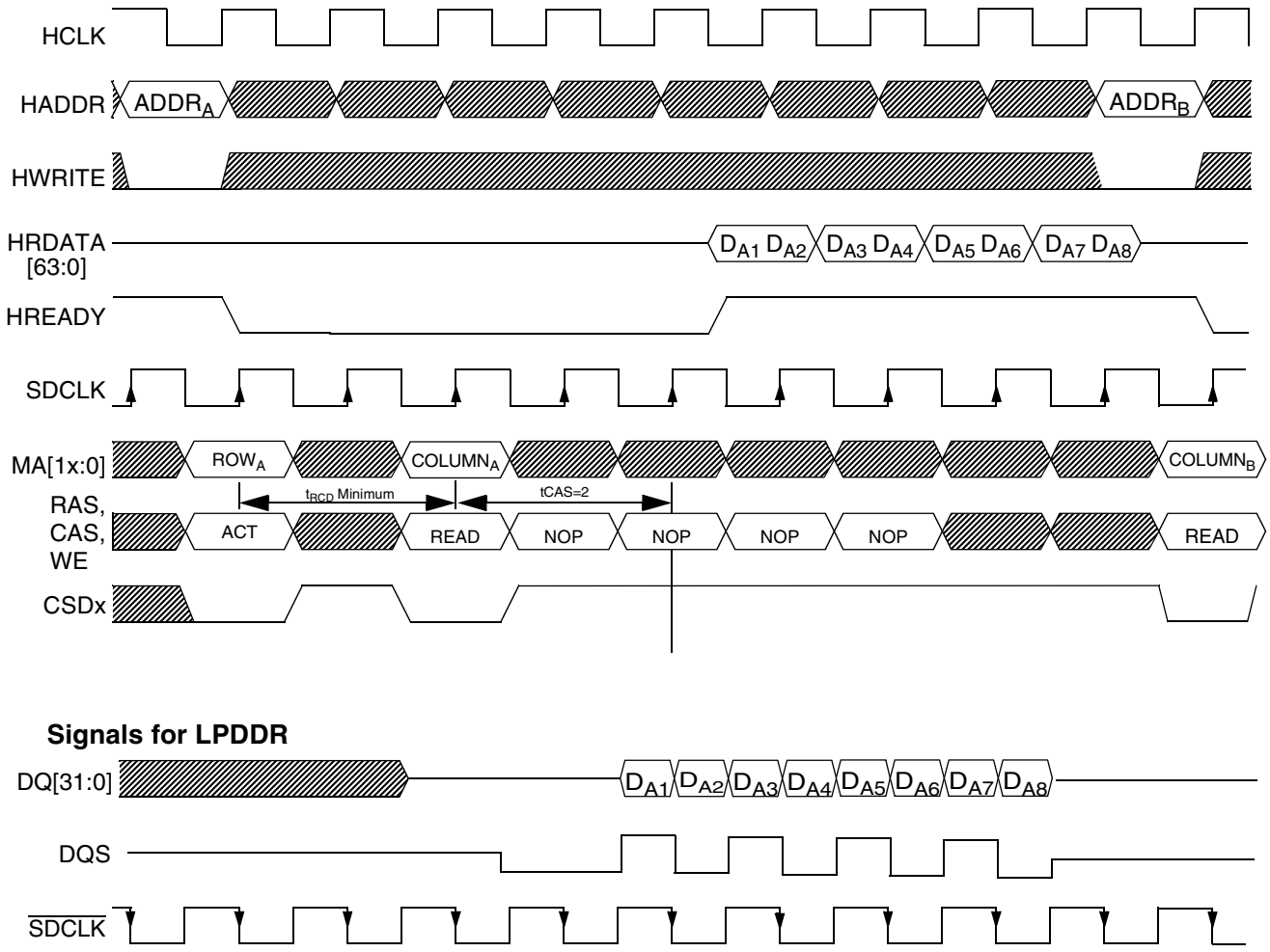


Figure 18-52. AHB 64-Bit read from a LPDDR: Off-Page Burst Read Timing Diagram (32-Bit)

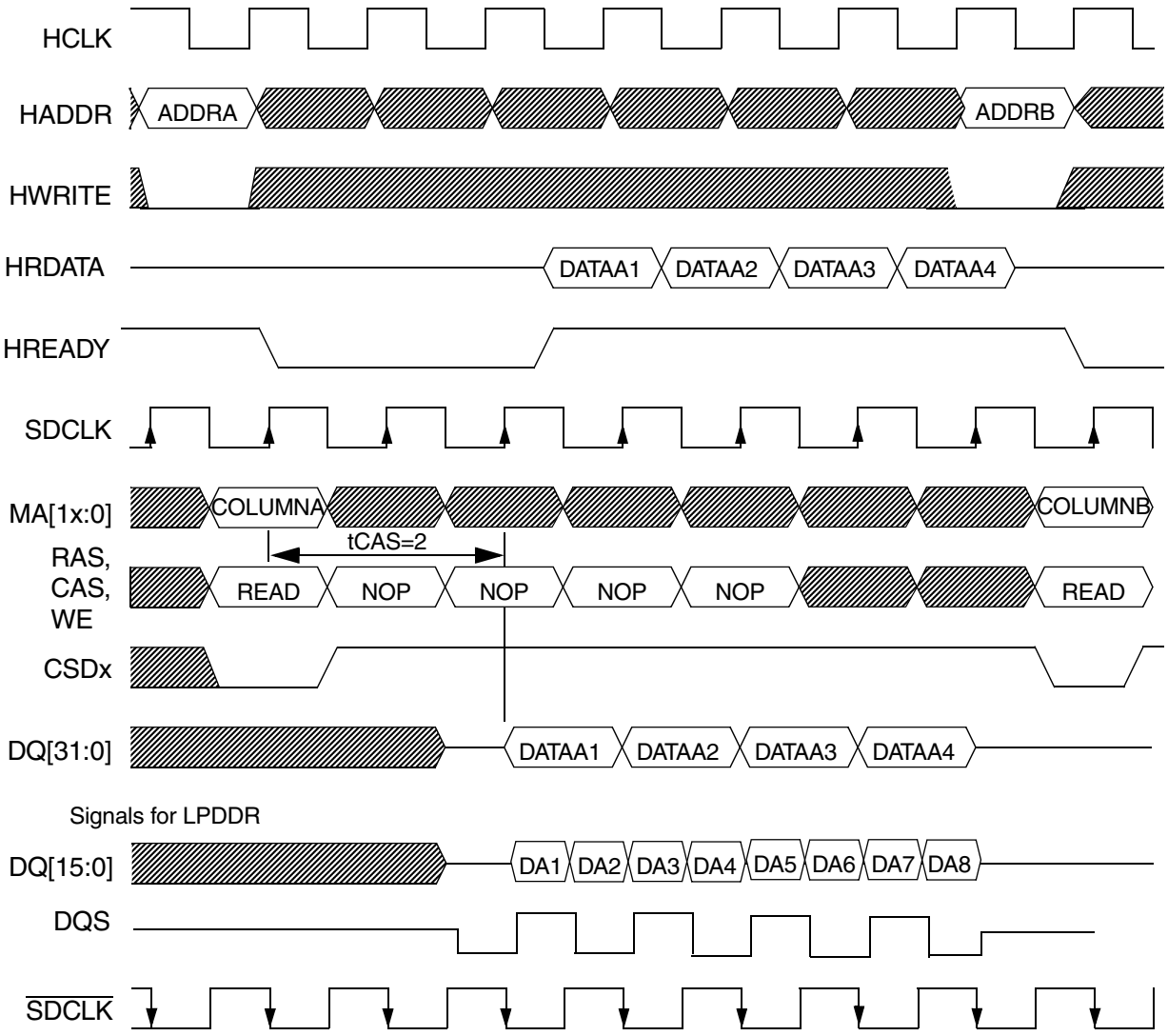


Figure 18-53. SDR and LPDDR On-Page Burst Read Timing Diagram (32-Bit Memory for SDR and 16-Bit for LPDDR)

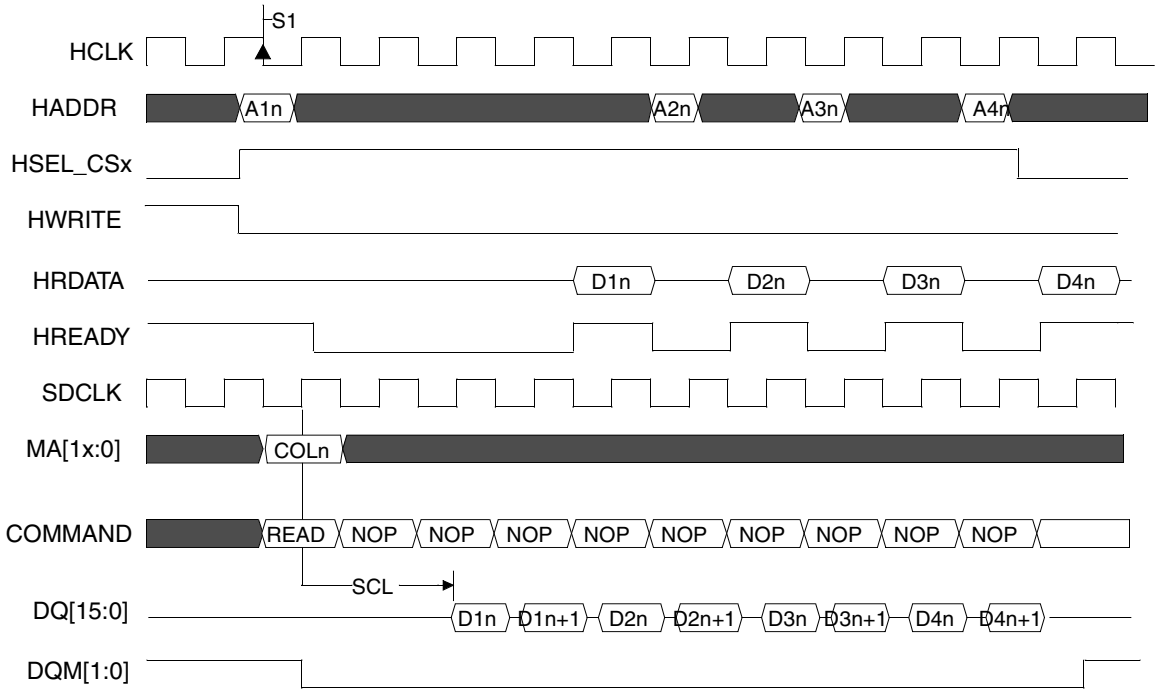


Figure 18-54. On-Page Burst Read Timing Diagram (16-Bit Memory for SDR, LPDDR 8-Bit Is Not Supported)

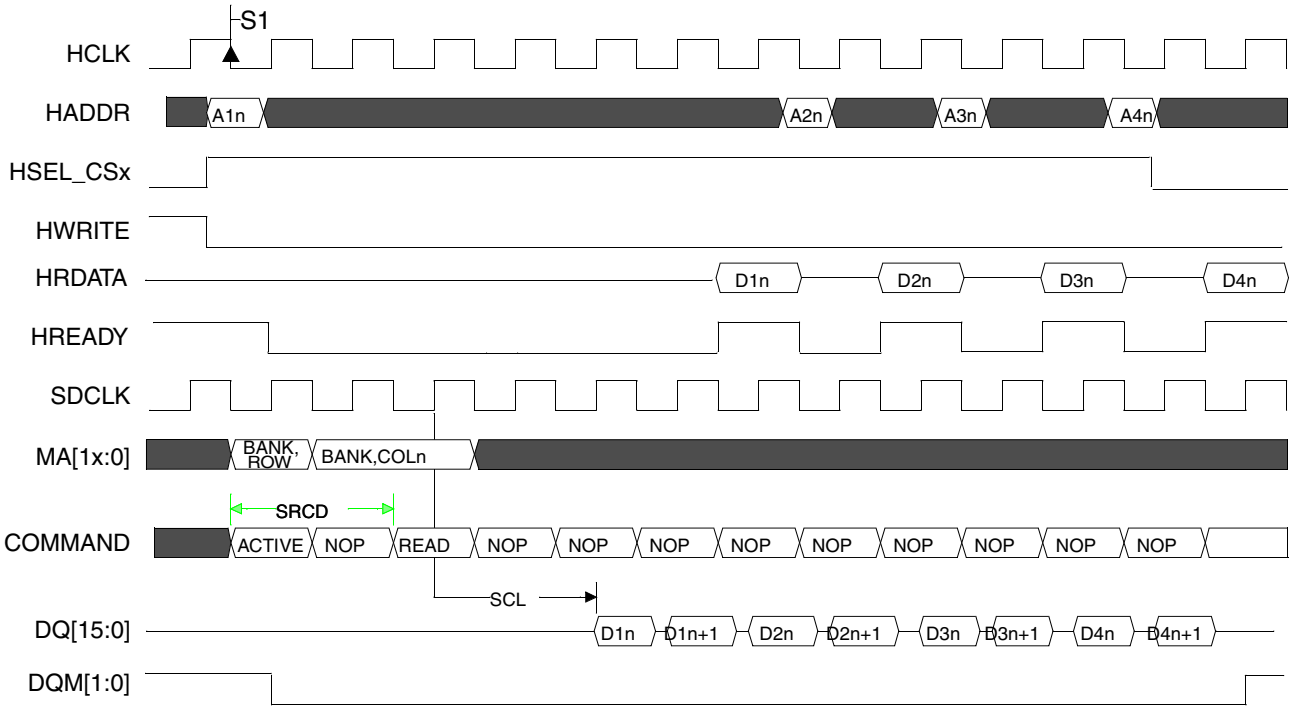


Figure 18-55. Off-Page Burst Read Timing Diagram (16-Bit Memory, LPDDR 8-Bit Is Not Supported)

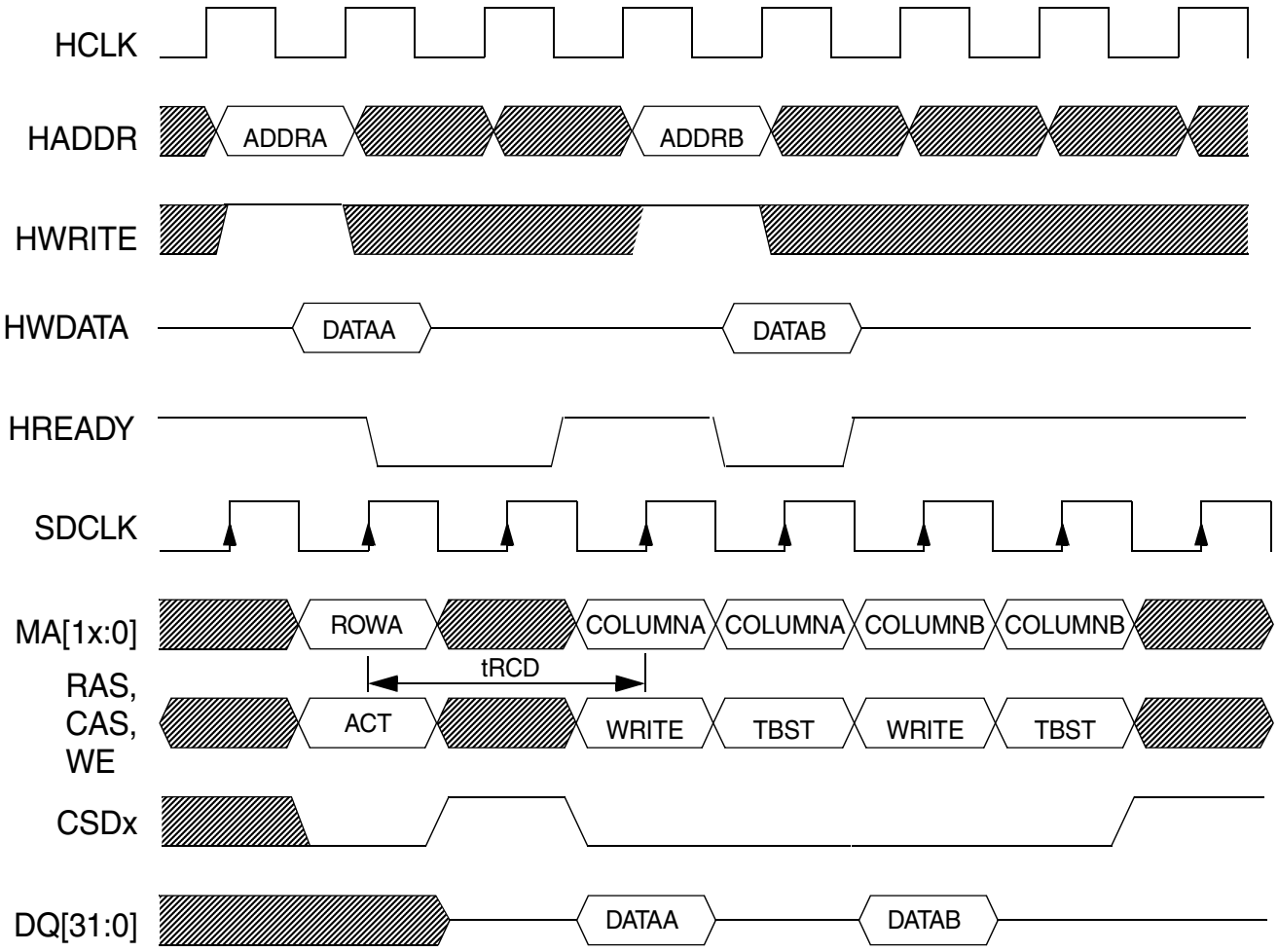
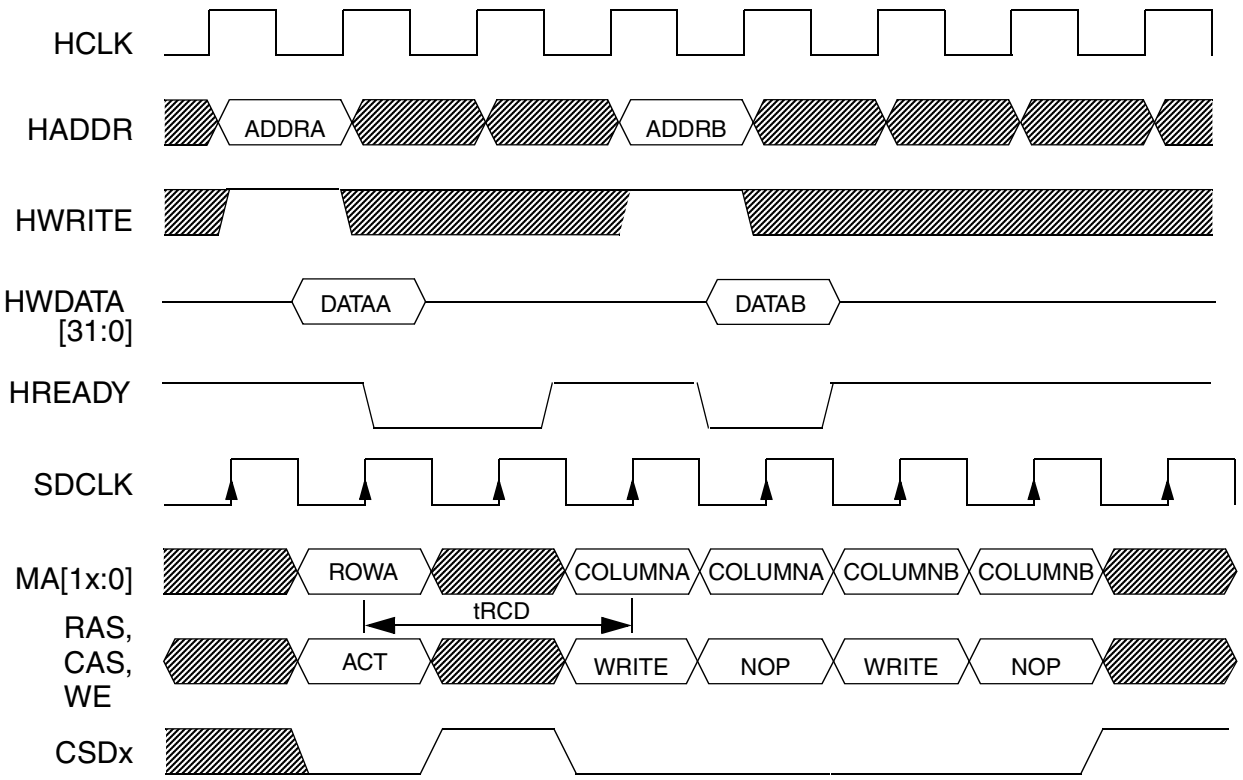


Figure 18-56. SDR Off-Page Write Followed by On-Page Write Timing Diagram



Signals for LPDDR

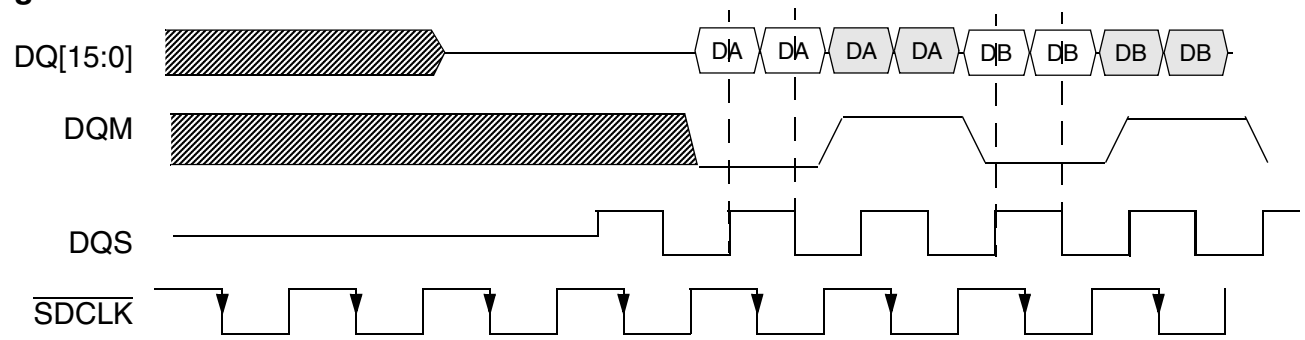


Figure 18-57. LPDDR Off-Page Write Followed by On-Page Write Timing Diagram

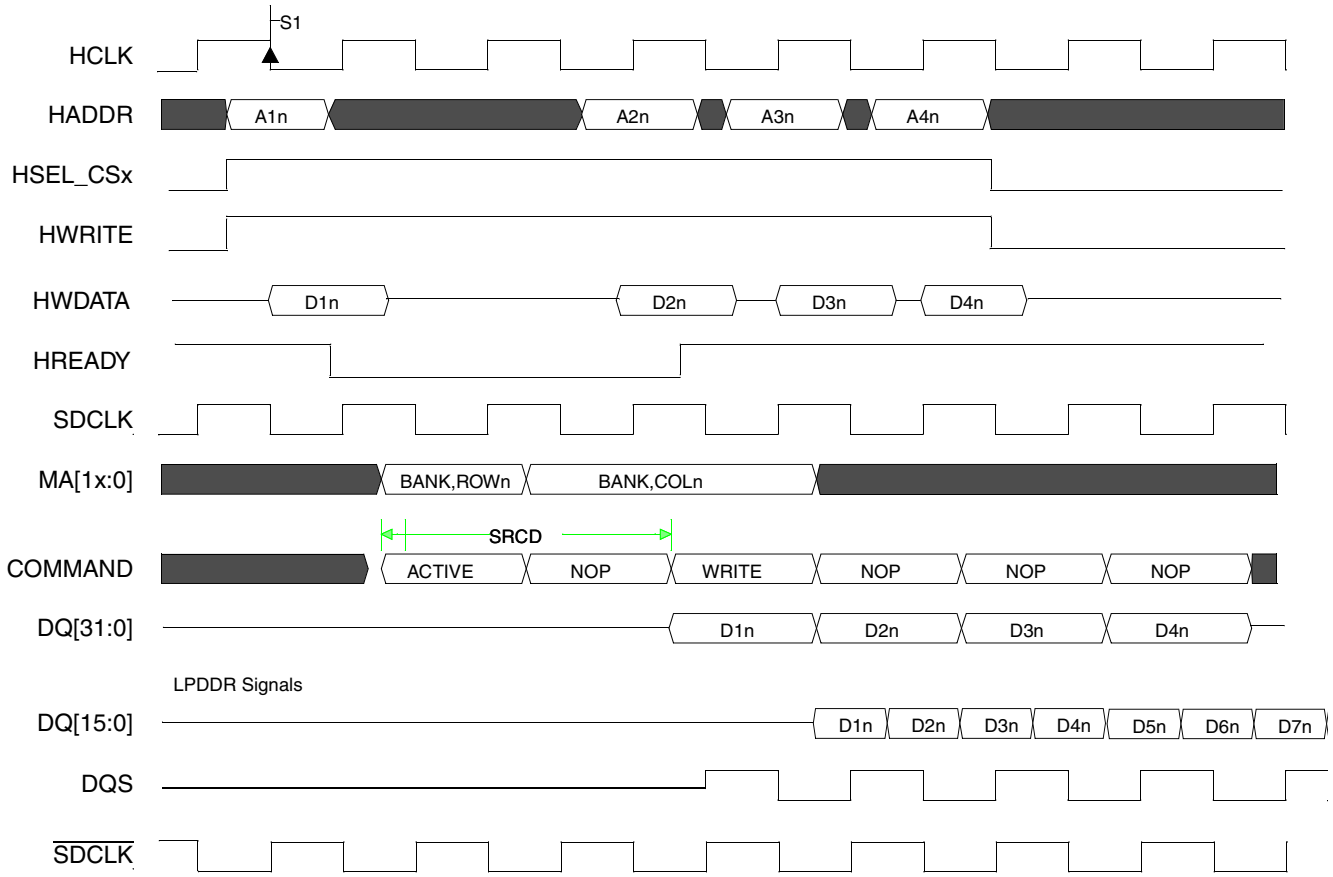


Figure 18-58. Off-Page Burst Write Timing Diagram (32-Bit Memory for SDR and 16-Bit for LPDDR)

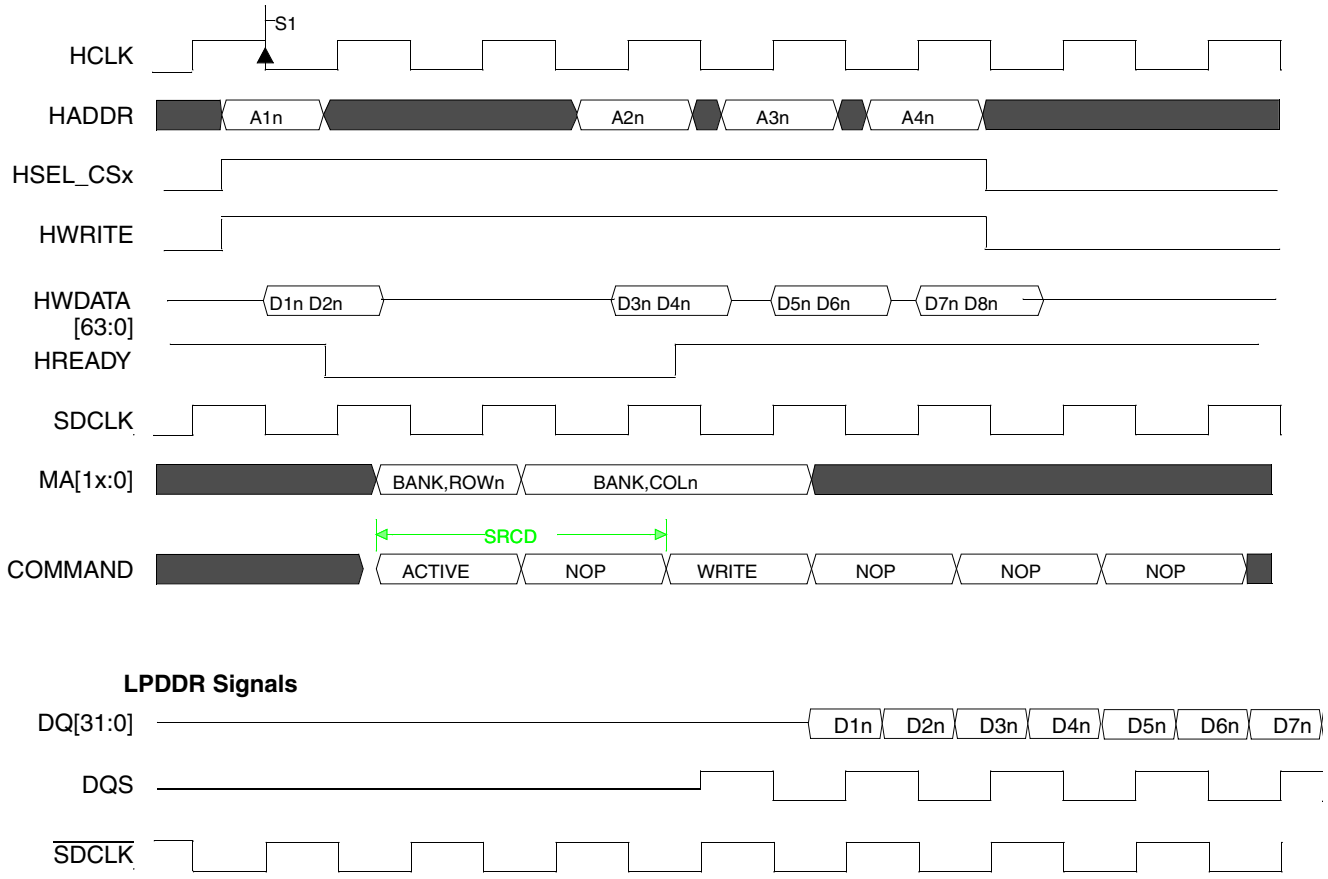


Figure 18-59. AHB 64-Bit Write to LPDDR: Off-Page Burst Write Timing Diagram (32-Bit Memory)

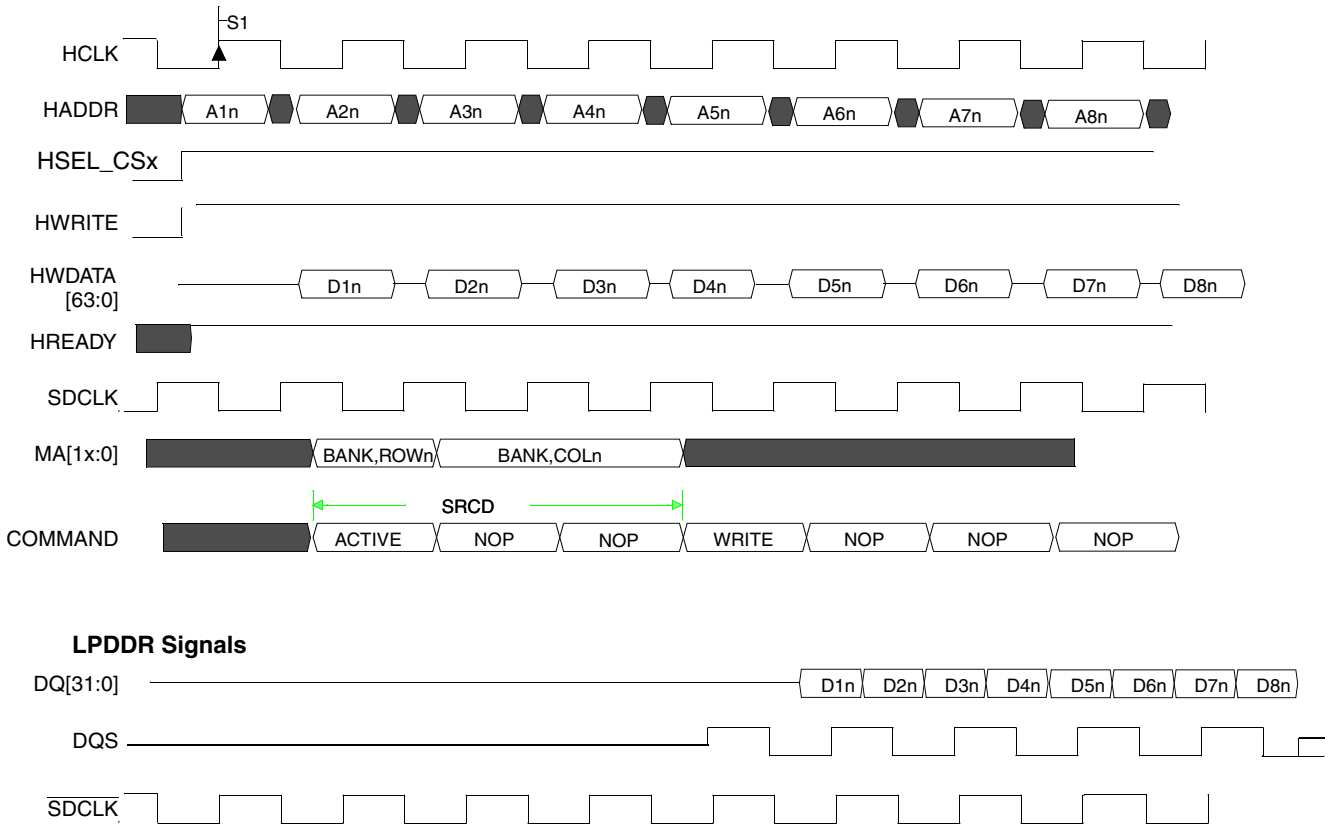


Figure 18-60. AHB 32-Bit write to LPDDR: Off-Page Burst Write Timing Diagram (32-Bit Memory)

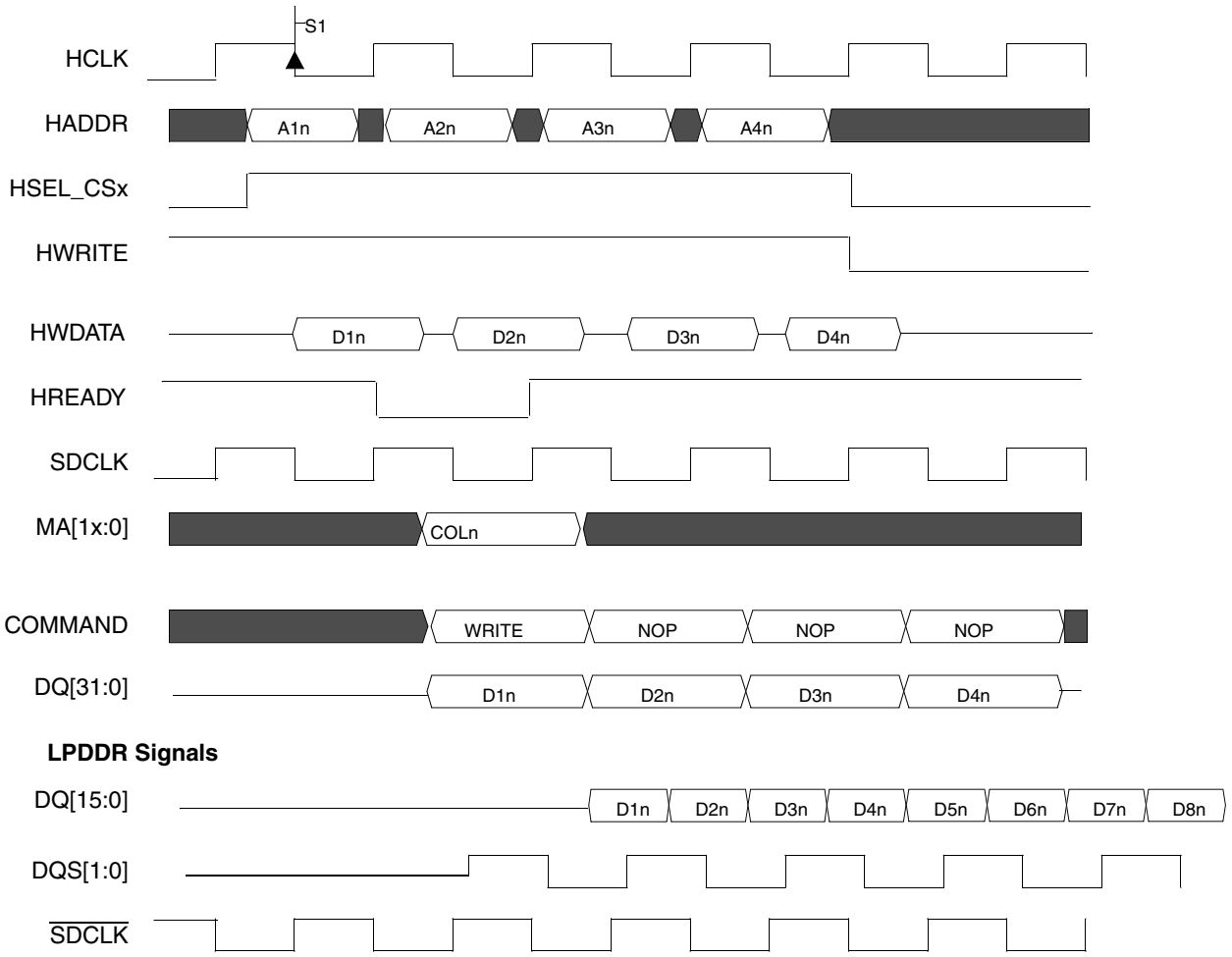


Figure 18-61. On-Page Burst Write Timing Diagram (32-Bit Memory for SDR and 16-Bit for LPDDR)

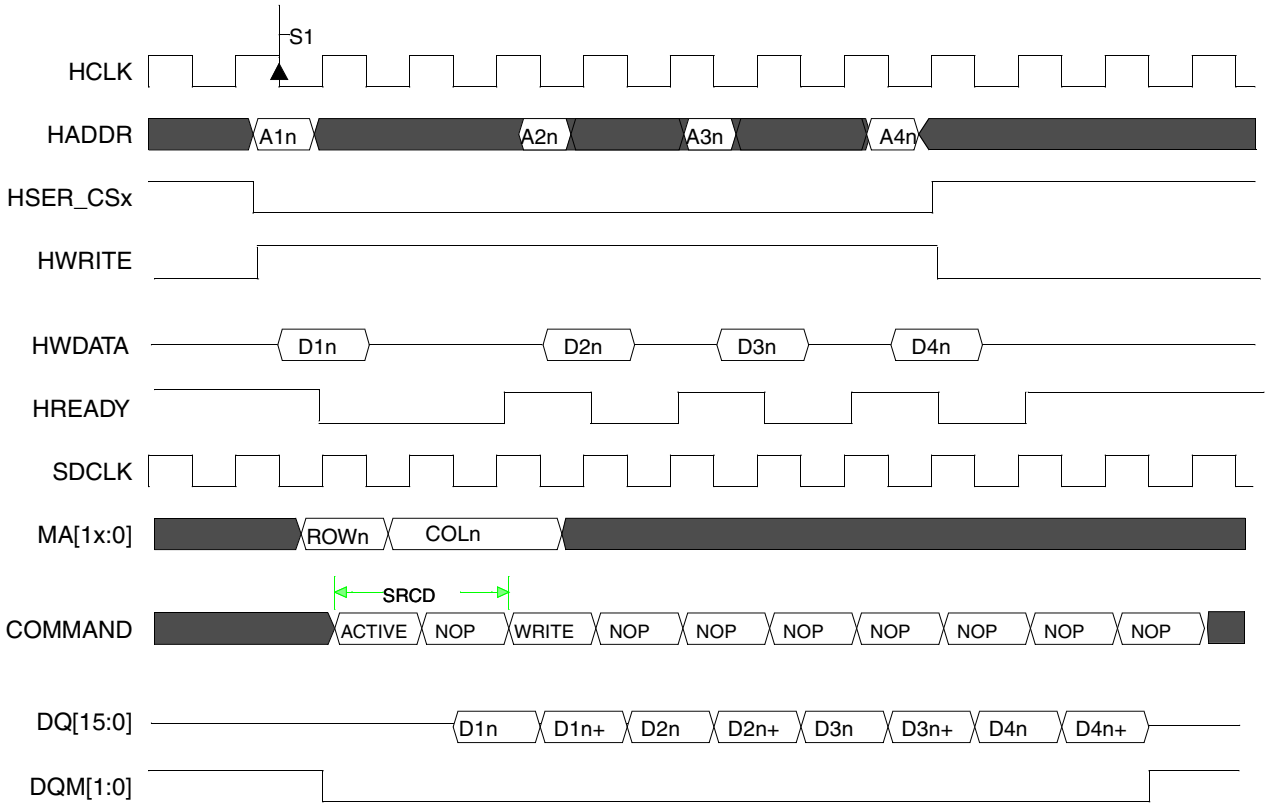


Figure 18-62. Off-Page Burst Write Timing Diagram (SDR 16-bit Memory, LPDDR 8-Bit is Not Supported)

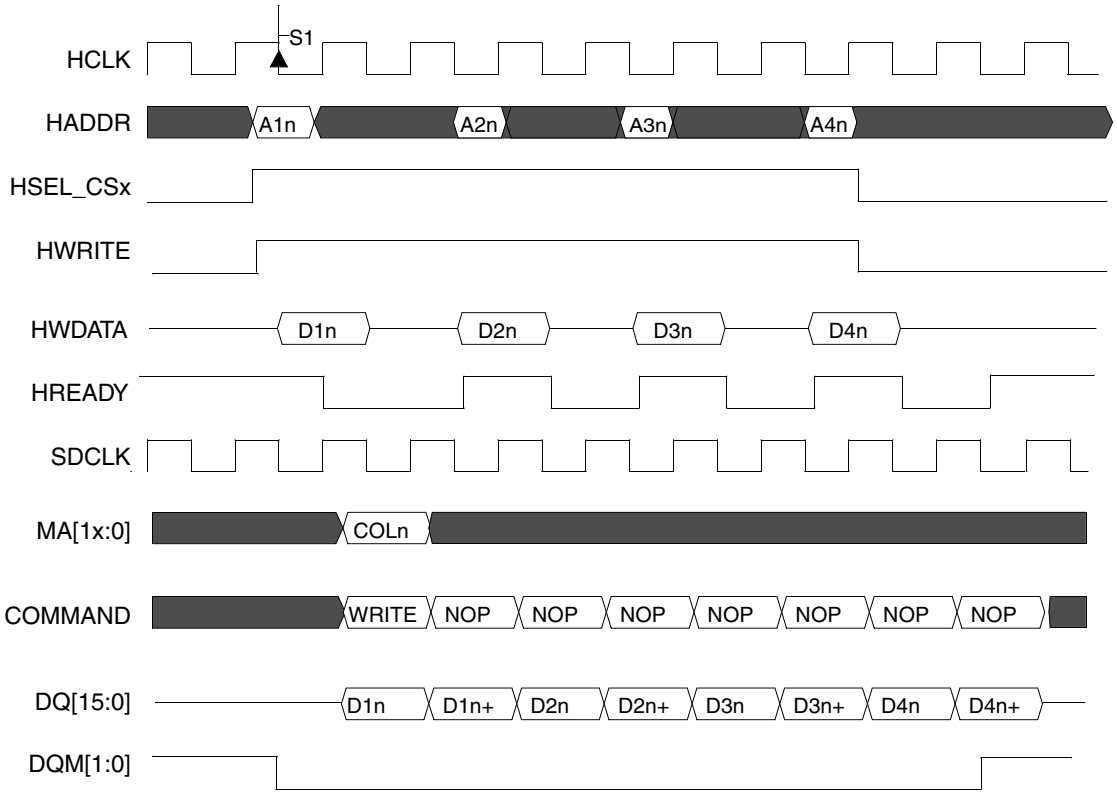


Figure 18-63. On-Page Burst Write Timing Diagram (SDR 16-Bit Memory, LPDDR 8-Bit Memory is Not Supported)

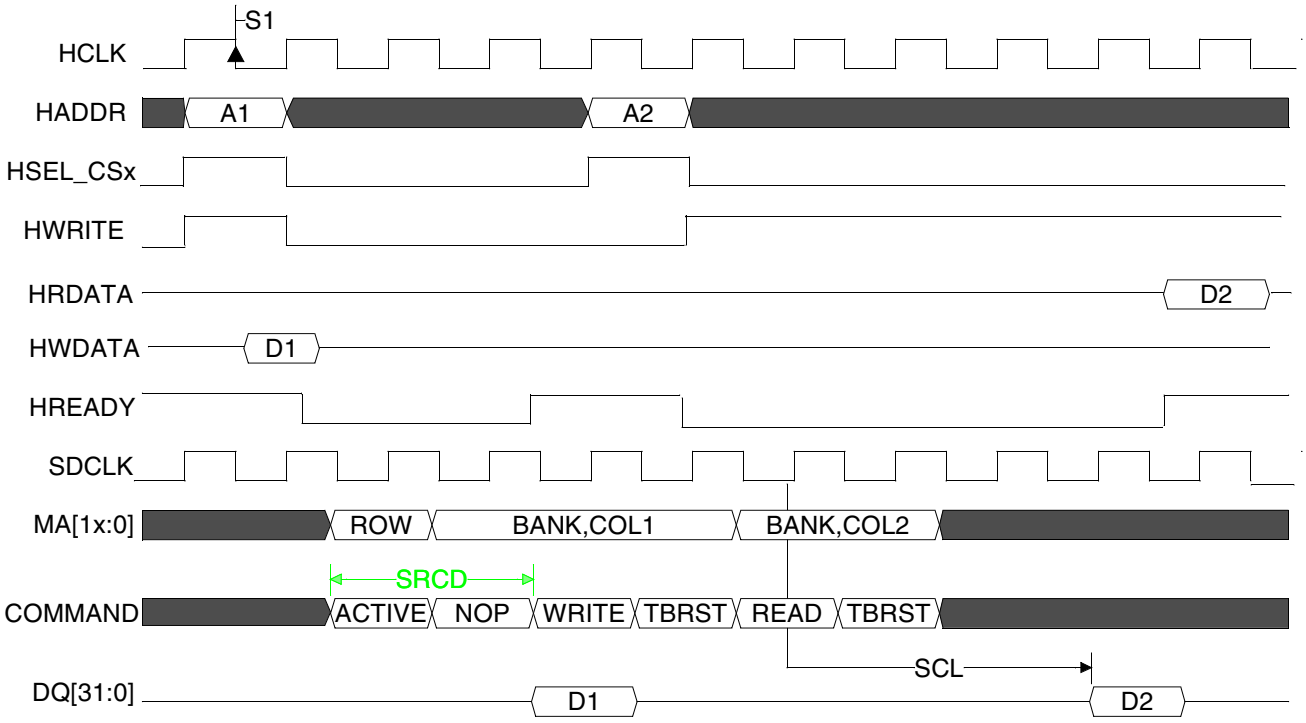


Figure 18-64. SDR Single Write followed by On-Page Read Timing Diagram

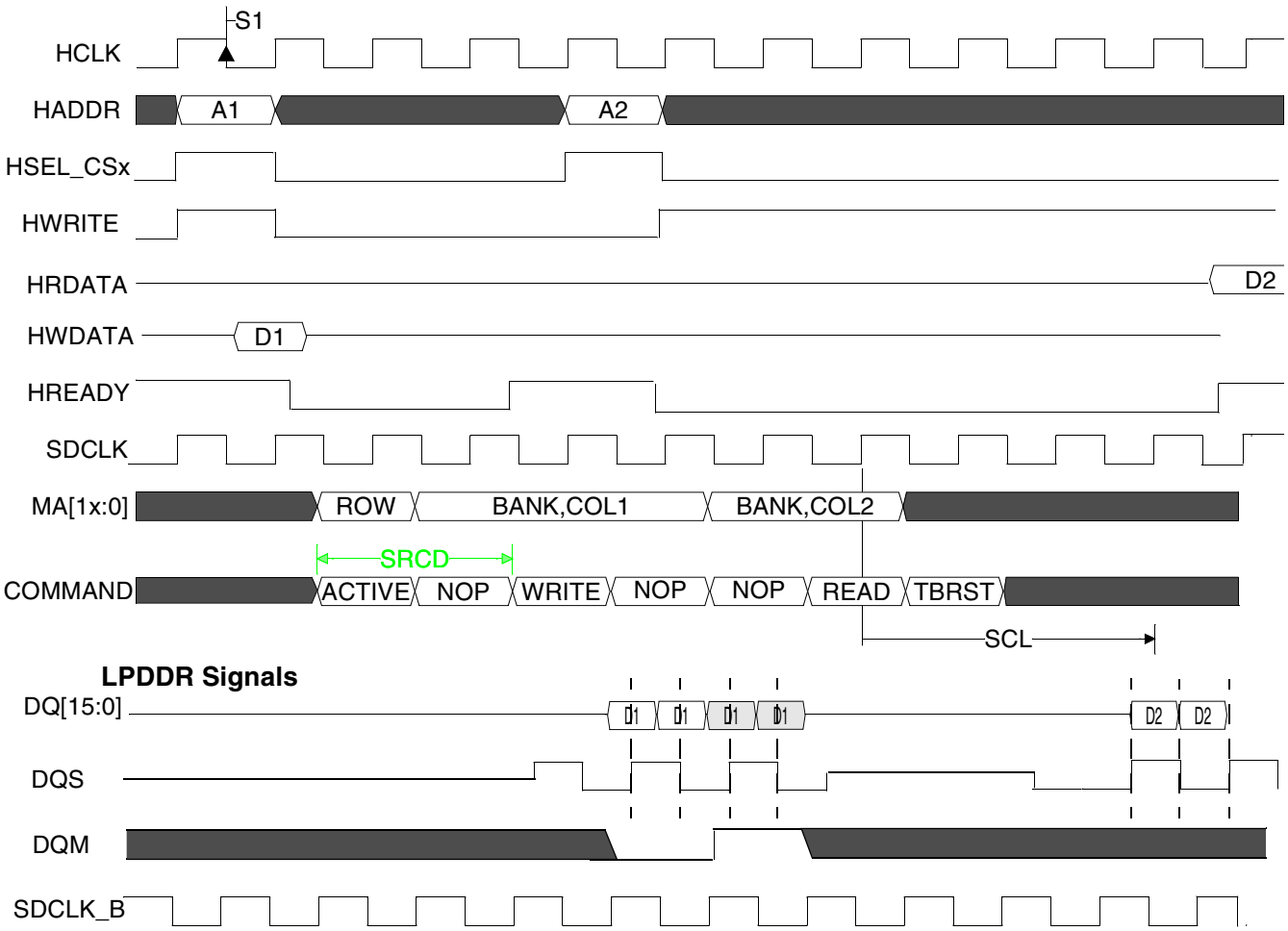


Figure 18-65. LPDDR Single Write Followed by On-Page Read Timing Diagram

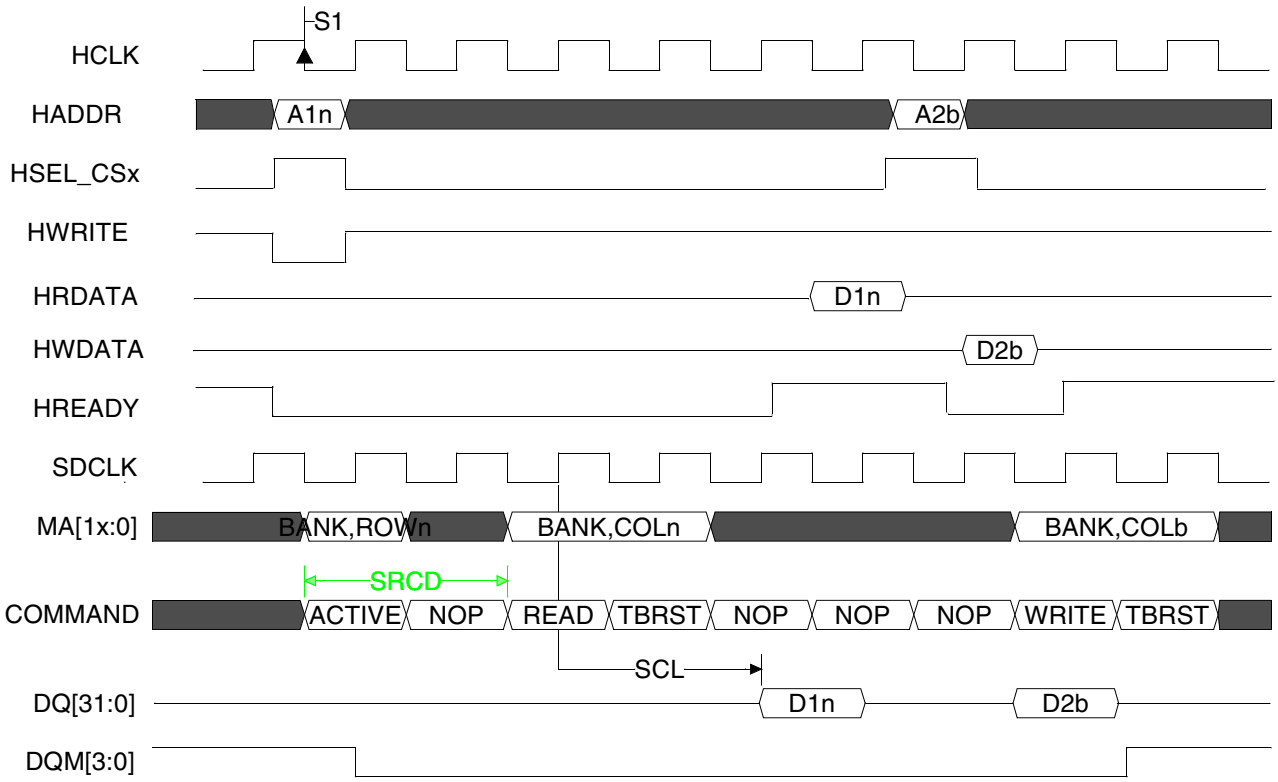


Figure 18-66. SDR Single Read Followed by On-Page Write Timing Diagram

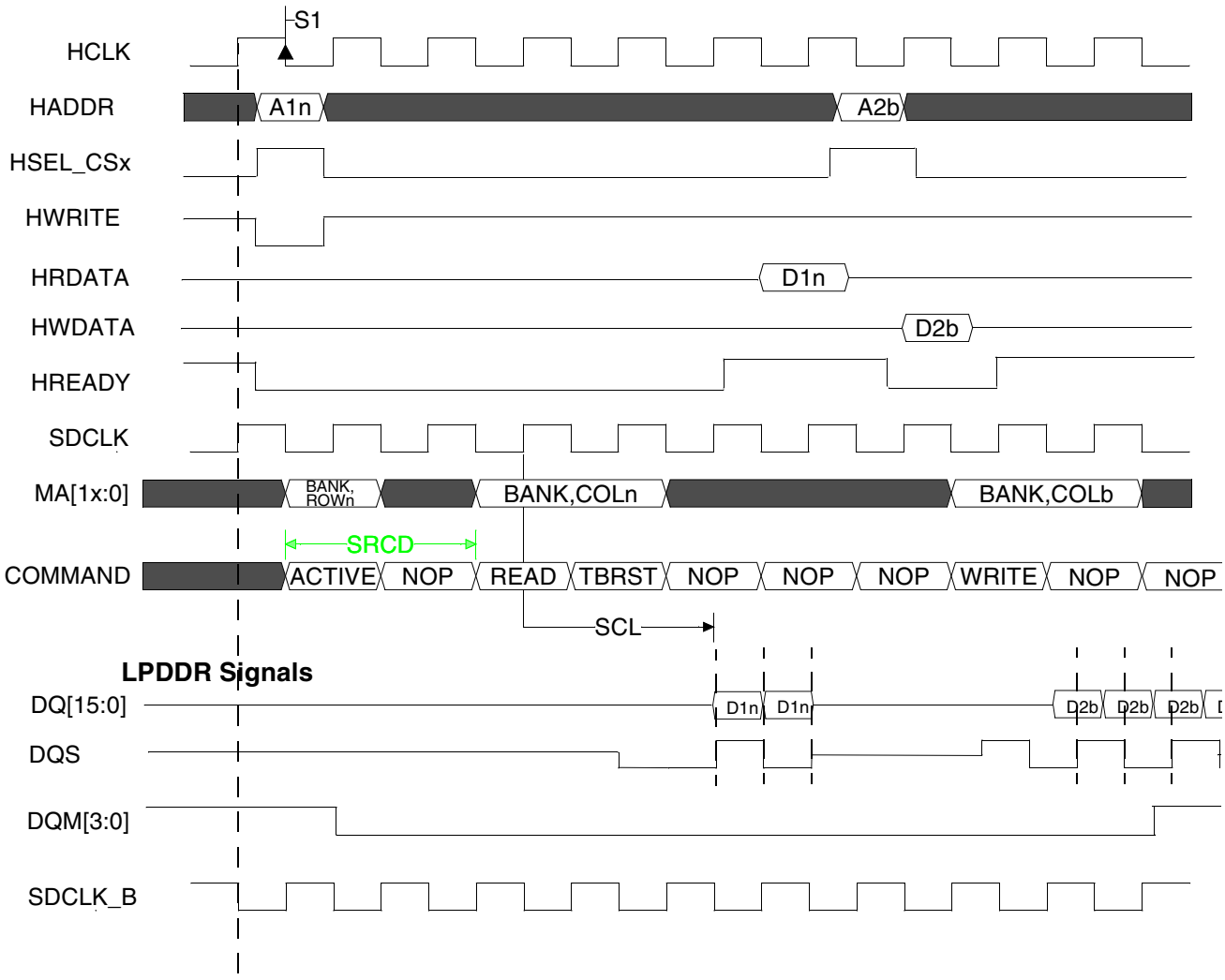


Figure 18-67. LPDDR Single Read Followed by On-Page Write Timing Diagram

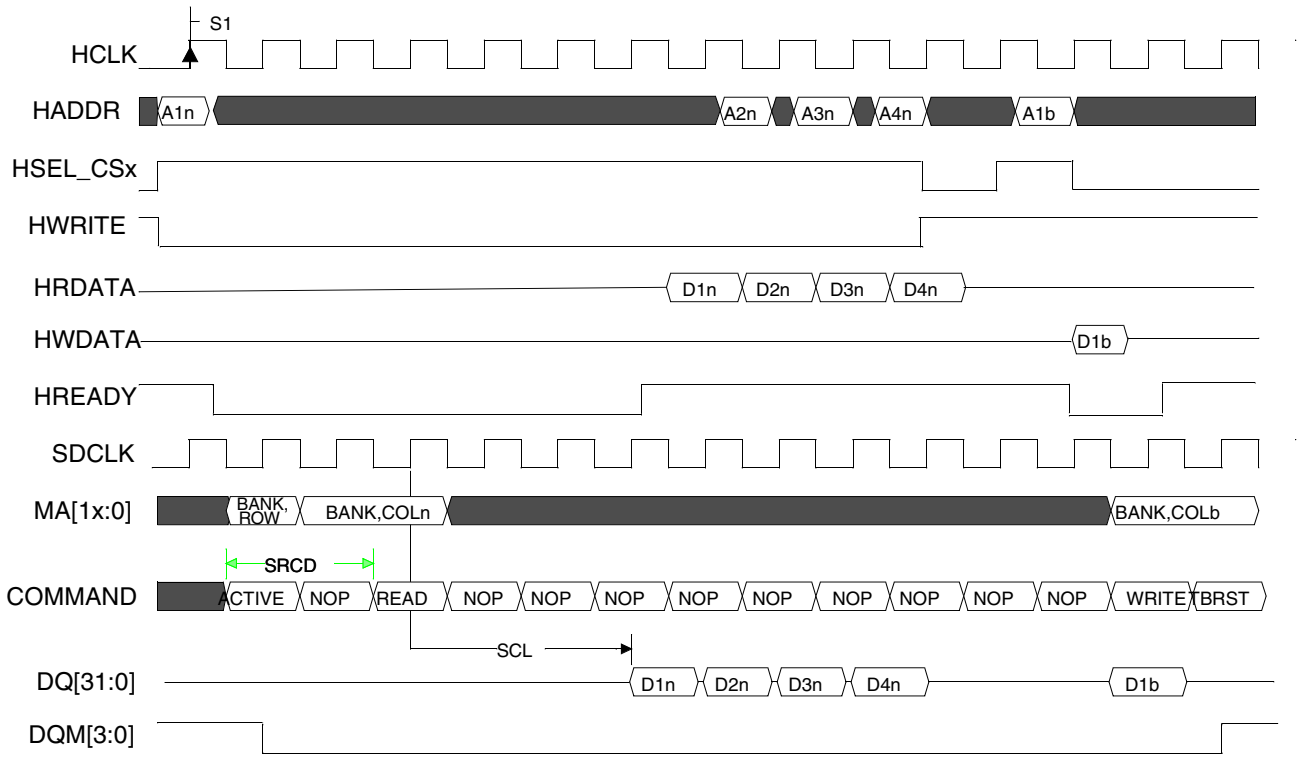


Figure 18-68. SDR Burst Read Followed by On-Page Write Timing Diagram

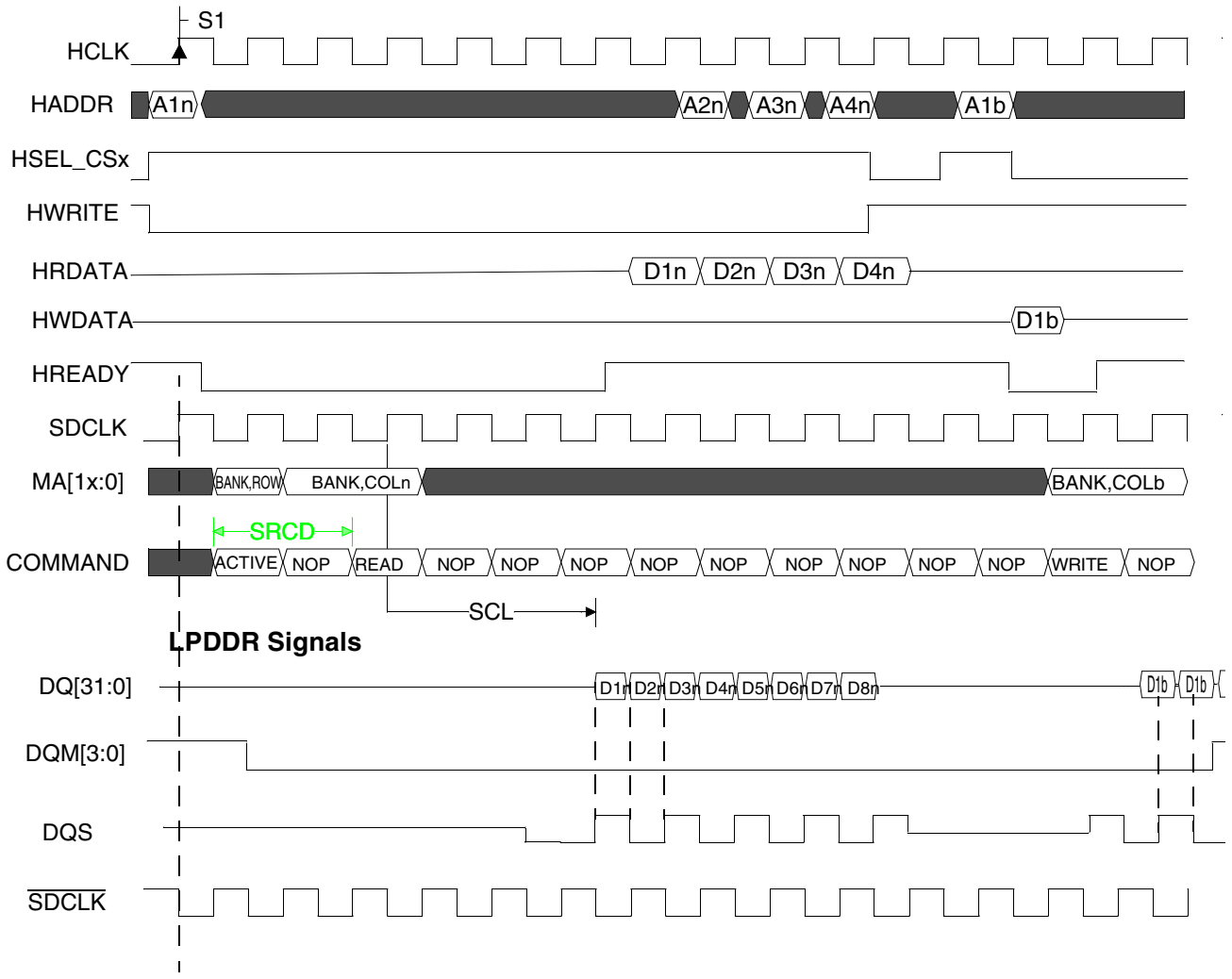


Figure 18-69. LPDDR Burst Read Followed by On-Page Write Timing Diagram

18.4.7.1 SDR Cycle Accurate Enhanced SDRAM Controller Accesses

This section provides cycle accurate timing diagrams for several ESDRAMC (AMBA AHB Lite) supported read and write accesses to both 16 and 32-bit data width SDR memory devices from only one master. This diagrams are provided to emphasis ESDRAMC performance for single master hit (ACTIVE row) requests. The CAS latency for all diagrams is 2 cycles and burst length is set to 4 words for 16-bit memory and 8 words for 32-bit memory.

18.4.7.1.1 Single Read Word Access to 16-Bit Memory

The markers in Figure 18-70 marks the request access time, for example, the time period between HREADY goes LOW (the ESDRAMC starts to execute the request) and HREADY goes HIGH (the ESDRAMC request execution is completed).

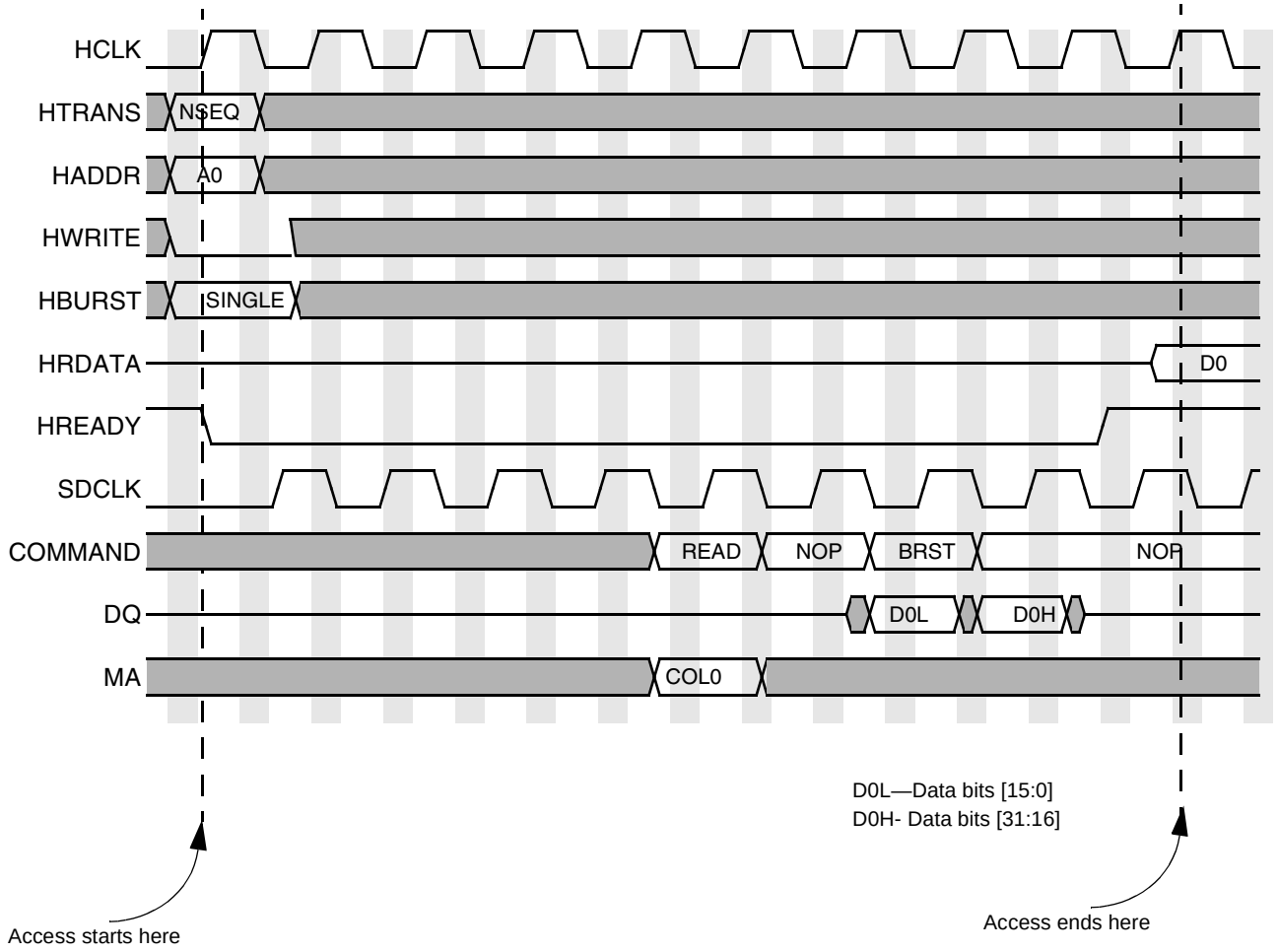


Figure 18-70. Single on Page Read—Word Access to 16-Bit Memory (Cycle Accurate)

18.4.7.1.2 Misaligned INCR4 Burst Read Access to 16-Bit Memory

The markers in Figure 18-71 marks the request access time, for example, the time period between HREADY goes LOW (the ESDRAMC starts to execute the request) and HREADY goes HIGH (the ESDRAMC request execution is completed).

Two READ commands are issued toward the SDRAM memory, since the misaligned read burst crosses the 16-bit SDRAM (4 words) memory boundary. The read addresses are 0x04, 0x08, 0x0C, and 0x10. Without issuing the second read the last SDRAM data will come from address 0x00. The ESDRAMC issue the second READ command in such a way (at a specific timing) so continuous data flow is obtained. There is no timing difference between an aligned and a misaligned access although the second one requires more commands.

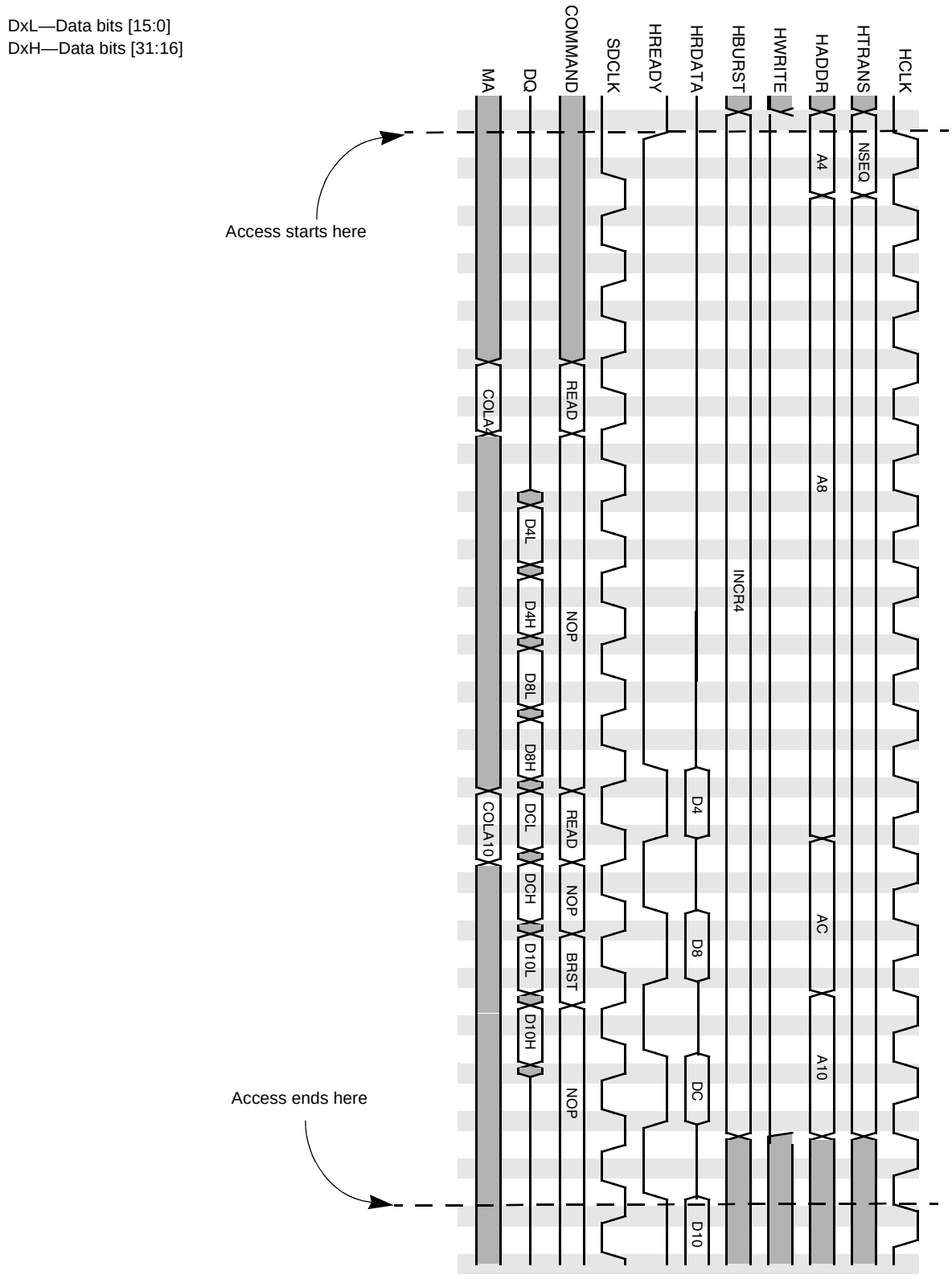


Figure 18-71. Misaligned on Page INCR4 Burst Read Access to 16-Bit Memory

18.4.7.1.3 Misaligned WRAP8 Burst Read Access to 32-Bit Memory

The markers in Figure 18-72 marks the request access time, that is, the time period between HREADY goes LOW (the ESDRAMC starts to execute the request) and HREADY goes HIGH (the ESDRAMC request execution is completed).

Only one READ command is issued toward the SDRAM memory, since the misaligned read burst crosses the 32-bit SDRAM memory (8 words) boundary at the memory “natural” boundary. The read addresses are 0x04, 0x08, 0x0C, 0x10, 0x14, 0x18, 0x1C and 0x00. Without issuing the second read the last SDRAM data will come from address 0x00 since this is the “natural” 32-bit memory device boundary as well.

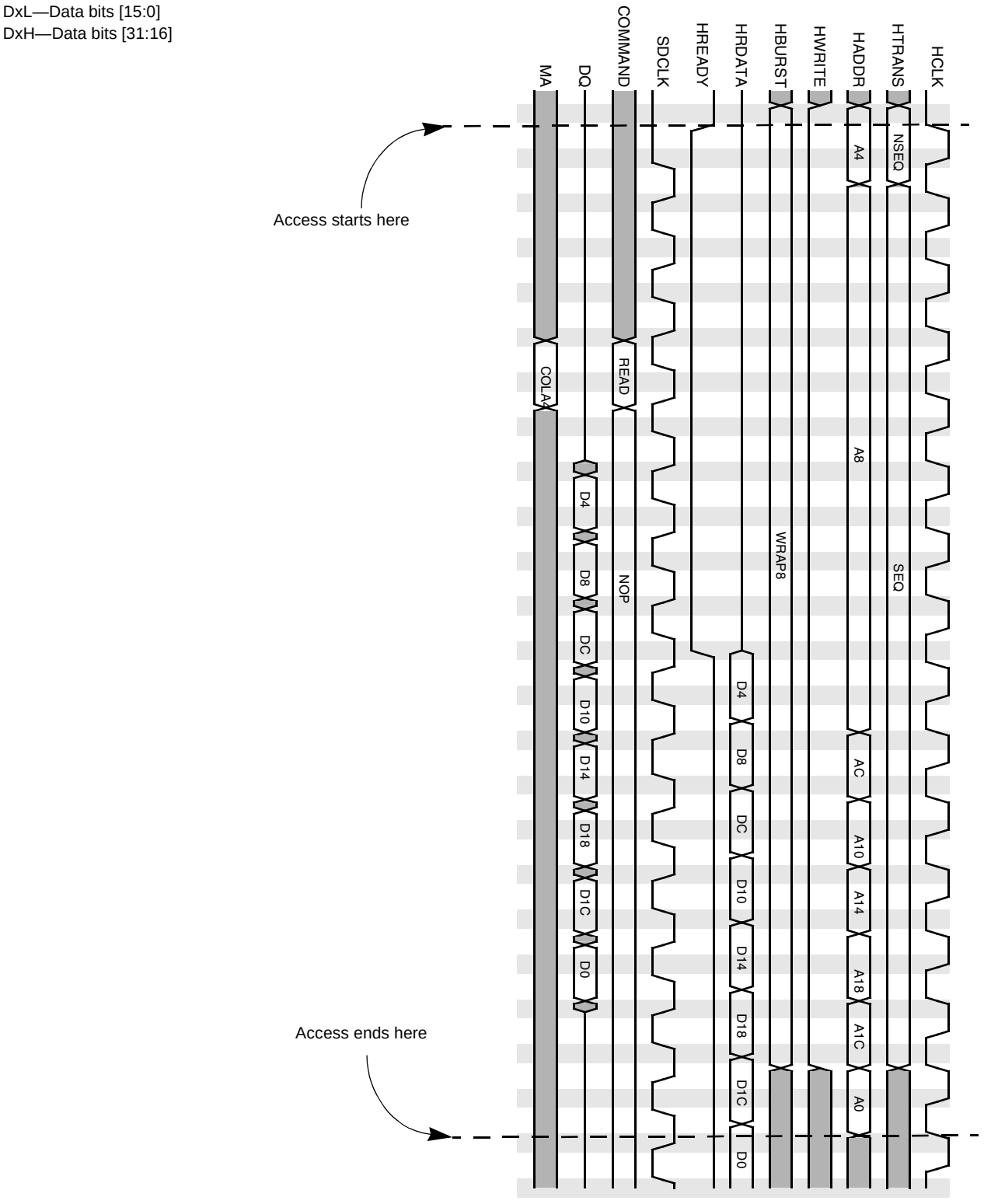


Figure 18-72. Misaligned WRAP8 Burst Read Access to 32-Bit Memory

18.4.7.2 Single Write Word Access to 32-Bit Memory

The markers in Figure 18-73 mark the request access time, for example, the time period between HREADY goes LOW (the ESDRAMC starts to execute the request) and HREADY goes HIGH (the ESDRAMC request execution is completed).

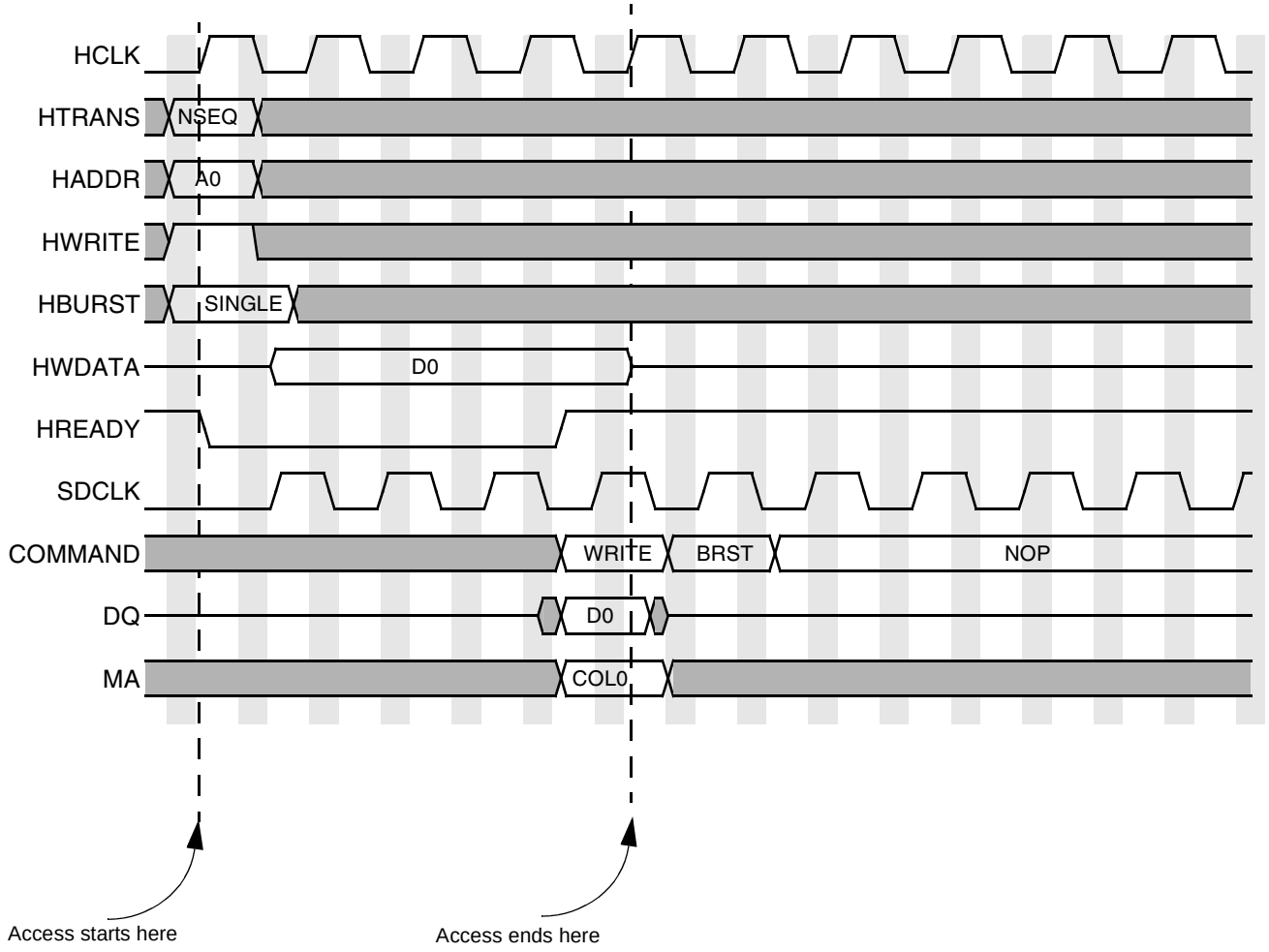


Figure 18-73. Single on Page Write—Word Access to 32-Bit Memory (Cycle Accurate)

18.4.7.2.1 INCR4 Burst Write Word Access to 32-Bit Memory

Two WRITE commands are issued toward the SDRAM memory, since the misaligned write burst crosses the 32-bit SDRAM (8 words) memory boundary. The write addresses are 0x14, 0x18, 0x1C, and 0x20. Without issuing the second write the last SDRAM data will be written to address 0x00. The ESDRAMC issue the second WRITE command in such a way (at a specific timing) so continuous data flow is obtained. There is no timing difference between an aligned and a misaligned access although the second one requires more commands. The markers in Figure 18-74 mark the request access time.

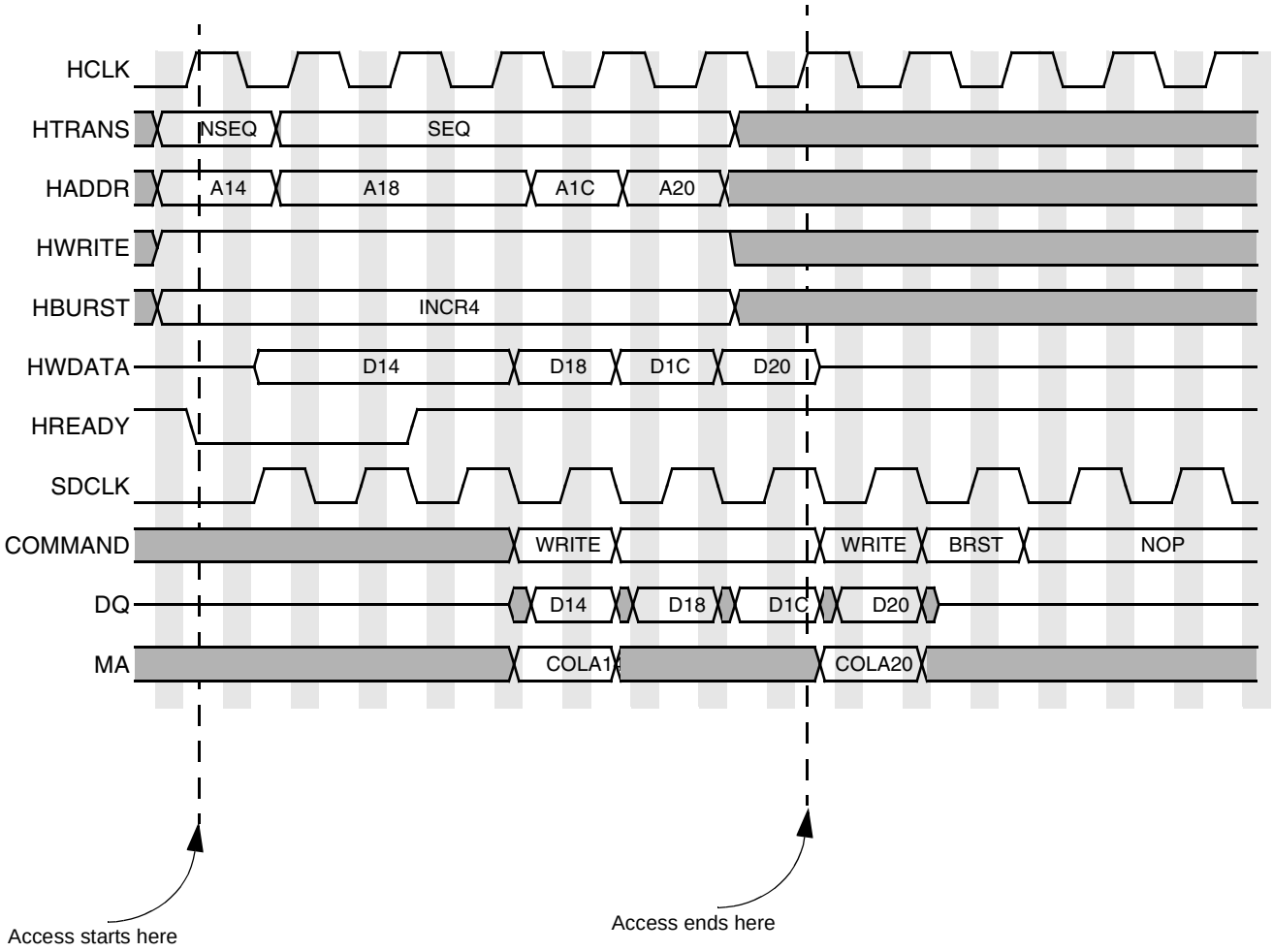


Figure 18-74. INCR4 Burst on Page Write—Word Access to 32-Bit Memory (Cycle Accurate)

18.4.7.3 SDRAM Command Sequence for Burst Accesses

Table 18-29 show the commands that the ESDRAMC will perform for WRAP and INCR accesses from the AHB. The controller will split the transaction when needed in cases that the access cross WRAP boundaries of the SDRAM. The memory configured to 8 beat burst (BL=8).

Unspecified INCR burst accesses will be translated to single accesses to allow the burst to terminate at any length with no additional delays.

The number in the brackets represent the address for READ command and the last Burst Address for BTERM command.

Table 18-29. SDRAM Command Sequence for Burst Accesses

Type	Bus	Address							
		0	4	8	C	10	14	18	1C
WRAP8	16-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
		READ(10)	READ(10)	READ(10)	READ(10)	READ(0)	READ(0)	READ(0)	READ(0)
			READ(0)	READ(0)	READ(0)		READ(10)	READ(10)	READ(10)
			BTERM(0)	BTERM(4)	BTERM(8)		BTERM(10)	BTERM(14)	BTERM(18)
	32-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
INCR8	16-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
		READ(10)	READ(10)	READ(10)	READ(10)	READ(20)	READ(20)	READ(20)	READ(20)
			READ(20)	READ(20)	READ(20)		READ(30)	READ(30)	READ(30)
			BTERM(20)	BTERM(24)	BTERM(28)		BTERM(30)	BTERM(34)	BTERM(38)
	32-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
			READ(20)	READ(20)	READ(20)	READ(20)	READ(20)	READ(20)	READ(20)
			BTERM(20)	BTERM(24)	BTERM(28)	BTERM(2C)	BTERM(30)	BTERM(34)	BTERM(38)
WRAP4	16-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
	32-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
		BTERM(C)	BTERM(10)	READ(0)	READ(0)	READ(0)	READ(10)	READ(10)	READ(10)
				BTERM(4)	BTERM(8)	BTERM(C)	BTERM(10)	BTERM(14)	BTERM(18)
INCR4	16-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
			READ(10)	READ(10)	READ(10)		READ(20)	READ(20)	READ(20)
			BTERM(10)	BTERM(14)	BTERM(18)		BTERM(20)	BTERM(24)	BTERM(28)
	32-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
		BTERM(C)	BTERM(10)	BTERM(14)	BTERM(18)	BTERM(1C)	READ(20)	READ(20)	READ(20)
							BTERM(20)	BTERM(24)	BTERM(28)

18.4.8 Precharge Command Mode

The Precharge Command Mode (SMODE=001) is used during SDRAM/LPDDR device initialization, and to manually deactivate an active bank(s). While in this mode, an access (either read or write) to the SDRAM/LPDDR address space will generate a precharge command cycle. SDRAM/LPDDR address bit A10 determines whether a single bank, or all banks, are precharged by the command. Accessing an address with the SDRAM/LPDDR address A10 low will precharge only the bank selected by the bank addresses, as illustrated in Figure 18-75. Conversely, accesses with A10 high will precharge all banks regardless of the bank address, as illustrated in Figure 18-76. Note that A10 is the SDRAM pin, not the A10 bit ARM address bus. Translation of the SDRAM A10 to the corresponding ARM address is dependent on the memory configuration. The precharge command access is two clocks in length on the ARM, and one cycle to the SDRAM/LPDDR.

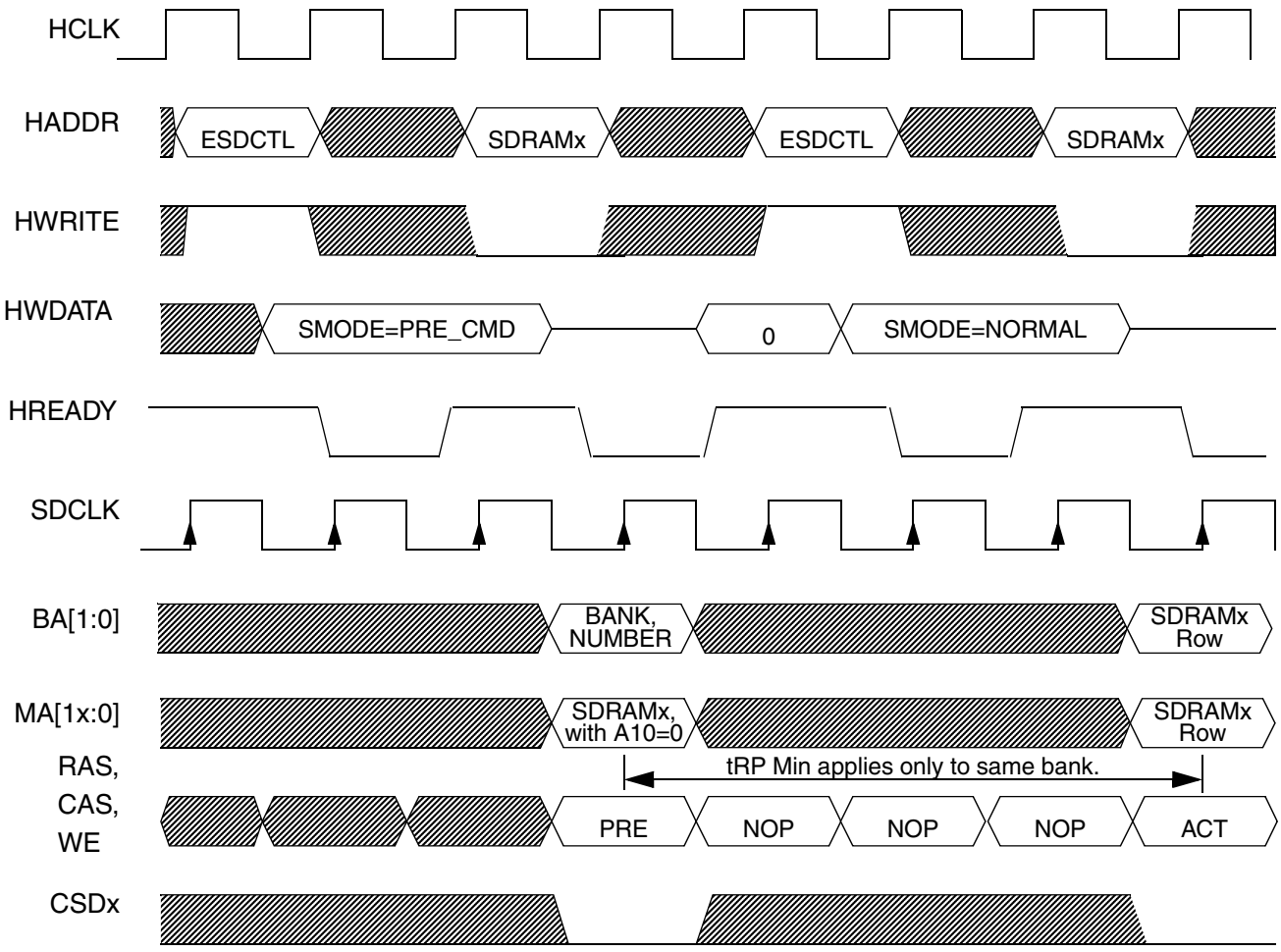


Figure 18-75. Precharge Specific Bank Timing Diagram

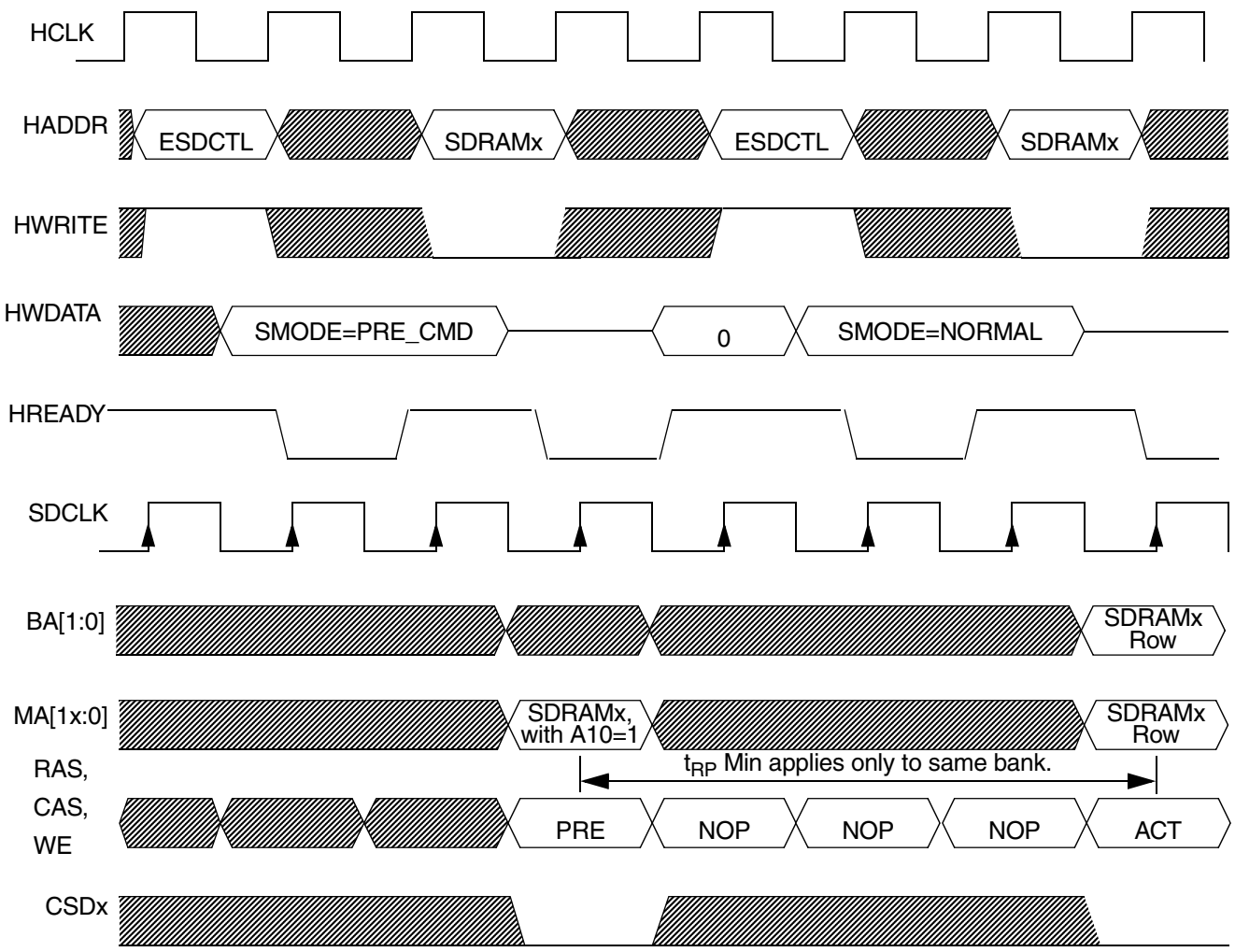


Figure 18-76. Precharge All Banks Timing Diagram

18.4.9 Auto-Refresh Mode

The Auto-Refresh Mode (SMODE=010) is used to manually request SDRAM/LPDDR refresh cycles and is normally used only during device initialization, since the ESDRAMC will automatically generate refresh cycles when properly configured. The auto-refresh command (see Figure 18-77) refreshes all banks in the device, therefore the address supplied during the refresh command need only specify the correct SDRAM/LPDDR device. The lower address lines are a don't care. Either a read or write cycle may be used. If a write is used, the data will be ignored and the external data bus will not be driven. The cycle will be 2 clocks on the ARM and a single clock to the SDRAM/LPDDR device.

The ESDRAMC doesn't guarantees that the SDRAM/LPDDR is in the idle state before the auto-refresh command is given. If one or more rows are active, a precharge-all command should be issued by the software prior to the auto-refresh command.

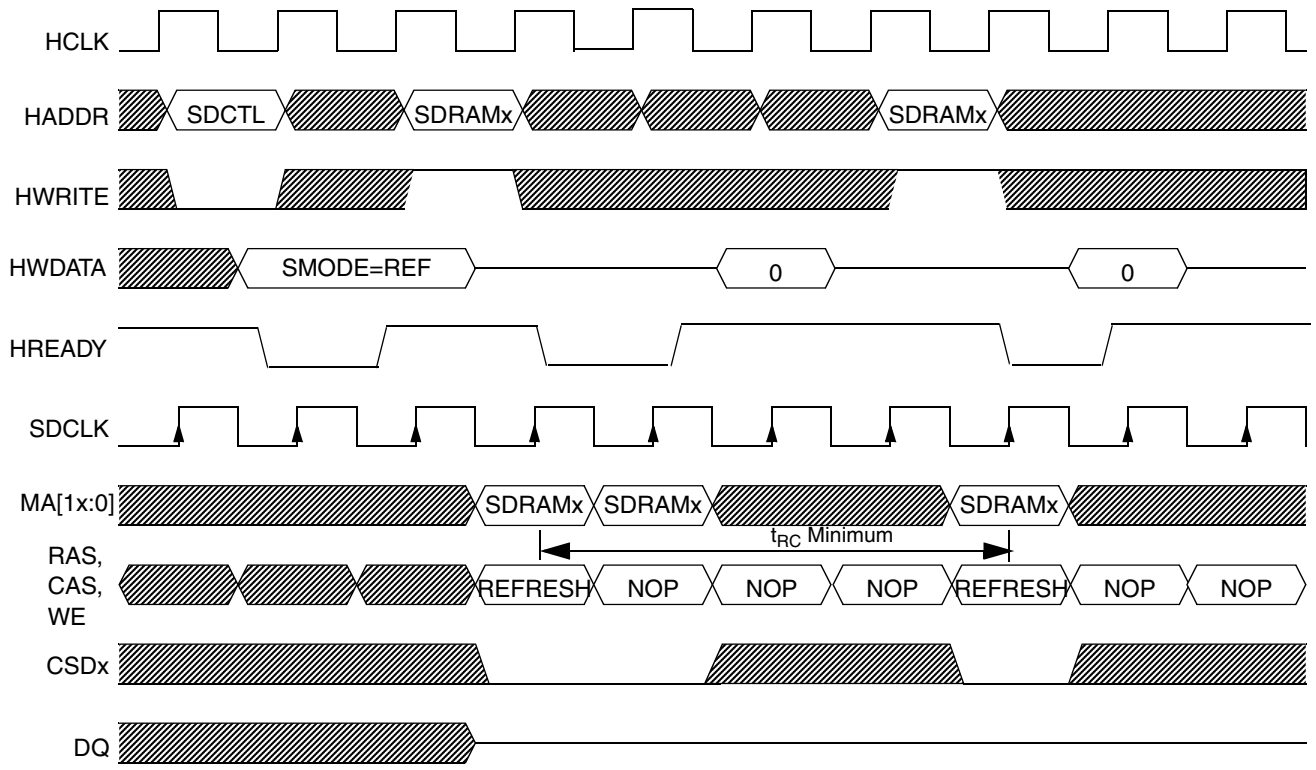


Figure 18-77. Software Initiated Auto-Refresh Timing Diagram

18.4.10 Manual Self Refresh Mode

Manual Self Refresh Mode (SMODE=100) is used to enter SDRAM/LPDDR external device in self refresh low power operating mode during RUN system operating mode. For more details, see Section 18.4.5.2, “Manual Self Refresh Mode for SDRAM/LPDDR Devices.”

18.4.11 Set Mode Register Mode

Set Mode Register mode (SMODE=011) is used to program SDRAM/LPDDR mode register (see Example 18-1 and Section 18.5.4.2, “SDR SDRAM Load Mode Register”). This mode differs from normal SDRAM write cycles because data to be written is transferred across the address bus. Mode Register Reads are not allowed.

After SMODE bits are set to 011, either a read or write cycle may be used to write the external memory device mode register. In both cases (read or write), the ARM data will be ignored and the external data bus will not be driven. The row and bank address signals are used to transfer the data toward the external memory device mode register (see Example 18-1 and Section 18.5.4.2, “SDR SDRAM Load Mode Register”). The cycle will be 2 clocks on the ARM and a single clock to the SDRAM device.

Figure 18-78 illustrates the bus sequence for a mode register set operation. Mode register set commands must be issued while the SDRAM/LPDDR is idle. The Enhanced SDRAM Controller does not guarantee that the SDRAMs have been returned to the idle state before issuing the mode register set command. Software must generate a precharge all sequence before issuing the mode register set command if there is

any possibility that one or more banks could be active. Also note, the row cycle time (t_{RC}) must be met before the mode register set command is issued. Section 18.5.4.2, "SDR SDRAM Load Mode Register" provides a detailed example of the mode register data value calculation and mapping to the ARM address.

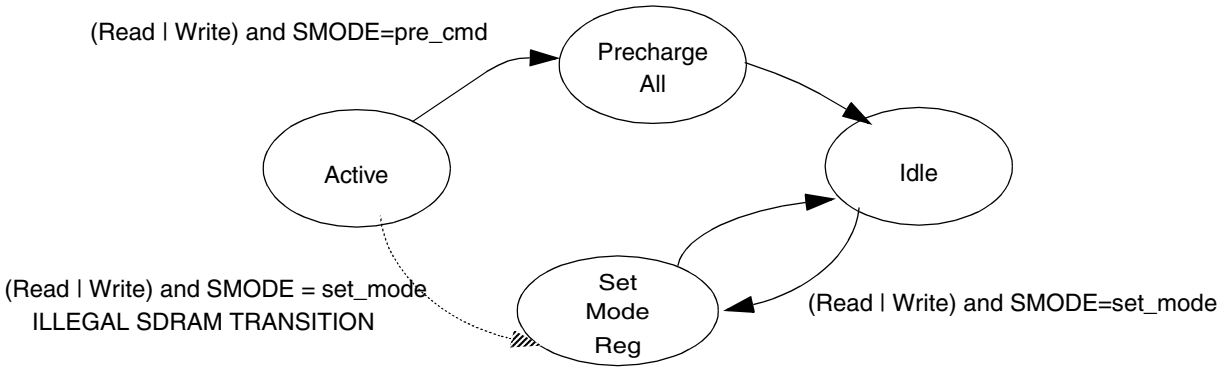


Figure 18-78. Set Mode Register State Diagram

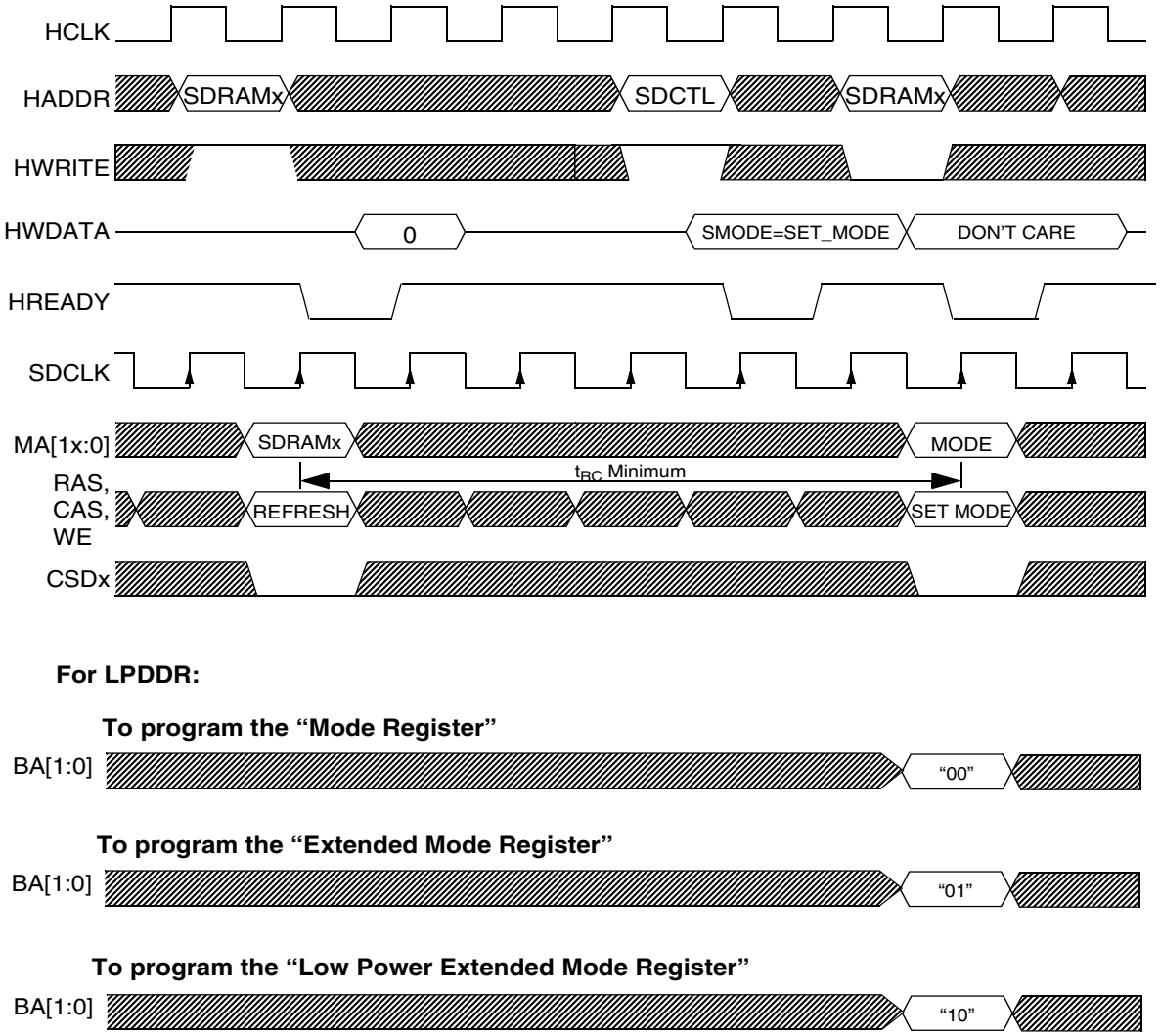


Figure 18-79. SDR and LPDDR Set Mode Register Timing Diagram

18.5 Initialization/Application Information

The following section provides details on selecting compatible SDRAM memories and configuring the controller to work with the memory system.

18.5.1 Memory Device Selection

Many SDRAM/LPDDR memory types are supported by the Enhanced SDRAM Controller. Important characteristics to consider when choosing a memory device are:

- The page comparators expect 4 bank memories. 2 bank devices are not supported. Their use will result in the memory and controller losing synchronization when crossing page boundaries.
- Page (column) addressing must match one of the supported sizes.
- Memory density can be larger or smaller than those directly supported, although some memory may be inaccessible or redundantly mapped. Bank addresses are the most significant addresses and connecting a memory smaller than the selected density will result in one or more banks being inaccessible.
- Controller is designed for memories meeting PC133 timing specifications up to 133 MHz system operation. Use of non-compliant memories require a thorough analysis of all timing parameters.

18.5.2 Configuring Controller for SDRAM Memory Array

Configuration register programming options and controller-memory physical connections provide flexibility to accommodate different memory types and system configurations. Options are broadly grouped into 3 categories:

- Physical Characteristics: Row and column address bus widths and data bus width.
- Timing Parametric: CAS latency, row precharge, cycle delays, refresh rate, etc.
- Functional Features: Clock suspend timer and supervisor/user protection.

Table 18-32–Table 18-42 are provided to assist the designer with the selection of the correct physical parameters for a number of preferred memory configurations. Timing parametric are addressed in the following subsections.

18.5.3 CAS Latency

CAS latency is determined by the operating frequency and access time of the memories. For a 133 MHz system clock frequency and PC133 compliant memories, the CAS latency will generally be programmed to 3 clocks, although the memory specifications should always be consulted to confirm this value. CAS latency must be programmed in two places: the chip select Control Register and the device Mode Register. See Table 18-17 for a description of the control register encoding, and Section 18.4.11, “Set Mode Register Mode” for the details on programming the SDRAM mode register.

18.5.4 SDRAM/LPDDR Initialization Sequence

Prior to normal operation (read/write accesses), external memory device must be initialized. The following paragraphs provide detailed information covering device initialization. Register definition, command descriptions and device operation information has been thoroughly described throughout the chapter.

18.5.4.1 SDRAM Initialization

SDR and LPDDR SDRAMs must be powered up and initialized in a predefined manner. Operational procedures other than those specified by SDRAM manufacture specification may results in undefined operation.

18.5.4.1.1 SDR SDRAM Initialization

Once power is applied to the device and the clock is stable, the SDRAM requires a 200µs delay prior to issuing any command other than a COMMAND INHIBIT or a NOP. Starting at some point during this 200µs period and continuing at least through the end of this period, COMMAND INHIBIT or NOP commands should be applied.

Once the 200µs delay has been satisfied with at least one COMMAND INHIBIT or NOP command have been applied (the SDRAM_RDY status bit will be asserted), a PRECHARGE command should be applied. All banks must then be precharged, thereby placing the device in the all banks idle state.

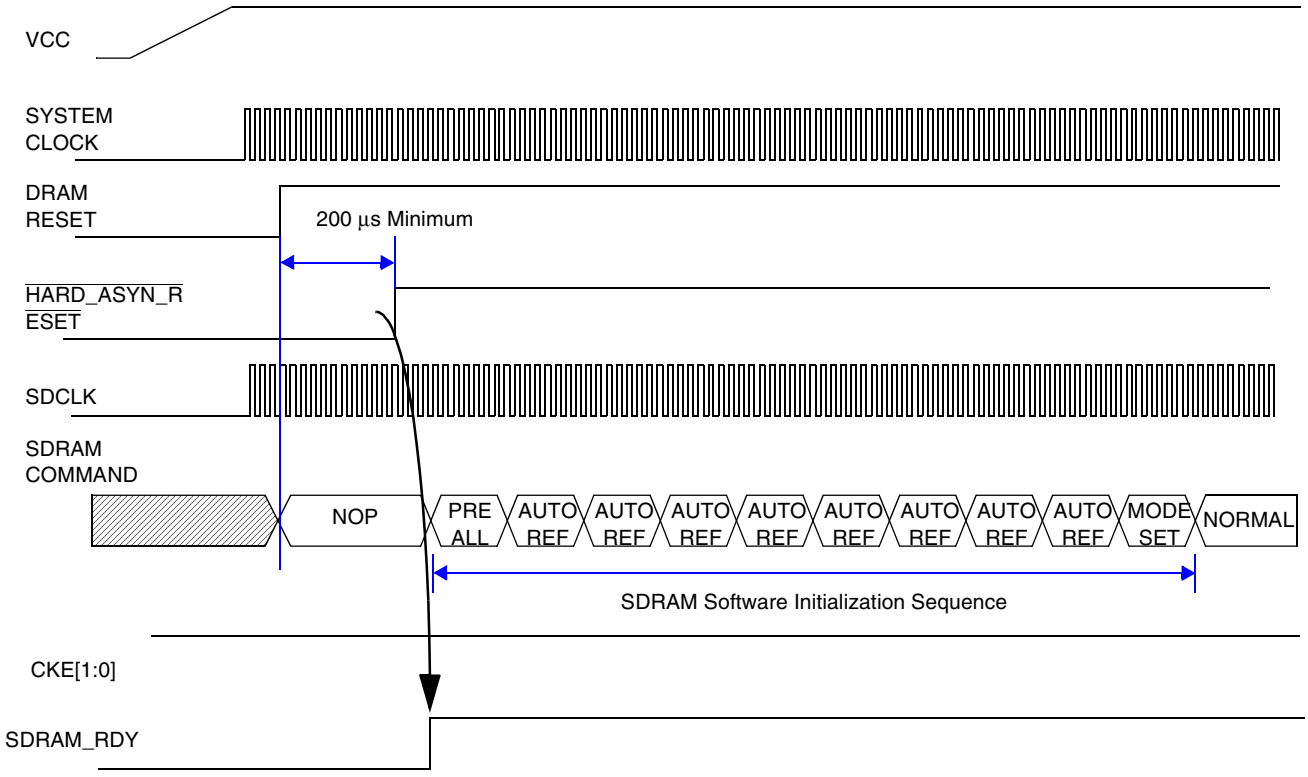


Figure 18-80. SDR SDRAM Initialization and Load Mode Register Sequence

Once in idle state, several (manufacture dependent) AUTO REFRESH cycles must be performed. After the AUTO REFRESH cycles are complete, the SDRAM is ready for Mode Register programming. Because the Mode Register will power up in an unknown state, it should be loaded prior to applying any operational command. Figure 18-80 illustrates a SDRAM initialization routine with 8 AUTO REFRESH cycles. Example 18-1 shows an initialization SDRAM example code.

It is crucial that the dual parameters (those parameters that are defined both in the SDRAM device register and in ESDRAMC registers, like CAS latency, burst length, etc.) to be identical for proper operation of both SDRAM memory and Enhanced SDRAM Controller.

Example 18-1. INIT_SDRAM Example Code (ARM Assembler)

```
init_sdram:
                ldr     r2, =ESD_ESDCTL0       // base address of registers
                ldr     r3, =PRE_ALL_CMD       // SMODE=001
                str     r3,(r2,#0x0)           // put CSD0 in precharge command mode
                ldr     r4, =SDRAM_CSD0        // CSD0 precharge address (A10=1)
                str     r1,(r4,#0x0)           // precharge CSD0 all banks
                ldr     r3, =AUTO_REF_CMD      // SMODE=010
                str     r3,(r2,#0x0)           // put array 0 in auto-refresh mode
                ldr     r4, =SDRAM_CSD0_BASE   // CSD0 base address
                ldr     r6,=0x7                // load loop counter
0:              ldr     r5,(r4,#0x0)           // run auto-refresh cycle to array 0
                subs    r6,r6,#1               // decrease counter value
                bne     0b
                ldr     r3, =SET_MODE_REG_CMD  // SMODE=011
                str     r3,(r2,#0x0)           // setup CSD0 for mode register write
                ldr     r3, =MODE_REG_VAL0     // array 0 mode register value
                ldrb    r5,(r3,#0x0)           // New mode register value on address bus

                ldr     r3, =NORMAL_MODE       // SMODE=000
                str     r3,(r2,#0x0)           // setup CSD0 for normal operation



ESD_ESDCTL0             .long   0xXXXX_XXXX    // system/external device dependent data
SDRAM_CSD0:             .long   0xXXXX_XXXX    // system/external device dependent data
SDRAM_CSD0_BASE:        .long   0xXXXX_XXXX    // system/external device dependent data
PRE_ALL_CMD             .long   0xXXXX_XXXX    // system/external device dependent data (SMODE=001)
AUTO_REF_CMD            .long   0xXXXX_XXXX    // system/external device dependent data (SMODE=010)
SET_MODE_REG_CMD        .long   0xXXXX_XXXX    // system/external device dependent data (SMODE=011)
MODE_REG_VAL0           .long   0xXXXX_XXXX    // system/external device dependent data
NORMAL_MODE             .long   0xXXXX_XXXX    // system/external device dependent data (SMODE=000)
```

NOTE

To do LOAD MODE REGISTER, address starts from bit 0, so LRDB should be used.

18.5.4.1.2 LPDDR SDRAM Initialization

DDR Mobile SDRAM must be powered up and initialized in a predefined manner. Operational procedures other than those specified may result in undefined operation. Power must first be applied to VDD and VDDQ according to the LPDDR SDRAM manufacture data sheet. Clock enable mast be driven through the SDRAM controller registers to cs0 and/or cs1.

After all power supply voltages are stable, and the clock is stable, the DDR Mobile-SDRAM requires a 200µs delay prior to applying a command other than DESELECT or NOP.

CKE is driven HIGH by the SDRAM controller on the first edge of the clock.

Once the 200us delay has been satisfied, the following command sequence shall be applied using the SDRAM controller registers:

1. A DESELECT or NOP command.

2. A PRECHARGE ALL command should then be applied, placing the device in the all banks idle state.

3. Once in the idle state, two AUTO REFRESH cycles must be performed (tRFC must be satisfied).

4. Two MODE REGISTER SET commands for the Mode Register and Extended Mode Register.

Following these cycles, the DDR Mobile-SDRAM is ready for normal operation.

NOTE

A WRITE access should be performed before the first READ access to the LPDDR (in order to assure that a 0 value will be driven on the DQS pins and held by the keeper of the DDR pads).

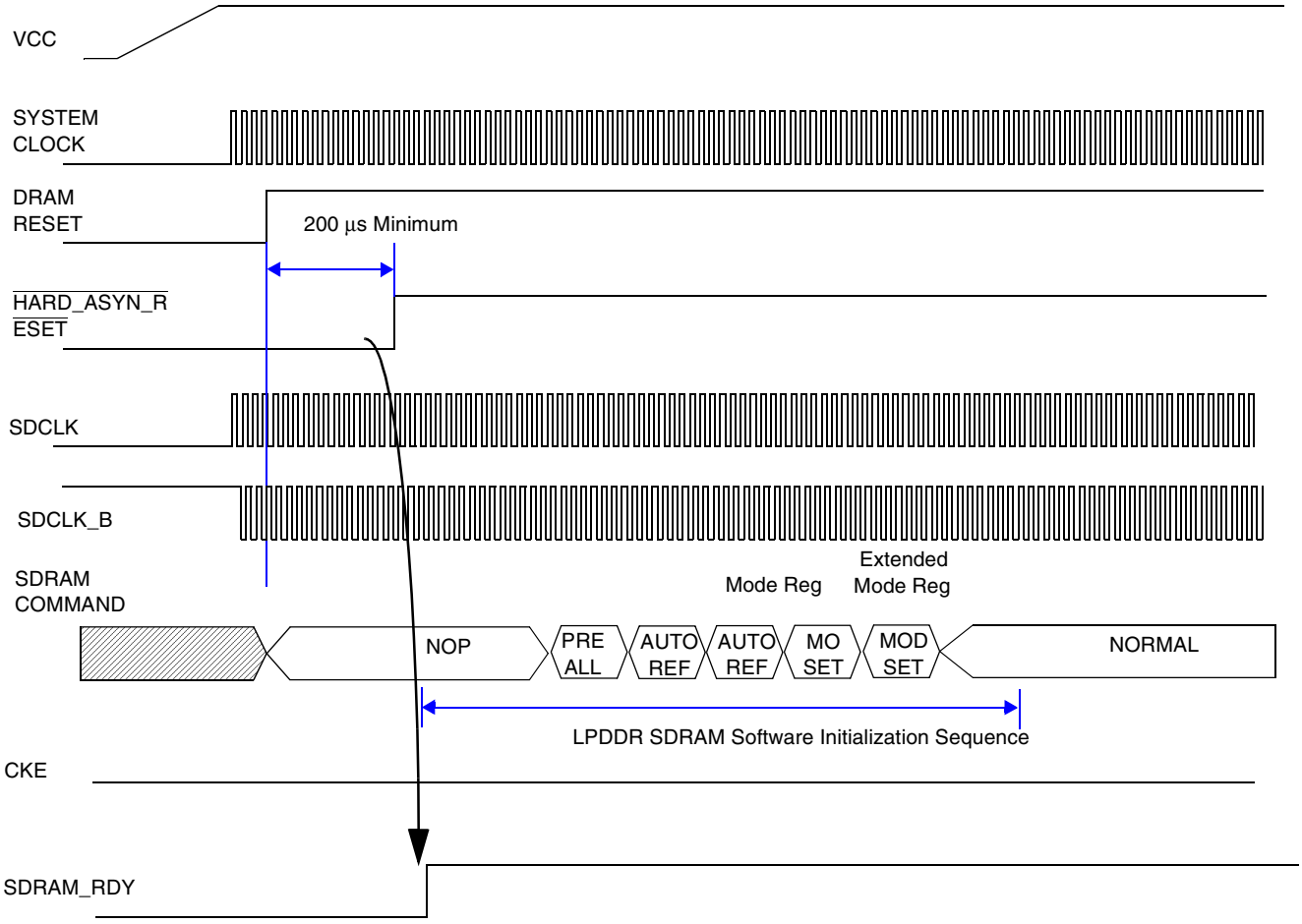


Figure 18-81. Simplified LPDDR SDRAM Initialization and Load Mode Register Sequence

18.5.4.2 SDR SDRAM Load Mode Register

The mode register is used to set the SDRAM operating characteristics including CAS latency, burst length, burst mode, and write data length. The settings depend on system characteristics including the operating frequency, memory device type, burst buffer/cache line length, and bus width. Operating characteristics vary by device type, so the data sheet must be consulted to determine the actual value to be written. In order to demonstrate the procedure, the following system characteristics will be used:

- Micron MT48LC4M32B2 128Mb (1M x 32 x 4 banks) SDR SDRAMs
- 133 MHz System Clock Frequency
- Sequential burst, burst length of 8
- Single word writes (for example, no bursting on writes)

Figure 18-82 shows the Mode Register bit assignments for the micron 128 Mb SDRAM and the bit field descriptions are listed in Table 18-30.

128 Mb SDRAM Mode Register

	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
TYPE	Reserved*		WB	Op Mode		CAS Latency			BT	Burst Length		

Figure 18-82. 128 Mbit SDR SDRAM Mode Register

Table 18-30. SDRAM Mode Register Description

Name	Description
A11-A10	Reserved
A9 Write Burst	Write Burst Mode (WB). Selects between burst writes and single location writes. 0 Programmed Burst Length 1 Single Location Access[1]
A8–A7 Op Mode	Operating Mode. Defines operating modes. 00 Standard operation All other states reserved
A6–A4 CAS Latency	CAS Latency (CL). Sets latency between column address and data 000 Reserved 001 1 clock[1] 010 2 clocks 011 3 clocks 1xx Reserved

Table 18-30. SDRAM Mode Register Description (continued)

Name	Description
A3 Burst Type	Burst Type (BT). Selects burst type 0 Interleave 1 Sequential
A2–A0 Burst Length	Burst Length (BL). A 16-bit wide SDRAM requires a burst length of eight because the four 32-bit line fill cycles will be decomposed into eight 16-bit accesses. 000 1[1] 001 2[1] 010 4[2] 011 8 111 Full Page 10x Reserved 1x0 Reserved

1 Not supported by the ESDCTL.

2 Not supported by the ESDCTL for 16-bit external memory device.

For this example:

- Sequential burst (BT = 0)
- Burst length of 8 (BL = 011)
- Programmed burst length (during writes) (WB = 0)
- 3 Clock Latency (CAS Latency= 011)

Once the mode register value has been determined, it must be converted to an address. The mode register is written via the address bus and the memory data sheet will specify the SDRAM address bits on which to place the data. The Enhanced SDRAM Controller drives the LSB address bits to the pins, so the memory density and bus width don't need to be taken into account during the conversion. Table 18-31 provides an example conversion using the same system characteristics used in the previous example.

Table 18-31. Example Address Calculation for Mode Register

Mode Register	0	0	WB	0	0	CAS LATENCY			BT	BL		
Program Value	0	0	0	0	0	0	1	1	0	0	1	1
SDRAM Pin	MA11	MA10	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0
ARM Address	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

18.5.4.3 SDRAM Memory Configuration Examples

15 different SDRAM (SDR and LPDDR) configurations are demonstrated. These examples are 64 Mb, 128 Mb, and 256 Mb SDRAM memories in single x16, dual x16, and single x32 configurations. All single-configuration 16-bit examples are shown connected to the lower half of the data bus. Alternatively, the memory can be connected to the upper half of the data bus by swapping the data connections to D

[31:16] and the data qualifier mask connections to DQM3 and DQM2. In this case, it will be necessary to program the DSIZ field in the Control Register to a value of 0 (configurations shown require a value of 1).

18.5.4.3.1 Single 64Mb (4Mx16) SDRAM Configuration

Table 18-32. Single 4Mx16 Control Register Value

Control Field	Value
Density	8 MB
Page Size	512
ROW	12
COL	8
DSIZ	16 (D [15:0])
SREFR	2

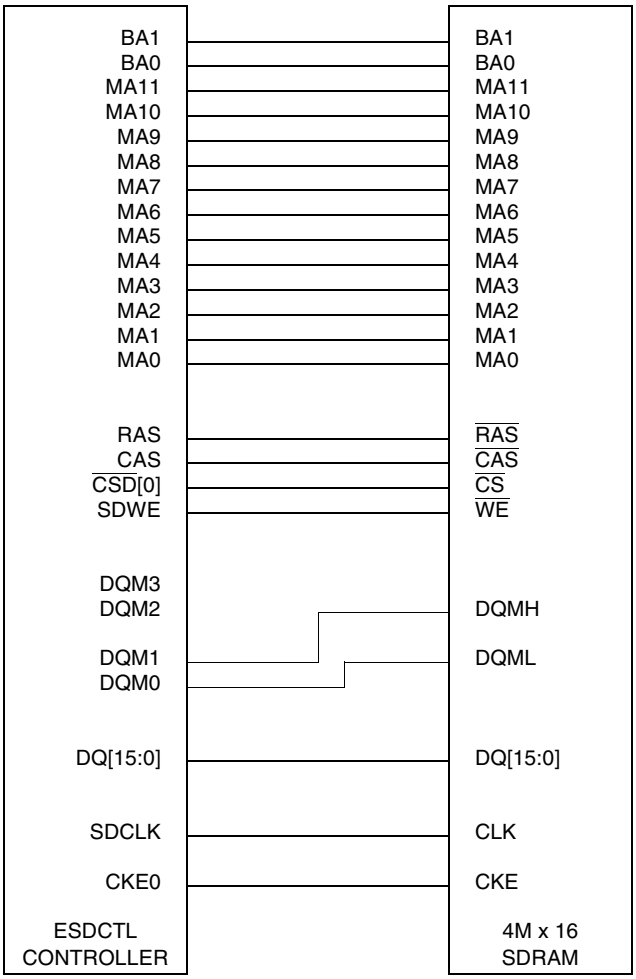


Figure 18-83. Single 64 Mb (4M x 16) SDRAM Connection Diagram

18.5.4.3.2 Single 128 Mb (8MBx16) SDRAM Configuration

Table 18-33. Single 8MBx16 Control Register Value

Control Field	Value
Density	16 Mb
Page Size	1024
ROW	12
COL	9
DSIZ	16 (D [15:0])
SREFR	4

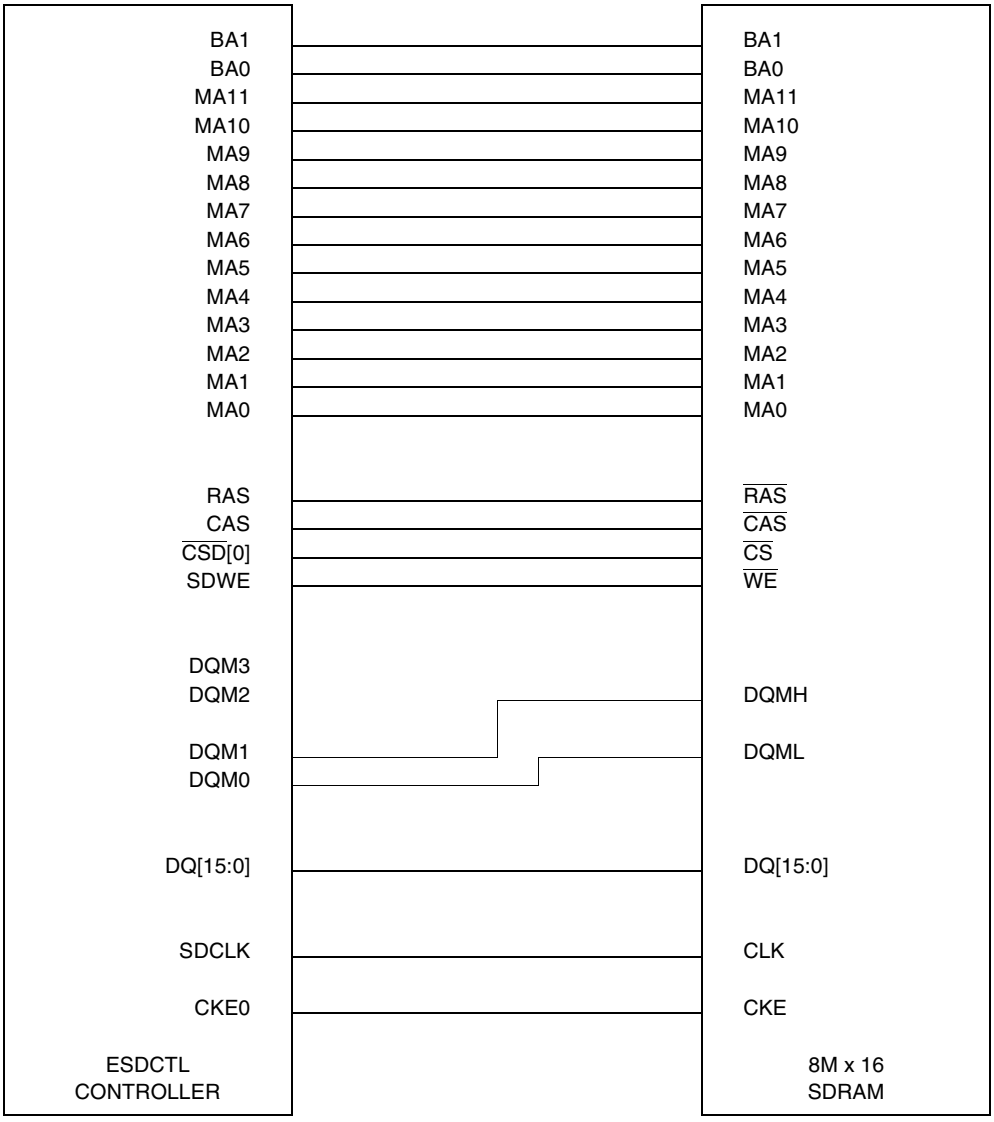


Figure 18-84. Single 128 Mb (8M x 16) SDRAM Connection Diagram

18.5.4.3.3 Single 256 Mb (16MBx16) SDRAM Configuration

Table 18-34. Single 16MBx16 Control Register Value

Control Field	Value
Density	32 Mb
Page Size	1024
ROW	13
COL	9
DSIZ	16 (D [15:0])
SREFR	4

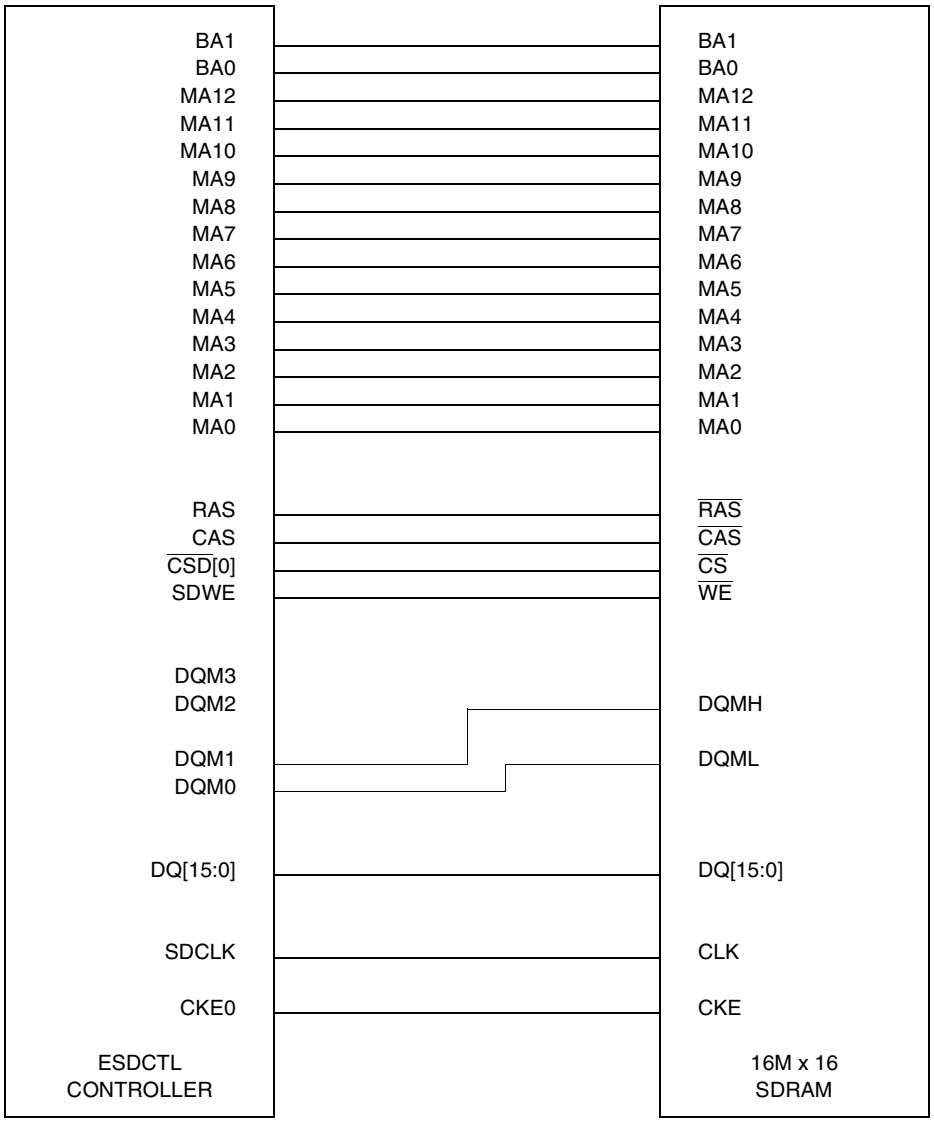


Figure 18-85. Single 256 Mb (16M x 16) Connection Diagram

18.5.4.3.4 Single 512 Mb (32MBx16) SDRAM Configuration

Table 18-35. Single 32MBx16 Control Register Value

Control Field	Value
Density	32 Mb
Page Size	1024
ROW	13
COL	10
DSIZ	16 (D [15:0])
SREFR	4

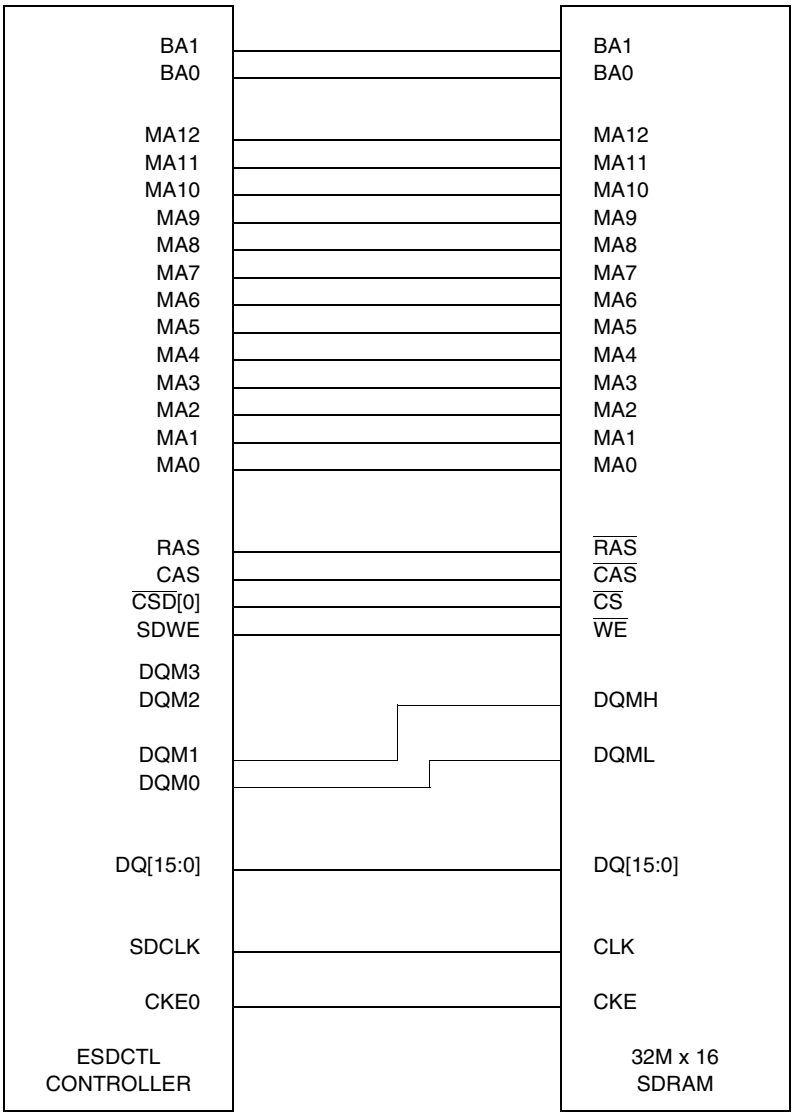


Figure 18-86. Single 512 Mb (32M x 16) SDRAM Connection Diagram

18.5.4.3.5 Single 1-Gb (64MBx16) SDRAM Configuration

Table 18-36. Single 64MBx16 Control Register Value

Control Field	Value
Density	64 Mb
Page Size	2048
ROW	14
COL	10
DSIZ	16 (D [15:0])
SREFR	8

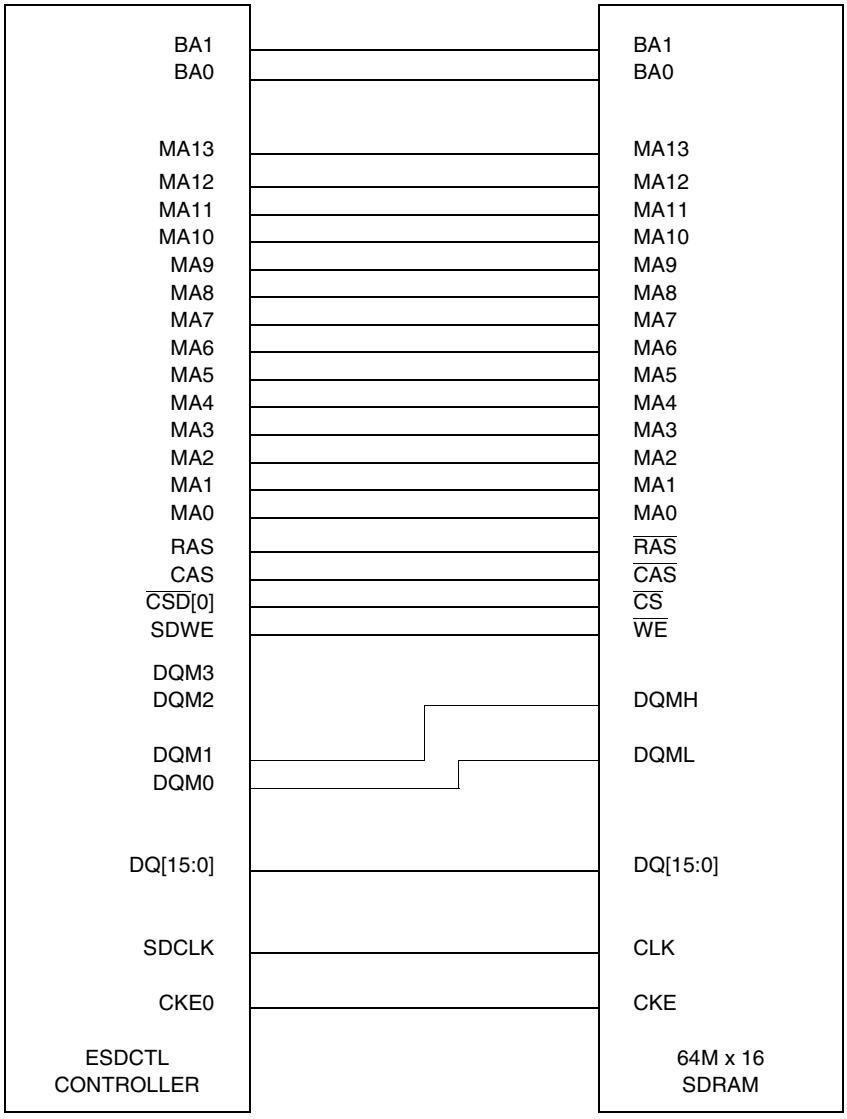


Figure 18-87. Single 1-Gb (64M x 16) SDRAM Connection Diagram

18.5.4.3.6 Dual 64 Mb (4MBx16) SDRAM Configuration

Table 18-37. Dual 4MBx16 Control Register Value

Control Field	Value
Density	16 Mb
Page Size	1024
ROW	12
COL	8
DSIZ	32 (D [31:0])
SREFR	4

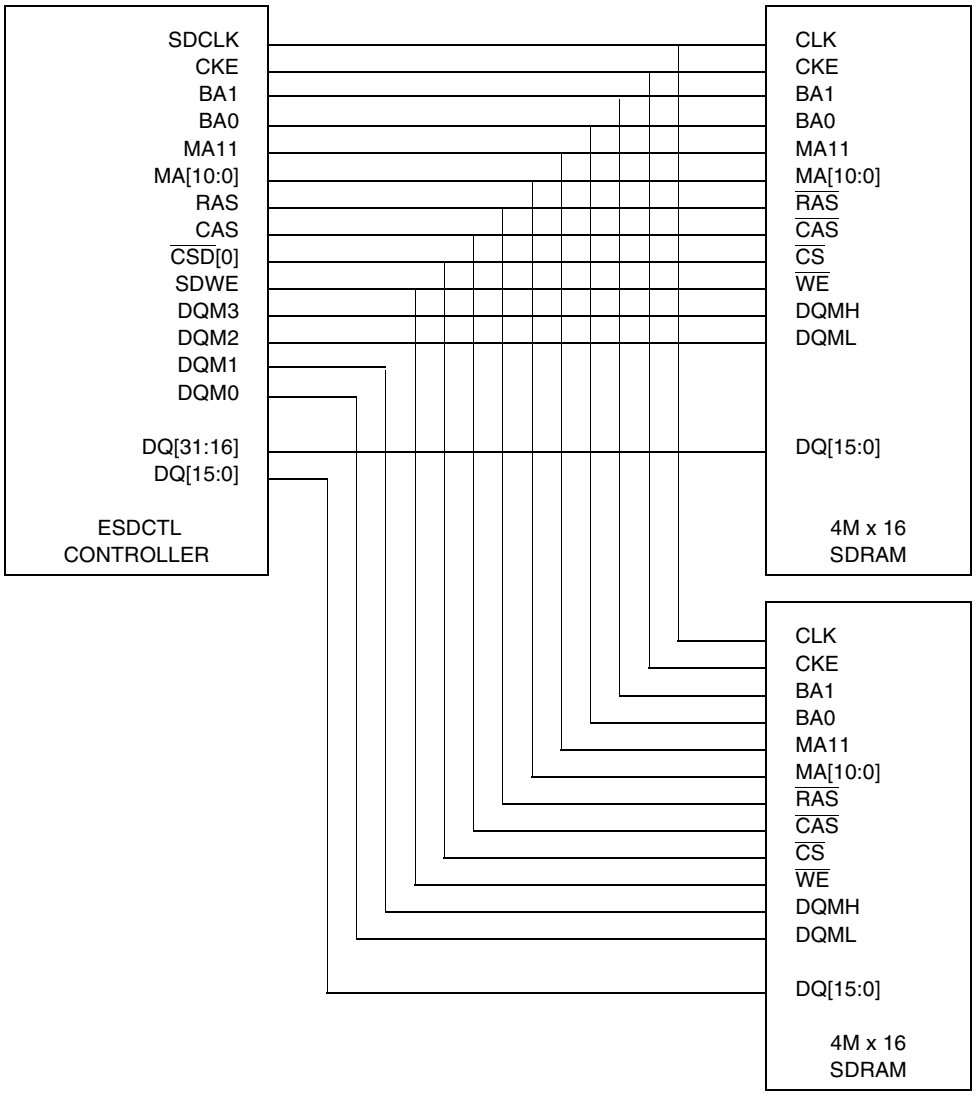


Figure 18-88. Dual 64 Mb (4M x 16 x 2) SDRAM Connection Diagram

18.5.4.3.7 Dual 128 Mb (8MBx16) SDRAM Configuration

Table 18-38. Dual 8MBx16 Control Register Value

Control Field	Value
Density	32 Mb
Page Size	2048
ROW	12
COL	9
DSIZ	32 (D [31:0])
SREFR	4

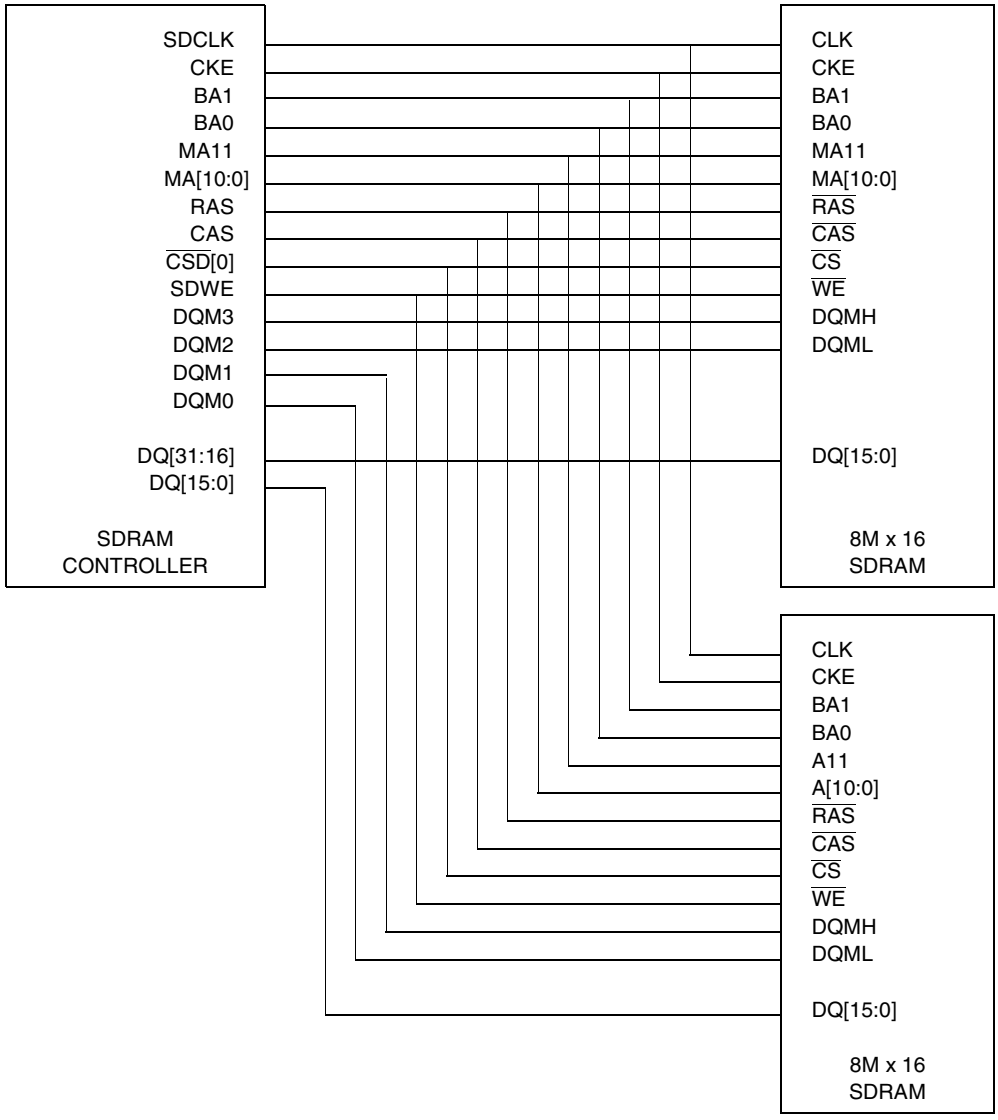


Figure 18-89. Dual 128 Mb (8M x 16 x 2) SDRAM Connection Diagram

18.5.4.3.8 Dual 256 Mb (16MBx16) SDRAM Configuration

Table 18-39. Dual 16MBx16 Control Register Value

Control Field	Value
Page Size	2048
ROW	13
COL	9
DSIZ	32 (D [31:0])
SREFR	4

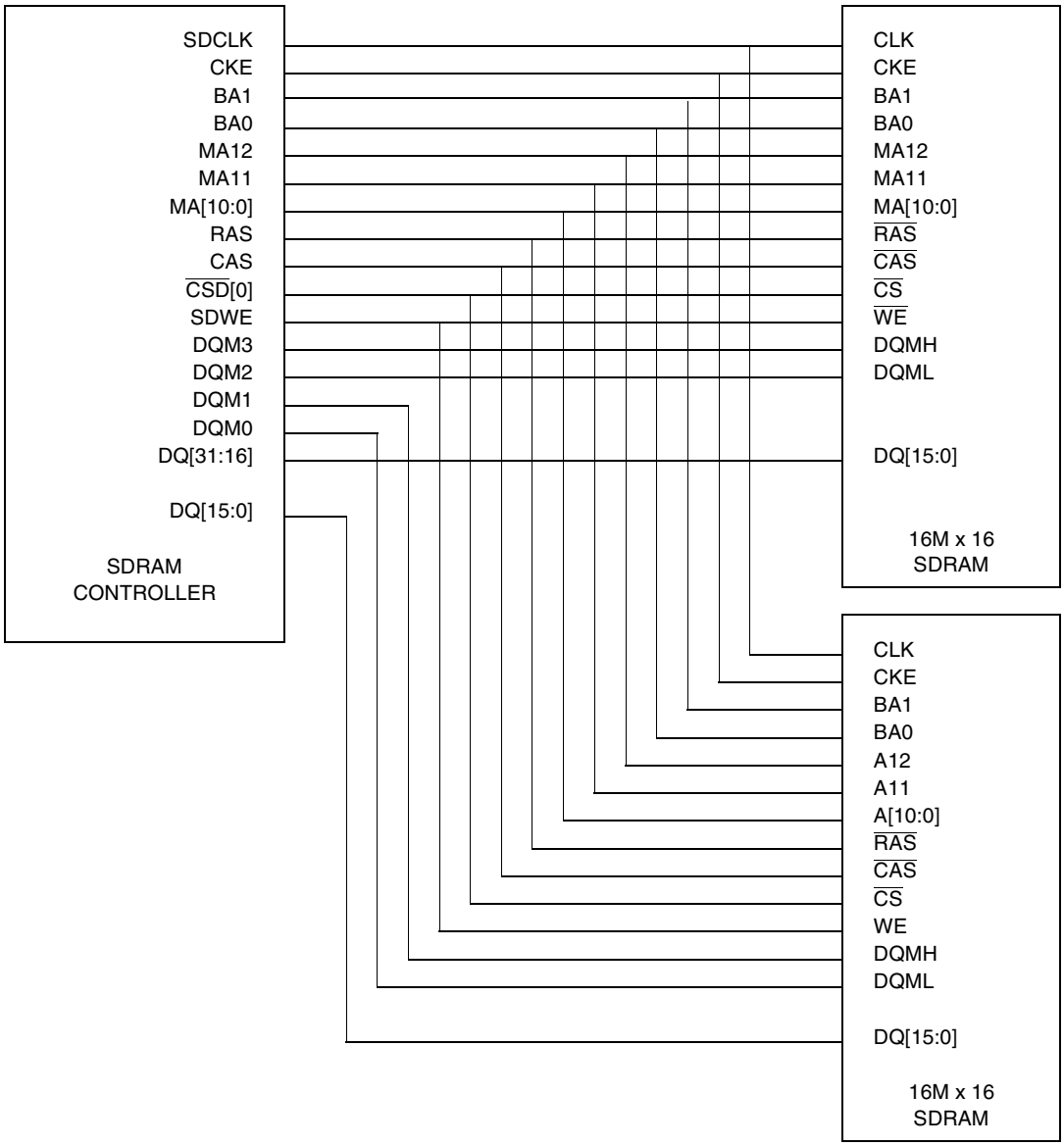


Figure 18-90. Dual 256-Mbyte (16M x 16 x 2) SDRAM Connection Diagram

18.5.4.3.9 Single 64-Mbyte (2MBx32) SDRAM Configuration

Table 18-40. Single 2MBx32 Control Register Value

Control Field	Value
Density	2 Mb
Page Size	1024
ROW	11
COL	8
DSIZ	32 (D [31:0])
SREFR	1

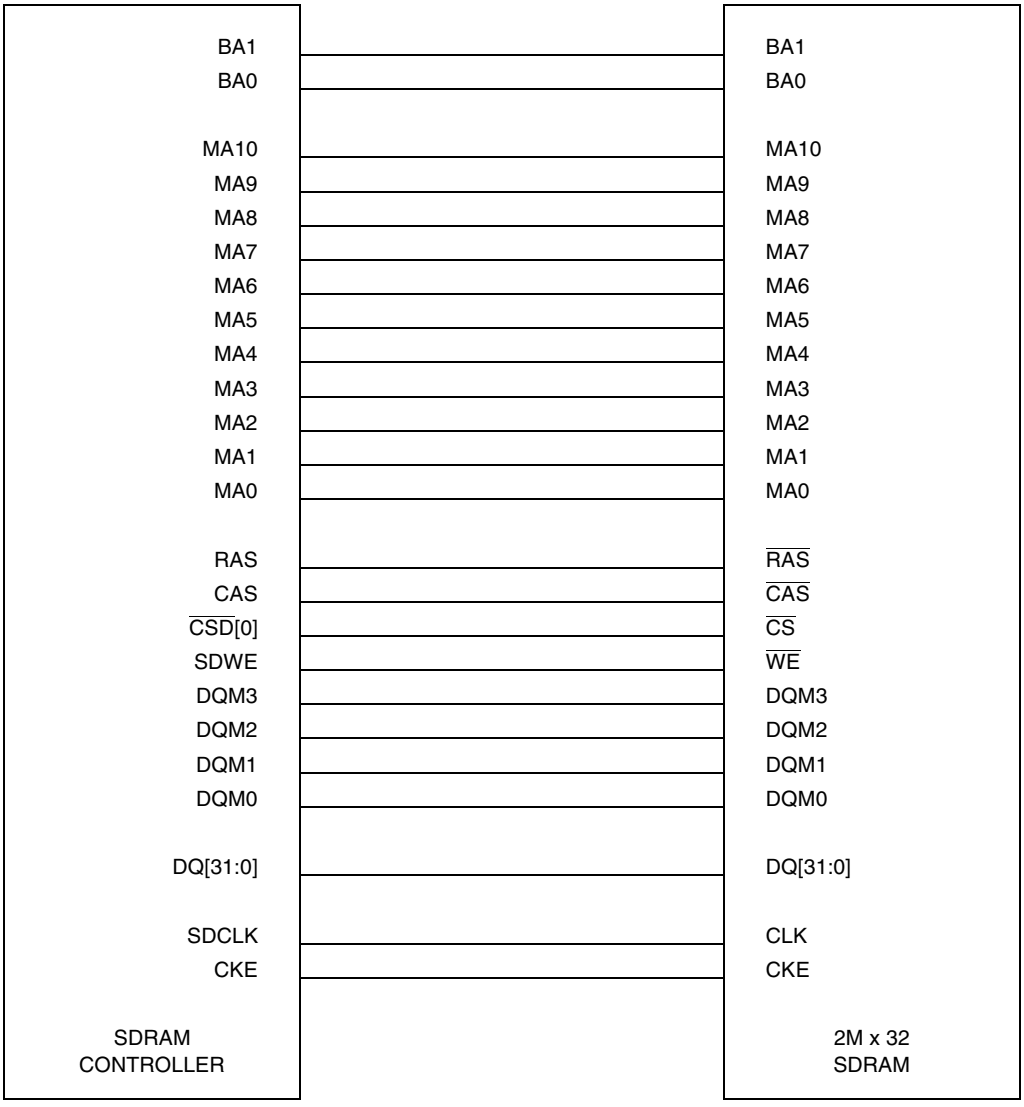


Figure 18-91. Single 64-Mbyte (2MBx32) SDRAM Connection Diagram

18.5.4.3.10 Single 128-Mbyte (4MBx32) SDRAM Configuration

Table 18-41. Single 4MBx32 Control Register Value

Control Field	Value
Density	4 Mb
Page Size	1024
ROW	12
COL	8
DSIZ	32 (D [31:0])
SREFR	2

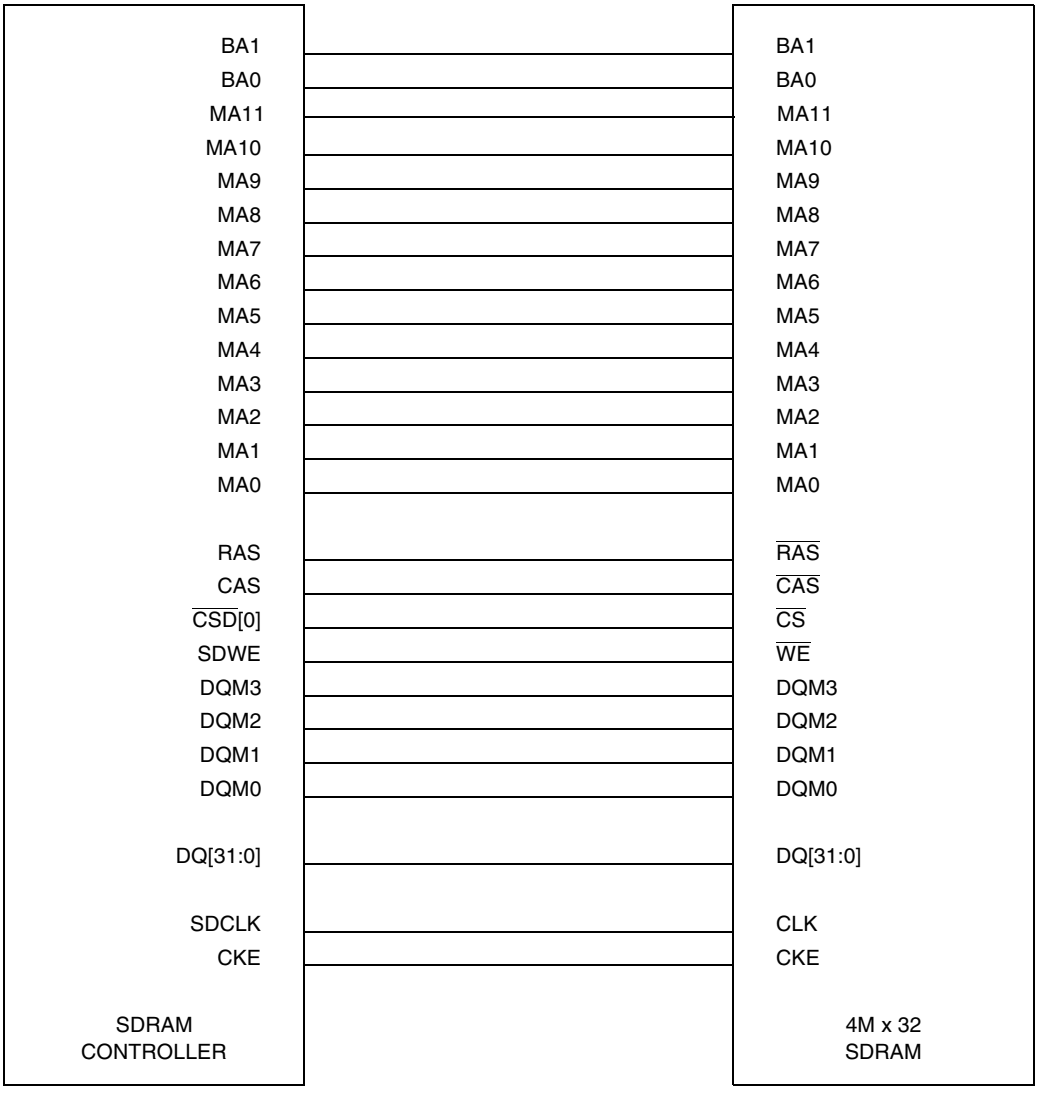


Figure 18-92. Single 128-Mb (4MBx32) SDRAM Connection Diagram

18.5.4.3.11 Single 256-Mb (8MBx32) SDRAM Configuration

Table 18-42. Single 8MBx32 Control Register Value

Control Field	Value
Density	8 Mb
Page Size	1024
ROW	13
COL	8
DSIZ	32 (D [31:0])
SREFR	4

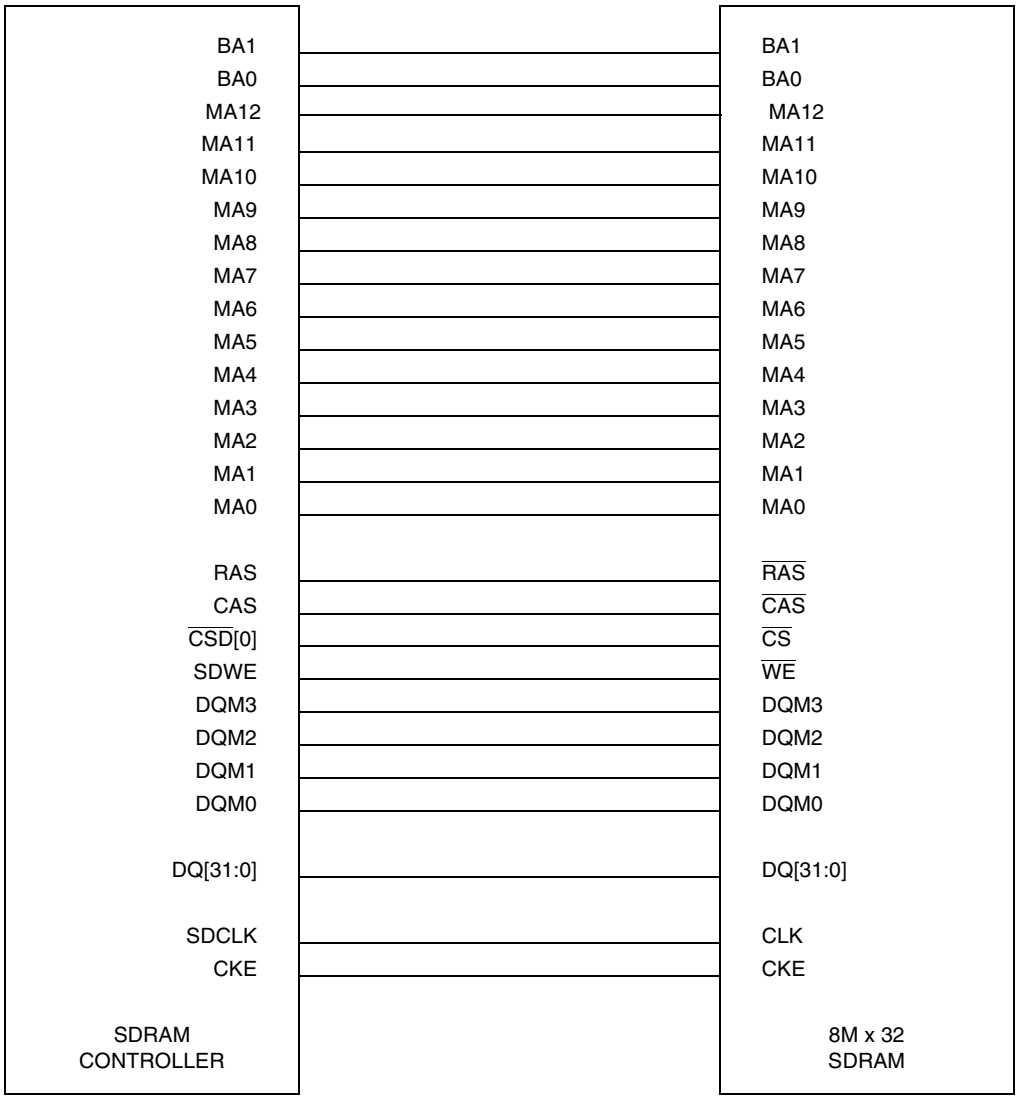


Figure 18-93. Single 256-MB (8MBx32) SDRAM Connection Diagram

18.5.4.3.12 Single 512-Mb (16MBx32) SDRAM Configuration

Table 18-43. Single 16MBx32 Control Register Value

Control Field	Value
Density	16 Mb
Page Size	1024
ROW	13
COL	9
DSIZ	32 (D [31:0])
SREFR	4

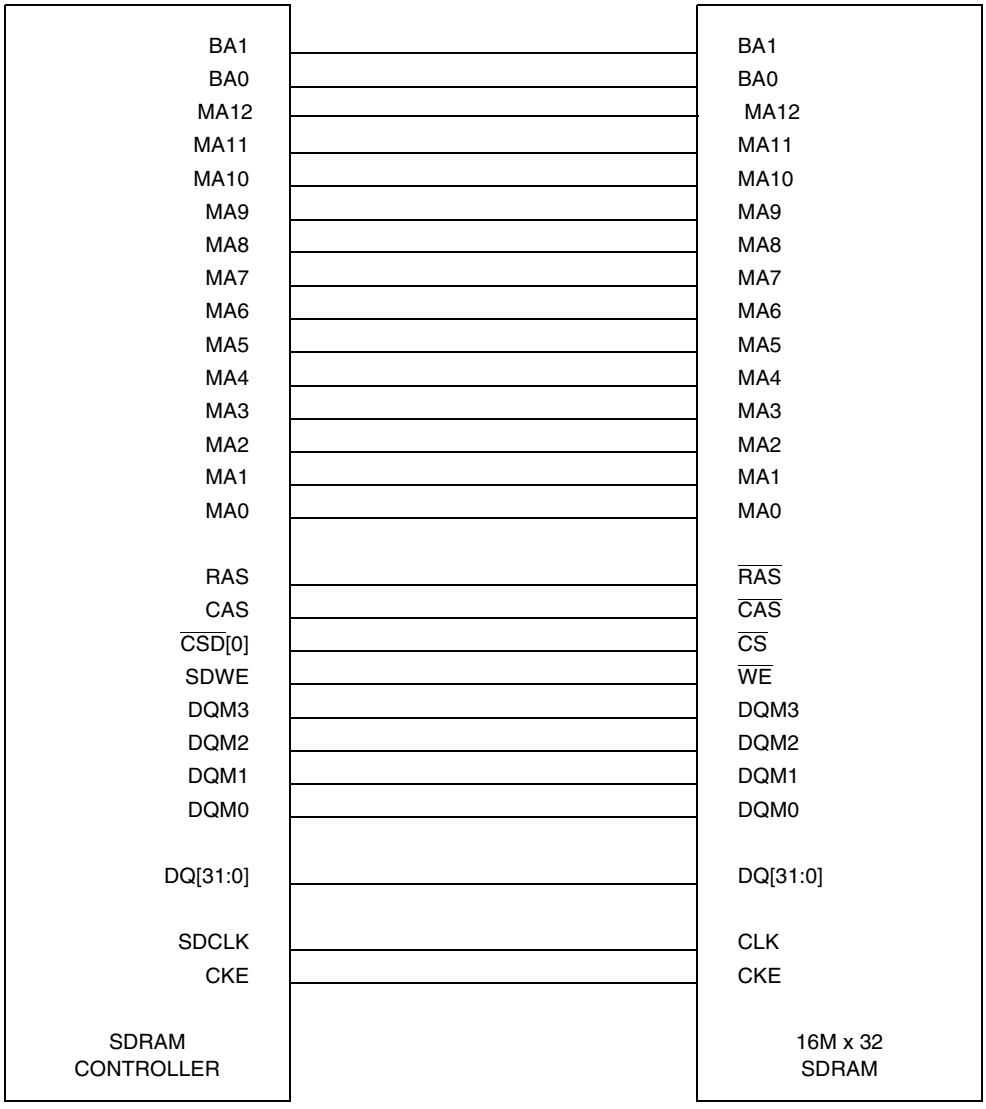


Figure 18-94. Single 512-Mb (16MBx32) SDRAM Connection Diagram

18.5.4.3.13 Single 1-Gb (32Mx32) SDRAM Configuration

Table 18-44. Single 32MBx32 Control Register Value

Control Field	Value
Density	32 Mb
Page Size	1024
ROW	14
COL	9
DSIZ	32 (D [31:0])
SREFR	8

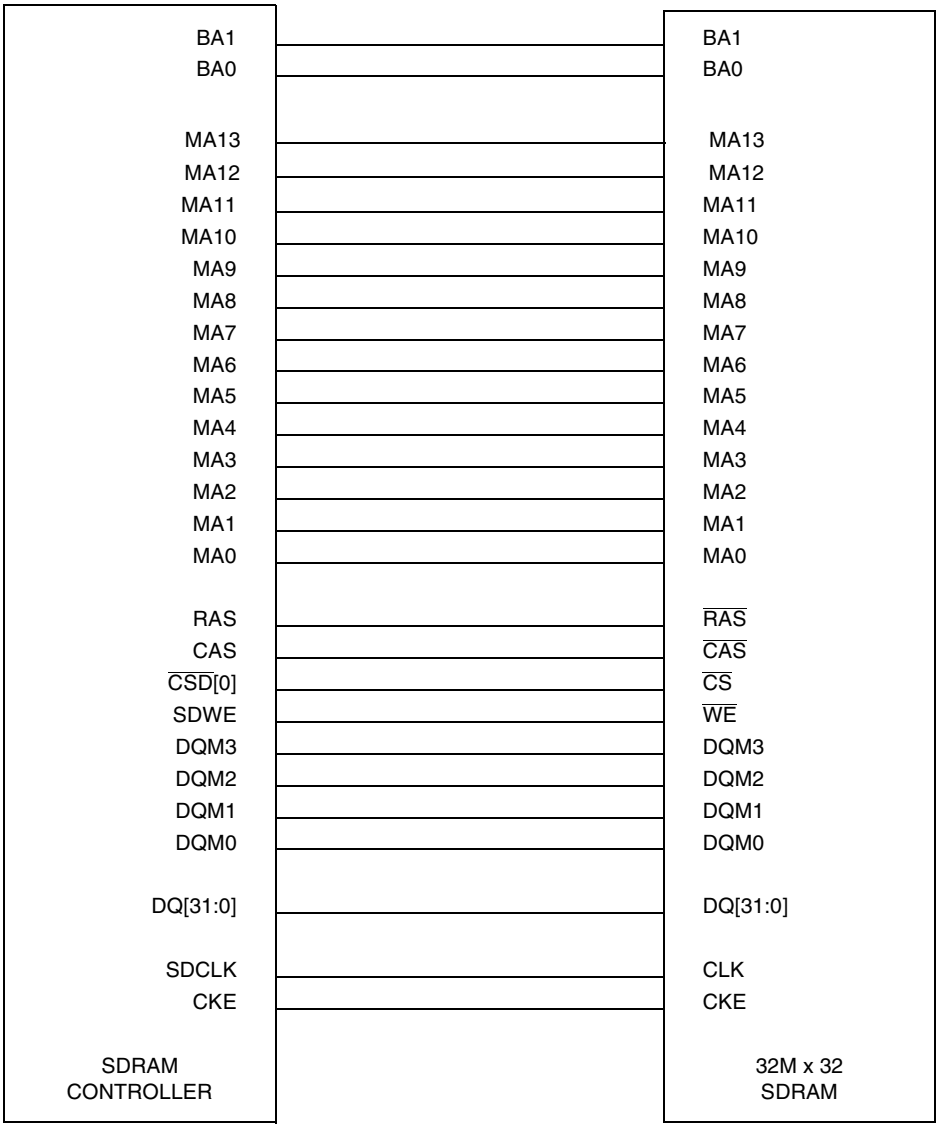


Figure 18-95. Single 1-Gb (32MBx32) SDRAM Connection Diagram

18.5.4.3.14 Single 2-Gb (64MBx32) SDRAM Configuration

Table 18-45. Single 64MBx32 Control Register Value

Control Field	Value
Density	64 Mb
Page Size	1024
ROW	14
COL	10
DSIZ	32 (D [31:0])
SREFR	8

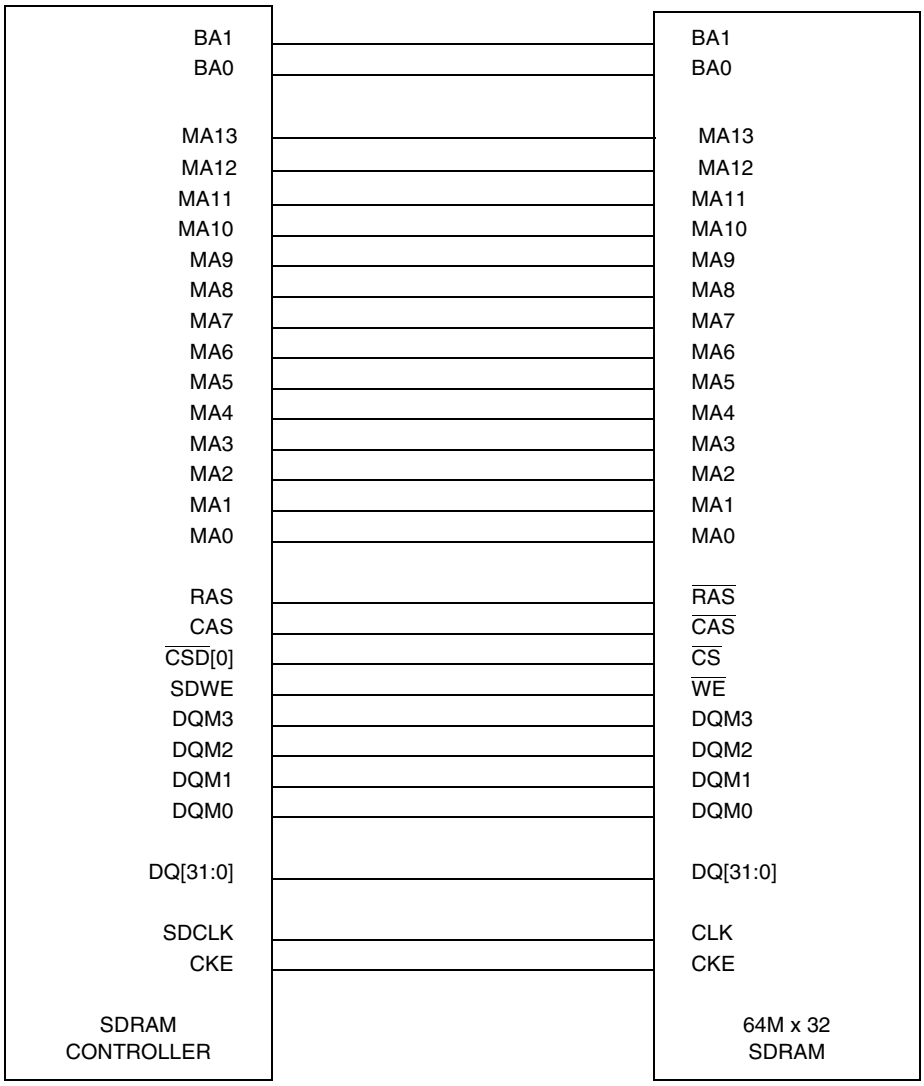


Figure 18-96. Single 2-Gb (64MBx32) SDRAM Connection Diagram

18.5.4.3.15 Single 512-Mb (16MBx32) Mobile DDR SDRAM Configuration

Table 18-46. Single 16MBx32 Control Register Value

Control Field	Value
Density	16 Mb
Page Size	1024
ROW	13
COL	9
DSIZ	32 (D [31:0])
SREFR	4

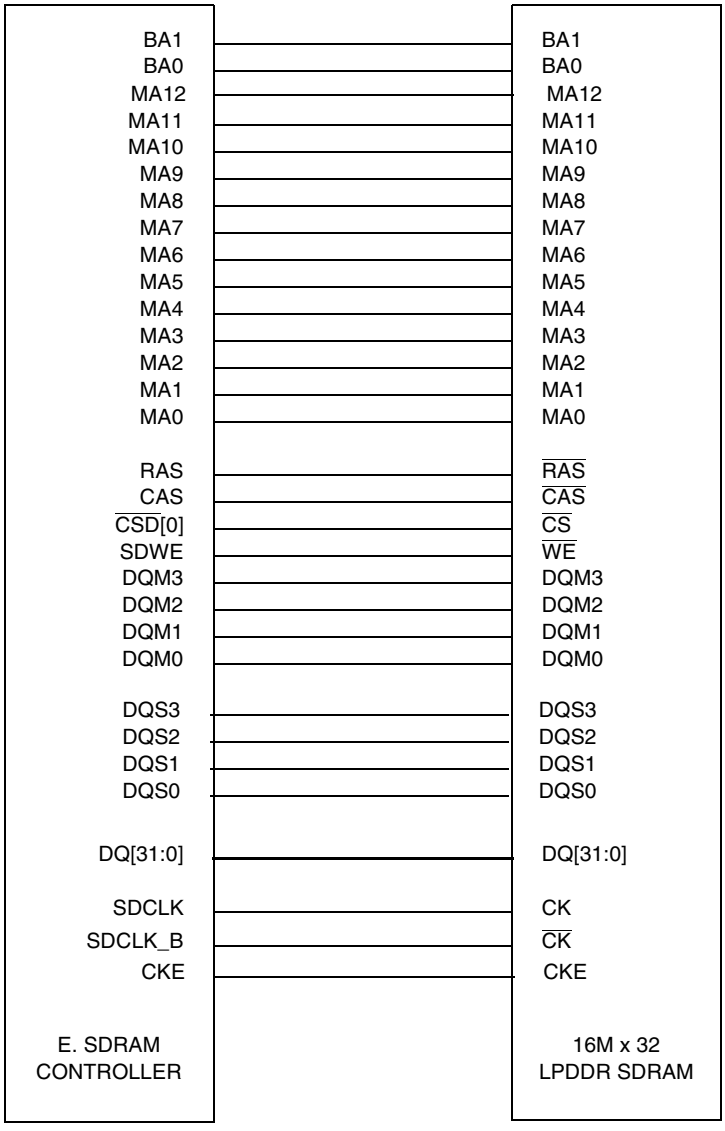


Figure 18-97. Single 512-Mb (16MBx32) LPDDR SDRAM Connection Diagram

18.5.4.3.16 Single 512-Mb (32MBx16) Mobile DDR SDRAM Configuration

Table 18-47. Single 32MBx16 Control Register Value

Control Field	Value
Density	32 Mb
Page Size	1024
ROW	13
COL	10
DSIZ	16 (D [15:0])
SREFR	4

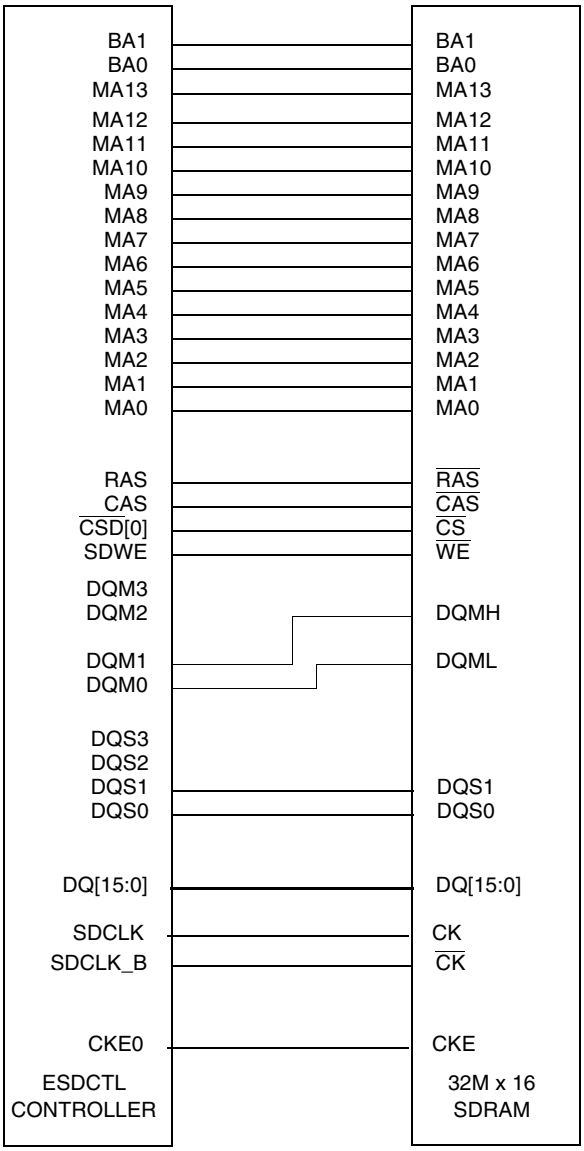


Figure 18-98. Single 512-Mb (32MBx16) LPDDR SDRAM Connection Diagram

Chapter 19
NAND Flash Controller (NFC)

Composed of various control logic units and a 2-Kbyte internal RAM buffer, the NAND Flash Controller (NFC) provides an interface to standard NAND Flash memory devices. See Figure 19-1 for the NFC block diagram.

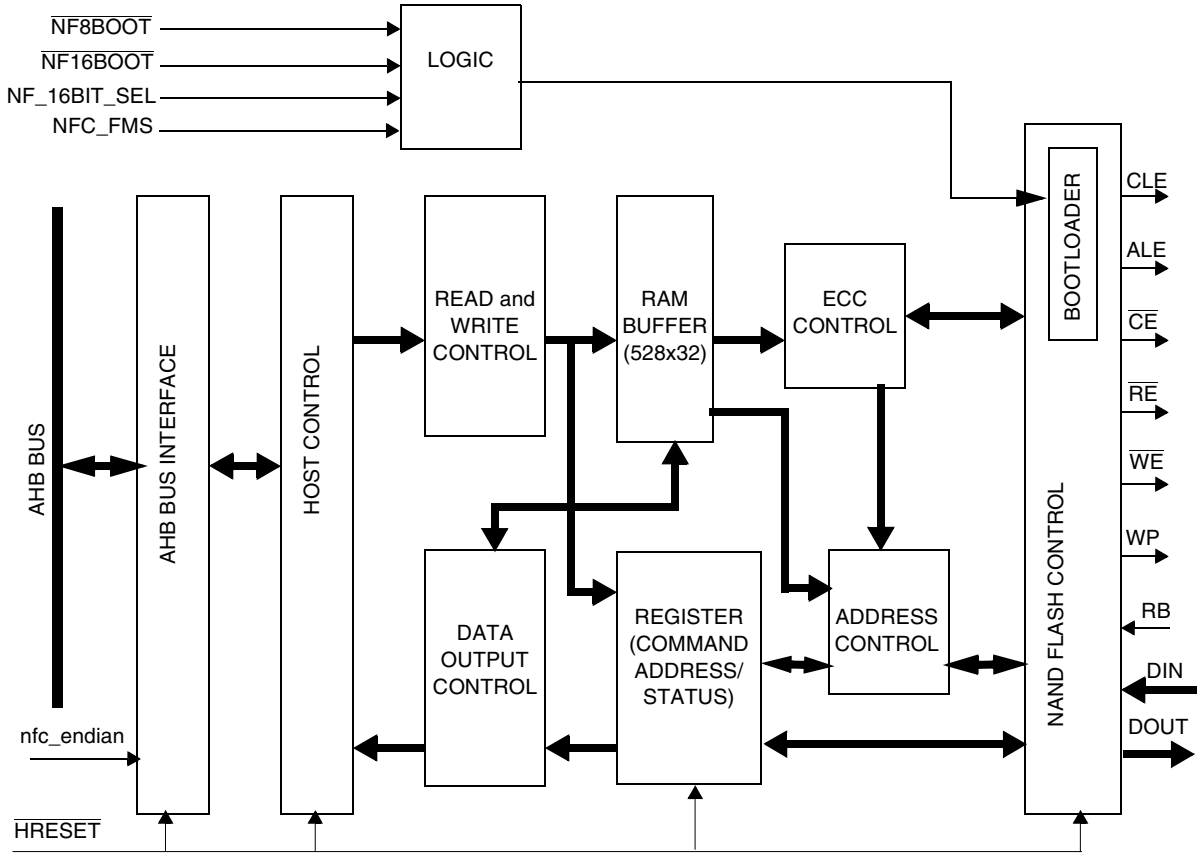


Figure 19-1. NAND Flash Controller Block Diagram

19.1 Overview

The NFC interfaces standard NAND Flash devices to the IC and hides the complexities of accessing the NAND Flash. It provides a glueless interface to both 8-bit and 16-bit NAND Flash parts with page sizes of 512 bytes or 2 Kbytes, and densities up to 64 Gbits per 2-Kbyte page size NAND Flash, and 8 Gbits per 512 byte page size NAND Flash.

19.2 Operation

NFC operation begins by AHB host initiating a read from the NAND Flash device by configuring the controller and then waiting for an interrupt from the NFC. When it receives the interrupt, NFC inputs a page from the NAND Flash device, and upon completion, generates an interrupt to AHB Host. When AHB Host receives this interrupt, it reads the content from NFC's internal RAM buffer. To complete the operation, AHB Host checks the status of the operation by reading the NFC status registers.

The 2 Kbyte RAM buffer is used as boot RAM during a cold reset (if IC is configured for a boot to be carried out from NAND Flash device). After boot procedure completes, RAM is available as buffer RAM. In addition, NAND Flash Controller provides an X16 bit and X32 bit interface to the AHB bus on the chip side, and an X8/X16 interface to the NAND Flash device on the external side.

19.3 Features

- 8-bits/16-bits (Pin Option) NAND Flash Interface
- Internal RAM buffer (2 Kbytes + 64 bytes)
  - — Can be configured as Boot RAM and operates as a buffer during normal operation
  - — Memory mapped (to the same AHB region) registers and internal RAM buffer
- Manual interface with NAND Flash devices
  - — Supports all NAND Flash products regardless of density/organization (with pages of 512 bytes/2 Kbytes)
- AHB Host Interface type
  - — Read/Write Burst
  - — 16-bits/32-bits bus transfers
- Programmable read latency for internal bus (directly affects AHB bus)
- ECC mode/Bypass ECC
- Multiple Reset
  - — Cold Reset/ Warm Reset/Hot Reset (Reset of NFC and NAND Flash device)
- Internal Bootcode loader during power-up (can be enabled/disabled), providing advanced data protection
  - — Data Protection
  - — Write Protection mode for RAM buffer: Write protection of RAM buffer (LSB 1 Kbyte of RAM buffer). For more details see Section 19.7.14, "NAND Flash Write Protection Status (NAND_FLASH_WR_PR_ST)."
  - — Write Protection mode for NAND Flash device: Block based write protection of NAND Flash
- Automatic Write protection for RAM buffer and NAND Flash during power-up
- Write Protection: automatic write protection for RAM buffer and NAND Flash during power-up, in addition to run-time write protection modes for both RAM buffer and NAND Flash device.
- Handshaking Feature: INT pin indicates ready/busy status of NFC
- IO pins sharing support
  - — Allows sharing of IO pins with other memory controllers through special arbitration logic.

19.4 External Signal Description

19.4.1 Overview

The signals shown in Table 19-1 are used to configure and control the NFC and its attached Flash device.

Table 19-1. NFC Signal Properties

Name	Port	Function	I/O	Reset
hclk_in	—	AHB clock of 133 Mhz	I	enable
$\overline{\text{hreset}}$	—	WARM Reset (active low)	I	0
$\overline{\text{ipi\_int\_nfc}}$	—	NAND Flash Controller interrupt	O	1
ipp_flash_clk	—	Clock for the flash side	I	enable
ipp_nfc_ale_out	NFALE	Flash Address Latch Enable	O	0
ipp_nfc_ce_out	$\overline{\text{NFCE}}$	Flash Chip Enable	O	1
ipp_nfc_cle_out	NFCLE	Flash Command Latch Enable	O	0
ipp_nfc_rb_in	NFRB	Flash Ready/Busy	I	1
ipp_nfc_re_out	$\overline{\text{NFRE}}$	Flash Read Enable	O	1
ipp_nfc_read_data_in[15:0]	IO[15:0]	NFC data input from the NAND Flash	I	xxxx
ipp_nfc_we_out	$\overline{\text{NFWE}}$	Flash Write Enable	O	1
ipp_nfc_wp_out	$\overline{\text{NFWP}}$	Flash Write Protect	O	1
ipp_nfc_write_data_out[15:0]	IO[15:0]	NFC data output towards the NAND Flash	O	0000
$\overline{\text{ipp\_reset}}$	—	Power on Reset for booting	I	1
nf_16bit_sel	—	Use 8 or 16-bits NAND Flash	I	1
$\overline{\text{nf8boot}}$	—	Boot from 8-bit NAND Flash	I	1
nfc_fms	—	Flash Memory Select (512 byte/2 Kbyte page size)	I	0
$\overline{\text{ng16boot}}$	—	Boot from 16-bit NAND Flash	I	1

19.4.2 Detailed Signal Descriptions

Table 19-2 gives a detailed description of the NFC signals.

Table 19-2. NFC Detailed Signal Descriptions

Signal	I/O	Description
$\overline{\text{NFCE}}$	O	Flash Chip Enable. This signal indicates the NAND Flash selection. When the NAND Flash device is in the Busy state, or when the NAND Flash device is accessed, this signal is low.
$\overline{\text{NFRE}}$	O	Flash Read Enable. This output is the NAND Flash device serial data output control. When active, this signal drives the data from the NAND Flash device onto the NAND Flash I/O bus, allowing the NFC to read the data. When reading a burst from the NAND Flash device, this signal increments the NAND Flash internal column address counter by one.

Table 19-2. NFC Detailed Signal Descriptions (continued)

Signal	I/O	Description
NFWE	O	Flash Write Enable.This output controls writes to NAND Flash I/O port, thus allowing the NAND Flash device to read data. Commands, address and data are latched on the rising edge of the WE signal.
NFCLE	O	Flash Command Latch Enable. The CLE output controls the activating path for commands sent to the command register of NAND Flash (NAND_Flash_CMD). When active high, commands are latched into the command register of NAND Flash through the I/O ports on the rising edge of the WE signal.
NFALE	O	Flash Address Latch Enable. The ALE output controls the activating path for addresses sent to the address register of NAND Flash (NAND_Flash_Add). When active high, addresses are latched into the NAND FC address register of NAND Flash through the I/O ports on the rising edge of the WE signal.
NFWP	O	Flash Write Protect. This signal provides inadvertent program/erase protection during power transition and is automatically controlled by NFC. This pin status is only active (held low) during power-up.
NFRB	I	Flash Ready/Busy. This signal indicates the status of the NAND Flash operation. When low, it indicates that a program, erase, or random read operation of NAND Flash is in process. Upon completion of the process, this signal returns to high state. **Note:** This signal is connected to an open drain output, via a 100 KΩ pull-up resister (outside the external NAND Flash memory device).
hreset		Warm Reset. This signal produces a Warm reset causing NFC and the NAND Flash device to cease current operation, and set all internal registers to their default state. See Figure 19-2 for a timing diagram of warm reset operation. AHB bus interface is connected directly to this signal (hreset) and will cause a reset immediately when this line goes to low state. NFC will not be reset if hreset pulses are shorter than two ipp_flash_clk cycles (50 ns if the clock period is 25 ns), but NFC will be reset if hreset pulse is longer than 20 ipp_flash_clk cycles (500 ns if this clock period is 25 nS). Warm reset has no effect on the contents of main/spare area buffers.
NF8BOOT NF16BOOT NF_16BIT_SEL		The NF8BOOT and NF16BOOT are boot signals that determine when the chip will boot from the NAND Flash device, in addition to indicating boot selection it is also used to controls the bus width of the NAND Flash (8-bit or 16-bit). If one of the boot inputs is asserted (NF8BOOT or NF16BOOT is low) at System Power-On reset (ipp_reset rising), a 2 Kbyte sized boot code is copied from the NAND Flash device to the RAM buffer. If none of the boot signals are asserted, then the input signal NF_16BIT_SEL is read. This is part of the logic of the operating modes of the NFC. For more information on operating modes, see Section 19.8.1, "Modes of Operation." **Note:** The boot signals should remain at the same value before and after the boot process.
NFC_FMS	I	Flash Memory Select. NFC_FMS signal indicates size of NAND Flash page (512 byte or 2 Kbyte). 0 NAND Flash page size is 512 bytes. 64Mb/128Mb/256Mb/512Mb/1Gb DDP 1 NAND Flash page size is 2 Kbytes.
ipp_reset	I	This input is Power On Reset (POR) signal in the NFC. When it is asserted high, a POR takes place.
ipi_int_nfc	O	NFC Interrupt. This output is the NFC interrupt, and is asserted when an NFC event takes place. It sets itself to '1' when basic operation and boot loading is done, or when a warm or hot reset is released. In addition, it is asserted when any of the following occur: • NAND Flash command input • NAND Flash address input • NAND Flash data input • NAND Flash data output
hclk_in	I	H Clock Input. This is NFC clock signal, which arrives from the AHB side. Its value can up to 133 MHz.
ipp_nfc_write_data_out[15:0]	O	AHB host uses this path to write data to registers or to internal memory.

Table 19-2. NFC Detailed Signal Descriptions (continued)

Signal	I/O	Description
ipp_nfc_read_data_in[15:0]	I	AHB host uses this path to read data from registers or from internal memory.
ipp_flash_clk		This clock signal controls NFC's state machine when interfacing with a NAND Flash device.

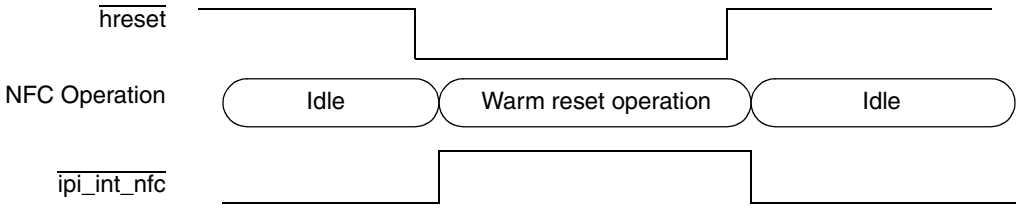


Figure 19-2. Warm Reset Operation

19.5 NFC Buffer Memory Space

Table 19-3 shows the organization of the buffer memory space in the NFC.

Table 19-3. Data (Buffer) Organization in Memory

Address	Use	Access
0xD800_0000– 0xD800_01FE	Main area Buffer 0	R/W
0xD800_0200–0xD800_03FE	Main area Buffer 1	R/W
0xD800_0400–0xD800_05FE	Main area Buffer 2	R/W
0xD800_0600–0xD800_07FE	Main area Buffer 3	R/W
0xD800_0800–0xD800_080E	Spare area Buffer 0	R/W
0xD800_0810–0xD800_081E	Spare area Buffer 1	R/W
0xD800_0820–0xD800_082E	Spare area Buffer 2	R/W
0xD800_0830–0xD800_083E	Spare area Buffer 3	R/W
0xD800_0840–0xD800_0BFE	Reserved	—
0xD800_0E00–0xD800_0E1C	Registers	R/W

19.5.1 Main and Spare Area Buffers

Main area buffer is a general data block. Spare area buffer is used for a variety of functions including Error Correction. Memory is organized in a different manner depending on Flash bus width (8-bit or 16-bit). Table 19-4 shows an 8-bit organization, and Table 19-5 shows a 16-bit configuration. Host can use all of the spare area except for BI and ECC code areas. For example, AHB host can write data to a reserved area in spare area buffer upon program operation. NFC automatically generates ECC code for both main and spare data during NFC's data loading to NAND Flash, and NFC updates ECC code to NAND Flash spare area, but does not update ECC code to spare buffer. When programming/reading spare area, the spare area buffer number (SB0–SB3) must be selected using the RAM buffer address register (NFC_RAM_BUFF).

Table 19-4. Spare Area Buffer (with X8 I/O bus)

Address	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0xD800_0800 (SB0)	LSN(2nd)[1]								LSN(1st)[1]							
0xD800_0802 (SB0)	WC(1st)[2]								LSN(3rd)[1]							
0xD800_0804 (SB0)	BI[3]								WC(2nd)[2]							
0xD800_0806 (SB0)	ECC Code for Main area data (2nd)								ECC Code for Main area data (1st)							
0xD800_0808 (SB0)	ECC Code for Spare area data (1st)								ECC Code for Main area data (3rd)							
0xD800_080A (SB0)	Reserved								ECC Code for Spare area data (2nd)							
0xD800_080C (SB0)	Reserved								Reserved							
0xD800_080E (SB0)	Reserved								Reserved							
0xD800_0810–0xD800_081E (SB1)	SB1–SB3 have same assignment like SB0.															
0xD800_0820–0xD800_082E (SB2)																
0xD800_0830–0xD800_083E (SB3)																

1 LSN: Logical Sector Number

2 WC: Wrap Count and other bytes have same wrap count information and are used as error correction for wrap count itself.

3 BI: Bad Block Information

Table 19-5. Spare Area Buffer (with X16 I/O bus)

Address	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0xD800_0800 (SB0)	LSN(2nd)[1]								LSN(1st)[1]							
0xD800_0802 (SB0)	WC(1st)[2]								LSN(3rd)[1]							
0xD800_0804 (SB0)	Reserved								WC(2nd)[2]							
0xD800_0806 (SB0)	ECC Code for Main area data (2nd)								ECC Code for Main area data (1st)							
0xD800_0808 (SB0)	ECC Code for Spare area data (1st)								ECC Code for Main area data (3rd)							
0xD800_080A (SB0)	BI[3]								ECC Code for Spare area data (2nd)							
0xD800_080C (SB0)	Reserved								Reserved							
0xD800_080E (SB0)	Reserved								Reserved							
0xD800_0810–0xD800_081E (SB1)	SB1–SB3 have same assignment like SB0.															
0xD800_0820–0xD800_082E (SB2)																
0xD800_0830–0xD800_083E (SB3)																

1 LSN: Logical Sector Number

2 WC: Wrap Count and other bytes have same wrap count information and are used as error correction for wrap count itself.

3 BI: Bad Block Information

19.6 Memory Map and Register Definition

Section 19.7, “Register Descriptions” provides detail descriptions for all the NFC registers.

19.6.1 Memory Map

Table 19-6 provides the NFC memory map.

Table 19-6. NFC Module Register Memory Map

Address	Register	R/W	Reset Value	Section/Page
0xD800_0E00 (NFC_BUFSIZ)	NAND Flash Controller Buffer Size Register	R	0x0000_0001	19.7.1/19-9
0xD800_0E02	Reserved	—	—	—
0xD800_0E04 (RAM_BUFFER_ADDRESS)	RAM Buffer Address Register	R/W	0x0000_0000	19.7.2/19-10
0xD800_0E06 (NAND_FLASH_ADD)	NAND Flash Address Register	R/W	0x0000_0000	19.7.3/19-10
0xD800_0E08 (NAND_FLASH_CMD)	NAND Flash Command Register	R/W	0x0000_0000	19.7.4/19-11
0xD800_0E0A (NFC_CONFIGURATION)	NFC Internal Buffer Lock Control	R/W	0x0000_0001	19.7.5/19-11
0xD800_0E0C (ECC_STATUS_RESULT)	Controller Status and Result of Flash Operation	R	0x0000_0000	19.7.6/19-12
0xD800_0E0E (ECC_RSLT_MAIN_AREA)	ECC Error Position Main Area Data Error x8 ECC Error Position Main Area Data Error x16	R	0x0000_0000	19.7.7/19-12 19.7.8/19-13
0xD800_0E10 (ECC_RSLT_SPARE_AREA)	ECC Error Position Spare Area Data Error x8 ECC Error Position Spare Area Data Error x16	R	0x0000_0000	19.7.9/19-14 19.7.10/19-14
0xD800_0E12 (NF_WR_PROT)	NAND Flash Write Protection	R/W	0x0000_0002	19.7.11/19-15
0xD800_0E14 (UNLOCK_START_BLK_ADD)	Address to Unlock in Write Protection Mode—Start	R/W	0x0000_0000	19.7.12/19-15
0xD800_0E16 (UNLOCK_END_BLK_ADD)	Address to Unlock in Write Protection Mode—End	R/W	0x0000_0000	19.7.13/19-16
0xD800_0E18 (NAND_FLASH_WR_PR_ST)	NAND Flash Write Protection Status	R/W	0x0000_0002	19.7.14/19-16
0xD800_0E1A (NAND_FLASH_CONFIG1)	NAND Flash Operation Configuration1	R/W	0x0000_0008	19.7.15/19-17
0xD800_0E1C (NAND_FLASH_CONFIG2)	NAND Flash Operation Configuration 2	R/W	0x0000_0000	19.7.16/19-18

19.6.2 Register Summary

Figure 19-3 shows the key to the register fields and Table 19-7 shows the register figure conventions.

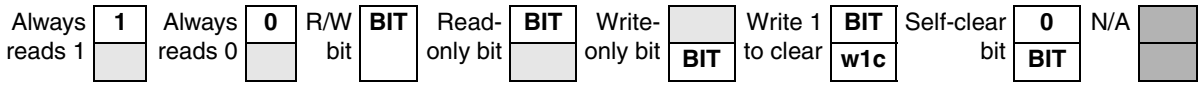


Figure 19-3. Key to Register Fields

Table 19-7. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 19-8. NFC Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xD800_0E00 (NFC_BUFSIZ)	R	0	0	0	0	0	0	0	0	0	0	0	0	BUFSIZE			
	W																
0xD800_0E02 (Reserved)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
0xD800_0E04 (RAM_BUFFER_ADDRESS)	R	0	0	0	0	0	0	0	0	0	0	0	0	RBA			
	W																
0xD800_0E06 (NAND_FLASH_ADD)	R	ADD															
	W																
0xD800_0E08 (NAND_FLASH_CMD)	R	CMD															
	W																

Table 19-8. NFC Register Summary (continued)

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xD800_0E0A (NFC_CONFIGURATION)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BLS	
	W																
0xD800_0E0C (ECC_STATUS_RESULT)	R	0	0	0	0	0	0	0	0	0	0	0	0	ERM		ERS	
	W																
0xD800_0E0E (ECC_RSLT_MAIN_AREA)	R	0	0	0	0	ECC Result 1									ECC Result2		
	W																
0xD800_0E10 (ECC_RSLT_SPARE_AREA)	R	0	0	0	0	0	0	0	0	0	0	0	ECC Result 4		ECC Result 3		
	W																
0xD800_0E12 (NF_WR_PROT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	WPC		
	W																
0xD800_0E14 (UNLOCK_START_BLK_ADD)	R	USBA															
	W																
0xD800_0E16 (UNLOCK_END_BLK_ADD)	R	UEBA															
	W																
0xD800_0E18 (NAND_FLASH_WR_PR_ST)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	US	LS	LTS
	W																
0xD800_0E1A (NAND_FLASH_CONFIG1)	R	0	0	0	0	0	0	0	0	NF_$\overline{CE}$	NFC_RST	NF_BIG	INT_MSK	ECC_EN	SP_EN	0	0
	W																
0xD800_0E1C (NAND_FLASH_CONFIG2)	R	INT	0	0	0	0	0	0	0	0	0	FDO			FDI	FADD	FCMD
	W	INT															

19.7 Register Descriptions

19.7.1 Internal SRAM SIZE (NFC_BUFSIZE)

This 16-bit read-only register contains internal SRAM size installed in the IC. Bit assignments for this register is shown in Figure 19-4 and the field descriptions are shown in Table 19-9.

0xD800_0E00 (NFC_BUFSIZ) Access: User read-only

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

Figure 19-4. NFC_BUFSIZE Register

Table 19-9. NFC_BUFSIZE Register Field Description

Name	Description
15–4	Reserved
3–0 BUFSIZE	Buffer Size. The size of the internal RAM buffer. 0000 1 Kbyte 0001 2 Kbytes (Default) 0010–1111 Reserved

19.7.2 Buffer Number for Page Data Transfer (RAM_BUFFER_ADDRESS)

RBA specifies which part of the RAM Buffer is transferred to/from flash memory. Bit assignments for this register is shown in Figure 19-5 and the field description is shown in Table 19-10.

0xD800_0E04 (RAM_BUFFER_ADDRESS) Access: User Read/Write

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

Figure 19-5. RAM Buffer Address Register

Table 19-10. RAM Buffer Address Field Descriptions

Field	Description
15–4	Reserved
3–0 RBA	RAM Buffer Address. Specifies the RAM buffer number to use for data transfers to/from the NAND Flash device. 0000 1st internal RAM buffer 0001 2nd internal RAM buffer 0010 3rd internal RAM buffer 0011 4th internal RAM buffer

19.7.3 NAND Flash Address (NAND_FLASH_ADD)

NAND Flash Address (NAND_FLASH_ADD) register is a read-write register containing the address of NAND Flash device that will be read, programmed or erased. The address in NAND_FLASH_ADD register is written to the Flash device. Bit assignments for this register is shown in Figure 19-6 and the field descriptions are shown in Table 19-11.

0xD800_0E06 (NAND_FLASH_ADD) Access: User Read/Write

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

Figure 19-6. NAND Flash Address Register

Table 19-11. NAND Flash Address Register Field Description

Field	Description
15–0 ADD	NAND Flash Address. NAND Flash address which will be read, programmed or erased. This address is written to the NAND Flash device.

19.7.4 NAND Flash Command (NAND_FLASH_CMD)

Bit assignments for this register is shown in Figure 19-7 and field descriptions are shown in Table 19-12.

0xD800_0E08 (NAND_FLASH_CMD) Access: User Read/Write

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

Figure 19-7. NAND_Flash_CMD Register

Table 19-12. NAND_Flash_CMD Register Field Description

Field	Description
15–0 CMD	NAND Flash Command. This field contains the CMD that is written to the NAND Flash device.

19.7.5 NFC Internal Buffer Lock Control (NFC_CONFIGURATION)

Bit assignments for this register is shown in Figure 19-8 and field descriptions are shown in Table 19-13.

0xD800_0E0A (NFC_CONFIGURATION) Access: User Read/Write

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

Figure 19-8. NFC_Configuration Register

Table 19-13. NFC_Configuration Register Field Descriptions

Field	Description
15–2	Reserved.
1–0 BLS	Buffer Lock Set. This field specifies the buffer lock status of first 2 pages in the internal buffer. The other two pages are always Unlocked. (For more details, see Section 19.9.3, “Write Protection Operation.”) 00 Locked 01 Locked (default) 10 Unlocked 11 Locked

19.7.6 Controller Status and Result of Flash Operation (ECC_STATUS_RESULT)

This register shows the number of errors in a page for Spare and Main Area as a result of ECC check upon a page read.Bit assignment is shown in Figure 19-9 and the field descriptions are shown in Table 19-14.

0xD800_0E0C (ECC_STATUS_RESULT) Access: User read-only

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

Figure 19-9. ECC_Status_Result

Table 19-14. ECC_STATUS_RESULT Register Field Description

Field	Description
15–4	Reserved
3–2 ERM	ECC Error for Main Area Data (ERM) and Spare Area Data (ERS). These field shows the number of errors in a page as a result of ECC check upon page read. The ECC algorithm of NFC doesn’t correct if there are greater than two fault bits per page. It interprets any ECC error count greater than two as non-correctable. 00 No error 01 1-bit Error (Correctable Error) 10 2-bits Error or more (Non-correctable Error) 11 Reserved
1–0 ERS	

19.7.7 ECC Error Position of Main Area Data Error x8 (ECC_RSLT_MAIN_AREA)

(NAND Flash X8 data bus case)

This register contains ECC error position address which is used to select the bit to repair in Main Area for 8-bit NAND FlashBit assignments for this register is shown in Figure 19-10, and the field descriptions are shown in Table 19-15.

0xD800_0E0E (ECC_RSLT_MAIN_AREA) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ECC RESULT1									ECC RESULT2		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-10. ECC_RSLT_MAIN_AREA Register

Table 19-15. ECC_RSLT_MAIN_AREA Register Field Descriptions

Field	Description
15–12	Reserved
11–3 ECC RESULT1	ECC Result 1: ECC error position address which selects one of Main area data bytes (one of 512 bytes).
2–0 ECC RESULT2	ECC Result 2: ECC error position address which selects one of the 8 data bits in the byte.

19.7.8 ECC Error Position of Main Area Data Error x16 (ECC_RSLT_MAIN_AREA)

(NAND Flash X16 data bus case)

This register contains ECC error position address which is used to select the bit to repair in Main Area for 16-bit NAND Flash bit assignments for this register is shown in Figure 19-11, and the field descriptions are shown in Table 19-16.

0xD800_0E0E (ECC_RSLT_MAIN_AREA) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ECC RESULT1								ECC RESULT2			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-11. ECC_RSLT_MAIN_AREA Register

Table 19-16. ECC_RSLT_MAIN_AREA Register Field Descriptions

Field	Description
15–12	Reserved
11–4 ECC RESULT1	ECC Result 1: ECC error position address which selects one of the 16 data bits in the half word
3–0 ECC RESULT2	ECC Result 2: ECC error position address which selects one of the 8 data bits in the byte

19.7.9 ECC Error Position of Spare Area Data Error x8 (ECC_RSLT_SPARE_AREA)

(NAND Flash X8 data bus case)

This register contains ECC error position address which is used to select the bit to repair in Spare Area for 8-bit NAND Flash bit assignments for this register is shown in Figure 19-12, and the field descriptions are shown in Table 19-17.

0xD800_0E10 (ECC_RSLT_SPARE_AREA) Access: User read-only

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

Figure 19-12. ECC_Rslt_Spare_Area Register

Table 19-17. ECC_Rslt_Spare_Area Descriptions

Field	Description
15–5	Reserved
4–3 ECC RESULT4	ECC Result 4: ECC error position address which selects one of Logical Sector Number (3 bytes)
2–0 ECC RESULT3	ECC Result 3: ECC error position address which selects one of 8 data bits in the byte.

19.7.10 ECC Error Position of Spare Area Data Error x16 (ECC_RSLT_SPARE_AREA)

(NAND Flash X16 data bus case) This register contains ECC error position address which is used to select the bit to repair in Spare Area for 16-bit NAND Flash. Bit assignments for this register is shown in Figure 19-13 and the field descriptions are shown in Table 19-18.

0xD800_0E10 (ECC_RSLT_SPARE_AREA) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	Result4	Result 3			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-13. ECC_Rslt_Spare_Area Register

Table 19-18. ECC_Rslt_Spare_Area Descriptions

Field	Description
15–5	Reserved
4 ECC RESULT4	ECC Result 4: ECC error position address which selects one of Logical Sector Number (3 bytes)
3–0 ECC RESULT3	ECC Result 3: ECC error position address which selects one of the 8 data bits in the byte.

19.7.11 NAND Flash Write Protection (NF_WR_PROT)

This register specifies the Protection command that the controller will perform: Lock, Unlock, or Lock Tight. Bit assignments for this register is shown in Figure 19-14 and field descriptions are shown in Table 19-19.

0xD800_0E12 (NF_WR_PROT) Access: User Read/Write

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

Figure 19-14. NAND Flash Write Protection Register

Table 19-19. NAND Flash Write Protection Register Field Descriptions

Field	Description
15–3	Reserved
2–0 WPC	Write Protection Command: This field specifies the operation which the controller will perform. 100 Unlock NAND Flash block(s) according to given block address range 010 Lock all NAND Flash block(s) 001 Lock-tight locked blocks(s) (See Section 19.9.3, “Write Protection Operation” for more details.)

19.7.12 Address to Unlock in Write Protection Mode—Start (UNLOCK_START_BLK_ADD)

Starting address of block memory in the NAND Flash that is unlocked in the Write Protection mode. Bit assignments for this register is shown in Figure 19-15 and the field descriptions are shown in Table 19-20.

0xD800_0E14 (UNLOCK_START_BLK_ADD) Access: User Read/Write

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

Figure 19-15. Unlock_Start_Blk_Add Register

Table 19-20. Unlock_Start_Blk_Add Register Field Description

Field	Description
15–0 USBA	Unlock Start Block Address. Starting address of block memory in the NAND Flash that is unlocked in Write Protection mode. For more details on this, see Section 19.9.3.4, "Write Protection Status."

19.7.13 Address to Unlock in Write Protection Mode—End (UNLOCK_END_BLK_ADD)

End address of block memory in the NAND Flash that is unlocked in the Write Protection mode. Bit assignments for this register is shown in Figure 19-16 and the field descriptions are shown in Table 19-21.

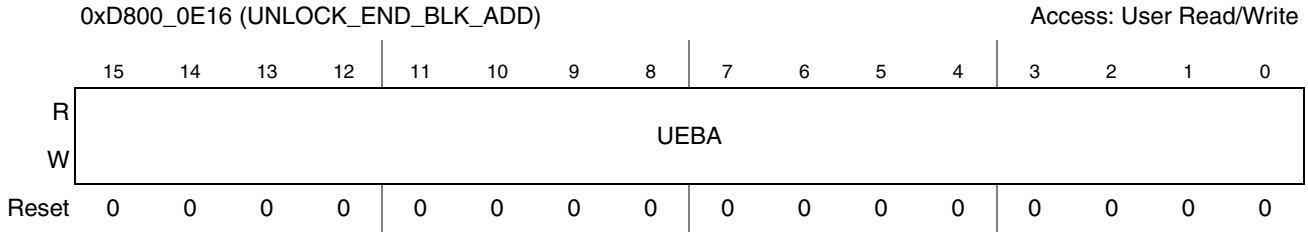


Figure 19-16. UNLOCK_END_BLK_ADD Register

Table 19-21. UNLOCK_END_BLK_ADD Register Field Description

Field	Description
15–0 UEBA	Unlock End Block Address. Ending address of block memory in the NAND Flash that is unlocked in Write Protection mode. For more details, see Section 19.9.3.4, "Write Protection Status."

19.7.14 NAND Flash Write Protection Status (NAND_FLASH_WR_PR_ST)

This status register reads the NAND Flash Write Protection Status: Lock, Unlock or Lock Tight status. Bit assignments for this register is shown in Figure 19-17 and the field descriptions are shown in Table 19-22.

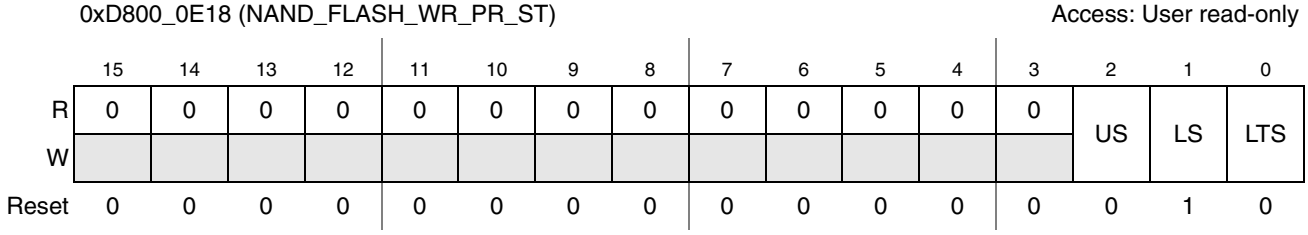


Figure 19-17. NAND_FLASH_WR_PR_ST Register

Table 19-22. NAND_FLASH_WR_PR_ST Register Field Descriptions

Field	Description
15–3	Reserved
0 LTS	Lock-tight Status. Indicates if any of the locked block(s) is (are) have a lock-tight status. 0 Locked block(s) is (are) not lock-tight. 1 Locked block(s) is (are) lock-tight.

Table 19-22. NAND_FLASH_WR_PR_ST Register Field Descriptions (continued)

Field	Description
1 LS	Locked Status. This bit indicate whether all NAND Flash blocks are in locked status or in lock-Unlock status. 0 Not all NAND Flash blocks are in locked status. 1 There are unlocked block(s) in NAND Flash.
2 US	Unlocked Status. This bit indicates whether there are any unlocked blocks in the NAND Flash. 0 There are no unlocked blocks in NAND Flash. 1 There are unlocked block(s) in NAND Flash.

There are four states for write protected: lock, unlock-lock, unlock-lockt, and lockt.

Table 19-23. Write Protected States

State	Status Bits—US -LS -LTS
Lock—All blocks are locked	010
Unlock-lock—There are unlocked blocks	110
Unlock-Lockt—There are unlocked blocks: cant change to other state	101
Lockt—All block are locked: cant change to other state	001

19.7.15 NAND Flash Operation Configuration (NAND_FLASH_CONFIG1)

This register is a configuration register for NAND Flash device to control the ECC Enable or Disable Mask Interrupt. Bit assignments for this register is shown in Figure 19-18 and the field descriptions are shown in Table 19-24.

0xD800_0E1A (NAND_FLASH_CONFIG1) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	NF_$\overline{CE}$	NFC_RST	NF_BIG	INT_MSK	ECC_EN	SP_EN	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 19-18. NAND_FLASH_CONFIG1 Register

Table 19-24. NAND_FLASH_CONFIG1 Register Field Descriptions

Field	Description
15–8	Reserved
7 NF_$\overline{CE}$	NAND Flash Force CE. This bit forces the $\overline{CE}$ signal to the NAND Flash device to 0 when enabled. This bit allows a greater range of support new NAND Flash devices. 0 $\overline{CE}$ signal operates normally. 1 $\overline{CE}$ signal is asserted as long as this bit is set to 1.
6 NFC_RST	NFC Reset. This bit resets the NFC state machine. 0 Do not reset the NFC state machine 1 Reset the NFC state machine

Table 19-24. NAND_FLASH_CONFIG1 Register Field Descriptions (continued)

Field	Description
5 NF_BIG	NAND Flash Big Endian Mode. This bit enables big Endian mode when writing from internal RAM to the NAND Flash device or reading from NAND Flash device to internal RAM. 0 Little Endian mode 1 Big Endian mode
4 INT_MSK	Mask interrupt Bit. This bit enables the interrupt by masking or not masking the interrupt bit. 0 Mask interrupt is disabled (interrupt enabled). 1 Mask interrupt is enabled (interrupt disabled).
3 ECC_EN	ECC operation Enable. This bit determines whether ECC operation is executed or bypassed 0 ECC operation is bypassed. 1 ECC operation is executed.
2 SP_EN	NAND Flash Spare Enable. This bit determines whether host reads/writes are to NAND Flash spare data only or NAND Flash main and spare data. 0 NAND Flash main and spare data is enabled. 1 NAND Flash spare only data is enabled.
1–0	Reserved

19.7.16 NAND Flash Operation Configuration 2 (NAND_FLASH_CONFIG2)

This register controls the NAND Flash signals: CLE, ALE, WE, RE, $\overline{\text{CE}}$. and sets the interrupt after command completion. Bit assignments for this register is shown in Figure 19-19 and the field descriptions are shown in Table 19-25.

0xD800_0E1C (NAND_FLASH_CONFIG2) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INT	0	0	0	0	0	0	0	0	0	FDO			FDI	FADD	FCMD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 19-19. NAND_FLASH_CONFIG2 Register

Table 19-25. NAND_FLASH_CONFIG2 Register Field Descriptions

Field	Description
15 INT	Interrupt. This field determines the state of the interrupt output of the NAND FLash Controller. It is set by the controller when a basic operation is done. It is set cleared by the Host writing “0” to this field. (Host can also set this bit by writing “1” to this field). 0 Basic operation or boot loading is still running. 1 Basic operation or boot loading is done.
14–6	Reserved
5–3 FDO	NAND Flash Data Output. This bit enables NAND Flash Data Output 001 One page data out[1] 010 NAND Flash ID data out 100 NAND Flash Status Register data out

Table 19-25. NAND_FLASH_CONFIG2 Register Field Descriptions (continued)

Field	Description
2 FDI	NAND Flash Data Input. This field enables NAND Flash Data Input 0 No NAND Flash data input operation 1 Enable NAND Flash data input operation
1 FADD	NAND Flash Address Input. This field enables NAND Flash Address Input 0 No NAND Flash Address input operation 1 Enable NAND Flash Address input operation
0 FCMD	NAND Flash Command Input. This field enables the NAND Flash Command Input 0 No NAND Flash Command input operation 1 Allow NAND Flash Command input operation

[1] Page size is determined by SP_EN register bit (main + spare or spare only). It is 528 bytes (main+spare) or 16 bytes (spare) regardless of the NFC_FMS setting.

NOTE

INT bit reset value is 0, but soon after power-up it will change to 1. INT will change from 0 to 1, when performing boot from NAND Flash, after boot code transfer is accomplished. For more information, see Section 19.8.1, "Modes of Operation."

When basic operation is completed, FCMD/FADD/FDI/FDO bits change to LOW automatically. Only one of the bit fields (FCMD/FADD/FDI/FDO) can be set at any given time.

19.8 Functional Description

This section provides the functional description for the NAND Flash Controller.

19.8.1 Modes of Operation

Operating mode is determined by four input lines: NFC_FMS, $\overline{\text{NF8BOOT}}$, $\overline{\text{NF16BOOT}}$, NF_16BIT_SEL, as shown in Table 19-26. It is possible to configure the i.MX31 to boot from a NAND Flash device. For this to occur, one of the signals $\overline{\text{NF8BOOT}}$ or $\overline{\text{NF16BOOT}}$ must be low (0). If both of these signals are high, a boot from the NAND Flash device does not occur. If both of these signals are low, the situation is undefined, and should not be used.

The value of the NFC_FMS determines the page size of NAND Flash: 512 bytes if NFC_FMS is low (0), and 2 Kbyte if NFC_FMS is high (1). While booting from NAND Flash device, bus width is determined by the bus width used during boot. While not booting from NAND Flash device, bus width is determined by the value of NF_16BIT_SEL signal (0=8-bit bus, 1=16-bit bus).

Table 19-26. NAND Flash Controller Operating Modes

NFC_FMS	$\overline{\text{NF8BOOT}}$	$\overline{\text{NF16BOOT}}$	NF_16BIT_SEL	Function
0	1	1	0	Do not Boot from NAND Flash. NAND Flash is configured to 8-bits I/O bus width and page size is 512 bytes.
0	1	1	1	Do not Boot from NAND Flash. NAND Flash is configured to 16-bits I/O bus width and page size is 512 bytes.
0	1	0	X	Boot from 16-bit NAND Flash. NAND Flash is configured to the same value as it booted from (16-bits) and page size is 512 bytes.
0	0	1	X	Boot from 8-bit NAND Flash. NAND Flash is configured to the same value as it booted from (8-bits) and page size is 512 bytes.
1	1	1	0	Do not Boot from NAND Flash. NAND Flash is configured to 8-bits I/O bus width and a page size is 2 Kbyte.
1	1	1	1	Do not Boot from NAND Flash. NAND Flash is configured to 16-bits I/O bus width and a page size is 2 Kbyte.
1	1	0	X	Boot from 16-bit NAND Flash. NAND Flash is configured to the same value as it booted from (16-bits) and page size is 2 Kbyte.
1	0	1	X	Boot from 8-bit NAND Flash. NAND Flash is configured to the same value as it booted from (8-bits) and page size is 2 Kbyte.
X	0	0	X	NOT DEFINED (Do not use this setting.)

19.8.2 Booting From a NAND Flash Device

Booting from NAND Flash device proceeds as follows[1]:

1. BOOTLOADER copies 1 page of 2 Kbytes or 4 Pages of 528 bytes (depending on NFC_FMS input value) from the NAND Flash to the NFC internal RAM buffer. The transfer is done in the following order:
   - For NAND Flash with 528-byte page depth case:
     1st page read => 2nd page read => 3rd page read => 4th page read
   - For NAND Flash with 2 K page depth case:
     One page read
2. AHB Host then reads (after exiting from reset state) the first code from NFC internal RAM buffer.

1. A Boot from the NAND Flash device will only occur if one of the Boot inputs is asserted (NF8BOOT or NF16BOOT is low) at System Power-On reset (ipp_resetb rising).

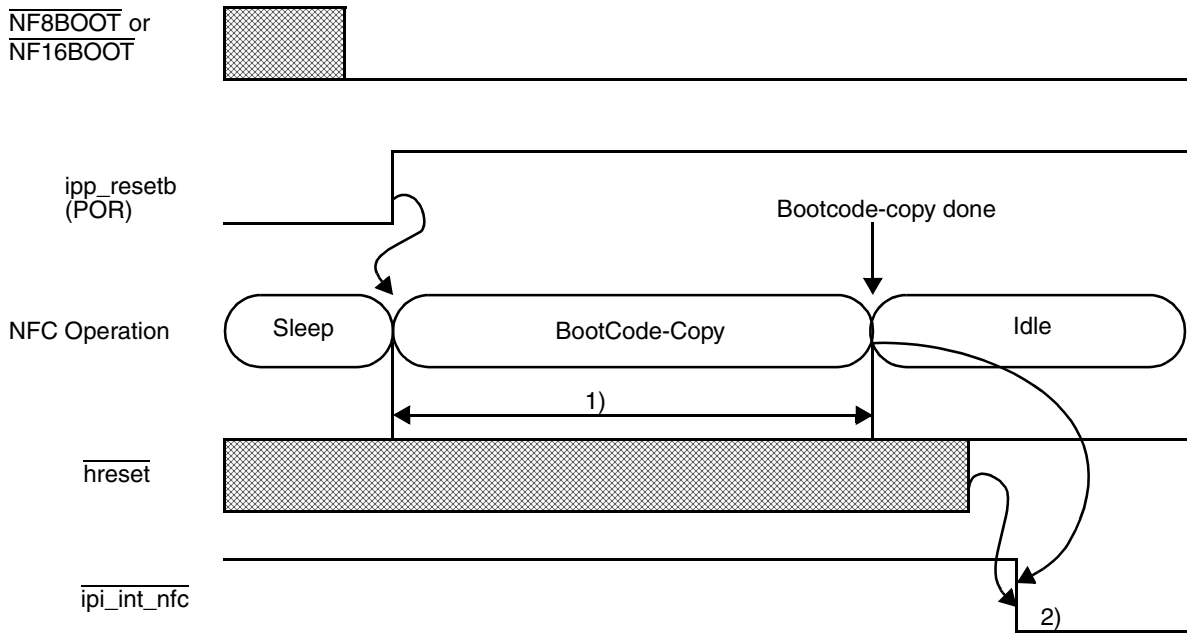


Figure 19-20. Boot Mode Operation

NOTE

2 Kbytes of bootcode copy takes about 160 µs. Host must read bootcode in the RAM buffer (2 Kbytes) after bootcode copy completion.

Interrupt pin (ipi_int_nfc) goes from high to low when the bootcode-copy is completed, and upon hreset rising edge. If hreset goes from Low to High before bootcode-copy is done, Interrupt pin (ipi_int_nfc) goes from High to Low as soon as bootcode-copy is completed.

The interrupt can be relevant for cases of secured boot (booting from ROM and then enabling the NFC boot).

19.8.3 NAND Flash Control

NAND Flash Control generates all control signals that control the NAND Flash: CE (Flash Chip Enable), RE (Read Enable for read operations), WE (Flash Write Enable), CLE (Flash Command Latch Enable), ALE (Flash Address Latch Enable). It monitors R/nB (Flash Ready/Busy indication) signal to check if the NAND Flash is in the middle of operation. BOOTLOADER is part of NAND Flash Control Block. Figure 19-21, Figure 19-22, and Figure 19-23 show NAND Flash read, program, and erase timing diagrams.

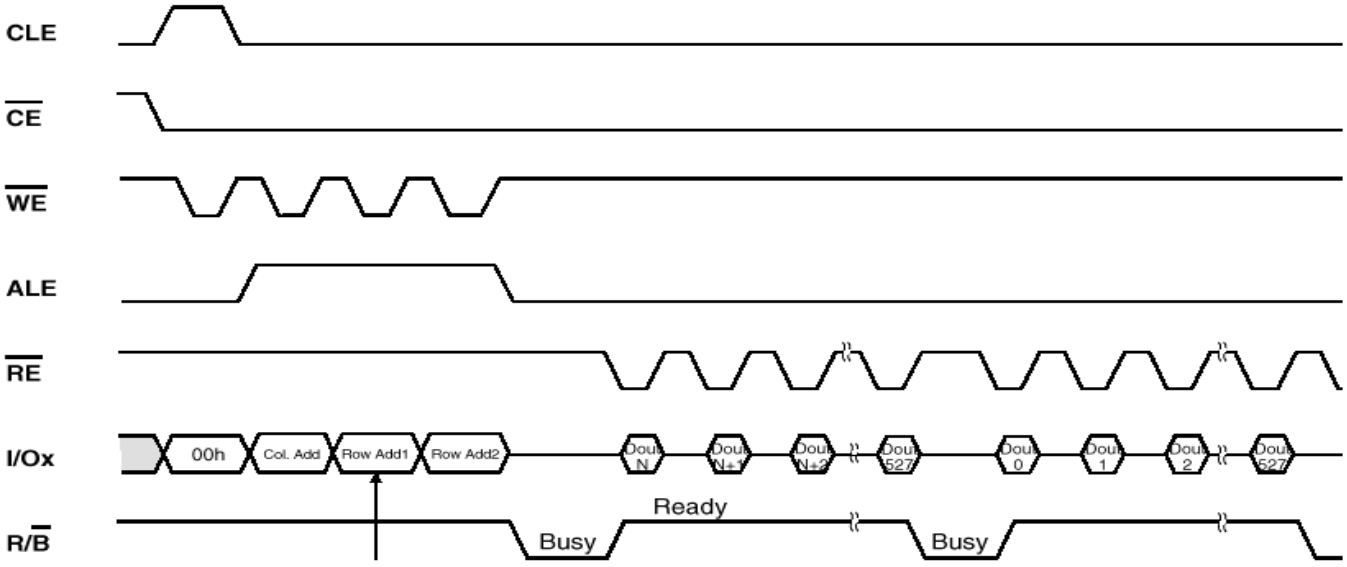


Figure 19-21. Read Operation

PAGE PROGRAM OPERATION

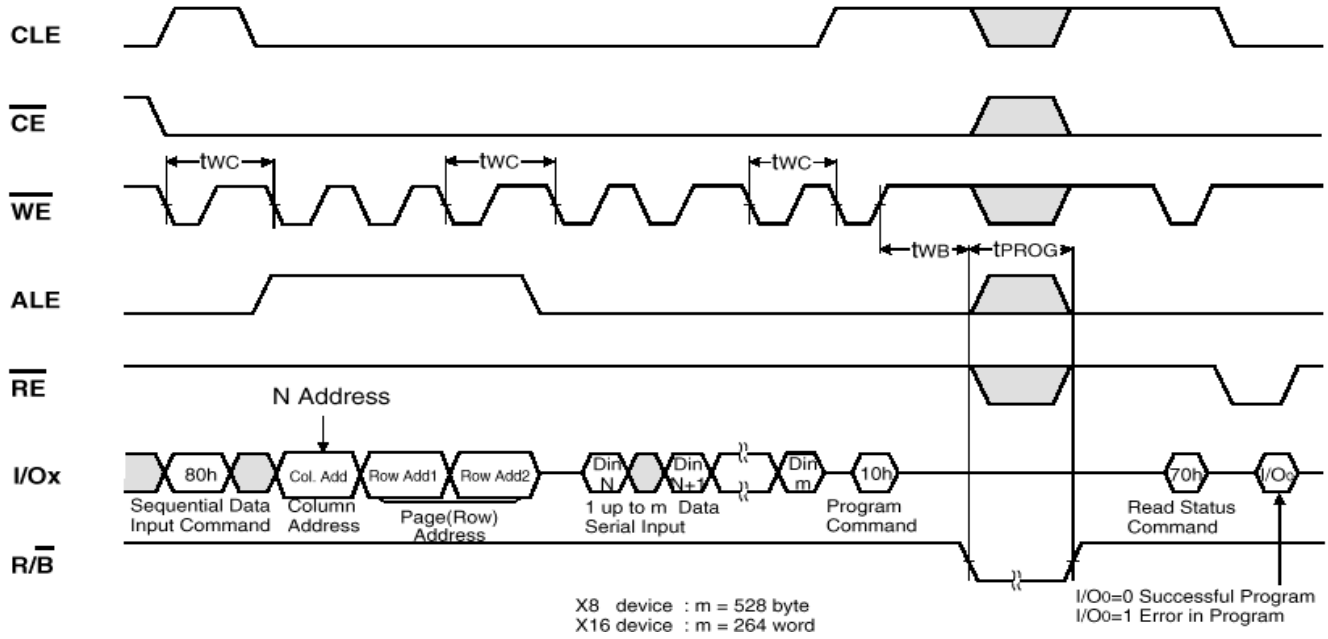


Figure 19-22. Program Operation

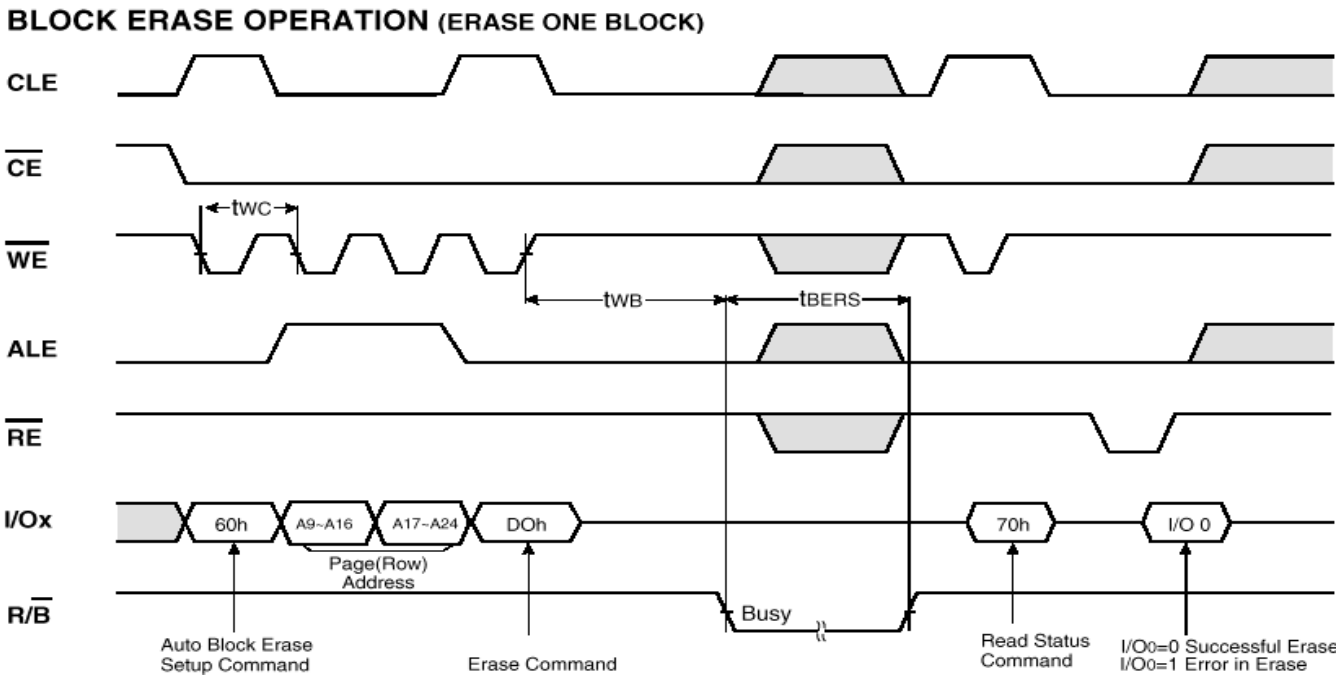


Figure 19-23. Erase Operation

19.8.4 Error Code Correction (ECC) Control

Error Code Correction (ECC) block is responsible for correcting one bit per read, and for detecting if there is more then one bit with an error.

When NFC accesses the NAND Flash device for Program operation, it generates code (24-bits for Main area data and 10-bits for Spare area data). When it accesses the NAND Flash device for a Read operation, it generates ECC code, and indicates how many errors were detected, and their positions in addition to correcting 1 error bit. The ECC code is updated by the NFC automatically. After a Read operation, AHB host can know whether there is an error or not by reading the status register (see ECC_Status_Result register in Section 19.7.6, “Controller Status and Result of Flash Operation (ECC_STATUS_RESULT)”). The indication in the status register is either a) no error, b) 1 bit error (correctable), or c) greater than 2 bit errors (uncorrectable). Since the generated ECC code at read/program operation is not updated to the internal RAM buffer, but is updated to the NAND Flash spare area upon program operation, the AHB host can read generated ECC code only from NAND Flash spare area.

19.8.5 Address Control

This module is responsible for address control and generation. It defines the RAM buffer Address Generation (RAM buffer Address for Data In/ Data Out). It generates and takes into account the Lock State Sequence (For more details see Section 19.9.3, “Write Protection Operation”) and therefore contains the Flash Memory Lock Address Comparator, and RAM buffer Lock Address Comparator which are used to determine if this area is protected or not. It also generates the RAM buffer Address for Boot Load and RAM buffer Address for Error Correction.

19.8.6 RAM Buffer (SRAM)

The internal RAM Buffer is a 2112 byte single Port RAM buffer which is a synchronous high performance design. This memory has 528 words of 32-bits each, from which 512 words are used for the main buffer and the remaining 16 Words are allotted to a spare area, which is used for ECC (Error Correction). This Memory is used as a BootRAM during boot from NAND Flash device, and as a buffer at normal operation.

19.8.7 Registers (Command, Address, Status, and Others.)

This module contains 15 registers of 16-bits each. Using these registers, the AHB host can control the NFC, read status on various operations and perform a direct access of commands, and insert addresses to the NAND Flash device. For more details, refer to Section 19.6.1, “Memory Map.”

19.8.8 Read and Write Control

The Read and Write Control Block contains a connection to the Internal bus (which is connected to the Internal RAM buffer and the registers). This Internal bus is responsible for the Internal Synchronous read and Asynchronous write. It supports Burst Read Latency (3, 4, 5, 6, 7 cycle) and Synchronous Read Burst Length (4, 8, 16, 32, continuous word). It is also responsible for RAM buffer Control and Register Control, RAM buffer Lock Control and Address and Data latches.

19.8.9 Data Output Control

This module defines Data output of 16-bits to the Internal bus which is driven to the AHB interface. It includes RAM buffer Data output, Register Data output and RAM buffer Synchronization for the read mode pipeline.

19.8.10 Host Control

This module defines Host control which is connected to the AHB Interface though the internal bus. It detects Chip Enable and controls the Reset and Output Enable, and generates the $\overline{\text{SRAM\_WE}}$ signal.

19.8.11 AHB BUS INTERFACE

AHB bus interface is an adapter between ABMA AHB bus and the internal bus. On the AHB bus side, it supports a 16-bit and 32-bit bus width, burst and non burst operations. On the internal bus side, it supports 32-bit bus width with a Synchronous Burst Read and an Asynchronous Random Write. It also supports Programmable Read latency for the internal bus (this also effects the latency on the AHB bus).

19.8.11.1 Big/Little Endian

AHB bus interface supports both Big and Little Endian data types. The nfc_endian pin controls the endian mode. Only the AHB side is controlled by nfc_endian pin; NAND Flash device side is always in little endian mode.

19.8.11.2 Burst Access Support

When a data transaction from the AHB is a burst, it will create a synchronous burst read on the internal bus for read cycles, and several asynchronous random writes for write cycles. Table 19-27 lists the NFC supported access burst types.

Table 19-27. NAND Flash Burst Access Support

HBURST	BURST TYPE	SUPPORTED	Description
000	SINGLE	Yes	Single transfer
001	INCR	Yes	Incrementing burst
010	WRAP4	No	4-beat wrapping burst
011	INCR4	Yes	4-beat incrementing burst
100	WRAP8	No	8-beat wrapping burst
101	INCR8	Yes	8-beat incrementing burst
110	WRAP16	No	16-beat wrapping burst
111	INCR16	Yes	16-beat incrementing burst

NOTE

NFC supports bursts of 16/32-bit words only. Bursts of byte words (8-bits) is not supported.

19.8.12 I/O Pins Sharing

NFC has logic that allows it to share I/O pins with pins of another memory controller. NFC's state machine halts when a request to free the pins is asserted. The NAND Flash signals when it finishes the existing transfer allowing the other memory controller to be able to control them. Since the NAND Flash accesses are long and relatively slow, the priority is given to the other memory controller and the NAND Flash Controller will have to wait till the pins are free before it can continue with its accesses.

One example for this pin muxing is sharing the 16 I/O pins of the NAND Flash Controller with the Data pins of the WEIM when interfacing to the PSRAM.

19.9 Initialization/Application Information

This section describes how to operate the NFC using its registers and its interrupts, and is divided into the following subjects:

- Normal operation–In order to operate a NAND Flash device using the NFC the user should use the instructions in Section 19.9.1, "Normal Operation."
- ECC operation–ECC operation is used when an error is detected.
- Write protection operation (both to the internal memory and the flash device)–Write protection is used when the programmer wishes to protect part of the NAND Flash device memory from being written except in certain cases. There are two levels of protection: software (for frequently-changed memory locations), and hardware (for memory locations whose contents are rarely changed).

19.9.1 Normal Operation

"Normal Operations" are composed of fundamental building block operations (in Section 19.9.1.1, "Fundamental Building Block Operations"), in addition to specific operations, as shown in the flowcharts below (in Section 19.9.1.2, "NAND Flash ID Read Operation" to Section 19.9.1.7, "Hot Reset (Controller and NAND Flash Reset)").

19.9.1.1 Fundamental Building Block Operations

19.9.1.1.1 Preset Operation

Figure 19-24 provides a flowchart of the NAND Flash preset operation.

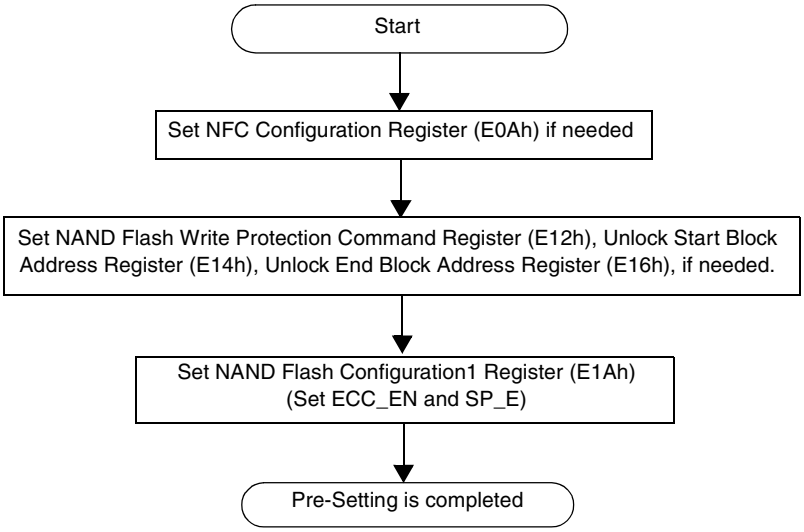


Figure 19-24. Flowchart of Preset Operation

19.9.1.1.2 NAND Flash Command Input Operation

Figure 19-25 provides a flowchart of the NAND Flash Command Input operation.

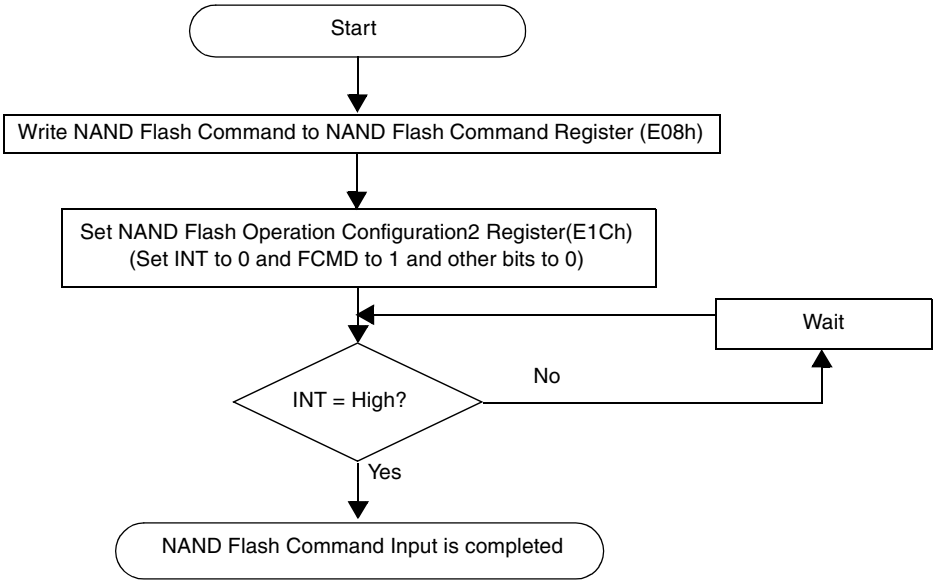


Figure 19-25. Flowchart of NAND Flash Command Input Operation

19.9.1.1.3 NAND Flash Address Input Operation

Figure 19-26 provides a flowchart of the NAND Flash Address Input operation.

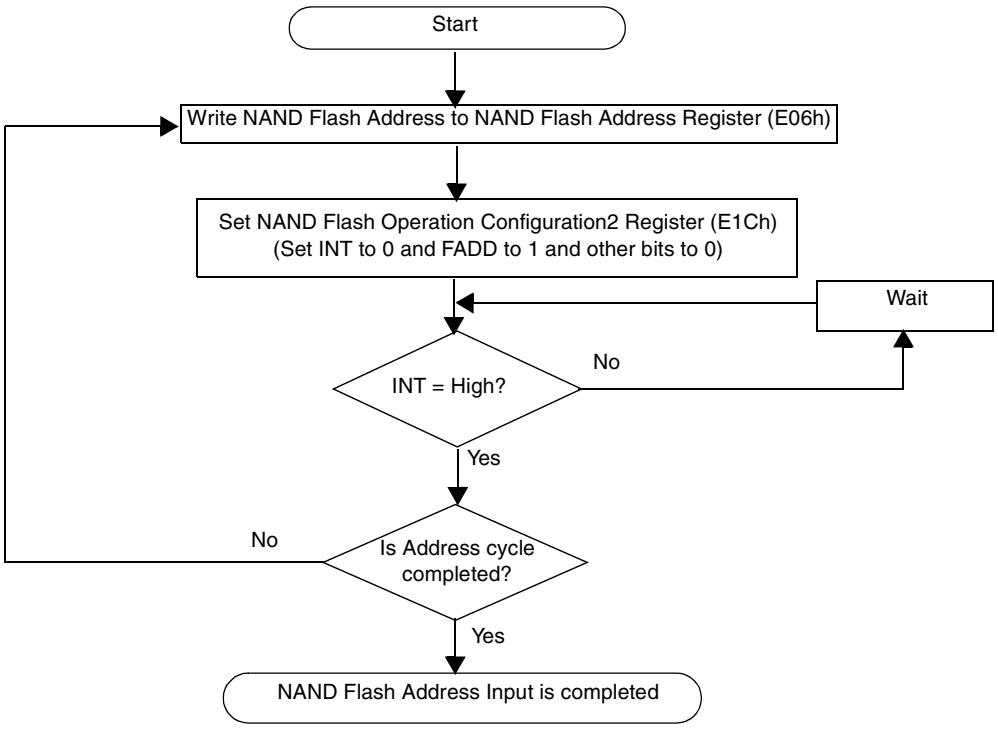


Figure 19-26. Flowchart of NAND Flash Address Input Operation

19.9.1.1.4 NAND Flash Data Input Operation

Figure 19-27 provides a flowchart of the NAND Flash Data Input operation.

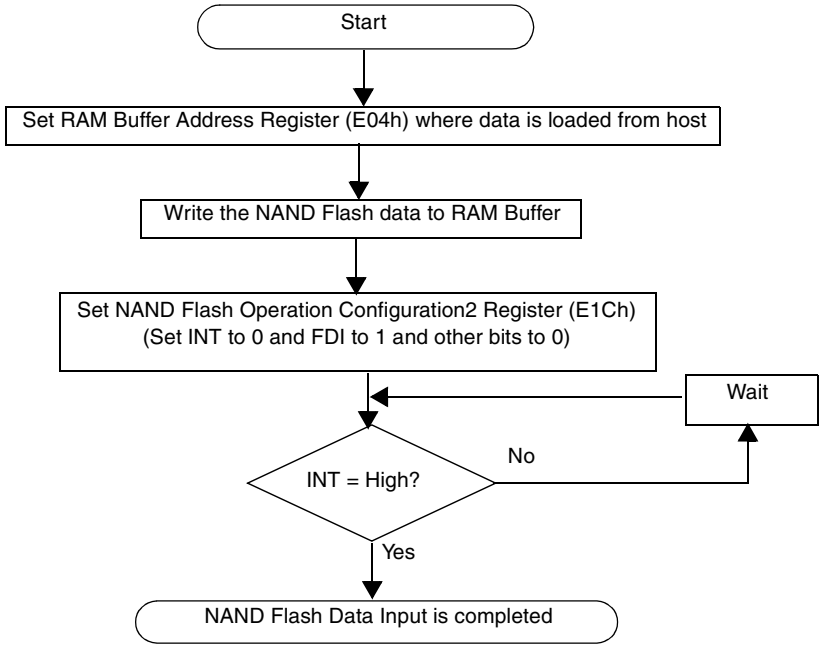


Figure 19-27. Flowchart of NAND Flash Data Input Operation

19.9.1.1.5 NAND Flash Data Output Operation

Figure 19-28 provides a flowchart of the NAND Flash Data Output operation.

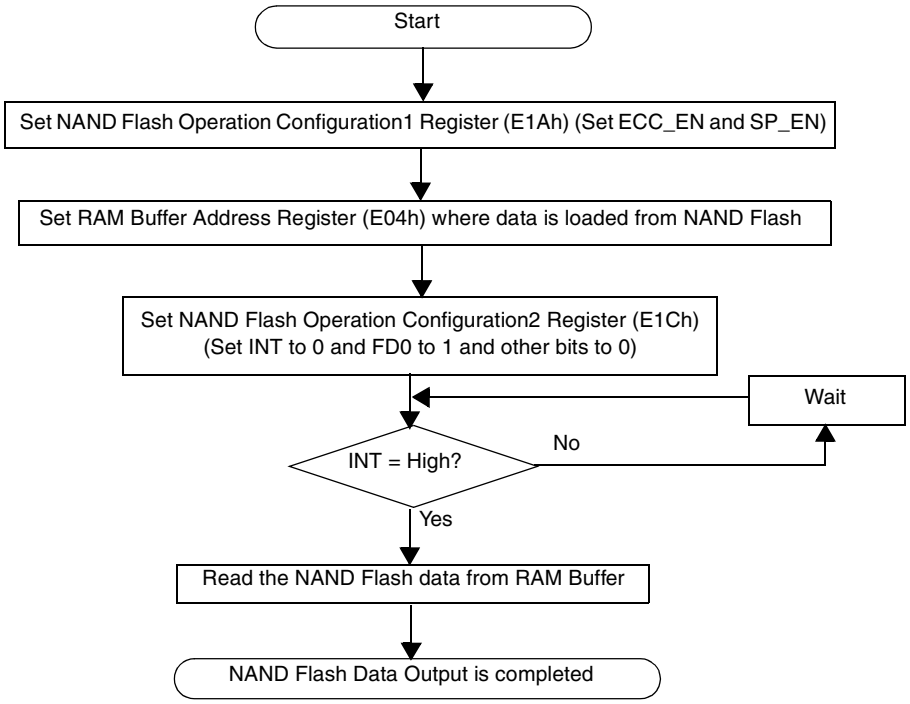


Figure 19-28. Flowchart of NAND Flash Data Output Operation

19.9.1.2 NAND Flash ID Read Operation

Figure 19-29 provides a flowchart of the NAND Flash ID Read operation.

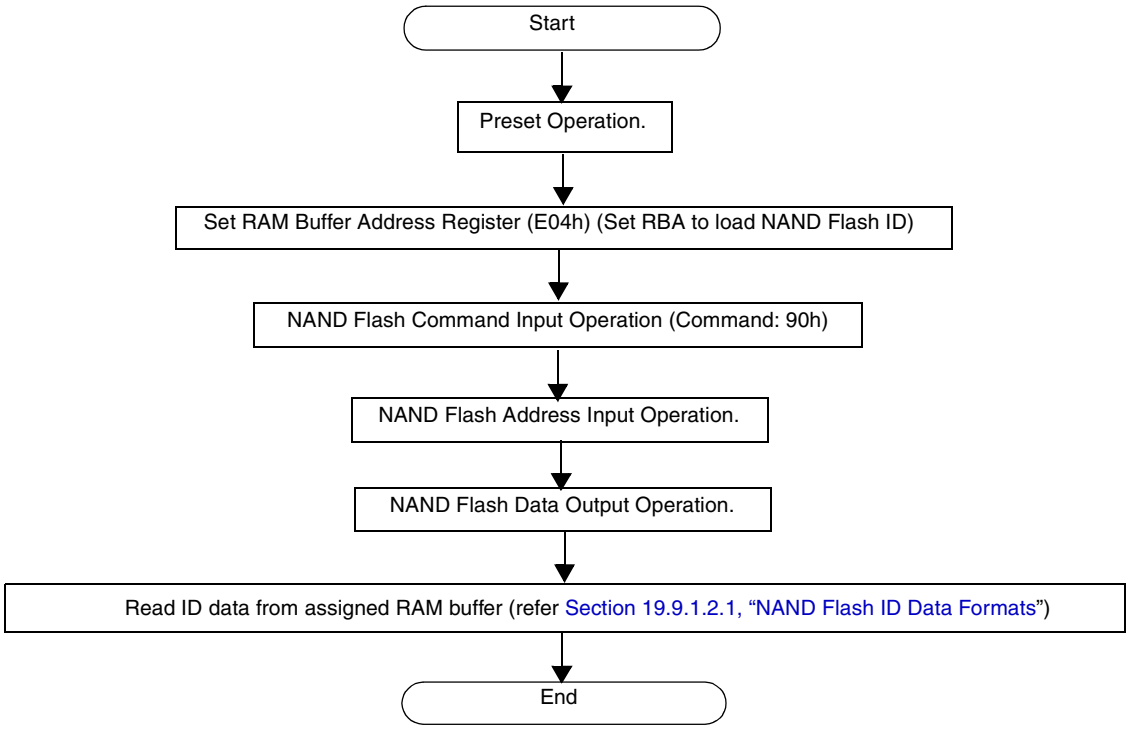


Figure 19-29. Flowchart of Read NAND Flash ID Operation

19.9.1.2.1 NAND Flash ID Data Formats

Format of NAND Flash ID data stored in RAM buffer (for X8 org. NAND Flash) is shown in Figure 19-30.

RAM Buffer of RBA address				1st Half-Word				2nd Half-Word				3rd Half-Word				
				1st byte of ID			2nd byte of ID	3rd byte of ID		4th byte of ID		5th byte of ID		6th byte of ID		
				LSB			MSB									

Figure 19-30. NAND Flash ID Data Format (x8)

The format of NAND Flash ID data stored in RAM buffer (for X16 org. NAND Flash) is shown in Figure 19-31.

RAM Buffer of RBA address				1st Half-Word				2nd Half-Word				3rd Half-Word				
				1st byte of ID		XXh		2nd byte of ID		XXh		3rd byte of ID		XXh		
				LSB			MSB									

RAM Buffer of RBA address				4th Half-Word				5th Half-Word				6thHalf-Word				
				4th byte of ID		XXh		5th byte of ID		XXh		6th byte of ID		XXh		
				LSB			MSB									

Figure 19-31. NAND Flash ID Data Format (x16)

19.9.1.3 NAND Flash Status Read Operation

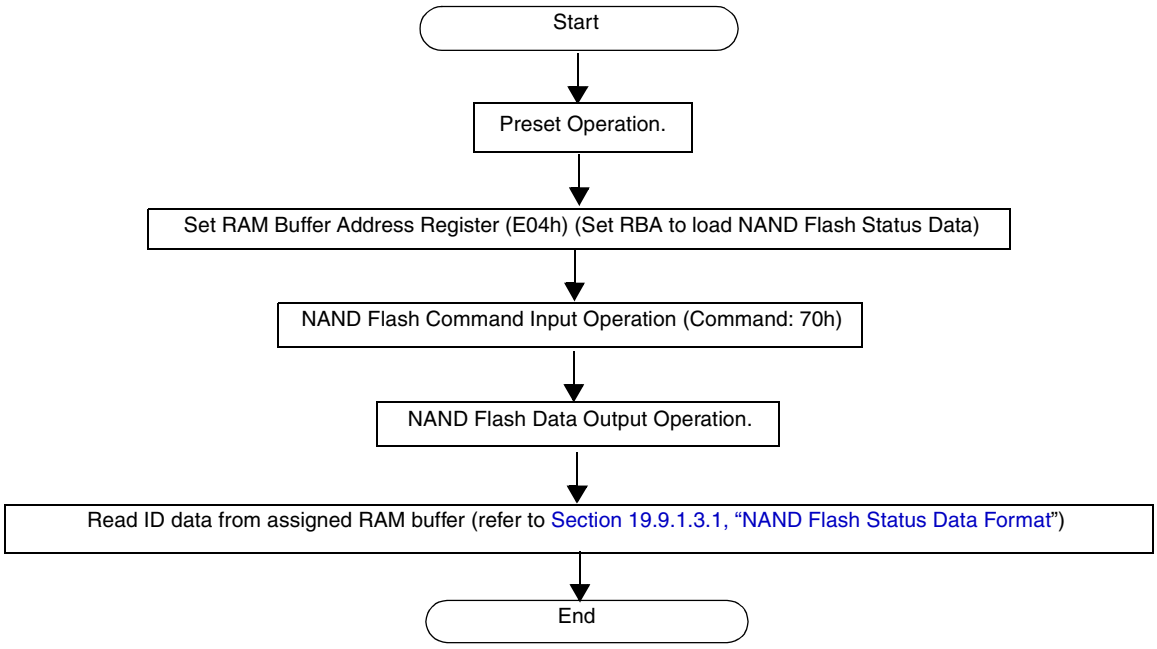


Figure 19-32. Flowchart of Read NAND Flash Status Operation

19.9.1.3.1 NAND Flash Status Data Format

The assignment of NAND Flash Status data stored in the RAM buffer (for both X8/X16 org. NAND Flash) is shown in Figure 19-33.

RAM Buffer of RBA address	1st half-word				- - - - - - - -
	1st byte of Status		XXh		
	LSB			MSB	

Figure 19-33. NAND Flash Status Data Format

19.9.1.4 Read NAND Flash Data Operation

Figure 19-34 shows a flowchart of the read NAND Flash data operation.

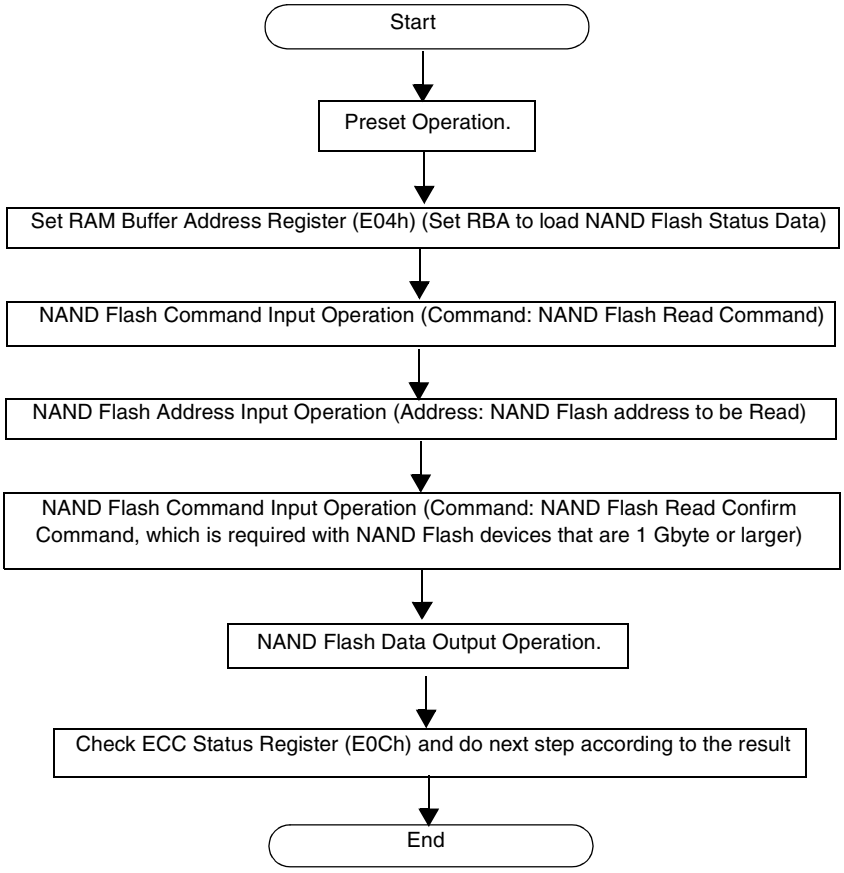


Figure 19-34. Flowchart of Read NAND Flash Data Operation

19.9.1.5 Program NAND Flash Data Operation

Figure 19-35 shows a flowchart of the program NAND Flash data operation.

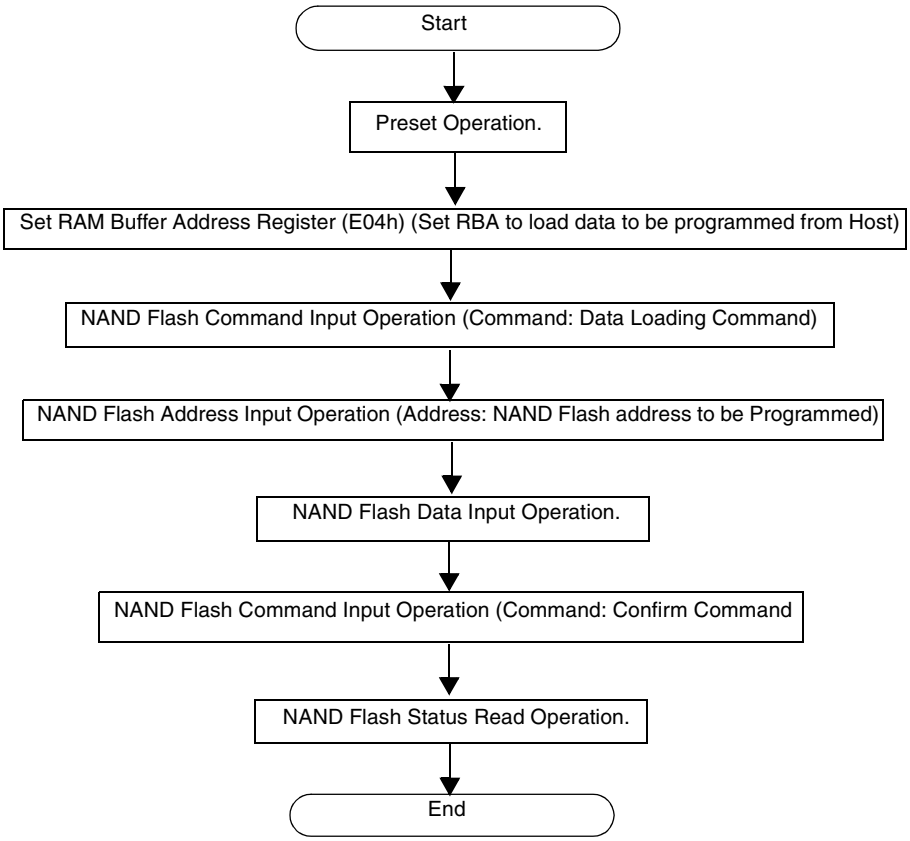


Figure 19-35. Flowchart of Program NAND Flash Data Operation

19.9.1.6 Erase NAND Flash Data Operation

Figure 19-36 shows a flowchart of the erase NAND Flash data operation.

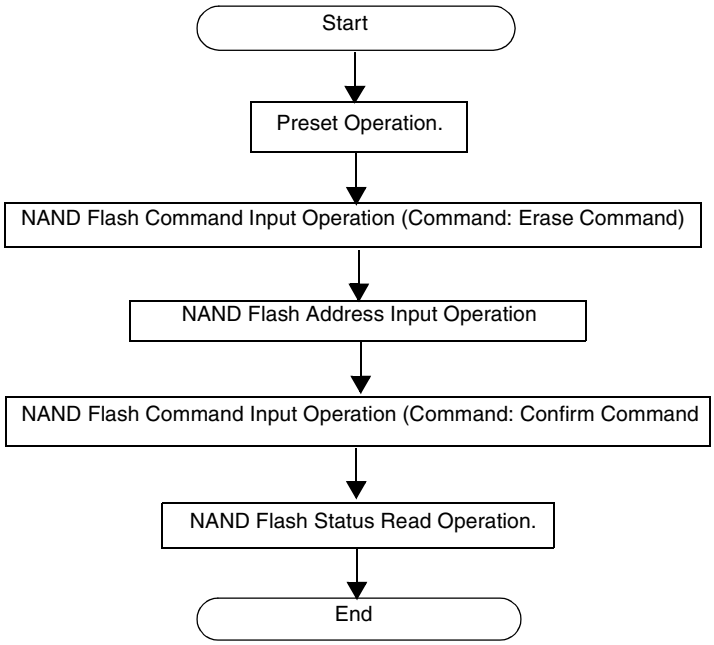


Figure 19-36. Flowchart of Erase NAND Flash Operation

19.9.1.7 Hot Reset (Controller and NAND Flash Reset)

A warm (or “hot”) reset causes the NFC and the NAND Flash device cease their current operation and causes the internal registers to revert to their default state. Figure 19-37 shows a flowchart of a hot reset operation.

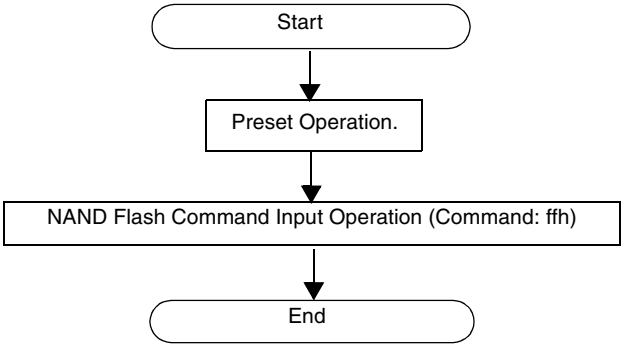


Figure 19-37. Flowchart of Hot Reset Operation

19.9.2 ECC Operation

19.9.2.1 ECC Normal Operation

When NFC accesses the NAND Flash device for Program operation, it generates ECC code (24 bits for Main area data and 10 bits for spare area data). When NFC accesses the NAND Flash device for a Read

operation, it generates ECC code, detects the error number and position and corrects a 1-bit error, if applicable. Table 19-28 shows ECC code assignment of NAND Flash spare area. This ECC code is updated by NFC automatically. After Read operation, AHB host can know whether there is error or not by reading the status register (see ECC_Status_Result register in Section 19.7.6, "Controller Status and Result of Flash Operation (ECC_STATUS_RESULT)"). Error type can be: a) no error, b) 1bit error (correctable), or c) 2 or more bit error (uncorrectable). Since generated ECC code at read/program operation is not updated to the internal buffer RAM, but is updated to NAND Flash spare area immediately upon program operation, AHB host can read generated ECC code only from NAND Flash spare area.

19.9.2.2 ECC Bypass Operation

In ECC bypass operation, NFC generates an ECC result which indicates error position (refer ECC Result table), but doesn't correct the error. After a Read operation, host can know whether there is an error or not by reading the status register (refer ECC status register table). Error type is divided into no error, 1bit error (correctable), or 2bits error (uncorrectable). In 1bit error case, the Host can correct the error by itself after reading the ECC Result register (see ECC_Status_Result register in Section 19.7.6, "Controller Status and Result of Flash Operation (ECC_STATUS_RESULT)").

Table 19-28. ECC Code/Result Readability

Operation	Read Operation		Program Operation	
	ECC Code from Spare Area Buffer	ECC Result from Register	ECC Code from Spare Area Buffer	ECC Result from Register
ECC operation	Invalid (Pre-written ECC code [1])	Valid	Invalid (old data [2])	—
ECC bypass	Invalid (Pre-written ECC code)	Valid	Invalid (old data)	—

1 Pre-written ECC code: ECC code which is previously written to NAND Flash spare area in program operation.

2 Old data: ECC code is not updated to spare buffer, so ECC code placement of spare buffer remains old data.

19.9.2.3 How to Operate ECC

In order to generate ECC and carry out correction by NFC, Program and Read with ECC operation. In order to generate ECC by NFC and carry out correction by AHB host, Program with ECC operation and Read without ECC operation

NOTE

AHB host can read ECC results from ECC_Status_Result register (see Section 19.7.6, "Controller Status and Result of Flash Operation (ECC_STATUS_RESULT)") after a read operation in both ECC Normal operation and ECC Bypass cases. When NFC reads NAND Flash data, ECC code for read data is ***not*** updated into RAM buffer.

19.9.3 Write Protection Operation

NFC offers a software write protection feature, and a hardware write protection feature.

19.9.3.1 Write Protection for RAM Buffer (LSB 1 Kbyte)

NFC offers a software write protection feature for the first 2 pages (main + spare area data) of RAM buffer, which protects RAM buffer data. This write protection is carried out by setting BLS bit of the NFC_CONFIGURATION register. The default state is locked state, and first 2 pages go to this state after a cold or warm reset. Write protection availability for main/spare memory regions in the RAM buffer are described on Table 19-29. A state diagram of RAM buffer write protection is shown in Figure 19-38.

Table 19-29. Write Protection for Main/Spare RAM Buffer

Main Area	Spare Area	
1st page RAM buffer		Write Protection Available
2nd page RAM buffer		
3rd page RAM buffer		Write Protection Not available
4th page RAM buffer		

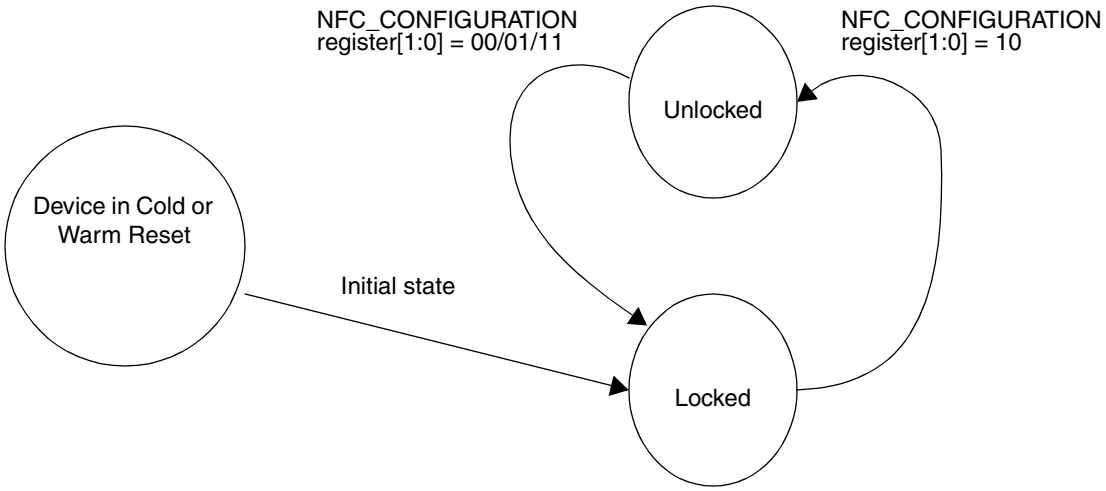


Figure 19-38. State Diagram of RAM Buffer Write Protection

19.9.3.2 Write Protection Modes

NFC offers both hardware and software Write Protection options for the NAND Flash device. Software Write Protection feature is used by executing LOCK BLOCK command or LOCK-TIGHT BLOCK command, and Hardware Write Protection feature is used by executing a cold or warm reset. The WP signal is asserted only upon POR.

19.9.3.3 Write Protection Commands

There are two write protection states: Locked and Lock-Tight.

- Locked state means that memory block in question is write protected (it cannot be written to), but UNLOCK command can “un”-lock it. Useful for frequently changed memory blocks.

- Lock-Tight state is a higher level of protection, and means that memory block in question is write protected, but UNLOCK command cannot unlock it. Useful for memory blocks whose contents are rarely changed.

The following summarizes the locking functionality:

- All blocks power-up in a locked state. The UNLOCK command can unlock these blocks.
- LOCK-TIGHT BLOCK command locks blocks and prevents it (them) from being unlocked.
- Lock-Tight state can be reverted to locked state only when Cold/Warm reset is executed.
- Writing to unlock start/end address registers (Unlock_Start_Blk_Add and Unlock_End_Blk_Add) while NFC is in Lock-Tight state does not affect the unlock address.

19.9.3.4 Write Protection Status

The current Write Protection status of the NFC can be read in NAND Flash Write Protection status register (NAND_Flash_WR_Pr_St). There are three bits: US,LS, and LTS, which are not cleared by hot reset. These Write Protection status bits are updated as soon as the Write Protection command is entered. Figure 19-39 shows a state diagram for the write protection of the NFC.

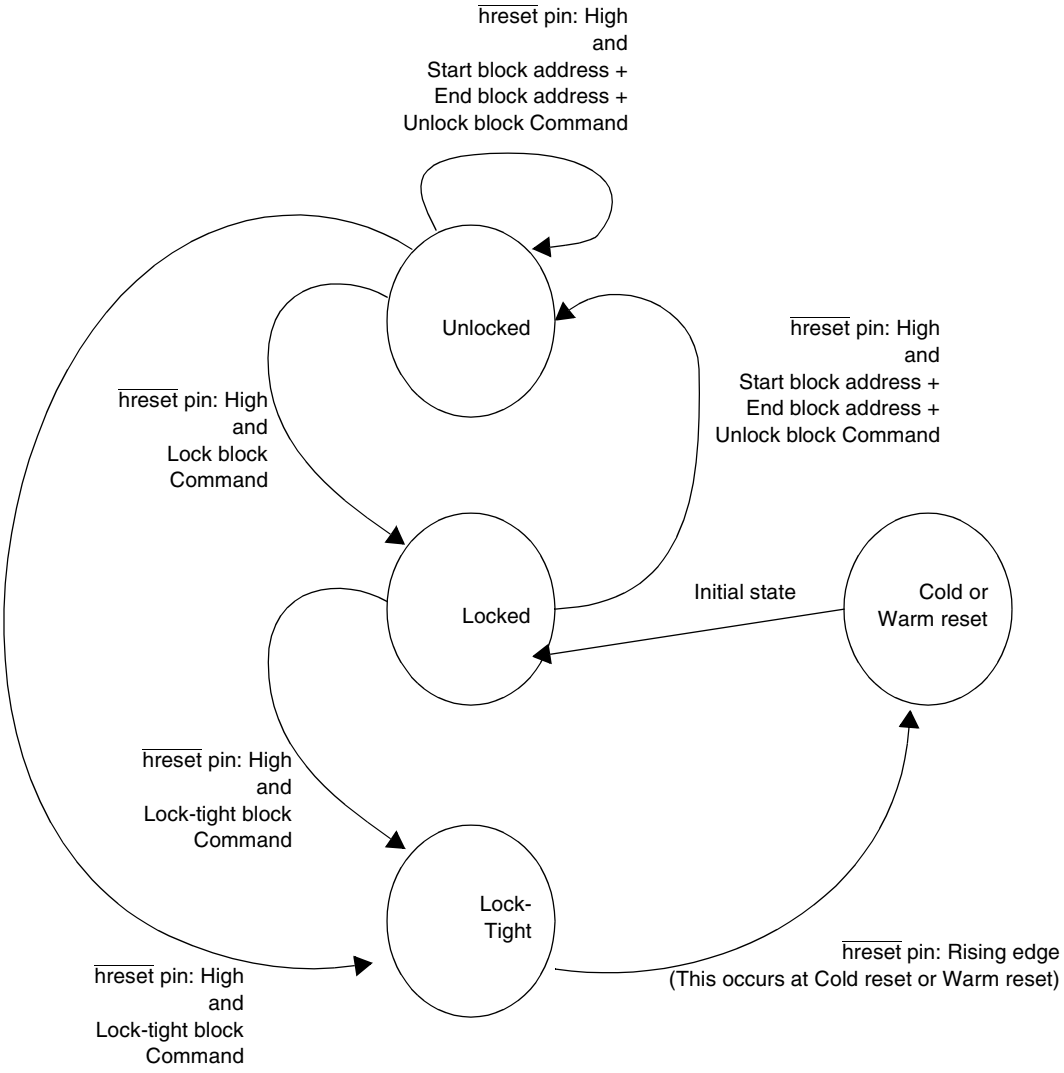


Figure 19-39. State Diagram of NAND Flash Write Protection

19.9.3.4.1 Lock Sequence

The following describes the “lock” sequence:

1. Command Sequence: LOCK BLOCK Command (02h)
2. All blocks default to locked after initial Cold reset or Warm reset.
3. Locking some of the blocks is not available; all memory blocks are locked upon reset.
4. Unlocked memory blocks can be locked by using the LOCK BLOCK command. Status of a locked memory block can be changed to Unlocked or Lock-tight using appropriate software commands.

19.9.3.4.2 Unlock Sequence

The following describes the “unlock” sequence:

1. Command Sequence: Start block address + End block address + UNLOCK BLOCK Command(04h)

2. Unlocked blocks can be programmed or erased.
3. Status of unlocked block can be changed to *locked*/*lock-tight* using appropriate software command.
4. Only one sequential area can be released to unlocked state from locked state; Unlocking multi-areas is not available.

19.9.3.4.3 Lock-tight Sequence

The following describes the “lock-tight” sequence:

1. Only locked blocks can be “locked-tight” by the `LOCK-TIGHT BLOCK` command.
2. Command Sequence: `LOCK-TIGHT BLOCK` Command (01h)
3. Unlocking multi area is not available
4. Lock-tight blocks revert to the locked state at Cold/Warm reset.

19.9.4 Memory Configuration Examples

Table 19-30 shows NFC Pin configurations for various NAND Flash devices. Figure 19-40 and Figure 19-41 show memory connection for various 8-bit and 16-bit configuration.

Table 19-30. Examples for NFC Pin Configuration for Selected Memory Devices

Device	BOOT	NFC_FMS	$\overline{\text{NF8BOOT}}$	$\overline{\text{NF16BOOT}}$	NF_16BIT_SEL
SAMSUNG K9F5608 (32M x 8-bit) Page size is 528 bytes.	NO	0	1	1	0
	YES	0	0	1	X
SAMSUNG K9F5616 (16M x 16bit) Page size is 528 bytes.	NO	0	1	1	1
	YES	0	1	0	X
SAMSUNG K9F1G08 (128M x 8-bit) Page size is 2112 bytes.	NO	1	1	1	0
	YES	1	0	1	X
SAMSUNG K9F1G16 (64M x 16bit) Page size is 2112 bytes.	NO	1	1	1	1
	YES	1	1	0	X

NOTE

The NFC can support High Speed (HS) NAND Flash by supplying higher frequencies (up to 50 MHz) to the Flash Clock input.

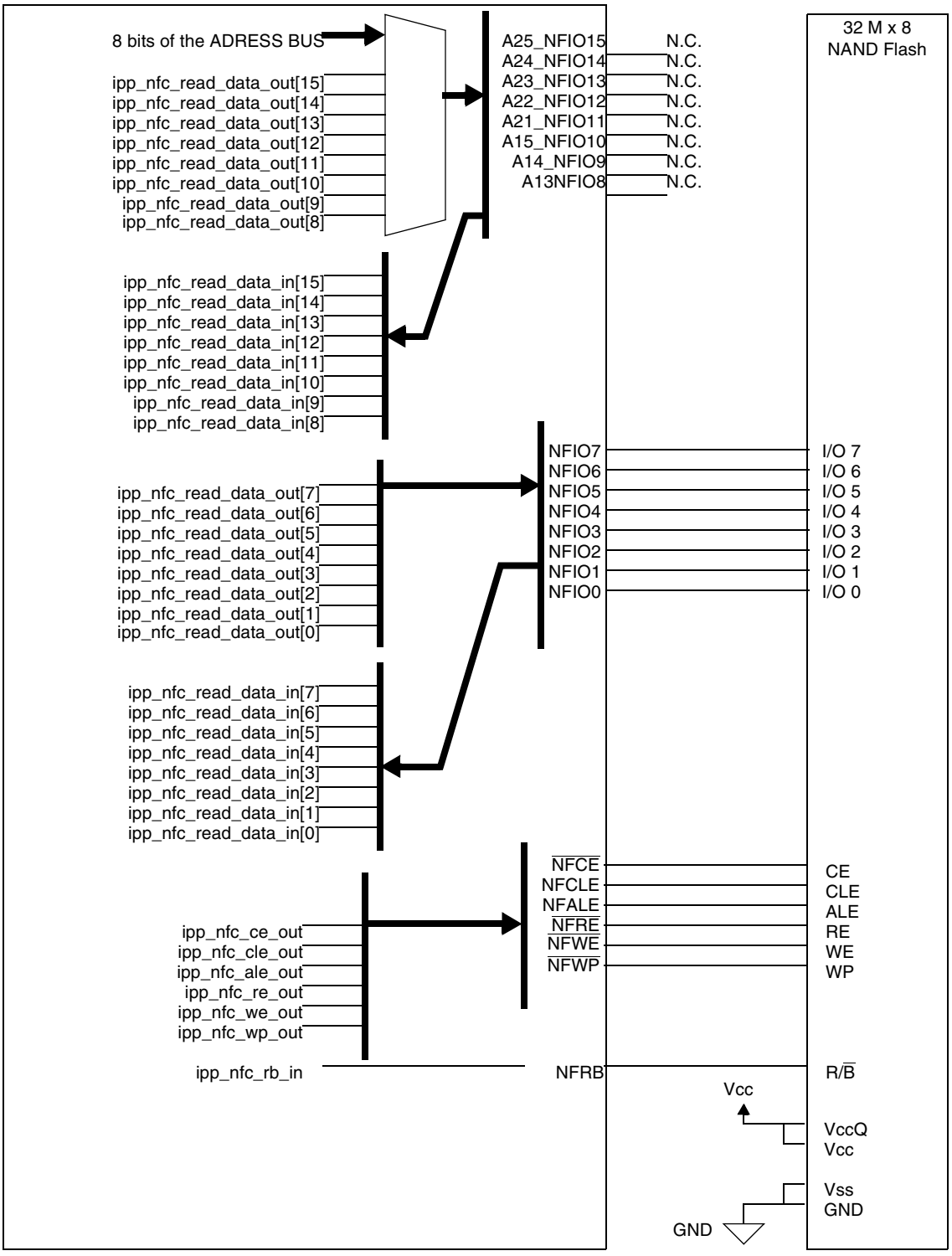


Figure 19-40. 256-Mbit (32 Mbit x 8 Bit) NAND Flash Connection Diagram

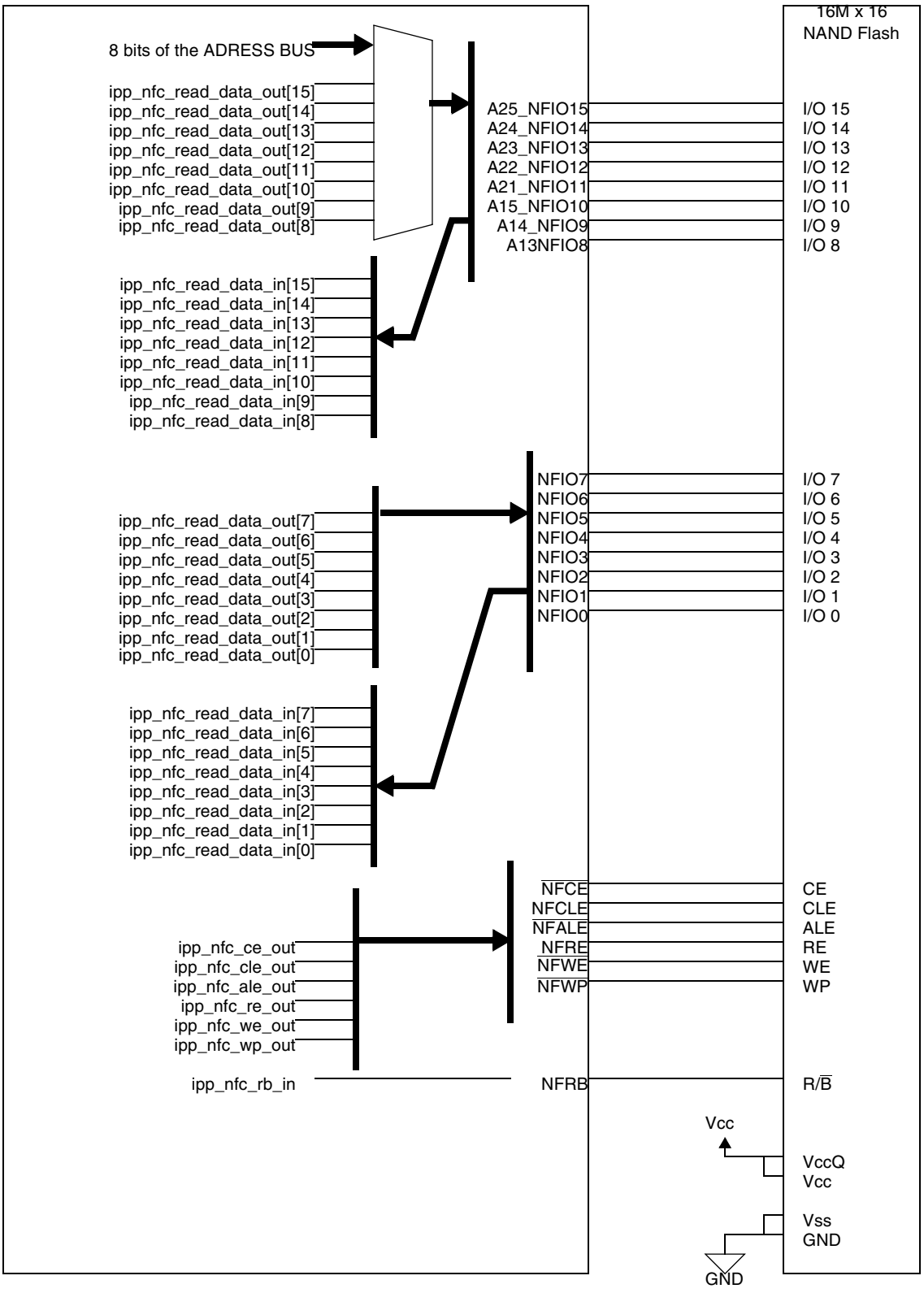


Figure 19-41. 256 Mbit (16 M x 16 Bit) NAND Flash Connection Diagram

Chapter 20
Personal Computer Memory Card International Association (PCMCIA) Controller

This chapter describes the Personal Computer Memory Card International Association (PCMCIA) controller for the i.MX27 processor. The association standard is PCMCIA 2.1, which defines the use of memory and I/O devices as insertable and exchangeable peripherals for personal computers or PDAs. Examples of these types of devices include compact flash and WLAN adapters.

20.1 Overview

The PCMCIA host adapter module provides the control logic for PCMCIA socket interfaces, and requires some additional external analog power switching logic and buffering. The additional external buffers allow the PCMCIA host adapter module to support one PCMCIA socket.

Figure 20-1 shows the PCMCIA controller block diagram.

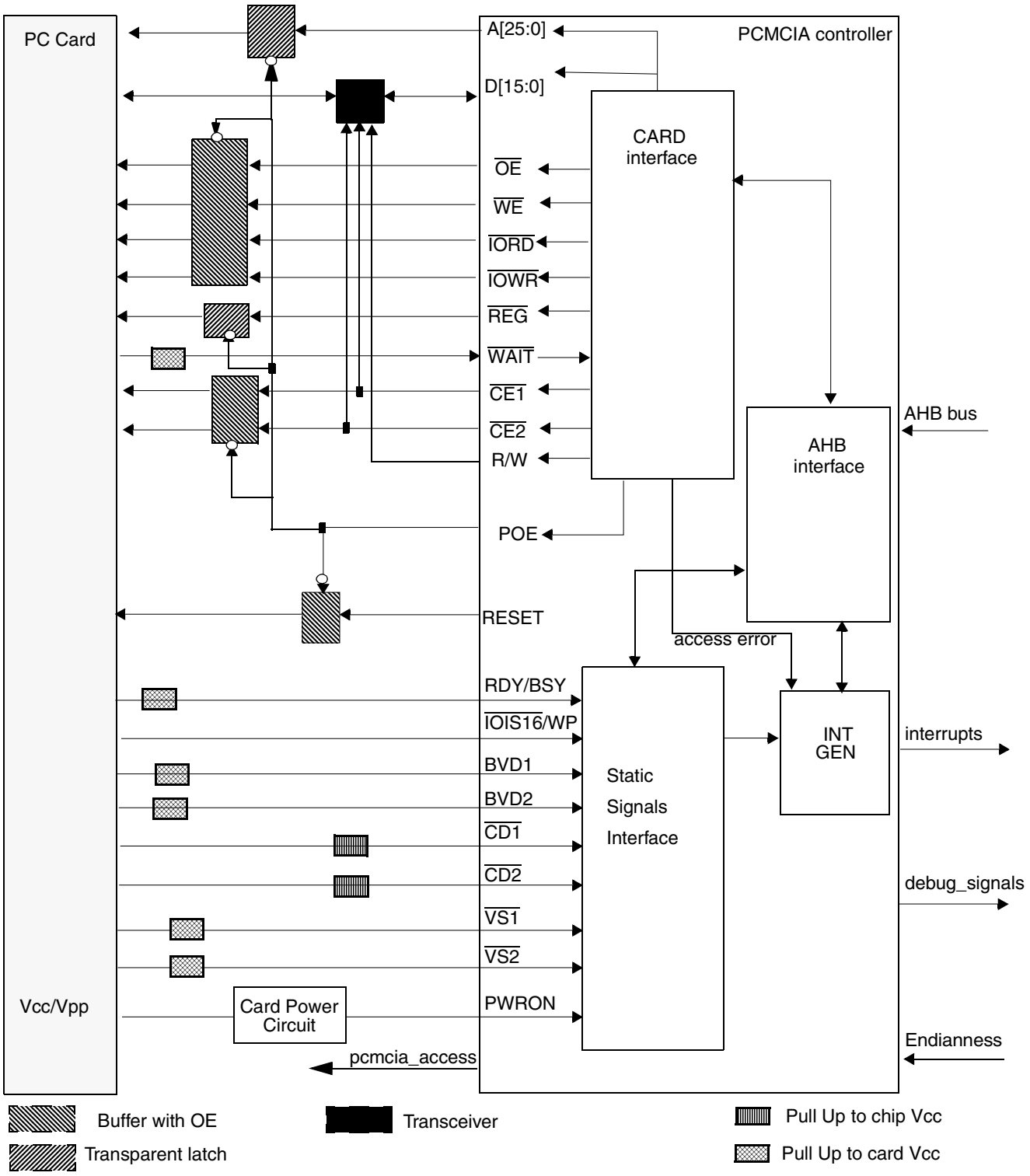


Figure 20-1. PCMCIA Controller Interface Block Diagram

20.2 Features

The PCMCIA controller includes the following features:

- A host adapter interface fully compliant with the PCMCIA standard release 2.1 (PC Card -16).
  - — Supports one PCMCIA socket
  - — Supports hot-insertion
  - — Supports card detection
  - — Provides mappings to common memory space, attribute memory space, and I/O space. Each space is up to 64 Mbyte in size.
  - — Supports 5 memory windows
  - — Generates a single interrupt to the ARM9 core
  - — Provides fully programmable PC card access timing
  - — Handles interrupts from the card
- The PCMCIA controller is part of the EMI complex and shares its pins with the EIM, SDRAMC, and NAND flash controller.
- Supports ATA disk emulation

20.3 External Signal Description

20.3.1 Detailed Signal Descriptions

Table 20-2 shows the PCMCIA signal descriptions for the pins that are used to control the PCMCIA interface.

Table 20-2. PCMCIA Signal Descriptions

Signal	In/Out	Description
Standard Pins		
A[25:0]	Output	Address Bus. These address bus output lines allows direct addressing of up to 64 Mbytes of linear memory on the PCMCIA card.
D[15:0]	I/O	Data Bus. Bidirectional. PCMCIA socket data I/O pins.
$\overline{CE1}$, $\overline{CE2}$	Output	Card Enable. When a PCMCIA access is performed, $\overline{CE1}$ enables even bytes, $\overline{CE2}$ enables odd bytes. See also Section 20.5.9, “Data and Control Signals Relations” and Table 20-20.
$\overline{OE}$	Output	Output Enable. During PCMCIA accesses, $\overline{OE}$ is used to drive memory read data from a PC card in a PCMCIA socket.
$\overline{WE}$	Output	Write Enable. Program during PCMCIA access, $\overline{WE}$ is used to latch memory write data to the PC card in a PCMCIA socket. Can also be used as the programming strobe for PC cards using programmable memory technologies.
$\overline{REG}$	Output	Register Accesses Attribute Memory Select. When $\overline{REG}$ is asserted during PCMCIA access, card access is limited to attribute memory when a memory access occurs ($\overline{WE}$ or $\overline{OE}$ are asserted) and to I/O ports when I/O access occurs ($\overline{IORD}$ or $\overline{IOWR}$ are asserted). if REG is asserted, accesses to common memory are blocked.

Table 20-2. PCMCIA Signal Descriptions (continued)

Signal	In/Out	Description
$\overline{\text{IORD}}$	Output	I/O Read. This output goes active (low) for I/O reads from the socket. This signal is asserted together with $\overline{\text{REG}}$ and it is used to read data from the PC card I/O space. $\overline{\text{IORD}}$ is valid only when $\overline{\text{REG}}$ and either $\overline{\text{CE1}}$ or $\overline{\text{CE2}}$ signals are also asserted.
$\overline{\text{IOWR}}$	Output	I/O Write. This output goes active (low) for I/O write to the socket. Asserted with $\overline{\text{REG}}$ during PCMCIA accesses, used to latch data into the PC cards I/O space. $\overline{\text{IOWR}}$ is valid only when $\overline{\text{REG}}$ and either $\overline{\text{CE1}}$ and $\overline{\text{CE2}}$ signals are also asserted.
$\overline{\text{WAIT}}$	Input	Extend bus cycle. Input. Asserted by the PC card to delay completion of the pending memory or I/O cycle.
$\overline{\text{IOIS16}}$/WP	Input	I/O port is 16-bits. When the card and its socket are programmed for I/O interface operation, this signal is used as $\overline{\text{IOIS16}}$ and must be asserted by the PC card when the address on the bus corresponds to an address on the PC card and the I/O port being addressed supports 16-bit accesses. If the I/O region in which the address resides is programmed as 8-bit wide $\overline{\text{IOIS16}}$ is ignored. Write Protect. When the card and socket are programmed for memory interface operation, this signal is used as WP. It reflects the state of the write protect of the PC card. The PC card must assert WP when the card switch is enabled. It must be negated when the switch is disabled. For a PC card that is writable without a switch, WP must be connected to ground.If the PC card is permanently write-protected, WP must be connected to Vcc.
$\overline{\text{VS1}}$, $\overline{\text{VS2}}$	Input	Voltage sense. Input. Generated by the card to notify the socket of the card's CIS VCC requirements.
$\overline{\text{CD1}}$, $\overline{\text{CD2}}$	Input	Card Detect. Provide proper detection of card insertion. They must be connected to ground internally on the PC card, thus, these signals are forced low when a card is placed on the socket. These signals must be pulled up to system Vcc to allow card detection to function when the card socket is powered down.
BVD1/$\overline{\text{STSCHG}}$, BVD2/$\overline{\text{SPKR}}$	Input	These two lines can be used for battery voltage detection or status change/speaker. Battery Voltage Detect. When the card and its socket are programmed for memory interface operation, these signals are generated by the PC card with on board battery to report the battery condition. See Table 20-3 for a description of the logical combinations representing battery condition. Status Change. When the card is in I/O interface operation, BVD1 is used as Status Change and is generated by the I/O PC card. Status Change must be held negated when the "signal on change" bit and the "change" bit in the card status register are either or both zero. STSCHG must be asserted when both bits = 1. Speaker. Input. When the card is in I/O interface operation BVD2 is used as Audio Digital Waveform. A card that does not this capability should drive $\overline{\text{SPKR}}$ high.
READY/$\overline{\text{IREQ}}$	Input	Ready. When the card and its socket are programmed for memory interface operation, this signal is used as RDY/BSY and must be asserted by a PC card to indicate that a PC card is busy processing a previous write command. When the card and its socket are programmed for I/O interface operation, this signal is used as $\overline{\text{IREQ}}$ and must be asserted by a PC card to indicate that a device on the PC card requires service by host software. Must be held negated when no interrupt is requested.
RESET	Output	Card Reset. Output. Provided to clear the card's configuration option register, thus placing the card in its default (memory only interface) state and beginning an additional card initialization. RESET signal has inverted polarity when in TrueIDE mode. See also Section 20.5.10, "True IDE Mode Access."

Table 20-2. PCMCIA Signal Descriptions (continued)

Signal	In/Out	Description
PCMCIA Controller Module Pins		
POWERON	Input	Power is On. The card supply circuitry can use this signal as an interrupt to notify when the card's power supply reaches the full required voltage.
R/W	Output	External Transceiver Direction. Negated during read cycles and asserted during write.
POE	Output	PCMCIA buffers output enable. An output line reflecting the value of PGCR[POE] bit. Used to three-state control signals and to latch the address. See Figure 20-1 for a simplified block diagram.
SPKROUT	Output	Speaker Output. Provides a digital audio waveform to be driven to the system's speaker. This signal is connected directly to the SPKR input.
Endianness	Input	Endianness control. Input. This input pin defines the Endianness mode of the module. '1' is Big Endian. This module should be tied high or low. See Section 20.5.13, "Endianness Support."
pcmcia_access	Output	This output signal is used to indicate that a valid access to the card is performed. This signal is used by the EMI muxing to select the pcmcia_if port and drive it to the pins.
$\overline{\text{ipi\_int\_pcmcia}}$	Output	This is the interrupt line from the pcmcia_if module. This signal is a logical OR of all interrupts generated by the pcmcia_if.
$\overline{\text{ipi\_int\_vs1}}$	Output	This interrupt is generated if the voltage sense #1 input signal from the card has changed.
$\overline{\text{ipi\_int\_vs2}}$	Output	This interrupt is generated if the voltage sense #2 input signal from the card has changed.
$\overline{\text{ipi\_int\_wp}}$	Output	This interrupt is generated if the write protect input signal from the card has changed.
$\overline{\text{ipi\_int\_cd1}}$	Output	This interrupt is generated if the card detect #1 input signal from the card has changed.
$\overline{\text{ipi\_int\_cd2}}$	Output	This interrupt is generated if the card detect #2 input signal from the card has changed.
$\overline{\text{ipi\_int\_bvd1}}$	Output	This interrupt is generated if the battery voltage detect #1 input signal from the card has changed.
$\overline{\text{ipi\_int\_bvd2}}$	Output	This interrupt is generated if the battery voltage detect #2 input signal from the card has changed.
$\overline{\text{ipi\_int\_rdy\_l}}$	Output	This interrupt is generated if RDY/$\overline{\text{IREQ}}$ pin is low.
$\overline{\text{ipi\_int\_rdy\_h}}$	Output	This interrupt is generated if RDY/$\overline{\text{IREQ}}$ pin is high.
$\overline{\text{ipi\_int\_rdy\_r}}$	Output	This interrupt is generated if a rising edge was detected on the RDY/$\overline{\text{IREQ}}$ pin.
$\overline{\text{ipi\_int\_rdy\_f}}$	Output	This interrupt is generated if a falling edge was detected on the RDY/$\overline{\text{IREQ}}$ pin.
$\overline{\text{ipi\_int\_poweron}}$	Output	This interrupt is generated if the POWERON input signal from the card has changed
$\overline{\text{ipi\_int\_sts}}$	Output	This status change interrupt is a logic AND of the following: $\overline{\text{ipi\_int\_vs1}}$, $\overline{\text{ipi\_int\_vs2}}$, $\overline{\text{ipi\_int\_wp}}$, ipi_int_cd1, ipi_int_cd2, ipi_int_bvd1, $\overline{\text{ipi\_int\_bvd2}}$, $\overline{\text{ipi\_int\_poweron}}$.
$\overline{\text{ipi\_int\_ireq}}$	Output	This interrupt line is a logic AND of the following: $\overline{\text{ipi\_int\_rdy\_l}}$, $\overline{\text{ipi\_int\_rdy\_h}}$, $\overline{\text{ipi\_int\_rdy\_r}}$, ipi_int_rdy_f.
$\overline{\text{ipi\_int\_err}}$	Output	This interrupt is generated following an error detected by the PCMCIA controller. See Section 20.5.4.1, "Error Interrupt Conditions" for detailed description of error cases.

Table 20-3 provides descriptions for the BVD1 and BVD2 signals.

Table 20-3. BVD1 and BVD2 Descriptions

BVD1	BVD2	Description
1	1	battery is in good condition
1	0	battery is in warning condition and should be replaced, although data integrity on the card is assured.
0	X	battery is in no longer serviceable and data is lost.

20.4 Memory Map and Register Definition

Table 20-4 shows the memory map of the PCMCIA controller.

Table 20-4. PCMCIA Controller Memory Map

Address	Register	Access	Reset Value	Section/Page
0xD800_4000 (PIPR)	PCMCIA input Pins Register	Read Only	0x0000_00——	20.4.1.1/20-9
0xD800_4004 (PSCR)	PCMCIA Status Changed Register	Read/Write	0x0000_0000	20.4.1.2/20-11
0xD800_4008 (PER)	PCMCIA Enable Register	Read/Write	0x0000_1018	20.4.1.3/20-12
0xD800_400C (PBR0)	PCMCIA Base Register 0	Read/Write	0x0000_0000	20.4.1.4/20-14
0xD800_4010 (PBR1)	PCMCIA Base Register 1	Read/Write	0x0000_0000	20.4.1.4/20-14
0xD800_4014 (PBR2)	PCMCIA Base Register 2	Read/Write	0x0000_0000	20.4.1.4/20-14
0xD800_4018 (PBR3)	PCMCIA Base Register 3	Read/Write	0x0000_0000	20.4.1.4/20-14
0xD800_401C (PBR4)	PCMCIA Base Register 4	Read/Write	0x0000_0000	20.4.1.4/20-14
0xD800_4028 (POR0)	PCMCIA Option Register 0	Read/Write	0x0000_0000	20.4.1.5/20-15
0xD800_402C (POR1)	PCMCIA Option Register 1	Read/Write	0x0000_0000	20.4.1.5/20-15
0xD800_4030 (POR2)	PCMCIA Option Register 2	Read/Write	0x0000_0000	20.4.1.5/20-15
0xD800_4034 (POR3)	PCMCIA Option Register 3	Read/Write	0x0000_0000	20.4.1.5/20-15
0xD800_4038 (POR4)	PCMCIA Option Register 4	Read/Write	0x0000_0000	20.4.1.5/20-15
0xD800_4044 (POFR0)	PCMCIA Offset Register 0	Read/Write	0x0000_0000	20.4.1.6/20-19

Table 20-4. PCMCIA Controller Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0xD800_4048 (POFR1)	PCMCIA Offset Register 1	Read/Write	0x0000_0000	20.4.1.6/20-19
0xD800_404C (POFR2)	PCMCIA Offset Register 2	Read/Write	0x0000_0000	20.4.1.6/20-19
0xD800_4050 (POFR3)	PCMCIA Offset Register 3	Read/Write	0x0000_0000	20.4.1.6/20-19
0xD800_4054 (POFR4)	PCMCIA Offset Register 4	Read/Write	0x0000_0000	20.4.1.6/20-19
0xD800_4060 (PGCR)	PCMCIA General Control Register	Read/Write	0x0000_0008	20.4.1.7/20-20
0xD800_4064 (PGSR)	PCMCIA General Status Register	Read/Write	0x0000_0000	20.4.1.8/20-21

20.4.1 Register Summary

Table 20-2 shows the key to the register fields, and Table 20-5 shows the register figure conventions.

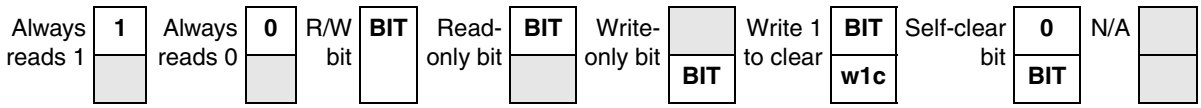


Figure 20-2. Key to Register Fields

Table 20-5. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.

Table 20-5. Register Figure Conventions (continued)

Convention	Description
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 20-6 shows the PCMCIA register summary.

Table 20-6. PCMCIA Controller Register Summary

Name		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
0xD800_4000 (PIPR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	POWERONt	RDY	BVD2	BVD1	$\overline{CD}$		WP	VS	
	W																
0xD800_4004 (PSCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	POWC	RDYR	RDYF	RDYH	RDYL	BVDC2	BVDC1	CDC2	CDC1	WPC	VSC2	VSC1
	W					w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c	w1c
0xD800_4008 (PER)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	ERRINTEN	POWERONEN	RDYRE	RDYFE	RDYHE	RDYLE	BVDE2	BVDE1	CDE2	CDE1	WPE	VSE2	VSE1
	W																
0xD800_400C (PBR0) 0xD800_4010 (PBR1) 0xD800_4014 (PBR2) 0xD800_4018 (PBR3) 0xD800_401C (PBR4)	R	0	0	0	0	0	0	PBA[25:16]									
	W																
	R	PBA[15:0]															
	W																

Table 20-6. PCMCIA Controller Register Summary (continued)

Name		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
0xD800_4028 (POR0) 0xD800_402C (POR1) 0xD800_4030 (POR2) 0xD800_4034 (POR3) 0xD800_4038 (POR4)	R	0	0	PV	WPEN	WP	PRS		PPS	PSL[6:0]							PSST[5]
	W																
	R	PSST[4:0]					PSHT[5:0]						0 BSIZE				
	W																
0xD800_4044 (POFR0) 0xD800_4048 (POFR1) 0xD800_404C (POFR2) 0xD800_4050 (POFR3) 0xD800_4054 (POFR4)	R	0	0	0	0	0	0	POFA[25:16]									
	W																
	R	POFA[15:0]															
	W																
0xD800_4060 (PGCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	LPMEN	SPKREN	POE	RESET
	W																
0xD800_4064 (PGSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	NWINE	LPE	SE	CDE	WPE
	W												w1c	w1c	w1c	w1c	w1c

20.4.1.1 PCMCIA Input Pins Register (PIPR)

This register indicates the status of inputs from the PCMCIA card to the host: battery voltage detect, card detect, ready, voltage sense, and write protect status (BVD, CD, RDY, VS, WP). PIPR is a read-only register.

The register should be clocked with a gated clock. This clock is active only when trying to access the peripheral. When accessing this register, a 2-wait state is added by the PCMCIA controller. The reset values in Figure 20-3 are the reset values of the PIPR flip-flops. The actual data read reflects the value to the PCMCIA controller. Table 20-7 provides the register's field descriptions.

0xD800_4000 (PIPR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	POWERON	RDY	BVD2	BVD1	CD		WP	VS	
W																
Reset	0	0	0	0	0	0	0	0	—	—	—	—	—	—	—	—

Figure 20-3. PCMCIA Input Pins Register (PIPR)

Table 20-7. PIPR Field Descriptions

Field	Description
31–9	Reserved.
8 POWERON	Power is on. This bit indicates the status of the power signal from the card. 0 Card indicates that it did not reach its power supply requirements. 1 Card indicates Power is on.
7 RDY	RDY/$\overline{\text{BSY}}$/$\overline{\text{IREQ}}$. When the card and its socket are in memory interface operation, this bit functions as RDY/BSY indicating that the card is busy processing a previous write command. When the card and its socket are in I/O interface operation, this bit functions as IREQ indicating that a device on the PC card requires service by host software. This interrupt could be either level or pulse and can have either high or low polarity. This data can be read in the CIS of the card itself. 0 PC card is busy processing a previous command or performing initialization. 1 PC card is ready to accept a new data-transfer command.
6 BVD2	Battery Voltage detect 2/SPKR IN. When the card and its socket are in memory interface mode, this bit reflects the BVD2 signal. For details about settings, see Table 20-3. When the card and its socket are in I/O mode, this bit is used as SPKR IN (speaker in) for a Binary Audio signal, an optional signal which is available only when the card and the socket have been configured for the I/O interface.
5 BVD1	Battery Voltage detect 1/STSCHG IN. When the card and its socket are in memory interface mode, this bit reflects the BVD1 signal. For details, see Table 20-3. When the card and its socket are in I/O mode, this bit is used as STSCHG (status change) indicator. 0 Status has not changed. 1 Status has changed. **Note:** To find out the exact signals that changed value, the Status Change register of the card itself should be read.

Table 20-7. PIPR Field Descriptions (continued)

Field	Description
4–3 $\overline{CD}$	Card Detect 1 and Card Detect 2. Card Detect 1 and Card Detect 2 bits indicate a proper detection of card insertion. When both bits are '0', the card is inserted properly. 00 Card is inserted properly. 01 Card is inserted improperly. 11 Card is inserted improperly. 11 Card is not inserted. These bits are asynchronous.
2 WP	Write Protect. This bit reflects the state of the write protect switch on the PC card. 0 Write Protect Switch is disabled. 1 Write Protect switch is enabled.
1–0 VS	Voltage Sensor. VS bits notify the host of the card's Card Information Structure (CIS) Vcc requirements. This data can be used by the host to control external voltage transceiver. For details see the PCMICA PCCARD standard.

20.4.1.2 PCMCIA Status Change Register (PSCR)

Each bit in the PSCR register is set any time a change in the signal it monitors occurs. The status is cleared using a "write 1 to clear" operation on the register. The contents of PSCR, shown in Figure 20-4, are logically ANDed with the PER register to generate a PCMCIA interrupt.

The register should be clocked with a gated clock. This clock is active only when trying to access the peripheral. When accessing this register, two wait states are added by the PCMCIA. The inputs to the module are sampled twice before being read, to avoid meta-stability. This is done in spite of the fact that interrupts are generated even when the clock is off.

The bit assignments for the PSCR register are shown in Figure 20-4. The bit field descriptions are provided in Table 20-8. Each of the bits in Table 20-8 (excluding RDYL and RDYHare) set (=1) when a change occurs in its corresponding parameter. It is zeroed on system reset. RDYL and RDYH are level sensitive and therefore, reflect the value of the RDY pin and have no reset value.

0xD800_4004 (PSCR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	POWC	RDYR	RDYF	RDYH	RDYL	BVDC 2	BVDC 1	CDC2	CDC1	WPC	VSC2	VSC1
W					w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	—	—	0	0	0	0	0	0	0

Figure 20-4. PCMCIA Status Change Register (PSCR)

Table 20-8. PSCR Field Descriptions

Field	Description
31–12	Reserved.
11 POWC	POWERON signal changed. 0 No change has occurred in the POWERON signal since system reset. 1 A change has occurred in the POWERON signal since system reset.
10 RDYR	RDY/$\overline{\text{IREQ}}$ pin rising edge detect. Device and socket positive edge interrupt. 0 No rising edge has occurred in RDY since system reset. 1 A rising edge occurred in RDY since system reset.
9 RDYF	RDY/$\overline{\text{IREQ}}$ pin falling edge detect. Device and socket negative edge interrupt. 0 No falling edge has occurred in RDY since system reset. 1 A falling edge occurred in RDY since system reset.
8 RDYH	This bit reflects value of RDY signal. RDY/$\overline{\text{IREQ}}$ pin is high indicating a device and socket high level interrupt.
7 RDYL	RDY/$\overline{\text{IREQ}}$ pin is low. Device and socket loveless interrupt. This bit is the inverted value of RDY signal
6 BVDC2	Battery Voltage 2/SPKR IN Changed. 0 No change has occurred in Battery Voltage #2 or SPKR since system reset. 1 A change has occurred in Battery Voltage #2 or SPKR since system reset.
5 BVDC1	Battery Voltage 1/STSCHG Changed. 0 No change has occurred in Battery Voltage #1 since system reset. 1 A change has occurred in Battery Voltage #1 since system reset.
4 CDC2	Card Detect 2 hanged. 0 No change has occurred in card detect #2 since system reset. 1 A change has occurred in card detect #2 since system reset.
3 CDC1	Card Detect 1 Changed. 0 No change has occurred in card detect #1 since system reset. 1 A change has occurred in card detect #1 since system reset.
2 WPC	Write Protect Changed. 0 No change has occurred in the write protect status since system reset. 1 A change has occurred in the write protect status since system reset.
1 VSC2	Voltage Sense2 Changed. 0 No change has occurred in voltage sensor #2 since system reset. 1 A change has occurred in voltage sensor #2 since system reset.
0 VSC1	Voltage Sense1 Changed. 0 No change has occurred in voltage sensor #1 since system reset. 1 A change has occurred in voltage sensor #1 since system reset.

20.4.1.3 PCMCIA Enable Register (PER)

Setting a bit in PER, shown in Figure 20-5, enables the corresponding interrupt. When accessing this register, one wait state will be added by the PCMCIA. Table 20-9 shows the register's field descriptions.

0xD800_4008 (PER) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	ERRINTEN	POWERONEN	RDYRE	RDYFE	RDYHE	RDYLE	BVDE2	BVDE1	CDE2	CDE1	WPE	VSE2	VSE1
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0

Figure 20-5. PCMCIA Enable Register (PER)

Table 20-9. PER Field Descriptions

Field	Description
31–13	Reserved.
12 ERRINTEN	Error Interrupt Enable. Setting this bit enables the interrupt as a result of an error signal. 0 Interrupt is disabled. 1 Interrupt is enabled. Note that the default is to enable the interrupt.
11 POWERONEN	Power is On Interrupt Enable. Setting this bit enables the interrupt as a result of any Power On signal change. 0 Interrupt is disabled. 1 Interrupt is enabled.
10 RDYRE	RDY/$\overline{\text{IREQ}}$ pin rising edge interrupt enable. 0 Interrupt is disabled. 1 Interrupt is enabled.
9 RDYFE	RDY/$\overline{\text{IREQ}}$ pin falling edge interrupt enable. 0 Interrupt is disabled. 1 Interrupt is enabled.
8 RDYHE	RDY/$\overline{\text{IREQ}}$ pin is high level interrupt enable. 0 Interrupt is disabled. 1 Interrupt is enabled.
7 RDYLE	RDY/$\overline{\text{IREQ}}$ pin is low level interrupt enable. 0 Interrupt is disabled. 1 Interrupt is enabled.
6 BVDE2	Battery Voltage 2/SPKR IN interrupt enable. Setting this bit enables the interrupt as a result of any signal change from battery voltage sensor #2 OR from the SPKR IN signal. 0 Interrupt is disabled. 1 Interrupt is enabled.
5 BVDE1	Battery Voltage 1/STSCHG interrupt enable. Setting this bit enables the interrupt as a result of any signal change from the battery voltage sensor #1. 0 Interrupt is disabled. 1 Interrupt is enabled.

Table 20-9. PER Field Descriptions (continued)

Field	Description
4 CDE2	Card Detect 2 interrupt enable. Setting this bit enables the interrupt as a result of any signal change from card detect #2. **Note:** The default setting enables the interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
3 CDE1	Card Detect 1 interrupt enable. Setting this bit enables the interrupt as a result of any signal change from card detect #1 **Note:** The default setting enables the interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
2 WPE	Write Protect interrupt enable. Setting this bit enables the interrupt as a result of any signal change in the write protect status. 0 Interrupt is disabled. 1 Interrupt is enabled.
1 VSE2	Voltage sense2 interrupt enable. Setting this bit enables the interrupt as a result of any signal change from voltage sensor #2. 0 Interrupt is disabled. 1 Interrupt is enabled.
0 VSE1	Voltage sense1 interrupt enable. Setting this bit enables the interrupt as a result of any signal change from voltage sensor #1. 0 Interrupt is disabled. 1 Interrupt is enabled.

20.4.1.4 PCMCIA Base Registers 0–4 (PBR0–PBR4)

This is compared to the address bus to determine if a PCMCIA window is being accessed by an internal bus master. PBA is used in conjunction with POR[BSIZE]. When accessing this register, 1-wait state will be added by the PCMCIA. The field assignments for this register are shown in Figure 20-6. Table 20-10 shows the register's field descriptions.

0xD800_400C (PBR0)
0xD800_4010 (PBR1)
0xD800_4014 (PBR2)
0xD800_4018 (PBR3)
0xD800_401C (PBR4)

Access: User Read/Write

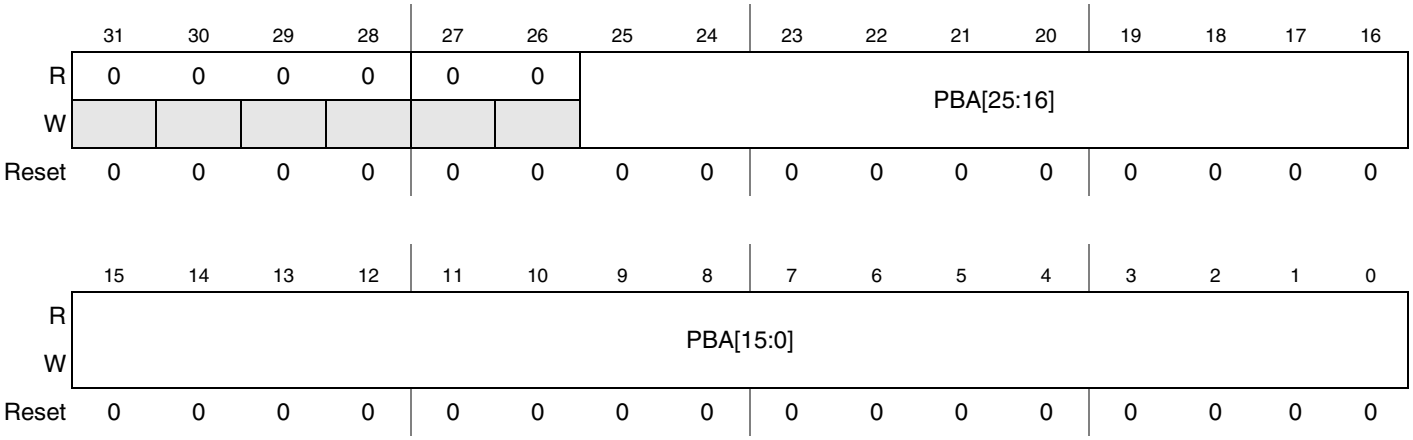


Figure 20-6. PCMCIA Base Registers 0–4 (PBR0–PBR4)

Table 20-10. PBR0–PBR4 Field Descriptions

Field	Description
31–26	Reserved.
25–0 PBA	PCMCIA Base Address.

20.4.1.5 PCMCIA Option Registers 0–4 (POR0–POR4)

The POR registers shown in Figure 20-7 handle time manipulation, provide the address mask for the bank size, and define the region, write protection, and validation. When accessing this register 1-wait state will be added by the PCMCIA. Table 20-11 shows the register's field descriptions.

0xD800_4028 (POR0)
0xD800_402C (POR1)
0xD800_4030 (POR2)
0xD800_4034 (POR3)
0xD800_4038 (POR4)

Access: User Read/Write

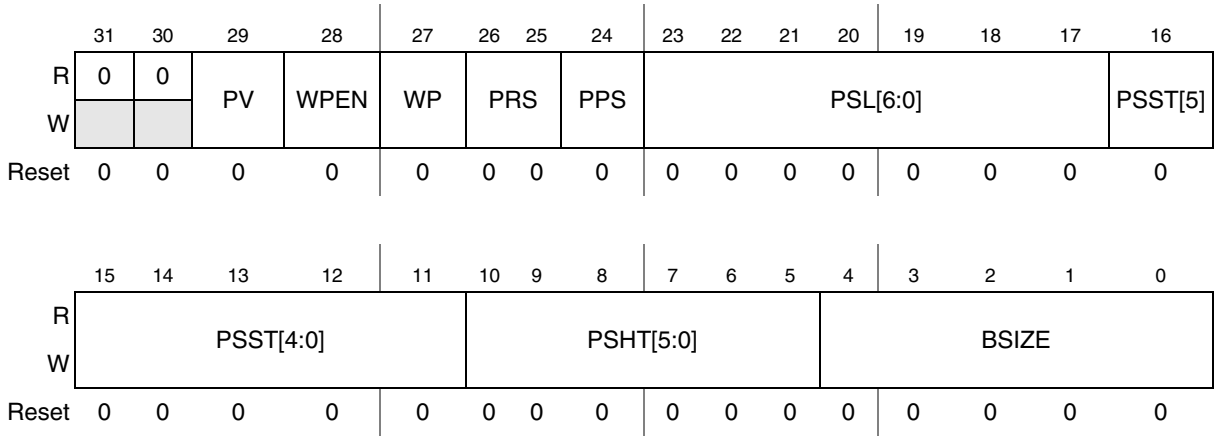


Figure 20-7. PCMCIA Option Registers 0–4 (POR0–POR4)

Table 20-11. POR0–POR4 Field Descriptions

Field	Description
31–30	Reserved.
29 PV	PCMCIA Valid. Defines whether the contents of the PBR and POR pair are valid (window enable). See also Table 20-13. 0 This bank is invalid. 1 This bank is valid.
28 WPEN	PCMCIA Write Protect Input Enable. This bit is the write protect input signal enable bit, controlled by software. When this bit is cleared, the WP input to the PCMCIA module is ignored. To see the relationship between this bit, the WP bit, and the WP signal coming from the card refer to Section 20.5.7, “Write Protect.” 0 Write Protect input (WP) signal is ignored. 1 Write Protect (WP) input signal is enabled.
27 WP	PCMCIA Write Protect Enable This bit is the write protect enable bit controlled by software. To see the relationship between this bit, the WPEN bit, and the WP signal coming from the card refer to Section 20.5.7, “Write Protect.” 0 Not write protected. 1 Write Protected. Attempting to write to this window causes an interrupt.
26–25 PRS	PCMCIA Region Select. 00 Common memory space. 01 TrueIDE mode. 10 Attribute memory space. 11 I/O space.
24 PPS	PCMCIA Port Size. Specifies the port size of the PCMCIA window. Refer to Section 20.5.8, “16-Bit/8-Bit Support” for more details. 0 16-bit port size 1 8-bit port size

Table 20-11. POR0–POR4 Field Descriptions (continued)

Field	Description
23–17 PSL	PCMCIA Strobe Length. Determines the number of cycles the strobe is asserted during a PCMCIA access for this window and, thus, it is the main parameter for determining cycle length. The cycle may be lengthened by asserting $\overline{\text{WAIT}}$. **Note:** To sample the $\overline{\text{WAIT}}$ signal it must be synchronized by two FF. This means that if the system must rely on the $\overline{\text{WAIT}}$ signal, the PSL should be calculated as the maximum valid time on WAIT plus two. For example, suppose we have a 100 MHz clock (one period is therefore 10 ns) and the card specification says that the time from WE/OE low to $\overline{\text{WAIT}}$ valid is 70 ns. The PSL value should be at least: (70 ns/10 ns)+2=9. 0000000 Strobe asserted 128 clocks cycles. 0000001 Strobe asserted 1 clocks cycles. 0000010 Strobe asserted 2 clocks cycles. ... 1111111 Strobe asserted 127 clocks cycles.
16–11 PSST	PCMCIA Strobe Set Up Time (address to strobe assertion). Specifies when $\overline{\text{IOWR}}$ or $\overline{\text{WE}}$ is asserted during a PCMCIA write access or when $\overline{\text{IORD}}$ or $\overline{\text{OE}}$ are asserted during a PCMCIA read access handled by the PCMCIA controller. This helps meet address/setup time requirements for slow memories and peripherals. 000000 Reserved. 000001 Address to strobe assertion 1 clock. 000010 Address to strobe assertion 2 clock. ... 111111 Address to strobe assertion 63 clock. **Note:** Using PSST=000001 is not allowed when WPEN bit is set since the synchronization of WP signal takes 2 clocks.
10–5 PSHT	PCMCIA Strobe Hold Time (strobe negation to address negation).Specifies when $\overline{\text{IOWR}}$ or $\overline{\text{WE}}$ are negated during a PCMCIA write or when $\overline{\text{IORD}}$ or $\overline{\text{OE}}$ are negated during a PCMCIA read. Used to meet address/data hold time requirements for slow memories and peripherals. 000000 Strobe negation to address change 0 clock. 000001 Strobe negation to address change 1 clock. ... 111111 Strobe negation to address change 63 clock.
4–0 BSIZE	PCMCIA Bank Size. Determines the address mask field of each POR and provides masking for any of the corresponding bits in the associated PBR. The bank size corresponds to values of this bit field as indicated in Table 20-12. BSIZE determines not only the bank size, but also how the address is compared with PBR[PBA]. If BSIZE is a virtual field, the MASK is defined as shown in Table 20-13. Addr, AND MASK PBA, AND MASK for a valid PCMCIA access; otherwise, it is not a valid PCMCIA access.

Table 20-12. BSIZE Values

Value	Meaning	Value	Meaning	Value	Meaning
00000	1 byte	01111	1 Kbyte	11110	1 Mbyte
00001	2 byte	01110	2 Kbyte	11111	2 Mbyte
00011	4 byte	01010	4Kbyte	11101	4 Mbyte
00010	8 byte	01011	8 Kbyte	11100	8 Mbyte
00110	16 byte	01001	16 Kbyte	10100	16 Mbyte
00111	32 byte	01000	32 Kbyte	10101	32 Mbyte

Table 20-12. BSIZE Values (continued)

Value	Meaning	Value	Meaning	Value	Meaning
00101	64 byte	11000	64 Kbyte	10111	64 Mbyte
00100	128 byte	11001	128 Kbyte		
01100	256 byte	11011	256 Kbyte		
01101	512 byte	11010	512 Kbyte		

NOTE

BSIZE determines not only the bank size, but also how the address is compared with PBR[PBA]. According to the virtual field, MASK as defined as shown on Table 20-13.

Table 20-13. BSIZE Mask

BSIZE	MASK																													
00000	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
00001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
00011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
00010	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
00110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
00111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
00101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
00100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
01100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
01101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
01111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
01110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
01010	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
01011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
01001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
01000	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
11000	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
11001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
11011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
11010	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
11110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
11111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-13. BSIZE Mask (continued)

BSIZE	MASK
11101	1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11100	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10100	1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10101	1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10111	1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

20.4.1.6 PCMCIA Offset Registers 0–4 (POFR0–POFR4)

The offset address of the window. PBA is used in conjunction with POR[BSIZE]. The external address is ext_addr= POFA + haddr and MASK. When accessing this register 1-wait state is added by the PCMCIA/CF controller. The field definition of the POFRx registers is shown in Figure 20-8. Table 20-14 shows the field descriptions.

0xD800_4044 (POFR0)
0xD800_4048 (POFR1)
0xD800_404C (POFR2)
0xD800_4050 (POFR3)
0xD800_4054 (POFR4)

Access: User Read/Write

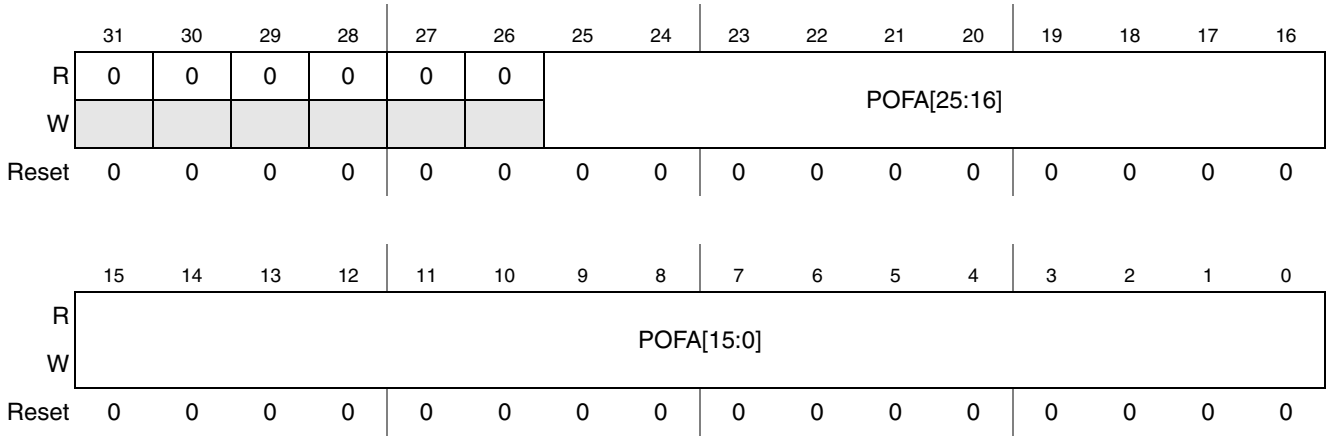


Figure 20-8. PCMCIA Offset Registers 0–4 (POFR0–POFR4)

Table 20-14. POFR0–POFR4 Field Descriptions

Field	Description
31–26	Reserved.
25–0 POFA	PCMCIA Offset Address. The offset address of the window. POFA is used in conjunction with POR[BSIZE]. The external address is ext_addr= POFA + haddr and $\overline{\text{MASK}}$.

Example 20-1. Calculating Offset Address

```
If:
haddr[25:0] = 0x0000263, MASK = 0x3FFFFC0 POFA = 0x0000161
then:
 ext_addr = 0x0000161 + 0x0000263 & (0x3FFFFC0) = 0x0000161 + 0x0000023 = 0x0000184
```

20.4.1.7 PCMCIA General Control Register (PGCR)

This is the general control register for the PCMCIA controller. When accessing this register, 1-wait state is added by the PCMCIA controller. Field definitions of the POFRx registers are shown in Figure 20-9. Table 20-15 shows the field descriptions.

0xD800_4060 (PGCR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	LPMEN	SPKR EN	POE	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 20-9. PCMCIA General Control Register (PGCR)

Table 20-15. PGCR Field Descriptions

Field	Description
31–4	Reserved.
3 LPMEN	Low Power Mode Enable. This bit puts the module into low power mode. In this case external memory accesses are disabled. The reset value is “1” (Low power mode). 0 Normal power mode. 1 Low power mode.
2 SPKREN	SPKROUT Routing Enable. This bit enables the routing of SPKRIN to SPKROUT. 0 Routing disabled. 1 Routing enabled.
1 POE	Card Output Enable. This bit enables the POE signal, used to three-state the external buffers. The POE signal will toggle as follows: POE (signal) PGCR[POE]&PCMCIA_ACCESS (signal). 0 POE signal is disabled. 1 POE signal is enabled.
0 RESET	Card Reset. This bit provides a software reset to the card. This bit is not self clearing, software must modify this bit to take the card out of reset. 0 Card is not in reset. 1 Card is in reset.

20.4.1.8 PCMCIA General Status Register (PGSR)

This is a general status register. If an error interrupt was generated to the host, the host can access this register to find out what caused the error interrupt. All the bits in this register are cleared by writing ‘1’ to the appropriate bit. When accessing this register, 1-wait state is added by the PCMCIA controller. Field definition of the POFRx registers are shown in Figure 20-10. Table 20-16 shows the field descriptions.

0xD800_4064 (PGSR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	NWINE	LPE	SE	CDE	WPE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 20-10. PCMCIA General Status Register (PGSR)

Table 20-16. PGSR Field Descriptions

Field	Description
31–5	Reserved.
4 NWINE	No Window Error. Attempt to access a card address to an address that is not mapped by any window. 0 No attempt to access a card address to an address that is not mapped by any window since the last system reset was made. 1 An attempt to access a card address to an address that is not mapped by any window since the last system reset was made.
3 LPE	Low Power Error. Attempt to access a card when in low power mode. 0 No attempt to access a card when in low power mode was made since the last system reset 1 An attempt to access a card when in low power mode was made since the last system reset
2 SE	Size Error. A 16-bit access to an 8-bit card was made. 0 No 16-bit access to an 8-bit card was made since the last system reset. 1 A 16-bit access to an 8-bit card was made since the last system reset
1 CDE	Card Detect Error. Attempt to access a card when the card is not inserted. 0 No attempt to access a card when the card was not inserted has been made since last system reset. 1 An attempt to access a card when the card was not inserted has been made since last system reset. In addition, once set no accesses are enabled until it is cleared (even if a card was inserted).
0 WPE	Write Protect Error. Attempt to write to a write protected address. 0 No attempt was made to write to a protected address since the last system reset. 1 An attempt was made to write to a protected address since the last system reset.

20.5 Functional Description

This section describes the operation of memory and I/O cards, interrupt detection and handling, power control, and reset.

20.5.1 Modes of Operation

The following are the PCMCIA modes of operation:

- Memory-only card mode
- I/O card mode
- TrueIDE mode
- Low power mode

20.5.2 Windowing Capabilities

The PCMCIA I/F provides five memory windows. The user can define each memory window as a common memory space, I/O space or attribute memory space. This is done by programming the region select bits (PRS) bits in each POR register for each window.

Configuring a window is done by programming the window's base address (PBA bits in the corresponding PBR register), and by programming the bank size (BSIZE bits in the corresponding POR register).

20.5.2.1 Window Overlapping

Window overlapping is not allowed. The PCMCIA I/F does not indicate to the CPU about window overlapping, software responsible for this. Having overlapping windows will cause unexpected results.

20.5.3 $\overline{\text{WAIT}}$ Signal

The $\overline{\text{WAIT}}$ signal is asserted by the PC card to delay completion of the pending cycle. The access must terminate before the bus time out monitor generates a bus time error.

20.5.4 Interrupts

There are 14 interrupt sources in the PCMCIA controller. The PCMCIA controller generates an interrupt signal for each interrupt source. In addition, the PCMCIA generates a signal which is a locator of all the possible interrupts. It is up to the system's integrator to decide which signal(s) to connect to the system's interrupt controller module. The PCMCIA's interrupt sources are described in Table 20-17.

Table 20-17. PCMCIA I/F Interrupt Sources

Interrupt	Enabled With	Comments
VS1	PER.VSE1	See ipi_int_vs1 on page 20-5.
VS2	PER.VSE2	
WP	PER.WPE	See ipi_int_wp on page 20-5.

Table 20-17. PCMCIA I/F Interrupt Sources (continued)

Interrupt	Enabled With	Comments
CD1	PER.CDE1	See ipi_int_cd1 on page 20-5.
CD2	PER.CDE2	See ipi_int_cd2 on page 20-5.
BVD1	PER.BVDE1	See ipi_int_bvd1 on page 20-5.
BVD2	PER.BVDE2	See ipi_int_bvd2 on page 20-5.
POWERON	PGCR.POWERONEN	This signal is not part of the PCMCIA standard. See ipi_int_poweron on page 20-5.
STATUS CHANGE		"OR" of all the above interrupts. See ipi_int_sts on page 20-5.
RDY_L	PER.RDYLE	
RDY_H	PER.RDYHE	
RDY_R	PER.RDRE	
RDY_F	PER.RDYFE	
IREQ		This interrupt is an "OR" of RDY_L, RDY_H, RDY_R, and RDY_F.
ERR	PER.ERRINTEN	

The PCMCIA input pins register (PIPR) reports any change of inputs from the PCMCIA card to the host (BVD,CD,RDY,VS). The content of the PCMCIA controller status changed register (PSCR) are logically ANDed with the PCMCIA controller enable register (PER) to generate a PCMCIA controller interrupt. The interrupt level is user programmable and the PCMCIA controller can generate an additional interrupt for RDY/IREQ that can trigger upon a level (low or high) change or edge (fall or rise) of the input signal.

20.5.4.1 Error Interrupt Conditions

Any of the following error conditions can cause an error interrupt:

- Attempt to access a card when the card is in low power mode (LPM).
- Attempt to write to a write protected area, see Section 20.5.7, "Write Protect."
- Attempt to access a card when the card is not inserted.
- Attempt to do a 16-bit access to an 8-bit card violating the PPS settings or 32-bit access. See Section 20.5.8, "16-Bit/8-Bit Support."
- Attempt to access card with an address which does not match any window.

An access to the card which yields an error will take 0-3 wait states to complete according to the following conditions:

- Card detect error when PGSR[CDE] is cleared and write protect error which results from the WP signal from the card will take 2 wait states. In case these signals (WP CD) change during access, the access will end two cycles after.
- Size error, no window error low power mode and write protect error which comes from POR[WP] will take one wait state.

Due to the synchronization mechanism on CD signals and WP signal, a change in these signals during access less then two cycles before it finishes would not yield an error. That should not be a problem since these signals typically do not change too much.

20.5.5 Power Control

When LPMEN is set in the PGCR, the PCMCIA I/F internal clocks should be gated off. The module is in "listening mode." It waits for an indication that a card has been inserted. All the static signals are synchronous in both cases and status change interrupts are generated as necessary (to wake up the core from stop on card detect, for example). In the first case (LPMEN) read from PIPR and read/write from/to PSCR should take 2 wait states to complete because of the synchronizations of the static signals to the core's clock.

20.5.6 Reset and Three-Score Control

The card can be reset by software. This is done by writing to the RESET bit in the PGCR register. Output of external latches can be disabled by writing to the POE bit in PGCR register.

20.5.7 Write Protect

Write protect is handled in two ways:

- Card's write protect—the WP signal comes from the card.
- Window's write protect—the WP bit in the POR register.

When the WPEN is cleared and the card is in memory interface mode, the WP signal coming from the card is ignored. This way it is possible to enable write protection on selected memory regions, even if the card's WP pin is asserted. The settings for the Write Protect bits are shown in Table 20-18. When the card is in IO mode (POR.PRS = 11),WPEN bit is ignored.

Table 20-18. Write Protect

POR.PRS	POR.WP bit	WP Signal	POR.WPEN	Protect Mode
X0 (memory I/F)	X	0	1	Write enabled
		1	1	Write protected
	0	X	0	Write enabled
	1	X	0	Write protected
11(I/O mode)	0	X	X	Write enabled
	1			Write protected

An interrupt is generated by the PCMCIA controller at any attempt to access a write protected area.

20.5.8 16-Bit/8-Bit Support

The PCMCIA controller supports 16-bit/8-bit accesses. The access size is defined by the $\overline{\text{IOIS16}}$ signal and the PPS bit in the corresponding POR register. The settings for the $\overline{\text{IOIS16}}$ and PPS bits are shown in Table 20-19.

Table 20-19. $\overline{\text{IOIS16}}$ and PPS Bit Relations

$\overline{\text{IOIS16}}$	PPS	Access Size
0	0	16-bit access to a 16-bit card, the host can generate 8-bit accesses and 16-bit access.
0	1	8-bit access although the card is 16-bit. The host should generate 8-bit accesses only. If the host tries to do a 16-bit access, an interrupt is generated
1	0	8-bit access although 16-bit access is selected by PPS. If the host attempts to do a 16-bit access, the PCMCIA I/F writes the lower part of the data to the card.
1	1	8-bit access to 8-bit card. The host should generate 8-bit accesses only. If the host tries to do a 16-bit access, an interrupt is generated.

20.5.9 Data and Control Signals Relations

Table 20-20 describes data and control signal relations in different access modes. Data bus (D) is the data bus of the PC-card.

Table 20-20. Data and Control Signal Relations

Function Mode	REG	CE2	CE1	A0	OE	WE	IORD	IOWR	D[15:8]	D[7:0]
Standby mode	x	1	1	x	x	x	x	x	High-z	High-z
8-bit read from common memory	1 1 1	1 1 0	0 0 1	0 1 x	0 0 0	1 1 1	1 1 1	1 1 1	High-z High-z Odd-Byte[1]	Even-Byte Odd-Byte High-z[1]
16-bit read from common memory	1	0	0	x	0	1	1	1	Odd-Byte	Even-Byte
8-bit write to common memory	1 1 1	1 1 0	0 0 1	0 1 x	1 1 1	0 0 0	1 1 1	1 1 1	xxx xxx Odd-Byte[1]	Even-Byte Odd-Byte xxx[1]
16-bit write to common memory	1	0	0	x	1	0	1	1	Odd-Byte	Even-Byte
8-bit read from attribute memory	0 0 0	1 1 0	0 0 0	0 1 x	0 0 0	1 1 1	1 1 1	1 1 1	High-z High-z Not-valid	Even-Byte Not-valid Even-Byte
16-bit read from attribute memory	0	0	0	x	0	1	1	1	Not-valid	Even-Byte
8-bit write to attribute memory	0 0	1 0	0 0	0 x	1 1	0 0	1 1	1 1	xxx xxx	Even-Byte Even-Byte

Table 20-20. Data and Control Signal Relations (continued)

Function Mode	REG	CE2	CE1	A0	OE	WE	IORD	IOWR	D[15:8]	D[7:0]
8-bit read from I/O	0 0 0	1 1 0	0 0 1	0 1 x	1 1 1	1 1 1	0 0 0	1 1 1	High-z High-z Odd-Byte[1]	Even-Byte Odd-Byte High-z[1]
16-bit read from I/O	0	0	0	x	1	1	0	1	Odd-Byte	Even-Byte
8-bit write to I/O	0 0 0	1 1 0	0 0 1	0 1 x	1 1 1	1 1 1	1 1 1	0 0 0	xxx xxx Odd-Byte	Even-Byte Odd-Byte xxx
16-bit write to I/O	0	0	0	x	1	1	1	0	Odd-Byte	Even-Byte
I/O inhibit	1	x	x	x	x	x	0	1	High-z	High-z

[1] Note these are all the access modes which are supported by the standard. In the PCMCIA controller, the 8-bit access to odd byte is done by CE1 and A0 only, that is, the data will be always driven on D[7:0].

20.5.10 True IDE Mode Access

In True IDE mode windows, the selection of either task file or alt reg is made by haddr[3]

Haddr[3] = 0—will yield an access to task file or data register

Haddr[3] = 1—will yield a write to control register or read of Alt. status register

Table 20-21 shows data, control and address relations in TrueIDE mode.

Table 20-21. Data, Control and Address Relations in TrueIDE Mode

Function mode	CE2	CE1	A0-3	IORD	IOWR	D[15:8]	D[7:0]
Standby mode	1	1	xx	x	x	High-z	High-z
Task file Write	1	0	1-7h	1	0	xxx	Data In
Task file Read	1	0	1-7h	0	1	High-z	Data Out
Data Register Write	1	0	0	1	0	Odd-Byte	Even-Byte
Data Register Read	1	0	0	0	1	Odd-Byte	Even-Byte
Control Register Write	0	1	6h	1	0	xxx	Data In
Alt Status Read	0	1	6h	0	1	High-z	Data out
Invalid Mode	0	0	x	x	x	High-z	High-z

20.5.11 Card Extraction

When the card is extracted the PCMCIA controller's registers are not reset. The registers settings remain the same as before the card's extraction. This allows the host software to quickly activate the card once the CIS indicates that it is the same card.

20.5.12 TrueIDE Support

The ATA standard specifies the AT attachment interface between host systems and storage devices. The PCMCIA controller can be dynamically configured to support a PCMCIA-compatible ATA disk interface (commonly known as IDE) instead of the standard PCMCIA card interface. Using the TrueIDE interface on the PCMCIA controller changes the function of some card socket signals to support the needs of the ATA disk interface. The TrueIDE signals assignment on the PCMCIA connector is described in Table 20-22.

Table 20-22. PCMCIA Card TrueIDE Signal Names and Assignments

PC Card Signal	TrueIDE	Comment
D[15:0]	D[15:0]	
$\overline{CE1}$	$\overline{CS0}$[1]	Task file register select in TrueIDE mode.
$\overline{CE2}$	$\overline{CS1}$[1]	Alternate status register select in TrueIDE mode.
$\overline{OE}$	ATASEL[2]	The CF card samples this bit on power-on sequence. If low, the card will enter TrueIDE mode, else it will enter PC CARD mode.
A[2:0]	A[2:0]	Address
A[10:3]	not used	These bits should be connected to ‘0’ on TrueIDE
$\overline{WE}$	$\overline{WE}$	
READY/$\overline{IREQ}$	INTRQ	Interrupt request—INTRQ is the ATA notation and is asserted HIGH.
WP/$\overline{IOIS16}$	$\overline{IOCS16}$	
$\overline{CD1}$	$\overline{CD1}$	
$\overline{CD2}$	$\overline{CD2}$	
$\overline{VS1}$	$\overline{VS1}$	
$\overline{VS2}$	$\overline{VS2}$	
RESET	$\overline{RESET}$	This signal is asserted LOW in ATA mode, HIGH in other modes.
$\overline{WAIT}$	IOCHRDY	IO Channel Ready—asserted HIGH—polarity inversion of $\overline{WAIT}$.
$\overline{REG}$	not used	this bit should be connected to ‘1’ on TrueIDE
BVD1/$\overline{STSCHG}$	$\overline{PDIAG}$	Diagnostics complete signal.
BVD2/$\overline{SPKR}$	$\overline{DASP}$	Disk Active
$\overline{IORD}$	$\overline{IORD}$	
$\overline{IOWR}$	$\overline{IOWR}$	

1 In TrueIDE mode, #CS0 and #CS1 (task file chip select) behave differently from #CE1 and #CE2 (byte lane chip selects)

2 Dynamic change of ATASEL is not supported since it requires power up of the card. Therefore the ATASEL pin of the card will grounded in the socket.

20.5.13 Endianness Support

The PCMCIA controller supports Big and Little Endian. The Endianness is defined according to the Endianness input pin to the module. This input should be tied high or low by the chip integrator. Connecting the input to “1” means Big Endian.Connecting the input to “0” means Little Endian. Dynamic Endianness is not supported.

20.6 Timing Diagrams

Figure 20-11 and Figure 20-12 show PCMCIA typical accesses.

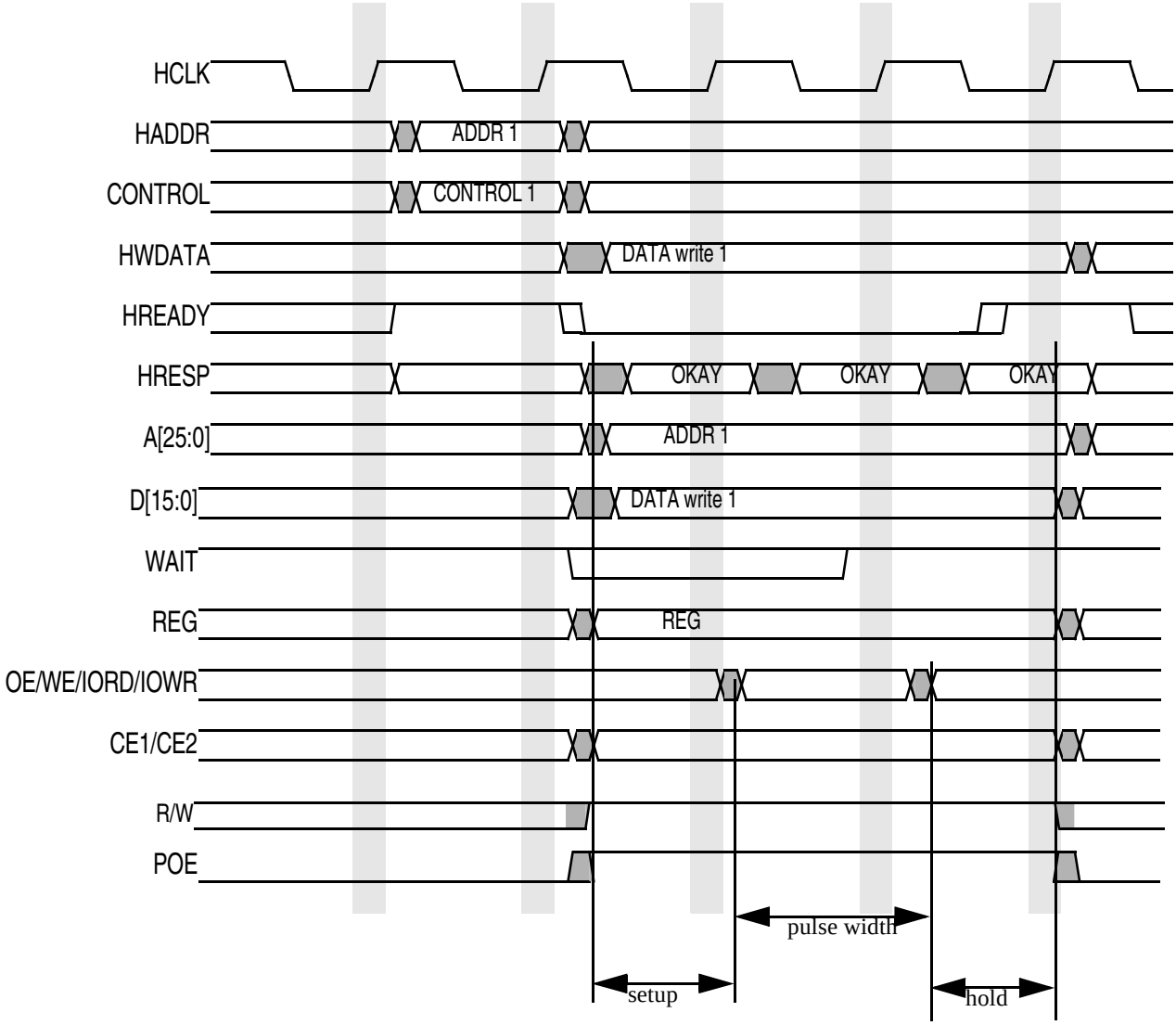


Figure 20-11. Write Accesses PSHT=1, PSST =1

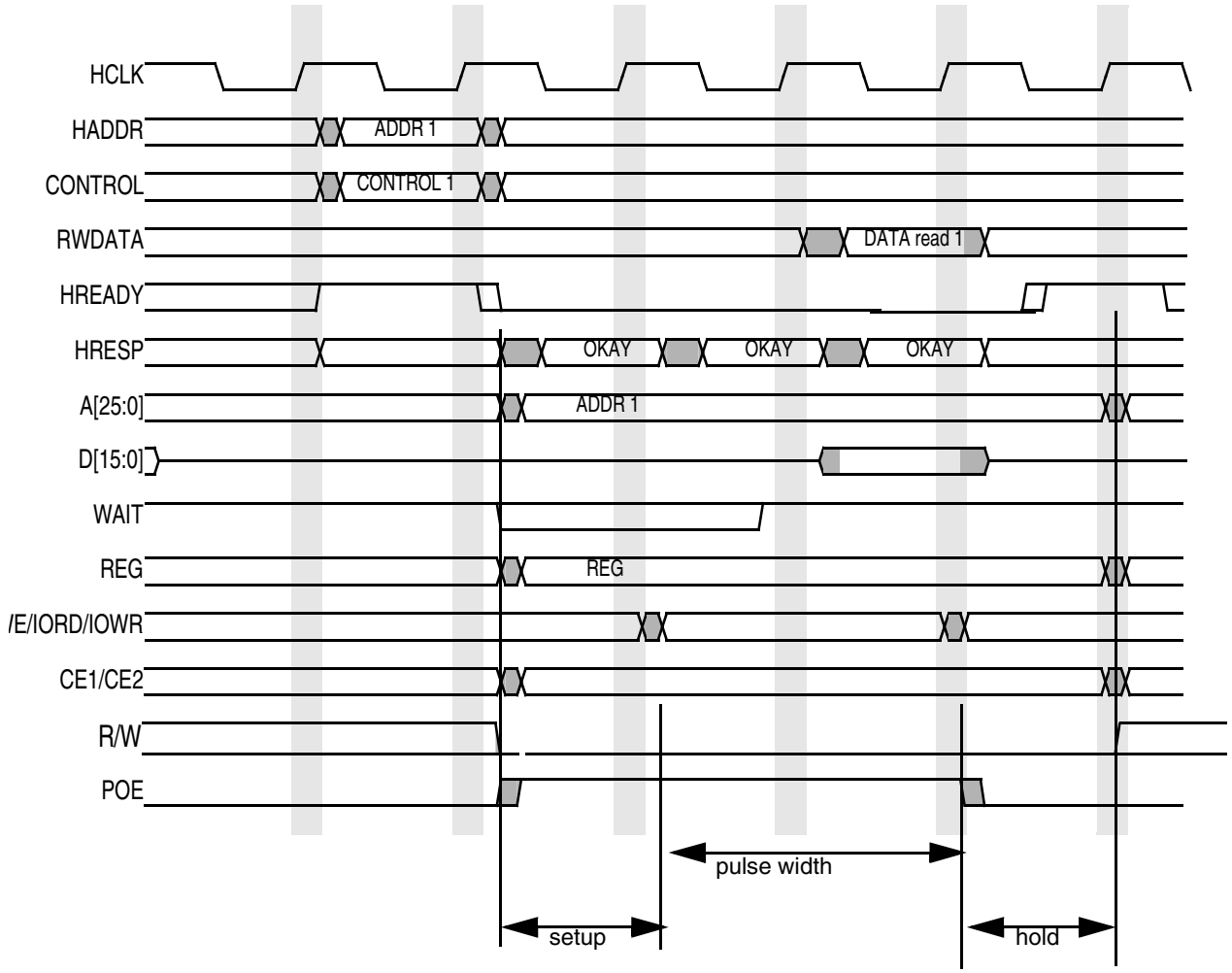


Figure 20-12. Read Cycle PSHT=1, PSST =1

Book II, Part 4: Connectivity Peripherals

Introduction

The i.MX27 processor contains the following modules that provide communication with a variety of different peripheral using several different interfaces:

1-Wire Module

The 1-Wire module provides bidirectional communication between the ARM9 core and the Add-Only-Memory EPROM (DS2502). The 1-kilobit EPROM is used to hold battery information and communicate with the ARM9 Platform using the IP interface. The ARM9 (through the 1-Wire interface) acts as the bus master and the DS2502 device is the slave. The 1-Wire peripheral does not trigger interrupts; hence, it is necessary for the ARM9 to poll of the 1-Wire to manage the module. The 1-Wire uses an external pin (to connect to the DS2502). Timing requirements are met in hardware with the help of a 1-MHz clock. The clock divider generates a 1-MHz clock that is used as time reference by the state machine. Timing requirements are crucial for proper operation, and the 1-Wire state machine and the internal clock provide the necessary signal.

Advanced Technology Attachment (ATA)

The Advanced Technology Attachment (ATA) module provides an AT attachment host interface for the i.MX27. Its main use is to provide an interface with IDE hard disc drives and ATAPI optical disc drives. It interfaces with the ATA device using industry standard ATA signals. The ATA interface is compliant to the ATA-6 standard, and supports following ATA standard protocols:

- PIO mode 0, 1, 2, 3, and 4

- Multiword DMA mode 0, 1, and 2
- Ultra DMA modes 0, 1, 2, 3, and 4 with a bus clock of 50 MHz or higher
- Ultra DMA modes 5 with bus clock of 80 MHz or higher

The ATA interface has 2 buses connected to it. The CPU bus provides communication with the ARM9 host processor and the DMA bus provides communication between the ATA module and the host DMA unit. All internal ATA registers are visible from both buses, allowing Smart Direct Memory Access (SDMA) access to program the interface.

There are basically 2 protocols that can be active at the same time on the ATA bus. The first and simplest protocol (PIO mode access) can be started at any time by either the ARM9 or the host SDMA to the ATA bus. The PIO mode is a slow protocol, mainly intended to be used to program an ATA disc drive, but also possible to use to transfer data to/from the disc drive.

The second protocol is the DMA mode access. DMA mode is started by the ATA interface after receiving a DMA request from the drive, and only if the ATA interface has been programmed to accept the DMA request. In DMA mode, either multiword DMA or ultra DMA protocol is used on the ATA bus. All transfers between FIFO and host IP or DMA IP bus are zero wait states transfer, so high speed transfer between FIFO and DMA/host bus is possible.

Configurable Serial Peripheral Interface (CSPI)

There are three identical CSPI modules in the i.MX27 IC. Each CSPI is equipped with data FIFOs, and is a master/slave configurable serial peripheral interface module, capable of interfacing to both SPI master and slave devices. The CSPI Ready ($\overline{\text{SPI\_RDY}}$) and Chip Select ($\overline{\text{SS}}$) control signals enable fast data communication with fewer software interrupts.

The CSPI is used for fast data communication with fewer software interrupts. It includes the following features:

- Full-duplex synchronous serial interface
- Master/Slave configurable
- Four chip selects to support multiple peripherals
- Transfer continuation function allows unlimited length data transfers
- 32-bit wide by 8-entry FIFO for both transmit and receive data
- Polarity and phase of the Chip Select ($\overline{\text{SS}}$) and SPI Clock (SCLK) are configurable
- DMA support

Inter-Integrated Circuit Bus (I^2C)

The Inter-Integrated Circuit Bus (I^2C) module provides a serial interface for controlling the Sensor Interface and other external devices. Data rates of up to 100 kbps are supported.

The I^2C module provides functionality of a standard I^2C slave and master. The I^2C module is designed to be compatible with the standard Phillips I^2C bus protocol. The I^2C is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. This bus is suitable for applications requiring occasional communications over a short distance

between many devices. The flexible I^2C allows additional devices to be connected to the bus for expansion and system development.

Keypad Port (KPP)

The Keypad Port (KPP) is designed to interface with keypad matrix with 2-contact or 3-point contact keys. The KPP is designed to simplify the software task of scanning a keypad matrix. With appropriate software support, the KPP is capable of detecting, debouncing, and decoding one or multiple keys pressed simultaneously in the keypad. The KPP supports up to 8 x 8 external key pad matrices. Its port pins can be used as general purpose I/O. Using an open drain design, the KPP includes glitch suppression circuit design, multiple keys, long key, and standby key detection.

Secure Digital Host Controller (SDHC)

The Security Digital Host Controller (SDHC) integrates both MultiMediaCard (MMC) support along with Secure Digital (SD) memory and I/O functions, including SD memory and I/O combo card.

The Multi Media Card (MMC), is a universal low cost data storage and communication media that is designed to cover a wide area of applications as, among others, electronic toys, organizers, PDAs, and smart phones.

The Secure Digital Card (SD), is an evolution of MMC technology, with two additional pins in the form factor. It is specifically designed to meet the security, capacity, performance, and environment requirement inherent in newly emerging audio and video consumer electronic devices.

Universal Asynchronous Receiver Transmitter (UART)

The Universal Asynchronous Receiver Transmitter (UART) provides serial communication capability with external devices through an RS-232 cable or through use of external circuitry that converts infrared signals to electrical signals (for reception), or transforms electrical signals to signals that drive an infrared LED (for transmission) to provide low speed IrDA compatibility.

The i.MX27 contains six UART modules. Each UART module is capable of standard RS-232 non-return-to-zero (NRZ) encoding format and IrDA-compatible infrared modes.

The UART transmits and receives characters containing either 7 or 8 bits (program selectable). To transmit, data is written from the IP data bus (SkyBlue line interface) to a 32-byte transmitter FIFO (TxFIFO). This data is passed to the shift register and shifted serially out on the transmitter pin (TXD). To receive, data is received serially from the receiver pin (RXD) and stored in a 32-halfwords-deep receiver FIFO (RxFIFO). The received data is retrieved from the RxFIFO on the IP data bus. The RxFIFO and TxFIFO generate maskable interrupts as well as DMA Requests when the data level in each of the FIFO reaches a programmed threshold level.

Universal Serial Bus, On-The-Go (USBOTG), High-Speed

The i.MX27 uses a Universal Serial Bus, On-The-Go (USBOTG) module that provides all of the functionality required to support three independent USB ports, compatible with the USB 2.0 specification.

In addition to the normal USB functionality, the module also provides support for direct connections to on-board USB peripherals and supports multiple interface types for serial transceivers. The USB module provides high performance USB On-The-Go (OTG) functionality, compliant with the USB 2.0 specification, the OTG supplement, and the ULPI 1.0 Low Pin Count specification.

Fast Ethernet Controller (FEC)

The Ethernet Media Access Controller (MAC) is designed to support both 10 and 100 Mbps Ethernet/IEEE 802.3 networks. An external transceiver interface and transceiver function are required to complete the interface to the media. The FEC supports three different standard MAC-PHY (physical) interfaces for connection to an external Ethernet transceiver. The FEC supports the 10/100 Mbps MII and the 10 Mbps-only 7-wire interface, using a subset of the MII pins.

Chapter 21
1-Wire Interface (1-Wire)

The 1-Wire® module is a peripheral device that communicates with the ARM926EJ-S Core via the IP interface and provides a communication line to a 1 Kbit Add-Only Memory (DS2502). The DS2502 is a 1 kbit 1-Wire EPROM. The 1-Wire interface is able to send or receive one bit at a time to the DS2502. The required protocol for accessing the DS2502 is defined by Maxim-Dallas Semiconductor. Figure 21-1 shows the 1-Wire Connection overview. A block-level description of the 1-Wire module is contained in Figure 21-2.

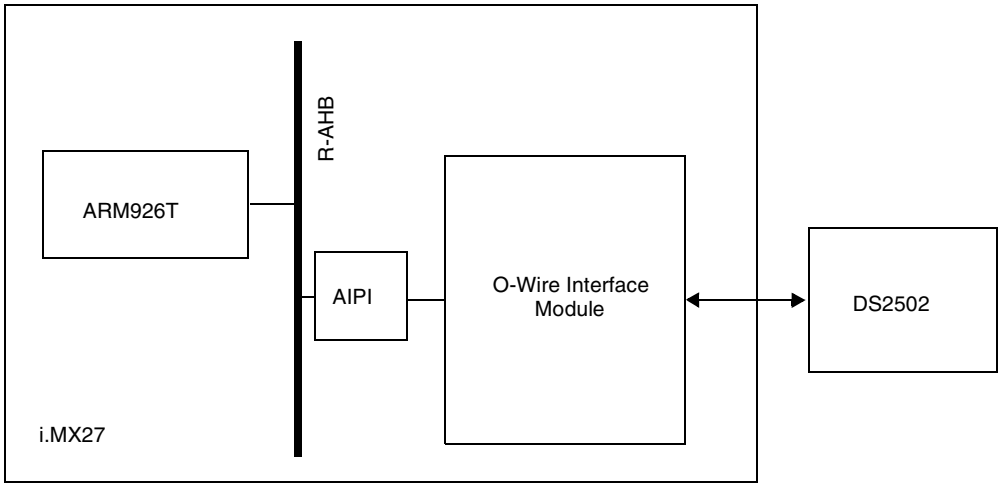


Figure 21-1. 1-Wire Connection

21.1 Overview

This chapter describes the 1-Wire function, timing diagrams, port definitions as well as notes on testing the 1-Wire module. The DS2502 is used to hold battery characteristic information. The clock divider generates a 1 MHz clock used as a time reference by the state machine. Transitions between the states of the state machine as well as actions triggered at precise time deadlines are expressed using this 1-MHz clock. The state machine performs all required actions to dialog with the external device.

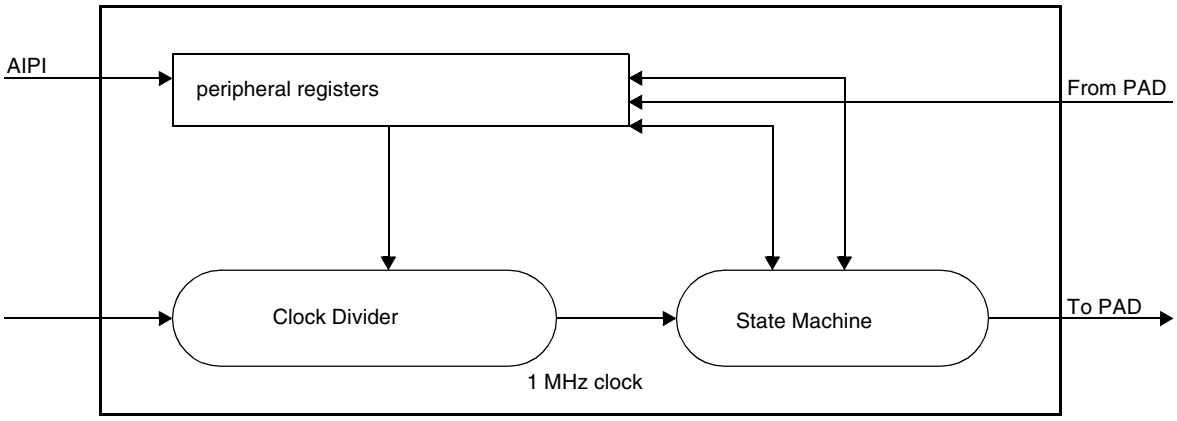


Figure 21-2. 1-Wire Block-Level Description

21.2 Port Definitions

The inputs and outputs for the 1-Wire are listed in Table 21-1. They are organized in such a way to show the major interfaces for the 1-Wire. The DS2502 input and output lines listed in Table 21-1 are the lines that interface with the DS2502. Table 21-1 lists the inputs and outputs relevant for the AIPI bus protocol. In Table 21-1, the clocks are described. In Table 21-1, the test signals are described.

Table 21-1. 1-Wire Port Definitions: DS2502

Signal	I/O	Comments
BATTERY_LINE_IN	input	1-Wire bus.
BATTERY_LINE_OUT	output	Connected to GND for open drain
OUTPUT_ENABLE	output	Enable for output driver 1-Wire bus. In hdl model, represents DS2502 input

Note: The outputs above have been set for a standard I/O pad.
The DS2502 specifies an external 5K pull-up should be used. The i.MX27 provides a 69K pull-up resistor on the 1-Wire pin. An external pull-up is not required if the 1-Wire module is connected within few inches to the DS2502.

21.3 Pin Configuration

Table 21-2 identifies the pin used for the 1-Wire module. This pin is multiplexed with other functions on the device and must be configured for 1-Wire operation.

Table 21-2. 1-Wire Pin Configuration

Module	Setting	Configuration Procedure
GPIO	Alternate Function of GPIO Port E [16]	1. Clear bit 16 of Port E GPIO In Use Register (GIUS_E) 2. Set bit 16 of Port E General Purpose Register (GPR_E)

21.4 Clock Enable and AIPI Configuration

Table 21-3. CRM and API Register Descriptions

Module	Setting	Configuration Procedure
PLL Clock Controller and Reset Module	CRM_PCCR0	Set bit [12] to enable the clock to 1-Wire
AIPI	AIPI1_PSR0 and AIPI1_PSR1	Set AIPI1_PSR0 bit [9] and Clear AIPI1_PSR1 bit [9] to match 1-Wire bus width 16 bits.

21.5 Functional Description

The 1-Wire interfaces with the 1Kbit Add-only Memory (DS2502) through a simple 1 bit bus. The DS2502 1 Kbit Add-only Memory, manufactured by Maxim-Dallas Semiconductor, uses the 1-Wire line to program and read a 1024-bit EPROM. The DS2502 also has a 64-bit lasered ROM and status bytes. The DS2502 requires a special protocol to access the EPROM. The protocol involves first issuing one of four ROM function commands before the EPROM is accessible: read ROM, match ROM, search ROM and skip ROM. Through the 1-Wire bus, the ARM926EJ-S Core interfaces with the DS2502 and allows the required commands to be issued to control the EPROM. The ARM926EJ-S (through the 1-Wire interface) is the bus master and the DS2502 device(s) are the slave(s). The 1-Wire peripheral does not trigger interrupts; hence a polling of the 1-Wire module register is necessary to manage a correct operation of the block.

21.5.1 Low-Power Modes

When the 1-Wire module enters a low power mode it gates off its clock when it is not in use—that is, when the RPP, WR0, and WR1 bits in the Control register are all cleared.

21.5.2 Reset Sequence with Reset Pulse Presence Pulse

To begin any communications with the DS2502, it is required that an initialization procedure be issued. A reset pulse must be generated and then a presence pulse must be detected. The minimum reset pulse length is 480 us. The bus master (1-Wire) will generate this pulse, then after the DS2502 detects a rising edge on the 1-Wire bus, it will wait 15-60 µs before it will transmit back a presence pulse. The presence pulse will exist for 60–240 µs.

The timing diagram for this sequence is shown in Figure 21-3.

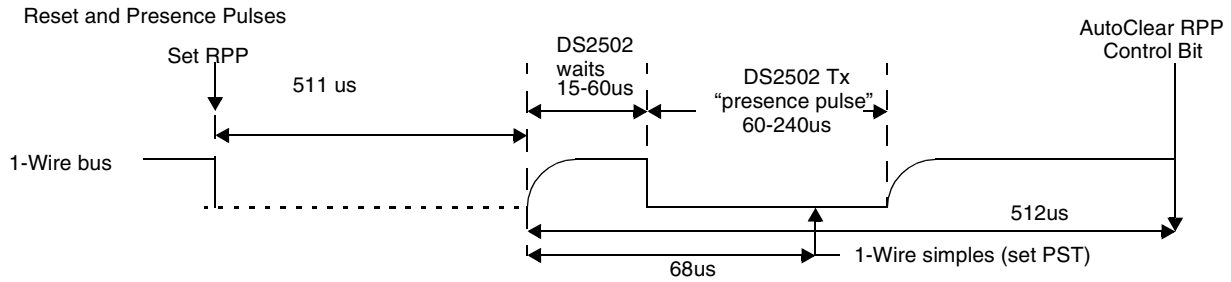


Figure 21-3. 1-Wire Initialization

The reset pulse begins the initialization sequence and it is initiated when the RPP control register bit is set. When the presence pulse is detected, this bit will be cleared. The presence pulse is used by the bus master to determine if at least one DS2502 is connected. Software will determine if more than one DS2502 exists. The 1-Wire module will sample for the DS2502 presence pulse. The presence pulse is latched in the 1-Wire control register PST. When the PST bit is set to a one, it means that a DS2502 is present; if the bit is set to a zero, then no device was found.

21.5.3 Write 0

The Write 0 function simply writes a zero bit to the DS2502. The sequence takes 117 µs. The 1-Wire bus is held low for 100 µs. Figure 21-4 shows the Write 0 timing.

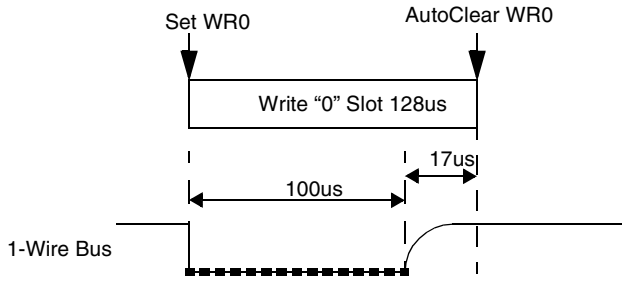


Figure 21-4. Write 0 Timing

The Write 0 pulse sequence is initiated when the WR0 control bit register is set. When the write is complete, the WR0 register will be auto cleared.

21.5.4 Write 1 and Read Data

The Write 1 and Read timing is identical. The time slot is first driven low. According to the DS2502 documentation, the DS2502 has a delay circuit which is used to synchronize the DS2502 with the bus master (1-Wire). This delay circuit is triggered by the falling edge of the data line and is used to decide when the DS2502 will sample the line. In the case of a write 1 or read 1, after a delay, a 1 will be transmitted/received. When a read 0 slot is issued, the delay circuit will hold the data line low to override the 1 generated by the bus master (1-Wire).

For the Write 1 or Read, the control register WR1/RD is set and auto-cleared when the sequence has been completed. After a Read, the control register RDST bit is set to the value of the read. Figure 21-5 shows the Write 1 timing.

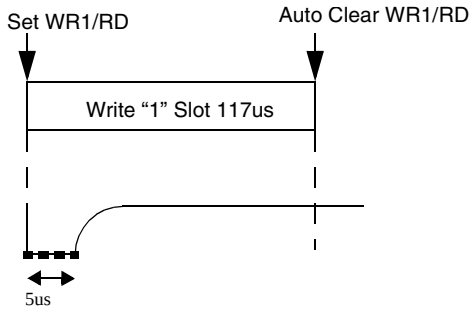


Figure 21-5. Write 1 Timing

Figure 21-6 shows the read timing.

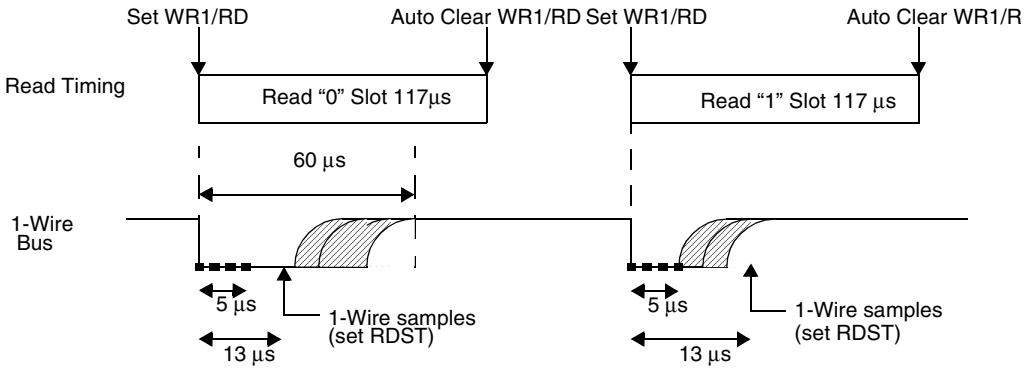


Figure 21-6. Read Timing

21.5.5 Program Pulse

The Program Pulse sequence is described in the DS2502 documentation as one of the functions of the 1-Wire signaling. The 12-volt programming pulse function is not used in the 1-Wire.

21.6 Memory Map and Register Definition

The 1-Wire module includes three user-accessible 16-bit registers. Table 21-6 summarizes these registers and their addresses.

Table 21-4. 1-Wire Memory Map

Address	Register	Reset Value	Access	Section/Page
0x1000_9000 (CONTROL)	Control register	0x0000	R/W	21.6.1.1/21-6
0x1000_9002 (TIME_DIVIDER)	Time divider register	0x0000	R/W	21.6.2/21-7
0x1000_9004 (RESET)	Reset register	0x0000	R/W	21.6.3/21-9

21.6.1 Register Summary

Figure 21-7 shows the key to the register fields, and Table 21-5 shows the register figure conventions.

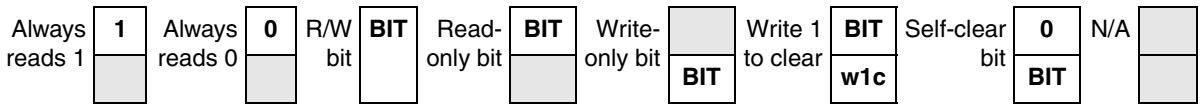


Figure 21-7. Key to Register Fields

Table 21-5. Register Figure Conventions Key to Register Fields

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.

Table 21-5. Register Figure Conventions Key to Register Fields (continued)

Convention	Description
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 21-6 shows the 1-Wire register summary.

Table 21-6. 1-Wire Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000_9000 (CONTROL)	R	0	0	0	0	0	0	0	0	RPP	PST	WR 0	WR 1	RDS T	0	0	0
	W																
0x1000_9002 (TIME_DIVIDER)	R	0	0	0	0	0	0	0	0	DVDR							
	W																
0x1000_9004 (RESET)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RES ET
	W																

21.6.1.1 Control Register (CONTROL)

The control register updates the status of the reset, presence, write0, write1, and read bits. when read, this register lets the user know whether the device (1-Wire) is connected.

Figure 21-8 shows the CONTROL register, and Table 21-7 shows the register's field descriptions.

0x1000_9000 (CONTROL) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RPP	PST	WR0	WR1	RDST	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-8. Control Register

Table 21-7. Control Register Field Descriptions

Field	Description
15–8	Reserved
7 RPP	Reset presence pulse. This bit is self-clearing and is cleared after the presence is determined. 0 Does nothing. Reset pulse is complete. 1 Generates a reset pulse and a sample for DS2502 presence pulse.
6 PST	Presence status. This bit is valid after the RPP bit is self-cleared. 0 Device is not present. 1 Device is present.
5 WR0	Write 0. This bit is self-clearing and is cleared when the write of the bit is complete. 0 Do nothing./ Write sequence complete. 1 Write a 0 bit to the interface.
4 WR1	Write 1/ Read. This bit is self-clearing and is cleared when the write of the bit is complete. This reads a bit, since the Write 1 and Read timing are identical. The value of the read bit is stored in RDST, and is valid after WR1/RD is self-cleared. 0 Do nothing./Write sequence complete. 1 Write a 1 bit to the interface.
3 RDST	Read status. This bit is valid after the WR1/RD bit is self cleared. 0 A 0 was sampled during a read. 1 A 1 was sampled during a read.
2–0	Reserved

21.6.2 Time Divider Register (TIME_DIVIDER)

1-Wire Time Divider Register clock divider register used to generate the internal time base within the module. Internal time generation is made up by a clock divider. The purpose of this internal time generation is to make a 1 MHz clock from the main clock.

Figure 21-9 shows the TIME_DIVIDER register, and Table 21-8 shows the register's field descriptions. Table 21-9 shows the system timing requirements.

0x1000_9002 (TIME_DIVIDER) Access: User read/write

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

Figure 21-9. Time Divider Register

Table 21-8. Time Divider Register Field Descriptions

Field	Description
15–8	Reserved
7–0 DVDR	Pre-divider factor. The 1-Wire also contains a clock divider register used to generate the internal time base within the module. Internal time generation is made up by a clock divider. The purpose of this internal time generation is to make a 1 MHz clock from the main clock. It is the user's responsibility to program this register so that the binary rate frequency is as close as possible to 1 MHz (1 /(divider+1). If the clock frequency is 30 MHz, then the proper value to write to the divider register is 29. 00 1 (default) 01 2 - - - - - - FF 256

Note: It is the user's responsibility to program this register so that the binary rate frequency is as close as possible to 1 MHz (1 MHz = clock / (divider + 1)). If the clock frequency is 30 MHz, then the proper value to write into the divider register is 29.

NOTE

The precision of the generated clock is very important to ensure the proper operation of the 1-Wire module. This module is based on a state machine which undertakes actions at defined times.

Table 21-9. System Timing Requirements

Times	Values (μs)	Minimum (μs)	Maximum (μs)	Absolute Precision	Relative Precision
RSTL	511	480	—	31	0.0645
PST	68	60	75	7	0.1
RSTH	512	480	—	32	0.0645
LOW0	100	60	120	20	0.2
LOWR	5	1	15	4	0.8
READ_sample	13	—	15	2	0.15

The most stringent constraint is 0.0645 as a relative time imprecision.
The time relative precision is directly derived from the frequency of the derivative clock (f):
time relative precision = 1/f –1 = divider/clock (MHz) –1

The Table 21-10 shows the relative time precision for different main clock frequencies.

Table 21-10. System Clock Requirements

Main Clock Frequency (MHz)	13	16.8	19.44
Clock divide ratio	13	17	19
Generated frequency (MHz)	1	0.9882	1.023
Relative time imprecision	0	0.0117	0.023

This demonstrates that the user must use care when selecting the main clock frequency if using the 1-Wire module. If the main clock is an exact integer multiple of 1 MHz, then the generated frequency will be exactly 1 MHz.

NOTE

A main clock frequency below 10 MHz could cause stability problems and incorrect operation in the 1-Wire module.

21.6.3 Reset Register

The Reset Register is used to reset the 1-Wire module through software. This register is not self-clearing, therefore the programmer must write a 1 to reset the register and then write a 0 to release the reset signal.

Figure 21-10 shows the RESET register, and Table 21-11 shows the register's field descriptions.

0x1000_9004 (RESET) Access: User read/write

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

Figure 21-10. Reset Register

Table 21-11. Reset Register Field Descriptions

Field	Description
15–1	Reserved
0 RST	Software reset. The reset register is used to reset the module using the software. This register is not self-clearing; therefore, the programmer must write a ‘1’ to reset the registers and then write a ‘0’ to release the reset signal. 0 1-Wire is out of reset. 1 1-Wire is in reset.

Chapter 22
Advanced Technology Attachment (ATA)

The ATA host controller complies with the ATA/ATAPI-6 specification. Its main use is to interface with IDE hard disc drives and ATAPI optical disc drives. It interfaces with the ATA device over a number of ATA signals.See Figure 22-1 for the block diagram of the ATA host controller.

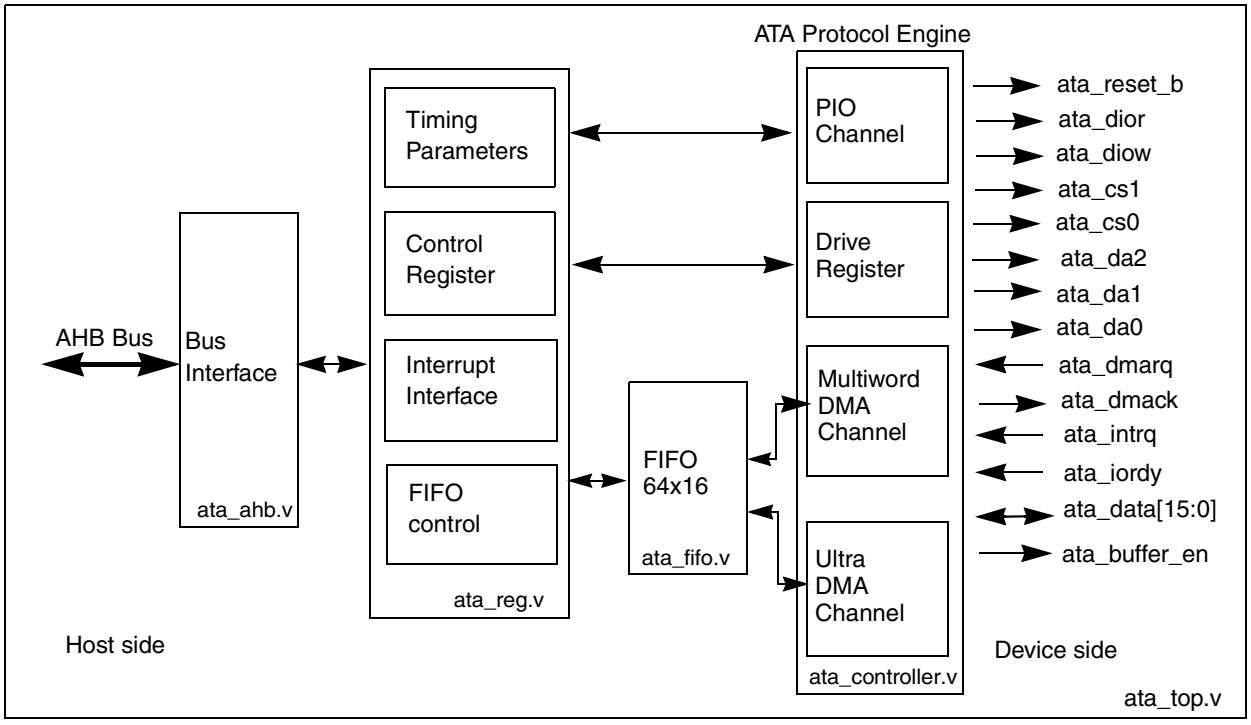


Figure 22-1. ATA Host Controller Block Diagram.

22.1 Overview

The ATA Host Controller consists of a bus interface compliant with AHB bus protocols, a control register for register setting, a 64x16 data FIFO and an ATA protocol engine. The ATA block is an AT attachment host interface. Its main use is to interface with hard disc drives and optical disc drives which compiles with ATA/ATAPI-6 standard. It interfaces with the ATA device over a number of ATA signals. It is possible to connect a bus buffer between the host side and the device side. In this case, the ata_buffer_en signal should be used to control the direction the buffer is driving to. If ata_buffer_en is high, it drives outward to the device. If ata_buffer_en is low, it drives inward to the host.

The ATA Host Controller supports interface protocols as specified in ATA/ATAPI-6 standard:

- PIO mode 0, 1, 2, 3, and 4

- Multiword DMA mode 0, 1, and 2
- Ultra DMA modes 0, 1, 2, 3, and 4 with bus clock of 50 MHz or higher
- Ultra DMA mode 5 with bus clock of 80 MHz or higher

Before accessing the ATA bus, the host must program the timing parameters to be used on the ATA bus. The timing parameters control the timing on the ATA bus. Most timing parameters are programmable as a number of clock cycles (1 to 255). Some are implied. All of the ATA device internal registers are visible to users, and they are defined as mirror registers in ATA host controller. As specified in ATA/ATAPI-6 standard, all the features/functions are implemented by reading/writing to the device internal registers.

After programming the timing parameters, there are two protocols that can be active at the same time on the ATA bus:

- First protocol. This protocol is a PIO mode access that can be performed at any time by the host to the ATA bus. During PIO mode access, the incoming AHB bus cycle is translated into an ATA bus cycle by the ATA protocol engine. The AHB bus cycle is stalled until completion of the ATA bus cycle on read, or until putting the write data on the ATA bus on write. The PIO mode is a slow protocol, mainly intended to program the ATA disc drive, but also possible to use to transfer data to/from the disc drive. During PIO mode, the FIFO is not active.
- Second protocol. This protocol is the DMA mode access. DMA mode is started by the ATA interface after receiving a DMA request from the drive, and only if the ATA interface has been programmed to accept the DMA request. In DMA mode, either multiword DMA or ultra DMA protocol is used on the ATA bus. Once started, data transfer is organized between the ATA bus and the FIFO. Data transfer will pause to prevent FIFO overflow/FIFO underflow. Data transfer will resume when there is again space in the FIFO, or when the FIFO has been refilled. During DMA transfer, there is no direct data transfer between the ATA bus and the host CPU or host DMA bus. Instead, the transfer takes place between the ATA bus and the FIFO; the FIFO informs the host DMA unit when it needs to be refilled or emptied. In this case, it sends an FIFO ALARM flag to the host DMA. When the host DMA receives the fifo_tx_alarm, it should write some data to the FIFO. (typically 32 bytes). When the host DMA receives the fifo_rcv_alarm, it should read some data from the FIFO (typically 32 bytes). The FIFO filling level at which the alarms are produced, is programmable. For completion, there is a third alarm associated with the host DMA operation fifo_txfer_end_alarm. This alarm signals the end of the transfer, and requests the host DMA to take steps to complete the transfer: transfer the bytes remaining in the FIFO to the host memory, and inform the host CPU the transfer is completed.

All transfers between FIFO and host CPU or DMA bus are zero wait states transfer, so high speed transfer between FIFO and host DMA bus is possible.

When a PIO access is performed during a running DMA transfer, the DMA transfer will be paused, the PIO access done, and the DMA transfer will resume again.

22.2 Features

The ATA host controller includes the following major features:

Programmable timing on the ATA bus. Works with wide range of bus clock frequencies.

- Compliant with ATA/ATAPI-6 standard
  - — Supports PIO modes 0, 1, 2, 3, and 4
  - — Supports multiword DMA modes 0, 1, and 2
  - — Supports ultra DMA modes 0, 1, 2, 3, and 4 with bus clock of at least 50 MHz
  - — Supports ultra DMA mode 5 with bus clock of at least 80 MHz
- Can be used with off-chip bus transceiver if pads are not compliant with ATA voltage levels
- 64-halfword FIFO part of interface
- FIFO receive alarm, FIFO transmit alarm, and FIFO end of transmission alarm to host DMA unit
- Zero-wait cycles transfer between host DMA bus and FIFO allows fast FIFO reading/writing

22.3 Operation

The interface offers two transfer modes that can be used together.

22.4 PIO Mode

An access to the ATA bus in PIO mode occurs whenever a ATA PIO register is read or written by the host CPU or the host (smart) DMA unit. During a PIO transfer the incoming AHB bus cycle is translated into an ATA PIO bus cycle by the ATA protocol engine. No buffering of data occurs, so the host CPU or host DMA cycle is stalled until the ATA bus read data is available on read, or is stalled until the AHB bus data can we put on the ATA bus during write.

PIO accesses can be done to the bus at any time, even during a running ATA DMA transfer. In this case, the DMA transfer is paused, the PIO cycle is completed, and the DMA transfer is resumed.

22.4.1 DMA Mode (Multi-Word DMA and Ultra DMA)

In DMA mode, data is transferred between the ATA bus and the FIFO. Two different DMA protocols are supported on the ATA bus: ultra DMA mode and multi-word DMA mode. Selection is by using a control register bit.

A DMA transfer will be started when DMA mode transfer has been enabled by writing some control bit, and when the drive connected to the ATA bus pulls its DMARQ line high.
During an ATA bus DMA transfer, data is transferred between the ATA bus and the FIFO. The transfer will pause to avoid FIFO overflow and FIFO underflow.

It is the task of the host CPU or the host smart DMA unit to read data or write data to the FIFO to keep the transfer going. Normal set-up is that the host (smart) DMA unit takes on this task. For this purpose, the fifo_rcv_alarm and fifo_tx_alarm signals are sent to the host DMA unit. fifo_rcv_alarm informs the host DMA unit that there is at least 1 packet of data waiting in the FIFO to be read by the host DMA. Whenever this signal is high, the host DMA should transfer one packet of data from the FIFO to the main memory. Typical packet size is 32 bytes (8 long words), but other packet sizes can be handled too. fifo_tx_alarm informs the host DMA unit that there is space for at least 1 packet to be written by the host DMA. Whenever this signal is high, the host DMA should transfer one packet of data from main memory to the FIFO. Typical packet size is 32 bytes (8 long words), but other packet sizes can be handled too.

22.5 External Signal Description

See Table 22-1 for the list of signals entering and existing this module to peripherals within the i.MX27 chip.

Table 22-1. Signal Properties

Name	Port	Function	Reset State	Type
External Signals				
ipp_do_ata_reset_b	out	ATA bus reset signal. Active low. If active, ata device is reset[1]	0	—
ipp_do_ata_dior	out	ATA bus read strobe	1	—
ipp_do_ata_diow	out	ATA bus write strobe	1	—
ipp_do_ata_cs1	out	ATA bus chip select 1	1	—
ipp_do_ata_cs1	out	ATA bus chip select 0	1	—
ipp_do_ata_da2	out	ATA bus address line 2	0	—
ipp_do_ata_da1	out	ATA bus address line 1	0	—
ipp_do_ata_da0	out	ATA bus address line 0	0	—
ipp_ind_ata_dmarq	in	ATA bus DMA request	—	—
ipp_do_ata_dmack	out	ATA bus DMA acknowledge	1	—
ipp_ind_ata_intrq	in	ATA bus interrupt request	—	—
ipp_ind_ata_iordy	in	ATA bus iordy	—	—
ipp_do_ata_data[15:0]	out	ATA output data bus	Hi-z	—
ipp_ind_ata_data[15:0]	in	ATA input data bus	Hi-z	—
ipp_obe_ata_data	out	Data transmit tri-state control signal	0	—
ipp_do_ata_buffer_en	out	Buffer enable for external bus transceiver[2]	0	—
Interface Signals				
ipbus_int	out	Active high ATA interrupt	0	—
DMA Signals				
ata_tx_fifo_alarm	out	DMA transmit fifo alarm request	0	—
ata_rcv_fifo_alarm	out	DMA receive fifo alarm request	0	—
ata_txfer_end_alarm	out	DMA transfer end alarm	0	—

1 This signal is a standard ATA bus signal. It conforms with the ATA-6 standard.

2 It is optional to put a 74xxx245 bus transceiver between the host side of the data bus and the device side of the data bus. If the transceiver is used, its enable should be tied low (always enable), and its direction pin should be tied to ata_buffer_en, in such a way that it drives from host to device when ata_buffer_en is high, and drives from device to host when ata_buffer_en is low.

22.5.1 Detailed Signal Descriptions

The following subsections describe each external signals separately. For a detailed description of the ATA bus signal, refer to the ATA/ATAPI-6 standard.

22.5.1.1 ipp_do_ata_reset_b (out)

This signal is the ATA reset signal. When low, the ATA bus is in reset state. When high, no reset. The ATA bus is in reset whenever the appropriate bit in the control register is cleared. After system reset, the ATA bus is in reset.

22.5.1.2 ipp_do_ata_dior (out)

This signal correspond to ATA signal DIOR. During PIO and multiword DMA transfer, function is read strobe. During ultra DMA in burst, function is HDMARDY. During ultra DMA out burst, function is host strobe (HSTROBE).

22.5.1.3 ipp_do_ata_diow (out)

This signal corresponds to ATA signal DIOW. During PIO and multiword DMA transfer, function is write strobe. During ultra DMA burst, function is STOP, signalling whenever host wants to terminate running ultra DMA transfer.

22.5.1.4 ipp_do_ata_cs0, ipp_do_ata_cs1, ipp_do_ata_da2, ipp_do_ata_da1, ipp_do_ata_da0 (out)

These signals are the address group of the ATA bus. ata_cs0, ata_cs1 are the chip selects; ata_da2, ata_da1 and ata_da0 are the 3 address lines. All these five lines follow the same timing.

22.5.1.5 ipp_ind_ata_dmarq (in)

This signal is the ATA bus device DMA request (DMARQ). Its pulled high by the device if it wants to transfer data using multiword DMA or ultra DMA mode.

22.5.1.6 ipp_do_ata_dmack (out)

This signal is the ATA bus host DMA acknowledge (DMACK). Its pulled low by the host when it grants the DMA request.

22.5.1.7 ipp_ind_ata_intrq (in)

This signal is the ATA bus interrupt request (INTRQ). Its pulled high by the device whenever it wants to interrupt the host CPU.

22.5.1.8 ipp_ind_ata_iordy (in)

This signal is the ATA bus IORDY line. It has three functions:

- IORDY—Active low wait during PIO cycles.
- DDMARDY—Active low device ready during ultra DMA out transfers.
- DSTROBE—Device strobe during ultra DMA in transfers.

22.5.1.9 ipp_do_ata_data[15:0] (out)

This is the module output data to ata bus.

22.6 Memory Map and Register Definition

Section 22.6.3, "Register Descriptions" provides the detailed descriptions for all of the ATA registers.

22.6.1 Memory Map

Table 22-2 shows the ATA memory map.

Table 22-2. ATA Memory Map

Address	Description	Access	Reset Value	Section/Page
0x8000_1000 (TIME_CONFIG0)	ATA timing parameter 0	R/W	0x0101_0101	22.6.3.1.1/22-11
0x8000_1004 (TIME_CONFIG1)	ATA Timing Parameter 1	R/W	0x0101_0101	22.6.3.1.2/22-11
0x8000_1008 (TIME_CONFIG2)	ATA Timing Parameter 2	R/W	0x0101_0101	22.6.3.1.3/22-12
0x8000_100C (TIME_CONFIG3)	ATA Timing Parameter 3	R/W	0x0101_0101	22.6.3.1.4/22-13
0x8000_1010 (TIME_CONFIG4)	ATA Timing Parameter 4	R/W	0x0101_0101	22.6.3.1.5/22-14
0x8000_1014 (TIME_CONFIG5)	ATA Timing Parameter 5	R/W	0x0101_0101	22.6.3.1.6/22-14
0x8000_1018 (FIFO_DATA_32)	32-Bit Wide Data Port to/from FIFO	R/W	0x0000_0000	22.6.3.2.1/22-15
0x8000_101C (FIFO_DATA_16)	16-Bit Wide Data Port to/from FIFO	R/W	0x0000_0000	22.6.3.2.2/22-16
0x8000_1020 (FIFO_FILL)	FIFO Filling In Half Words	R	0x0000_0000	22.6.3.3/22-17
0x8000_1024 (ATA_CONTROL)	ATA Interface Control Register	R/W	0x0000_0000	22.6.3.4/22-17
0x8000_1028 (INT_PENDING)	Interrupt Pending Register	R	0x0000_0010	22.6.3.5/22-19
0x8000_102C (INT_ENABLE)	Interrupt Enable Register	R/W	0x0000_0000	22.6.3.5/22-19
0x8000_1030 (INT_CLEAR)	Interrupt Clear Register	W	0x0000_00— —	22.6.3.5/22-19

Table 22-2. ATA Memory Map (continued)

Address	Description	Access	Reset Value	Section/Page
0x8000_1034 (FIFO_ALARM)	FIFO Alarm Threshold	R/W	0x0000_0001	22.6.3.6/22-21
0x8000_A0 (DDTR)	Drive Data Register	16-bit RW	— — — —	22.6.3.7/22-22
0x8000_A4 (DFTR)	Drive Features Register	R/W	— — — —	22.6.3.7/22-22
0x8000_A8 (DSCR)	Drive Sector Count Register	R/W	— — — —	22.6.3.7/22-22
0x8000_AC (DSNR)	Drive Sector Number Register	R/W	— — — —	22.6.3.7/22-22
0x8000_B0 (DCLR)	Drive Cylinder Low Register	R/W	— — — —	22.6.3.7/22-22
0x8000_B4 (DCHR)	Drive Cylinder High Register	R/W	— — — —	22.6.3.7/22-22
0x8000_B8 (DDHR)	Drive Device Head Register	R/W	— — — —	22.6.3.7/22-22
0x8000_BC (DCDR)	Drive Command Register	W	— — — —	22.6.3.7/22-22
0x8000_BC (DCDR)	Drive Status Register	R	— — — —	22.6.3.7/22-22
0x8000_D8 (DCTR)	Drive Alternate Status Register	R	— — — —	22.6.3.7/22-22
0x8000_D8 (DCTR)	Drive Control Register	W	— — — —	22.6.3.7/22-22

22.6.2 Register Summary

Figure 22-2 shows the key to the register fields and Table 22-3 shows the register figure conventions.

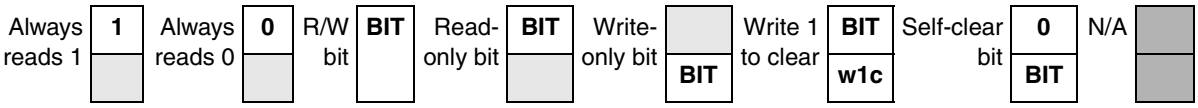


Figure 22-2. Key to Register Fields

Table 22-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	

Table 22-3. Register Figure Conventions (continued)

Convention	Description
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 22-4 shows the ATA register summary.

Table 22-4. ATA Register Summary

Name		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
0x8000_1000 (TIME_CONFIG0)	R	TIME_2W								TIME_1							
	W																
	R	TIME_ON								TIME_OFF							
	W																
0x8000_1004 (TIME_CONFIG1)	R	TIME_4								TIME_PIO_RDX							
	W																
	R	TIME_AX								TIME_2R							
	W																
0x8000_1008 (TIME_CONFIG2)	R	TIME_D								TIME_JN							
	W																
	R	TIME_M								TIME_9							
	W																
0x8000_100C (TIME_CONFIG3)	R	TIME_RPX								TIME_ENV							
	W																
	R	TIME_ACK								TIME_K							
	W																
0x8000_1010 (TIME_CONFIG4)	R	TIME_DZFS								TIME_DVH							
	W																
	R	TIME_MLIX								TIME_ZAH							
	W																
0x8000_1014 (TIME_CONFIG5)	R	TIME_CYC								TIME_SS							
	W																
	R	TIME_CVH								TIME_DVS							
	W																
0x8000_1018 (FIFO_DATA_32)	R	FIFO_DATA_32															
	W																
	R																
	W																

Table 22-4. ATA Register Summary (continued)

Name		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
0x8000_101C (FIFO_DATA_16)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	FIFO_DATA_16															
	W																
0x8000_1020 (FIFO_FILL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	FIFO_FILL[7:0]							
	W																
0x8000_1024 (ATA_CONTROL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	FIFO_	ATA_R	FIFO_	FIFO_	DMA_P	DMA_U	DMA_W	IORDY_
	W																
0x8000_1028 (INT_PENDING)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	ATA	FIF	FIF	CO	ATA	0	0	0
	W																
0x8000_102C (INT_ENABLE)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	ATA_IN	FIFO_	FIFO_	CONT	ATA_IN	0	0	0
	W																
0x8000_1030 (INT_CLEAR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0			0	0	0	0	0
	W										FIFO_UNDER-	FIFO_OVER-					
0x8000_1034 (FIFO_ALARM)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	FIFO_ALARM[7:0]							
	W																
0x8000_A0 (DDTR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DDTR[15:0]															
	W																
0x8000_A4 (DFTR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DFTR[7:0]							
	W																
0x8000_A8 (DSCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DSCR[7:0]							
	W																

Table 22-4. ATA Register Summary (continued)

Name		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
0x8000_AC (DSNR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DSNR[7:0]							
	W																
0x8000_B0 (DCLR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DCLR[7:0]							
	W																
0x8000_B4 (DCHR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DCHR[7:0]							
	W																
0x8000_B8 (DDHR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DDHR[7:0]							
	W																
0x8000_BC (DCDR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DCDR[7:0]							
	W																
0x8000_D8 (DCTR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DCTR[7:0]							
	W																

22.6.3 Register Descriptions

This section contains the detailed register descriptions for the ATA host controller registers and mapped device registers. All ATA host controller registers except FIFO_DATA_16 are 32-bit size accessible. FIFO_DATA_16 is only support 16-bit size accessible. All ATA device registers except DDTR are 8-bit size accessible. The DDTR is only support 16-bit size accessible.

22.6.3.1 Timing Registers

Registers (ata_base +$00) till (ata_base + $17) contain timing parameters. These timing parameters control the timing on the ATA bus.

Every timing parameter is 8-bit wide and can assume valid values between 1 and 255. Reset value is always 1. And all timing parameter registers are 32-bit size accessible.

All figures in this section show timing registers.

22.6.3.1.1 TIME_CONFIG0

See Figure 22-3 for an illustration of valid bits in the ATA TIME_CONFIG0 Register and Table 22-5 for descriptions of the bit fields.

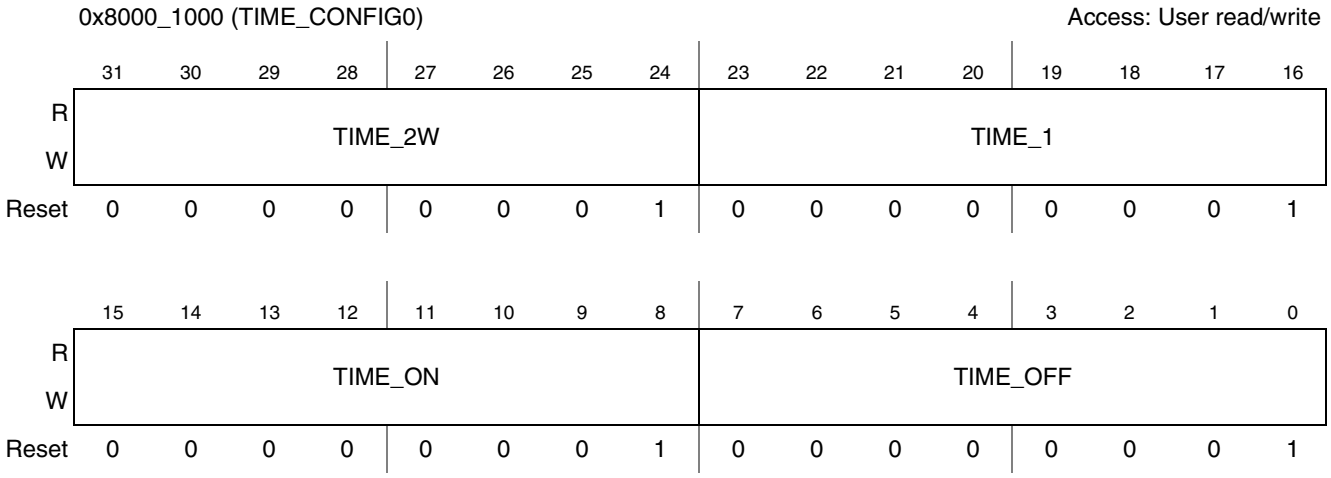


Figure 22-3. ATA TIME_CONFIG0 Register

Table 22-5. ATA TIME_CONFIG0 Register Field Descriptions

Field	Description
31–24 TIME_2W	Pio mode time parameter counter for time of DIOW- pulse width (t2w or t2), these shall be used the max time of 8-bit and 16-bit.
23–16 TIME_1	Pio mode time parameter counter for time of address valid to DIOR-/DIOW- setup(t1).
15–8 TIME_ON	Time parameter counter for transceiver to turn on.
7–0 TIME_OFF	Time parameter counter for transceiver to turn off.

22.6.3.1.2 TIME_CONFIG1

See Figure 22-4 for an illustration of valid bits in the ATA TIME_CONFIG1 Register and Table 22-6 for descriptions of the bit fields.

0x8000_1004 (TIME_CONFIG1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TIME_4								TIME_PIO_RDX							
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TIME_AX								TIME_2R							
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 22-4. ATA TIME_CONFIG1 Register

Table 22-6. ATA TIME_CONFIG1 Register Field Descriptions

Field	Description
31–24 TIME_4	PIO mode time parameter counter for controlling the time of DIOR- data hold (t4).
23–16 TIME_PIO_RDX	Pio mode time parameter counter for controlling the time of read data valid to IORDY active (tRD).
15–8 TIME_AX	Pio time parameter counter for timing of IORDY setup time (tA).
7–0 TIME_2R	Pio mode time parameter counter for time of DIOR- pulse width (t2r or t2), these shall be used the max time of 8-bit and 16-bit.

22.6.3.1.3 TIME_CONFIG2

See Figure 22-5 for an illustration of valid bits in the ATA TIME_CONFIG2 Register and Table 22-7 for descriptions of the bit fields.

0x8000_1008 (TIME_CONFIG2) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TIME_M								TIME_9							
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 22-5. ATA TIME_CONFIG2 Register

Table 22-7. ATA TIME_CONFIG2 Register Field Descriptions

Field	Description
31–24 TIME_D	Multiword DMA mode time parameter counter for time of DIOR-/DIOW- asserted pulse width (tD).
23–16 TIME_JN	Multiword DMA mode time parameter counter for time of DIOW- data hold time (tH).
15–8 TIME_M	Multiword DMA mode time parameter counter for time from CS valid to DIOR-/DIOW- (tM).
7–0 TIME_9	Pio mode time parameter counter for controlling the DIOR-/DIOW- to address valid hold time.

22.6.3.1.4 TIME_CONFIG3

See Figure 22-6 for an illustration of valid bits in the ATA TIME_CONFIG3 Register and Table 22-8 for descriptions of the bit fields.

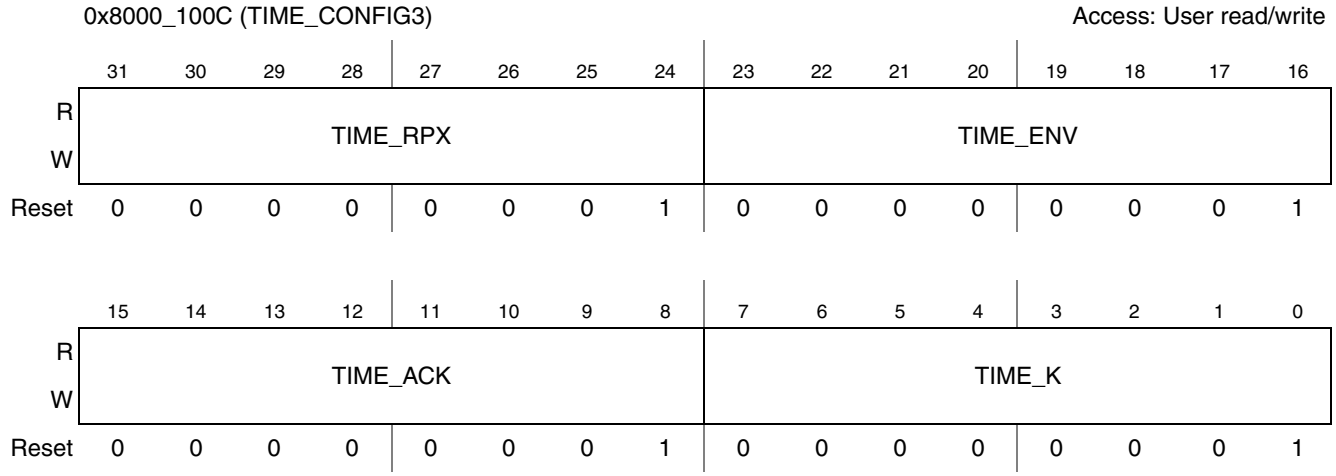


Figure 22-6. ATA TIME_CONFIG3 Register

Table 22-8. ATA TIME_CONFIG3 Register Field Descriptions

Field	Description
31–24 TIME_RPX	Ultra DMA mode time parameter counter for time of tRP.
23–16 TIME_ENV	Ultra DMA mode time parameter counter for min. time of tENV.
15–8 TIME_ACK	Ultra DMA mode time parameter counter for time of tACK.
7–0 TIME_K	Multiword DMA mode time parameter counter for time of DIOW- negated pulse width (tKW).

22.6.3.1.5 TIME_CONFIG4

See Figure 22-7 for an illustration of valid bits in the ATA TIME_CONFIG4 Register and Table 22-9 for descriptions of the bit fields.

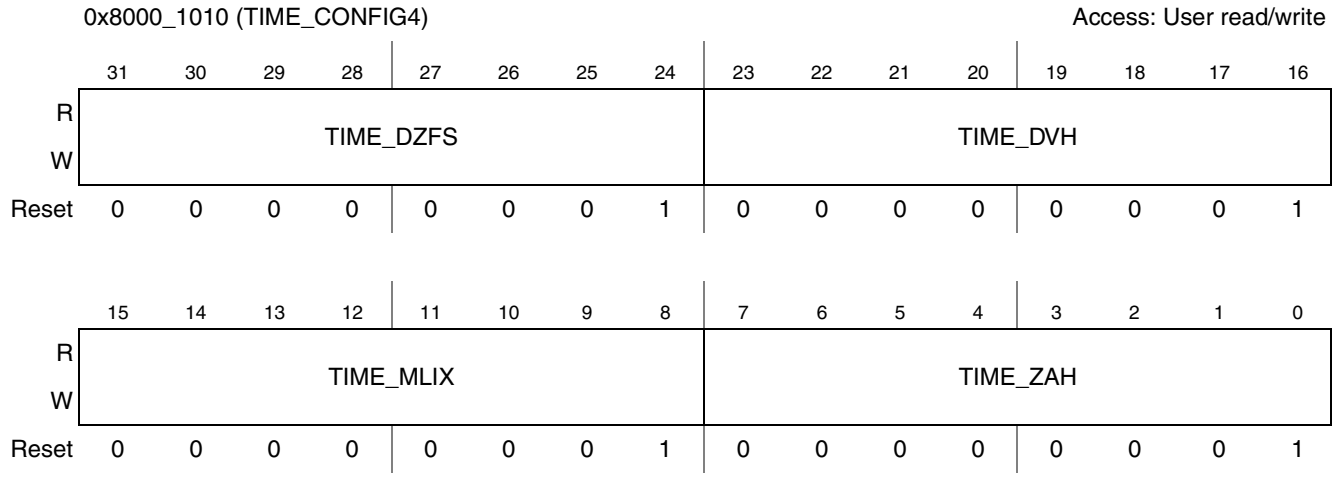


Figure 22-7. ATA TIME_CONFIG4 Register

Table 22-9. ATA TIME_CONFIG4 Register Field Descriptions

Field	Description
31–24 TIME_DZFS	Ultra DMA mode time parameter counter for tDZFS.
23–16 TIME_DVH	Ultra DMA mode time parameter counter for tDVH.
15–8 TIME_MLIX	Ultra DMA mode time parameter counter for tMLI.
7–0 TIME_ZAH	Ultra DMA mode time parameter counter for tZAH.

22.6.3.1.6 TIME_CONFIG5

See Figure 22-8 for an illustration of valid bits in the ATA TIME_CONFIG5 Register and Table 22-10 for descriptions of the bit fields.

0x8000_1014 (TIME_CONFIG5) Access: User read/write

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

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

Figure 22-8. ATA TIME_CONFIG5 Register

Table 22-10. ATA TIME_CONFIG5 Register Field Descriptions

Field	Description
31–24 TIME_CYC	Ultra DMA mode time parameter counter for tCYC.
23–16 TIME_SS	Ultra DMA mode time parameter counter for tSS.
15–8 TIME_CVH	Ultra DMA mode time parameter counter for tCVH.
7–0 TIME_DVS	Ultra DMA mode time parameter counter for tDVS.

22.6.3.2 FIFO Data Registers

The FIFO_DATA register is used to read or write data to the internal FIFO. It can be accessed as a 16-bit register or as a 32-bit register. Any long write to the register will put the four bytes written into the FIFO. Any word write will put the two bytes written into the FIFO. Any long read will read four bytes from the FIFO. Any word read will read two bytes from the FIFO.

22.6.3.2.1 FIFO_DATA_32 Register in 32-bit Mode

See Figure 22-9 for an illustration of valid bits in the FIFO_DATA_32 Register in 32-bit mode and Table 22-11 for descriptions of the bit fields.

0x8000_1018 (FIFO_DATA_32) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FIFO_DATA_32[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIFO_DATA_32[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-9. ATA FIFO_DATA_32 Register

Table 22-11. ATA FIFO_DATA_32 Register Field Descriptions

Field	Description
31–0 FIFO_DATA_32	Read/Write 32-bit data from/to the FIFO, reads from this register return zero when the FIFO is empty.

22.6.3.2.2 FIFO_DATA_16 Register

See Figure 22-10 for an illustration of valid bits in the FIFO_DATA_16 Register in 16-bit mode and Table 22-12 for descriptions of the bit fields.

0x8000_101C (FIFO_DATA_16) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIFO_DATA_16[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-10. ATA FIFO_DATA_16 Register

Table 22-12. FIFO_DATA_16 Register Field Descriptions

Field	Description
31–16	Reserved.
15–0 FIFO_DATA_16	Read/Write 16-bit data from/to the FIFO, reads from this register return zero when the FIFO is empty.

22.6.3.3 FIFO_FILL Register

See Figure 22-11 for an illustration of valid bits in the FIFO_FILL Register and Table 22-13 for descriptions of the bit fields.

0x8000_1020 (FIFO_FILL) Access: User read only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	FIFO_FILL[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-11. ATA FIFO_FILL Register

Table 22-13. FIFO_FILL Register Field Descriptions

Field	Description
31–8	Reserved.
7–0 FIFO_FILL	FIFO_FILL is a read-only register. Any read to it returns the current number of half-words present in the fifo.

22.6.3.4 ATA_CONTROL Register

See Figure 22-12 for an illustration of valid bits in the ATA_CONTROL Register and Table 22-13 for descriptions of the bit fields.

0x8000_1024 (ATA_CONTROL) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	FIFO_RST_B	ATA_RST_B	FIFO_TX_EN	FIFO_RCV_EN	DMA_PENDING	DMA_ULTRA_SELECTED	DMA_WRITE	IORDY_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-12. ATA_CONTROL Register

Table 22-14. ATA Control Register Field Descriptions

Field	Description
31–8	Reserved
7 FIFO_RST_B	This field controls if the internal FIFO is in reset or enabled 0 FIFO reset 1 FIFO normal operation
6 ATA_RST_B	This bit controls the level on the ata_reset_b pin, and controls the reset of the internal ata protocol engine. 0 ATA_RST_B = 0, ata drive is reset, and internal protocol engine reset. 1 ATA_RST_B = 1, ata drive is not reset and internal protocol engine normal operation.
5 FIFO_TX_EN	FIFO transmit enable. This bit controls if the FIFO will make transmit data requests to the DMA. If enabled, the FIFO will request the DMA to refill it whenever FIFO filling drops below the alarm level. 0 FIFO refill by DMA disabled 1 FIFO refill by DMA enabled
4 FIFO_RCV_EN	FIFO receive enable. This bit controls if the FIFO will make receive data requests to the DMA. If enabled, the FIFO will request the DMA to empty it whenever FIFO filling becomes greater or equal to the alarm level. 0 FIFO empty by DMA disabled 1 FIFO empty by DMA enabled
3 DMA_PENDING	DMA pending bit. This bit controls if the ATA interface will respond to a DMA request originating in the drive. If this bit is asserted, the ATA interface will start a multi-word DMA or ultra DMA burst whenever the drive asserts ata_dmarq. 0 ATA interface will not start DMA burst 1 ATA interface will start multi-word DMA or ultra DMA burst whenever drive asserts dmarq
2 DMA_ULTRA_SELETED	This bit indicates if a DMA burst started, the UDMA or MDMA protocol will be used 0 Multiword DMA protocol will be used 1 Ultra DMA protocol will be used

Table 22-14. ATA Control Register Field Descriptions

Field	Description
1 DMA_WRITE	This bit indicates the data direction on any DMA burst started 0 DMA in burst, ATA interface reads from drive 1 DMA out burst, ATA interface writes to drive
0 IORDY_EN	This bit indicates if the ata_iordy handshake will be used during PIO mode 0 IORDY will be disregarded 1 IORDY handshake will be used

22.6.3.5 INT_PENDING, INT_ENABLE, INT_CLEAR Registers

These 3 registers control interrupts coming from the ATA and going to the CPU and DMA.

See Figure 22-13 for an illustration of valid bits in the INT_PENDING Register and Table 22-15 for descriptions of the bit fields.

0x8000_1028 (INT_PENDING) Access: User read only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ATA_INTRQ1	FIFO_UNDERFLOW	FIFO_OVERFLOW	CONTROLLER_IDLE	ATA_INTRQ2	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0[1]	0	0	1	0	0	0	0

Figure 22-13. ATA INT_PENDING Register

[1] Interrupts ata_intrq1 and ata_intrq2 only reset to 0 if during reset the interrupt input is low.

See Figure 22-14 for an illustration of valid bits in the INT_ENABLE Register and Table 22-15 for descriptions of the bit fields.

0x8000_102C (INT_ENABLE) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ATA_INTRQ1	FIFO_UNDERFLOW	FIFO_OVERFLOW	CONTROLLER_IDLE	ATA_INTRQ2	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-14. ATA INT_ENABLE Register

See Figure 22-15 for an illustration of valid bits in the INT_CLEAR Register and Table 22-15 for descriptions of the bit fields.

0x8000_1030 (INT_CLEAR) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0			0	0	0	0	0
W										FIFO_UNDERFLOW	FIFO_OVERFLOW					
Reset	0	0	0	0	0	0	0	0	0	—	—	—	—	—	—	—

Figure 22-15. ATA INT_CLEAR Register

Table 22-15. INT_PENDING Register Field Description

Field	Description
31–8	Reserved.
7 ATA_INTRQ1	ATA interrupt request 1. This bit reflects the value of the ata_intrq interrupt input. It is set in the interrupt pending register when the drive interrupt is pending, cleared otherwise. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, fifo_txfer_end_alarm will be asserted, signalling the DMA the end of the transfer. The interrupt clear register has no influence on this bit.

Table 22-15. INT_PENDING Register Field Description (continued)

Field	Description
6 FIFO_UNDERFLOW	FIFO underflow. This bit reports FIFO underflow. Sticky bit. It is set in the interrupt pending register when there is a FIFO underflow condition. It is cleared by writing a '1' to this bit in the interrupt clear register. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int will be active, signalling interrupt to the CPU.
5 FIFO_OVERFLOW	FIFO overflow. This bit reports FIFO overflow. Sticky bit. It is set in the interrupt pending register when there is a FIFO overflow condition. It is cleared by writing a '1' to this bit in the interrupt clear register. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int will be active, signalling interrupt to the CPU.
4 CONTROLLER_IDLE	Controller Idle. This bit reports controller idle. It is set when the ATA protocol engine is idle, there is no activity on the ATA bus. It is cleared when there is activity on the ATA bus. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int will be active, signalling interrupt to the CPU. The interrupt clear register has no influence on this bit.
3 ATA_INTRQ2	ATA interrupt request 2. This bit reflects the value of the ata_intrq interrupt input. It is set in the interrupt pending register when the drive interrupt is pending, cleared otherwise. It has exactly same functioning as ata_intrq1, but this bit affects ipbus_int, while the other affects interrupt to the DMA. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int will be asserted, signalling the CPU the drive is requesting attention. The interrupt clear register has no influence on this bit.
2–0	Reserved

A group of three registers control the interrupt interface from the ATA module and going to the CPU and DMA. There are two interrupts controlled by these registers:

- ipbus_int. This interrupt is controlled by bits 3,4, 5 and 6 of the interrupt registers. It will be asserted if one of the 4 bits is set in the INT_PENDING register, while the same bit is set in the INT_ENABLE register. This interrupt goes to the CPU.
- fifo_txfer_end_alarm. This interrupt is controlled by bit 7 of the interrupt registers. If ata_intrq1 is set in both the interrupt enable and interrupt pending register, fifo_txfer_end_alarm will be asserted. The goal of this interrupt is to inform the DMA that the running data transfer has ended. This interrupt goes to the smart DMA.

These three registers have mostly the same bits. If a bit is set in the interrupt pending register, its interrupt is pending, and will produce an interrupt if the same bit is set in the interrupt enable register. Some bits in the interrupt pending register are sticky bits. Writing a '1' to the corresponding bit in the interrupt clear bit, will reset them.

22.6.3.6 FIFO Alarm Register

See Figure 22-16 for an illustration of valid bits in the FIFO_ALARM Register and Table 22-16 for descriptions of the bit fields.

0x8000_1034 (FIFO_ALARM) Access: User Read/Write

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

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

Figure 22-16. ATA FIFO_ALARM Register

Table 22-16. ATA FIFO_ALARM Register Field Descriptions

Field	Description
31–8	Reserved

7–0
FIFO_ALARM
	This register contains the threshold to generate fifo_rcv_alarm and fifo_tx_alarm to the DMA interface.
	If (fifo_tx_en == 1 && fifo_fill < fifo_alarm): fifo_tx_alarm is set 1, request is made to DMA to refill fifo.
	If (fifo_rcv_en == 1 && fifo_fill >= fifo_alarm): fifo_rcv_alarm is set 1, request is made to DMA to empty fifo.

22.6.3.7 Drive Registers Mapped to Host Module

Table 22-17. Drive Registers connected to ATA Bus

Address	Name	Description	Access
0x8000_A0 (DDTR)	drive_data	Drive Data Register	R/W
0x8000_A4 (DFTR)	drive_features	Drive Features Register	R/W
0x8000_A8 (DSCR)	drive_sector_count	Drive Sector Count Register	R/W
0x8000_AC (DSNR)	drive_sector_num	Drive Sector Number Register	R/W
0x8000_B0 (DCLR)	drive_cyl_low	Drive Cylinder Low Register	R/W
0x8000_B4 (DCHR)	drive_cyl_high	Drive Cylinder High Register	R/W
0x8000_B8 (DDHR)	drive_dev_head	Drive Device Head Register	R/W
0x8000_BC (DCDR)	drive_command	Drive Command Register When Write	W
0x8000_BC (DCDR)	drive_status	Drive Status Register When Read	R
0x8000_D8 (DCTR)	drive_alt_status	Drive Alternate Status Register When Read	R
0x8000_D8 (DCTR)	driver_control	Drive Counter Register When Write	W

Device registers are addressable and all registers except DDTR are 8-bit size accessible, The register DDTR is 16-bit size accessible, but all these registers are not present in the ATA interface module. A list is given in Table 22-17. If a read or write access is made to one of these registers, the read or write is

mapped to a PIO read or write cycle on the ATA bus, and the corresponding register in the device attached to the ATA bus is accessed.

22.7 Functional Description

To best describe the organization of the ATA Host controller module from a user's point of view, it is instructive to view the module at a number of different levels of hierarchy. See Figure 22-17 for an illustration of the organization of ATA and connection to a HDD device.

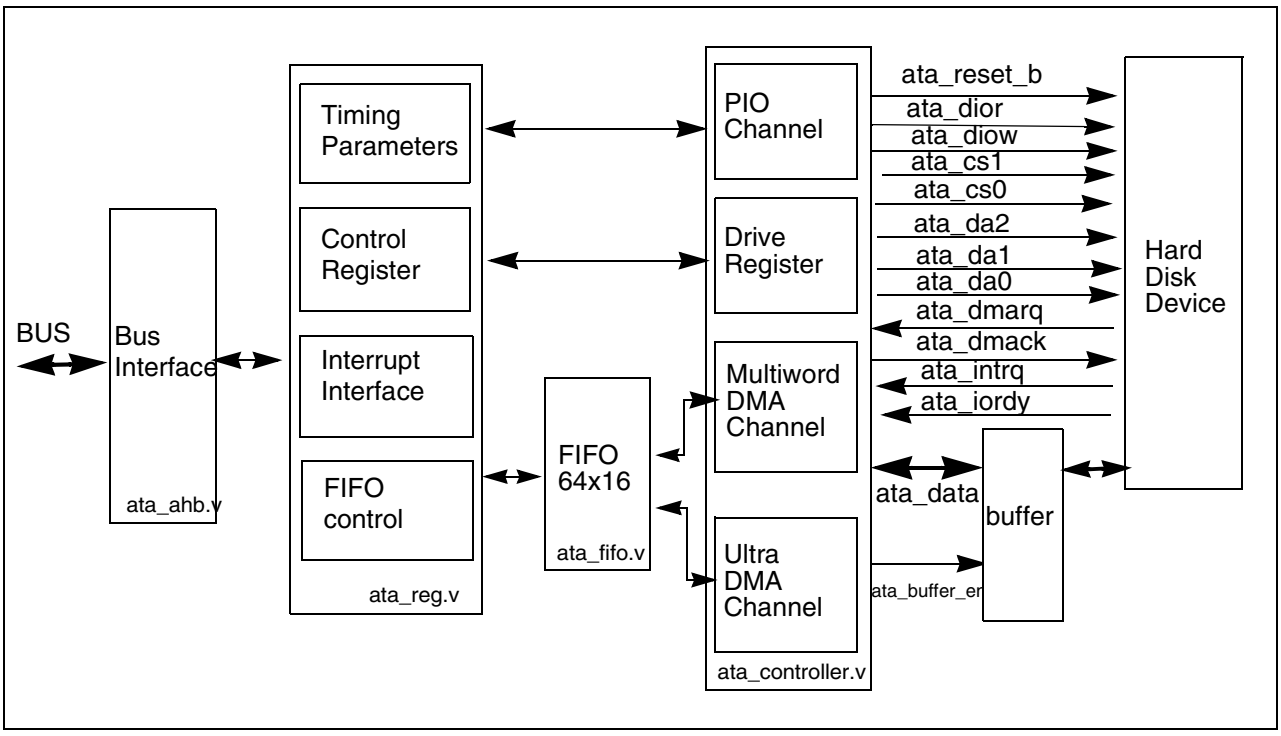


Figure 22-17. Block Diagram for ATA Module

The ATA host controller module is complied with the ATA/ATAPI-6 specification. The module consists of six main parts:

- AHB Bus Interface
- Register Block
- 64x16 FIFO for data buffer
- PIO/MDMA/UDMA protocol engine
- CRC block
- Input signal synchronizer

22.8 Initialization/Application Information

The ATA interface provides two ways to communicate with the ATA peripherals connected to the ATA bus

- PIO mode read/write operation to the ATA bus.

- DMA transfers with the ATA bus

The operation of the peripheral is described in detail in the following sections.

22.8.1 Resetting ATA Bus

The ATA bus reset ata_reset_b is asserted whenever bit 6 ata_rst_b of register ata_control is cleared to 0. At the same time, the ATA protocol engine is reset. When this bit is set to 1, the reset is released.

22.8.2 Access to ATA Bus in PIO Mode

Access to the ATA bus in PIO mode is possible after:

- ata_rst_b bit in register ata_control is set.
- Timing parameters have been programmed.

To access the drive in PIO mode, simply read or write to the correct drive register. The bus cycle will be translated to an ATA cycle, and the drive is accessed.

When drive registers are accessed while the ATA bus is in reset, the read or write is discarded, not done.

22.8.3 Using DMA Mode to Receive Data from ATA bus

Apart from PIO mode, the ATA interface supports also MDMA and UDMA mode to transfer data. DMA mode can be used to receive data from the drive (DMA in transfer). In DMA receive mode, the protocol engine will transfer data from the drive to the FIFO using multiword DMA or ultra DMA protocol. The transfer will pause when one of following occurs:

- The FIFO is full.
- The drive deasserts its DMA request signal ata_dmarq.
- The bit dma_pending in the ata_control register is cleared.

When the cause of the transfer pausing is removed, the transfer restarts. The end of the transfer is signalled by the drive to the host by asserting the ata_intrq signal. Alternatively, the host can read the device status register. In this register, the drive will also indicate if the transfer has ended.

The transfer of data from the FIFO into the memory is handled by the host system DMA. Whenever the FIFO filling is above the alarm threshold, the DMA should read one packet of data from the FIFO, and store this in main memory. In doing so, the DMA prevents the FIFO from getting full, and keeps the transfer from drive to FIFO running.

The steps for setting up a DMA data transfer from device to host are:

1. Make sure the ATA bus is not in reset and all timing registers are programmed.
2. Make sure the FIFO is empty by reading it until empty or by resetting it.
3. Initialize the DMA channel connected to fifo_rcv_alarm. Every time fifo_rcv_alarm is high, the DMA should read <packetsize> long integers from the FIFO, and store them to main memory. (typical packetsize is 8 longs)

4. Write 2 * <packetsize> to fifo_alarm register. In this way, FIFO will request attention to DMA when there is at least one packet ready for transfer.
5. To make the ATA ready for a DMA transfer from device to host, take the following steps:
   a) Make sure the FIFO is out of reset by setting bit fifo_rst_b to 1 in the ata control register.
   b) Program fifo_rcv_en=1 in ata_ control register. This enables the FIFO to be emptied by the DMA.
   c) Program dma_pending =1, dma_write=0, ultra_mode_selected=0/1 in ata_control register. ultra_mode_selected should be 1 if you want to transfer data using UDMA mode, it should be 0 if you want to transfer data using MDMA mode.
6. Now, the host side of the DMA is ready. Send commands to the drive in PIO mode that cause it to request DMA transfer on the ATA bus. The nature of these commands is beyond the scope of this document. You should consult the ATA specification to know how to communicate with the drive.
7. When the drive now requests DMA transfer by pulling ata_dmarq high, the ATA interface will acknowledge with ata_dmack, and the transfer will start. Data is transferred automatically to the FIFO, and from there on to the host memory.
8. During the transfer, the host can monitor for end of transfer by reading some device ATA registers. These reads will cause the running DMA to pause; after the read is completed, the DMA resumes. The host can also wait unit the drive asserts ata_intrq. This also indicates end of transfer.
9. On end of transfer, the host or host DMA should wait until controller_idle is set, and next read the remaining half words from the FIFO, and transfer these to memory.

NOTE

There may be less than <packetsize> remaining bytes, so transfer will not be automatic by the DMA.

22.8.4 Using DMA Mode to Transmit Data to ATA bus

Apart from PIO mode, the ATA interface supports also MDMA and UDMA mode to transfer data. DMA mode can be used to transmit data to the drive (DMA out transfer). In DMA transmit mode, the protocol engine will transfer data from the FIFO to the drive using multi word DMA or ultra DMA protocol. The transfer will pause when one of following occurs:

- The FIFO is empty.
- The drive deasserts its DMA request signal ata_dmarq.
- The bit dma_pending in the ata_control register is cleared.

When the cause of the transfer pausing is removed, the transfer restarts. The end of the transfer is signalled by the drive to the host by asserting the ata_intrq signal. Alternatively, the host can read the device status register. In this register, the drive will also indicate if the transfer has ended.

The transfer of data from the memory to the FIFO is handled by the host system DMA. Whenever the FIFO filling is below the alarm threshold, the DMA should read one packet of data from the main memory, and store this in the FIFO. In doing so, the DMA prevents the FIFO from getting empty, and keeps the transfer from FIFO to drive running.

The steps for setting up a DMA data transfer from device to host are:

1. Make sure the ATA bus is not in reset and all timing registers are programmed.
2. Make sure the FIFO is empty by reading it until empty, or by resetting it.
3. Initialize the DMA channel connected to fifo_tx_alarm. Every time fifo_tx_alarm is high, the DMA should read <packetsize> long integers from the main memory, and write them to the FIFO. (typical packetsize is 8 longs). Program the DMA such that it will not transfer more than <sectorsize> long words in total.
4. Write FIFO_SIZE - 2 * <packetsize> to fifo_alarm register. In this way, FIFO will request attention to DMA when there is room for at least one extra packet. FIFO_SIZE should be given in half words. (typical 64 half words)
5. To make the ATA ready for a DMA transfer from host to device, perform the following steps:
   a) Make sure the FIFO is out of reset by setting bit fifo_rst_b to 1 in the ata control register.
   b) Program fifo_tx_en=1 in ata_control register. This enables the FIFO to be filled by DMA.
   c) Program dma_pending =1, dma_write=1, ultra_mode_selected=0/1 in ata_control register. ultra_mode_selected should be 1 if you want to transfer data using UDMA mode, it should be 0 if you want to transfer data using MDMA mode.
6. Now, the host side of the DMA is ready. Send commands to the drive in PIO mode that cause it to request DMA transfer on the ATA bus. The nature of these commands is beyond the scope of this document. You should consult the ATA specification to know how to communicate with the drive.
7. When the drive now requests DMA transfer by pulling ata_dmarq high, the ATA interface will acknowledge with ata_dmack, and the transfer will start. Data is transferred automatically from the FIFO, and also from host memory to FIFO.
8. During the transfer, the host can monitor for end of transfer by reading some device ATA registers. These reads will cause the running DMA to pause; after the read is completed, the DMA resumes. The host can also wait unit the drive asserts ata_intrq. This also indicates end of transfer.

At end of transfer, no extra FIFO manipulations are needed.

Chapter 23
Configurable Serial Peripheral Interface (CSPI)

The i.MX27 processor contains three Configurable Serial Peripheral Interface (CSPI) modules that allow rapid data communication with fewer software interrupts than conventional serial communications. Each CSPI is equipped with two data FIFOs and is a master/slave configurable serial peripheral interface module, allowing i.MX27 to interface with both external SPI master and slave devices.

This chapter describes how the CSPI module communicates with external devices. Each CSPI has one 8×32-bit data-in FIFO and one 8×32-bit data-out FIFO. Incorporating the $\overline{\text{CSPI1\_RDY}}$ and SS control signals, it enables fast data communication with fewer software interrupts. Figure 23-1 illustrates the configurable serial peripheral interface block diagram.

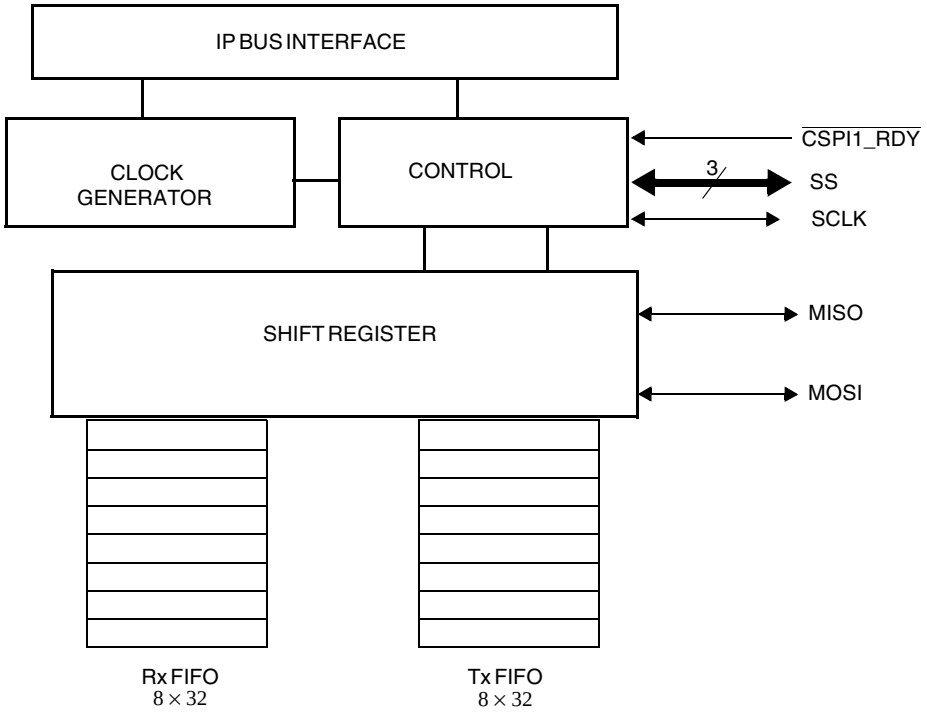


Figure 23-1. Configurable Serial Peripheral Interface Block Diagram

23.1 Features

The primary features of the CSPIs include:

- Master/Slave configurable for CSPI1 and CSPI2. CSPI3 is only a master.
- CSPI1 and CSPI2 have three chip-selects (SS0-SS3) respectively. CSPI3 has one chip select (SS0).
- Up to 32-bit programmable data transfer

- 8 × 32-bit FIFO for both Tx and Rx data
- Transfer continuation function allows unlimited length data transfers
- Polarity and phase of the Chip Select ($\overline{SS}$) and SPI Clock (SCLK) are configurable
- DMA support
- Full-duplex synchronous serial interface
- Maximum SPI clock frequency up to 22.167 MHz as a master, 16.625 MHz as a slave

23.1.1 External Signals Description

The following signals are visible to external SPI devices. They are used to control the serial peripheral interface:

- MOSI—Master Out Slave In bidirectional signal, which is TxD output signal from the data shift register in master mode. In Slave mode it is RxD input to the data shift register.
- MISO—Master In Slave Out bidirectional signal, which is RxD input signal to the data shift register in master mode. In Slave mode it is TxD output from the data shift register.
- SCLK—CSPI Clock bidirectional signal, which is CSPI clock output in master mode. In slave mode it is an input CSPI clock signal.
- SS[2:0], Slave Select bidirectional signal, output in master mode, and input in slave mode.
- $\overline{CSPI1\_RDY}$—This input signal is used for hardware control only in master mode. It indicates that external SPI slave is ready to receive data. It will edge or level trigger a CSPI burst if used. This signal is only available for CSPI1: it is not present on CSPI2 and CSPI3. It is ignored in slave mode. This signal is controlled by DRCTL(ControlReg[13:12]) bits. If the hardware control enabled, CSPI will transfer data only when external SPI slave is ready.

23.2 Module Input/Output Signals

Table 23-1. Signal Listing

Signal	IN/OUT	BITS	DESCRIPTION
Pad Level Signals			
IPP_DO_MOSI	OUT	1	Tx data when CSPI is in Master mode.
IPP_DO_MISO	OUT	1	Tx data when CSPI is in Slave mode.It is tri-stated if the selected SSn is not asserted.
IPP_CSPI_CLK_OUT	OUT	1	Clock output when CSPI is in Master mode.Max. frequency is IPG_CLK_PERCLK/3.
IPP_DO_SS0	OUT	1	Slave select0 generated by CSPI in Master mode when CS[1:0] bits are set to ‘00’ in the CONTROL Register.
IPP_DO_SS1	OUT	1	Slave select1 generated by CSPI in Master mode when CS[1:0] bits are set to ‘01’ in the CONTROL Register.

Table 23-1. Signal Listing (continued)

Signal	IN/OUT	BITS	DESCRIPTION
IPP_DO_SS2	OUT	1	Slave select2 generated by CSPI in Master mode when CS[1:0] bits are set to '10' in the CONTROL Register.
IPP_DO_SS3	OUT	1	Slave select3 generated by CSPI in Master mode when CS[1:0] bits are set to '11' in the CONTROL Register.
IPP_OBE_MOSI	OUT	1	Output enable for IPP_DO_MOSI, IPP_CSPI_CLK_OUT, IPP_DO_SS0, IPP_DO_SS1 and IPP_DO_SS2,IPP_DO_SS3 when the CSPI is in master mode.
IPP_OBE_MISO	OUT	1	Output enable for ipp_do_miso.
IPP_IND_MIS0	IN	1	Rx data when CSPI is in Master mode.
IPP_IND_MOSI	IN	1	Rx data when CSPI is in Slave mode.
IPP_CSPI_CLK_IN	IN	1	Clock input when CSPI is in Slave mode.
IPP_IND_SS0	IN	1	Slave Select0 signal from an external master when CSPI is in Slave mode.
IPP_IND_SS1	IN	1	Slave Select1 signal from an external master when CSPI is in Slave mode.
IPP_IND_SS2	IN	1	Slave Select2 signal from an external master when CSPI is in Slave mode.
IPP_IND_SS3	IN	1	Slave Select3 signal from an external master when CSPI is in Slave mode.
IPP_IND_DATAREADY	IN	1	This input signal is used only in master mode. It will edge or level trigger a CSPI burst if used.
IPP_IND_FORCE_MASTER	IN	1	Used to force CSPI Master mode from top level.
IP Bus Interface Signals			
IPS_BYTE_31_24	IN	1	Module byte access enable. Enables write to bits [31:24] of addressed register.
IPS_BYTE_23_16	IN	1	Module byte access enable. Enables write to bits [23:16] of addressed register.
IPS_BYTE_15_8	IN	1	Module byte access enable. Enables write to bits [15:8] of addressed register.
IPS_BYTE_7_0	IN	1	Module byte access enable. Enables write to bits [7:0] of addressed register.
IPS_MODULE_EN	IN	1	Peripheral module enable
IPS_ADDR	IN	11	Address bus
IPS_RWB	IN	1	Read access signal. Active low.
IPS_WDATA	IN	32	Write Data bus
IPS_RDATA	OUT	32	Read Data bus

Table 23-1. Signal Listing (continued)

Signal	IN/OUT	BITS	DESCRIPTION
IPS_XFR_WAIT	OUT	1	Wait signal to ARM
IPS_XFR_ERR	OUT	1	Transfer error acknowledge. ips_xfr_err is generated when a write to a read-only register(RXDATA register) is performed. It is not generated when the write-only register (TXDATA register) is read.
INTERRUPTS			
IPI_INT_CSPI	OUT	1	CSPI interrupt request line to the core.
DMA INTERFACE SIGNALS			
IPD_REQ_CSPI_TDMA	OUT	1	DMA Tx request
IPD_REQ_CSPI_RDMA	OUT	1	DMA Rx request
GLOBAL SIGNALS			
IPG_HARD_NEG_ASYNC_RESET	IN	1	Asynchronous, Active Low Hardware Reset for FFs clocked by inverted clocks.
IPG_HARD_POS_ASYNC_RESET	IN	1	Asynchronous, Active Low Hardware Reset
IPG_CLK_S	IN	1	Clock gated by the OR function of all the ips_module_en signals from AIPI.
IPG_CLK	IN	1	Continuous Clock gated by the ipg_clk_en.
IPG_CLK	IN	1	Inverted ipg_clk clock.
IPG_CLK_EN	OUT	1	Used to gate off the ipg_clk depending on the module enable bit residing in the CSPI that is, the SPIEN bit.
IPG_CLK_PERCLK	IN	1	Reference baud rate clock. It must be slower than or equal to IPG_CLK.
IPG_CLK_32K	IN	1	32khz Continuous clock used for counting purpose.
RESP_SEL	IN	1	When this pin is ‘1’, the module will always return an OKAY response (that is, ips_xfr_err is not asserted) when there is an access to any location within the 4kbyte boundary, whereas, if this pin is’0’, the module will return an ERROR response (ips_xfr_err is asserted) when there is an access to an unused location within the 4kbyte boundary.
TEST MODE SIGNALS			
IPT_TEST_MODE	IN	1	This is used to activate all scan testable logic into scan mode and non-scan logic into bypass mode.
IPT_TEST_ASYNC_SE	IN	1	Used to handle internally generated resets.
IPT_TEST_CLK_SE	IN	1	Used to switch between 32Khz clock and the ipg_clk.
IPT_TEST_RESET_B	IN	1	Test mode reset.

23.3 Operation

When CSPI is configured as master, the SS (output) and $\overline{\text{CSPI1\_RDY}}$ (input) signals, are used for data transfer rate control. The sample period control register can be set if a fixed data transfer rate is required.

When CSPI is configured as slave, the SS signal becomes an input signal and can optionally be used for data latching and loading to the internal data shift registers, as well as incrementing internal data FIFO pointers. Figure 23-2 shows the generic CSPI timing.

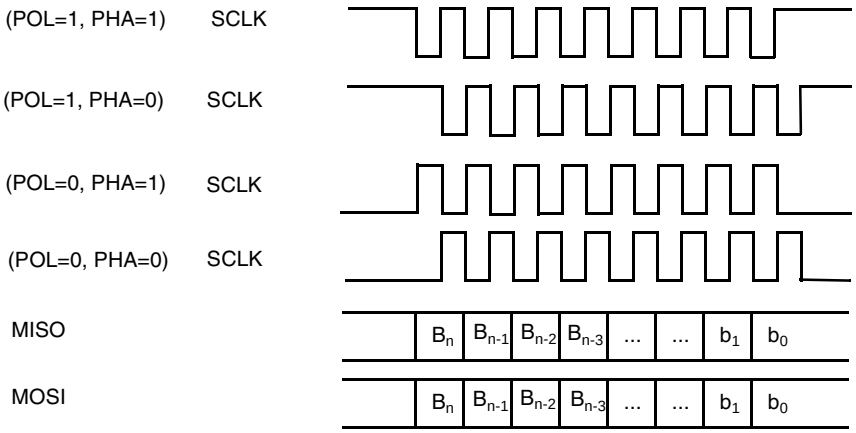


Figure 23-2. Generic CSPI Timing

23.3.1 Phase and Polarity Configurations

The serial peripheral interface master uses the SCLK signal to transfer data in and out of the shift register. Data is clocked using any one of four programmable clock phase and polarity combinations. During Phase 0, Polarity 0 and Phase 1, Polarity 1 operations, output data changes on the falling clock edge and input data is shifted in on the rising edge. The most-significant bit is output when the CPU loads the transmit data. During Phase 1, Polarity 0 and Phase 0, Polarity 1 operations, output data changes on the rising edges of the clock and is shifted in on falling edges. The most-significant bit is output on the first rising SCLK edge. Polarity inverts SCLK, but does not change the edge-triggered events that are internal to the serial peripheral interface master. This flexibility allows it to operate with most serial peripheral devices. Figure 23-2 shows the CSPI timing with various POL and PHA configurations.

23.3.2 Master Mode

The CSPI master uses the $\overline{\text{SS}}$ signal to enable an external SPI device and uses SCLK to transfer data in and out of the Shift register. The $\overline{\text{SPI\_RDY}}$ enables fast data communication with fewer software interrupts. By using PeriodReg, the CSPI can be used for a fixed data transfer rate.

When CSPI is in Master mode the $\overline{\text{SS}}$, SCLK, and MOSI are output signals and the MISO is an input. Figure 23-3 shows a typical SPI transfer.

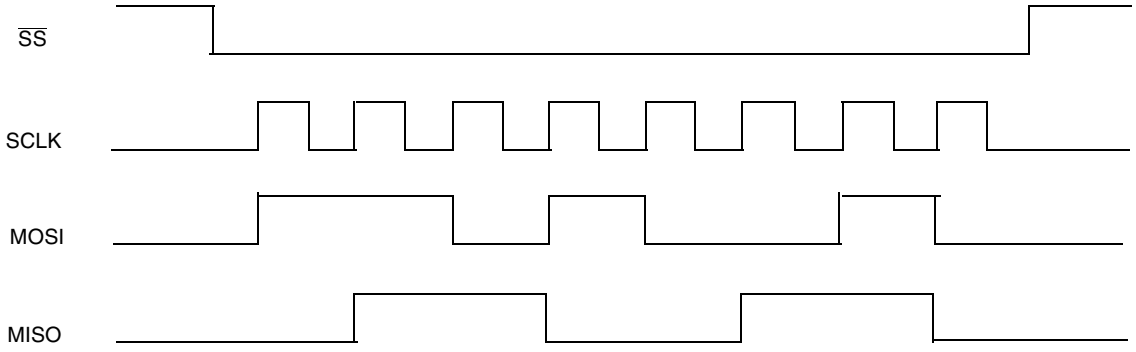


Figure 23-3. Typical SPI Transfer (8-Bit)

In Figure 23-3, the SS signal enables the selected external SPI device and the SCLK synchronizes data transfer. MOSI and MISO change on rising edge of SCLK and the MISO is latched on the falling edge of the SLCK clock. The data shifted out is 0xD2, and the data shifted in is 0x66.

23.3.2.1 Master Mode with SPI_RDY

By default, the CSPI does not use SPI_RDY in master mode. a SPI transfer begins when the following events happen: the CSPI is enabled, TXFIFO has data in it, and ControlReg[XCH] is set. When ControlReg[DRCTL] contains either 01 or 10, the SPI_RDY controls when a SPI burst starts.

If ControlReg[DRCTL] is set to 01, the SPI burst can be triggered only if a falling edge of SPI_RDY has been detected. Figure 23-4 shows the relationship between a SPI transfer and the falling edge of SPI_RDY.

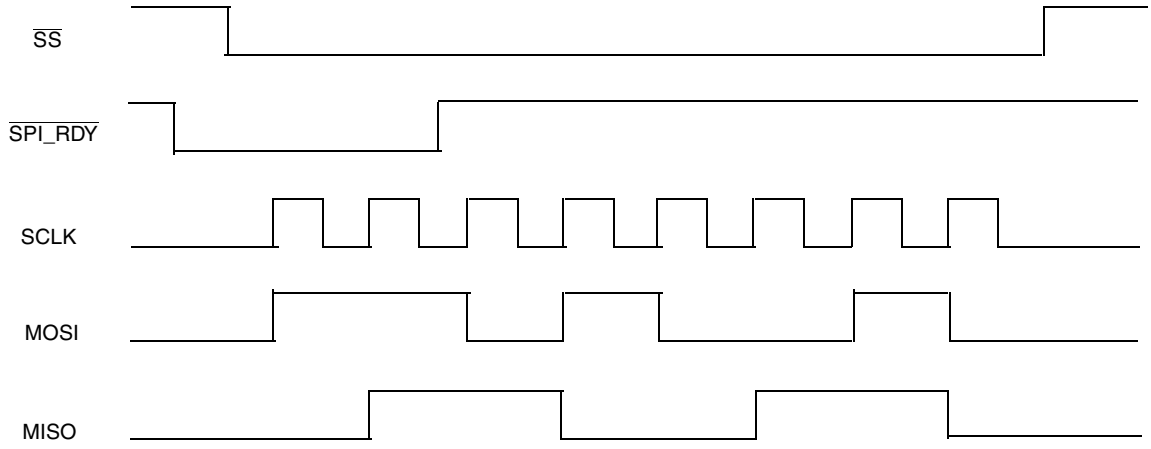


Figure 23-4. Relationship between a SPI transfer and the Falling Edge of SPI_RDY

a SPI transfer does not start until the falling edge of SPI_RDY is detected. The next SPI burst starts when the next SPI_RDY falling edge is detected, after the last transfer has finished.

If ControlReg[DRCTL] is set to 10, the SPI burst can be triggered only if $\overline{\text{SPI\_RDY}}$ is low. Figure 23-5 shows the relationship between a SPI transfer and $\overline{\text{SPI\_RDY}}$. The SPI transfer does not begin until $\overline{\text{SPI\_RDY}}$ goes low. The next SPI transfer begins after the previous transfer has finished if $\overline{\text{SPI\_RDY}}$ remains low.

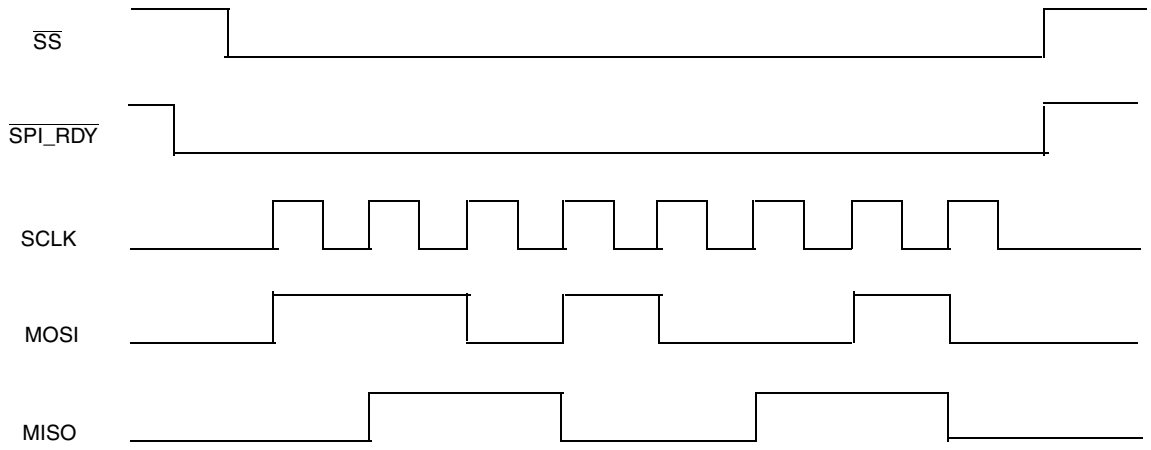


Figure 23-5. Relationship between a SPI Transfer and $\overline{\text{SPI\_RDY}}$

23.3.2.2 Master Mode with Wait States

Wait states can be inserted between SPI transfers. This provides a way for the user to slow down the SPI transfer to meet the timing requirements of a slower SPI device. Figure 23-6 shows wait states inserted between SPI bursts.

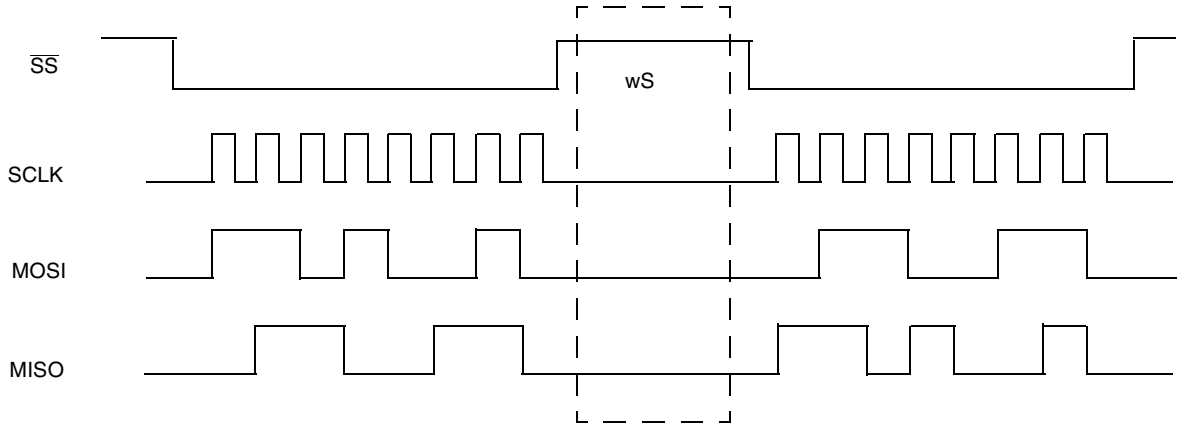


Figure 23-6. SPI Transfers with Wait States

In this case, the number of wait states is controlled by PeriodReg [WAIT] and the wait states' clock source is selected by PeriodReg [CSRC].

23.3.2.3 Master Mode with Continuation

Many transfers can be continuous without idle time insertion. This provides a way for the user to maximize data rate without any delay. Figure 23-7 shows the detail timing.

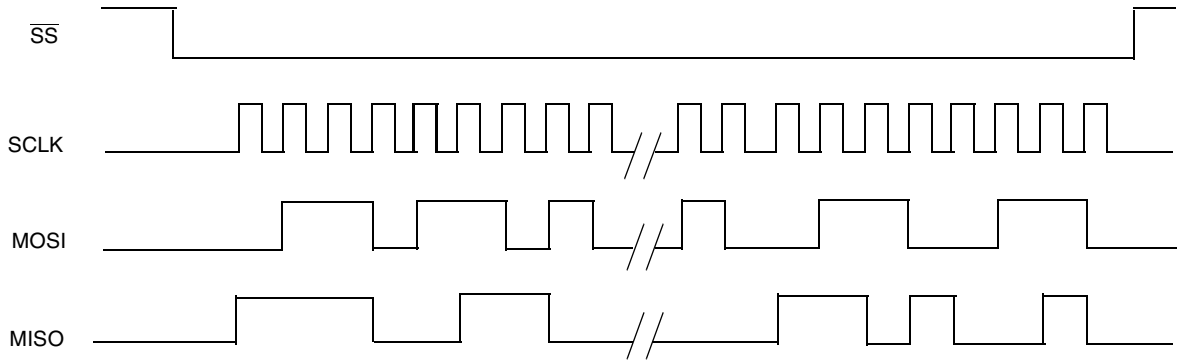


Figure 23-7. SPI continuous transfer with BURST=1

To obtain the Continuation function, the ControlReg [BURST] should be set to 1, with ControlReg [SSCTL] and PeriodReg [WAIT] set to 0.

23.3.2.4 Master Mode with SSCTL Control

SSCTL controls whether a $\overline{SS}$ pulse is inserted between two data transfers. When SSCTL is set, a $\overline{SS}$ pulse will be inserted in multiple transfers. When SSCTL is cleared, $\overline{SS}$ signal will stay asserted in multiple transfers. Figure 23-8 and Figure 23-9 show the detail timing.

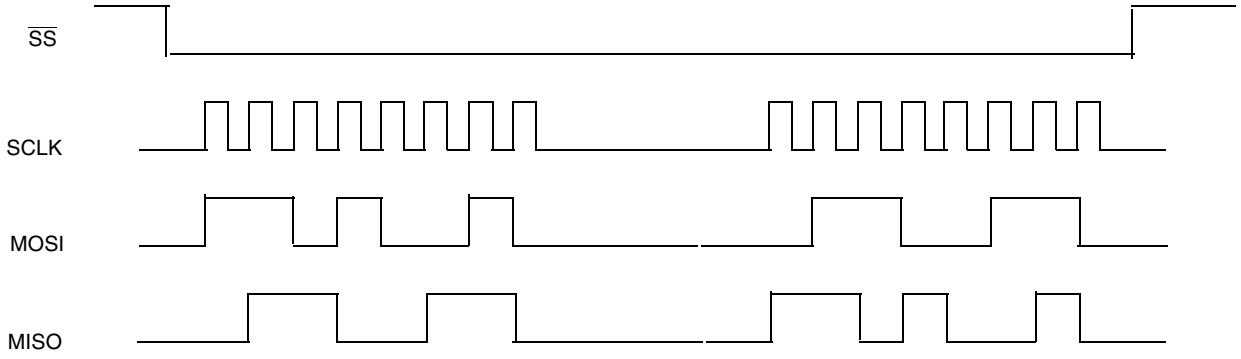


Figure 23-8. SPI Transfer while SSCTL is Clear

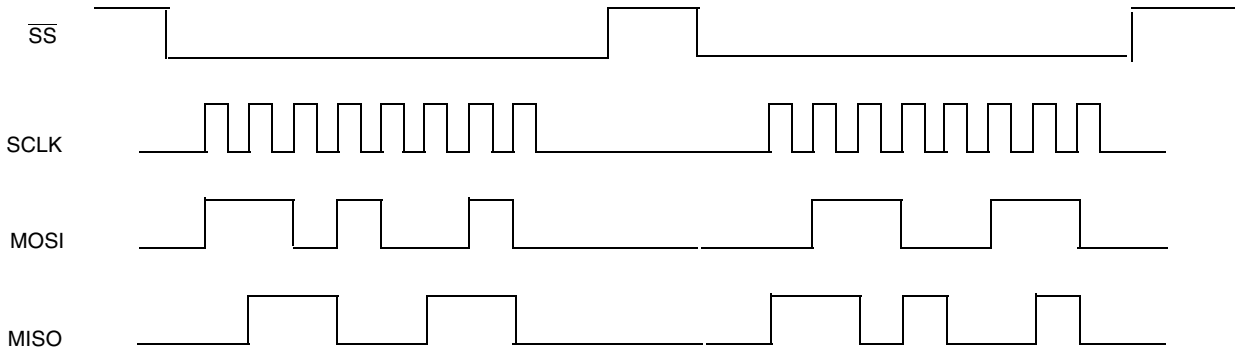


Figure 23-9. SPI Transfers while SSCTL is Set

23.3.2.5 Master Mode with various configurations of WAIT, BURST and SSCTL

Table 23-2 shows CSPI behavior in master mode in various configurations of PeriodReg[WAIT], ControlReg [BURST] and ControlReg [SSCTL]. In this table, x means don't care (0 or 1).

Table 23-2. CSPI Behavior In Master Mode In Various Configurations

WAIT	BURST	SSCTL	SCLK Pin	$\overline{SS}$ Pin
0	0	0	7*Tsclk[1] idle time inserted between consecutive data transfers	No pulse
0	1	0	No idle time inserted between consecutive data transfers	No pulse
0	x	1	7*Tsclk idle time inserted between consecutive data transfers	2*Tsclk width pulse
Non-zero	x	0	(7*Tsclk + WAIT*Tsclk_32k[2]) idle time inserted between consecutive data transfers	No pulse
Non-zero	x	1	(7*Tsclk + WAIT*Tsclk_32k) idle time inserted between consecutive data transfers	(2*Tsclk + WAIT*Tsclk_32k) width pulse

1 Tsclk is CSPI SCLK period

2 Tsclk_32k is either SCLK or 32KHz clock period

23.3.3 Slave Mode

When the CSPI module is configured as a slave, the user can configure the CSPI Control register to match the external SPI master's timing. In this configuration, $\overline{SS}$ becomes an input signal, and is used to latch data into or load data out to the internal data Shift registers, as well as to increment the data FIFO.

The $\overline{SS}$, SCLK, and MOSI are inputs and MISO is output. Most of their timing diagrams are the same as in Master mode, because the inputs come from a SPI master device.

However, it is different when $\overline{SS}$ is used to increment data FIFO. When the SSCTL is set while CSPI is in Slave mode, the data FIFO will increment at $\overline{SS}$ rising edge.

Figure 23-10 shows a SPI burst in which data FIFO is incremented by $\overline{SS}$ rising edge.

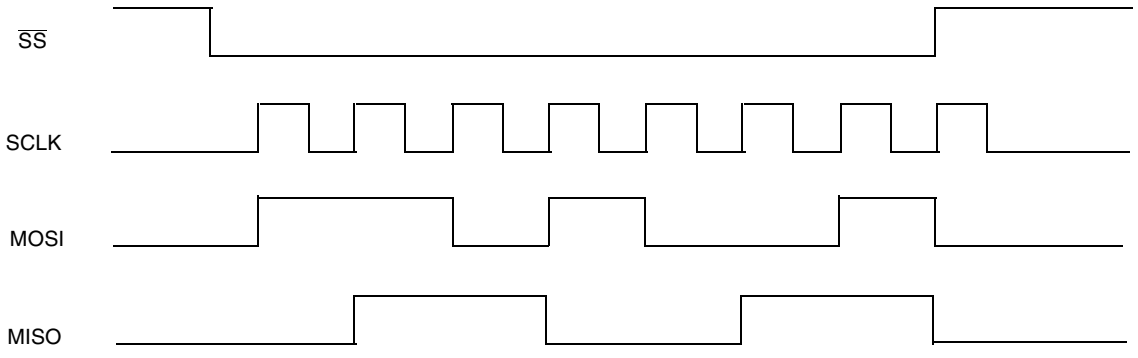


Figure 23-10. Increment Data FIFO by SS Rising Edge

In this case, the data received is not 0xD2 but 0x69. Only the most significant 7 bits are loaded to RXFIFO.

23.3.4 Interrupt Control

Interrupt control is not a specific mode of operation; however, it provides a basic method to utilize the CSPI FIFOs.

You can program the CSPI to enable the TXFIFO empty, TXFIFO half, and TXFIFO full interrupt. You can also use the interrupt service routine to fill the TXFIFO with data to be transferred. Furthermore, you can also enable RXFIFO ready, RXFIFO half, and RXFIFO full to retrieve data from RXFIFO by using the interrupt service routine.

Three other interrupt sources can be used to control/debug the SPI transfer. The TxFIFO and Tx Shift Register Empty interrupt tells the user that there is not data left in TXFIFO and the data in the Shift register is shifted out. The bit counter overflow interrupt tells the user that the CSPI received more than 32 bits in a SPI burst and the remaining bits will be lost. The RXFIFO overflow interrupt tells the user that the RXFIFO received more than 8 words and will not accept any other word. Figure 23-11 shows a program sequence of SPI bursts using interrupt.

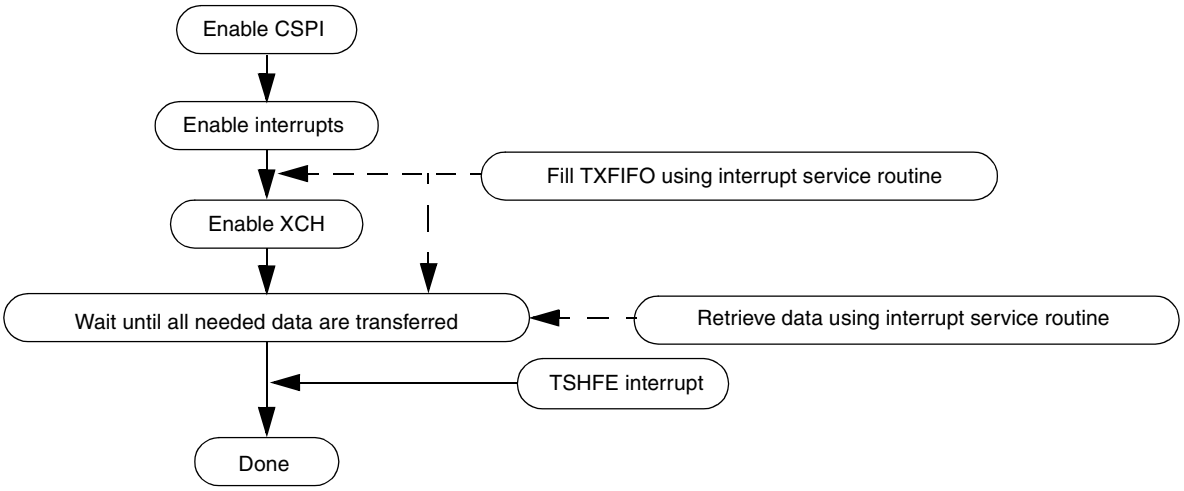


Figure 23-11. Program Sequence of SPI Burst Using Interrupt

23.3.5 DMA Control

DMA control provides another way to utilize the FIFOs in the CSPI module. Peripherals such as the CSPI which support DMA, use DMA request and acknowledge signals. Larger amounts of data can be transferred using DMA control, thereby reducing interrupts and CPU loading. When the appropriate conditions are matched, the module will send out a DMA request, and the DMA will deal with the following cases: TXFIFO empty, TXFIFO half, RXFIFO half, and RXFIFO full.

Figure 23-12 shows a program sequence of SPI bursts using the DMA.

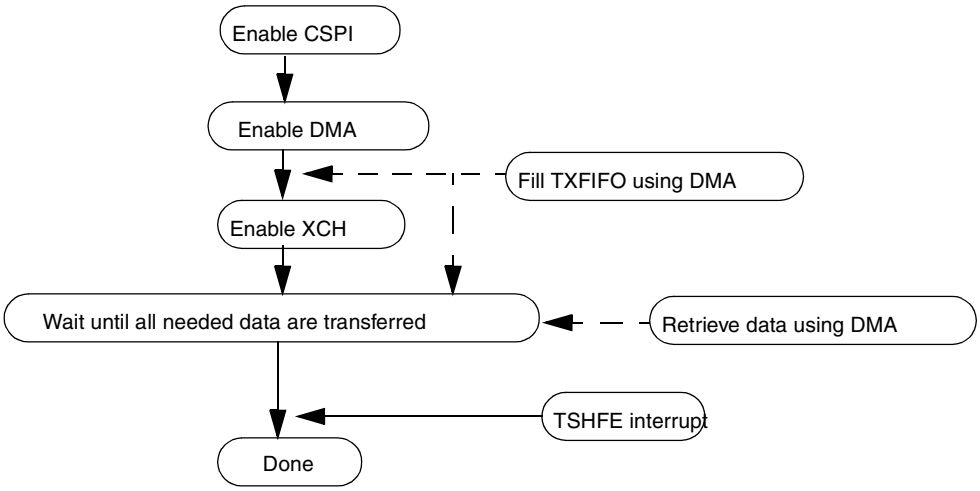


Figure 23-12. Program Sequence of SPI Burst Using DMA

23.4 Initialization/Application Information

This section provides initialization and application information for CSPI.Figure 23-13 shows two flow charts for the master and slave mode of operations supported by the CSPI.

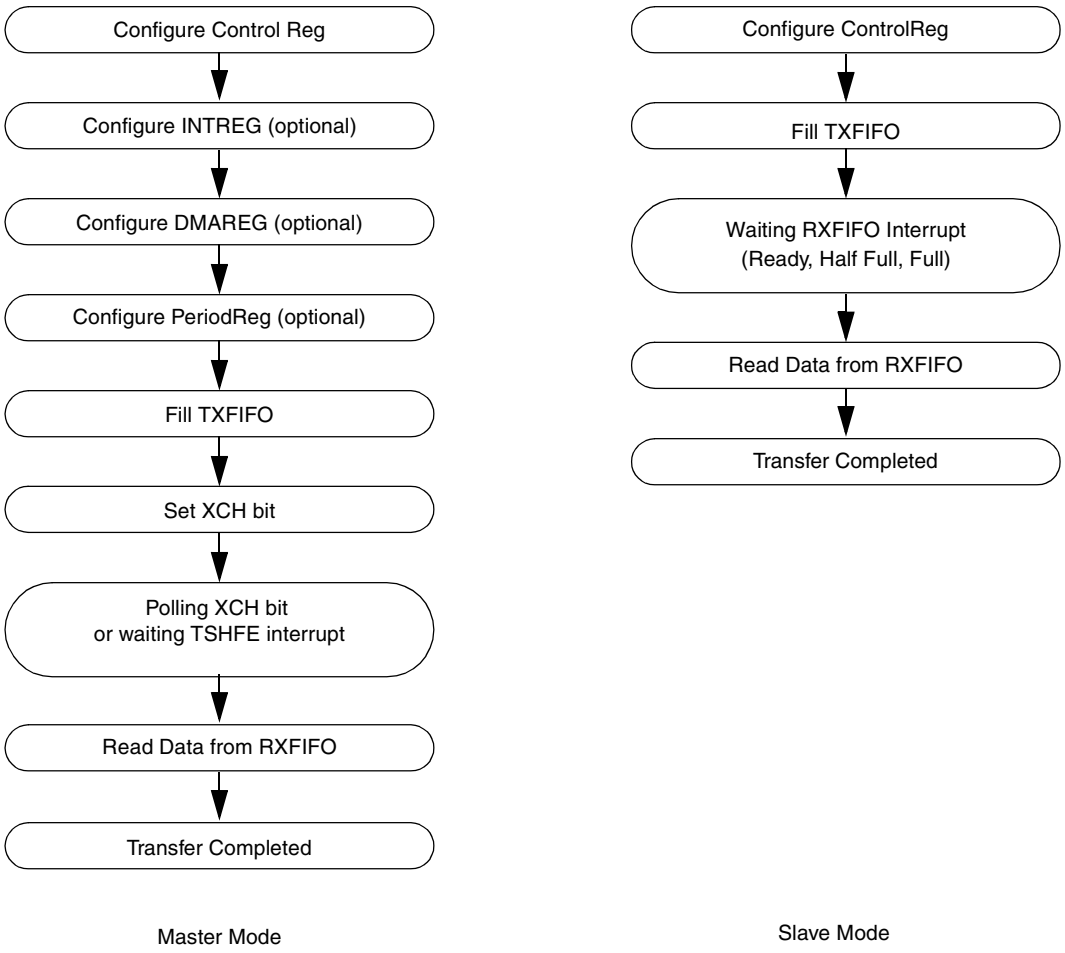


Figure 23-13. Flow Chart of CSPI Operation

Example 23-1 shows a normal example code of CSPI operation using ARM instructions.

Example 23-1. CSPI Operation using ARM Instructions

```
LDR R0, =CSPI_BASE_ADDRESS          ; Load CSPI Base Address to R0
LDR R1, =0x00008C1F                 ; Master Mode, 32-bit transaction
STR R1, [R0, #0x08]
LDR R1, =0x00000021                 ; Enable RXFIFO half and TXFIFO empty
STR R1, [R0, #0x0C]                 ; interrupt (Alternatively with DMA Mode)
LDR R1, =0x00005000                 ; Enable RXFIFO half and TXFIFO empty
```

```
        STR R1, [R0, #0x18]             ; DMA (Alternatively with interrupt)
        LDR R5, =0x05                   ; R5 as number of words to be transferred.
        LDR R1, =0x11111111             ; R1 as increment to generate the data.
        LDR R2, =0x12345678             ; R2 load the data to be transferred.
Loop_00
        STR R2, [R0,#0x04]              ; Store data into TXFIFO.
        ADD R2, R2, R1                  ; Generating next data to be transferred.
        SUB R5, R5, #1                  ; Decrease the R5.
        CMP R5, #0x00                   ; Check R5 if it is zero.
        BNE Loop_00                     ; Loop until R5 is zero.

        LDR R1, =0x00008E1F             ; set XCH bit to start transaction.
        STR R1, [R0, #0x08]
Loop_01
        LDR R1, [R0, #0x0C]             ; check TSHFE bit if it is set.
        LDR R2, =0x00000008
        AND R1, R2, R1
        CMP R1, #0x00
        BNE PASS_00                     ; if TSHFE bit is set then finish.
        B Loop_01                       ; if it isn't set then continue loop

        LDR R1, [R0, #0x00]             ; Read data from RXFIFO.
```

23.4.1 Software Restrictions

The section should include software restrictions that impact the customer.

- All reserved bits cannot be written and always read as 0.
- Writes to the TXDATA register are ignored when the CSPI module is disabled (SPIEN bit of CSPI ControlReg is cleared).
- The SPI Module Enable Control bit must be asserted before writing to other registers or initiating an exchange.

23.5 Memory Map and Register Definition

The CSPI includes eight 32-bit registers. Section 23.5.3, "Register Descriptions" provides the detailed descriptions for the CSPI registers. The following sections provide the register summary and the

programming model for the 3 CSPI modules in the i.MX27. The Register Summary in Table 23-3 lists all registers of the CSPI module by ascending address. The absolute address of each register is given, as is the value of each bit for reads and writes.

23.5.1 Memory Map

Table 23-1 shows the CSPI memory map.

Table 23-1. CSPI Memory Map

Address	Register	Access	Reset Value	Section/Page
0x1000_E0000 (RXDATA)	Receive Data Register (RXDATA)	R	0x0000_0000	23.5.3.1/23-16
0x1000_E0004 (TXDATA)	Transmit Data Register (TXDATA)	W	0x0000_0000	23.5.3.2/23-17
0x1000_E0008 (CONREG)	Control Register (CONREG)	R/W	0x0000_0000	23.5.3.3/23-18
0x1000_E000C (INTREG)	Interrupt Control Register (INTREG)	R/W	0x0000_0000	23.5.3.4/23-20
0x1000_E0010 (DMAREG)	DMA Control Register (DMAREG)	R/W	0x0000_0000	23.5.3.5/23-22
0x1000_E0014 (STATREG)	Status Register (STATREG)	R/W	0x0000_0003	23.5.3.6/23-23
0x1000_E0018 (PERIODREG)	Sample Period Control Register (PERIODREG)	R/W	0x0000_0000	23.5.3.7/23-24

23.5.2 Register Summary

Figure 23-14 shows the key to the register fields, and Table 23-2 shows the register figure conventions.

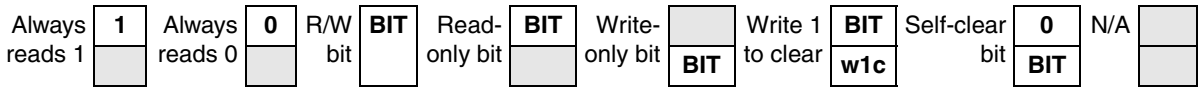


Figure 23-14. Key to Register Fields

Table 23-2. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.

Table 23-2. Register Figure Conventions (continued)

Convention	Description
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 23-3. CSPI Register Summary

Name		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
RxDataReg 0x1000 E000 0x1000 F000 0x1001 7000	R	RxData[31:16]															
	W																
	R	RxData[15:0]															
	W																
TxDataReg 0x1000 E004 0x1000 F004 0x1001 7004	R																
	W	Tx Data [31:16]															
	R																
	W	Tx Data [15:0]															
ControlReg 0x1000 E008 0x1000 F008 0x1001 7008	R	0	0	0	0	0	0	0	0	BURST	SDHC_SPIEN	SWAP	CS[1:0]		DataRate[4:2]		
	W																
	R	DataRate[1:0]		DRCTL[1:0]		MODE	SPIEN	XCH	SSPOL	SSCTL	PHA	POL	BIT COUNT[4:0]				
	W																
INTREG 0x1000 E00C 0x1000 F00C 0x1001 700C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BOEN	ROEN
	W																
	R	RFEN	RHEN	RREN	TSHFEEN	TFEN	THEN	TEEN	BO	RO	RF	RH	RR	TSHFE	TF	TH	TE
	W																
TestReg 0x1000 E010 0x1000 F010 0x1001 7010	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R		LBC	INIT	SS_ASSERT	SSTATUS[3:0]				RXCNT[3:0]			TXCNT[3:0]				
	W																

Table 23-3. CSPI Register Summary (continued)

Name		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
PeriodReg 0x1000 E014 0x1000 F014 0x1001 7014	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CSRC	WAIT[14:0]														
	W																
DMAREG 0x1000 E018 0x1000 F018 0x1001 7018	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	THDEN	TEDEN	RFDEN	RHDEN	0	0	0	0	THDMA	TEDMA	RFDMA	RHDMA	0	0	0	0
	W																
ResetReg 0x1000 E01C 0x1000 F01C 0x1001 701C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	START
	W																

23.5.3 Register Descriptions

The following section describes the detailed register descriptions for the CSPI registers.

23.5.3.1 Receive Data Register (RXDATA)

The Receive Data register (RXDATA) is a read-only register that forms the top word of the 8×32 receive FIFO. This register holds the data received from an external SPI device during a data transaction. Only word-sized read operations are allowed.

Figure 23-15 shows the RXDATA register, and Table 23-4 shows the register's field descriptions.

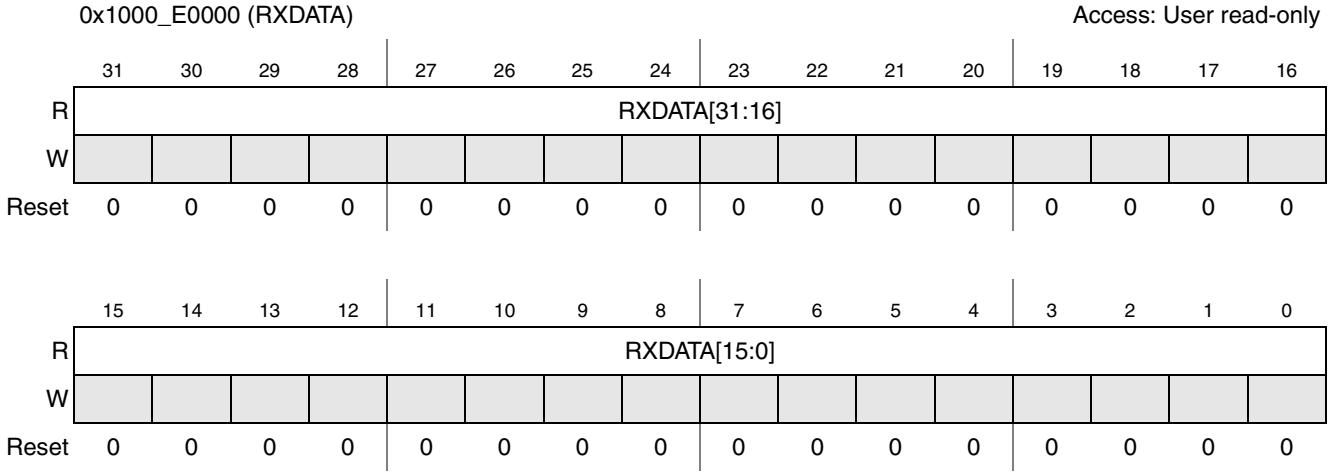


Figure 23-15. RXDATA Register Diagram

Table 23-4. RXDATA Register Field Descriptions

Field	Description
31–0 RXDATA	Receive Data. This register holds the top word of the receive data FIFO. The FIFO is advanced for each read of this register. The data read is undefined when the Receive Data Ready (RR) bit in the Interrupt Control/Status register is cleared. Zeros are read when CSPI is disabled.

23.5.3.2 Transmit Data Register (TXDATA)

The Transmit Data (TXDATA) register is a write-only data register that forms the top word of the 8×32 TXFIFO. The TXFIFO can be written to as long as it is not full, even when the XCH bit in CONREG is set. This allows the user write access to the TXFIFO during a SPI data exchange process. Writes to this register are ignored when the CSPI module is disabled (EN bit of CSPI CONREG is cleared).

Figure 23-16 shows the CNTRL register, and Table 23-5 shows the register's field descriptions.

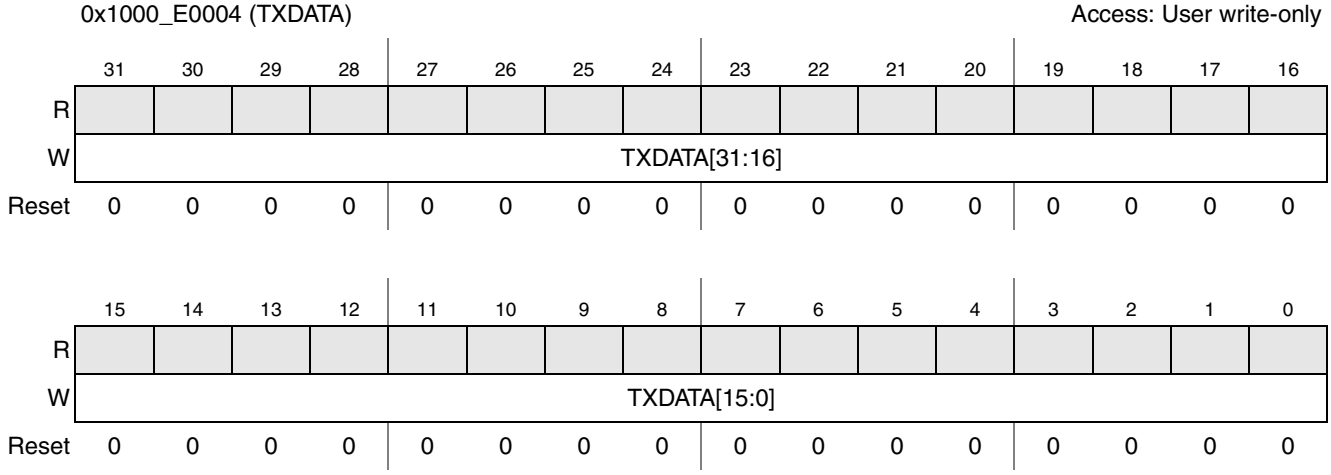


Figure 23-16. TXDATA Register Diagram

Table 23-5. TXDATA Register Field Descriptions

Field	Description
31–0 TXDATA	Transmit Data. This register holds the top word of data loaded into the FIFO. Data written to this register must be a word operation. The number of bits actually transmitted is determined by the BIT_COUNT field of the corresponding SPI Control register. If this field contains more bits than the number specified by BIT_COUNT, the extra bits are ignored. For example, to transfer 10 bits of data, a 32-bit word must be written to this register. Bits 9-0 are shifted out and bits 31-10 are ignored. When the CSPI module is operating in Slave mode, zeros are shifted out when the FIFO is empty. Zeros are read when CSPI is disabled.

23.5.3.3 Control Register (CONREG)

The Control Register (CONREG) allows the user to enable the CSPI module, configure its operating modes, specify the divider value, phase, and polarity of the clock, configure the $\overline{SS}$ and $\overline{SPI\_RDY}$ control signal, and define the transfer length. The reserved bits are always read as 0.

Figure 23-17 shows the CNTRL register, and Table 23-6 shows the register's field descriptions.

0x1000_E0008 (CONREG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	CHIP SELECT		0	0	DRCTL		0	DATA RATE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	BIT COUNT					SSPOL	SSCTL	PHA	POL	SMC	XCH	MODE	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-17. CSPI Control Register

Table 23-6. CONREG Register Field Descriptions

Field	Description
31–26	Reserved, all bits should read zero.
25–24 CHIP SELECT	CHIP SELECT. Select one of four external SPI Master/Slave Devices. In master mode, these two bits select the external slave devices by asserting the $\overline{SS}$n outputs.Only the selected $\overline{SS}$n signal will be active while the remaining 3 signals will be negated. Chip Select 00 $\overline{SS}$0 will be asserted. 01 $\overline{SS}$1 will be asserted. 10 $\overline{SS}$2 will be asserted. 11 $\overline{SS}$3 will be asserted.
23–22	Reserved, all bits should read zero.

Table 23-6. CONREG Register Field Descriptions (continued)

Field	Description
21–20 DRCTL	SPI Data Ready Control. This 2-bit field selects the utilization of the $\overline{\text{SPI\_RDY}}$ in master mode. CSPI will check this fields before it start a SPI burst. 00 Don't Care $\overline{\text{SPI\_RDY}}$ 01 Burst will be triggered by failing edge of $\overline{\text{SPI\_RDY}}$. 10 Burst will be triggered by low level of $\overline{\text{SPI\_RDY}}$. 11 $\overline{\text{RSV}}$.
19	Reserved, all bits should read zero.
18–16 DATA RATE	SPI Data Rate Control. This three-bit field selects the baud rate of the SCLK based on a division of the ipg_clk. These bits allow CSPI to synchronize with different external SPI devices. The max frequency is one quarter of ipg_clk. The divide ratio is determined according to the following table using the equation: $2^{(n+2)}$. SPI Data Rate Control (Master Mode only) 000 Divide by 4. 001 Divide by 8. 010 Divide by 16. 011 Divide by 32. 100 Divide by 64. 101 Divide by 128. 110 Divide by 256. 111 Divide by 512.
15-13	Reserved, all bits should read zero.
12–8 BIT COUNT	This field selects the length of a word to be transferred. A maximum of 32 bits can be transferred in a single SPI transfer. Multiple transfers may be chained together to form unlimited length messages using the SSCTL bit to keep the $\overline{\text{SS}}$ asserted between transfers. In master mode, BITCOUNT controls the number of bits per serial transfer. The transmit FIFO transfers a 32-bit data word to the shift register, however only the n least-significant (n=BIT COUNT + 1) are shifted out. The remaining bits are ignored. In slave mode, provided SSCTL= 0, this field controls the number of bits (BIT COUNT + 1) received in each data word. After BITCOUNT + 1 bits have been shifted into the shift register, the contents are transferred to the receive FIFO regardless of the state of the $\overline{\text{SS}}$ input. When SSCTL bit is 1, data transfer to/from the FIFO are controlled by the $\overline{\text{SS}}$ input and this field is ignored. SPI Data Rate Control (Master Mode only) 00000 Least 1 bit of a word to be transferred. 00001 Least 2 bits of a word to be transferred. ..... 01111 Least 16 bits of a word to be transferred. 10000 Least 17 bits of a word to be transferred. ..... 11110 Least 31 bits of a word to be transferred. 11111 All 32 bits of a word to be transferred.
7 SSPOL	SPI SS Polarity Select. In both Master and Slave mode, this bit selects the polarity of the $\overline{\text{SS}}$ signal. 0 Active low 1 Active high

Table 23-6. CONREG Register Field Descriptions (continued)

Field	Description
6 SSCTL	In master mode, this bit selects the output wave form for the SS signal. 0 SS remains asserted between SPI bursts. 1 Negate SS between SPI bursts. In slave mode, this bit controls the timing of data transfer from the shift register to the receive FIFO. 0 RXFIFO advanced by BIT COUNT. 1 RXFIFO advanced by SS edge. (SSPOL = 0: rising edge; SSPOL = 1: falling edge)
5 PHA	SPI Clock/Data Phase Control. This bit controls the clock/data phase relationship. 0 Phase 0 operation. 1 Phase 1 operation.
4 POL	SPI Clock Polarity Control. This bit controls the polarity of the SCLK signal. 0 Active high polarity (0 = Idle) 1 Active low polarity (1 = Idle)
3 SMC	Start Mode Control. This bit is used in master mode only and it controls how CSPI start a SPI burst. 0 XCH bit controls when a SPI burst can start. Write a 1 to XCH bit will start a SPI burst or multiple bursts. (controlled by SSCTL) 1 Immediately start a SPI burst when data is written in TXFIFO.
2 XCH	SPI Exchange Bit. If the SMC bit is cleared, writing a 1 to this bit starts one SPI bursts/multiple SPI bursts according to SSCTL bit. This bit remains set while either the exchange is in progress, or the CSPI is waiting for active input if $\overline{\text{SPIRDY}}$ is enabled through DRCTL. This bit is cleared automatically when all data in the TXFIFO and Shift register have been shifted out. In Slave mode, this bit is ignored. 0 Idle 1 Initiates exchange (write) or busy (read)
1 MODE	SPI Function Mode Select. This bit selects the operating mode of the CSPI. 0 Slave Mode 1 Master Mode
0 EN	SPI Module Enable Control. This bit enables the CSPI. This bit must be asserted before writing to other registers or initiating an exchange. Writing zero to this bit disables the module and resets the internal logic with the exception of the CONREG. The module's internal clocks are gated off whenever the module is disabled. 0 CSPI is disabled. 1 CSPI is enabled.

23.5.3.4 Interrupt Control Register (INTREG)

The 32-bit Interrupt Control Register (INTREG) enables the generation of interrupts to the MCU. The reserved bits cannot be written and always read as 0. If CSPI is disabled, this register reads zero.

Figure 23-18 shows the CNTRL register, and Table 23-7 shows the register's field descriptions.

0x1000_E000C (INTREG) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	TCEN	BOEN	ROEN	RFEN	RHEN	RREN	TFEN	THEN	TEEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-18. Interrupt Control Register Diagram

Table 23-7. INTREG Register Field Descriptions

Field	Description
31–9	Reserved, all bits should read zero.
8 TCEN	Transfer Completed Interrupt enable. This bit enables the Transfer Completed Interrupt. 0 Disable 1 Enable
7 BOEN	Bit Counter Overflow Interrupt enable. This bit enables the Bit Counter overflow Interrupt (More than 32 bits are received in a word). 0 Disable 1 Enable
6 ROEN	RXFIFO Overflow Interrupt enable. The bit enables the RXFIFO Overflow Interrupt. 0 Disable 1 Enable
5 RFEN	RXFIFO Full Interrupt enable. The bit enables the RXFIFO Full Interrupt. 0 Disable 1 Enable
4 RHEN	RXFIFO Half Full Interrupt enable. The bit enables the RXFIFO Half Full Interrupt. 0 Disable 1 Enable
3 RREN	RXFIFO Ready Interrupt enable. The bit enables the RXFIFO Ready Interrupt. 0 Disable 1 Enable
2 TFEN	TXFIFO Full Interrupt enable. The bit enables the TXFIFO Full Interrupt. 0 Disable 1 Enable

Table 23-7. INTREG Register Field Descriptions (continued)

Field	Description
1 THEN	TXFIFO Half Empty Interrupt enable. The bit enables the TXFIFO Half Empty Interrupt. 0 Disable 1 Enable
0 TEEN	TXFIFO Empty Interrupt enable. The bit enables the TXFIFO Empty Interrupt. 0 Disable 1 Enable

23.5.3.5 DMA Control Register (DMAREG)

The DMA Control Register (DMAREG) provides the user a way to use the CSPI in DMA. Direct Memory Access (DMA) allows transfer of data between device and memory. Peripherals such as the CSPI supporting DMA use DMA request and acknowledge signals. The CSPI sends out DMA requests when the appropriate FIFO conditions are matched. The reserved bits cannot be written to and are always read as 0. If the CSPI is disabled, this register is also read as 0.

Figure 23-19 shows the CNTRL register, and Table 23-8 shows the register's field descriptions.

0x1000_E0010 (DMAREG) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	RF DEN	RH DEN	0	0	TH DEN	TE DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-19. DMA Control Register Diagram

Table 23-8. DMAREG Register Field Descriptions

Field	Description
31–6	Reserved, all bits should read zero.
5 RFDEN	RXFIFO Full DMA Request Enable. This bit enables/disables the RXFIFO Full DMA Request. 0 Disable 1 Enable
4 RHDEN	RXFIFO Half Full DMA Request Enable. This bit enables/disables the RXFIFO Half Full DMA Request. 0 Disable 1 Enable
3–2	Reserved, should be cleared.

Table 23-8. DMAREG Register Field Descriptions (continued)

Field	Description
1 THDEN	TXFIFO Half Empty DMA Request Enable. This bit enables/disables the TXFIFO Half Empty DMA Request. 0 Disable 1 Enable
0 TEDEN	TXFIFO Empty DMA Request Enable. This bit enables/disables the TXFIFO Empty DMA Request. 0 Disable 1 Enable

23.5.3.6 Status Register (STATREG)

The CSPI Status Register (STATREG) reflects the status of the CSPI module operating condition. The reserved bits cannot be written and always read as 0. If the CSPI is disabled, this register reads 0x0000_0003.

Figure 23-20 shows the CNTRL register, and Table 23-9 shows the register's field descriptions.

0x1000_E0014 (STATREG) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	TC	BO	RO	RF	RH	RR	TF	TH	TE
W								w1c	w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 23-20. Status Register Diagram

Table 23-9. STATREG Register Field Descriptions

Field	Description
31–9	Reserved, should be cleared.
8 TC	Transfer Completed. When set, this bit indicates that all the data in TXFIFO has been loaded in the Shift register, and the Shift register has shifted out all the bits. Writing 1 to this bit clears it. 0 Busy 1 Transfer Completed
7 BO	Bit Counter Overflow. When set, this bit indicates that Bit Counter is overflows while the Configurable Serial Peripheral Interface is in slave mode (MODE = 0) with SSCTL = 1. Writing 1 to this bit clears it. 0 Bit Counter is not overflowed. 1 Bit Counter is overflowed.

Table 23-9. STATREG Register Field Descriptions (continued)

Field	Description
6 RO	RXFIFO Overflow. When set, this bit indicates that RXFIFO has overflowed. 0 RXFIFO is available. 1 RXFIFO has overflowed.
5 RF	RXFIFO Full. This bit is set when the RXFIFO is full (8 words). 0 Not Full 1 Full
4 RH	RXFIFO Half Full. This bit is set if the RXFIFO is half full (≥ 4 words in RXFIFO). 0 Less than 4 words are stored in RXFIFO. 1 Four or more words are available in RXFIFO.
3 RR	RXFIFO Ready. This bit is set any time there is one or more words stored in RXFIFO (≥ 1 words). 0 No valid data in RXFIFO 1 More than 1 word in RXFIFO
2 TF	TXFIFO Full. This bit is set when if the TXFIFO is full (8 words). 0 TXFIFO is not Full. 1 TXFIFO is Full.
1 TH	TXFIFO Half empty. This bit is set if the TXFIFO is more than half empty (≤ 4 words in TXFIFO). 0 TXFIFO holds more than 4 words. 1 TXFIFO holds 4 or fewer words.
0 TE	TXFIFO Empty. This bit is set if the TXFIFO is empty. 0 TXFIFO contains one or more words. 1 TXFIFO is empty.

23.5.3.7 Sample Period Control Register (PERIODREG)

The Sample Period Control Register (PERIODREG) provides the user a way to insert delays (wait states) between consecutive SPI transfers. Control bits in this register select the clock source for the sample period counter and the delay count indicating the number of wait states to be inserted between data transfers. Delay counts are only applicable when the CSPI module is operating in master mode.

Figure 23-21 shows the CNTRL register, and Table 23-10 shows the register's field descriptions.

0x1000_E0018 (PERIODREG) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CSRC	SAMPLE PERIOD														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-21. Sample Period Control Register Diagram

Table 23-10. PERIODREG Register Field Descriptions

Field	Description
31–16	Reserved, all bits should read zero.
15 CSRC	Clock Source Control. This bit selects the clock source for the sample period counter. 0 SPI Clock (SCLK) 1 CKIL (32.768 KHz)
14–0 SAMPLE PERIOD	Sample Period Control. These bits control the number of wait states to be inserted in data transfers. During the idle clocks, the state of the $\overline{SS}$ output will operate according to the SSCTL control field in CONREG. 0x0000 0 wait states inserted 0x0001 1 wait state inserted .... 0x7FFE 32766 wait states inserted 0x7FFF 32767 wait states inserted

23.5.3.8 Test Control Register (TESTREG)

The Test Control Register (TESTREG) provides the user a mechanism to internally connect the receive and transmit devices of the CSPI module, display the status of the state machine, monitor the contents of the receive and transmit FIFO, and debug the CSPI.

Figure 23-22 shows the CNTRL register, and Table 23-11 shows the register's field descriptions.

0x1000_E01C0 (TESTREG) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SWAP	LBC	0	0	SMSTATUS				RXCNT				TXCNT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-22. Test Control Register Diagram

Table 23-11. TESTREG Register Field Descriptions

Field	Description
31–16	Reserved, All bits should be read as zero.
15 SWAP	Data Swap. This bit is used to swap data as it is read from the RXFIFO. When this bit is set, data read from RXFIFO is swapped. RXDATA[31:0] is swapped as follows: {RXDATA[7:0],RXDATA[15:8], RXDATA[23:16],RXDATA[31:24]} 0 Data read from RXFIFO is unchanged. 1 Data read from RXFIFO is swapped.

Table 23-11. TESTREG Register Field Descriptions (continued)

Field	Description
14 LBC	Loopback Control. This bit is used in Master mode only. When this bit is set, the CSPI module connects the transmitter and receiver sections internally, and the data shifted out from the most-significant bit of the Shift register is looped back into the least-significant bit of the Shift register. In this way, a self-test of the complete transmit/receive path can be made. The output pins are not affected, and the input pins are ignored. 0 Not connected. 1 Internally connected.
13–12	Reserved, all bits should read zero.
11–8 SMSTATUS	State Machine Status. These bits indicate status of the state machine for test purpose.
7–4 RXCNT	RXFIFO Counter. These bits indicate the number of words in RXFIFO. 0000 0 word in RXFIFO 0001 1 word in RXFIFO .... 0111 7 words in RXFIFO 1000 8 words in RXFIFO
3–0 TXCNT	TXFIFO Counter. These bits indicate the number of words in TXFIFO. 0000 0 word in TXFIFO 0001 1 word in TXFIFO .... 0111 7 words in TXFIFO 1000 8 words in TXFIFO

23.6 Timing Diagrams

Figure 23-23 and Figure 23-24 depict the master mode and slave mode timing diagrams of the CSPI and Table 23-12 lists the timing parameters. The values shown in timing diagrams were tested using a worst case core voltage of 1.1 V, slow pad voltage of 2.68 V, and fast pad voltage of 1.65 V.

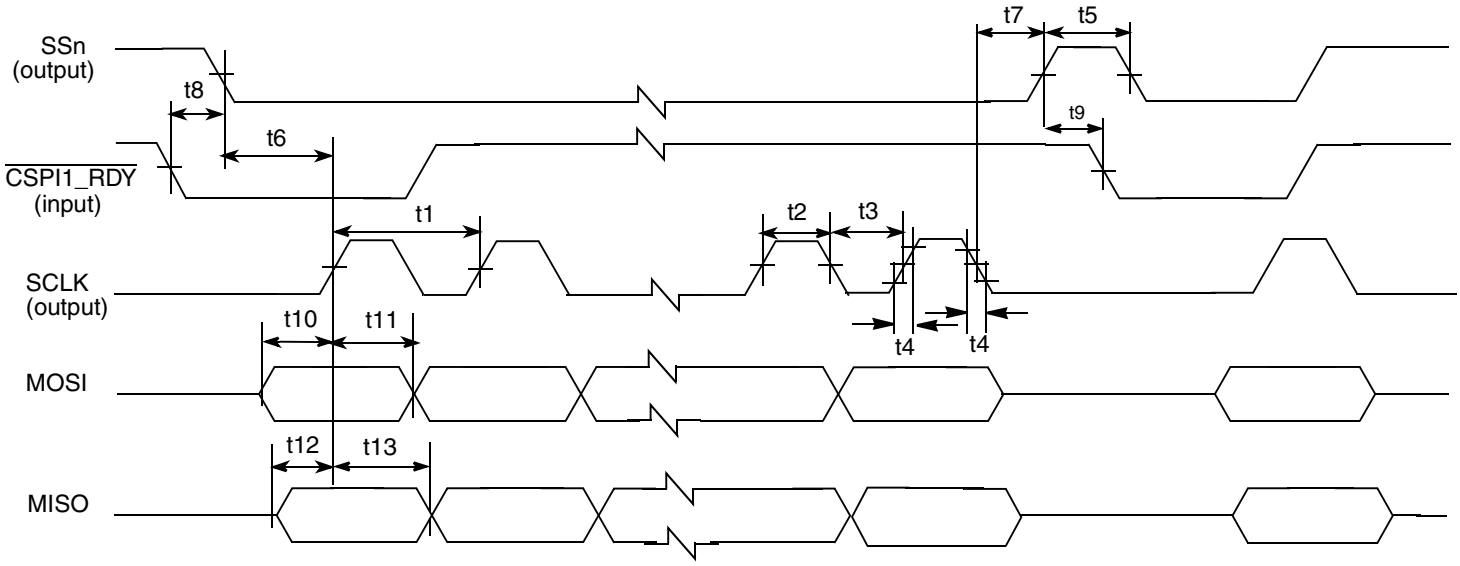


Figure 23-23. CSPI Master Mode Timing Diagram

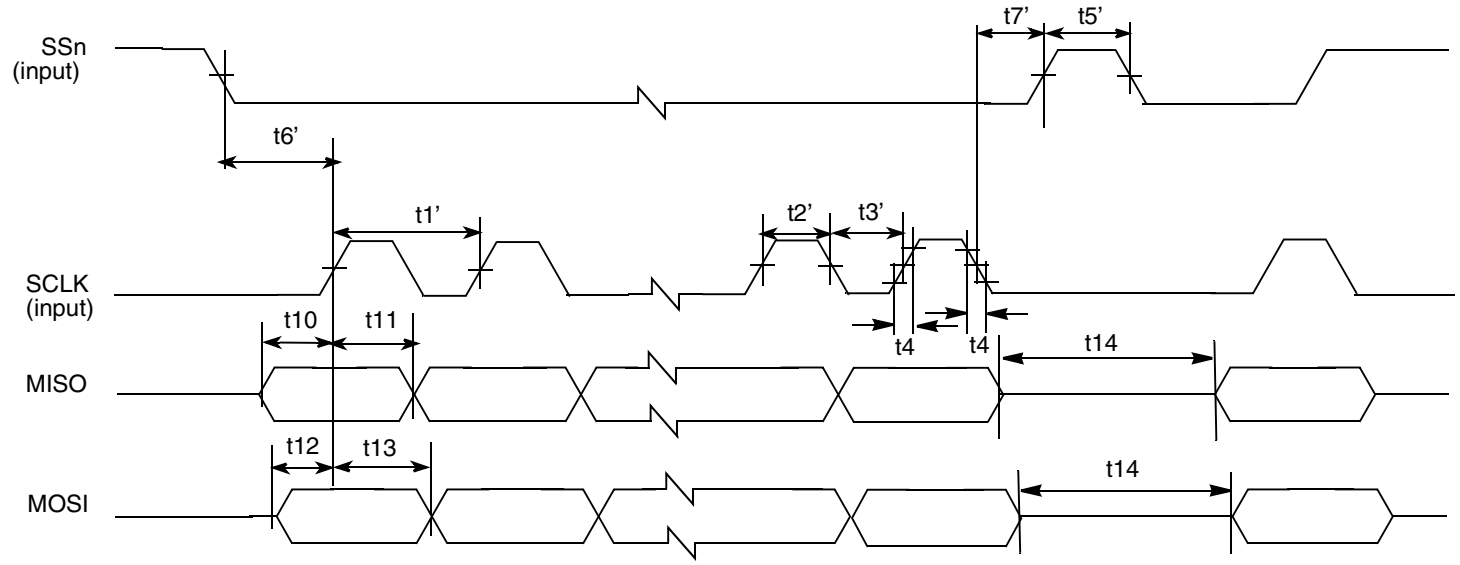


Figure 23-24. CSPI Slave Mode Timing Diagram

Table 23-12. CSPI Interface Timing Parameters

ID Num	Parameter Description	Symbol	Minimum	Maximum	Units
t1	CSPI master SCLK cycle time	t_{clko}	45.12	-	ns
t2	CSPI master SCLK high time	t_{clkoH}	22.65	—	ns
t3	CSPI master SCLK low time	t_{clkoL}	22.47	—	ns

Table 23-12. CSPI Interface Timing Parameters (continued)

ID Num	Parameter Description	Symbol	Minimum	Maximum	Units
t1’	CSPI slave SCLK cycle time	t_{clki}	60.2	—	ns
t2’	CSPI slave SCLK high time	t_{clkiH}	30.1	—	ns
t3’	CSPI slave SCLK low time	t_{clkiL}	30.1	—	ns
t4	CSPI SCLK transition time	t_{pr}[1]	2.6	8.5	ns
t5	SSn output pulse width	t_{Wsso}	$2T_{sclk}$[2] $+T_{wait}$[3]	—	—
t5’	SSn input pulse width	t_{Wssi}	T_{per}[4]	—	—
t6	SSn output asserted to first SCLK edge (SS output setup time)	t_{Ssso}	$3T_{sclk}$	—	—
t6’	SSn input asserted to first SCLK edge (SS input setup time)	t_{Sssi}	T_{per}	—	—
t7	CSPI master: Last SCLK edge to SSn deasserted (SS output hold time)	t_{Hsso}	$2T_{sclk}$	—	—
t7’	CSPI slave: Last SCLK edge to SSn deasserted (SS input hold time)	t_{Hssi}	30	—	ns
t8	CSPI master: CSPI1_RDY low to SSn asserted (CSPI1_RDY setup time)	t_{Srdy}	$2T_{per}$	$5T_{per}$	—
t9	CSPI master: SSn deasserted to CSPI1_RDY low	t_{Hrdy}	0	—	ns
t10	Output data setup time	t_{Sdatao}	(t_{clkoL} or t_{clkoH} or t_{clkiL} or t_{clkiH}) - T_{ipg}[5]	—	—
t11	Output data hold time	t_{Hdatao}	t_{clkoL} or t_{clkoH} or t_{clkiL} or t_{clkiH}	—	—
t12	Input data setup time	t_{Sdatai}	T_{ipg} + 0.5	—	ns
t13	Input data hold time	t_{Hdatai}	0	—	ns
t14	Pause between data word	t_{pause}	0	—	ns

1 The output SCLK transition time is tested with 25pF drive.

2 T_{sclk} = CSPI clock period

3 T_{wait} = Wait time as per the Sample Period Control Register value.

4 T_{per} = CSPI reference baud rate clock period (PERCLK2)

5 T_{ipg} = CSPI main clock IPG_CLOCK period

Chapter 24
Inter-Integrated Circuit (I^2C)

The Inter-Integrated Circuit (I^2C) module provides the functionality of a standard I^2C slave and master. The I^2C module is designed to be compatible with the standard Philips I^2C bus protocol. The i.MX27 device contains two identical I^2C modules. Figure 24-1 shows the I^2C block diagram.

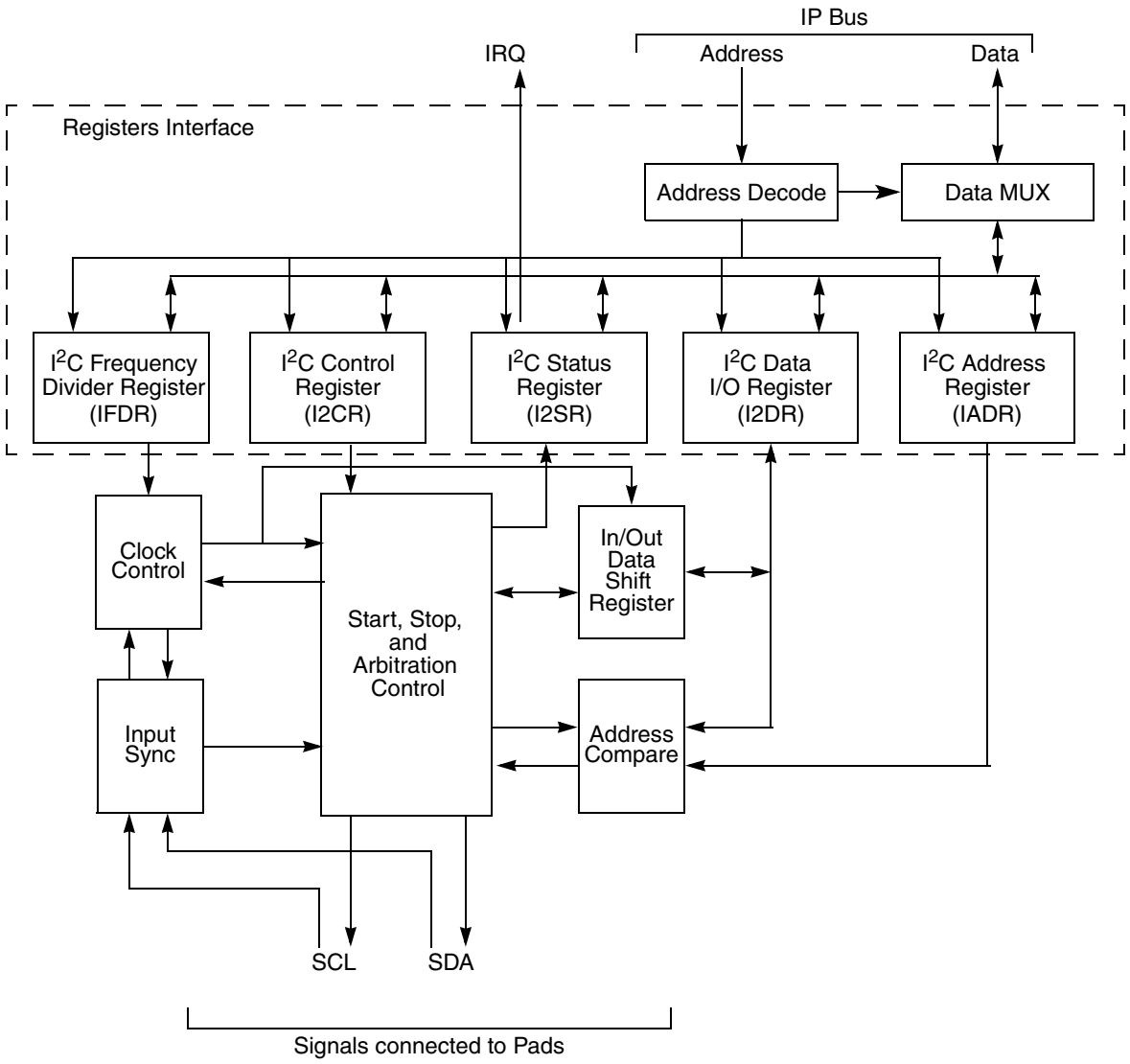


Figure 24-1. I^2C Block Diagram

24.1 Overview

The I^2C is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. This bus is suitable for applications requiring occasional communications over a short distance between many devices. The flexible I^2C allows additional devices to be connected to the bus for expansion and system development. Figure 24-2 provides the connection diagram.

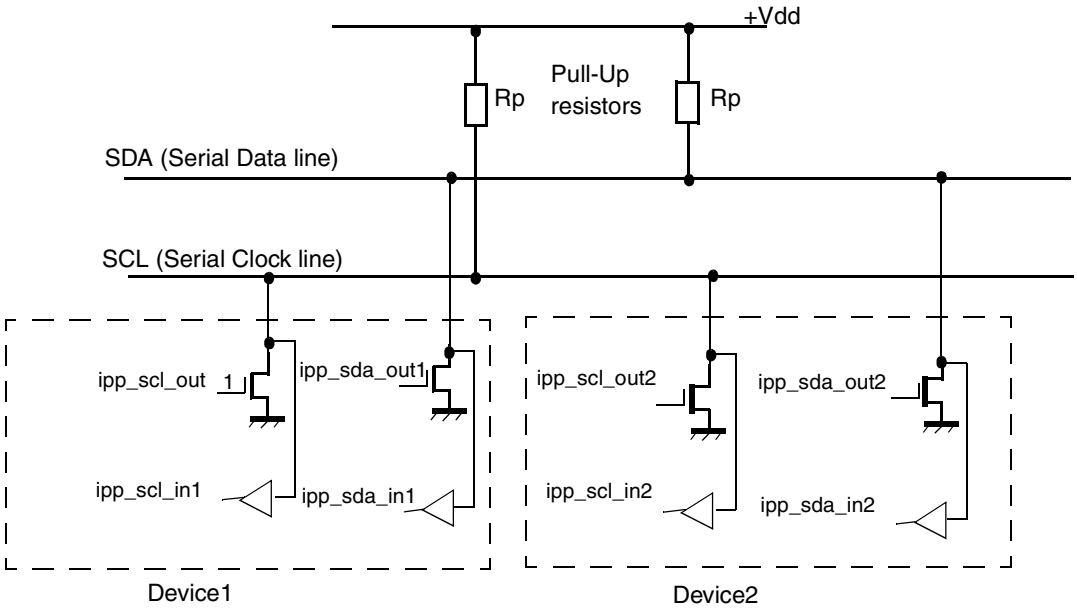


Figure 24-2. Connection of Devices to I^2C Bus

The I^2C operates up to 400 kbps, but it depends on the pad loading and timing. For pad requirement details, refer to Philips I^2C Bus Specification, Version 2.1. The I^2C system is a true multiple-master bus including arbitration and collision detection that prevents data corruption if multiple devices attempt to control the bus simultaneously. This feature supports complex applications with multiprocessor control and can be used for rapid testing and alignment of end products through external connections to an assembly-line computer.

24.1.1 Features

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

- Compatibility with I^2C bus standard
- Multiple-master operation
- Software-programmable for one of 64 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt

- Start and stop signal generation/detection
- Repeated START signal generation
- Acknowledge bit generation/detection
- Bus-busy detection

24.2 External Signal Description

Two pins are required for I^2C:

- I2C_SCL Bidirectional Clock Pin
- I2C_SDA Bidirectional Data Pin

For I^2C compliance, all devices connected to the SCL and SDA signals must have open-drain or open-collector outputs. The logic AND function is exercised on both lines with external pull-up resistors.

The module port signals going to the pad are tabulated in Table 24-2.

Table 24-2. Signal Properties

Name	Port	Function	Reset State	Pull-Up
IPP_SCL_IN	—	Serial Clock input	1	—
IPP_SCL_OUT	—	Serial Clock output	1	Active
IPP_SCL_OUT_EN	—	Serial Clock output enable	0	—
IPP_SDA_IN	—	Serial Data input	1	—
IPP_SDA_OUT	—	Serial Data output	1	Active
IPP_SDA_OUT_EN	—	Serial Data output enable	0	—

24.2.1 Detailed External Signal Descriptions

The following three signals are connected to the bidirectional driver for the SCL I/O Pad (through the IOMUX module):

- IPP_SCL_IN—Serial Input Clock
- IPP_SCL_OUT—Serial Output Clock
- IPP_SCL_OUT_EN—Serial Output Clock Enable

The pad needs to have open-drain connectivity. IPP_SCL_IN will be the input signal from the pad. IPP_SCL_OUT will be the output to the pad from the module. IPP_SCL_OUT_EN will act as the output enable.

The following three signals are connected to the bidirectional driver for the SDA I/O Pad (through the IOMUX module):

- IPP_SDA_IN—Serial Input Data
- IPP_SDA_OUT—Serial Output Data
- IPP_SDA_OUT _EN—Serial Output Data Enable

The pad should have open-drain connectivity. IPP_SDA_IN will be the input signal from the pad. IPP_SCL_OUT will be the output to the pad from module. IPP_SDA_OUT_EN will act as the output enable.

24.3 Memory Map and Register Definition

The I^2C module contains five 16-bit registers. Section 24.3.3, "Register Descriptions" provides the detailed descriptions for all of the I^2C registers.

24.3.1 I^2C Memory Map

Table 24-3 shows the I^2C memory map.

Table 24-3. I^2C Memory Map

Address	Register	Access	Reset Value	Section/Page
0x1001_2000 (IADR1) 0x1001_D000 (IADR2)	I^2C Address Register	R/W	0x0000	24.3.3.1/24-6
0x1001_2004 (IFDR1) 0x1001_D004 (IFDR2)	I^2C Frequency Divider Register	R/W	0x0000	24.3.3.2/24-6
0x1001_2008 (I2CR1) 0x1001_D008 (I2CR2)	I^2C Control Register	R/W	0x0000	24.3.3.3/24-7
0x1001_200C (I2SR1) 0x1001_D008 (I2CR2)	I^2C Status Register	R/W	0x0081	24.3.3.4/24-9
0x1001_2010 (I2DR1) 0x1001_D010 (I2DR2)	I^2C Data I/O Register	R/W	0x0000	24.3.3.5/24-10

NOTE

There are registers at addresses 0x1001_02, 0x1001_06, 0x1001_a, 0x1001_0e, which are reserved for future additions.

24.3.2 Register Summary

Figure 24-3 shows the key to the register fields, and Table 24-5 shows the register figure conventions.

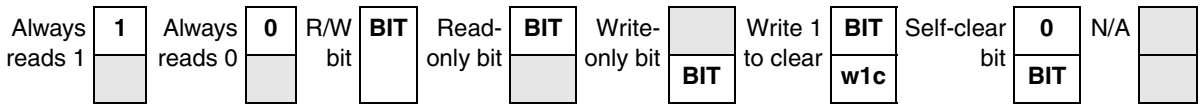


Figure 24-3. Key to Register Fields

Table 24-5. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.

Table 24-5. Register Figure Conventions (continued)

Convention	Description
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 24-4 shows the I^2C register summary.

Table 24-4. I^2C Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1001_2010 (I2DR1) 0x1001_D010 (I2DR2)	R	0	0	0	0	0	0	0	0	ADR							0
	W																
0x1001_2004 (IFDR1) 0x1001_D004 (IFDR2)	R	0	0	0	0	0	0	0	0	0	0	IC					
	W																
0x1001_2008 (I2CR1) 0x1001_D008 (I2CR2)	R	0	0	0	0	0	0	0	0	IEN	IIEN	MSTA	MTX	TXAK	0	0	0
	W														RSTA		
0x1001_200C (I2SR1) 0x1001_D00C (I2SR2)	R	0	0	0	0	0	0	0	0	ICF	IAAS	IBB	IAL	0	SRW	IIF	RXAK
	W																
0x1001_2010 (I2DR1) 0x1001_D010 (I2DR2)	R	0	0	0	0	0	0	0	0	DATA							
	W																

24.3.3 Register Descriptions

This section contains the detailed register descriptions for the I2C registers in address order.

24.3.3.1 I2C Address Register (IADR)

Figure 24-6 shows the I2C Address Register; Table 24-5 provides its field descriptions.

0x1001_2000 (IADR1)
0x1001_D000 (IADR2)

Access: User read/write

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

Figure 24-6. I2C Address Register

The IADR holds the address the I2C responds to when addressed as a slave.

NOTE

The slave address is not the address sent on the bus during the address transfer. This register is not reset by a software reset.

Table 24-5. I2C Address Register Field Descriptions

Field	Description
15–8	Reserved
7–1 ADR	Slave address. Contains the specific slave address to be used by the I2C module. Slave mode is the default I2C mode for an address match on the bus.
0	Reserved

24.3.3.2 I2C Frequency Register (IFDR)

The IFDR provides a programmable prescaler to configure the clock for bit-rate selection. The register does not get reset by software reset. Figure 24-7 shows the I2C Frequency Register; Table 24-6 provides its field descriptions.

0x1001_2004 (IFDR1)
0x1001_D004 (IFDR2)

Access: User read/write

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

Figure 24-7. I2C Frequency Register

Table 24-6. I^2C Frequency Register Field Descriptions

Field	Description
15–6	Reserved
5–0 IC	I^2C clock rate. Prescales the clock for bit-rate selection. Due to potentially slow SCL and SDA rise and fall times, bus signals are sampled at the prescaler frequency. The serial bit clock frequency is equal to IPG_CLK_PATREF divided by the divider shown in Table 24-7. **Note:** The IC can be changed anywhere in a program. I^2C protocol supports bit rates up to 400 kbps. The IC bits need to be programmed in accordance with this constraint.

Table 24-7. IFDR Register Field Values

IC	Divider	IC	Divider	IC	Divider	IC	Divider
0x00	30	0x10	288	0x20	22	0x30	160
0x01	32	0x11	320	0x21	24	0x31	192
0x02	36	0x12	384	0x22	26	0x32	224
0x03	42	0x13	480	0x23	28	0x33	256
0x04	48	0x14	576	0x24	32	0x34	320
0x05	52	0x15	640	0x25	36	0x35	384
0x06	60	0x16	768	0x26	40	0x36	448
0x07	72	0x17	960	0x27	44	0x37	512
0x08	80	0x18	1152	0x28	48	0x38	640
0x09	88	0x19	1280	0x29	56	0x39	768
0x0A	104	0x1A	1536	0x2A	64	0x3A	896
0x0B	128	0x1B	1920	0x2B	72	0x3B	1024
0x0C	144	0x1C	2304	0x2C	80	0x3C	1280
0x0D	160	0x1D	2560	0x2D	96	0x3D	1536
0x0E	192	0x1E	3072	0x2E	112	0x3E	1792
0x0F	240	0x1F	3840	0x2F	128	0x3F	2048

24.3.3.3 I^2C Control Register (I2CR)

The I2CR is used to enable the I^2C module and the I^2C interrupt. It also contains bits that govern operation as a slave or a master. Figure 24-8 shows the I^2C Control Register; Table 24-8 provides its field descriptions.

0x1001_2008 (I2CR1)
0x1001_D008 (I2CR2)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	IEN	IIEN	MSTA	MTX	TXAK	0	0	0
W														RSTA		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-8. I^2C Control Register

Table 24-8. I^2C Control Register Field Descriptions

Field	Description
15–8	Reserved
7 IEN	I^2C enable. Also controls the software reset of the entire I^2C module. Resetting the bit generates an internal reset to the module. If the module is enabled in the middle of a byte transfer, slave mode ignores the current bus transfer and starts operating when the next start condition is detected. Master mode is not aware that the bus is busy; so initiating a start cycle may corrupt the current bus cycle, ultimately causing either the current master or the I^2C module to lose arbitration. After which, bus operation returns to normal. 0 The module is disabled, but registers can still be accessed. 1 The I^2C module is enabled. This bit must be set before any other I2CR bits have any effect.
6 IIEN	I^2C interrupt enable. 0 I^2C module interrupts are disabled, but the status flag I2SR[IIF] continues to be set when an interrupt condition occurs. 1 I^2C module interrupts are enabled. An I2C interrupt occurs if I2SR[IIF] is also set.
5 MSTA	Master/slave mode select bit. If the master loses arbitration, MSTA is cleared without generating a STOP signal. **Note:** Module clock should be on for writing to the MSTA bit. 0 Slave mode. Changing MSTA from 1 to 0 generates a STOP and selects slave mode. 1 Master mode. Changing MSTA from 0 to 1 signals a START on the bus and selects master mode.
4 MTX	Transmit/receive mode select bit. Selects the direction of master and slave transfers. 0 Receive. When a slave is addressed, the software should set MTX according to the slave read/write bit in the I^2C status register (I2SR[SRW]). 1 Transmit. In master mode, MTX should be set according to the type of transfer required. Therefore, for address cycles, MTX is always 1.
3 TXAK	Transmit acknowledge enable. Specifies the value driven onto SDA during acknowledge cycles for both master and slave receivers. **Note:** Writing TXAK applies only when the I^2C bus is a receiver. 0 An acknowledge signal is sent to the bus at the ninth clock bit after receiving one byte of data. 1 No acknowledge signal response is sent (that is, the acknowledge bit = 1).
2 RSATA	Repeat start. Always reads as 0. Attempting a repeat start without bus mastership causes loss of arbitration. 0 No repeat start 1 Generates a repeated START condition
1–0	Reserved

24.3.3.4 I^2C Status Register (I2SR)

The I2SR contains bits that indicate transaction direction and status. Figure 24-9 shows the I^2C Address Register; and Table 24-9 provides its field descriptions.

0x1001_200C (I2SR1)
0x1001_D00C (I2SR2)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ICF	IAAS	IBB	IAL	0	SRW	IIF	RXAK
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

Figure 24-9. I^2C Status Register

Table 24-9. I^2C Status Register Field Descriptions

Field	Description
15–8	Reserved
7 ICF	Data transferring bit. While one byte of data is transferred, ICF is cleared. 0 Transfer is in progress. 1 Transfer is complete, and set by the falling edge of the ninth clock of a byte transfer.
6 IAAS	I^2C addressed as a slave bit. The CPU is interrupted if the interrupt enable (I2CR[IIEN]) is set. The CPU must check the slave read/write bit (SRW) and set its TX/RX mode accordingly. Writing to I2CR clears this bit. 0 Not addressed 1 Addressed as a slave. Set when its own address (IADR) matches the calling address.
5 IBB	I^2C bus busy bit. Indicates the status of the bus. 0 Bus is idle. If a STOP signal is detected, IBB is cleared. 1 Bus is busy. When START is detected, IBB is set.
4 IAL	Arbitration lost. Set by hardware in the following circumstances (IAL must be cleared by software by writing a “0” to it): • SDA input sampled low when the master drives high during an address or data-transmit cycle. • SDA input sampled low when the master drives high during the acknowledge bit of a data-receive cycle. For the above two cases, the bit is set at the falling edge of 9th SCL clock during the ACK cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in slave mode. • A stop condition is detected when the master did not request it. **Note:** Software cannot set the bit. 0 No arbitration is lost. 1 Arbitration is lost.
3	Reserved
2 SRW	Slave read/write. When the I^2C is addressed as a slave, IAAS is set, and the slave read/write bit (SRW) indicates the value of the R/W command bit of the calling address sent from the master. SRW is valid only when a complete transfer has occurred, no other transfers have been initiated, and the I^2C module is a slave and has an address match. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave

Table 24-9. I2C Status Register Field Descriptions (continued)

Field	Description
1 IIF	I2C interrupt. Must be cleared by the software by writing a “0” to it in the interrupt routine. **Note:** The software cannot set the bit. 0 No I2C interrupt is pending. 1 An interrupt is pending. This causes a processor interrupt request (if the interrupt enable is asserted [IIEN = 1]). The interrupt is set when one of the following occurs: • One byte transfer is completed (the interrupt is set at the falling edge of the ninth clock). • An address is received that matches its own specific address in slave-receive mode. • Arbitration is lost.
0 RXAK	Received acknowledge. This is the value received of the SDA input for the acknowledge bit during a bus cycle. 0 An “acknowledge” signal was received after the completion of an 8-bit data transmission on the bus. 1 A “No acknowledge” signal was detected at the ninth clock.

24.3.3.5 I2C Data Register (I2DR)

In master-receive mode, reading the data register (I2DR) allows a read to occur and initiates the next byte to be received. In slave mode, the same function is available after it is addressed. Figure 24-10 shows the I2C Data Register; Table 24-10 provides its field descriptions.

0x1001_2010 (I2DR1)
0x1001_D010 (I2DR2)

Access: User read/write

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

Figure 24-10. I2C Data Register

Table 24-10. I2C Data Register Field Descriptions

Field	Description
15–8	Reserved
7–0 DATA	Data Byte. Holds the last data byte received or the next data byte to be transferred. Software writes the next data byte to be transmitted or reads the data byte received.

NOTE

The core-written value in I2DR cannot be read back by the core: Only data written by the I2C bus side can be read.

24.4 Functional Description

24.4.1 I^2C System Configuration

Out of a reset, the I^2C module defaults to slave receive operations. Thus, when not operating as a master or responding to a slave transmit address, the I^2C module will default to the slave receiver state. For exceptions, see Section 24.5.1, "Initialization Sequence."

NOTE

The I^2C module is designed to be compatible with the Philips I^2C bus protocol. For information on system configuration, protocol, and restrictions, refer to the I^2C Bus Specification, Version 2.1. The I^2C module supports Standard and Fast modes only.

24.4.2 I^2C Protocol

The I^2C communication protocol consists of six components, as follows:

- START
- Data Source/Recipient
- Data Direction
- Slave Acknowledge
- Data Acknowledge
- STOP

See Figure 24-11 for the I^2C standard communication protocol, as defined in the following sections.

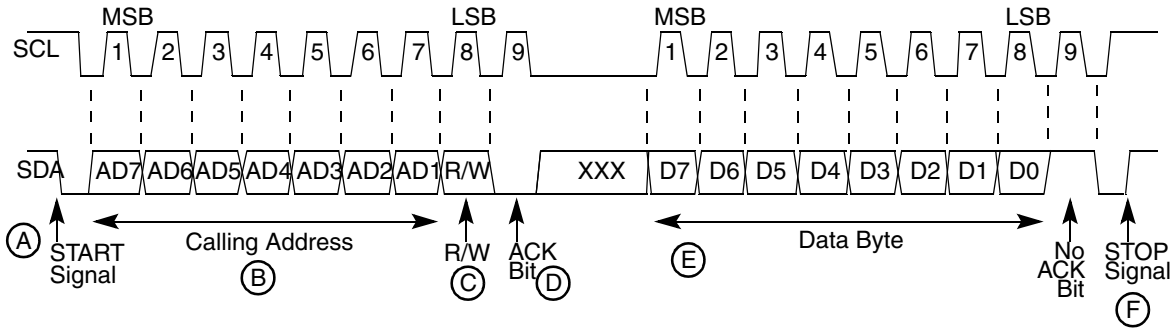


Figure 24-11. I^2C Standard Communication Protocol

24.4.2.1 START Signal

When no other device is a bus master (both SCL and SDA lines are at logic high), a device can initiate communication by sending a START signal (see A in Figure 24-11). A START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a data transfer (each data transfer can be several bytes long) and awakens all slaves.

24.4.2.2 Slave Address Transmission

The master sends the slave address in the first byte after the START signal (B). After the seven-bit calling address, it sends the R/W bit (C), which tells the slave data transfer direction.

Each slave must have a unique address. An I^2C master must not transmit an address that is the same as its slave address; it cannot be master and slave at the same time.

The slave whose address matches that sent by the master pulls SDA low at the ninth clock (D) to return an acknowledge bit.

24.4.2.3 Data Transfer

When successful slave addressing is achieved, the data transfer can proceed (E) on a byte-by-byte basis in the direction specified by the R/W bit sent by the calling master.

Data can be changed only while SCL is low and must be held stable while SCL is high, as shown in Figure 24-11. SCL is pulsed once for each data bit, with the mishap being sent first. The receiving device must acknowledge each byte by pulling SDA low at the ninth clock; therefore, a data byte transfer takes nine clock pulses.

If it does not acknowledge the master, the slave receiver must leave SDA high. The master can then generate a STOP signal to abort the data transfer or generate a START signal (a repeated start, as shown in Figure 24-12) to start a new calling sequence.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means end-of-data to the slave. The slave releases SDA for the master to generate a STOP or START signal.

24.4.2.4 STOP Signal

The master can terminate communication by generating a STOP signal to free the bus. A STOP signal is defined as a low-to-high transition of SDA while SCL is at logical high (F).

NOTE

A master can generate a STOP even if the slave has made an acknowledgment; at which point, the slave must release the bus.

24.4.2.5 Repeat Start

Instead of signalling a STOP, the master can repeat the START signal, followed by a calling command (see A in Figure 24-12). A repeated START occurs when a START signal is generated without first generating a STOP signal to end the communication. The master uses a repeated START to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

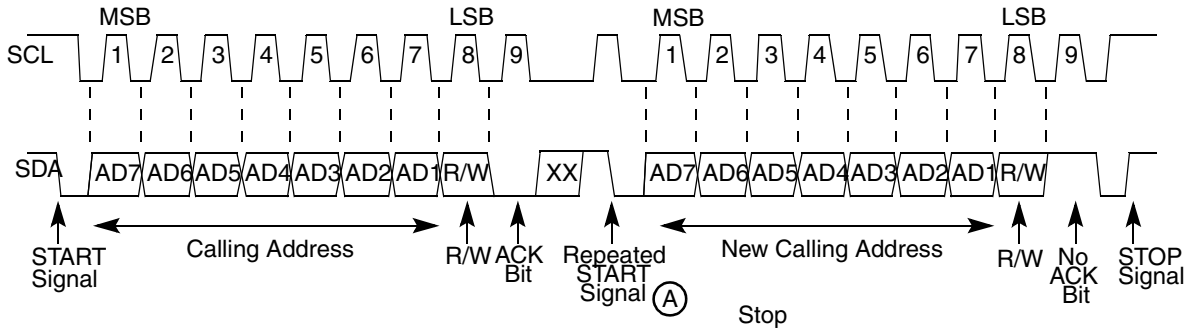


Figure 24-12. Repeated START

24.4.3 Arbitration Procedure

If multiple devices simultaneously request the bus, the bus clock is determined by a synchronization procedure in which the low period equals the longest clock-low period among the devices, and the high period equals the shortest. A data arbitration procedure determines the relative priority of competing devices. A device loses arbitration if it sends logic high while another sends logic low; it immediately switches to slave-receive mode and stops driving SDA. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, hardware sets the arbitration lost bit in the I^2C Status register (I2SR[IAL] to indicate loss of arbitration.

24.4.4 Clock Synchronization

Because wire-AND logic is used, a high-to-low transition on SCL affects devices connected to the bus. Devices start counting their low period when the master drives SCL low. When a device clock goes low, it holds SCL low until the clock high state is reached. However, the low-to-high change in this device clock may not change the state of SCL if another device clock is still in its low period. Therefore, the device with the longest low period holds the synchronized clock SCL low. Devices with shorter low periods enter a high wait state during this time (see Figure 24-13). When all devices involved have counted off their low period, the synchronized clock SCL is released and pulled high. There is then no difference between device clocks and the state of SCL, so all of the devices start counting their high periods. The first device to complete its high period pulls SCL low again.

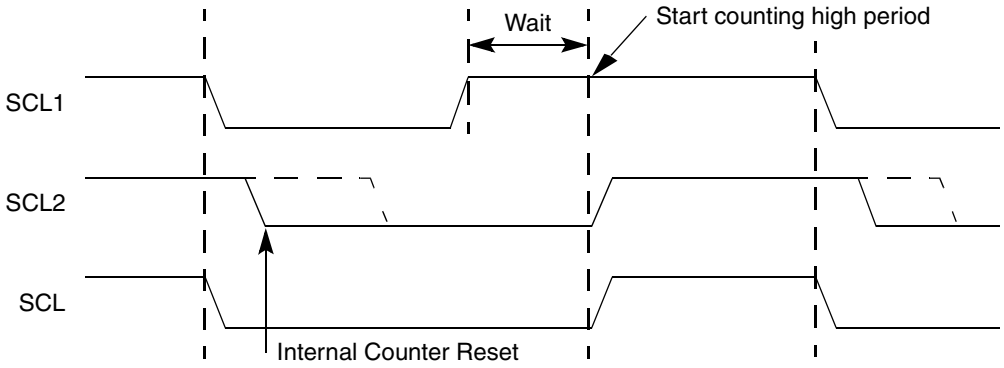


Figure 24-13. Synchronized Clock SCL

24.4.5 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfers. Slave devices can hold SCL low after completing one byte transfer (9 bits). In such a case, the clock mechanism halts the bus clock and forces the master clock into a wait state until the slave releases SCL.

24.4.6 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is longer than the master SCL low period, the resulting SCL bus signal low period is stretched.

24.4.7 IP Bus Accesses

I^2C is a 16-bit IP module. Only halfword accesses should be performed to the module.

24.4.8 Generation of Transfer Error on IP Bus

If an address is received on the IP slave bus interface that is not implemented, an access error is generated (IPS_XFR_ERR is asserted). The input pin resp_sel provides the configuration capability to generate this response. The resp_sel pin must be asserted to enable the IPS_XFR_ERR signal.

24.5 Initialization/Application Information

24.5.1 Initialization Sequence

Before the interface can transfer serial data, registers must be initialized, as follows:

1. Set the data sampling rate (IFDR[IC] to obtain SCL frequency from the system bus clock. See Section 24.3.3.2, “I2C Frequency Register (IFDR).”
2. Update the address in the (IADR) to define its slave address (address can range from 0 to 0x7f).
3. Set the I^2C enable bit (I2CR[IEN]) to enable the I^2C bus interface system.
4. Modify the bits in the I^2CR to select master/slave mode, transmit/receive mode, and interrupt-enable or not.

24.5.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the master transmitter mode. On a multiple-master bus system, the busy bus (I2SR[IBB]) must be tested to determine whether the serial bus is free. If the bus is free (IBB = 0), the START signal and the first byte (the slave address) can be sent. The data written to the data register comprises the address of the desired slave and the LSB indicates the transfer direction.

The free time between a STOP and the next START condition is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be

necessary to wait until the I^2C is busy after writing the calling address to the data register (I2DR) before proceeding to load data into the data register (I2DR).

24.5.3 Post-Transfer Software Response

Sending or receiving a byte sets the data transferring bit (I2SR[ICF]), which indicates one byte communication is finished. Upon completion, the interrupt status (I2SR[IIF]) is also set. An external interrupt is generated if the interrupt enable (I2CR[IIEN]) is set. The software must first clear the interrupt status (I2SR[IIF]) in the interrupt routine. (See the flowchart in Figure 24-14.) The data transferring bit (I2SR[ICF]) is cleared either by reading from I2DR in receive mode or by writing to this register in transmit mode.

The software can service the I^2C I/O in the main program by monitoring the interrupt status (I2SR[IIF]) if the interrupt enable is de-asserted. In this case, the interrupt status should be polled of the data transferring bit (I2SR[ICF]) because the operation is different when arbitration is lost.

When an interrupt occurs at the end of the address cycle, the master is always in transmit mode; that is, the address is sent. If master receive mode is required, then (I2DR[R/W], I2CR[MTX] should be toggled.

During slave-mode address cycles (I2SR[IAAS] = 1), the slave read/write bit I2SR[SRW] is read to determine the direction of the next transfer. The transmit/receive bit (I2CR[MTX]) should also be programmed accordingly. For slave-mode data cycles (IAAS = 0), SRW is invalid. MTX should be read to determine the current transfer direction.

24.5.4 Generation of STOP

A data transfer ends when the master signals a STOP, which can occur after all data is sent.

For a master receiver to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last data byte. This is done by setting the transmit acknowledge bit (I2CR[TXAK]) before reading the next-to-last byte. Before the last byte is read, a STOP signal must be generated.

24.5.5 Generation of Repeated START

After the data transfer, if the master still wants the bus, it can signal another START followed by another slave address without signalling a STOP.

24.5.6 Slave Mode

In the slave interrupt service routine (see Figure 24-14), the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (I2CR[MTX]) according to the I2SR[SRW]. Writing to the I2CR clears the IAAS automatically. The only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred; interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer can now be initiated by writing information to I2DR for slave transmits, or read from I2DR in slave-receive mode. A dummy read of I2DR in slave/receive mode releases SCL, allowing the master to send data.

In the slave transmitter routine, the receive acknowledge bit (I2SR[RXAK]) must be tested before sending the next byte of data. Setting RXAK means an end-of-data signal from the master receiver, after which the software must switch it from transmitter to receiver mode. Reading the data register (I2DR) then releases SCL so that the master can generate a STOP signal.

24.5.7 Arbitration Lost

If several devices try to engage the bus at the same time, one becomes master. Hardware immediately switches devices that lose arbitration to slave receive mode. Data output to SDA stops, but SCL is still generated until the end of the byte during which arbitration is lost. An interrupt occurs at the falling edge of the ninth clock of this transfer if the arbitration is lost (I2SR[IAL] = 1), and the slave mode is selected (I2CR[MSTA] = 0). See the flowchart in Figure 24-14.

If a device that is not a master tries to transmit or do a START, hardware inhibits the transmission, clears MSTA without signalling a STOP, generates an interrupt to the CPU, and sets IAL to indicate a failed attempt to engage the bus. When considering these cases, the slave service routine should first test IAL, and the software should clear it if it is set.

For Multi-master mode, when an I^2C module is enabled when the bus is busy and asserts START, the IAL bit gets set only for SDA=0, SCL=0/1, SDA=1, and SCL=0, but not for SDA=1 and SCA=1, which is the same as bus idle state.

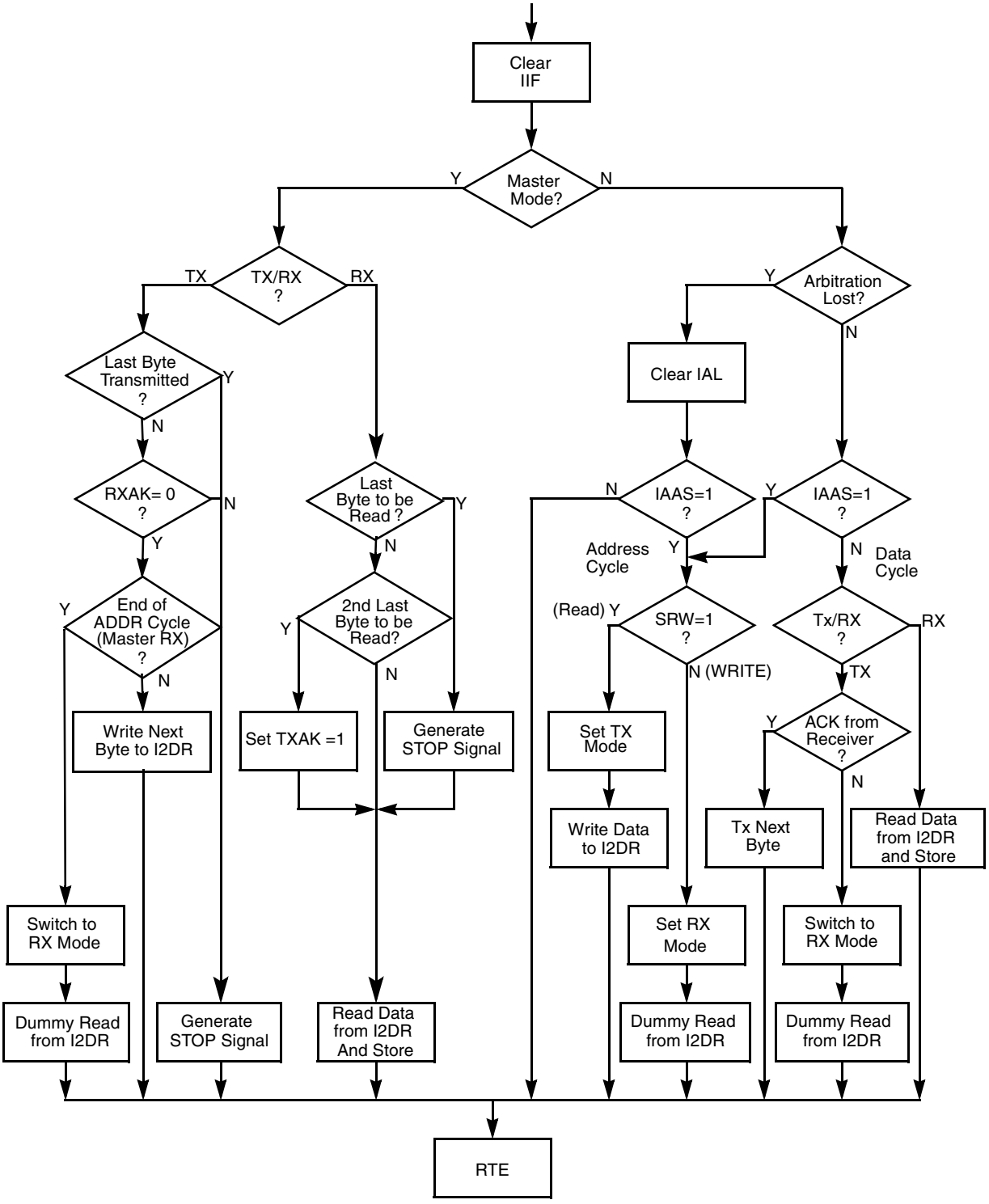


Figure 24-14. Flowchart of Typical I^2C Interrupt Routine

NOTE

For a repeated start-only, the stop generation stage will not occur in master mode. A loop will repeat itself without stopping for the next start.

24.5.8 Timing Section

Figure 24-15 provides an illustration of the timing for the serial data line (SDA) and serial clock line (SCL) devices on the I^2C bus.

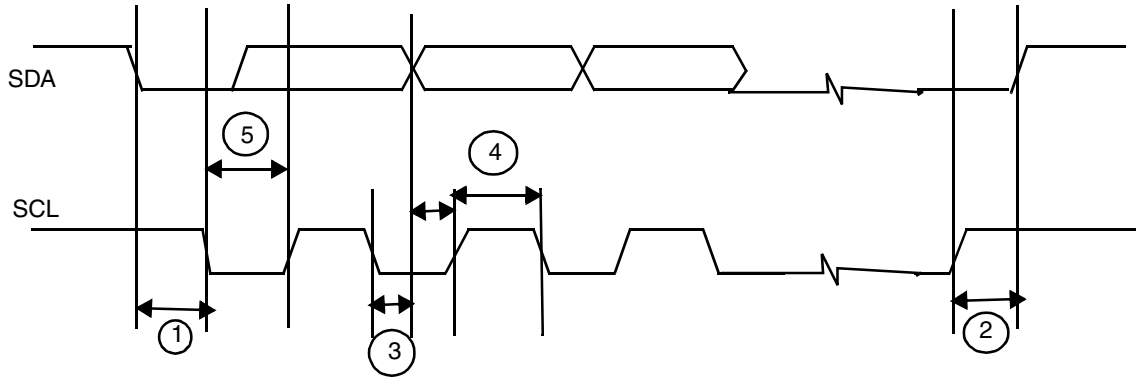


Figure 24-15. Definition of Timing for Devices on I^2C Bus

Table 24-11 provides a list of bus timing parameters.

Table 24-11. I^2C Bus Timing Parameters

Reference Number	Parameter	Maximum (w.r.t ipg_clk_patref)	Minimum (w.r.t ipg_clk_patref)
1	Hold time (repeated) START condition	—	4
2	Setup time for STOP condition	—	4
3	Data hold time	(0.27) * Divider	—
4	HIGH of the SCL Period	—	(0.4) * Divider (Master mode)
5	LOW period of the SCL Clock	—	(0.4) * Divider (Master mode)

NOTE

See Table 24-7 for details for Divider values.

Chapter 25
Keypad Port (KPP)

The Keypad Port (KPP) is a 16-bit peripheral that can be used as a keypad matrix interface or as general purpose input/output (I/O). Figure 25-1 shows the KPP block diagram.

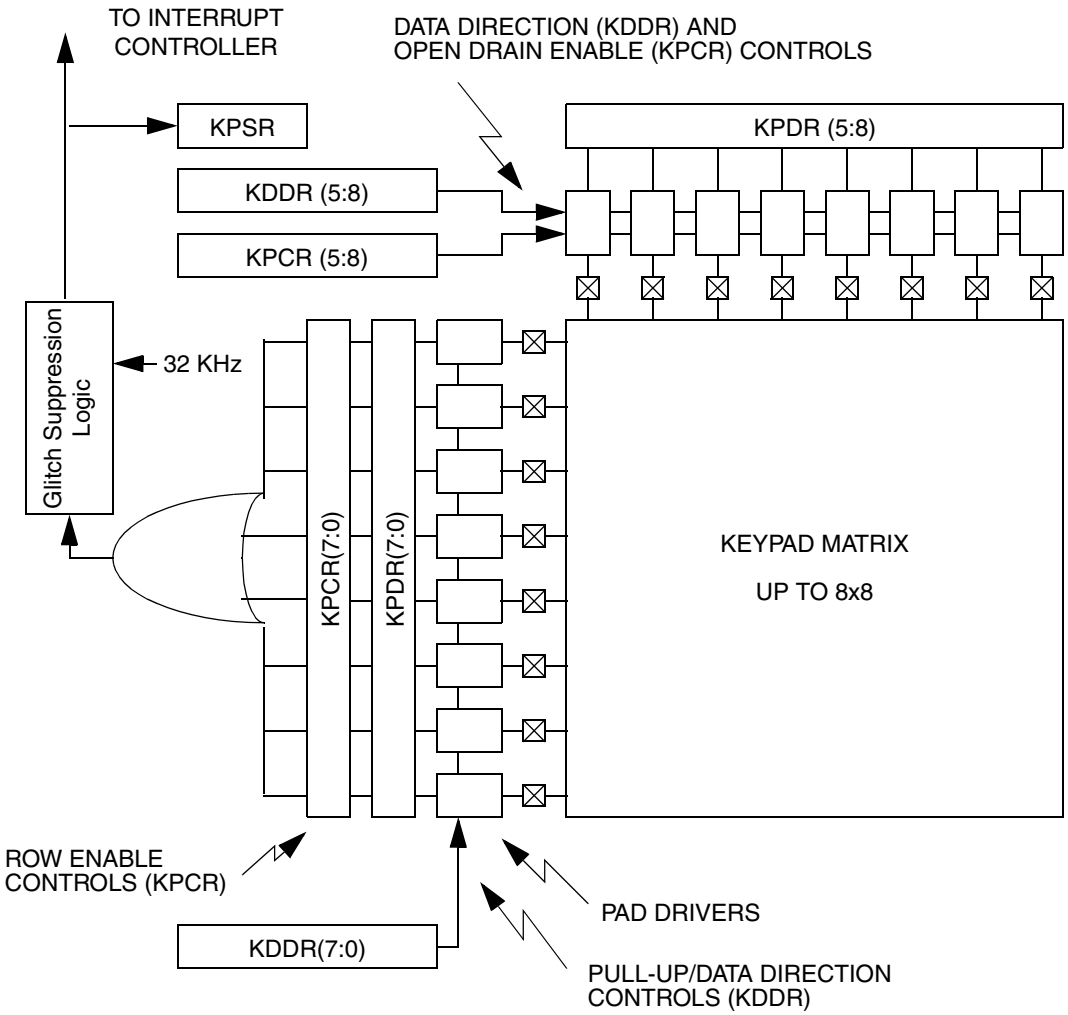


Figure 25-1. KPP Peripheral Block Diagram

25.1 Overview

The KPP is designed to interface with the keypad matrix with 2-point contact or 3-point contact keys. The KPP is designed to simplify the software task of scanning a keypad matrix. With appropriate software

support, the KPP is capable of detecting, debouncing, and decoding one or multiple keys pressed simultaneously on the keypad.

25.1.1 Features

The KPP includes these distinctive features:

- Supports up to an 8 x 8 external key pad matrix
- Port pins can be used as general purpose I/O
- Open drain design
- Glitch suppression circuit design
- Multiple-key detection
- Long key-press detection
- Standby key-press detection
- Synchronizer chain clear
- Supports a 2-point and 3-point contact key matrix

25.1.2 Modes of Operation

This module supports the following modes of operation:

- Run Mode—This is the normal functional mode in which the KPP can detect any key press event.
- Low Power Modes—The keypad can detect any key press even in low power modes (when there is no MCU clock).

25.2 External Signal Description

25.2.1 Overview

There are 16 pins dedicated to the KPP. Keypads of any configuration up to eight rows and eight columns are supported through the software configuration of the peripheral pins. Any pins not used for the keypad are available as general purpose I/O. The registers are configured such that the pins can be treated as an I/O port up to 16 bits wide.

25.2.1.1 Input Pins

Any of the 16 pins associated with the KPP can be configured as inputs by writing a “0” to the appropriate bits in the KDDR. Additionally, the least significant 8 bits (ROW inputs) corresponding to KDDR7:0 have internal pull-ups, which are enabled when the pin is used as an input.

25.2.1.2 Output Pins

Any of these 16 KPP pins can be configured as outputs by writing the appropriate bits in the KDDR to a “1”. Additionally, the 8 most significant bits (15–8) can be designated as open drain outputs by writing a

"1" to the appropriate bits in the KPCR. The lower 8 bits (7–0) are always in "totem pole" style, driven when configured as outputs. See Table 25-1.

Table 25-1. Keypad Port Column Modes

KDDR (15:8)	KPCR (15:8)	Pin Function
0	x	Input
1	0	Totem-Pole Output
1	1	Open-Drain Output

NOTE

Totem pole capability should be provided for column pins. Totem pole configuration helps for a faster discharge of keypad capacitance when all columns need to be quickly brought to a "1" during the scan routine. With this configuration, a time delay between the scanning of two subsequent columns is reduced.

25.3 Memory Map and Register Definition

The KPP module contains four registers. Section 25.3.3, "Register Descriptions" provides detailed descriptions of the KPP registers.

25.3.1 KPP Memory Map

Table 25-2 shows the KPP memory map.

Table 25-2. KPP Memory Map

Address	Use	Access	Reset Value	Section/Page
0x1000_8000 (KPCR)	Keypad Control Register	R/W	0x0000	25.3.3.1/25-5
0x1000_8002 (KPSR)	Keypad Status Register	R/W	0x0000	25.3.3.2/25-5
0x1000_8004 (KDDR)	Keypad Data Direction Register	R/W	0x0000	25.3.3.3/25-7
0x1000_8006 (KPDR)	Keypad Data Register	R/W	0x— — — —	25.3.3.4/25-8

25.3.2 Register Summary

Figure 25-2 shows the key to the register fields, and Table 25-3 shows the register figure conventions.

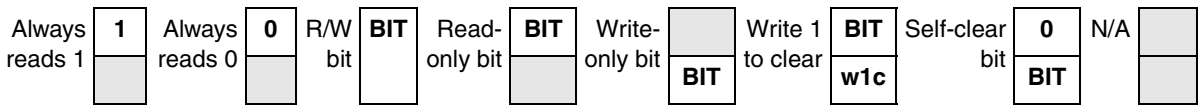


Figure 25-2. Key to Register Fields

Table 25-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 25-4 shows the KPP register summary.

Table 25-4. KPP Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000_8000 (KPCR)	R	KCO7	KCO6	KCO5	KCO4	KCO3	KCO2	KCO1	KCO0	KRE7	KRE6	KRE5	KRE4	KRE3	KRE2	KRE1	KRE0
	W																
0x1000_8002 (KPSR)	R	0	0	0	0	0	KPP_EN	KRIE	KDIE	0	0	0	0	0	0	KPKR	KPKD
	W													KRSS	KDSC	w1c	w1c
0x1000_8004 (KDDR)	R	KCDD7	KCDD6	KCDD5	KCDD4	KCDD3	KCDD2	KCDD1	KCDD0	KRDD7	KRDD6	KRDD5	KRDD4	KRDD3	KRDD2	KRDD1	KRDD0
	W																
0x1000_8006 (KPDR)	R	KCD7	KCD6	KCD5	KCD4	KCD3	KCD2	KCD1	KCD0	KRD7	KRD6	KRD5	KRD4	KRD3	KRD2	KRD1	KRD0
	W																

25.3.3 Register Descriptions

This section consists of register descriptions. Each register is listed in the order of its address.

25.3.3.1 Keypad Control Register (KPCR)

The Keypad Control Register determines which of the eight possible column strobes are to be open drain when configured as outputs, and which of the eight row sense lines are considered in generating an interrupt to the core.

It is up to the programmer to ensure that pins being used for functions other than the keypad are properly disabled. The KPCR register is byte- or halfword–addressable.

Figure 25-3 shows the valid bits in the KPCR register, and Table 25-5 provides its field descriptions.

0x1000_8000 (KPCR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KCO7	KCO6	KCO5	KCO4	KCO3	KCO2	KCO1	KCO0	KRE7	KRE6	KRE5	KRE4	KRE3	KRE2	KRE1	KRE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-3. KPCR Register

Table 25-5. Keypad Control Register Field Descriptions

Field	Description
15–8 KCO	Keypad Column Strobe Open-Drain Enable. Setting a column open-drain enable bit (KCO7–KCO0) disables the pull-up driver on that pin. Clearing the bit allows the pin to drive to the high state. This bit has no effect when the pin is configured as an input. 0 Column strobe output is totem pole drive. 1 Column strobe output is open drain. **Note:** Configuration of external port control logic (for example, GPIO) should be done properly so that the KPP module controls an open-drain enable of the pin.
7–0 KCO	Keypad Row Enable. Setting a row enable control bit in this register enables the corresponding row line to participate in interrupt generation. Likewise, clearing a bit disables that row from being used to generate an interrupt. This register is cleared by a reset, disabling all rows. The row-enable logic is independent of the programmed direction of the pin. Writing a “0” to the data register of the pins configured as outputs will cause a keypad interrupt to be generated if the row enable associated with that bit is set. 0 Row is not included in the keypad key press detect. 1 Row is included in the keypad key press detect.

25.3.3.2 Keypad Status Register (KPSR)

The Keypad Status Register reflects the state of the key press detect circuit. The KPSR register is byte- or halfword–addressable.

Figure 25-4 shows the KPSR register, and Table 25-6 provides its field descriptions.

0x1000_8002 (KPSR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	KPP_EN	KRIE	KDIE	0	0	0	0	0	0	KPKR	KPKD
W													KRSS	KDSC	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-4. KPSR Register

Table 25-6. Keypad Status Register Field Descriptions

Field	Description
15–11	Reserved
10 KPP_EN	Keypad Clock Gating Enable. The signal generated using this bit can be used by the ‘chip clock-control module’ to gate the module’s high frequency clock for register access and synchronization. Output of this bit is not used anywhere inside KPP module. 0 Disable high frequency clock to keypad module 1 Enable high frequency clock to keypad module
9 KRIE	Keypad Release Interrupt Enable. The software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0 No interrupt request is generated when KPKR is set. 1 An interrupt request is generated when KPKR is set.
8 KDIE	Keypad Key Depress Interrupt Enable. Software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice-versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0 No interrupt request is generated when KPKD is set. 1 An interrupt request is generated when KPKD is set.
7–4	Reserved, should be cleared
3 KRSS	Key Release Synchronizer Set. Self-clear bit. The Key release synchronizer is set by writing a logic one into this bit. Reads return a value of “0”. 0 No effect 1 Set bits which sets keypad release synchronizer chain
2 KDSC	Key Depress Synchronizer Clear. Self-clear bit. The Key depress synchronizer is cleared by writing a logic “1” into this bit. Reads return a value of “0”. 0 No effect 1 Set bits that clear the keypad depress synchronizer chain

Table 25-6. Keypad Status Register Field Descriptions (continued)

Field	Description
1 KPKR	Keypad Key Release. The keypad key release (KPKR) status bit is set when all enabled rows are detected high after synchronization (the KPKR status bit will be set when cleared by a reset). The KPKR bit may be used to generate a maskable key release interrupt. The key release synchronizer may be set high by software after scanning the keypad to ensure a known state. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by software prior to the system exiting the state it represents. 0 No key release is detected. 1 All keys have been released. Reset value of register is “0” as long as reset is asserted. However when reset is de-asserted, the value of the register depends upon the external row pins and can become “1”.
0 KPKD	Keypad Key Depress. The keypad key depress (KPKD) status bit is set when one or more enabled rows are detected low after synchronization. The KPKD status bit remains set until cleared by the software. The KPKD bit may be used to generate a maskable key depress interrupt. If desired, the software may clear the key press synchronizer chain to allow a repeated interrupt to be generated while a key remains pressed. In this case, a new interrupt will be generated after the synchronizer delay (4 cycles of the 32 KHz clock) elapses if a key remains pressed. This functionality can be used to detect a long key press. This allows detection of additional key presses of the same key or other keys. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by the software prior to the system exiting the state it represents. 0 No key presses have been detected. 1 A key has been depressed.

25.3.3.3 Keypad Data Direction Register (KDDR)

The bits in the KDDR control the direction of the keypad port pins. The upper eight bits in the register affect the pins designated as column strobes, while the lower eight bits affect the row sense pins. Setting any bit in this register configures the corresponding pin as an output. Clearing any bit in this register configures the corresponding port pin as an input. For the Keypad Row DDR, an internal pull-up is enabled if the corresponding bit is clear. This register is cleared by a reset, configuring all pins as inputs. The KDDR register is byte- or halfword–addressable.

NOTE

When a pin is used as row pin for keypad purposes, all corresponding pull-ups should be enabled at the upper level (for example, IOMUX) when the bit in row DDR is cleared.

Figure 25-5 shows the valid bits in the KDDR register, and Table 25-7 provides its field descriptions.

0x1000_8004 (KDDR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KCD D7	KCD D6	KCD D5	KCD D4	KCD D3	KCD D2	KCD D1	KCD D0	KRD D7	KRD D6	KRD D5	KRD D4	KRD D3	KRD D2	KRD D1	KRD D0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-5. KDDR Register

Table 25-7. Keypad Data Direction Register Field Descriptions

Field	Description
15–8 KCDD	Keypad Column Data Direction Register. Setting any bit configures the corresponding pin as an output. 0 COLn pin is configured as an input. 1 COLn[1] pin is configured as an output.
7–0 KRDD	Keypad Row Data Direction. Setting any bit configures the corresponding pin as an output. 0 ROWn pin configured as an input. 1 ROWn[2] pin configured as an output.

[1] n=7–0
[2] n=7–0

25.3.3.4 Keypad Data Register (KPDR)

This 16-bit register is used to access the column and row data. Data written to this register is stored in an internal latch, and for each pin configured as an output, the stored data is driven onto the pin. A read of this register returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register.

The KPDR register is byte- or halfword–addressable. This register is not initialized by a reset. Valid data should be written to this register before any bits are configured as outputs.

Figure 25-6 shows the KPDR register, and Table 25-8 provides its field descriptions.

0x1000_8006 (KPDR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KCD7	KCD6	KCD5	KCD4	KCD3	KCD2	KCD1	KCD0	KRD7	KRD6	KRD5	KRD4	KRD3	KRD2	KRD1	KRD0
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 25-6. KPDR Register

Table 25-8. Keypad Data Register Field Descriptions

Field	Description
15–8 KCD	Keypad Column Data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write “0” from/to column ports 1 Read/Write “1” from/to column ports
7–0 KRD	Keypad Row Data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write “0” from/to row ports 1 Read/Write “1” from/to row ports

25.4 Functional Description

The Keypad Port (KPP) is designed to simplify the software task of scanning a keypad matrix. With appropriate software support and matrix organization, the KPP is capable of detecting, debouncing, and decoding one or more keys pressed simultaneously on the keypad.

Logic in the KPP is capable of detecting a key press even while the processor is in one of the low power standby modes providing that a 32 KHz clock is on. The KPP may generate a CPU interrupt any time a key press or key release is detected. This interrupt is capable of forcing the processor out of a low power mode.

25.4.1 Keypad Matrix Construction

The KPP is designed to interface to a keypad matrix, which shorts the intersecting row and column lines together whenever a key is depressed. The interface is not optimized for any other switch configuration.

25.4.2 Keypad Port Configuration

The software must initialize the KPP for the size of the keypad matrix. Pins connected to the keypad columns should be configured as open-drain outputs. Pins connected to the keypad rows should be configured as inputs. On-chip, pull-up resistors should be implemented for active keypad rows.

In addition to enabled row inputs in the Keypad Control register, corresponding interrupt (depress or/and release) must also be enabled to generate an interrupt.

Discrete switches that are not part of the matrix may be connected to any unused row inputs. The second terminal of the discrete switch is connected to ground. The hardware detects closures of these switches without the need for software polling.

25.4.3 Keypad Matrix Scanning

Keypad scanning is performed by a software loop that walks a zero across each of the keypad columns, reading the value on the rows at each step. The process is repeated several times in succession, with the results of each pass optionally compared to those from the previous pass. When several (3 or 4) consecutive scans yield the same key closures, a valid key press has been detected. Software then can decode exactly which switch was depressed and pass the value up to the next higher software layer.

The basic debouncing period, which must be defined in the software routine, may be controlled with an internal timer. The basic period is the period between the scan of two consecutive columns, so the debouncing time between two consecutive scans of the whole matrix shall be the number of columns multiplied by the basic period.

25.4.4 Keypad Standby

There is no need for the CPU to continually scan the keypad. Between key presses, the keypad can be left in a state that requires no software intervention until the next key press is detected. To place the keypad in a standby state, software should write all column outputs low. Row inputs are left enabled. At this point,

the CPU can attend to other tasks or revert to a low power standby mode. The KPP will interrupt the CPU if any key is pressed.

Upon receiving a keypad interrupt, the CPU should set all the column strobes high, and begin a normal keypad scanning routine to determine which key was pressed. It is important that open-drain drivers be used when scanning to prevent a possible DC path between power and ground through two or more switches.

25.4.5 Glitch Suppression on Keypad Inputs

A glitch suppression circuit qualifies the keypad inputs to prevent noise from inadvertently interrupting the CPU. The circuit is a 4-state synchronizer clocked from a 32 KHz clock source. This clock must continue to run in any low power mode where the keypad is a wake-up source, as the CPU interrupt is generated from the synchronized input. An interrupt is not generated until all four synchronizer stages have latched a valid key assertion. This guarantees the filtering out of any noise less than three clock periods (for 32 KHz clock: 93.75 µs) in duration. Noise filtering of the duration between three to four clock periods (for the 32 KHz clock: between 93.75 µs and 125 µs) cannot be guaranteed. The interrupt output is latched in an S-R latch and remains asserted until cleared by the software. The Set input of the latch is rising-edge clocked. See Figure 25-7.

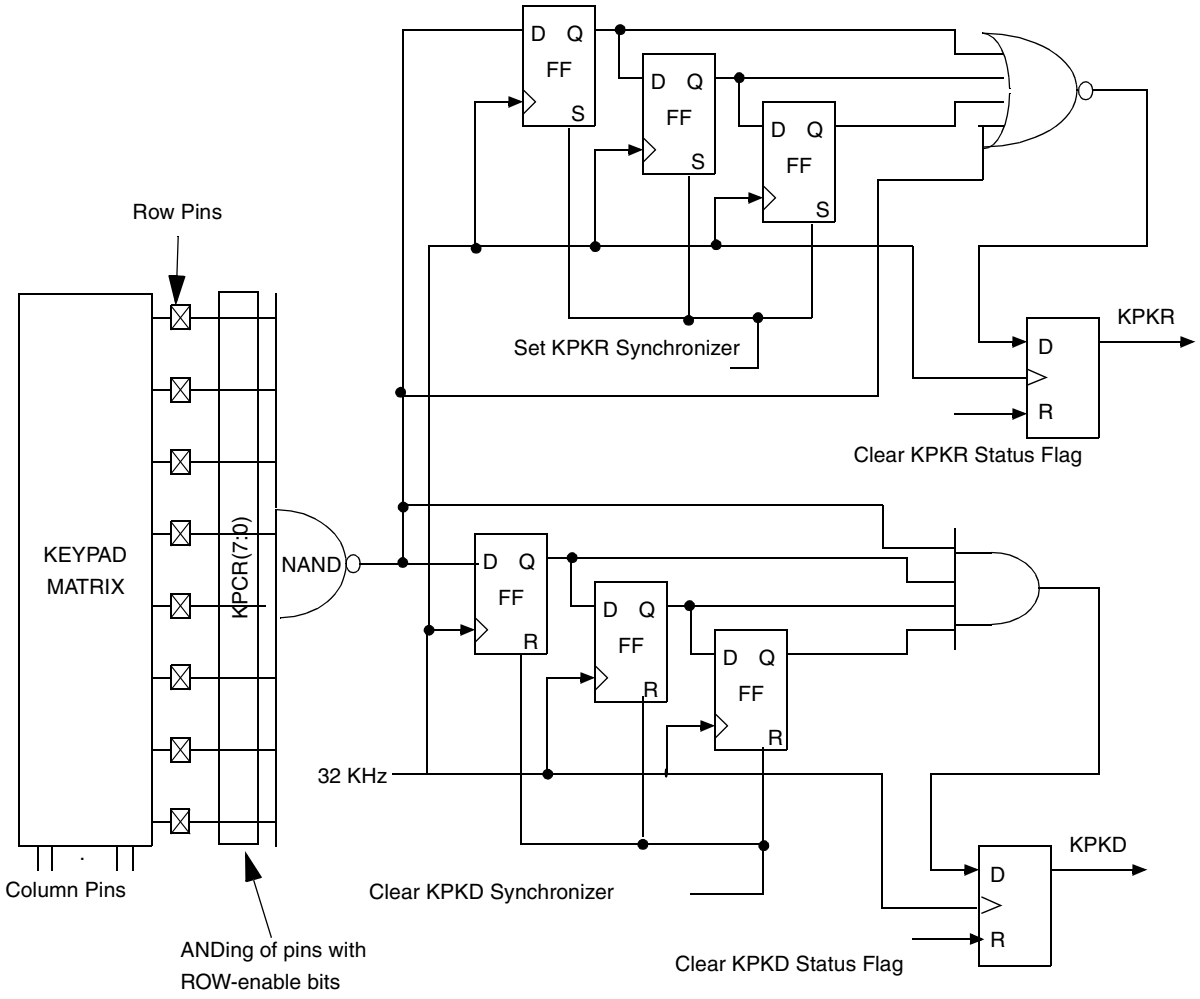


Figure 25-7. Keypad Synchronizer Functional Diagram

25.4.6 Multiple Key Closures

Using the key press and Key release interrupts, the software can detect multiple keys or achieve n key rollover. The key scanning routine can be programmed accordingly. Refer to Section 25.5, “Initialization/Application Information” for more information.

See Figure 25-5 and Figure 25-9 for illustrations of the interfacing of a 2-contact keypad matrix with the KPP controller. With proper enabling of row lines and the performing scan-routine, multiple key presses can be detected. When keys present on the same row are pressed, corresponding row lines (multiple lines) become low when the column is driven low during a scan-routine. By reading the data-register, pressed keys can be detected. Similarly, when keys present on same row line are pressed, the corresponding row line (only one line) becomes low when logic “0” is driven on the column line during a scan-routine.

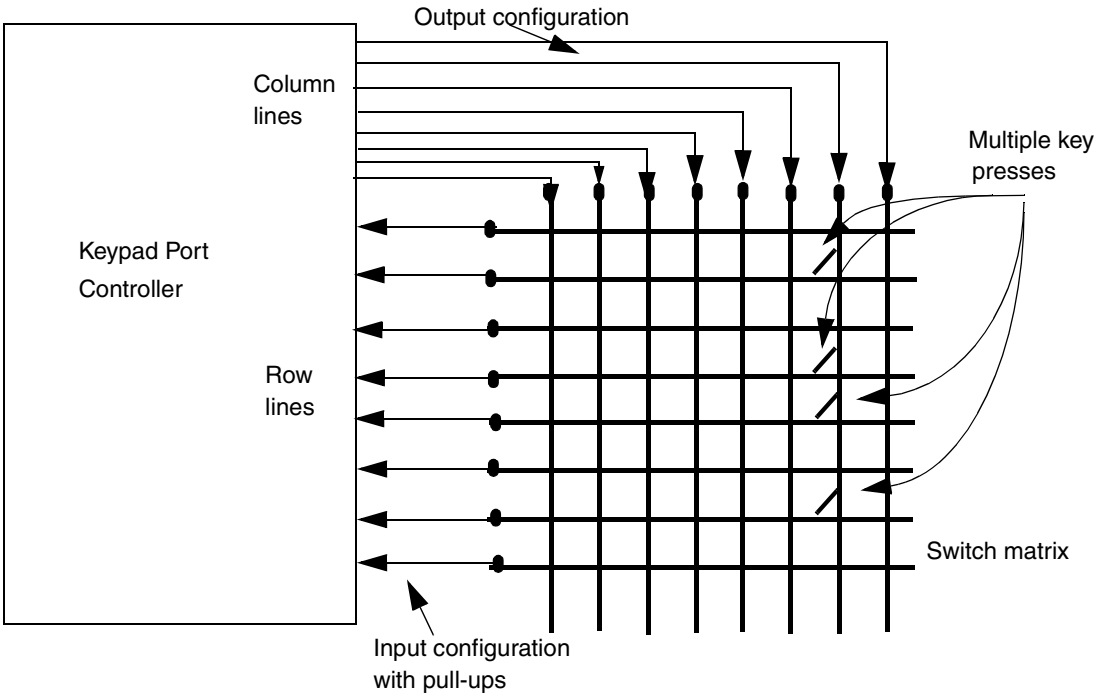


Figure 25-8. Multiple Key Presses on Same Column Line (Simplified View)

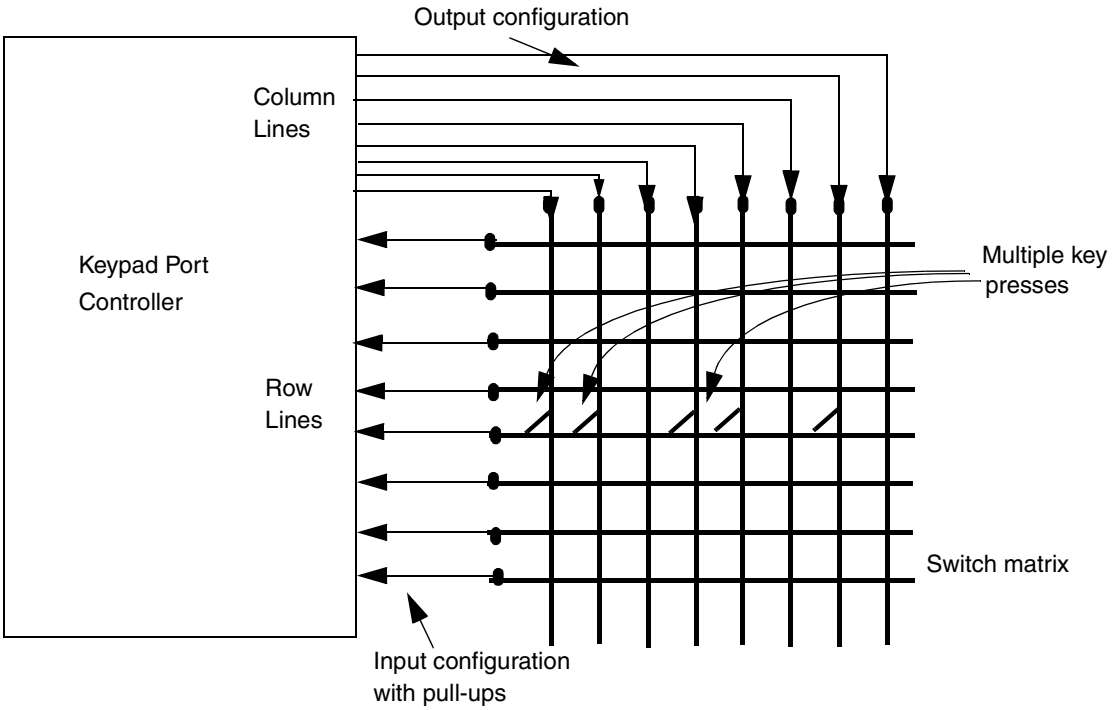


Figure 25-9. Multiple Key Presses on Same Row Line (Simplified View)

NOTE

An n key rollover is a technique the system uses to recognize the order in which keys are pressed.

25.4.6.1 Ghost Key Problem and Correction

The KPP module detects if one or multiple keys are pressed or released. In the case where a simple keypad matrix with two-contact switches is used, there is a chance of “ghost” key detection when three or more keys are pressed. This is a limitation imposed by such a keypad matrix. As can be seen in Figure 25-10, three keys pressed simultaneously can cause a short between the column currently “scanned” by the software and another column. Depending on the location of the third key pressed, a “ghost” key press may be detected.

However, this can be corrected by using a keypad matrix that provides “ghost” key protection. Such a matrix implements a one-way “diode” at all keypad points between rows and columns. This way, the multiple pressing of three keys will not cause a short at a fourth key (see Figure 25-11).

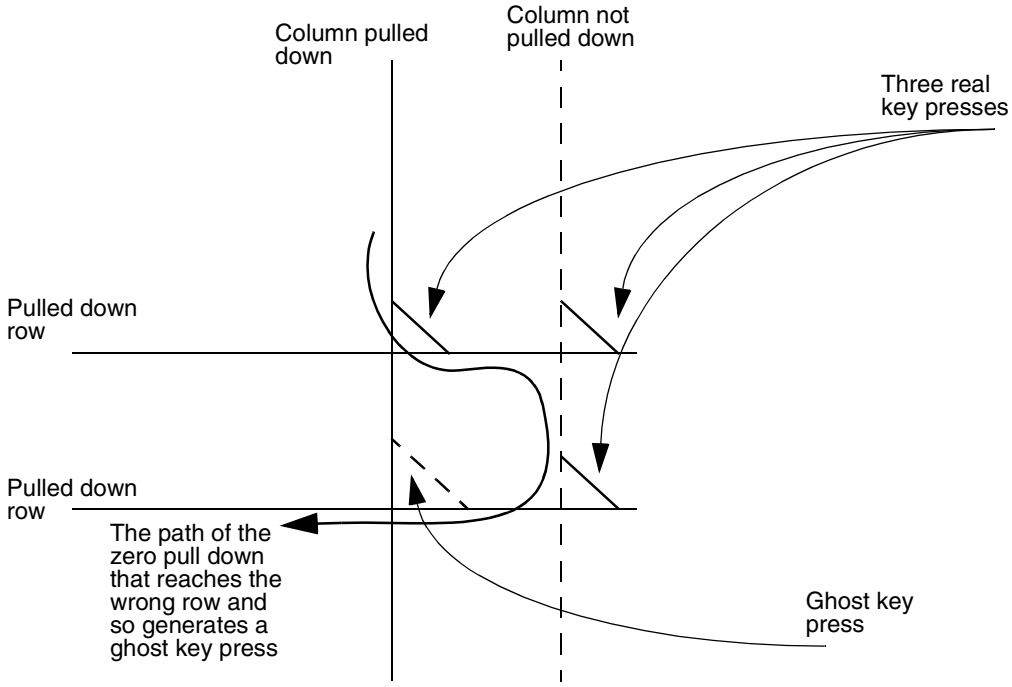


Figure 25-10. Decoding Wrong Three-Key Presses

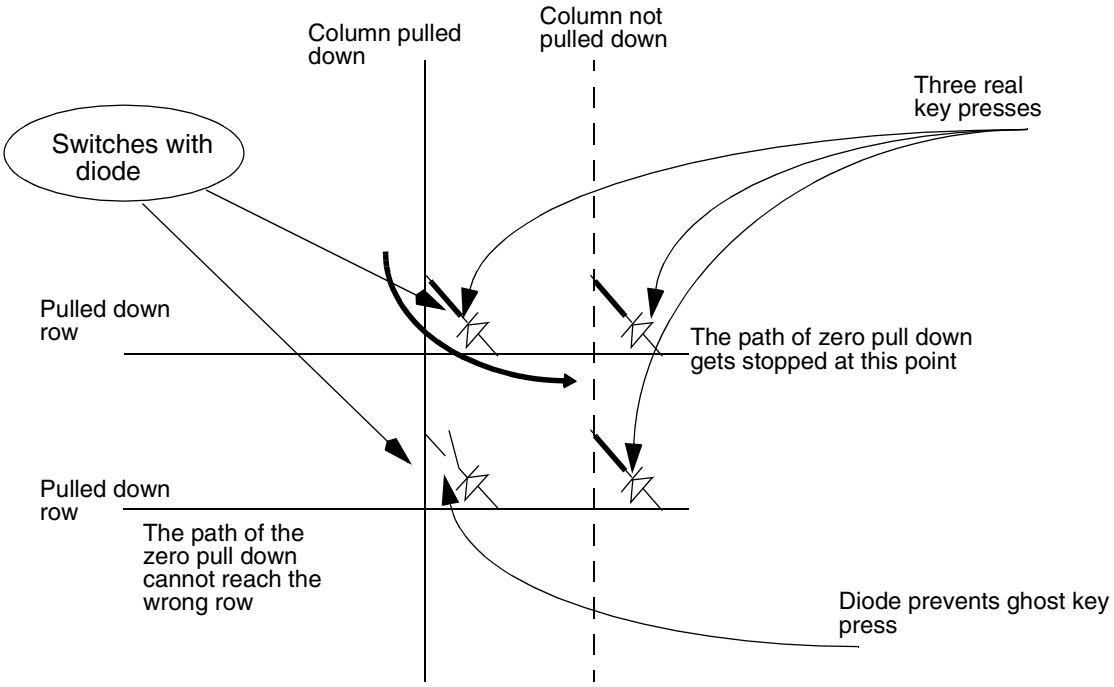


Figure 25-11. Matrix with "Ghost" Key Protections

25.4.7 3-Point Contact Keys Support

The KPP module supports interfacing to a matrix consisting of 3-point contact keys. As shown in Figure 25-12, two points of such a key are connected to keypad lines, while a third point is connected to ground (low logic). The keypad lines should be configured as input and a pull-up should to be present on these lines. When such a key is pressed, corresponding keypad lines go low and an interrupt is generated. There is no need to perform a scanning routine for identification of pressed key as it can be done by reading the keypad data-register. A limitation with such a matrix is that for every key at least one keypad row line should be used.

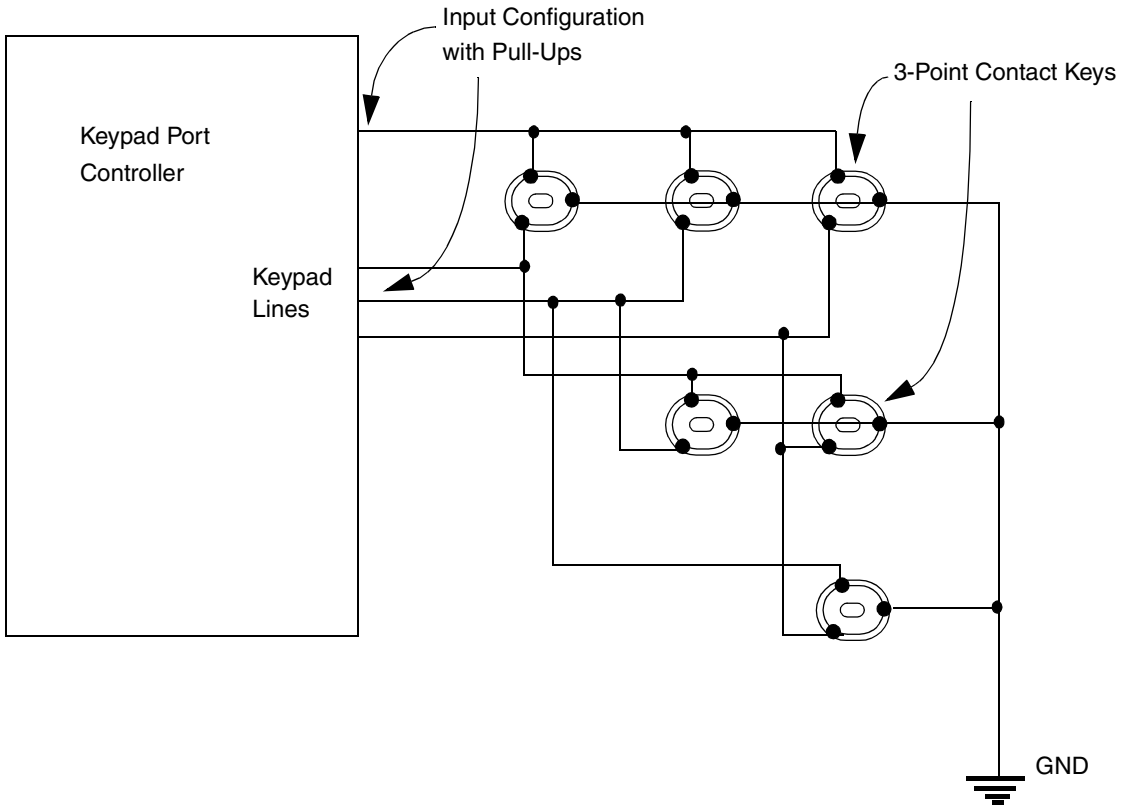


Figure 25-12. KPP Interface with 3-Point Contact Key Matrix (Simplified View)

25.5 Initialization/Application Information

This section provides initialization and application information.

25.5.1 Typical Keypad Configuration and Scanning Sequence

Perform the following steps to configure the keypad:

1. Enable the number of rows in the keypad (KPCR[7:0]).
2. Write 0s to KPDR[15:8].
3. Configure the keypad columns as open-drain (KPCR[15:8]).
4. Configure columns as output and rows as input (KDDR[15:0]).
5. Clear the KPKD Status Flag and Synchronizer chain.
6. Set the KDIE control bit, and clear the KRIE control bit (avoid false release events).

(The system is now in standby mode, and awaiting a key press.)

25.5.2 Key Press Interrupt Scanning Sequence

Perform the following steps to perform a keypad scanning routine:

1. Disable both (depress and release) keypad interrupts.

2. Write 1s to KPDR[15:8], setting column data to 1s.
3. Configure columns as totem pole outputs (for quick discharging of keypad capacitance).
4. Configure columns as open-drain.
5. Write a single column to 0, and other columns to 1.
6. Sample row inputs and save data. Multiple key presses can be detected on a single column.
7. Repeat Steps 2–6 for remaining columns.
8. Return all columns to 0 in preparation for standby mode.
9. Clear KPKD and KPKR status bit(s) by writing to a “1”; set the KPKR synchronizer chain by writing a “1” to the KRSS register; and clear the KPKD synchronizer chain by writing a “1” to the KDSC register.
10. Re-enable the appropriate keypad interrupt(s) so that the KDIE detects a key hold condition, or the KRIE detects a key-release event.

25.5.3 Additional Comments

The order of key press detection can be done in software only. Therefore, the software may need to run the scan routines at very short intervals of time per the application’s demands.

For the keys that require a very precise order (such as game keys), individual GPIO pins may be more useful.

Chapter 26
Memory Stick Host Controller (MSHC)

The Memory Stick Host Controller (MSHC) consists of two sub modules—the MSHC gasket and the Sony Memory Stick Host Controller (SMSC). The SMSC module, which is the actual memory stick host controller, is compatible with Sony Memory Stick Ver. 1.x and Memory Stick Pro. The gasket connects the AIPI IP bus to the SMSC interface to allow IP transfers. The MSHC is placed between the AIPI and the Sony Memory Stick to support data transfer from the chip to the MS.

The MSHC top level block diagram with input and output signals is shown in Figure 26-1.

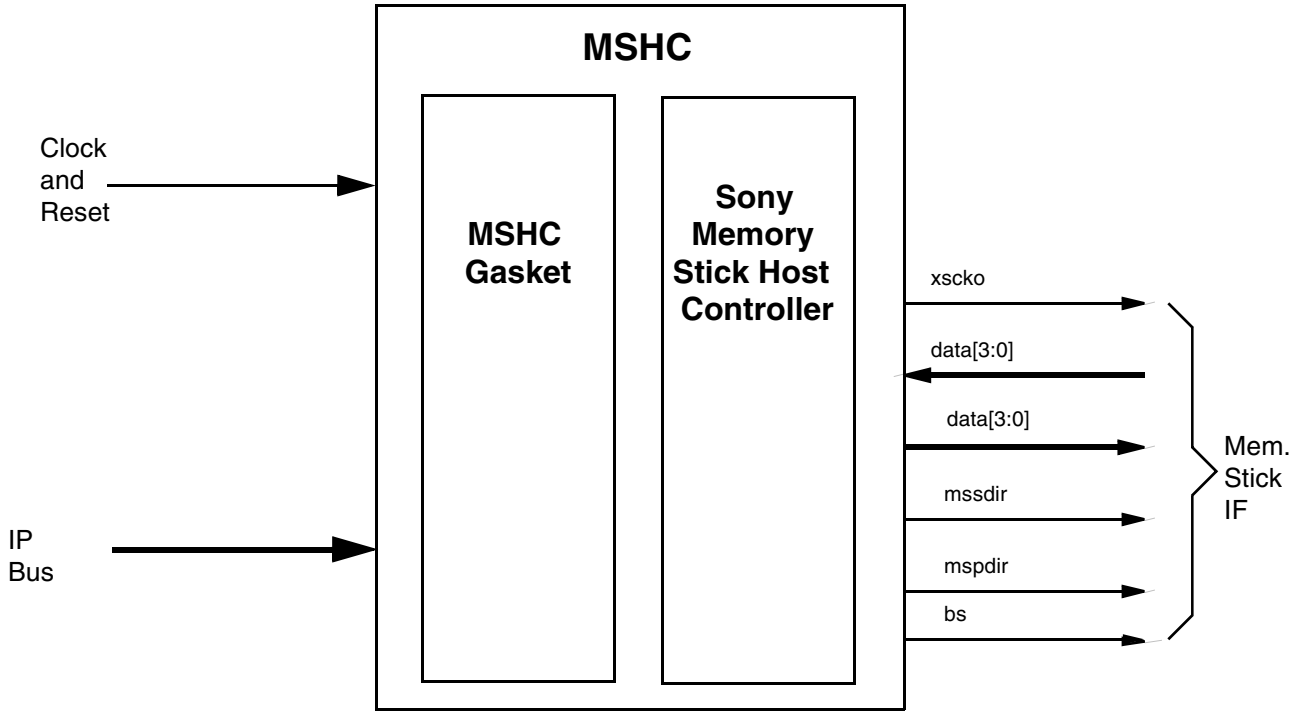


Figure 26-1. Memory Stick Controller Block Diagram

26.1 Overview

This chapter describes the MSHC gasket module in detail. All details regarding the SMSC module can be found separately in *Memory Stick/Memory Stick Pro Host Controller IP Specification 1.3*.

26.1.1 Features

The MSHC includes the following features:

- A gasket between IP bus and SMSC
  - IP bus interface transfer functionality as slave
  - Three internal registers (timeout, interrupt status/clear, and interrupt enable register)
  - Gasket interrupt for transfer errors, or wait timeout
  - Timeout function for abnormal transfer wait states
  - Fixed 32-bit data bus
  - Little endian to IP data bus and big endian to SMSC data bus
- SMSC to communicate to the Sony Memory Stick
  - Four internal registers structured in 64-bit format
  - FIFO (4 x 64-bit)
  - Interrupt after Memory Stick communication completes
  - DMA in dual address mode (Note: SMSC supports single address mode as well)
- Test mode and DFT implementation

26.1.2 Modes of Operation

The MSHC gasket has a reduced IP interface and supports the IP bus read/write transfers that include a back-to-back read or write. DMA transfers also take place via the IP interface.

A transfer can be initiated by the DMA or the host (through AIPI) in response to an MSHC DMA request or interrupt. The SMSC has two DMA address modes, a single address mode and a dual address mode.

The MSHC is set to dual address mode for transfers with the DMA. In dual address mode, when the MSHC requests a transfer with the DMA request (XDRQ), the DMA will initiate a transfer to the MSHC.

The MSHC still has the external memory ports and the DMA acknowledge input (XDAK) even though the single address mode is not used in some chip.

26.2 External Signal Description

26.2.1 Overview

The MSHC signals are listed in Table 26-1.

Table 26-1. MSHC I/O Signals

Name	Port	Active	Function
data[3:0]	Input	—	MSHC data input from MS
data[3:0]	Output	—	MSHC data output to MS
ipp_do_ms_mdo[63:0]	Output	—	MSHC external memory output data

Table 26-1. MSHC I/O Signals (continued)

Name	Port	Active	Function
ipp_do_ms_mdsl	Output	High	MSHC external memory data select. If set to one, external memory is selected.
ipp_do_ms_mreq	Output	High	MSHC external memory data request
ipp_do_ms_mrws	Output	—	MSHC external memory read/write select (write if one)
mspdir	Output	—	MSHC parallel data direction (write transfer—MSHC to MS—if one, or read transfer if zero)
mssdir	Output	—	MSHC serial data direction (write transfer—MSHC to MS—if one, or read transfer if zero)
xscko	Output	—	MSHC serial clock output

26.2.1.1 Memory Stick Interface

The MSHC module supports Memory Stick Pro, which means a data transfer can be done in serial mode or parallel mode.

26.3 Memory Map and Register Definition

The MSHC has seven internal registers; four registers in the SMSC and three registers in the gasket.

The gasket internal registers do not require any extra addresses as they share the address space for the SMSC internal registers. IP bus accesses are 32-bit, but all SMSC registers (except the data register) use only the upper 16 bits, so the gasket can utilize the lower 8 bits.

The MSHC internal registers are mapped to 64-bit structure, so the required address signal is [4:3].

Due to this address sharing scheme, the host must keep the SMSC register values as necessary while updating the gasket internal registers.

Table 26-2 shows the memory map for the MSHC module.

Table 26-2. MSHC Memory Map

<table>
<tr><th rowspan="2">Memory Map</th><th rowspan="2">addr[4:3]</th><th colspan="8">Data Bit</th></tr>
<tr><th>[63:57]</th><th>[56:48]</th><th>[47:40]</th><th>[39:32]</th><th>[31:24]</th><th>[23:16]</th><th>[15:8]</th><th>[7:0]</th></tr>
<tr><td>‘h00</td><td>0</td><td colspan="2">SMSC command register</td><td>Gasket timeout register</td><td colspan="4">Not used</td></tr>
<tr><td>‘h08</td><td>1</td><td colspan="4">SMSC data register</td><td colspan="4">Not used</td></tr>
</table>

Table 26-2. MSHC Memory Map (continued)

Memory Map	addr[4:3]	Data Bit							
		[63:57]	[56:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
'h10	2	SMSC status register		Gasket interrupt status/ clear register	Not used				
'h18	3	SMSC system register		Gasket interrupt enable register	Not used				

26.3.1 Register Descriptions

The MSHC module uses big endian with 64-bit data while the chip system uses the little endian with 32-bit data. Therefore, the data must be assigned as per Table 26-3.

Table 26-3. MSHC Data Endianism and Connection to IP Bus

Bus	Endian	Connection							
MSHC data	Big	[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
IP bus data	Little	[7:0]	[15:8]	[23:16]	[31:24]	—	—	—	—

26.3.1.1 SMSC Registers

The SMSC command register contains the MS transfer protocol command (TPC) and transfer data size. It must be set by either the host or DMA before starting a transfer.

The SMSC data register is used to access the SMSC internal FIFO, which is of size 4 x 64-bit. It takes 8 32-bit transfers to fill or empty the FIFO.

The SMSC status register reflects SMSC status, such as FIFO status, ready flag for transfers and CRC error flag. The host needs to read this register in order to start and manage the transfer. This register is only readable through the IP bus.

The SMSC system register has a user command for SMSC modes and options that are required for transfer and communication with the MS. This register must be set before starting a transfer and should not be changed during the communication with the MS.

More details can be found in *Memory Stick/ Memory Stick PRO Host Controller IP Specification, Ver. 1.3*.

26.3.1.2 Gasket Register

26.3.1.2.1 Gasket Timeout Register

This register is readable or writable. The value after synchronous or asynchronous reset is 0. This register gives a mechanism to time-out if long wait states are encountered.

Once the FIFO becomes full while the IP bus has more data to be written, the gasket asserts the wait signal until the FIFO space is available. The gasket will also start to decrement an internal counter from the timeout value for wait, TOVW[7:0]. When the counter reaches zero, the ips_xfr_wait signal is deasserted and an interrupt is generated If the wait state is cleared before the timeout counter reaches zero the suspended transfer will be resumed.

When the TOVW[7:0] is set to zero the gasket will not produce wait states if the FIFO is full/empty, but will generate an interrupt.

If the gasket interrupt enable bits - INTEN_WFUL and INTEN_REMP - are disabled, the timeout counter has no functionality (it is effectively set to infinity). Once the wait state is entered the MSHC will stay in wait until either the AIPI disables the module (using ips_module_en) or the FIFO becomes available for another transfer.

Table 26-4. Timeout Register

Bit	47	46	45	44	43	42	41	40
Name	TOVW[7:0]							
Reset value	0							
R/W	R/W							

26.3.1.2.2 Gasket Interrupt Status/Clear Register

This register is readable or writable, and is reset to 0 by asynchronous or synchronous reset. Bits should be written as 1 to clear an interrupt. The unused bits are read as zero. The SMSC status register, which has the same address as this register, is a read-only register.

The interrupt status/clear register reflects the transfer status. It is used for the gasket to decide if it supports a current transfer, and it can also be used for the gasket interrupt generation.

The gasket provides four different interrupts, which are all directly related to the IP bus transfer. Once an interrupts occurs the host needs to check this register to determine which interrupt occurred and also to clear the interrupt (by writing 1 to the register bit).

Table 26-5. Interrupt Status/Clear Register

Bit	47	46	45	44	43	42	41	40
Name	IDA	IXFR	—	—	WFUL	REMP	—	—
Reset value	0	0	—	—	0	0	—	—
R/W	R/W	R/W	R/–	R/–	R/W	R/W	R/-	R/-

- IDA (Illegal data access)

  This bit is asserted if a transfer is not 32-bit, that is, if the ips_byte inputs are not all asserted. It is always readable and can be cleared (if the interrupt enable bit INTEN_IDA is enabled) by writing to this register with ips_wdata[23] set to one.

  Any illegal data access always asserts IDA, but will not generate a gasket interrupt if the interrupt enable bit INTEN_IDA is disabled.

- IXFR (Illegal transfer)

  This bit is asserted if a data transfer direction is not consistent with the SMSC FIFO data direction. It is always readable and can be cleared (if the interrupt enable bit INTEN_IXFR is enabled) by writing to this register with ips_wdata[22] set to one.

  Any illegal data transfer always asserts IXFR, but will not generate a gasket interrupt if the interrupt enable bit INTEN_IXFR is disabled.

- WFUL (Write to FIFO when full)

  This bit is asserted if the SMSC FIFO is full and the current write transfer is not finished. It is always readable and can be cleared (if the interrupt enable bit INTEN_WFUL is set) by writing to this register with ips_wdata[19] set to one.

  The gasket asserts WFUL when a data write transfer is attempted while the FIFO is full. The update time will be different depending on whether INTEN_WFUL is enabled and the value of TOVW[7:0]. If INTEN_WFUL is disabled WFUL will be updated without the timeout delay. The gasket uses this bit to generate a transfer wait state as well as the interrupt.

- REMP (Read from FIFO when empty)

  This bit is asserted if the SMSC FIFO is empty and the current read transfer is not finished. It is always readable and can be cleared (if the interrupt enable bit INTEN_REMP is set) by writing to this register with ips_wdata[18] set to one.

  The gasket asserts REMP when a normal data read transfer is attempted while the FIFO is empty. The version number read after a reset also asserts REMP as the FIFO is empty at this time. The update time will be different depending on whether INTEN_REMP is enabled and the value of TOVW[7:0]. If INTEN_REMP is disabled REMP will be updated without the timeout delay. The gasket uses this bit to generate a transfer wait state as well as the interrupt.

26.3.1.2.3 Gasket Interrupt Enable Register

This register is readable or writable, and is reset to 0 by asynchronous or synchronous reset.

This register is used to enable the gasket interrupts. The gasket interrupts are disabled by the default. To enable the gasket interrupt the relevant bit should be set to 1.

Table 26-6. Interrupt enable register

Bit	47	46	45	44	43	42	41	40
Name	INTEN_IDA	INTEN_IXFR	—	—	INTEN_WFUL	INTEN_REMP	—	—
Reset value	0	0	—	—	0	0	—	—
R/W	R/W	R/W	R/–	R/–	R/W	R/W	R/-	R/-

- INTEN_IDA

  Illegal data access interrupt enable bit. If it is one it will enable the gasket interrupt for an illegal data access.

- INTEN_IXFR

  Illegal transfer interrupt enable bit. If it is set to one it will enable the gasket interrupt for an illegal data transfer.
- INTEN_WFUL

  Interrupt enable bit for a write transfer to fifo with full. If it is set to one it will enable the gasket interrupt and the wait timeout function.
- INTEN_REMP

  Interrupt enable bit for a read transfer from fifo with empty. If it is set to one it will enable the gasket interrupt and the wait timeout function.

26.4 Functional Description

26.4.1 Sony Memory Stick Controller (SMSC)

The details of the SMSC functionality can be found in *Memory Stick/ Memory Stick PRO Host Controller IP Specification, Ver. 1.3*.

26.4.2 MSHC Gasket

This section describes MSHC clocks, MS interface, the gasket state-machine, IP bus error and wait conditions, the gasket interrupt, and IP bus transfer.

26.4.2.1 Resetting and Clocking

The MSHC uses one asynchronous reset and one synchronous reset to reset gasket internal registers.

The asynchronous reset is the green-line hardware asynchronous reset that is active low while the synchronous reset is a soft reset from the MSHC system register RST bit that is active high. Once the soft reset is asserted the SMSC will automatically clear the RST bit in the system register after initializing all internal registers. During the soft reset period the behavior of any IP Bus transactions is undefined.

MSHC has several clocks, as shown in Figure 26-2.

The MSHC has three clock inputs ipg_clk_s is directly connected to the gasket, and the others to SMSC through muxing logic. The MSHC also has one clock output that is inverted in the gasket.

The muxing logic allows the MSHC to have only two clock domains during test mode. ipg_clk_s in the gasket and HCKI in the SMSC are connected to one test clock, and all other clocks (SCKI, XSCKI, MSCKI, and XMSCKI) are connected to a different test clock. The test clocks are generated inside the CRM.

In normal operation, XSCKI is an inverted version of SCKI and MSCKI is an inverted version of XMSCKI.

The gasket uses the ipg_clk_s for all logic that is synchronous to the IP interface signals. This clock is only active when the gasket module enable signal is asserted.

HCKI in the SMSC is the main clock for most of the internal registers and the FIFO. The MS has a slower clock speed and is asynchronous to HCKI, so XSCKO is generated by the SMSC and is output to provide the MS clock.

All logic shown in Figure 26-2 is implemented in the gasket.

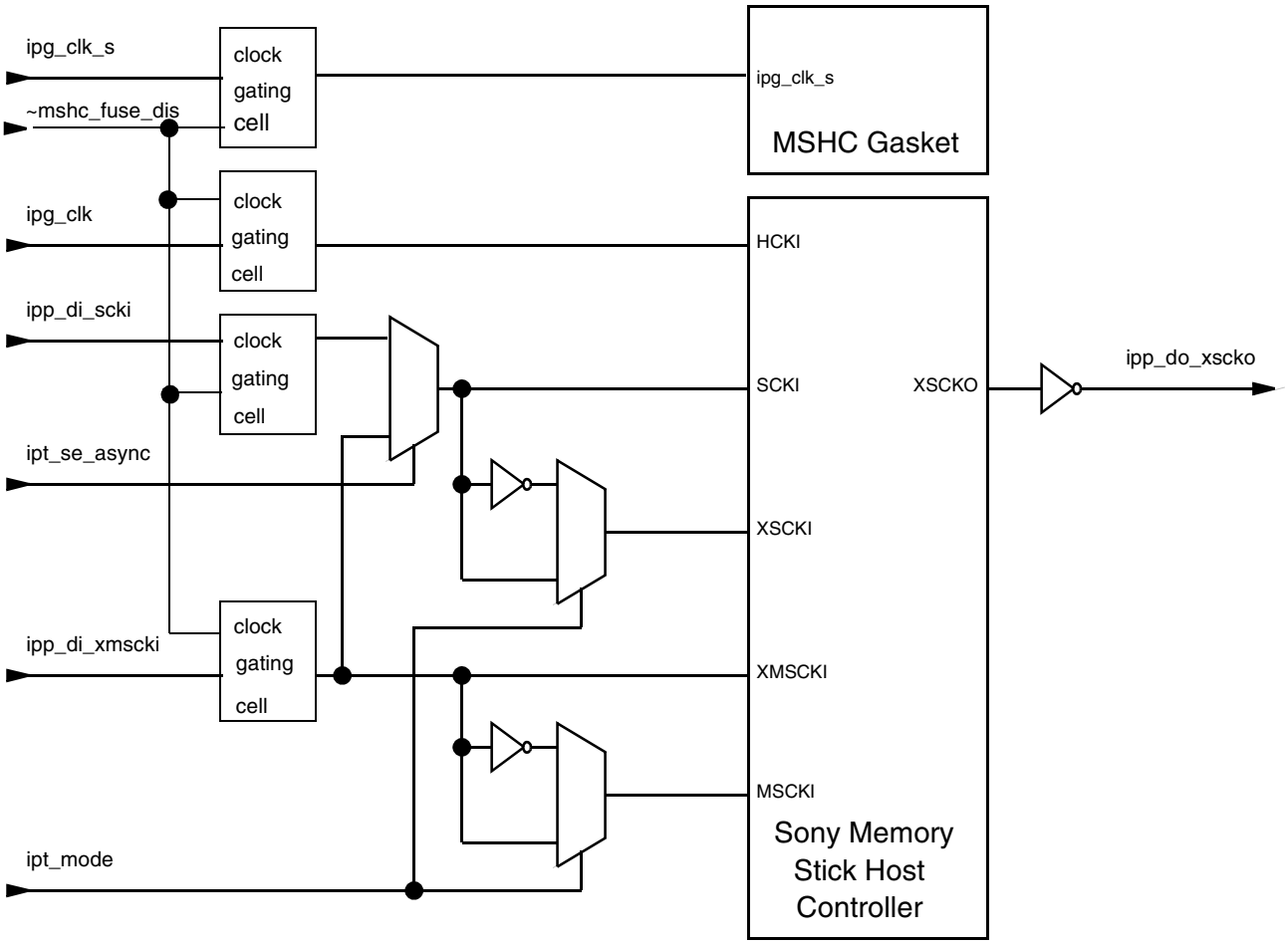


Figure 26-2. MSHC Clock Structure

26.4.2.2 Memory Stick Interface

The gasket includes logic to generate the IO signals for the MS. This logic is included in the gasket so that it does not need to be provided at the chip level. The multiplex logic and signal connections included in the gasket are shown in the Table 26-3.

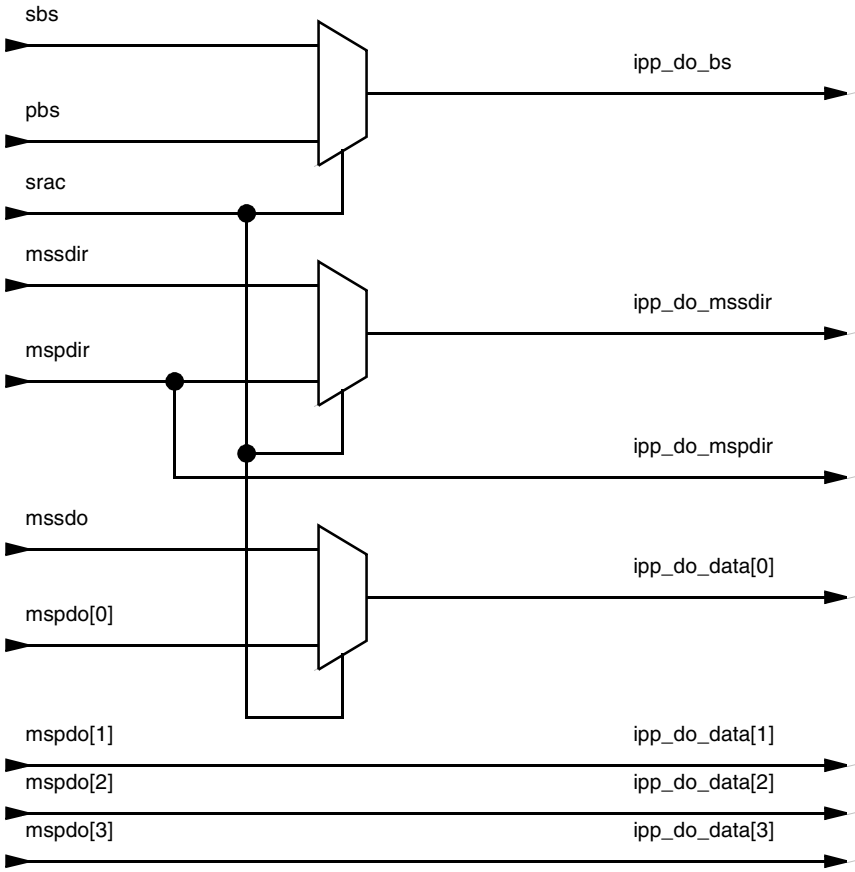


Figure 26-3. Gasket Memory Interface Logic

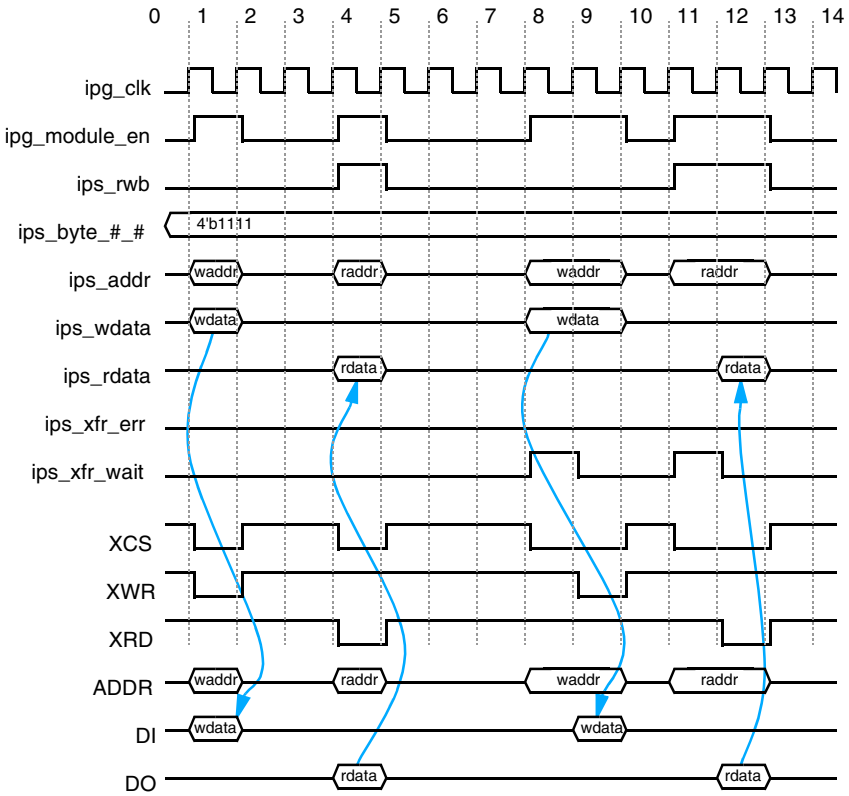


Figure 26-4. Basic Read/Write Timing Diagram

26.4.2.2.1 Back-to-Back Transfer

The Figure 26-5 shows the back-to-back timing for IP bus and the SMSC interface.

In the timing diagram there are three write transfers (cycles 1–6) and two read transfers (cycles 8–11).

The IP bus sends back-to-back transfers and the gasket inserts one cycle wait states for each data transfer. This allows the SMSC read/write signal to be asserted for one HCKI cycle.

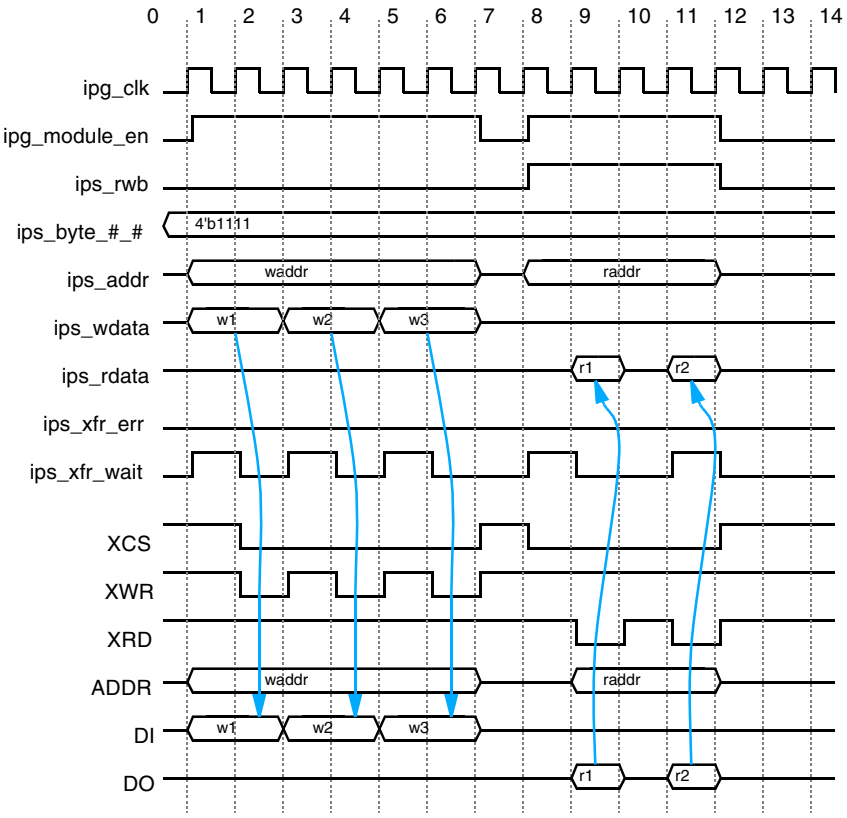


Figure 26-5. Back-to-Back Write/Read Timing Diagram

26.4.2.3 Transfer Error

When the gasket detects a transfer error it generates an interrupt. The IP bus transfer error interrupt is an optional feature that can be enabled by the gasket interrupt enable register.

Two transfer error conditions are:

- Illegal data access
- Illegal transfer

The data access error shows if the IP bus transfer is not 32-bit. The gasket checks all four byte enable signals and if any one of these becomes zero during a transfer the gasket will set the interrupt status/clear bit, IDA, and generate the interrupt.

The illegal transfer error shows whether the IP bus data transfer direction is correct. The SMSC system register includes the fifo data direction bit that must be set before the transfer is started. If this bit is set to one, the data direction is from AIPI to MSHC. Then the next IP bus transfer must be a write. If the data direction and data transfer direction are not consistent, the gasket updates the interrupt status/clear bit, IXFR, to one and generates the interrupt.

Note that this error condition is applied only to transfers to the SMSC data register.

26.4.2.4 Transfer Wait Condition

In normal transfers, a wait state may be inserted for one clock cycle for each 32-bit transfer, because of the read/write timing requirements for the SMSC when the IP bus is dealing with back-to-back type transfers.

If the transfer initiator ensures a maximum transfer burst to fill the SMSC FIFO there will be no multiple wait cycles. If IP bus performs a large transfer burst to the MSHC, the SMSC FIFO will fill due to the slow transfer speed between the SMSC and the MS. In this case, the gasket inserts wait states until the FIFO is available for further transfers, that is, one space in the FIFO is available. At this time the gasket will end the wait state and resume the next transfer. The behavior is similar for read transfers when the FIFO is empty.

There are two conditions for multiple cycles of wait states:

- Write transfers on the IP bus with the SMSC FIFO full
- Read transfers on IP bus with the SMSC FIFO empty

Exceptional operations can also cause a long wait state. For instance, if one 32-bit data write transfer is performed with the SMSC FIFO empty, the empty flag will remain set. Thus, if the following transfer is a read operation it will not be processed due to the asserted state of the FIFO empty flag. Therefore the gasket will stay in a wait state as long as the read transfer request remains pending on the IP bus. A minimum of two 32-bit write transfers are required to deassert the empty flag. The transfer initiator must make sure that a minimum of two write transfers are made in order to read back the data written to the FIFO.

Another unexpected long wait state may happen if the communication between the MSHC and the MS is abnormally terminated. To prevent this behavior, the gasket provides a wait timeout function, with which a timeout value can be used to disable the wait state and generate an interrupt if the transfer wait state goes for too long. This option is available when the interrupt enable bits, INTEN_WFUL and INTEN_REMP, are enabled as described in Section 26.3, “Memory Map and Register Definition.”

The timeout counter has no functionality if the INTEN_WFUL and INTEN_REMP interrupts are disabled. In this case the transfer initiator is responsible for disabling the module to end a long wait state.

26.4.2.5 Gasket Interrupt

The gasket interrupts are generated from transfer error and wait conditions and is ORed with the SMSC interrupt. Table 26-6 is the flow chart that shows how the gasket inserts wait states and generates an interrupt.

The flow chart also shows how the gasket timeout function performs. The interrupt enable bits INTEN_WFUL and INTEN_REMP disable all the gasket status/clear and timeout register functions.

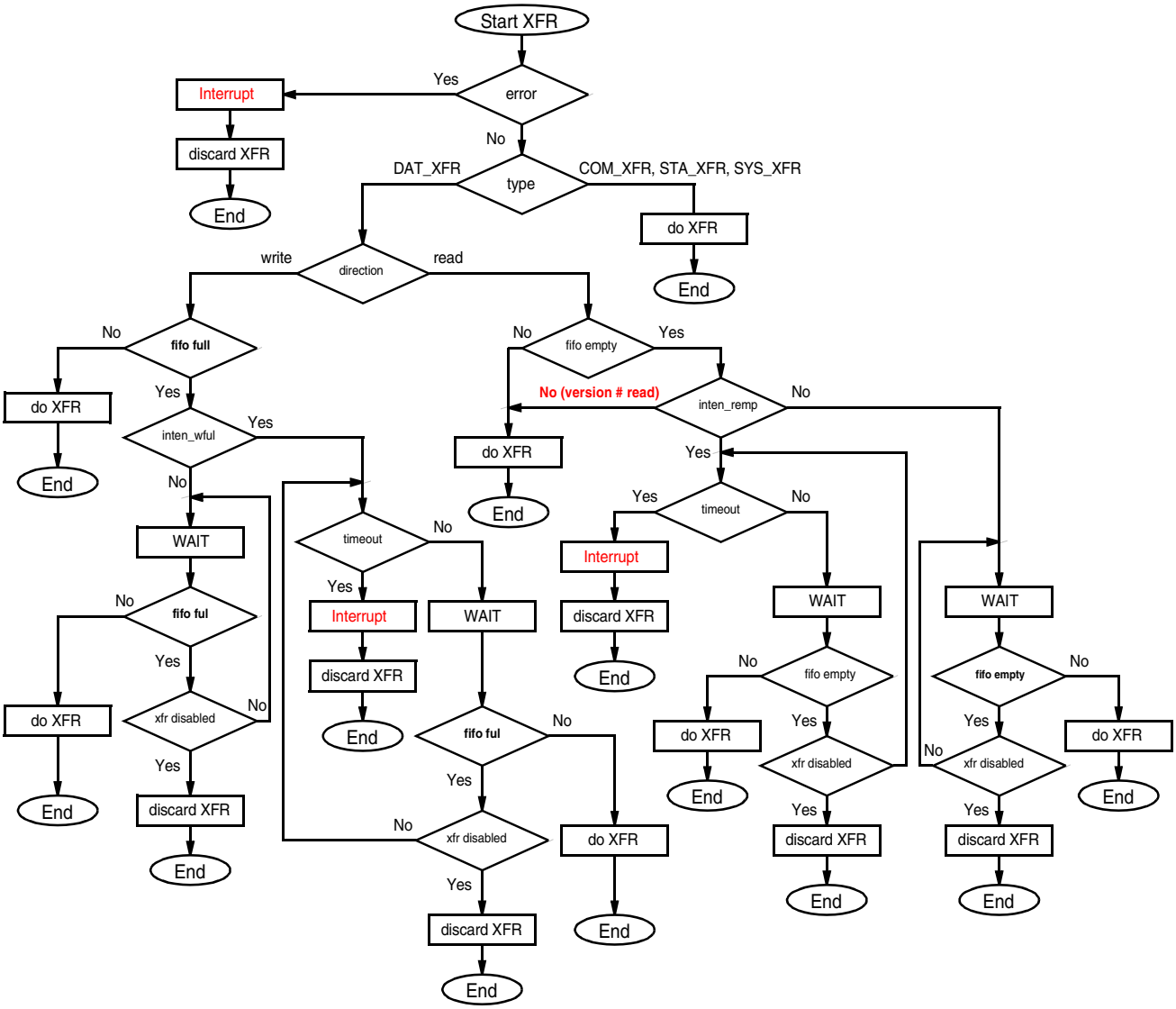


Figure 26-6. Gasket Interrupt Generation

Chapter 27
Secured Digital Host Controller (SDHC)

The MultiMediaCard (MMC) is a universal low-cost data storage and communication medium that is designed to cover a wide area of applications, such as electronic toys, organizers, PDAs, and smart phones. The MMC communication is based on an advanced 7-pin serial bus designed to operate in a low-voltage range.

The Secure Digital Card (SD), is an evolution of MMC technology, with two additional pins in the form factor. It is specifically designed to meet the security, capacity, performance, and environment requirement inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment, and data transfer protocol are forward-compatible with the MultiMediaCard with some additions. Under SD protocol, it can be categorized into Memory card, I/O card, and Combo card (has both memory and I/O functionality). The memory card invokes a copyright-protection mechanism that complies with the security of the SDMI standard, is faster, and provides the capability for a higher memory capacity. The I/O card provides high-speed data I/O with low-power consumption for mobile electronic devices.

The Security Digital Host Controller (SDHC) integrates both MMC support along with SD memory and I/O functions, including SD memory and I/O combo card.

See the block diagram of SDHC in Figure 27-1.

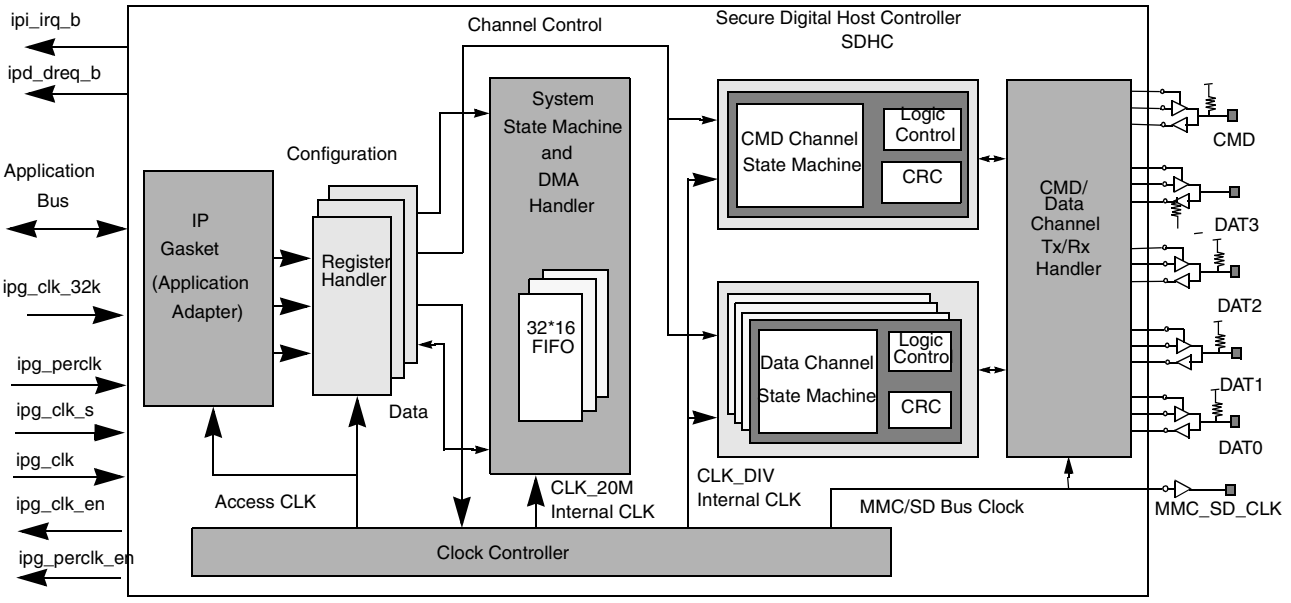


Figure 27-1. Secure Digital Host Controller Block Diagram

See Figure 27-2 for the illustration of system interconnection with the SDHC module.

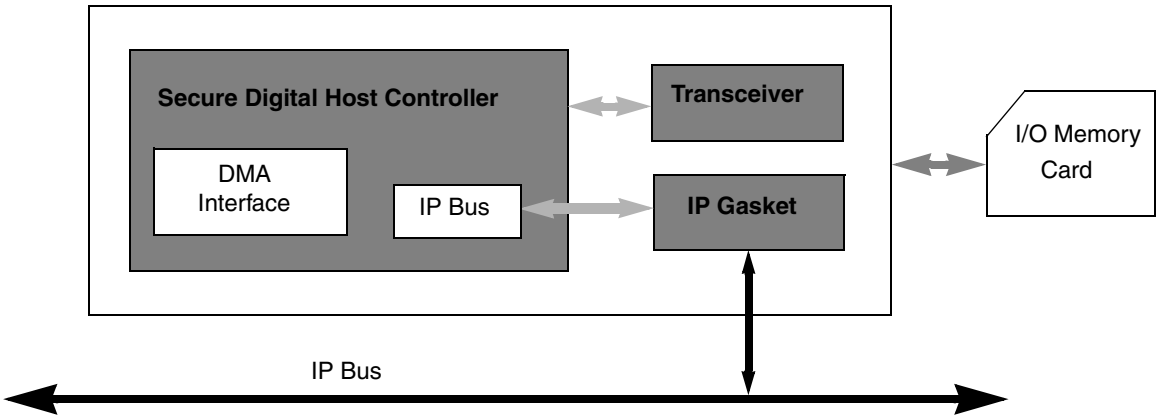


Figure 27-2. System Interconnection with the Secure Digital Host Controller

27.1 Overview

The Security Digital Host Controller (SDHC) controls the MMC, SD memory, and I/O cards by sending commands to cards and performing data accesses to/from the cards. Refer to Section 27.4, "Functional Description" for a detailed description.

27.1.1 Features

The features of the Secure Digital Host Controller module include the following:

- Full compatibility with the MMC system specification, version 3.2
- Compatibility with the SD Memory Card Specification 1.01, and SD I/O Specification 1.1 with 1/4 channel(s)
- 100 Mbps maximum data rate in 4-bit mode, SD bus clock up to 25 MHz
- Built-in programmable frequency counter for SDHC bus
- Maskable hardware interrupt for SDIO Interrupt, Internal status and FIFO status
- Built-in dual 16 x 32-bit data FIFO buffer
- Plug and play (PnP) support
- Single/Multi-block access to the card, including erase operation
- Multi-SD function support, including multiple I/O and combined I/O and memory
- Up to seven I/O functions, plus one memory supported on single SD I/O card (Combo card)
- IRQ supported enable card to interrupt host
- Support of SDIO interrupt detection during 1/4-bit access
- Support of SDIO Read/Wait and suspend/resume operation
- Controller core—Freescale Semiconductor IP bus compatible
- Block-based data transfer between MMC card and SDHC (stream mode not supported)
- Block length of data transfer capability between host and card of approximately 1 to 2048 bytes

27.2 External Signal Description

The SDHC has a six-chip I/O. The MMC_SD_CLK is an internally generated clock used by the MMC/SD card. The CMD I/O is used to send commands and receive responses from the card. Four data lines, DAT3–DAT0, are used to perform data transfers between the host controller and the card.

See Table 27-1 for the signal properties of the I/Os.

Table 27-1. Signal Properties

Name	Port	Function	Reset State	Pull up
MMC_SD_CLK	O	Clock for MMC/SD/SDIO card	0	
CMD	I/O	CMD line connect to card	1	Pull up
DAT3	I/O	Card Detect in Power up Data line in 4-bit mode Not used in 1-bit mode	0	Pull down DAT3 if need card detection through this bit, otherwise pull up
DAT2	I/O	Data line or Read wait in 4-bit mode Read wait in 1-bit mode	1	Pull up
DAT1	I/O	Data line or interrupt in 4-bit mode Interrupt in 1-bit mode	1	Pull up
DAT0	I/O	Data line both in 1-bit and 4-bit mode	1	Pull up

27.3 Memory Map and Register Definition

SDHC contains 16 32-bit registers. Section 27.3.3, "Register Descriptions" provides the detailed descriptions for all of the SDHC registers. All the registers can be accessed only in 32-bit sizes. Byte/Half-word access is not allowed.

27.3.1 Memory Map

Table 27-3 shows the SDHC memory map. The SDHC memory map space is 4 Kbytes. Only address offsets from 0 x 00 to 0 x 44 are implemented. The address space above the offset of 0 x 44 is reserved. For write access to the reserved address region, the access is ignored by SDHC. For read access to the reserved address region, 0 x 0 will be returned on the IP Bus. The user should not access the reserved region to ensure compatibility with possible future revisions of this module.

Table 27-3. SDHC Memory Map

Address	Register	Access	Reset Value	Section/Page
0x1001_3000 (STR_STP_CLK1) 0x1001_4000 (STR_STP_CLK2)	SDHC Clock Control register	R/W	0x0000_0000	27.3.3.1/27-8
0x1001_3004 (STATUS1) 0x1001_4004 (STATUS2)	SDHC Status register	R	0x3000_0000	27.3.3.2/27-9
0x1001_3008 (CLK_RATE1) 0x1001_4008 (CLK_RATE2)	SDHC Card Clock Rate register	R/W	0x0000_0008	27.3.3.3/27-13
0x1001_300C (CMD_DAT_CONT1)	SDHC Command Data Control register	R/W	0x0000_0000	27.3.3.4/27-15

Table 27-3. SDHC Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x1001_3010 (RES_TO1) 0x1001_4010 (RES_TO2)	SDHC Response Time-out register	R/W	0x0000_0040	27.3.3.5/27-17
0x1001_3014 (READ_TO1) 0x1001_4014 (READ_TO2)	SDHC Read Time-out register	R/W	0x0000_FFFF	27.3.3.6/27-18
0x1001_3018 (BLK_LEN1) 0x1001_4018 (BLK_LEN2)	SDHC Block Length register	R/W	0x0000_0000	27.3.3.7/27-19
0x1001_301C (NOB1) 0x1001_401C (NOB2)	SDHC Number of Block register	R/W	0x0000_0000	27.3.3.8/27-20
0x1001_3020 (REV_NO1) 0x1001_4020 (REV_NO2)	SDHC Revision Number register	R	0x0000_0400	27.3.3.9/27-21
0x1001_3024 (INT_CNTR1) 0x1001_4024 (INT_CNTR2)	SDHC Interrupt Control register	R/W	0x0000_0000	27.3.3.10/27-22
0x1001_3028 (CMD1) 0x1001_4028 (CMD2)	SDHC Command Number register	R/W	0x0000_0000	27.3.3.11/27-26
0x1001_302C (ARG1) 0x1001_402C (ARG2)	SDHC Command Argument register	R/W	0x0000_0000	27.3.3.12/27-27
0x1001_3034 (RES_FIFO1) 0x1001_4034 (RES_FIFO2)	SDHC Command Response FIFO Access register	R	0x0000_0000	27.3.3.13/27-28
0x1001_3038 (BUFFER_ACCESS1) 0x1001_4038 (BUFFER_ACCESS2)	SDHC Data Buffer Access register	R/W	0x0000_0000	27.3.3.14/27-29

Note: The following addresses are reserved:
0x1001_3030
0x1001_330C
0x1001_4030
0x1001_430C

27.3.2 Register Summary

Figure 27-3 shows the key to the register fields and Table 27-4 shows the register figure conventions.

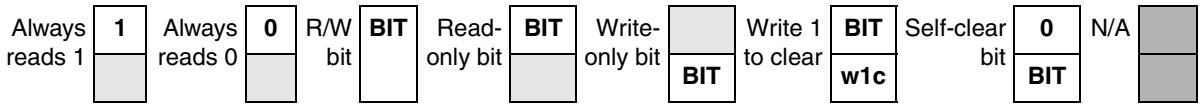


Figure 27-3. Key to Register Fields

Table 27-4. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.

Table 27-4. Register Figure Conventions (continued)

Convention	Description
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 27-5 shows the SDHC register summary.

Table 27-5. SDHC Register Summary

Name		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
0x1001_3000 (STR_STP_CLK1) 0x1001_4000 (STR_STP_CLK2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	slfclr	0	START_CLK	STOP_CLK
	W													SDHC Reset			

Table 27-5. SDHC Register Summary (continued)

Name		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
0x1001_3004 (STATUS1) 0x1001_4004 (STATUS2)	R	CARD_INSERTION	CARD_REMOVAL	YBUF_EMPTY	XBUF_EMPTY	YBUF_FULL	XBUF_FULL	BUF_UND_RUN	BUF_OVFL	0	0	0	0	0	0	0	0
	W	w1c	w1c														
	R	0	SDIO_INT_ACTIVE	END_CMD_RESP	WRITE_OP_DONE	READ_TRANS_DONE	WR_CRC_ERR_CODE		CARD_BUS_CLK_RUN	BUF_READ_RDY	BUF_WR_RDY	RESP_CRC_ERR	0	READ_CRC_ERR	WRITE_CRC_ERR	TIME_OUT_RESP	TIME_OUT_READ
	W		w1c	w1c	w1c	w1c	w1c	w1c				w1c		w1c	w1c	w1c	w1c
0x1001_3008 (CLK_RATE1) 0x1001_4008 (CLK_RATE2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CLK_PRESCALER[15:4]												CLK_DIVIDER[3:0]			
	W																
0x1001_300C (CMD_DAT_CONT1) 0x1001_400C (CMD_DAT_CONT2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CMD_RESUME	0	0	CMD_RESP_LONG_OFF	STOP_READWAIT	START_READWAIT	BUS_WIDTH		INIT	0	0	WRITE_READ	DATA_ENABLE	FORMAT_OF_RESPONSE		
	W																
0x1001_3010 (RES_TO1) 0x1001_4010 (RES_TO2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	RESPONSE TIME OUT[7:0]							
	W																
0x1001_3014 (READ_TO1) 0x1001_4014 (READ_TO2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DATA_READ_TIME_OUT[15:0]															
	W																

Table 27-5. SDHC Register Summary (continued)

Name		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
0x1001_3018 (BLK_LEN1) 0x1001_4018 (BLK_LEN2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	BLOCK LENGTH[11:0]											
	W																
0x1001_301C (NOB1) 0x1001_401C (NOB2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	NOB[15:0]															
	W																
0x1001_3020 (REV_NO1) 0x1001_4020 (REV_NO2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	Revision Number[15:0]															
	W																
0x1001_3024 (INT_CNTR1) 0x1001_4024 (INT_CNTR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	SDIO_INT_WKP_EN	CARD_INSERT_WKP_EN	CARD_REMOVAL_WKP_EN
	W																
	R	CARD_INSERTION_EN	CARD_REMOVAL_EN	SDIO_IRQ_EN	DAT0_EN	0	0	0	0	0	0	0	BUF_READ_EN	BUF_WRITE_EN	END_CMD_RES	WRITE_OP_DONE	READ_OP_DONE
	W																
0x1001_3028 (CMD1) 0x1001_4028 (CMD2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	COMMAND NUMBER					
	W																
0x1001_302C (ARG1) 0x1001_402C (ARG2)	R	ARG[31:16]															
	W																
	R	ARG[15:0]															
	W																

Table 27-5. SDHC Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x1001_3034 (RES_FIFO1) 0x1001_4034 (RES_FIFO2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RESPONSE_CONTENT[15:0]															
	W																
0x1001_3038 (BUFFER_ACCESS1) 0x1001_4038 (BUFFER_ACCESS2)	R	FIFO CONTENT[31:16]															
	W																
	R	FIFO CONTENT[15:0]															
	W																

27.3.3 Register Descriptions

Many of SDHC registers have reserved bits. Reserved bits identified in the registers are read as zero and write to these bits are ignored. However, the user should write zeros to these bits to ensure compatibility with possible future revisions of this module.

27.3.3.1 SDHC Clock Control Register (STR_STP_CLK)

The SDHC Clock Control Register allows the user to reset the whole module and to enable or disable the MMC_SD_CLK to card.

See Figure 27-4 for an illustration of valid bits in the SDHC Clock Control Register and Table 27-6 for descriptions of the bit fields.

0x1001_3000 (STR_STP_CLK1)
0x1001_4000 (STR_STP_CLK2)

Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	slfclr	0	START_CLK	STOP_CLK
W													SDHC Reset			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-4. SDHC Clock Control Register

Table 27-6. SDHC Clock Control Register Field Descriptions

Field	Description
31–4	Reserved
3 SDHC Reset	SDHC Reset. Writes to the SDHC reset bit triggers the reset logic inside the SDHC. Reads from this bit always return '0'. To reduce the power consumption, the clock to the reset logic in SDHC is off in normal operation. When there is one access to this register, the clock will be enabled for one cycle. To complete the entire reset period, it will need at least 8 clock pulses to finish the reset cycle. To reset the SDHC module, it is recommended to write this register with value 0x0008, followed by 0x0009, and then 0x0001 eight times. Refer to Section 27.5.2.2, "Reset" for detailed information on software reset. 0 No effect 1 Reset the SDHC module
2	Reserved
1 START_CLK	Start Clock. Writing a '1' to this bit starts the MMC_SD_CLK clock. Setting a value of 11 on Bits [1:0] of this register is not allowed. **Note:** The SDHC bus clock does not start immediately after writing to this bit. Polling needs to be done on the status CARD_BUS_CLK_RUN bit to ensure the SDHC clock is running. Refer to Example 27-1 for procedures to start the SD bus clock. 0 No effect 1 To start MMC/SD clock
0 STOP_CLK	Stop Clock. Stops the MMC_SD_CLK clock when the user writes a value of '1' to this bit. The MMC_SD_CLK should not be stopped by software during a transmission period. Setting a value of 11 on Bits [1:0] of this register is not allowed. **Note:** A transmission period is defined as the time from when a card data or access related command is submitted to the end of the access operation. **Note:** The SDHC bus clock does not stop immediately after writing to this bit. Polling needs to be done on the status CARD_BUS_CLK_RUN bit to ensure the SDHC clock is running. Refer to Example 27-1 for procedures to start the SD bus clock. 0 No effect 1 To stop the MMC/SD clock

27.3.3.2 SDHC Status Register (STATUS)

The read-only SDHC Status Register provides the programmer with information about the status of SDHC operations, application FIFO status, error conditions, and interrupt status.

There are eight interrupt status bits, which are bit-31 card insertion status bit, bit-30 card removal status bit, bit-14 SDIO card interrupt status bit, bit-13 end command and response status bit, bit-12 write operation done status bit, bit-11 read operation done status bit, bit-7 data buffer read ready status bit, and bit-6 data buffer write ready status bit. When the corresponding interrupt enable is enabled in SDHC interrupt control register for any of these interrupts, the SDHC will generate an interrupt request to the CPU. The user needs to clear the appropriate status bit to clear the corresponding interrupt. The interrupt status bits are cleared by using a write '1' to clear operation except for the data buffer ready status bits which can be cleared only by the read or write operation on the data buffer.

See Figure 27-5 for an illustration of valid bits in the SDHC Status Register and Table 27-7 for descriptions of the bit fields.

0x1001_3004 (STATUS1)
0x1001_4004 (STATUS2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CARD_INSERTION	CARD_REMOVAL	YBUF_EMPTY	XBUF_EMPTY	YBUF_FULL	XBUF_FULL	BUF_UND_RUN	BUF_OVFL	0	0	0	0	0	0	0	0
W	w1c	w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	SDIO_INT_ACTIVE	END_CMD_RESP	WRITE_OP_DONE	READ_TRANS_DONE	WR_CRC_ERR_CODE		CARD_BUS_CLK_RUN	BUF_READ_RDY	BUF_WR_RDY	RESP_CRC_ERR	0	READ_CRC_ERR	WRITE_CRC_ERR	TIME_OUT_RESP	TIME_OUT_READ
W		w1c	w1c	w1c	w1c	w1c	w1c				w1c		w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-5. SDHC Status Register

Table 27-7. SDHC Status Register Field Descriptions

Field	Description
31 CARD_INSERTION	Card Insertion. When this bit is set, the SDHC detects a value transition on the SD_DAT[3:0] from 0111 to 1111. This can be used to detect whether a card is inserted to the card socket based on SD_DAT[3] pull-up resistor of the card DAT3. When this bit is set, the SDHC will generate an interrupt request if the card detection interrupt is enabled. This bit is read-only and can be cleared by writing a '1' to this bit. 0 No card insertion detected 1 Card insertion detected based on logic level changed on SD_DAT[3]
30 CARD_REMOVAL	Card Removal. When this bit is set, the SDHC detects a logic transition on the SD_DAT[3:0] from 1111 to b0111. This can be used to detect whether a card is removed from the card socket based on the SD_DAT[3] pull-up resistor of the card DAT3. When this bit is set, SDHC will generate an interrupt request if the card removal interrupt is enabled. This bit is read-only and can be cleared by writing a '1' to it. The user needs to clear this bit to clear the interrupt request from SDHC when the card detection interrupt is enabled. 0 No card removal detected 1 Card removal detected based on logic level changed on SD_DAT[3]
29 YBUF_EMPTY	Y Data Buffer Empty. When this is set, it indicates that the Y data buffer is empty during a write transfer. This bit is automatically cleared when the first byte of data is moved into the FIFO. Refer to Section 27.4.1, "Data Buffers" for more information about the data buffers. 0 Y buffer is not empty. 1 Y buffer is empty.

Table 27-7. SDHC Status Register Field Descriptions (continued)

Field	Description
28 XBUF_EMPTY	X Data Buffer Empty. When this is set, it indicates that the X data buffer is empty during a write transfer. This bit is automatically cleared when the first byte of data is moved into the FIFO. Refer to Section 27.4.1, "Data Buffers" for more information about the data buffers. 0 X buffer is not empty. 1 X buffer is empty.
27 YBUF_FULL	Y Data Buffer Full. When this is set, it indicates that the Y data buffer is full during a read transfer. This bit is automatically cleared when the last byte of data is read out from the FIFO. Refer to Section 27.4.1, "Data Buffers" for more information about the data buffers. 0 Y buffer is not full. 1 Y buffer is full.
26 XBUF_FULL	X Data Buffer Empty. When this is set, it indicates that the X data buffer is full during a read transfer. This bit is automatically cleared when the last byte of data is read out from the FIFO. Refer to Section 27.4.1, "Data Buffers" for more information about the data buffers. 0 X buffer is not full. 1 X buffer is full.
25 BUF_UND_RUN	Buffer Underrun. When this is set, it indicates that both X and Y data buffers are empty during a write transfer. In this case, the card clock MMC_SD_CLK will be stopped automatically by hardware to wait for the DMA or CPU to put data into the buffers. An interrupt is triggered if the corresponding interrupt control bit is enabled. 0 No buffer underrun 1 Buffer underrun during a write operation
24 BUF_OVFL	Buffer Overflow. When this is set, it indicates that both data buffers are full during a read operation. In this case, the card clock MMC_SD_CLK will be stopped automatically by hardware to wait for the DMA or CPU to remove data out of one of the buffers. An interrupt is triggered if the corresponding interrupt control bit is enabled. Excess data will be ignored by the SDHC. 0 No buffer overflow 1 Buffer overflow during a read operation
23–15	N/A
14 SDIO_INT_ACTIVE	SDIO Interrupt Active. This indicates whether an interrupt from the SDIO card has been detected. When this bit is set, the SDHC generates an interrupt request if the SDIO interrupt is enabled. The user should clear the status to clear the interrupt request. A separate acknowledge command to the card may be required to clear the source of the SDIO interrupt. Writing a '1' to this bit clears it. 0 No interrupt detected 1 Interrupt detected using SDIO card bus
13 END_CMD_RESP	End Command Response. This indicates that a command was successfully transmitted to the card and the corresponding response stored in the Response FIFO. This occurs after each command operation. When this bit is set, the SDHC generates an interrupt request if the End_CMD_RESP interrupt is enabled. The user needs to clear this bit to negate the interrupt request. This bit is cleared by using a write '1' to clear operation. 0 Command not successful, incomplete, or not applicable (no response) 1 Command transmitted successfully (response received) **Note:** When this bit is set, the user also needs to check if the command send and response receive operation completed without error. The user also needs to check the RESP_CRC_ERR (Status[5]) and TIME_OUT_RESP(STATUS[1]) bits to determine if an error occurred.

Table 27-7. SDHC Status Register Field Descriptions (continued)

Field	Description
12 WRITE_OP_DONE	Write Operation Done. This indicates that a write operation has completed. The FLASH card needs extra idle time for write accesses. This requires the SDHC module to wait until the card writes the buffered data to the inner Flash memory. The SDHC module automatically detects the status. The WRITE_OP_DONE flag indicates the end of the write operation. When this bit is set, the pre-defined data bytes are written to the card. The user needs to send a STOP command to the card if the write command is a MMC/SD card write multi-block command. When this bit is set, SDHC generates an interrupt request if the WRITE_OP_DONE interrupt enable is enabled. The user needs to clear this bit to clear the interrupt. This is accomplished by writing '1' to this bit. 0 Write operation in progress or incomplete 1 Write operation complete **Note:** When this bit is set, the user also needs to check if the write operation completed without a CRC error. The user needs to check the WR_CRC_ERR_CODE[1:0] (Status[10:9]) and WRITE_CRC_ERR (STATUS[2]) bits to determine if an error has occurred.
11 READ_OP_DONE	Read Operation Done. The READ_OP_DONE status bit is activated at the end of a read operation, which means all the data is received from the card. When this bit is set, the pre-defined data bytes are read from the card or a READ TIMEOUT occurs. The software needs to send a STOP command to card if the read command is a MMC or SD card read multi-block command. This bit can be cleared by writing '1' to it. When this bit is set, SDHC generates an interrupt request if the READ_OP_DONE interrupt is enabled. The user needs to clear this bit to clear the interrupt request. Writing '1' to this bit clears the status. 0 Read operation in progress or incomplete 1 Read operation complete **Note:** When this bit is set, the user also needs to check if the read operation complete without error. The user needs to check the READ_CRC_ERR (Status[3])and TIME_OUT_READ(STATUS[0]) bits to determine if an error has occurred.
10–9 WR_CRC_ERROR_CODE	Write CRC Error Code. This indicates the CRC results from the card at the end of write operations. After receiving a block of data, the card checks the CRC bit and sends the CRC status. These two bits reflect the CRC status of the recent written data. If the card returns a negative CRC status, the data is not written to the card. These two bits can be cleared by writing a value of '11' to them. 00 No transmission error, CRC status is 010 (positive) 01 Transmission error, CRC status is 101 (negative) 10 No CRC response 11 Reserved **Note:** These bits have valid value only when the WRITE_CRC_ERR status bit (STATUS[2]) is set.
8 CARD_BUS_CLK_RUN	Card Bus Clock Run. This indicates whether the MMC_SD_CLK clock to the card is running. The clock rate setting and system configuration can be modified when the clock is turned off by setting the STOP_CLK bit in STR_STP_CLK Register. This bit can only be cleared by writing '1' to STOP_CLK bit in STR_STP_CLK clock control register to stop MMC_SD_CLK. 0 MMC/SD clock is stopped. 1 MMC/SD clock is running. **Note:** Polling needs to be done on this bit to assure the SDHC clock is running or stopped. Refer to Example 27-1.

Table 27-7. SDHC Status Register Field Descriptions (continued)

Field	Description
7 BUF_READ_READY	Buffer Read Ready. This status is set if a buffer (either X buffer or Y buffer) is full during read operations. An interrupt is triggered for non-DMA transfers if BUF_READ_EN is set, or a DMA request is asserted for DMA transfers. **Note:** The read_op_done status bit will be set once all the data is read from the card. At the end of read, the buffer read ready interrupt occurs the same time as the read done. The user can service the buffer read ready interrupt and then handle the read_op_done interrupt. 0 Not ready to read buffer 1 Ready to read buffer
6 BUF_WRITE_READY	Buffer Write Ready. This status is set if a buffer (either X buffer or Y buffer) is available during write operations. An interrupt is triggered for non-DMA transfers if the BUF_WRITE_EN bit is set, or a DMA request is asserted for DMA transfers. This bit is set only when SDHC performs a write operation to the card. 0 Not ready to write buffer 1 Ready to write buffer
5 RESP_CRC_ERR	Response CRC Error. This indicates that a transmission error occurred on the SD_CMD line during a response transfer. Write ‘1’ to this bit to clear it. 0 No error 1 Response CRC error occurred.
4	N/A
3 READ_CRC_ERR	Read CRC Error. This indicates that a transmission error occurred on the SD_DAT line during a card read. The user should re-try the transmission. Write ‘1’ to this bit to clear it. 0 No error 1 CRC read error occurred.
2 WRITE_CRC_ERR	Write CRC Error. This indicates that a transmission error occurred on the SD_DAT line during a card write operation. The user should check the WR_CRC_ERR_CODE field for more information about the CRC error. Write ‘1’ to this bit to clear it. 0 No error 1 CRC write error occurred.
1 TIME_OUT_RESP	Time Out Response. This indicates that the command response was not received in the time specified in the RES_TO Register. This can be caused by: • An unsupported command was received at the card(s). • Another MMC/SD_OP_COND command submitted after all cards had already sent their voltage ranges and the power-up routine was finished. • An identification command issued when all cards were already in standby state. • No card is on the bus. Write ‘1’ to this bit to clear this condition. 0 No error 1 Time out response error occurred.
0 TIME_OUT_READ	Time Out Read. Indicates that the expected data from the card was not received in the time specified in the READ_TO Register. The TIME_OUT_READ is cleared by an internal status change or by removing the source of the error. Write ‘1’ to this bit to clear it. 0 No error 1 Time out read data error occurred

27.3.3.3 SDHC Clock Rate Register (CLK_RATE)

Refer to Section 27.4.8, “System Clock Controller” for the clock scheme.

The high frequency input clock, IPG_PERCLK, is used to derive the low frequency clock to be used by the card and some of its internal logic. The divide circuitry consists of a 4-bit divider followed by a 12-bit prescaler. The IPG_PERCLK is first divided by the 4-bit divider to derive the signal, CLK_DIV. CLK_DIV is then divided by the 12-bit prescaler to derive CLK_20M, which is the source clock to be gated for internal logic and external cards.

CLK_20M is used internally by the SDHC. The MMC_SD_CLK is a buffered and gated version of the CLK_20M clock.

See Figure 27-6 for an illustration of valid bits in the SDHC Clock Rate Register and Table 27-8 for descriptions of the bit fields.

0x1001_3008 (CLK_RATE1)
0x1001_4008 (CLK_RATE2)

Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CLK_PRESCALER[15:4]												CLK_DIVIDER[3:0]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 27-6. SDHC Clock Rate Register

Table 27-8. SDHC Clock Rate Register Field Descriptions

Field	Description
31–16	N/A

Table 27-8. SDHC Clock Rate Register Field Descriptions (continued)

Field	Description
15–4 CLK_PRESCALER	Clock Prescaler. Specifies the divider value to generate CLK_20M from CLK_DIV. 0x000 CLK_20M is CLK_DIV 0x001 CLK_20M is CLK_DIV/2 0x002 CLK_20M is CLK_DIV/4 0x004 CLK_20M is CLK_DIV/8 0x008 CLK_20M is CLK_DIV/16 0x010 CLK_20M is CLK_DIV/32 0x020 CLK_20M is CLK_DIV/64 0x040 CLK_20M is CLK_DIV/128 0x080 CLK_20M is CLK_DIV/256 0x100 CLK_20M is CLK_DIV/512 0x200 CLK_20M is CLK_DIV/1024 0x400 CLK_20M is CLK_DIV/2048 0x800 CLK_20M is CLK_DIV/4096 Others Reserved
3–0 CLK_DIVIDER	Clock Divider. Specifies the divider value to generate CLK_DIV from input clock IPG_PERCLK. 0x1 CLK_DIV is IPG_PERCLK/2 0x2 CLK_DIV is IPG_PERCLK/3 0x3 CLK_DIV is IPG_PERCLK /4 0x4 CLK_DIV is IPG_PERCLK /5 0x5 CLK_DIV is IPG_PERCLK /6 0x6 CLK_DIV is IPG_PERCLK /7 0x7 CLK_DIV is IPG_PERCLK /8 0x8 CLK_DIV is IPG_PERCLK /9 0x9 CLK_DIV is IPG_PERCLK /10 0xa CLK_DIV is IPG_PERCLK /11 0xb CLK_DIV is IPG_PERCLK /12 0xc CLK_DIV is IPG_PERCLK /13 0xd CLK_DIV is IPG_PERCLK /14 0xe CLK_DIV is IPG_PERCLK /15 0xf CLK_DIV is IPG_PERCLK /16 Others Reserved

NOTE

The maximum frequency of MMC_SD_CLK is IPG_PERCLK /2.

27.3.3.4 SDHC Command and Data Control Register (CMD_DAT_CONT)

The SDHC Command and Data Control Register allows the user to specify the format of data and response, and to control the Read/Wait cycle. After configuring this register, enabling the MMC_SD_CLK will cause the command and argument configured in the CMD Number register and the CMD Argument register to be sent out to the card.

See Figure 27-7 for an illustration of valid bits in the SDHC Command and Data Control Register and Table 27-9 for descriptions of the bit fields.

0x1001_300C (CMD_DAT_CONT1)
0x1001_400C (CMD_DAT_CONT2)

Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_RESUME	0	0	CMD_RESP_LONG_OFF	STOP_READWAIT	START_READWAIT	BUS_WIDTH		INIT	0	0	WRITE_READ	DATA_ENABLE	FORMAT_OF_RESPONSE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-7. SDHC Command and Data Control Register

Table 27-9. SDHC Command and Data Control Register Field Description

Field	Description
31–16	N/A
15 CMD_RESUME	Command Resume. Used to restore Command and Data Control Register after Read/Wait cycle for SDIO card. 0 Issues command to card. 1 Does not issue command to card.
14–13	N/A
12 CMD_RESP_LONG_OFF	Command Response Long Off. Allows STATUS[13] END_CMD_RESP bit to be self cleared when the condition to generate this bit disappears. This is utilized in the Read/Wait cycle. For SD/MMC operation, the user should keep this bit set as ‘0’. 0 Bit was not cleared when read. 1 Allows bit to be cleared.
11 STOP_READWAIT	Stop Read/Wait. Ends the Read/Wait cycle for SDIO. When this bit is set, the SDHC will not drive DAT2 output low so that the SDIO card would end the Read/Wait cycle. For operation of SD/MMC, the user should keep this bit set as ‘0’. 0 No effect 1 Ends Read/Wait cycle.
10 START_READWAIT	Start Read/Wait. Starts the Read/Wait cycle for SDIO. When this bit is set, the SDHC will make the DAT2 output low and force the SDIO card to enter READWAIT cycle. For SD/MMC operation, the user should keep this bit set as ‘0’. 0 No Effect 1 Starts Read/Wait cycle.
9–8 BUS_WIDTH	Bus Width. Specifies the width of the data bus. These two bits must be set according to current SD/SDIO card bit mode. For MMC card, only 1-bit bus mode is supported. 00 1-bit 01 Reserved 10 4-bit 11 Reserved

Table 27-9. SDHC Command and Data Control Register Field Description (continued)

Field	Description
7 INIT	Initialize. Specifies whether the additional 80-clock cycle prefix (to initialize the card) will occur before every command. INIT enables/disables the additional 80-clock initialization time. 0 Disable 80 initialization clocks. 1 Enable 80 initialization clocks.
6–5	N/A
4 WRITE_READ	Write/Read. Specifies whether the data transfer of the current command is a write or read. operation 0 Read 1 Write
3 DATA_ENABLE	Data Enable. Specifies whether the current command includes a data transfer. 0 No data transfer included 1 Date transfer included
2–0 FORMAT_OF_RESPONSE	Format of Response. Sets the expected response format for current command. Refer to the SD I/O Specification 1.0 for detail information of the response format. 000 No response for current command 001 48-bit response with CRC7 check. For example: Format R1/R1b/R5/R6. 010 136-bit, CSD/CID read response. For example: Format R2. 011 48-bit Response without CRC check. For example: Format R3/R4. Others Reserved

27.3.3.5 SDHC Response Time Out Register (RES_TO)

The MMC/SD Response Time Out Register defines an interval within which a response must be returned, or a time-out error will occur. After the SDHC sends out a command, if the card does not respond within the specified interval, the RESPONSE TIMEOUT status bit (STATUS[1]) and the END_CMD_RESP status bit (STATUS[13]) will be set.

See Figure 27-8 for an illustration of valid bits in the MMC/SD Response Time Out Register and Table 27-10 for descriptions of the bit fields.

0x1001_3010 (RES_TO1)
0x1001_4010 (RES_TO2)

Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RESPONSE TIME OUT[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 27-8. MMC/SD Response Time Out Register

Table 27-10. MMC/SD Response Time Out Register Field Descriptions

Field	Description
31–8	N/A
7–0 RESPONSE TIME OUT	Response Time Out. This value determines the interval by which Response time-out is detected. The clock starts counting when the last bit of the command is sent. The clock counts unit is MMC_SD_CLK to card. 0x01 1 clock count 0x02 2 clock counts ... ... 0xFF 255 clock counts

27.3.3.6 SDHC Read Time Out Register (READ_TO)

The MMC/SD Read Time Out Register defines an interval that read data must be returned within or a time out error will occur. After the SDHC sends out the data read command, if no data is returned within the specified interval, the READ TIMEOUT status bit (STATUS[0]) and the READ_OP_DONE status bit (STATUS[11]) will be set.

See Figure 27-9 for an illustration of valid bits in the SDHC Read Time Out Register and Table 27-11 for descriptions of the bit fields.

0x1001_3014 (READ_TO1)
0x1001_4014 (READ_TO2)

Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA_READ_TIME_OUT[15:0]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 27-9. SDHC Read Time Out Register

Table 27-11. SDHC Read Time Out Register Field Descriptions

Field	Description
31–16	N/A
15–0 DATA_READ_TIME_OUT	Data Read Time Out. This value determines the interval by which Read Data time-outs are detected. The user needs to check the timeout limit of the card and the clock frequency to configure this register. For safety, 0xFFFF is recommended. The timeout clock starts counting when the last bit of the command is sent. The count unit is MMC_SD_CLK/256. The maximum delay the SDHC can tolerate for a data timeout is related to the card clock. If the clock is 25 MHz and this register is 0xFFFF, the maximum delay which the SDHC waits will be about 670ms. If the card does not give data in 670 ms, SDHC will issue a data read time-out error and terminate the current data read operation. This is designed to meet the SD physical layer specification, with typical time-out limit to be 100ms~200ms. But for some SDIO cards, the time-out limit may be up to 1 s. In such case, the user needs to lower the MMC_SD_CLK frequency to accommodate the delay to 1 s, which requires configuring the MMC_SD_CLK to about 16 MHz and setting this register for 0xFFFF.

27.3.3.7 SDHC Block Length Register (BLK_LEN)

The SDHC Block Length Register defines the number of bytes in a block (block size). Since the stream mode of MMC is not supported, the block length must be set for every transfer. The block length supported by the SDHC ranges from 1 bytes to 2048 bytes, but the user needs to check the block size supported by the card before configuring this register. For the SDIO, the block length must be less than the maximum block size defined in the card's CCCR. For the SD/MMC, the block length must be less than the maximum block size defined in the card's CSD register.

NOTE

The software should write to this register only when no SD bus transaction is executing.

See Figure 27-10 for an illustration of valid bits in the SDHC Block Length Register and Table 27-12 for descriptions of the bit fields.

0x1001_3018 (BLK_LEN1)
0x1001_4018 (BLK_LEN2)

Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	BLOCK LENGTH[11:0]											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-10. SDHC Block Length Register

Table 27-12. SDHC Block Length Register Field Descriptions

Field	Description
31–12	N/A
11–0 BLOCK LENGTH	Block Length. Specifies the number of bytes in a block during data transfer (block size). For the MMC and SD cards, the value set must keep same as the blk_len set in the CARD. For the SDIO, the IO access is performed through the CMD53 IO_RW_EXTEND command. This command has two modes: • Byte mode. For byte mode, its operation is similar to a single block transfer command for SD where the block length is the byte count in the command argument. • Block mode. For block mode, its operation is similar to a multi-block transfer command for the SD where the block length is the block size defined in the command argument. For multi-block data transfers, a block length that is equal to an integer multiple of the data buffer size is preferred. Otherwise, the buffer utilization would be poor. If the data size that needs to be transferred is not an integer multiple of the buffer size, there are two options to transfer the data: • Option 1: The user needs to split the transaction. The remainder of block size data is transferred by using a single block command for the last transfer. • Option 2: The user needs to add filler data in the last block to fill the block size to be as large as the buffer size. The data buffer size is 64 bytes in 4-bit mode and 16 bytes in 1-bit mode.Refer to Section 27.4.1, “Data Buffers” for more information about data buffers. 0x000 0 byte 0x001 1 byte ... ... 0x7FF 2047 bytes 0x800 2048 bytes 0x801–0XFFF Reserved

27.3.3.8 SDHC Number of Blocks Register (NOB)

The SDHC Number of Blocks Register defines the number of blocks in the multi-block transfer mode. This register and the Block Length Register determines the number of bytes to be transferred during one command. The number is decremented every time a block transfer is completed and stops when the count reaches zero. When all data transfers are completed, the STATUS[11] READ_OP_DONE is set if it is a read (from card) transfer, or the STATUS[12] WRITE_OP_DONE is set if it is a write (to card) transfer.

The software should write to this register only when no MMC/SD transaction is executing.

The NOB and the BLK_LEN defines the maximum data to be transferred in a single data transfer command.

Maximum data size to be transferred in bytes = block length x number of blocks.

See Figure 27-11 for an illustration of valid bits in the SDHC Number of Blocks Register and Table 27-13 for descriptions of the bit fields.

0x1001_301C (NOB1)
0x1001_401C (NOB2)

Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NOB[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-11. SDHC Number of Blocks Register

Table 27-13. SDHC Number of Blocks Register Field Descriptions

Field	Description
31–16	N/A
15–0 NOB	Specifies the number of blocks in a block transfer. One block should be set if the data transfer command is a single block transfer command or IO_RW_EXTEND (CMD53) in byte mode. For multi-block transfer commands to SD/MMC card and IO_RW_EXTEND (CMD53) in block mode to SDIO card, this register should be set for the block count the software expected. The maximum block number to be transferred for command can be as large as 65535. Blocks can range in length from 0 to 65535 bytes. For SD Memory card or a memory parts of a SDIO combo card, the user will need to send CMD12 to stop the multi-block transfer. For a SDIO CMD53 in block mode and if user needs to abort the transfer earlier, the user needs to use CMD52 IO-Abort to abort the transfer. 0x0000 0 Block 0x0001 1 Block ... ... 0xFFFF 65535 Blocks **Note:** The maximum transfer blocks is 64 Kbytes. If the user uses infinite transfer command to transfer data, such as multi-block transfer command for memory card or infinite block transfer CMD53 for SDIO card, this register needs to be set to the real number of blocks that the user expected to transfer. The user also needs to abort the transfer through CMD12 or CMD52 IO-Abort to do this.

27.3.3.9 SDHC Revision Number Register (REV_NO)

The read-only SDHC Revision Number Register is a read-only register that displays the revision number of the module.

See Figure 27-12 for an illustration of valid bits in the SDHC Revision Number Register and Table 27-14 for descriptions of the bit fields.

0x1001_3020 (REV_NO1)
0x1001_4020 (REV_NO2)

Access: User read

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Revision Number[15:0]															
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 27-12. SDHC Revision Number Register

Table 27-14. SDHC Revision Number Register Field Descriptions

Field	Description
31–16	N/A
15–0 REVISION NUMBER	Revision Number. Specifies the revision number of the MMC/SD module. This is fixed at 0x0000_0400.

27.3.3.10 SDHC Interrupt Control Register (INT_CNTR)

When certain events occur in the module, the SDHC has the ability to set an interrupt as well as to set corresponding Status register bits. The SDHC Interrupt Control Register allows the user to control whether these interrupts should occur.

See Figure 27-13 for an illustration of valid bits in the SDHC Interrupt Control Register and Table 27-15 for descriptions of the bit fields.

0x1001_3024 (INT_CNTR1)
0x1001_4024 (INT_CNTR2)

Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CARD_INSERTION_EN	CARD_REMOVAL_EN	SDIO_IRQ_EN	DAT0_EN	0	0	0	0	0	0	0	BUF_READ_EN	BUF_WRITE_EN	END_CMD_RES	WRITE_OP_DONE	READ_OP_DONE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-13. SDHC Interrupt Control Register

Table 27-15. SDHC Interrupt Control Register Field Descriptions

Field	Description
31–19	N/A
18 SDIO_INT_WKP_EN	SDIO Interrupt Wake-Up Enable. When INT_CNTR[13] is set to a ‘1’ (SDIO interrupt enabled), this bit controls whether a SDIO card interrupt is detected synchronously or asynchronously. To set this bit, the SDIO card interrupt is detected asynchronously and the SDIO card interrupt is used as a wake-up event. Set this bit only in low power mode or when all the clocks to SDHC are off. 0 Disable SDIO card interrupt as wake-up event. 1 Enable SDIO card interrupt as wake-up event.
17 CARD_INSERTION_WKP_EN	Card Insertion Wake-Up Enable. When INT_CNTR[15] is set to a ‘1’ (card insertion interrupt enabled), this bit controls whether a card insertion interrupt is detected synchronously or asynchronously. To set this bit, the card insertion interrupt is captured asynchronously and the interrupt is used as a wake-up event. Set this bit only in low power mode or when all the clocks to SDHC are off. 0 Disable card insertion as wake-up event. 1 Enable card insertion as wake-up event.

Table 27-15. SDHC Interrupt Control Register Field Descriptions (continued)

Field	Description
16 CARD_REMOVAL_WKP_EN	Card Removal Wake-Up Enable. When INT_CNTR[14] is set to a '1' (card removal interrupt enabled), this bit controls whether the card removal interrupt is detected synchronously or asynchronously. To set this bit, the card removal interrupt is detected asynchronously and the interrupt is used as a wake-up event. Set this bit only in low power mode or when all the clocks to SDHC are off. 0 Disable card removal as wake-up event 1 Enable card removal as wake-up event
15 CARD_INSERTION_EN	Card Insertion Enable. Setting this bit enables the card insertion interrupt. Since the card detection is through the value of DAT3 data line, if this card is in 4-bit mode, any data transfers in the DAT3 line cause false card insertion interrupts to be generated. The card insertion interrupt should be disabled after the first time the card insertion is detected. To avoid the false status bit generation during data transfer, the card insertion status will be masked by this bit. It should be enabled only after the card is removed from the socket. The default of this bit is to disable the card insertion interrupt. When this interrupt is detected, the user needs to write a '1' to the STATUS[31] bit to clear the card insertion status interrupt. 0 Card insertion interrupt disabled 1 Card insertion interrupt enabled **Note:** INT_CNTR[17] CARD_INSERTION_WKP_EN controls whether this interrupt is detected asynchronously or synchronously.
14 CARD_REMOVAL_EN	Card Removal Enable. Setting this bit enables the card removal interrupt. Since card detection is through the value of the DAT3 data line, if this card is in 4-bit mode, the data transfer through the DAT3 line causes false card removal interrupt to be generated. The card removal interrupt should be enabled only when there are no active data transfers on the DAT3 line. To avoid the false status bit generation during data transfer, the card insertion status will be masked by this bit. The default of this bit is to disable the card removal interrupt. When this interrupt is detected, the user needs to write a '1' to the STATUS[30] bit to clear the card removal status interrupt. 0 Card removal interrupt disabled 1 Card removal interrupt enabled **Note:** INT_CNTR[16] CARD_REMOVAL_WKP_EN controls whether this interrupt is detected asynchronously or synchronously.
13 SDIO_INT_EN	SDIO Interrupt Enable. Masks the interrupt from the SD I/O card to the SDHC module interrupt. 0 SD I/O interrupt disabled 1 SD I/O interrupt enabled **Note:** INT_CNTR[18] SDIO_INT_WKP_EN controls whether this interrupt is detected asynchronously or synchronously.
12 DAT0_EN	Data Enable. Identifies how the SD I/O interrupt is detected. An interrupt is determined by SD_DAT [1] low, but this bit is an option setting for the SDIO bit. When SDHC is preforming data transfer and the SD bus mode is 1-bit mode, the user should set this bit to '0'. 0 SD I/O's Interrupt detection based on SD_DAT[3:0] = 110x 1 SD I/O's Interrupt detection based on SD_DAT[3:0] = 1101
11–5	N/A

Table 27-15. SDHC Interrupt Control Register Field Descriptions (continued)

Field	Description
4 BUF_READ_EN	Bus Read Enable. This bit controls the Buffer Read Ready interrupt. If the bit is '1', the interrupt is enabled. When the buffer becomes full during a read operation, an interrupt is generated. The user needs to move the data out of the FIFO and clear the BUF_READ_READY bit to clear the interrupt. 0 Buffer status interrupt is disabled. 1 Buffer status interrupt is enabled.
3 BUF_WRITE_EN	Bus Write Enable. This bit controls the Buffer Write Ready interrupt. If the bit is '1', the interrupt is enabled. When the buffer becomes empty-during a write operation, an interrupt will be generated. The user needs to write data to the FIFO and clear the BUF_WRITE_READY bit to clear the interrupt. 0 Buffer status interrupt is disabled. 1 Buffer status interrupt is enabled.
2 END_CMD_RES	End Command Response. This bit controls the interrupt generation on the status at the end of the command response. When this bit is '1', the SDHC generates an interrupt at the end of the command response status. 0 End Command-response interrupt is disabled. 1 End Command-response interrupt is enabled.
1 WRITE_OP_DONE	Write Operation Done. This bit controls the interrupt generation for the status of write operation. When the interrupt enabled, the SDHC generates an interrupt when the configured bytes of data are transferred to the card. 0 Write_OP_DONE interrupt is disabled. 1 Write_OP_DONE interrupt is enabled.
0 READ_OP_DONE	Read Operation Done. This bit controls the interrupt generation for the status of read operation completion. When the interrupt is enabled, the SDHC generates an interrupt when the pre-defined bytes of data are transferred from the card. 0 READ_OP_DONE interrupt is disabled. 1 READ_OP_DONE interrupt is enabled.

When an interrupt is generated, there may be some error bits in the STATUS register set as well as the interrupt status. The user needs to check the error status bit to make sure there is no error in the SDHC operation. For example, when the READ_OP_DONE (STATUS[11]) status is set or the READ_OP_DONE interrupt is detected, the user needs to check both the STATUS[3] and STATUS[0] bits as well to make sure the read operation completed without a CRC error or a time out error. Another example is a write operation, if both the WRITE_OP_DONE(STATUS[12]) and WRITE_CRC_ERR (STATUS[2]) bits are set. This means the write operation ended with CRC error. See Table 27-16 for a summary of the relationship between the interrupt, interrupt control register, and status registers in the SDHC.

Table 27-16. Interrupt Mechanisms

Source STATUS Bit Name (Status Bit Number)	Does this status generate interrupt directly?	INT_Control Register Bit Name (INT_CNTR Bit Number)	Interrupt/Status Clear Method
TIME_OUT_READ (0)	No, alert by using the READ_OP_DONE bit in the SDHC Status Register	READ_OP_DONE (0)	Clear status by writing ‘1’.
TIME_OUT_RESP (1)	No, alert by using the END_CMD_RESP bit in the SDHC Status Register	END_CMD_RES (2)	Clear status by writing ‘1’.
WRITE_CRC_ERR (2)	No, alert by using the WRITE_OP_DONE bit in the SDHC Status Register	WRITE_OP_DONE (1)	Clear status by writing ‘1’.
READ_CRC_ERR (3)	No, alert by way of the READ_OP_DONE bit in the SDHC Status Register	READ_OP_DONE (0)	Clear status by writing ‘1’.
RESP_CRC_ERR (5)	No, alert by using the END_CMD_RESP bit in the SDHC Status Register	END_CMD_RES (2)	Clear status by writing ‘1’.
BUF_WR_RDY (6)	Yes	BUF_WRITE_EN (3)	Clear status by writing data to FIFO buffer.
BUF_READ_RDY (7)	Yes	BUF_READ_EN (4)	Clear status by reading data from FIFO buffer.
READ_OP_DONE(11)	Yes	READ_OP_DONE (0)	Clear status by writing ‘1’.
WRITE_OP_DONE(12)	Yes	WRITE_OP_DONE (1)	Clear status by writing ‘1’.
END_CMD_RESP(13)	Yes	END_CMD_RESP (2)	Clear status by writing ‘1’.
SDIO_INT_ACTIVE(14)	Yes	SDIO_INT_EN (13)	Clear status by writing ‘1’.
CARD_REMOVAL(30)	Yes	CARD_REMOVAL_EN (14)	Clear status by writing ‘1’.
CARD_INSERTION(31)	Yes	CARD_INSERTION_EN (15)	Clear status by writing ‘1’.

27.3.3.11 SDHC Command Number Register (CMD)

The command to the SD card is always 48 bits long. It contains one start bit, one direction bit, six command number bits, 32 argument bits, seven CRC bits, and one end bit. For more details on the format of the command, refer to the SD physical layer specification. Refer to Table 27-21 for MMC/SD/SDIO cards command list and the related card specification for the detailed information about each command.

In SDHC, the command start bit, direction bit, CRC7 bits, and end bit are automatically generated by the hardware. Configure the SDHC command number register and SDHC command argument register to issue a command to the card.

See Figure 27-14 for an illustration of valid bits in the SDHC Command Number Register and Table 27-17 for descriptions of the bit fields.

0x1001_3028 (CMD1)
0x1001_4028 (CMD2)

Access: User read/write

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

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

Figure 27-14. SDHC Command Number Register

Table 27-17. SDHC Command Number Register Field Descriptions

Field	Description
31–6	N/A
5–0 COMMAND Number	Command Number. The SDHC module communicates with the MMC/SD/SDIO card(s) by sending commands and arguments. The command to send is set in the MMC/SD Command Number Register (CMD) and the argument is defined in SDHC CMD Argument Register (ARG). See Table 27-21 for the brief information of the full list of MMC/SD/SDIO commands. 0x00 CMD0 0x01 CMD1 ... ... 0x3F CMD63 **Note:** The user should check the detailed information from the related card specification.

27.3.3.12 SDHC CMD Argument Register (ARG)

This register contains the MMC/SD/SDIO command argument.

See Figure 27-15 for an illustration of valid bits in the SDHC Command Argument Register and Table 27-18 for descriptions of the bit fields.

0x1001_302C (ARG1)
0x1001_402C (ARG2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ARG[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ARG[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-15. SDHC Command Argument Register

Table 27-18. SDHC Command Argument Register Field Descriptions

Field	Description
31–0 ARG	Command Argument. Specifies the argument for the current command. **Note:** The user should check the detailed command argument information from the related card specification.

27.3.3.13 SDHC Response FIFO Access Register (RES_FIFO)

There is an 8 x 16 bit FIFO to store the response from the card in SDHC. This register is used to access this FIFO. The MSB 16 bits of the response is accessed first and the LSB 16 bits is accessed last.

See Figure 27-16 for an illustration of valid bits in the SDHC Response FIFO Access Register and Table 27-19 for descriptions of the bit fields.

0x1001_3034 (RES_FIFO1)
0x1001_4034 (RES_FIFO2)

Access: User read

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

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

Figure 27-16. SDHC Response FIFO Register

Table 27-19. SDHC Response FIFO Register Field Descriptions

Field	Description
31–16	N/A
15–0 RESPONSE_CONTENT	RESPONSE CONTENT FIFO access register. There is a FIFO in the SDHC that is used to store the command response received from card. Every time the Host sends a command to a card, the current contents stored in the FIFO are cleared and new response arguments are stored into the Response FIFO. According to the SD card specification, the command response size can be 48 bits or 136 bits (R2 response). Refer to the SD Memory Card specification for more detailed information about the command response format. The size of Response FIFO is 8 x 16 bits (128 bits). For a 48-bit response, only 48 bits of the FIFO have valid contents. The user must perform three reads to this response FIFO access register to retrieve the entire 48-bit response content. For a 136-bit R2 Response (response for CID[127:0] or CSD[127:0] register), only the contents of the 128-bit CID and CSD register are stored in the Response FIFO. This first byte of the R2 response is not stored in the Response FIFO. The user can retrieve the CIS/CSD register from the Response FIFO through eight accesses to the FIFO access register. All the CRC bits in the response are not stored in the response FIFO. This Response FIFO is read-only. **Note:** The CRC7 and end bit for response is hardware checked by SDHC and the corresponding field of the response will not be stored in the response FIFOs.

27.3.3.14 SDHC Data Buffer Access Register (BUFFER_ACCESS)

The SDHC uses two 64-byte data buffers in a ping-pong manner—while one buffer is receiving new transmission information, the other buffer is deleting the previous transmission. Data can be transferred by the DMA and the SD card simultaneously to maximize throughput between the two clock domains (that is, the IP peripheral clock, IPG_PERCLK, and the host clock, CLK_20M). These buffers are used as temporary storage for data being transferred between the host system and the card and vice versa. The user can read or write data to the buffers through this Buffer Access Register. Refer to Section 27.4.1, “Data Buffers” for more information about the data buffers.

In the read operation, the SDHC stores the data received from the card into the buffer. The user needs to move the data out of the buffer when the buffer is full.

In the write operation, the SDHC fetches data from the buffer and transfers them to the card. The user can then access the data buffer through the SDHC Data buffer Access Register. The user needs to move data into the buffer when the buffer is empty.

See Figure 27-17 for an illustration of valid bits in the SDHC Data Buffer Access Register and Table 27-20 for descriptions of the bit fields.

0x1001_3038 (BUFFER_ACCESS1)
0x1001_4038 (BUFFER_ACCESS2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FIFO CONTENT[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIFO CONTENT[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-17. SDHC Buffer Access Register

Table 27-20. SDHC Buffer Access Register Field Descriptions

Field	Description
31–0 FIFO CONTENT	First in/First Out Content. These bits hold 32-bit data upon a read or write transfer. The size of the FIFO is 4 x 32 bits (16 bytes in total) for SD 1-bit mode and 16 x 32 bits (64 bytes in total) for SD 4-bit mode. For reception, the SDHC controller generates a DMA request when the FIFO is full. Upon receiving this request, DMA starts transferring data from the SDHC FIFO to system memory by reading the Data Buffer Access Register for a number of pre-defined bytes. For transmit, the SDHC controller generates a DMA request when the FIFO is empty. Upon receiving this request, DMA starts moving data from the system memory to the SDHC FIFO by writing to the Data Buffer Access Register for a number of pre-defined bytes.

27.4 Functional Description

The following sections provide a brief functional description of the major system blocks, including the DMA interface, memory controller, logic/command controller, and system clock controller.

27.4.1 Data Buffers

The SDHC uses two data buffers in a ping-pong manner so that data can be transferred by the DMA and the SD card simultaneously to maximize throughput between the two clock domains (that is, the IP peripheral clock, IPG_PERCLCK, and the host clock, CLK_20M). See Figure 27-18 for an illustration of the buffer scheme. These buffers are used as temporary storage for data being transferred between the host system and the card.

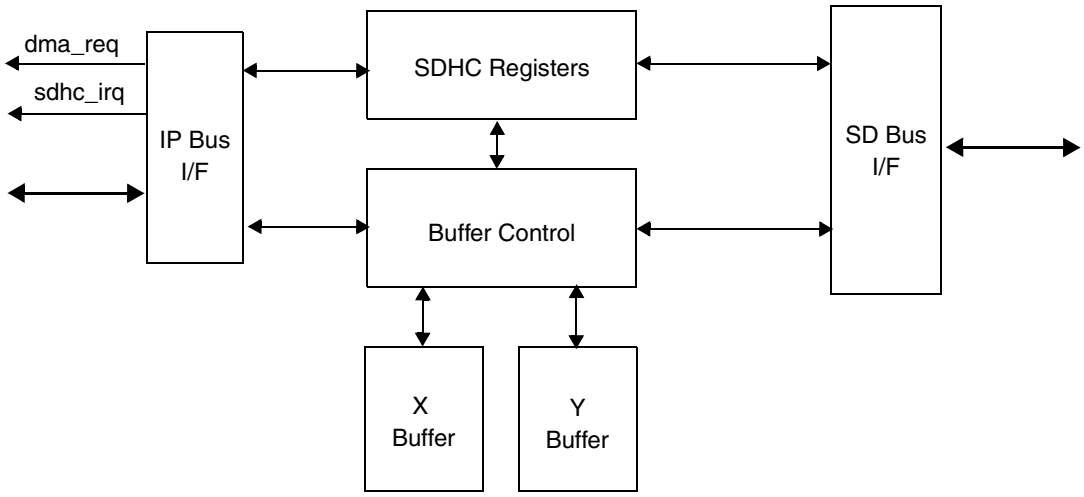


Figure 27-18. SDHC Buffer Scheme

For a host-read operation, the SDHC automatically transfers data into the next available buffer. It is then read out by the DMA and written to the system memory when the DMA request from SDHC becomes highest priority in the DMA arbiter. Conversely, for a host-write operation, the DMA writes data into the next available buffer. The SDHC then reads the data out of the buffer and writes it to the card through the SD Host interface.

27.4.1.1 Data Buffer Access

DMA/CPU accesses the data buffer of SDHC through the 32-bit Data Buffer Access (DBA) register. Internally, the SDHC maintains a pointer into the data buffer. Accesses to the DBA register will increase the address value of the pointer. The pointer value of SDHC is not directly accessible by the software. The pointer refers to a 32-bit port-size FIFO, so all the access to the FIFO must be 32-bit size. Sequential and contiguous access is necessary to increase the pointer address value correctly. Random or skipped access is not allowed. In some cases, when the block length of the data transfer is not a multiple of 32 bits, the last data access to the FIFO may be 24-bit, 16-bit or 8-bit. Since SDHC FIFO allows only 32-bit access sizes, the user must put/get the data bytes on the correct byte lanes of the SDHC 32-bit data bus. The byte arrangement order is little endian format. For an 8-bit data access to the FIFO, it will be in bit[7–0] of the data bus. For 16-bit data access, the data will be in the bit[15–0] of the data bus. For a 24-bit data access, the data will be in the bit[23–0] of the data bus.

When data goes to the card, the 32-bit data in the data buffer FIFO will be shifted out to card from the LSB byte to the MSB byte, but for each byte, the bit sequence of the shift will be from MSB bit to LSB bit. When read data leaves the card, the data shifted in will be stored from LSB byte to MSB byte in the data buffer FIFO. See Figure 27-19 for the bytes lane relationship between the card bus and SDHC IP bus.

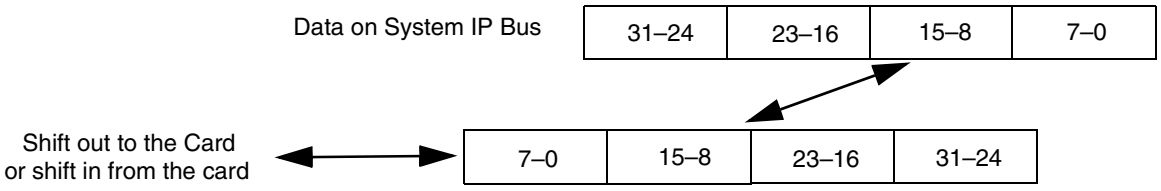


Figure 27-19. Byte Lanes Relationship Between System IP Bus and SD Card Bus

27.4.1.2 Write Operation Sequence

There are two ways to write data to the buffer when the user transfer data to the card. One way is by using DMA through the SDHC DMA request signal. The other way is by using the CPU through the BUF_WR_RDY (STATUS[6]) bit (interrupt or polling).

The SDHC asserts a DMA request when the data buffer is empty and ready to receive new data. At the same time, the SDHC would set the BUF_WR_RDY (STATUS[6]) bit. The buffer write ready interrupt will be generated if it is enabled by the software.

The buffer accumulates the data written through the data port until the data count reaches the buffer size. The SDHC will not start data transmission until a full buffer size of data is written to the data buffer. The SDHC will start data transmission when the SD bus is ready for new transfer. When the other buffer is empty and more data is to be transferred, the SDHC will assert a new DMA request and set the BUF_WR_RDY bit. In case the DMA does not keep up with moving data into the FIFOs, the SDHC will stop the SD_CLK at the block gap to avoid an data buffer underrun situation.

27.4.1.3 Read Operation Sequence

There are two ways to fetch data from the buffer when the user read data from the card. One way is by using DMA through the SDHC DMA request signal. The other way is by using the CPU through the BUF_READ_RDY (STATUS[7]) bit (interrupt or polling).

The SDHC asserts a DMA request when data buffer is full and ready for DMA/CPU to fetch the data out of the buffer. At the same time, the SDHC would set the BUF_WR_RDY (STATUS[7]) bit. The buffer write ready interrupt will be generated if it is enabled by the software.

The SDHC starts receiving data only when either of the dual data FIFO is empty. The buffer accumulates data read from the card until the data count reaches the buffer size. The SDHC asserts a DMA request when either one of the data buffer is full. For multiple block data transfers, while the DMA/CPU is moving data by reading the DBA register, the SDHC will receive data into the other FIFO if it is empty and the SD bus is ready. In case the DMA/CPU does not keep up with reading data out of the FIFOs, the SDHC will stop the SD_CLK at the block gap to avoid an overflow situation.

27.4.1.4 Data Buffer Size

The user needs to know the buffer size for buffer operation during data transfer. In SDHC, both of two data buffers are 64 bytes in size. Each data buffer is divided into four 16 bytes data buffers that correspond to

the four data lines of SD bus. Thus, each data line in SDHC contains a dual 16-byte buffer. The data buffer size will be 64 bytes in 4-bit SD mode or 16 bytes in 1-bit SD mode.

During multi-block data transfer, the block length, which is an integer multiple of the buffer size is preferred. The buffer would be ready to read by CPU/DMA when either of the buffer is full (STATUS[27] or STATUS[26] is set, and STATUS[7] is set) when full buffer data is written to either of the X or Y buffer. The buffer would be ready to write by CPU/DMA (STATUS[29] or STATUS[28] is set, and STATUS[6] is set) when the full buffer of data is fetched out of the buffer. The buffer ready status bit and DMA request would be set accordingly.

For single-block data transfer, when the block length is smaller than the buffer size or when the block length is not an integer multiple of the buffer size, it is possible that the data size that needs to be written to the buffer or to be fetched out of the buffer is smaller than the buffer size. In this case, the buffer would be full (SDHC set STATUS[27] or STATUS[26]) when this data is written to the buffer. The buffer would be empty (SDHC set STATUS[29] or STATUS[28]) when this buffer of data is fetched out of the buffer. The buffer ready status bit and DMA request would be set accordingly. From the software aspect, the buffer size become variable and equal to the real data size that needs to be transferred. This will ease the software programming of SDHC. The user does not need to fill dummy data to make the buffer full.

27.4.1.5 Dividing Large Data Transfer

This SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Maximum data size = block size x block count

The block size can be a multiple of the size of the data buffer. However, it is recommended the block size be set equal to the size of data buffer. This allows the SDHC to stop the SD_CLK during block gaps of an overflow or underrun condition occurs. Stopping the SD_CLK while the DATA lines are active may cause data corruption (when the clock resumes) on some card designs available on the market. If an application or Card Driver is to transfer larger sizes of data, the Host Driver will divide larger data sizes into multiple blocks.

The length of a multiple block transfer needs to be in block size units. If the total data length cannot be divided evenly to a multiple of the block size, then there are two ways to transfer the data depending on the function and card design. One way is for the Card Driver to split the transaction. The remainder of block size data is then transferred by using a single block command at the end. Another way is to add dummy data in the last block to fill the block size. In the second method, the card must be able to remove the dummy data.

See Figure 27-20 for an example that shows dividing of large data transfers.

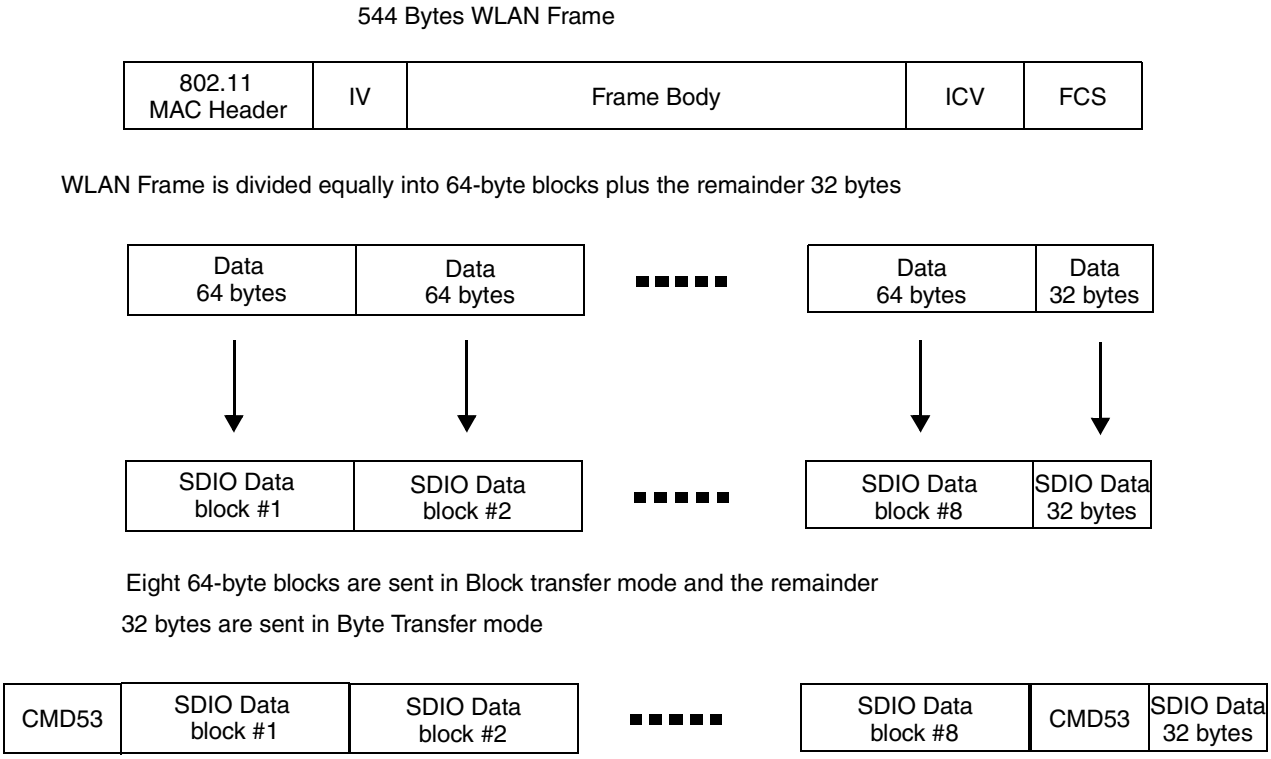


Figure 27-20. Example for Dividing Large Data Transfer

27.4.2 DMA Interface

The DMA interface block controls all data routing between the external data bus (DMA access), internal SDHC module data bus, and internal system FIFO access through dedicated state machine. This state machine monitors the status of FIFO content (empty or full), FIFO address, and byte/block counters for the SDHC module and the application. See Figure 27-21 for an illustration of the DMA interface block.

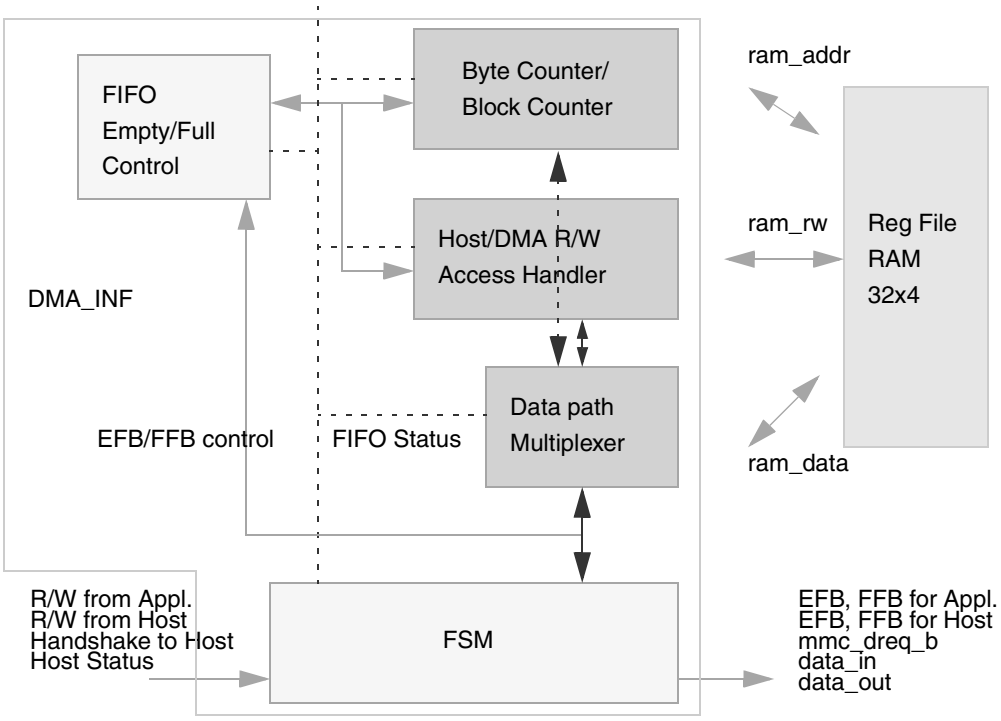


Figure 27-21. DMA Interface Block

In addition, this block also handles the burst request to the external DMA controller, internal register write-error detection, Read/Wait handling of SDIO, and all IP-related output responses.

27.4.2.1 DMA Request

If the SDHC is in the data transfer state, the SDHC generates DMA requests according to its buffer status. During read operations, the SDHC generates DMA requests if one of its data buffer is full. During write operations, the SDHC generates DMA requests if one of its data buffers is empty.

To avoid buffer under-run conditions during a write operation, the MMC_SD_CLK stops automatically when both buffers are empty. After the DMA or CPU completes writing data into one of the buffers, the MMC_SD_CLK automatically resumes to continue the data transfer.

Similarly, to avoid buffer overflow during read operations, the MMC_SD_CLK stops automatically when both buffers are full. After the DMA or CPU moves the data out of the buffer, the MMC_SD_CLK automatically resumes to continue the data transfer.

27.4.3 Memory Controller

This controller provides the SDIO-IRQ and Read/Wait service handling, card detection, command response handling, and all SDHC interrupt handling. The memory controller also contains the register table. See Figure 27-22 for an illustration of the block diagram for the memory controller.

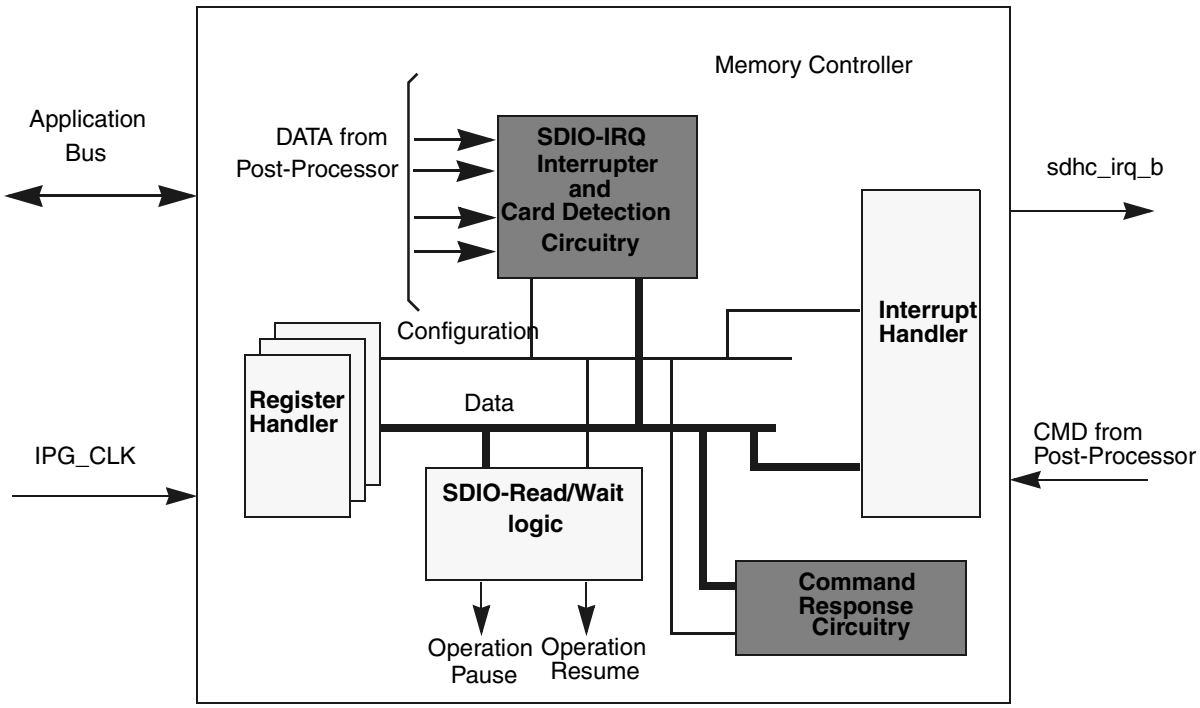


Figure 27-22. Memory Controller Block Diagram

A summary of events when a SDIO card generates an interrupt is detailed in this section. When an SDIO card generates an interrupt request, it sets its interrupt pending bit in the CSR register and asserts the interrupt line, which is shared with DAT[1] line in 4-bit mode. The SDHC detects and steers the card's interrupt to the selected IRQ line and to the interrupt controller.

27.4.4 SDIO Card Interrupt

27.4.4.1 Interrupts in 1-Bit Mode

In this case, the SD_DAT[1] pin is dedicated to providing the interrupt function. An interrupt is asserted by pulling the SD_DAT[1] low until the host clears the interrupt.

27.4.4.2 Interrupt in 4-Bit Mode

Since the interrupt and data line 1 share pin 8 in 4-bit mode, an interrupt will be sent by the card and recognized by the host only during a specific time. This is known as the interrupt period. The SDHC samples the level on Pin 8 only during the interrupt period. At all other times, the host interrupt controller ignores the level on Pin 8. The definition of the interrupt period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the interrupt period becomes active two clock cycles after the completion of a data packet. This interrupt period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in 4-bit mode, there is limited time that the interrupt period can be active because of the limited period of data line availability between the multiple blocks of data. This requires a more strict definition of the interrupt period. For this case, the interrupt period is limited to two clock cycles. This period begins two clock cycles after the end bit of the previous data block. During this two-clock cycle interrupt period, if an interrupt is pending, the SD_DAT[1] line is held low for one clock cycle and the last clock cycle pulling SD_DAT[1] high. On completion of the interrupt period, the card releases the SD_DAT[1] line into the high Z state.

When in 4-bit mode, the SDHC differentiates a data start bit and the interrupt period by checking that all four data lines are low for the start of new data. In the case of an interrupt, only the DAT[1] should have gone low. After the last data block is sent, the interrupt period starts as normal. The interrupt period ends after the next command with data instead of lasting only two cycles.

Refer to SDIO Card Specification v1.0 for further information about SDIO card interrupt.

27.4.4.3 Card Interrupt Handling

When the SDIO bit in the Interrupt Control Register is set to 0, the Host Controller clears the interrupt request to the system Interrupt Controller. The SDIO Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to ‘1’. The Host Driver should clear the SDIO Interrupt enable bit before servicing the SDIO Interrupt. The Host Driver should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO Status bit is cleared by resetting the SDIO interrupt. Writing to this bit has no effect in 1-bit mode, as the Host Controller detects the SDIO Interrupt with or without SD clock (to support wake-up). In 4-bit mode, the interrupt signal is sampled during the interrupt period. There are some sample delays between the interrupt signal from the SDIO card and the interrupt to the Host System Interrupt Controller. When the SDIO status has been set and the Host Driver needs to start this interrupt service, the SDIO bit in the Interrupt Control Register is set to ‘0’ to clear the SDIO interrupt status latched in the SDHC and to stop driving the interrupt signal to the System Interrupt Controller. The Host Driver must issue a CMD52 to clear the interrupts at the card. After completion of the card interrupt service, the SDIO Interrupt Enable bit is set to ‘1’. The SDHC starts sampling the interrupt signal again.

- Figure 27-23 (a) shows the SDIO card interrupt scheme.
- Figure 27-23 (b) shows the sequences of software and hardware events that occur during the card interrupt handling procedure.

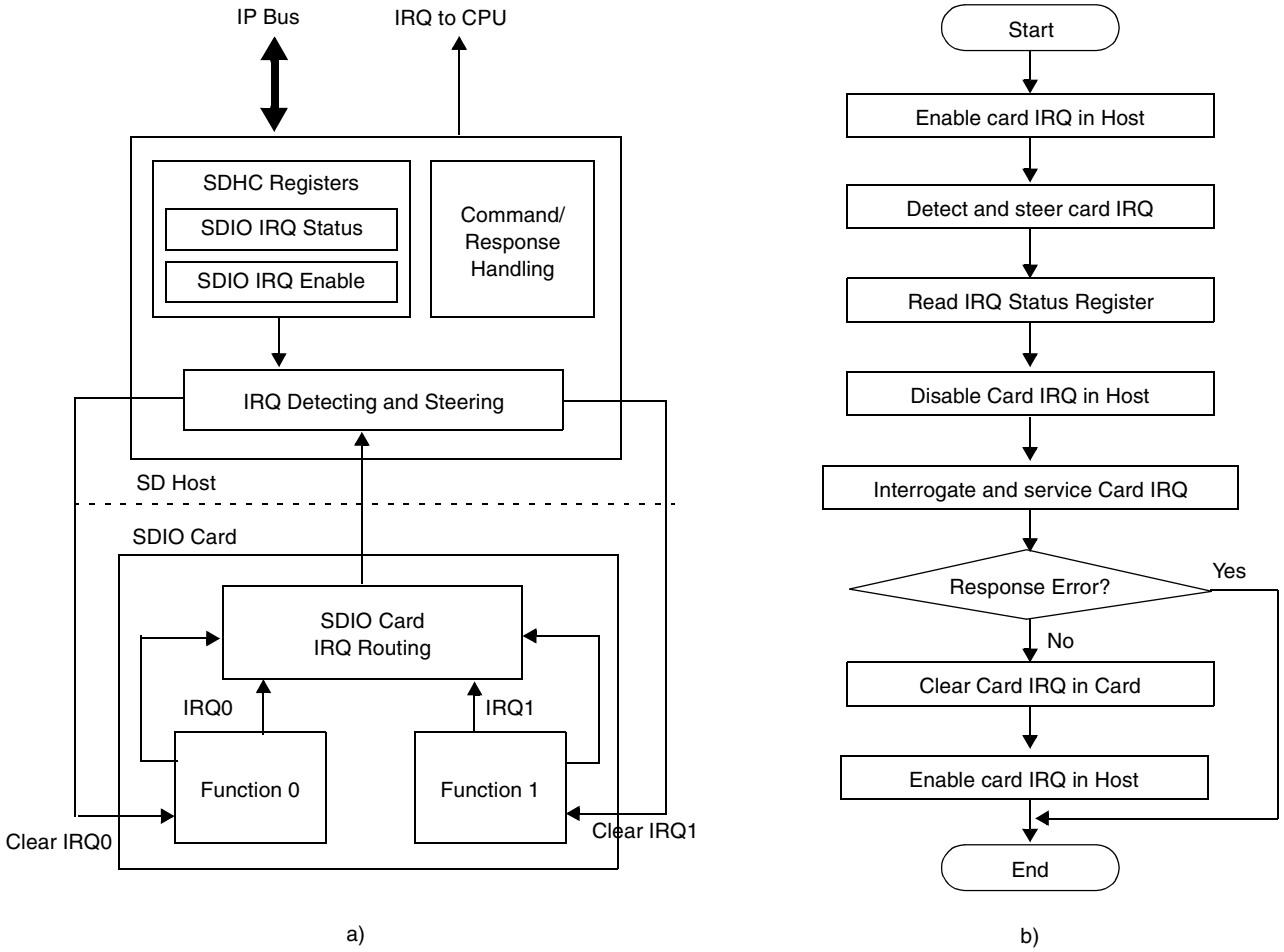


Figure 27-23. a) Card Interrupt Scheme; b) Card Interrupt Detection and Handling Procedure

27.4.5 Card Insertion and Removal Detection

SDHC uses the SD_DAT[3] pin to detect card insertion or removal. To utilize this feature of the SDHC, the chip level integration needs to pull down this pad as a default state. When there is no card on the MMC/SD bus, the SD_DAT[3] defaults to a low voltage level. When any card is inserted or removed from the socket, SDHC detects the logic value changes on the SD_DAT[3] pin and generates an interrupt.

Since the mechanism is based on the value of the SD_DAT[3] line, only single-card systems can benefit from card detection. To avoid the conflict of card insertion/removal detection and the data value changes on SD_DAT[3] because of data transfer, the user should disable the Card Insertion interrupt when there is a card detected in the socket but enable the interrupt when the card is removed from the socket. The card removal interrupt can be enabled only when there is no bus activity on SD_DAT[3].

To avoid the false status bit generation during data transfer, the card insertion/removal is masked by the corresponding interrupt enable bit in INT_CNTR register.

NOTE

The user can send a command (ACMD42 for SDMem or CMD52 for SDIO) to the card to disable the card internal pull-up resistor after card detection and identification. Since the SD protocol requires that the DAT line must be pulled up for data transfer, the user must disable the host side of the SD_DAT[3] pull-down feature and configure it as pull-up. In the meantime, if the card internal pull-up resistor is disabled, the card removal interrupt can not be detected through SD_DAT[3].

27.4.6 Power Management and Wake-Up Events

When there is no operation between SDHC and the card through SD bus, the user can disable the ipg_clk and ipg_perclk in chip level clock control module to save power. When the user needs to use SDHC or communicate with the card, he or she can enable the clock and perform the operation.

In some circumstances, when the clocks to SDHC are disabled, or when system is in low power mode, there are some events when the user needs to enable the clock and handle the event. These events are called wake-up interrupts. SDHC can generate these interrupts even if there are no clocks enabled. The three interrupts which can be used as wake-up events are:

- Card Removal Interrupt
- Card Insertion Interrupt
- SDIO card interrupt

The SDHC offers a power management feature. By clearing the clock-enabled bits in the Clock Control Register, the clocks are gated in the low position to the SDHC. For maximum power saving, the user can disable all the clocks to SDHC when there is no operation in progress.

While in this state, it is possible that interrupts can occur that require the SDHC to respond. These interrupts are called wake-up events and are defined as follows:

- Wake-Up Event on SD Card Removal through card removal interrupt
- Wake-Up Event on SD Card Insertion through card insertion interrupt
- Wake-Up Event on Card Interrupt through SDIO interrupt

These three wake-up events (or wake-up interrupts) can be also used to wake the system from low-power modes.

NOTE

To make the interrupt as a wake-up event when all the clocks to SDHC are disabled or when whole system is in low power mode, the corresponding wake-up enabled bit needs to be set. Refer to Section 27.3.3.10, “SDHC Interrupt Control Register (INT_CNTR)” for more information on SDHC Interrupt Control register.

NOTE

According to the SDIO specification, if the user wants the card interrupt to wake up from low power mode (no clock to SDIO card), the card should be set to 1-bit mode through CMD52.

27.4.6.1 Dynamic Voltage/Frequency Scaling (DVFS) Operation

Any change in ipg_perclk impacts the MMC/SD/SDIO transfer clock rate.

27.4.6.2 Setting Wake-Up Events

For the SDHC to respond to a wake-up event, the software must set the respective wake-up enable bit before the CPU enters sleep mode. Before the software disables the host clock, it should ensure that all of the following conditions have been met:

- No Read or Write Transfer is active.
- Data and Command lines are not active.
- No interrupts are pending.
- Internal FIFOs are empty.

The software is responsible to ensure that the clock to the SDHC is fully operational before making accesses to the peripheral.

27.4.7 Command/Data Interpreter

Command and Data Interpreters are based on similar principles. Both devices consist of three parts:

- Inner State Machine
- Sub-module controller
- CRC hardware accelerator

The CMD Interpreter handles everything related to Command Line (CMD). The CMD Interpreter includes Command data sequence generation, Command response extraction, CRC generation and checking, and a Response Time-out detection. To achieve the above functions, a state machine, a logic control, and a CRC accelerator are used. See Figure 27-24 for an illustration of the block diagram for the Command Interpreter.

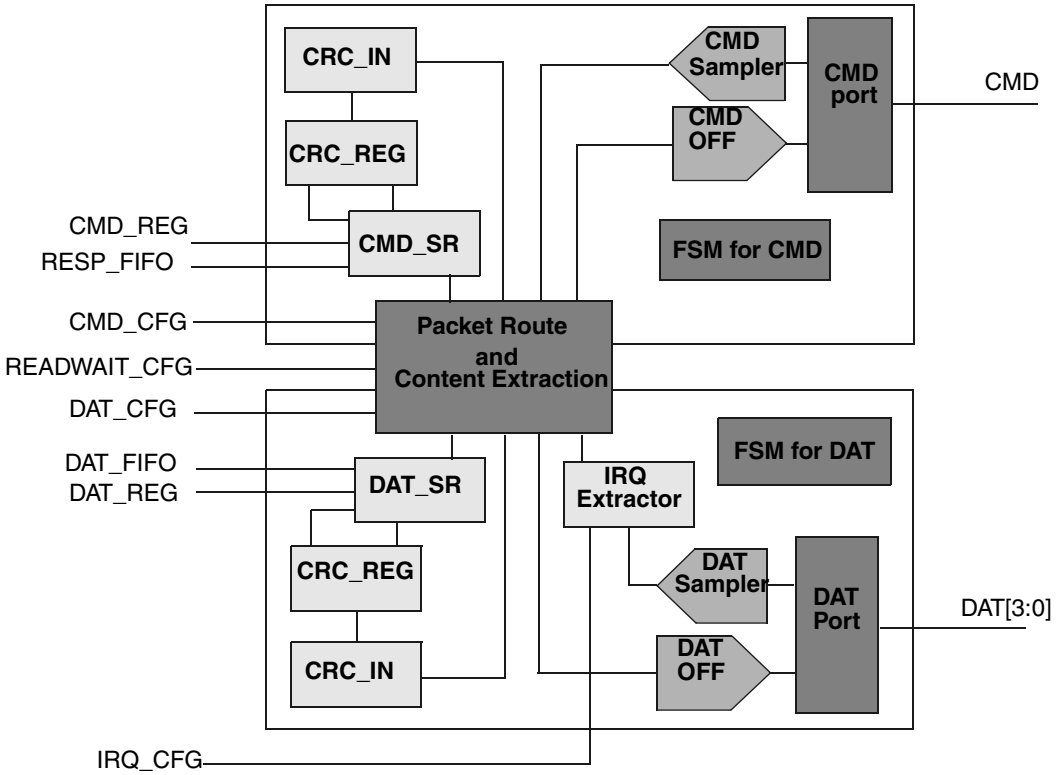


Figure 27-24. Block Diagram for Command Interpreter

See Figure 27-25 for an illustration of the structure for the Command CRC Shift Register.

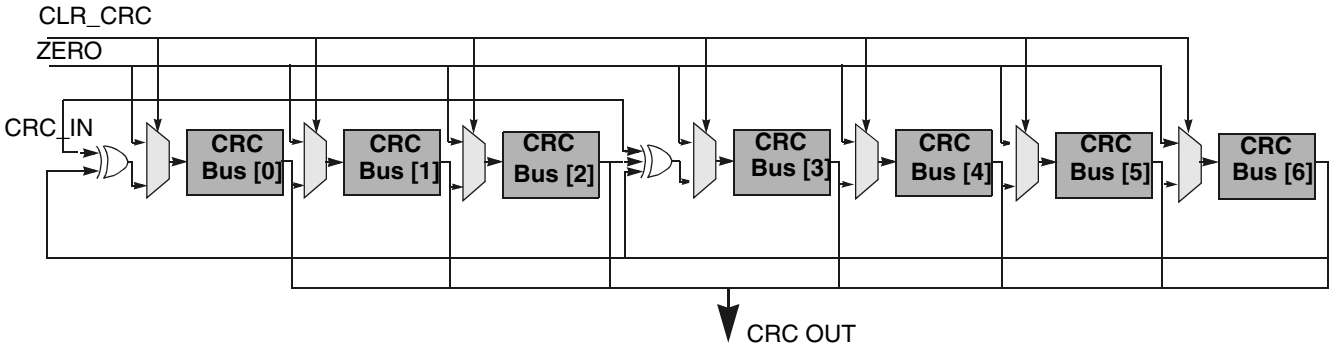


Figure 27-25. Command CRC Shift Register (DATs Has Similar Structure)

To minimize the gate count, the internal command shift register is re-used for the CRC shift register. The polynomials for the CMD and the DAT are as follows:

For the CMD:

Generator polynomial: $G(x) = x^7 + x^3 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit}) * x^0$
$CRC[6:0] = \text{Remainder } [(M(x) * x^7) / G(x)]$

For the DAT:

Generator polynomial: $G(x) = x^{16} + x^{12} + x^5 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit}) * x^0$
$CRC[15:0] = \text{Remainder } [(M(x) * x^{16}) / G(x)]$

27.4.8 System Clock Controller

There are two clock domain in SDHC. One is IPG_PERCLK and the other is IPG_CLK.

On the IPG_PERCLK domain, there is one clock divider and one clock prescaler in SDHC to divide the high frequency input clock IPG_PERCLK to a low frequency clock, which can be used by card and most of the logic of SDHC. See Figure 27-26. The input clock first goes through a 4-bit divider and then a 12-bit prescaler to generate a clock named CLK_20M. This clock is used internally by SDHC for DAT line control, CMD line control and almost all the functional logic. The MMC_SD_CLK to the card, which is a gated version of CLK_20M, has the same clock frequency as CLK_20M. CLK_20M is derived from the CLK_DIV by using the 12-bit prescaler. The CLK_DIV is derived from the input clock IPG_PERCLK by using the 4-bit divider. SDHC clock rate register controls the divide rate for both the divider and the prescaler. Refer to Section 27.3.3.3, “SDHC Clock Rate Register (CLK_RATE)” for the clock rate register information.

The IPG_CLK is of the MCU clock domain and used for SDHC registers/FIFO read write access.

To maximize power-saving during the operation, the SDHC bus clock pauses and resumes according to the SDHC status. For example, when the FIFO is full during the Card Read operation, the bus clock is stopped if no further data is written to the FIFO by the card. The bus clock is resumed when the FIFO empty status is cleared by the user (DMA). Also, there are other conditions where the SDHC stops the clock to save power.

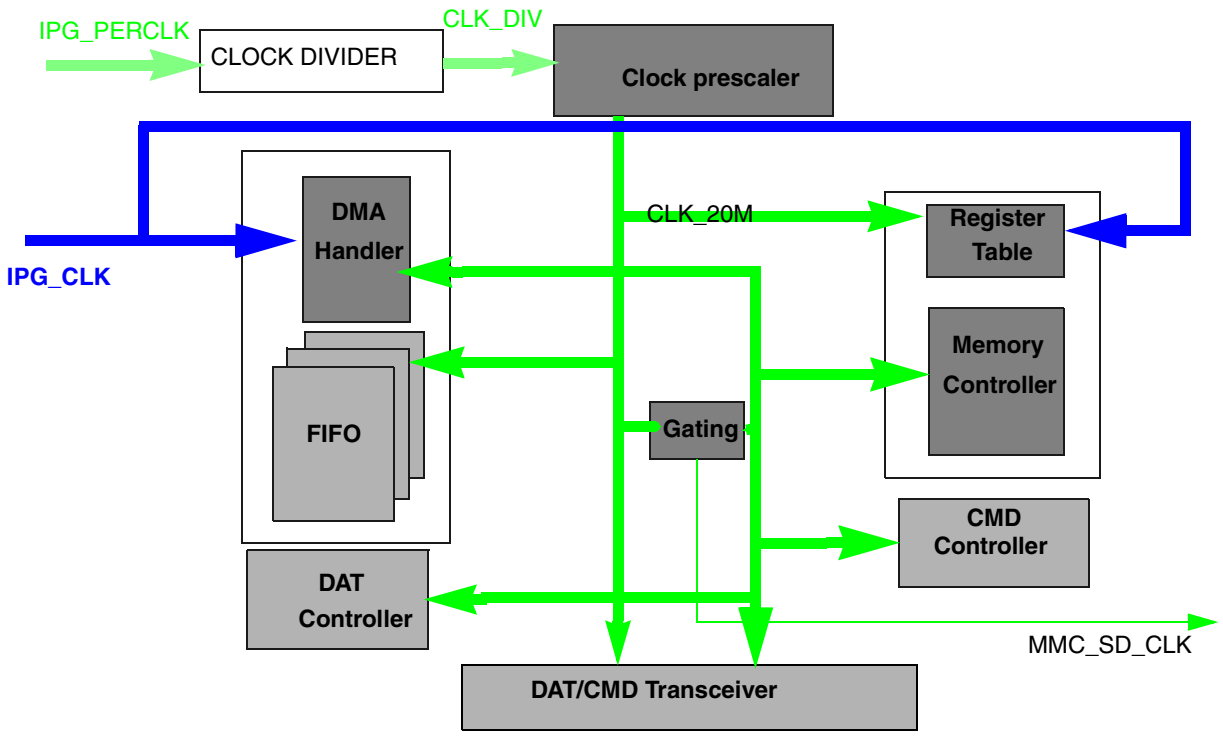


Figure 27-26. Clock Used in SDHC

The controller controls the rate of the card clock MMC_SD_CLK and checks whether it is on or off. The clock is turned off by setting the bit[0] of the STR_STP_CLK register and is turned on by setting the bit[1] of the STR_STP_CLK. To change the clock rate, the application must write a new value in the CLK_RATE register.

27.4.9 DAT/CMD Transceiver

The transceiver unit is designed to do the following:

- Control the I/O buffers.
- Synchronize the input data to the system clock domain.

The bi-directional signals CMD and DAT are each connected by OE, OEB, IN, and OUT.

OUT and OEB are used for the high-impedance state output buffer, while OE and IN are used for the input buffer. The use of OE allows the input to be disabled during floating and minimizes the current consumption.

The data buffers are in the system clock domain but the input data is in the MMC_SD_CLK clock domain. The transceiver will synchronize the input data to system clock domain.

27.5 Initialization/Application of SDHC

This section provides initialization and application information for SDHC.

All communication between system and cards are controlled by the host. The host sends commands of two types: broadcast and addressed (point-to-point) commands.

Broadcast commands are intended for all cards, such as: “Go_Idle_State”, “Send_Op_Cond”, “All_send_CID” and “Set_relative_Addr”. In Broadcast mode, all cards are in the open-drain mode to avoid bus contention. If the socket support only one card, the broadcast command is similar as the point-to-point command.

After the Broadcast command “Set_relative_Addr” is issued, the cards enter standby mode. Addressed type commands are used from this point. In this mode, the CMD/DAT I/O will return to push-pull mode, to have the driving capability for maximum frequency operation.

The MMC and the SD are similar products. Other than the 4x bandwidth, they are being programmed similarly. The following example will show how to “initialize” and perform “content access” and “content protection” on the cards.

To improve the readability, we are going to use a program-like function for Example 27-1.

Example 27-1. MMC_SD_CLK Control

The MMC_SD_CLK clock to the card is controlled by STR_STP_CLK register. The clock should be supplied to the card for:

- Submitting command to card and receive response
- Transferring data between SDHC and the card
- Detecting an interrupt from a SD card in 4-bit

The steps below show how to start the MMC_SD_CLK to card:

1. Write 0x2 to STR_STP_CLK register.
2. Polling STATUS[8], wait until clock starts.

The steps below show how to stop MMC_SD_CLK to card:

1. Write 0x1 to STR_STP_CLK register.
2. Polling STATUS[8], wait until clock is stopped.

NOTE

The user should not change the ipg_clk_gating_disable and ipg_perclk_gating_disable bits when start and stop MMC_SD_CLK. And if the SDHC clock gating features is used by

27.5.1 Command Submit—Response Receive Basic Operation

This section shows the program flow used to submit a command to the card(s). The commands are as follows:

- <command_no>—the targeted command
- <arg_no>—the corresponding argument

- <cmd_dat_cont>—the command configuration required
- <int_Control_value>—the interrupt control used in the user program.

The steps below show how to submit a command to the card:

1. Start MMC_SD_CLK if it is stopped
2. Enable END_CMD_RESP interrupt by write 0 to INT_CNTR[2].
3. Set command number to CMD register.
4. Set the command argument to ARG register.
5. Set the appropriate value to Command Data control register (CMD_DAT_CONT).
6. Wait for the end command response interrupt and check for the response CRC/time-out status.
7. Read the response from the response FIFO to check the response. Read three or eight times from the response FIFO access register, depending upon whether the response is 48-bit or 136-bit.
8. Stop the MMC_SD_CLK if the clock is not needed (if there is data transfer following the command/response transfer, the clock should not be stopped until the data transfer completes).

This following is a function defining command submission. This function will be used in the examples in the following subsection:

```
send_cmd_wait_resp(command_no, arg, cmd_dat_cont, int_cntr_value)
{
write_reg(STR_STP_CLK, 0x02);//1. to start mmc_sd_clk
read_reg(STATUS);
while(!STATUS[8]) Read_reg(STATUS); // 2. Wait till the clock has started
write_reg(COMMAND, <command_no>);// 3. configure the CMD
write_reg(ARG, <arg_no>);//4. configure the command argument
write_reg(CMD_DAT_CONT, <cmd_dat_cont>);//5. configure the command data control register,
writing to this register will trigger SDHC send command to the card.
while(irq_status);// 6. Wait interrupt (End Command Response)
Write_reg(INT_CNTR, <int_cntr_value>);//7. negate the irq request from SDHC
read_reg(STATUS); //8. Check whether the interrupt is an End_CMD_RES or a response time out or a
CRC error.
write_reg(STR_STP_CLK, 0x001);// 9. Stop the card clock if the clock is not needed any longer for
this cmd
read_reg(STATUS);
while(STATUS[8]) Read_reg(STATUS); // 10. Wait till the clock is stopped, command - response end.
read_reg(RES_FIFO); // 11. read the response fifo to determine if the command has a response
}
```

27.5.2 Card Identification Mode

When a card is inserted to the socket or the card was reset by the host, the host needs to validate the operation voltage range, identify the cards, and request the cards to publish the Relative Card Address (RCA) or to set the RCA for the MMC cards. All data communications in the Card Identification mode use the command line (CMD) only.

27.5.2.1 Card Detect

See Figure 27-27 for flow diagram showing the detection of card using the host controller.

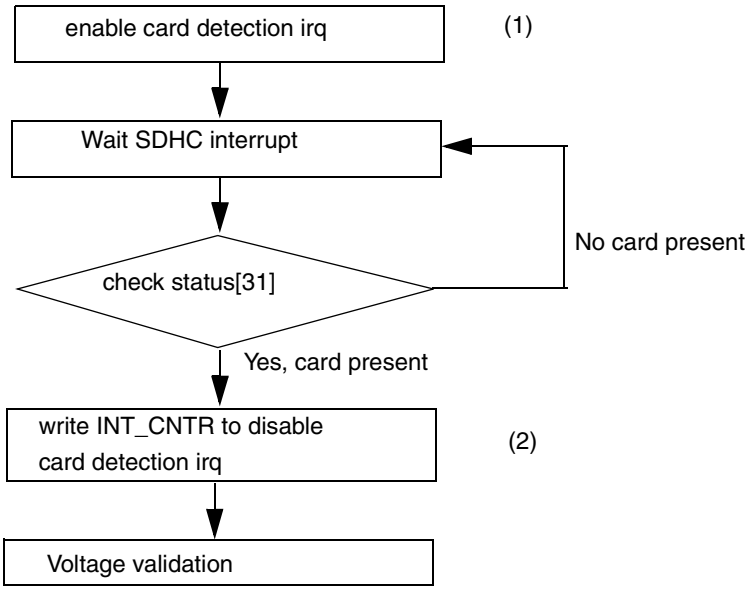


Figure 27-27. Flow Diagram for Card Detection

- Write ‘1’ to INT_CNTR[15] to enable card detection interrupt
- Write ‘0’ to INT_CNTR[15] to disable card detection interrupt

27.5.2.2 Reset

The host consists of three types of reset:

- Hardware reset (card and host), which is driven by POR (power on reset).
- Software reset (host Only), which is preceded by the write operation on register “STR_STP_CLK”. Follow the recommended sequence as specified in Section 27.3.3.3, “SDHC Clock Rate Register (CLK_RATE).” The reset will reset all the SDHC registers, but will not reset the card. The card reset is through CMD0. Once the user applies the software reset to SDHC, it should also be using CMD0 to reset the card in case the card is in unknown state.
- Card reset (card only). The command, `Go_Idle_State`, CMD0 is the software reset command for both the MMC and the SD Memory Card. This sets each card into Idle State regardless of the current card state. When used as a SD I/O card, CMD52 is used to write IO reset in CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. See Figure 27-28 for the software flow to reset both SDHC and the card.

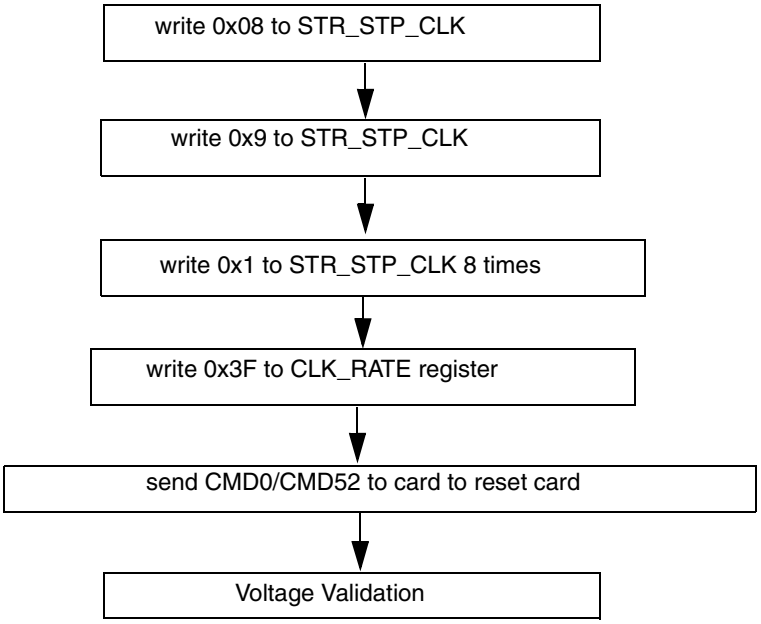


Figure 27-28. Flow Chart for Reset of SDHC and SD I/O Card

```
software_reset()
{
write_reg(STR_STP_CLK, 0x8);
write_reg(STR_STP_CLK, 0x9);// 1. reset the SDHC host;
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);// 2. write 0x1 to STR_STP_CLK 8 times;
write_reg(CLK_RATE, 0x3F);// 3. Set the lowest clock for initialization
write_reg(READ_TO, 0x2DB4);// 4. set READ timeout register
send_cmd_wait_resp(CMD_GO_IDLE_STATE, 0x0,0x80, 0x40); //5. reset the card with CMD0
}
```

27.5.2.3 Voltage Validation

All cards must be able to establish communications with the host using any operation voltage in the maximum allowed voltage range specified in this standard. However, the supported minimum and maximum values for Vdd are defined in Operation Conditions Register (OCR) and might not cover the entire range. Cards that store the CID and CSD data in the preload memory are able to communicate the information only under data transfer Vdd conditions. That means if the host and card have non-compatible Vdd ranges, the card will not be able to complete the identification cycle, nor be able to send CSD data.

Commands such as `Send_Op_Cont` (CMD1 for MMC), `SD_Send_Op_Cont` (CMD41 for SD Memory), and `IO_Send_Op_Cont` (CMD5 for SD I/O) are designed to provide a mechanism to identify and reject cards that

do not match the Vdd range desired by the host. This is accomplished by the host sending the required Vdd voltage window as the operand of this command. Cards that can not perform data transfer in the specified range must detach themselves from further bus operations and go into the inactive state. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the inactive state. This query should be used if the host is able to select a common voltage range or if a notification to the application of non-usable cards in the stack is desired.

The following steps show how to perform voltage validation when a card inserted:

```
voltage_validation(voltage_range_arguement)
{
send_cmd_wait_resp(IO_SEND_OP_COND, 0x0, 0x04, 0x40); // CMD5, send SDIO operation voltage,
command argument is zero
if(End Command Response true & No. of IO functions> 0)// it is SDIO and have IO function
{IORDY = 0;
while(!(IORDY in I/O ORC response)) {// set voltage range for each IO
send_cmd_wait_resp(IO_SEND_OP_COND, voltage_range_arguement, 0x04, 0x40);}
if(Memory Present flag true)
Card = combo; // that is, SDIO + SD Memory, need to set operation voltage to memory portion as
well
send_cmd_wait_resp(APP_CMD, 0x0, 0x01, 0x40);// CMD55, Application Command follows
send_cmd_wait_resp(SD_APP_OP_COND, voltage_range_arguement, 0x01, 0x40);//ACMD41
else
Card = sdio;

// if No response to CMD5 IO_SEND_OP_COND or No. of IO Function is zero in response
else// the card should be SD or MMC
{send_cmd_wait_resp(APP_CMD, 0x0, 0x01, 0x40);// CMD55, Application Command follows
if(End Command Response true and no response timeout)
{send_cmd_wait_resp(SD_APP_OP_COND, voltage_range_arguement, 0x01, 0x40); // ACMD41, SD card
found
Card = sd;
}
else // the card have no response to APP_CMD, it is not SD card
{send_cmd_wait_resp(SEND_OP_COND, voltage_range_arguement, 0x01, 0x40); //CMD1, MMC card found
if(End Command Response true and no response timeout)
{Card = mmc;}
else{ Card = No card or failed contact;}
}
}
```

27.5.2.4 Card Registry

Card registry between MMC and SD card is different.

For the SD card, the identification process starts at clock rate Fod (below 400 kHz for most of the card) as defined by the card specification. After the bus is activated, the host requests the card to send valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command is sent to all of the new cards in the system. Incompatible cards are put into the inactive state. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card

identification (CID) number. Cards that are currently unidentified (that is, in Ready State), send their CID number as the response. After the CID is sent by the card, the card goes into the identification state.

The host then issues `Send_Relative_Addr` (CMD3), requesting the card to publish a new relative card address (RCA), which is shorter that CID. This ID is used to address the card for future data transfer operations. Once the RCA is received, the card state changes to the stand-by state. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system.

For the MMC operation, the host starts the card identification process in open-drain mode with the identification clock rate Fod. (Fod is the initialization clock frequency defined by the card specification.) The open-drain driver stages on the CMD line allow parallel card operations during card identification. After the bus is activated, the host requests the cards to send their valid operation conditions (CMD1). The response to CMD1 is the ‘wired OR’ operation on the condition restrictions of all cards in the system. Incompatible cards are sent into inactive state. The host then issues the broadcast command `All_Send_CID` (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards (that is, those which are in Ready State) simultaneously start sending their CID numbers serially, while bit-wise is monitoring their outgoing bitstream. These cards, for which outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CIDs immediately. They must wait for the next identification cycle. Since the CID is unique for each card, only one card can successfully send its full CID to the host. This card then goes into identification state. Thereafter, the host issues `Set_Relative_Addr` (CMD3) to assign a relative card address (RCA) to this card. Once the RCA is received, the card state changes to the stand-by state, does not react to further identification cycles, and its output switches from open-drain to push-pull. The host repeats the process, that is, CMD2 and CMD3, until the host receives the timeout condition to recognize completion of the identification process.

```
card_registry()
{
while (ResponseTO from STATUS){
if(card==combo or sdio)
{
send_cmd_wait_resp(SET_RELATIVE_ADDR, 0x00, 0x01, 0x40);   //card publish the RCA in response
rca = SDIO_RCA = address from response FIFO;
}
else if(card==sd)
{
send_cmd_wait_resp(ALL_SEND_CID, 0x00, 0x02, 0x40);
send_cmd_wait_resp(SET_RELATIVE_ADDR, 0x00, 0x01, 0x40); //card publish the RCA in response
rca = SD_RCA = address from response FIFO;
}
else if(card==mmc)
{
send_cmd_wait_resp(ALL_SEND_CID, 0x00, 0x00, 0x02, 0x40);
rca = MMC_RCA = 0x1;
send_cmd_wait_resp(SET_RELATIVE_ADDR, MMC_RCA_arguement, 0x01, 0x40);
}
```

```
else
exit due to card not identified;
}
send_cmd_wait_resp(SELECT_CARD, RCA_arguement, 0x41, 0x40);
}
```

27.5.3 Card Access

27.5.3.1 Block Access—Block Write and Block Read

27.5.3.1.1 Block Write

During block write (CMD24–27), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. A card supporting block write will always be able to accept a block of data defined by WRITE_BL_LEN. If the CRC fails, the card shall indicate the failure on the DAT line. The transferred data is discarded and not written, and all further transmitted blocks (in multiple-block write mode) will be ignored.

If the host uses partial blocks for which accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and, while ignoring all further data transfer, waits in the Receive-data-State for a stop command. The write operation is aborted if the host tries to write over a write-protected area. In this case, however, the card sets the WP_VIOLATION bit.

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC-protected. If a part of the CSD or CID register is stored in ROM, this unchangeable part must match the corresponding part of the receive buffer. If this match fails, the card reports an error and does not change any register contents. Some cards may require long and unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the card begins writing and holds the DAT line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, then the card responds with its status. The status bit READY_FOR_DATA indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing CMD7 (to select a different card), which places the card into the disconnect state and releases the DAT line without interrupting the write operation. When re-selecting the card, it reactivates the busy indication by pulling DAT to low if programming is still in progress and the write buffer is unavailable.

The software flow to write to the card with DMA enable is:

1. Start MMC_SD_CLK if it is stopped.
2. Check the card status, wait until card is ready for data.
3. For SD/MMC, set the card block length, using SET_BLOCKLEN (CMD16).
4. Set the SDHC block length register to be same as block length set to the card in Step 2.

— For SDIO, if the CMD53 is in byte mode, the SDHC block length register should be set according to bytes count in CMD53; if the CMD53 is in block mode, the SDHC block length register should be set according to the block size in CCCR registers.

5. Set SDHC number block register (NOB), nob is 1 for single block write or CMD53 in byte mode for SDIO.
6. Disable the buffer ready interrupt, configure the DMA setting and enable the SDHC DMA channel:
   a) Write '0' to bit[3] of INT_CNTR register in SDHC to disable buffer write ready interrupt.
   b) Set DMA destination to be SDHC_Buffer Access register.
   c) Set DMA destination port size to be 32-bit.
   d) Set DMA Burst length to be 16 bytes in 1-bit mode or 64 bytes in 4-bit mode.
   e) Set DMA transfer count to be number of bytes which is a multiple of the Block_length (nob*blk_len = total number of bytes).
7. Check the card status and wait until the card is ready for data.
8. Set SDHC CMD register to any of the following:
   — CMD24(WRITE_BLOCK)
   — CMD25(WRITE_MULTIPLE_BLOCK)
   — CMD53 in byte mode or block mode
9. Set SDHC CMD Argument register.
10. Set SDHC Command Data Control register.
11. Wait for end command response and check if there any CRC error or timeout error.
12. Wait for DMA done.
13. Check for Write_OP_DONE and check status bit to see if write CRC error occurred.
14. Send STOP_TRANSMISSION command to the card if the write command is WRITE_MULTIPLE_BLOCK (CMD25).
15. Stop the MMC_SD_CLK, finished the write operation. (This step is optional.)

If the write operation is without DMA, the system needs to write data to the buffer through buffer write ready interrupt or by polling the buffer write ready status bit (STATUS[6]: BUF_WR_RDY). For high performance, data transfer using DMA is preferred.

27.5.3.1.2 Block Read

For block reads, the basic unit of data transfer is a block for which the maximum size is defined in the CSD (READ_BL_LEN). If READ_BL_PARTIAL is set, smaller blocks for which the starting and ending addresses are entirely contained within one physical block (as defined by READ_BL_LEN) may also be transmitted. A CRC is appended to the end of each block, ensuring data transfer integrity. CMD17 (READ_SINGLE_BLOCK) initiates a block read. After completing the transfer, the card returns to the transfer state. CMD18 (READ_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. Blocks is continuously transferred until a stop command is issued. If the host uses partial blocks for which accumulated length is not block aligned and block misalignment is not allowed, the card detects a block misalignment at the beginning of the first mis-aligned block, sets the ADDRESS_ERROR error bit in the status register, aborts transmission, and waits in the data state for a STOP command.

The software flow to write to card with DMA enable is:

1. Start MMC_SD_CLK if it is stopped.
2. Check the card status and wait until the card is ready for data.
3. For SD/MMC, set the card block length, using SET_BLOCKLEN (CMD16).
4. Set the SDHC block length register to be same as block length set to the card in step 3.
   — For SDIO, if the CMD53 is in byte mode, the SDHC block length register should be set according to bytes count in CMD53; if the CMD53 is in block mode, the SDHC block length register should be set according to the block size in CCCR registers.
5. Set SDHC number block register (NOB) to 1 for single block write or CMD53 in byte mode for SDIO.
6. Disable the buffer ready interrupt, configure the DMA setting, and enable the SDHC DMA channel:
   a) Write ‘0’ to bit[4] of INT_CNTR register in SDHC to disable the buffer read ready interrupt.
   b) Set DMA source to be SDHC_Buffer Access register.
   c) Set DMA source port size to be 32-bit.
   d) Set DMA burst length to be 16 bytes in 1-bit mode or 64 bytes in 4-bit mode.
   e) Set DMA transfer count to be number of bytes that is a number of blocks multiple of the Block_length (nob*blk_len).
7. Check the card status and wait until the card is ready for data.
8. Set SDHC CMD register to be CMD17(READ_SINGLE_BLOCK) or CMD18 (READ_MULTIPLE_BLOCK) or CMD53 in byte mode or block mode.
9. Set SDHC CMD Argument register.
10. Set SDHC Command Data Control register.
11. Wait for END_CMD_RESP interrupt and check response FIFO, check CRC error and timeout error.
12. Wait for DMA done.
13. Check for READ_OP_DONE and check status bit to see if read CRC error occurred.
14. Send STOP_TRANSMISSION command to the card if the read command is READ_MULTIPLE_BLOCK (CMD18).
15. Stop the MMC_SD_CLK, finished the read operation.

If the read transfer operation does not use DMA, the system will need to fetch data out of the data buffer through utilizing the buffer read ready interrupt or by polling the buffer read ready status bit (STATUS[7]: BUF_READ_RDY). For high performance, data transfer using DMA is preferred.

27.6 Commands for MMC/SD/SDIO

See Table 27-21 for the list of commands for MMC/SD/SDIO.

Refer to corresponding card specifications for details about the command information.

Table 27-21. Commands for MMC/SD/SDIO

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	—	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.
CMD3	ac	[31:6] RCA [15:0] stuff bits	R1 R6(SDIO)	SET_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	—	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SD I/O cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD6	Reserved				
CMD7	ac	[31:6] RCA [15:0] stuff bits	R1b	SELECT/DESELECT_CARD	Command toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address; address 0 deselects all.
CMD8	Reserved				
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	MMC Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the MMC/SD Memory card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed MMC/SD card sends its status register.
CMD14	Reserved				

Table 27-21. Commands for MMC/SD/SDIO (continued)

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CMD15	ac	[31:6] RCA [15:0] stuff bits	—	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	Reserved				
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	MMC card writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows.
CMD21–23	Reserved				
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the MMC card identification register. This command shall be issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally, this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write-protection features, this command sets the write-protection bit of the addressed group. The properties of write-protection are coded in the card specific data (WP_GRP_SIZE).

Table 27-21. Commands for MMC/SD/SDIO (continued)

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write-protection features, this command clears the write-protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write-protection features, this command asks the card to send the status of the write-protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	ERASE_WR_BLK_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	ERASE_WR_BLK_END	Sets the address of the last sector of the continuous range of the erase group.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erases all previously selected sectors.
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command address a card and a register and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command accesses application dependent registers which are not defined in MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				

Table 27-21. Commands for MMC/SD/SDIO (continued)

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CDM42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43~51	Reserved				
CMD52	—	[31] R/W flag [30:28] Function Number [27] RAW (read after write) flag [26] stuff bit [25:9] Register Address [8] Stuff bit [7:0] Write Data/stuff bits	R5	IO_RW_DIRECT	Used to access a single register within the total 128 Kbytes of register space in any I/O function.
CMD53	—	[31] R/W flag [30:28] Function Number [27] Block mode [26] OP Code [25:9] Register Address [8:0] Byte/Block Count	R5	IO_RW_EXTENDED	Used to access a multiple I/O register with a single command, It allows the reading or writing of a large number of I/O registers.
CMD54	Reserved				
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application-specific command rather than a standard command.
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general-purpose/application-specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57~63	Reserved				
ACMDs are preceded with APP_CMD command. (Command listed below are used for SD only. Other unlisted SD commands are not supported in this module.)					

Table 27-21. Commands for MMC/SD/SDIO (continued)

CMD INDEX	Type	Argument	Response	Abbreviation	Description
ACMD6	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the SD Memory Card data bus width (‘00’=1-bit or ‘10’=4-bit bus) to be used for data transfer. The allowed data bus widths are given in the SCR register.
ACMD13	adtc	[31:0] stuff bits	R1	SD_STATUS	Sends the SD Memory Card status.
ACMD22	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Sends the number of the written (without errors) sectors. Responds with 32-bit + CRC data block.
ACMD23	ac	[31:23] stuff bits [22:0] Number of blocks	R1	SET_WR_BLK_ERASE_COUNT	Set the number of write blocks to be pre-erased before writing (to be used for faster Multi-block write command).
ACMD41	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating condition register (OCR) content in the response on the CMD line.
ACMD42	ac	[31:1] stuff bits [0] set_cd	R1	SET_CLR_CARD_DETECT	Connects/disconnects the 50 kΩ pull-up resistor on CD/DAT3 of the card.
ACMD51	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).

Chapter 28
Universal Asynchronous Receiver/Transmitters (UART)

The Universal Asynchronous Receiver/Transmitter (UART) module is capable of standard RS-232 non-return-to-zero (NRZ) encoding format and IrDA-compatible infrared modes. The UART provides serial communication capability with external devices through an RS-232 cable or through use of external circuitry that converts infrared signals to electrical signals (for reception) or transforms electrical signals to signals that drive an infrared LED (for transmission) to provide low speed IrDA compatibility.

Figure 28-1 shows the UART block diagram.

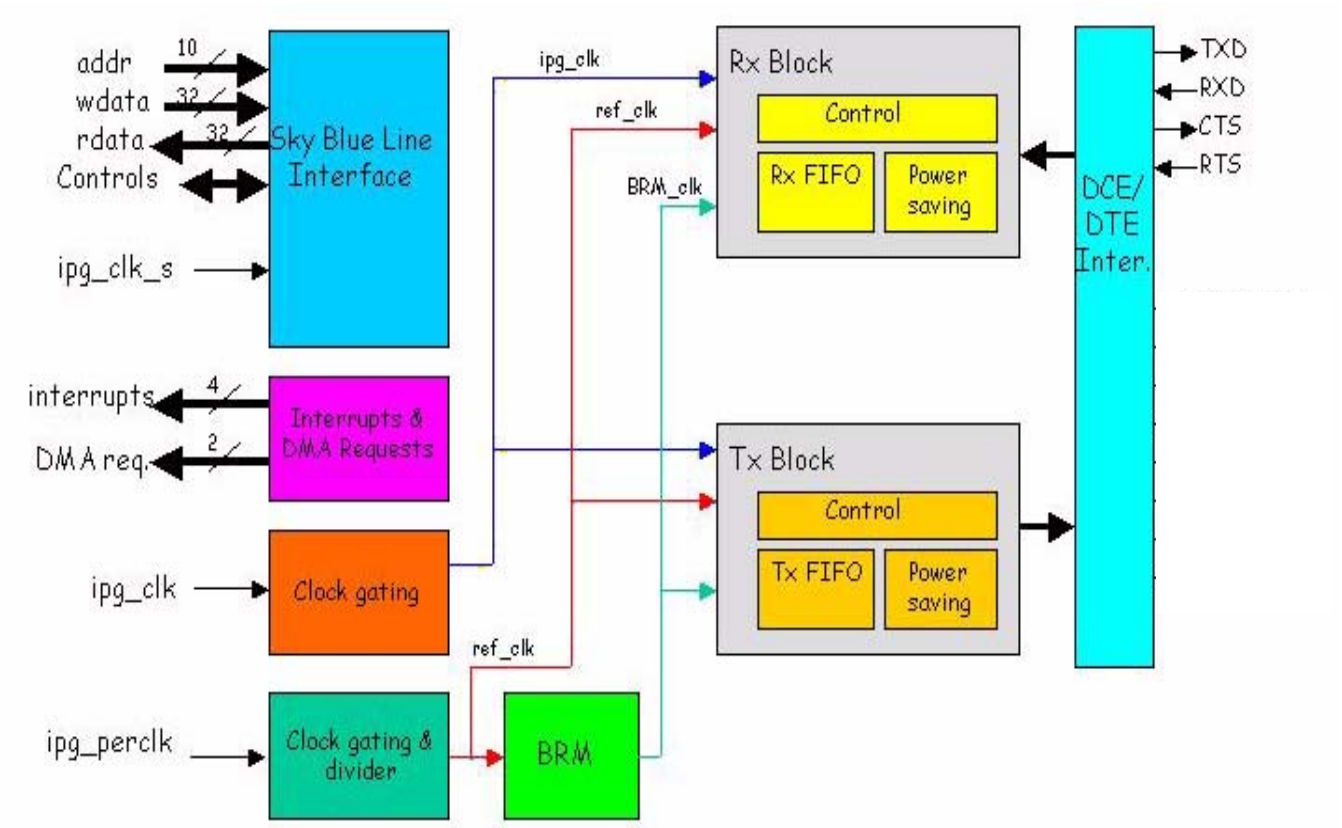


Figure 28-1. UART Block Diagram

28.1 Overview

The UART transmits and receives characters that are either 7 or 8 bits in length (program selectable). To transmit, data is written from the peripheral data bus to a 32-byte transmitter FIFO (TxFIFO). This data is passed to the shift register and shifted serially out on the transmitter pin (TXD). To receive, data is received serially from the receiver pin (RXD) and stored in a 32-half-words-deep receiver FIFO (RxFIFO). The

received data is retrieved from the RxFIFO on the peripheral data bus. The RxFIFO and TxFIFO generate maskable interrupts as well as DMA Requests when the data level in each of the FIFO reaches a programmed threshold level.

The UART generates baud rates based on a programmable divisor and input clock. The UART also contains programmable auto baud detection circuitry to receive 1 or 2 stop bits as well as odd, even, or no parity. The receiver detects framing errors, idle conditions, BREAK characters, parity errors, and overrun errors.

The UART module uses a software interface for control of modem operations and have a serial infrared (IR) module that decodes and encodes IrDA-compatible serial IR data.

28.1.1 Features

The UART includes the following features:

- High speed TIA/EIA-232-F compatible, up to 4.125Mbit/s
- 7 or 8 data bits
- 1 or 2 stop bits
- Programmable parity (even, odd, and no parity)
- Hardware flow control support for request to send (RTS) and clear to send (CTS) signals
- Edge selectable RTS and edge detect interrupts
- Status flags for various flow control and FIFO states
- Serial IR interface low speed, IrDA-compatible (up to 115.2 kbit/s).
- Voting logic for improved noise immunity (16x oversampling)
- Transmitter FIFO empty interrupt suppression
- UART internal clocks enable/disable
- Auto baud rate detection (up to 115.2 kbit/s)
- Receiver and transmitter enable/disable for power saving
- RTS, IrDA asynchronous wake (AIRINT), receive asynchronous wake (AWAKE), interrupts wake the MCU from Sleep Mode
- Maskable interrupts
- Two DMA Requests (TxFIFO DMA Request and RxFIFO DMA Request)
- Escape character sequence detection
- Software reset ($\overline{\text{SRST}}$)
- Dedicated BRM clock (ipg_perclk) to allow frequency scaling on main clock (ipg_clk) without reprogramming BRM registers

28.1.2 Modes of Operation

The following are the UART modes of operation:

- Serial RS-232 NRZ format
- IrDA

28.2 External Signal Description

28.2.1 Overview

Figure 28-1 provides external signal descriptions.

Table 28-1. Interface Signals

Signal Name	I/O	Active state	Description	Reset State
Reset				
ipg_hard_async_reset_b	I	Low	Asynchronous reset	
Interrupts				
ipi_uart_rx_b	O	Low	Receiver interrupt	High
ipi_uart_tx_b	O	Low	Transmitter interrupt	High
ipi_uart_mint_b	O	Low	Common interrupt	High
ipi_uart_anded_b	O	Low	Anded interrupt (see below for comment)	High
DMA Requests				
ipd_uart_rx_dmareq_b	O	Low	Receiver DMA request	High
ipd_uart_tx_dmareq_b	O	Low	Transmitter DMA request	High
Serial/IrDA Signals				
ipp_uart_rxd_mux	I		Serial/infrared data receive	
ipp_uart_txd_mux	O		Serial/infrared data transmit	High
Modem Control Signals				
ipp_uart_cts_b	O	Low	Clear to send	High
ipp_uart_rts_b	I	Low	Request to send	
Clocks				
ipg_clk	I		Main clock	
ipg_clk_s	I		Bus clock	
ipg_perclk	I		Binary Rate Multiplier (BRM) clock	

28.3 Memory Map and Register Definition

28.3.1 Memory Map and Register Summary

The UART supports 8-bit and 16-bit accesses to 32-bit memory-mapped addresses only.

All the memory mapped registers are 32 bits wide, however as the 16 MSB are not used:

- For 32-bits write access, the 16 MSB will not be taken into account.

- For 32-bits read access the 16 MSB will be read as 0.

28.3.2 Memory Map

Table 28-2 shows the memory map.

Table 28-2. UART Memory Map

Address	Register	Access	Reset Value	Section/Page
0x1000_A000 (URXD1) – 0x1001_C000 (URXD6)	UART Receiver Registers 1–6	R	0x8000	28.3.4.1/28-7
0x1000_A040 (UTXD1) – 0x1001_C040 (UTXD6)	UART Receiver Registers 1–6	W	0x00– –	28.3.4.2/28-9
0x1000_A080 (UCR1_1) – 0x1001_C080 (UCR1_6)	UART Control Registers 1_1–1_6	R/W	0x0000	28.3.4.3/28-9
0x1000_A084 (UCR2_1) – 0x1001_C084 (UCR2_6)	UART Control Registers 2_1–2_6	R/W	0x0001	28.3.4.4/28-11
0x1000_A088 (UCR3_1) – 0x1001_C088 (UCR3_6)	UART Control Registers 3_1–3_6	R/W	0x0700	28.3.4.5/28-14
0x1000_A08C (UCR4_1) – 0x1001_C08C (UCR4_6)	UART Control Registers 4_1–4_6	R/W	0x8000	28.3.4.6/28-15
0x1000_A090 (UFCR1) – 0x1001_C090 (UFCR6)	UART FIFO Control Registers 1–6	R/W	0x0801	28.3.4.7/28-17
0x1000_A094 (USR1) – 0x1001_C094 (USR6)	UART Status Registers 1_1–1_6	R/W	0x2040	28.3.4.8/28-18
0x1000_A098 (USR2_1) – 0x1001_C098 (USR2_6)	UART Status Registers 2_1–2_6	R/W	0x4008	28.3.4.9/28-20
0x1000_A09C (UESC1) – 0x1001_C09C (UESC6)	UART Escape Character Registers 1–6	R/W	0x002B	28.3.4.10/28-22
0x1000_A0A0 (UTIM1) – 0x1001_C0A0 (UTIM6)	UART Escape Timer Registers 1–6	R/W	0x0000	28.3.4.11/28-23
0x1000_A0A4 (UBIR1) – 0x1001_C0A4 (UBIR6)	UART BRM Incremental Registers 1–6	R/W	0x0000	28.3.4.12/28-23
0x1000_A0A8 (UBMR1) – 0x1001_C0A8 (UBMR6)	UART BRM Modulator Registers 1–6	R/W	0x0000	28.3.4.13/28-24

Table 28-2. UART Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x1000_A0AC (UBRC1) – 0x1001_C0AC (UBRC6)	UART Baud Rate Count Registers 1–6	R	0x0000	28.3.4.14/28-24
0x1000_A0B0 (ONEMS1) – 0x1001_C0B0 (ONEMS6)	UART One Millisecond Registers 1–6	R/W	0x0000	28.3.4.15/28-25
0x1000_A0B4 (UTS1) – 0x1001_C0B4 (UTS6)	UART Test Registers 1–6	R/W	0x0060	28.3.4.16/28-26

28.3.3 Register Summary

The conventions in Figure 28-3 and Table 28-3 serve as a key for the register summary and individual register diagrams.

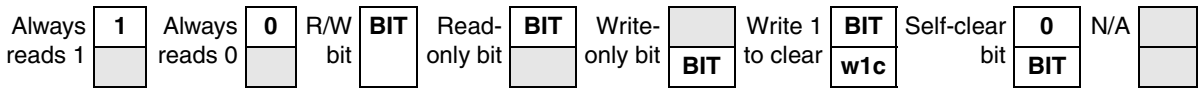


Figure 28-2. Key to Register Fields

Table 28-3 provides a key for register figures and tables and the register summary.

Table 28-3. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero (previously designated slfclr).
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

All registers described in this section are for 16 LSB.

Figure 28-3 shows the key to the register fields.

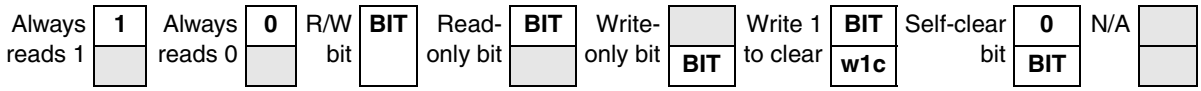


Figure 28-3. Key to Register Fields

Table 28-4. UART Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000_A000 (URXD1) – 0x1001_C000 (URXD6)	R	0	ERR	OVRRUN	FRMERR	BRK	PRERR	0	0	RX_DATA							
	W																
0x1000_A040 (UTXD1) – 0x1001_C040 (UTXD6)	R																
	W	0	0	0	0	0	0	0	0	TX_DATA							
0x1000_A080 (UCR1_1) – 0x1001_C080 (UCR1_6)	R	ADEN	ADBR	TRDYEN	IDEN	ICD		RRDYEN	RXDMAEN	IREN	TXMPTYEN	RTSDEN	SNDBRK	TXDMAEN	0	DOZE	UARTEN
	W																
0x1000_A084 (UCR2_1) – 0x1001_C084 (UCR2_6)	R	ESCI	IRTS	CTSC	CTS	ESCEN	RTEC		PREN	PROE	STPB	WS	RTSEN	ATEN	TXEN	RXEN	SRST
	W																
0x1000_A088 (UCR3_1) – 0x1001_C088 (UCR3_6)	R	0	0	0	PARERREN	FRAERREN	0	0	0	ADNIMP	RXDSEN	AIRINTEN	AWAKEN	0	RXDMUXSEL	INVT	ACIEN
	W																
0x1000_A08C (UCR4_1) – 0x1001_C08C (UCR4_6)	R	CTSTL						INVR	ENIRI	WKEN	0	IRSC	LPBYP	TCEN	BKEN	OREN	DREN
	W																
0x1000_A090 (UFCR1)	R	TXTL						RFDIV			0	RXTL					
	W																
0x1000_A090 (UFCR1) – 0x1001_C090 (UFCR6)	R	PARITYERR	RTSS	TRDY	RTSD	ESCF	FRAMERR	RRDY	AGTIM	0	RXDS	AIRINT	AWAKE	0	0	0	0
	W																

Table 28-4. UART Register Summary (continued)

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000_A098 (USR2_1) – 0x1001_C098 (USR2_6)	R	ADET	TXFE	0	IDLE	ACST	0	0	IRINT	WAKE	0	0	RTSF	TXDC	BRCD	ORE	RDR
	W																
0x1000_A09C (UESC1) – 0x1001_C09C (UESC6)	R	0	0	0	0	0	0	0	0	ESC_CHAR							
	W																
0x1000_A0A0 (UTIM1) – 0x1001_C0A0 (UTIM6)	R	0	0	0	0	TIM											
	W																
0x1000_A0A4 (UBIR1) – 0x1001_C0A4 (UBIR6)	R	INC															
	W																
0x1000_A0A8 (UBMR1) – 0x1001_C0A8 (UBMR6)	R	MOD															
	W																
0x1000_A0AC (UBRC1) – 0x1001_C0AC (UBRC6)	R	BCNT															
	W																
0x1000_A0B0 (ONEMS1) – 0x1001_C0B0 (ONEMS6)	R	ONEMS															
	W																
0x1000_A0B4 (UTS1) – 0x1001_C0B4 (UTS6)	R	0	0	FRCPERR	LOOP	DBGEN	LOOPIR	RXDBG	0	0	TXEMPTY	RXEMPTY	TXFULL	RXFULL	0	0	SOFTRST
	W																

28.3.4 Register Descriptions

28.3.4.1 UART Receiver Register (URXD)

Figure 28-4 shows the URXD register, and Table 28-5 shows the register's field descriptions.

0x1000_A000 (URXD1)
0x1000_B000 (URXD2)
0x1000_C000 (URXD3)
0x1000_D000 (URXD4)
0x1001_B000 (URXD5)
0x1001_C000 (URXD6)

Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	ERR	OVR RUN	FRM ERR	BRK	PRER R	0	0	RX_DATA							
W																
Reset	1	0	0	0	0	0	0	0	—	—	—	—	—	—	—	—

Figure 28-4. UART Receiver Register (URXD)

Table 28-5. Receiver Register Field Descriptions

Name	Description
31–16	Reserved
15	Reserved. This bit is reserved and should read 1. **Note:** Reset value is read at 1 for compatibility with existing software (previously CHARRDY was bit 15 and always read as 1).
14 ERR	Error Detect. Indicates whether the character present in the RX_DATA field has an error (OVRRUN, FRMERR, BRK or PRERR) status. The ERR bit is updated and valid for each received character. 0 No error status was detected. 1 An error status was detected.
13 OVRRUN	Receiver overrun. This read-only bit, when HIGH, indicates that the corresponding character was stored in the last position (32nd) of the RxFIFO. Even if a 33rd character has not been detected, this bit is set to ‘1’ for the 32nd character. 0 No RxFIFO overrun was detected. 1 A RxFIFO overrun was detected.
12 FRMERR	Frame error. Indicates whether the current character had a framing error (a missing stop bit) and is possibly corrupted. FRMERR is updated for each character read from the RxFIFO. 0 The current character has no framing error. 1 The current character has a framing error.
11 BRK	BREAK detect. Indicates whether the current character was detected as a BREAK character. The data bits and the stop bit are all 0. The FRMERR bit is set when BRK is set. When odd parity is selected, PRERR is also set when BRK is set. BRK is valid for each character read from the RxFIFO. 0 The current character is not a BREAK character. 1 The current character is a BREAK character.
10 PRERR	Parity error. Indicates if the current character was detected with a parity error and is possibly corrupted. PRERR is updated for each character read from the RxFIFO. When parity is disabled, PRERR always reads as 0. 0 No parity error was detected for data in the RX_DATA field. 1 A parity error was detected for data in the RX_DATA field.

Table 28-5. Receiver Register Field Descriptions (continued)

Name	Description
9–8	Reserved
7–0 RX_DATA	Received data. Holds the received character. In 7-bit mode, the most significant bit (MSB) is forced to 0. In 8-bit mode, all bits are active.

28.3.4.2 UART Transmitter Register (UTXD)

Figure 28-5 shows the UTXD register, and Table 28-6 shows the register's field descriptions.

0x1000_A040 (UTXD1)
0x1000_B040 (UTXD2)
0x1000_C040 (UTXD3)
0x1000_D040 (UTXD4)
0x1001_B040 (UTXD5)
0x1001_C040 (UTXD6)

Access: User write

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

Figure 28-5. UART Transmitter (UTXD) Register

Table 28-6. UART Transmitter Register Field Descriptions

Field	Description
31–16	Reserved
15–8	Reserved
7–0 TX_DATA	Transmit data. Holds the parallel transmit data inputs. In 7-bit mode, D7 is ignored. In 8-bit mode, all bits are used. Data is transmitted least significant bit (LSB) first. A new character is transmitted when the TX_DATA field is written. The TX_DATA field must be written only when the TRDY bit is high to ensure that corrupted data is not sent.

28.3.4.3 UART Control Register 1 (UCR1)

Figure 28-6 shows the UCR1 register, and Table 28-7 shows the register's field descriptions.

0x1000_A080 (UCR1_1)
0x1000_B080 (UCR1_2)
0x1000_C080 (UCR1_3)
0x1000_D080 (UCR1_4)
0x1001_B080 (UCR1_5)
0x1001_C080 (UCR1_6)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADEN	ADBR	TRDY EN	IDEN	ICD		RRDY EN	RXD MAE N	IREN	TXMP TYEN	RTSD EN	SNDB RK	TXD MAE N	0	DOZ E	UAR TEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-6. UART Control Register 1 (UCR1)

Table 28-7. UART Control Register 1 (UCR1) Field Descriptions

Field	Description
31–16	Reserved
15 ADEN	Automatic baud rate detection interrupt enable. Enables/disables the automatic baud rate detect complete (ADET) bit to generate an interrupt ($\overline{\text{ipi\_uart\_mint}}$ = 0). 0 Disable the automatic baud rate detection interrupt 1 Enable the automatic baud rate detection interrupt
14 ADBR	Automatic detection of baud rate. Enables/disables automatic baud rate detection. When the ADBR bit is set and the ADET bit is cleared, the receiver detects the incoming baud rate automatically. The ADET flag is set when the receiver verifies that the incoming baud rate is detected properly by detecting an ASCII character “A” or “a” (0x61 or 0x41). 0 Disable the automatic baud rate detection 1 Enable the automatic baud rate detection
13 TRDYEN	Transmitter ready interrupt enable. Enables/disables the transmitter ready interrupt (TRDY) when the transmitter has one or more slots available in the TxFIFO. The fill level in the TxFIFO at which an interrupt is generated is controlled by TxTL bits. When TRDYEN is negated, the transmitter ready interrupt is disabled. 0 Disable the transmitter ready interrupt 1 Enable the transmitter ready interrupt
12 IDEN	Idle condition detected interrupt enable. Enables/disables the IDLE bit to generate an interrupt ($\overline{\text{ipi\_uart\_rx}}$ = 0). 0 Disable the IDLE bit 1 Enable the IDLE bit
11–10 ICD	Idle condition detect. Controls the number of frames RXD is allowed to be idle before an idle condition is reported. 00 Report idle of more than 4 frames 01 Report idle of more than 8 frames 10 Report idle of more than 16 frames 11 Report idle of more than 32 frames
9 RRDYEN	Receiver ready interrupt enable. Enables/disables the RRDY interrupt when the RxFIFO contains data. The fill level in the RxFIFO at which an interrupt is generated is controlled by the RXTL bits. When RRDYEN is negated, the receiver ready interrupt is disabled. 0 Disable the RRDY interrupt 1 Enable the RRDY interrupt

Table 28-7. UART Control Register 1 (UCR1) Field Descriptions (continued)

Field	Description
8 RXDMAEN	Receive ready DMA enable. Enables/disables the receive DMA request ipd_uart_rx_dmareq when the receiver has data in the RxFIFO. The fill level in the RxFIFO at which a DMA request is generated is controlled by the RXFL bits. When negated, the receive DMA request is disabled. 0 Disable the DMA request 1 Enable the DMA request
7 IREN	Infrared interface enable. Enables/disables the IR interface. 0 Disable the IR interface 1 Enable the IR interface
6 TXMPTYEN	Transmitter empty interrupt enable. Enables/disables the transmitter FIFO empty (TXFE) interrupt ipi_uart_tx. When negated, the TXFE interrupt is disabled. 0 Disable the transmitter FIFO empty interrupt 1 Enable the transmitter FIFO empty interrupt
5 RTSDEN	RTS delta interrupt enable. Enables/disables the RTSD interrupt. The current status of the *ipp_uart_rts* pin is read in the RTSS bit. 0 Disable the RTSD interrupt 1 Enable the RTSD interrupt
4 SNDBRK	Send BREAK. Forces the transmitter to send a BREAK character. The transmitter finishes sending the character in progress (if any) and sends BREAK characters until SNDBRK is reset. Because the transmitter samples SNDBRK after every bit is transmitted, it is important that SNDBRK is asserted high for a sufficient period of time to generate a valid BREAK. After the BREAK transmission completes, the UART transmits 2 mark bits. The user can continue to fill the TxFIFO. Any characters remaining are transmitted when the BREAK is terminated. 0 Does not send a BREAK character 1 Send a BREAK character (continuous 0s)
3 TXDMAEN	Transmitter ready DMA enable. Enables/disables the transmit DMA request ipd_uart_tx_dmareq when the transmitter has one or more slots available in the TxFIFO. The fill level in the TxFIFO that generates the ipd_uart_tx_dmareq is controlled by the TXTL bits. 0 Disable the transmit DMA request 1 Enable the transmit DMA request
2	Reserved
1 DOZE	DOZE. Determines the UART enable condition in the doze state. When ipg_doze input pin is at '1', meaning the CPU executes a doze instruction and the system is placed in the doze state, the DOZE bit affects operation of the UART. While in the doze state, if this bit is asserted, the UART is disabled. Refer to the description in Section 28.4.8, "UART Operation in Low-Power System States." 0 Enable the UART when it is in doze state 1 Disable the UART when it is in doze state
0 UARTEN	UART Enable. Enables/disables the UART. If UARTEN is negated in the middle of a transmission, the transmitter stops and pulls the TXD line to a logic 1. UARTEN must be set to 1 before any access to UTXD and URXD registers; otherwise, an ipg_xfr_error is returned. Output ipg_uart_clk_en is internally connected to UARTEN and can be used for software controlled clock gating purpose. 0 Disable the UART 1 Enable the UART

28.3.4.4 UART Control Register 2 (UCR2)

Figure 28-7 shows the UCR2 register, and Table 28-8 shows the register's field descriptions.

0x1000_A084 (UCR2_1)
0x1000_B084 (UCR2_2)
0x1000_C084 (UCR2_3)
0x1000_D084 (UCR2_4)
0x1001_B084 (UCR2_5)
0x1001_C084 (UCR2_6)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ESCI	IRTS	CTSC	CTS	ESCEN	RTEC		PREN	PROE	STPB	WS	RTSEN	ATEN	TXEN	RXEN	SRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 28-7. UART Control Register 2 (UCR2) Summary

Table 28-8. UART Control Register 2 Field Descriptions

Field	Description
15 ESCI	Escape Sequence Interrupt Enable. Enables/Disables the ESCF bit to generate an interrupt. 0 Disable the escape sequence interrupt 1 Enable the escape sequence interrupt
14 IRTS	Ignore RTS Pin. Forces the RTS input signal presented to the transmitter to always be asserted (set to low), effectively ignoring the external pin. When in this mode, the RTS pin serves as a general purpose input. 0 Transmit only when the RTS pin is asserted. 1 Ignore the RTS pin
13 CTSC	CTS Pin Control. Controls the operation of the ipp_uart_cts_b output pin. When CTSC is asserted, the ipp_uart_cts_b output pin is controlled by the receiver. When the RxFIFO is filled to the level of the programmed trigger level and the start bit of the overflowing character (TRIGGER LEVEL + 1) is validated, the ipp_uart_cts_b output pin is negated to indicate to the far-end transmitter to stop transmitting. When the trigger level is programmed for less than 32, the receiver continues to receive data until the RxFIFO is full. When the CTSC bit is negated, the ipp_uart_cts_b output pin is controlled by the CTS bit. On reset, because CTSC is cleared to 0, the ipp_uart_cts_b pin is controlled by the CTS bit, which again is cleared to 0 on reset. This means that on reset the ipp_uart_cts_b signal is negated. 0 The ipp_uart_cts_b pin is controlled by the CTS bit. 1 The ipp_uart_cts_b pin is controlled by the receiver.
12 CTS	Clear to Send. Controls the ipp_uart_cts_b pin when the CTSC bit is negated. CTS has no function when CTSC is asserted. 0 The ipp_uart_cts_b pin is high (inactive). 1 The ipp_uart_cts_b pin is low (active).
11 ESCEN	Escape Enable. Enables/Disables the escape sequence detection logic. 0 Disable escape sequence detection 1 Enable escape sequence detection
10–9 RTEC	Request to Send Edge Control. Selects the edge that triggers the RTS interrupt. This has no effect on the RTS delta interrupt. RTEC has an effect only when RTSEN = 1. 00 Trigger interrupt on a rising edge 01 Trigger interrupt on a falling edge 1X Trigger interrupt on any edge

Table 28-8. UART Control Register 2 Field Descriptions (continued)

Field	Description
8 PREN	Parity Enable. Enables/Disables the parity generator in the transmitter and parity checker in the receiver. When PREN is asserted, the parity generator and checker are enabled, and disabled when PREN is negated. 0 Disable parity generator and checker 1 Enable parity generator and checker
7 PROE	Parity Odd/Even. Controls the sense of the parity generator and checker. When PROE is high, odd parity is generated and expected. When PROE is low, even parity is generated and expected. PROE has no function if PREN is low. 0 Even parity 1 Odd parity
6 STPB	Stop. Controls the number of stop bits transmitted after a character. When STPB is high, 2 stop bits are sent. When STPB is low, 1 stop bit is sent. STPB has no effect on the receiver, which expects 1 or more stop bits. 0 1 stop bit transmitted 1 2 stop bits transmitted
5 WS	Word Size. Controls the character length. When WS is high, the transmitter and receiver are in 8-bit mode. When WS is low, they are in 7-bit mode. The transmitter ignores bit 7 and the receiver sets bit 7 to 0. WS can be changed in-between transmission (reception) of characters, however not when a transmission (reception) is in progress, in which case the length of the current character being transmitted (received) is unpredictable. 0 7-bit transmit and receive character length (not including START, STOP, or PARITY bits) 1 8-bit transmit and receive character length (not including START, STOP, or PARITY bits)
4 RTSEN	Request to Send Interrupt Enable. Controls the RTS edge sensitive interrupt. When RTSEN is asserted and the programmed edge is detected on the ipp_uart_rts_b pin, the RTSF bit is asserted. 0 Disable request to send interrupt 1 Enable request to send interrupt
3 ATEN	Aging Timer Enable. This bit is used to enable the aging timer interrupt (triggered with AGTIM). 0 AGTIM interrupt is disabled. 1 AGTIM interrupt is enabled.
2 TXEN	Transmitter Enable. Enables/Disables the transmitter. When TXEN is negated the transmitter is disabled and idle. When the UARTEN and TXEN bits are set the transmitter is enabled. If TXEN is negated in the middle of a transmission, the UART disables the transmitter immediately, and starts marking 1s. The transmitter FIFO cannot be written when this bit is cleared. 0 Disable the transmitter 1 Enable the transmitter
1 RXEN	Receiver Enable. Enables/Disables the receiver. When the receiver is enabled, if the RXD input is already low, the receiver does not recognize BREAK characters, because it requires a valid 1-to-0 transition before it can accept any character. 0 Disable the receiver 1 Enable the receiver
0 SRST	Software Reset. Resets the transmitter and receiver state machines, all FIFOs, and all status registers. Once the software writes 0 to $\overline{\text{SRST}}$, the software reset remains active for 4 clock cycles of CKIH before the hardware deasserts $\overline{\text{SRST}}$. The software can only write 0 to $\overline{\text{SRST}}$. Writing 1 to $\overline{\text{SRST}}$ is ignored. 0 Reset the transmit and receive state machines, all FIFOs and all status registers 1 No reset

28.3.4.5 UART Control Register 3 (UCR3)

Figure 28-8 shows the UCR3 register, and Table 28-9 shows the register's field descriptions.

0x1000_A088 (UCR3_1)
0x1000_B088 (UCR3_2)
0x1000_C088 (UCR3_3)
0x1000_D088 (UCR3_4)
0x1001_B088 (UCR3_5)
0x1001_C088 (UCR3_6)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	PARERREN	FRAERREN	0	0	0	ADNIMP	RXDSEN	AIRINTEN	AWAKEN	0	RXDMUXSEL	INVT	ACIEN
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

Figure 28-8. UART Control Register 3 (UCR3) Summary

Table 28-9. UART Control Register 3 (UCR3) Field Descriptions

Field	Description
15–14	Reserved.
13	Reserved.
12 PARERREN	Parity Error Interrupt Enable. Enables/Disables the interrupt. When asserted, PARERREN causes the PARITYERR bit to generate an interrupt. 0 Disable the parity error interrupt 1 Enable the parity error interrupt
11 FRAERREN	Frame Error Interrupt Enable. Enables/Disables the interrupt. When asserted, FRAERREN causes the FRAMERR bit to generate an interrupt. 0 Disable the frame error interrupt 1 Enable the frame error interrupt
10	Reserved.
9	Reserved.
8	Reserved.
7 ADNIMP	Autobaud Detection Not Improved. Disables new features of autobaud detection (see Section 28.4.6.2, "Baud Rate Automatic Detection Protocol Improved" for more details). 0 Autobaud detection new features selected 1 Keep old autobaud detection mechanism

Table 28-9. UART Control Register 3 (UCR3) Field Descriptions (continued)

Field	Description
6 RXDSEN	Receive Status Interrupt Enable. Controls the receive status interrupt (ipi_uart_rx_b). When this bit is enabled and RXDS status bit is set, the interrupt ipi_uart_rx_b will be generated. 0 Disable the RXDS interrupt 1 Enable the RXDS interrupt
5 AIRINTEN	Asynchronous IR WAKE Interrupt Enable. Controls the asynchronous IR WAKE interrupt. An interrupt is generated when AIRINTEN is asserted and a pulse is detected on the UART_RX pin. 0 Disable the AIRINT interrupt 1 Enable the AIRINT interrupt
4 AWAKEN	Asynchronous WAKE Interrupt Enable. Controls the asynchronous WAKE interrupt. An interrupt is generated when AWAKEN is asserted and a falling edge is detected on the RXD pin. 0 Disable the AWAKE interrupt 1 Enable the AWAKE interrupt
3	Reserved
2 RXDMUXSEL	RXD Muxed Input Selected. Selects the ipp_uart_rxd_mux input pin for serial and Infrared input signal 0 Serial input pin is ipp_uart_rxd and IrDA input pin is ipp_uart_rxd_ir 1 Input pin is ipp_uart_rxd_mux for serial and IR interfaces In i.MX27, this bit should always be set to 1. Otherwise, UART receiver will not work.
1 INVT	Inverted Infrared Transmission. Sets the active level for the transmission. When INVT is cleared, the infrared logic block transmits a positive IR 3/16 pulse for all 0s and 0s are transmitted for 1s. When INVT is set (INVT = 1), the infrared logic block transmits an active low or negative infrared 3/16 pulse for all 0s and 1s are transmitted for 1s. 0 Active low transmission 1 Active high transmission
0 ACIEN	Autobaud Counter Interrupt Enable. This bit is used to enable the autobaud counter stopped interrupt (triggered with ACST (USR2[11]). 0 ACST interrupt disabled 1 ACST interrupt enabled

28.3.4.6 UART Control Register 4 (UCR4)

Figure 28-9 shows the UCR2 register, and Table 28-10 shows the register's field descriptions.

0x1000_A08C (UCR4_1)
0x1000_B08C (UCR4_2)
0x1000_C08C (UCR4_3)
0x1000_D08C (UCR4_4)
0x1001_B08C (UCR4_5)
0x1001_C08C (UCR4_6)

Access: User read/write

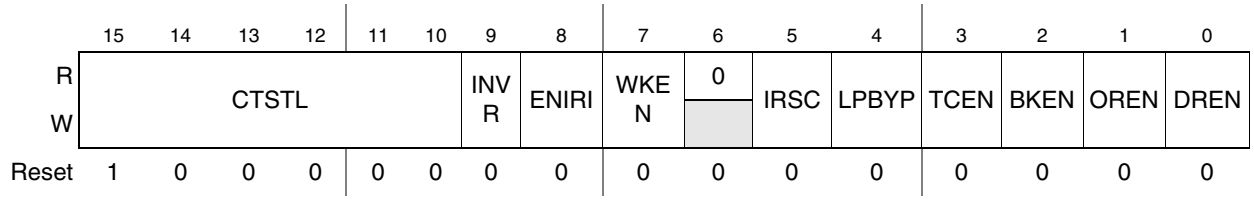


Figure 28-9. UART Control Register 4 (UCR4) Summary

Table 28-10. UART Control Register 4 (UCR4) Field Descriptions

Field	Description
15–10 CTSTL	CTS Trigger Level. Controls the threshold at which the ipp_uart_cts_b pin is deasserted by the RxFIFO. After the trigger level is reached and the ipp_uart_cts_b pin is deasserted, the RxFIFO continues to receive data until it is full. The CTSTL bits are encoded as shown in the Settings column. 000000 0 characters received 000001 1 characters in the RxFIFO ... 100000 32 characters in the RxFIFO (maximum) All Other Settings Reserved
9 INVR	Inverted Infrared Reception. Determines the logic level for the detection. When cleared, the infrared logic block expects an active low or negative IR 3/16 pulse for 0s and 1s are expected for 1s. When INVR is set (INVR = 1), the infrared logic block expects an active high or positive IR 3/16 pulse for 0s and 0s are expected for 1s. 0 Active low detection 1 Active high detection
8 ENIRI	Serial Infrared Interrupt Enable. Enables/Disables the serial infrared interrupt. 0 Serial infrared Interrupt disabled 1 Serial infrared Interrupt enabled
7 WKEN	WAKE Interrupt Enable. Enables/Disables the WAKE bit to generate an interrupt. The WAKE bit is set at the detection of a start bit by the receiver. 0 Disable the WAKE interrupt 1 Enable the WAKE interrupt
6	Reserved
5 IRSC	IR Special Case. Selects the clock for the vote logic. When set, IRSC switches the vote logic clock from the sampling clock to the UART reference clock. The IR pulses are counted a predetermined amount of time depending on the reference frequency. See Section 28.4.7.3, “InfraRed Special Case (IRSC) Bit.” 0 The vote logic uses the sampling clock (16x baud rate) for normal operation 1 The vote logic uses the UART reference clock
4 LPBYP	Low Power Bypass. Allows to bypass the low power new features in UART. To use during debug phase. 0 Low power features enabled 1 Low power features disabled

Table 28-10. UART Control Register 4 (UCR4) Field Descriptions (continued)

Field	Description
3 TCEN	Transmit Complete Interrupt Enable. Enables/Disables the TXDC bit to generate an interrupt (ipi_uart_tx_b = 0). 0 Disable TXDC interrupt 1 Enable TXDC interrupt
2 BKEN	BREAK Condition Detected Interrupt Enable. Enables/Disables the BRCD bit to generate an interrupt. 0 Disable the BRCD interrupt 1 Enable the BRCD interrupt
1 OREN	Receiver Overrun Interrupt Enable. Enables/Disables the ORE bit to generate an interrupt. 0 Disable ORE interrupt 1 Enable ORE interrupt
0 DREN	Receive Data Ready Interrupt Enable. Enables/Disables the RDR bit to generate an interrupt. 0 Disable RDR interrupt 1 Enable RDR interrupt

28.3.4.7 UART FIFO Control Register Summary (UFCR)

Figure 28-10 shows the UART FIFO register, and Table 28-11 shows the register's field descriptions.

0x1000_A090 (UFCR1)
0x1000_B090 (UFCR2)
0x1000_C090 (UFCR3)
0x1000_D090 (UFCR4)
0x1001_B090 (UFCR5)
0x1001_C090 (UFCR6)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXTL						RFDIV			0	RXTL					
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

Figure 28-10. UART FIFO Control Register (UFCR) Summary

Table 28-11. UART FIFO Control Register Description

Field	Description
15–10 TXTL	Transmitter Trigger Level. Controls the threshold at which a maskable interrupt is generated by the TxFIFO. A maskable interrupt is generated whenever the data level in the TxFIFO falls below the selected threshold. The bits are encoded as shown in the Settings column. 000000 = Reserved 000001 = Reserved 000010 = TxFIFO has 2 or fewer characters ... 011111 = TxFIFO has 31 or fewer characters 100000 = TxFIFO has 32 characters (maximum) All Other Settings Reserved
9–7 RFDIV	Reference Frequency Divider. Controls the divide ratio for the reference clock. The input clock is ipg_perclk. The output from the divider (ref_clk) is used by BRM to create the 16x baud rate oversampling clock. 000 Divide input clock by 6 001 Divide input clock by 5 010 Divide input clock by 4 011 Divide input clock by 3 100 Divide input clock by 2 101 Divide input clock by 1 110 Divide input clock by 7
6	Reserved.
5–0 RXTL	Receiver Trigger Level—Controls the threshold at which a maskable interrupt is generated by the RxFIFO. A maskable interrupt is generated whenever the data level in the RxFIFO reaches the selected threshold. The RXTL bits are encoded as shown in the Settings column. 000000 0 characters received. 000001 RxFIFO has 1 character. ... 011111 RxFIFO has 31 characters. 100000 RxFIFO has 32 characters (maximum). All Other Settings are reserved.

28.3.4.8 UART Status Register 1 Summary (USR1)

Figure 28-11 shows the register, and Table 28-12 shows the register's field descriptions.

0x1000_A094 (USR1)
0x1000_B094 (USR2)
0x1000_C094 (USR3)
0x1000_D094 (USR4)
0x1001_B094 (USR5)
0x1001_C094 (USR6)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PARITY ERR	RTSS	TRDY	RTS D	ESC F	FRAM ERR	RRDY	AGTIM	0	RXDS	AIRINT	AWAKE	0	0	0	0
W																
Reset	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 28-11. UART Status Register 1 (USR1) Summary

Table 28-12. UART Status Register 1 (USR1) Field Descriptions

Field	Description
15 PARITYERR	Parity Error Interrupt Flag. Indicates a parity error is detected. PARITYERR is cleared by writing 1 to it. Writing 0 to PARITYERR has no effect. When parity is disabled, PARITYERR always reads 0. At reset, PARITYERR is set to 0. 0 No parity error detected 1 Parity error detected
14 RTSS	$\overline{\text{RTS}}$ Pin Status. Indicates the current status of the ipp_uart_rts_b pin. A “snapshot” of the pin is taken immediately before RTSS is presented to the data bus. RTSS cannot be cleared because all writes to RTSS are ignored. At reset, RTSS is set to 0. 0 The ipp_uart_rts_b pin is high (inactive). 1 The ipp_uart_rts_b pin is low (active).
13 TRDY	Transmitter Ready Interrupt/DMA Flag. Indicates that the TxFIFO emptied below its target threshold and requires data. TRDY is automatically cleared when the data level in the TxFIFO goes beyond above the set threshold level by TXFL bits. At reset, TRDY is set to 1. 0 The transmitter does not require data. 1 The transmitter requires data (interrupt posted).
12 RTSD	RTS Delta. Indicates whether the ipp_uart_rts_b pin changed state. It (RTSD) generates a maskable interrupt. When in Sleep Mode, RTS assertion sets RTSD and can be used to wake the ARM9 core. The current state of the ipp_uart_rts_b pin is available on the RTSS bit. Clear RTSD by writing 1 to it. Writing 0 to RTSD has no effect. At reset, RTSD is set to 0. 0 ipp_uart_rts_b pin did not change state since last cleared. 1 ipp_uart_rts_b pin changed state (write 1 to clear)
11 ESCF	Escape Sequence Interrupt Flag. Indicates if an escape sequence was detected. ESCF is asserted when the ESCEN bit is set and an escape sequence is detected in the RxFIFO. Clear ESCF by writing 1 to it. Writing 0 to ESCF has no effect. 0 No escape sequence detected 1 Escape sequence detected (write 1 to clear)
10 FRAMERR	Frame Error Interrupt Flag. Indicates that a frame error is detected. The ipi_uart_mint_b interrupt generated by this. Clear FRAMERR by writing 1 to it. Writing 0 to FRAMERR has no effect. 0 No frame error detected 1 Frame error detected
9 RRDY	Receiver Ready Interrupt/DMA Flag. Indicates that the RxFIFO data level is above the threshold set by the RXFL bits. (See the RXFL bits description in Table 28-11 for setting the interrupt threshold.) When asserted, RRDY generates a maskable interrupt or DMA request. In conjunction with the CHARRDY bit in the URXDn_1 or URXDn_2 register, the software can continue to read the RxFIFO in an interrupt service routine until the RxFIFO is empty. RRDY is automatically cleared when data level in the RxFIFO goes below the set threshold level. At reset, RRDY is set to 0. 0 No character ready 1 Character(s) ready (interrupt posted)
8 AGTIM	Aging Timer Interrupt Flag. Indicates that data in the RxFIFO has been idle for a time of 8 character lengths (where a character length consists of 7 or 8 bits, depending on the setting of the WS bit in UCR2, with the bit time corresponding to the baud rate setting) and FIFO data level is less than RxFIFO threshold level (RxTL in the UFCR). Clear by writing a 1 to it. 0 AGTIM is not active. 1 AGTIM is active (write 1 to clear).
7	Reserved.

Table 28-12. UART Status Register 1 (USR1) Field Descriptions (continued)

Field	Description
6 RXDS	Receiver IDLE Interrupt Flag. Indicates that the receiver state machine is in an IDLE state, the next state is IDLE, and the receive pin is high. RXDS is automatically cleared when a character is received. RXDS is active only when the receiver is enabled. 0 Receive is in progress. 1 Receiver is IDLE.
5 AIRINT	Asynchronous IR WAKE Interrupt Flag. Indicates that the IR WAKE pulse was detected on the ipp_uart_rxd_ir pin, or on ipp_uart_rxd_mux if RXDMUXSEL is set to 1. Clear AIRINT by writing 1 to it. Writing 0 to AIRINT has no effect. 0 No pulse was detected on the RXD IrDA pin. 1 A pulse was detected on the RXD IrDA pin.
4 AWAKE	Asynchronous WAKE Interrupt Flag. Indicates that a falling edge was detected on the ipp_uart_rxd pin, or on ipp_uart_rxd_mux if RXDMUXSEL is set to 1. Clear AWAKE by writing 1 to it. Writing 0 to AWAKE has no effect. 0 No falling edge was detected on the RXD Serial pin. 1 A falling edge was detected on the RXD Serial pin.
3–0	Reserved.

28.3.4.9 UART Status Register 2 (USR2)

Figure 28-12 shows the register, and Table 28-13 shows the register's field descriptions.

0x1000_A098 (USR2_1)
0x1000_B098 (USR2_2)
0x1000_C098 (USR2_3)
0x1000_D098 (USR2_4)
0x1001_B098 (USR2_5)
0x1001_C098 (USR2_6)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADET	TXFE	0	IDLE	ACST	0	0	IRINT	WAKE	0	0	RTSF	TXDC	BRCD	ORE	RDR
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 28-12. UART Status Register 2 (USR2) Summary

Table 28-13. UART Status Register 2 Field Descriptions

Field	Description
15 ADET	Automatic Baud Rate Detect Complete. Indicates that an “A” or “a” was received and that the receiver detected and verified the incoming baud rate. Clear ADET by writing 1 to it. Writing 0 to ADET has no effect. 0 ASCII “A” or “a” was not received. 1 ASCII “A” or “a” was received (write 1 to clear).
14 TXFE	Transmit Buffer FIFO Empty. Indicates that the transmit buffer (TxFIFO) is empty. TXFE is cleared automatically when data is written to the TxFIFO. Even though TXFE is high, the transmission might still be in progress. 0 The transmit buffer (TxFIFO) is not empty. 1 The transmit buffer (TxFIFO) is empty.
13	Reserved.
12 IDLE	Idle Condition. Indicates that an idle condition has existed for more than a programmed amount frame (see Section 28.4.4.2.1, “Idle Line Detect”). An interrupt can be generated by this IDLE bit if IDEN (UCR1[12]) is enabled. IDLE is cleared by writing 1 to it. Writing 0 to IDLE has no effect. 0 No idle condition is detected. 1 Idle condition is detected (write 1 to clear).
11 ACST	Autobaud Counter Stopped. In autobaud detection (ADBR=1), indicates the counter which determines the baudrate was running and is now stopped. This means either START bit is finished (if ADNIMP=1), or Bit0 is finished (if ADNIMP=0). See Section 28.4.6.3.1, “New Autobaud Counter Stopped Bit and Interrupt” for more details. An interrupt can be flagged on ipi_uart_mint_b if ACIEN=1. 0 Measurement of bit length is not finished (in autobaud). 1 Measurement of bit length is finished (in autobaud). (Write a “one” to clear.)
10	Reserved.
9	Reserved.
8 IRINT	Serial Infrared Interrupt Flag. When an edge is detected on the RX pin during SIR Mode, this flag will be asserted. This flag can cause an interrupt on ipi_uart_mint_b which can be masked using the control bit ENIRI: UCR4 [8]. 0 No edge was detected. 1 Valid edge was detected (write a “one” to clear).
7 WAKE	Wake. Indicates the start bit is detected. WAKE can generate an interrupt on ipi_uart_mint_b that can be masked using the WKEN bit. Clear WAKE by writing 1 to it. Writing 0 to WAKE has no effect. 0 Start bit was not detected. 1 Start bit was detected (write 1 to clear).
6	Reserved.
5	Reserved.

Table 28-13. UART Status Register 2 Field Descriptions (continued)

Field	Description
4 RTSF	RTS Edge Triggered Interrupt Flag. Indicates if a programmed edge is detected on the ipp_uart_rts_b pin. The RTEC bits select the edge that generates an interrupt. RTSF can generate an interrupt on ipi_uart_mint_b that can be masked using the RTSEN bit. Clear RTSF by writing 1 to it. Writing 0 to RTSF has no effect. 0 Programmed edge is not detected on ipp_uart_rts_b. 1 Programmed edge is detected on ipp_uart_rts_b (write 1 to clear).
3 TXDC	Transmitter Complete. Indicates that the transmit buffer (TxFIFO) and Shift Register is empty; therefore the transmission is complete. TXDC is cleared automatically when data is written to the TxFIFO. 0 Transmit is incomplete. 1 Transmit is complete.
2 BRCD	BREAK Condition Detected. Indicates that a BREAK condition was detected by the receiver. Clear BRCD by writing 1 to it. Writing 0 to BRCD has no effect. 0 No BREAK condition was detected. 1 A BREAK condition was detected (write 1 to clear).
1 ORE	Overrun Error. When set to 1, ORE indicates that the receive buffer (RxFIFO) was full (32 chars inside), and a 33rd character has been fully received. This 33rd character has been discarded. Clear ORE by writing 1 to it. Writing 0 to ORE has no effect. 0 No overrun error 1 Overrun error (write 1 to clear)
0 RDR	Receive Data Ready. Indicates that at least 1 character is received and written to the RxFIFO. If the URXD register is read and there is only 1 character in the RxFIFO, RDR is automatically cleared. 0 No receive data ready 1 Receive data ready

28.3.4.10 UART Escape Character Register Summary (UESC)

Figure 28-13 shows the register, and Table 28-14 shows the register's field descriptions.

0x1000_A09C (UESC1)
0x1000_B09C (UESC2)
0x1000_C09C (UESC3)
0x1000_D09C (UESC4)
0x1001_B09C (UESC5)
0x1001_C09C (UESC6)

Access: User read/write

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

Figure 28-13. UART Escape Character Register Summary (UESC)

Table 28-14. UART Escape Character Register Field Descriptions

Name	Description
15–8	Reserved.
7–0 ESC_CHAR	UART Escape Character. Holds the selected escape character that all received characters are compared against to detect an escape sequence.

28.3.4.11 UART Escape Timer Register Summary (UTIM)

Figure 28-14 shows the register, and Table 28-15 shows the register's field descriptions.

0x1000_A0A0 (UTIM1)
0x1000_B0A0 (UTIM2)
0x1000_C0A0 (UTIM3)
0x1000_D0A0 (UTIM4)
0x1001_B0A0 (UTIM5)
0x1001_C0A0 (UTIM6)

Access: User read/write

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

Figure 28-14. UART Escape Timer Register Summary (UTIM)

Table 28-15. UART Escape Timer Register (UTIM) Field Descriptions

Name	Description
15–12	Reserved.
11–0 TIM	UART Escape Timer. Holds the maximum interval allowed between escape characters.

28.3.4.12 UART BRM Incremental Register (UBIR)

Figure 28-15 shows the register, and Table 28-16 shows the register's field descriptions.

0x1000_A0A4 (UBIR1)
0x1000_B0A4 (UBIR2)
0x1000_C0A4 (UBIR3)
0x1000_D0A4 (UBIR4)
0x1001_B0A4 (UBIR5)
0x1001_C0A4 (UBIR6)

Access: User read/write

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

Figure 28-15. UART BRM Incremental Register Summary (UBIR)

Table 28-16. UART BRM Incremental Register Field Descriptions

Name	Description
15–0 INC	Incremental Numerator. Holds the numerator value minus one of the BRM ratio (see Section 28.4.5, “Binary Rate Multiplier (BRM)”). The UBIR register MUST be updated before the UBMR register for the baud rate to be updated correctly. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this field using byte accesses is not recommended and is undefined.

28.3.4.13 UART BRM Modulator Register Summary (UBMR)

Figure 28-13 shows the register, and Table 28-17 shows the register’s field descriptions.

0x1000_A0A8 (UBMR1)
0x1000_B0A8 (UBMR2)
0x1000_C0A8 (UBMR3)
0x1000_D0A8 (UBMR4)
0x1001_B0A8 (UBMR5)
0x1001_C0A8 (UBMR6)

Access: User read/write

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

Figure 28-16. UART BRM Modulator Register Summary (UBMR)

Table 28-17. UART BRM Modulator Register Field Descriptions

Field	Description
15–0 MOD	Modulator Denominator. Holds the value of the denominator minus one of the BRM ratio (see Section 28.4.5, “Binary Rate Multiplier (BRM)”). The UBIR register MUST be updated before the UBMR register for the baud rate to be updated correctly. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this register using byte accesses is not recommended and undefined.

28.3.4.14 UART Baud Rate Count Register Summary (UBRC)

Figure 28-14 shows the register, and Table 28-18 shows the register’s field descriptions.

0x1000_A0AC (UBRC1)
0x1000_B0AC (UBRC2)
0x1000_C0AC (UBRC3)
0x1000_D0AC (UBRC4)
0x1001_B0AC (UBRC5)
0x1001_C0AC (UBRC6)

Access: User read

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

Figure 28-17. UART Baud Rate Count Register Summary (UBRC)

Table 28-18. UART Baud Rate Count Register Field Descriptions

Field	Description
15–0 BCNT	Baud Rate Count Register. This read only register is used to count the start bit of the incoming baud rate (if ADNIMP=1), or start bit + bit0 (if ADNIMP=0). When the measurement is done, the Baud Rate Count Register contains the number of UART internal clock cycles (clock after divider) present in an incoming bit. BCNT retains its value until the next Automatic Baud Rate Detection sequence has been initiated. The 16-bit Baud Rate Count register is reset to 4 and stays at hex FFFF in the case of an overflow.

28.3.4.15 UART One Millisecond Register (ONEMS)

Figure 28-15 shows the register, and Table 28-19 shows the register's field descriptions.

0x1000_A0B0 (ONEMS1)
0x1000_B0B0 (ONEMS2)
0x1000_C0B0 (ONEMS3)
0x1000_D0B0 (ONEMS4)
0x1001_B0B0 (ONEMS5)
0x1001_C0B0 (ONEMS6)

Access: User read/write

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

Figure 28-18. UART One Millisecond Register Summary (ONEMS)

Table 28-19. UART One Millisecond Register Field Descriptions

Field	Description
15–0 ONEMS	One Millisecond Register. This 16-bit register must contain the value of the UART internal frequency divided by 1000. The internal frequency is obtained after the UART internal divider. In fact this register contains the value corresponding to the number of UART internal clock cycles are present in one millisecond.

28.3.4.16 UART Test Register (UTS)

Figure 28-16 shows the register, and Table 28-20 shows the register's field descriptions.

0x1000_A0B4 (UTS1)
0x1000_B0B4 (UTS2)
0x1000_C0B4 (UTS3)
0x1000_D0B4 (UTS4)
0x1001_B0B4 (UTS5)
0x1001_C0B4 (UTS6)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	FRCPERR	LOOP	DBGEN	LOOPIR	RXDBG	0	0	TXEMPTY	RXEMPTY	TXFULL	RXFULL	0	0	SOFTRST
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Figure 28-19. UART Test Register Summary (UTS)

Table 28-20. UART Test Register Description

Field	Description
15–14	Reserved. These bits are reserved and should read 0.
13 FRCPERR	Force Parity Error. Forces the transmitter to generate a parity error if parity is enabled. FRCPERR is provided for system debugging. 0 Generate normal parity 1 Generate inverted parity (error)
12 LOOP	Loop TX and RX for Test. Controls loopback for test purposes. When LOOP is high, the receiver input is internally connected to the transmitter and ignores the RXD pin. The transmitter is unaffected by LOOP. If RXDMUXSEL (UCR3[2]) is set to 1, the loopback is applied on serial and IrDA signals. If RXDMUXSEL is set to 0, the loopback is only applied on serial signals. 0 Normal receiver operation 1 Internally connect the transmitter output to the receiver input
11 DBGEN	Debug_enable_b. This bit controls whether to respond to the debug_b input signal. 0 UART will go into debug mode when debug_b is LOW. 1 UART will not go into debug mode even if debug_b is LOW.
10 LOOPIR	Loop TX and RX for IR Test (LOOPIR). This bit controls loopback from transmitter to receiver in the InfraRed interface. 0 No IR loop 1 Connect IR transmitter to IR receiver
9 RXDBG	RX_fifo_debug_mode. This bit controls the operation of the RX fifo read counter when in debug mode. 0 RX FIFO read pointer does not increment. 1 RX_FIFO read pointer increments as normal.
8–7	Reserved. These bits are reserved and should read 0.
6 TXEMPTY	TxFIFO Empty. Indicates that the TxFIFO is empty. 0 The TxFIFO is not empty. 1 The TxFIFO is empty.

Table 28-20. UART Test Register Description (continued)

Field	Description
5 RXEMPTY	RxFIFO Empty. Indicates the RxFIFO is empty. 0 The RxFIFO is not empty. 1 The RxFIFO is empty.
4 TXFULL	TxFIFO FULL. Indicates the TxFIFO is full. 0 The TxFIFO is not full. 1 The TxFIFO is full.
3 RXFULL	RxFIFO FULL. Indicates the RxFIFO is full. 0 The RxFIFO is not full. 1 The RxFIFO is full.
2–1	Reserved. These bits are reserved and should read 0.
0 SOFTRST	Software Reset. Indicates the status of the software reset ($\overline{\text{SRST}}$). 0 No software reset 1 Generate software reset

28.4 Functional Description

28.4.1 Interrupts and DMA Requests

Table 28-21 lists all of the different interrupts signals available on the interrupt pins. See the individual register descriptions for explanation of available enables and status flags.

Table 28-21. Interrupts and DMA

Interrupt Output	Interrupt Enable	Enable Register Location	Interrupt Flag	Flag Register Location
ipi_uart_rx_b	RRDYEN IDEN DREN RXDSEN ATEN	UCR1 (bit 9) UCR1 (bit 12) UCR4 (bit 0) UCR3 (bit 6) UCR2 (bit 3)	RRDY IDLE RDR RXDS AGTIM	USR1 (bit 9) USR2 (bit 12) USR2 (bit 0) USR1 (bit 6) USR1 (bit 8)
ipi_uart_tx_b	TXMPTYEN TRDYEN TCEN	UCR1 (bit 6) UCR1 (bit 13) UCR4 (bit 3)	TXFE TRDY TXDC	USR2 (bit 14) USR1 (bit 13) USR2 (bit 3)

Table 28-21. Interrupts and DMA

Interrupt Output	Interrupt Enable	Enable Register Location	Interrupt Flag	Flag Register Location
ipi_uart_mint_b	OREN BKEN WKEN ADEN ACIEN ESCI ENIRI AIRINTEN AWAKEN FRAERREN PARERREN RTSDEN RTSEN	UCR4 (bit 1) UCR4 (bit 2) UCR4 (bit 7) UCR1 (bit 15) UCR3 (bit 0) UCR2 (bit 15) UCR4 (bit 8) UCR3 (bit 5) UCR3 (bit 4) UCR3 (bit 11) UCR3 (bit 12) UCR1 (bit 5) UCR2 (bit 4)	ORE BRCD WAKE ADET ACST ESCF IRINT AIRINT AWAKE FRAERR PARITYERR RTSD RTSF	USR2 (bit 1) USR2 (bit 2) USR2 (bit 7) USR2 (bit 15) USR2 (bit 11) USR1 (bit 11) USR2 (bit 8) USR1 (bit 5) USR1 (bit 4) USR1 (bit 10) USR1 (bit 15) USR1 (bit 12) USR2 (bit 4)
ipd_uart_rx_dmareq_b	RXDMAEN	UCR1 (bit 8)	RRDY	USR1 (bit 9)
ipd_uart_tx_dmareq_b	TXDMAEN	UCR1 (bit 3)	TRDY	USR1 (bit 13)

28.4.2 Clocking Considerations

28.4.2.1 Minimum and Maximum Clock Frequencies

UART module receives the following three clocks:

- *ipg_clk*
- *ipg_clk_s*
- *ipg_perclk*

ipg_clk is the main clock and must always be running when UART is enabled. There is an exception in Sleep Mode (see Section 28.4.2.2, “Clocking in Low-Power Modes”).

ipg_clk_s is the bus clock, it is active only when there a bus access (read/write) to UART registers.

ipg_clk and *ipg_clk_s* are synchronous and have the same frequency.

But UART receives also another clock, ipg_perclk. This clock is the Binary Multiplier Clock and it must always be running when UART is sending or receiving characters. This clock has been added in order to allow frequency scaling on *ipg_clk* (and *ipg_clk_s*) without changing configuration of BRM (*ipg_perclk* staying at a fixed frequency).

Constraints:

- *ipg_clk*, *ipg_clk_s*, and *ipg_perclk* must be synchronous and their clock trees must be balanced. But ipg_perclk frequency is not necessary equal to ipg_clk frequency (and *ipg_clk_s*).This specific relationship between *ipg_perclk* and *ipg_clk* is obtained by extracting those clocks from the same source clock but on which different dividers have been applied. The dividers ratios must always be integer values. With this constraint UART receives either *ipg_perclk* and *ipg_clk* rising edges at the same time or separated by at minimum a fixed guard-band.

For example:

Main source clock frequency is 270 MHz.

ipg_perclk frequency is fixed at 16.8 MHz (270 MHz/16).

ipg_clk (and *ipg_clk_s*) frequency can vary from 16.8 MHz to a maximum value. This maximum frequency must always comes from an integer division of main source clock (270 MHz/n). It must also be used to constrain the design during synthesis phase.

- At any moment, *ipg_clk* (and *ipg_clk_s*) frequency must be higher or equal to *ipg_perclk* frequency.

  See Figure 28-20 for examples of working configurations.
- Due to the 16x oversampling of the incoming characters, *ipg_perclk* frequency must always be greater or equal to 16x the maximum baud rate. For example, if max baud rate is 1.875 Mbit/s, *ipg_perclk* must be greater or equal to 1.875 M x 16 = 30 MHz.

NOTE

If the architecture of the IC does not require a clock dedicated to BRM, *ipg_perclk* input pin must receive same clock than *ipg_clk*. Clock trees between both pins must be balanced.

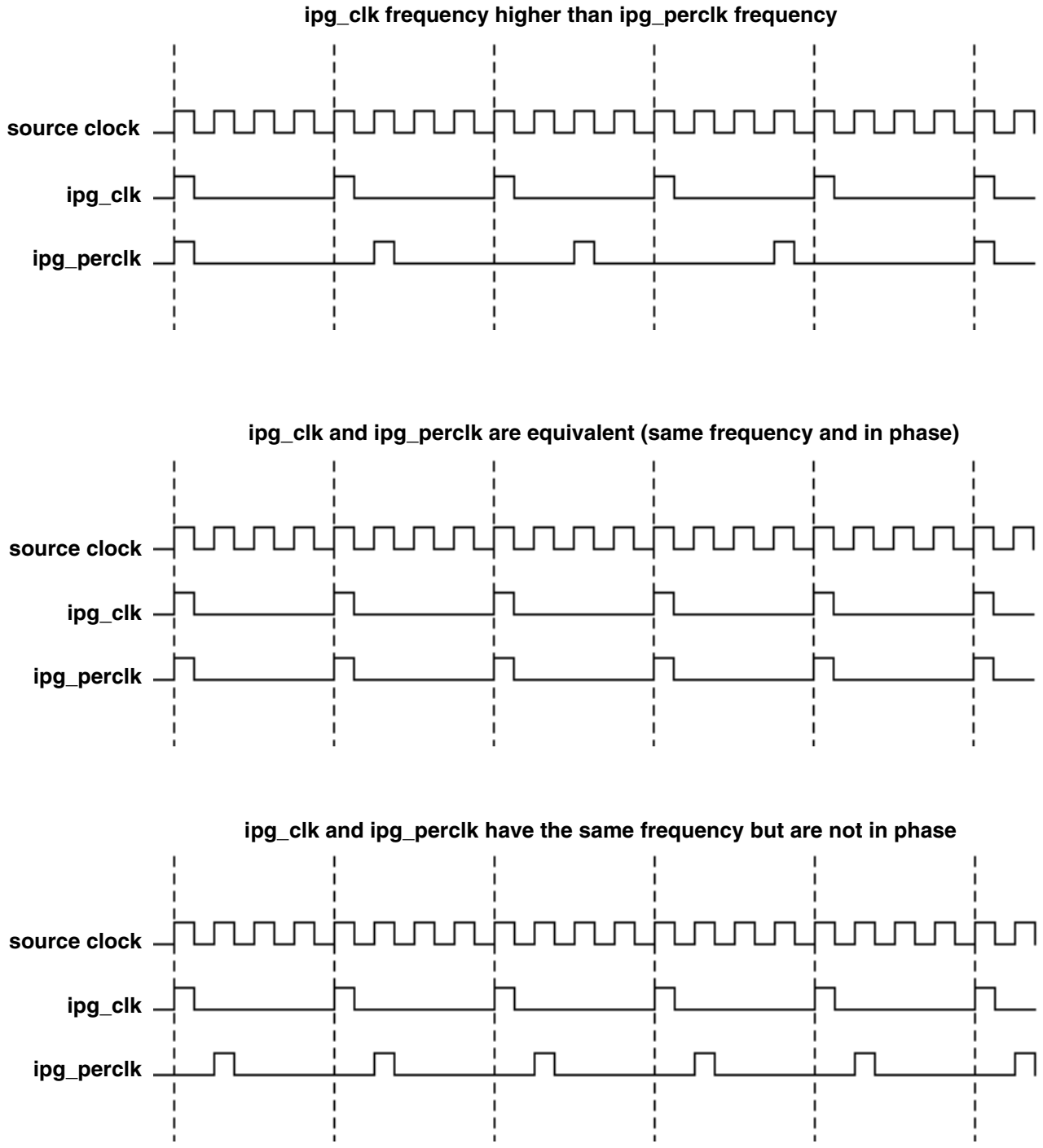


Figure 28-20. Examples of Working Relationships Between ipg_clk and ipg_perclk

28.4.2.2 Clocking in Low-Power Modes

The UART supports 2 low-power modes: DOZE and STOP.

In Sleep Mode (input pin ipg_stop is at '1'), the UART does not need any clock. In this mode the UART can wake-up the MCU with the asynchronous interrupts (see Section 28.4.8, "UART Operation in Low-Power System States"). An application of this feature is when the system must be waken-up by the arrival of a frame of characters.

- If before entering in Sleep Mode the software has enabled RTSDEN interrupt, then when RTS will change state (put at '0' by external device started to send), the asynchronous interrupt will wake-up the system, and *ipg_clk* (and *ipg_perclk*) will be provided to the UART before first start bit, so that no data will be lost.
- If RTS does not change state (already at '0' before entering in Sleep Mode), then wake-up interrupt (AWAKE) will be sent at the arrival of first Start bit (on falling edge). In this case, the UART must receive the *ipg_clk* and *ipg_perclk* during the first half of start bit to correctly receive this character (for example, at 115.2 Kbit/s, UART must receive *ipg_clk* and *ipg_perclk* at maximum 4.3 microseconds after falling edge of Start bit). If the UART receives *ipg_clk* and *ipg_perclk* too late, first character will be lost, and so should be dropped. Also, if autobaud detection is enabled, the first character will not be correctly received and another autobaud detection will need to be initiated.

In Doze mode, UART behavior is programmable through DOZE bit (UCR1[1]). If DOZE bit is set to '1', then UART is disabled in Doze mode, and in consequence, UART clocks can be switched-off (after being sure UART is not transmitting nor receiving). On the contrary, if DOZE bit is set to '0', UART is enabled and it must receive *ipg_clk* and *ipg_perclk* (and *ipg_clk_s* during accesses to registers).

28.4.3 General UART Definitions

Definitions of terms that occur the following discussions are given in this section.

- Bit Time—The period of time required to serially transmit or receive 1 bit of data (1 cycle of the baud rate frequency).
- Start bit—The bit time of a logic 0 that indicates the beginning of a data frame. A start bit begins with a 1-to-0 transition, and is preceded by at least 1 bit time of logic 1.
- Stop bit—1 bit time of logic 1 that indicates the end of a data frame.
- BREAK—A frame in which all of the data bits, including the stop bit, are logic 0. This type of frame is usually sent to signal the end of a message or the beginning of a new message.
- Frame—A start bit followed by a specified number of data or information bits and terminated by a stop bit. The number of data or information bits depends on the format specified and must be the same for the transmitting device and the receiving device. The most common frame format is 1 start bit followed by 8 data bits (least significant bit first) and terminated by 1 stop bit. An additional stop bit and a parity bit also can be included.
- Framing Error—An error condition that occurs when the stop bit of a received frame is missing, usually when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. Framing errors can go undetected if a data bit in the expected stop bit time happens to be a logic 1. A framing error is always present on the receiver side when the transmitter is sending

BREAKs. However, when the UART is programmed to expect 2 stop bits and only the first stop bit is received, this is not a framing error by definition.

- Parity Error—An error condition that occurs when the calculated parity of the received data bits in a frame does not match the parity bit received on the RXD input. Parity error is calculated only after an entire frame is received.
- Idle—One in NRZ encoding format and selectable polarity in IrDA mode.
- Overrun Error—An error condition that occurs when the latest character received is ignored to prevent overwriting a character already present in the UART receive buffer (RxFIFO). An overrun error indicates that the software reading the buffer (RxFIFO) is not keeping up with the actual reception of characters on the RXD input.

28.4.3.1 $\overline{\text{RTS}}$—UART Request To Send

The UART Request To Send input controls the transmitter. The modem or other terminal equipment signals the UART when it is ready to receive by setting $\overline{\text{RTS}}$ to '0' on the *ipp_uart_rts_b* pin. Normally, the transmitter waits until this signal is active (low) before transmitting a character, however when the Ignore RTS (IRTS) bit is set, the transmitter sends a character as soon as it is ready to transmit. An interrupt (RTSD) can be posted on any transition of this pin and can wake the MCU from Sleep Mode on its assertion. When $\overline{\text{RTS}}$ is set to '1' during a transmission, the UART transmitter finishes transmitting the current character and shuts off. The contents of the TxFIFO (characters to be transmitted) remain undisturbed.

28.4.3.2 $\overline{\text{RTS}}$ Edge Triggered Interrupt

The input to the *ipp_uart_rts_b* pin can be programmed to generate an interrupt on a selectable edge. The operation of the $\overline{\text{RTS}}$ edge triggered interrupt (RTSF) is summarized in Table 28-22.

To enable the *ipp_uart_rts_b* pin to generate an interrupt, set the request to send interrupt enable (RTSEN) bit (UCR2[4]) to 1. Writing 1 to the RTS edge triggered interrupt flag (RTSF) bit (USR2[4]) clears the interrupt flag.The interrupt can occur on the rising edge, falling edge, or either edge of the $\overline{\text{RTS}}$ input. The request to send edge control (RTEC) field (UCR2[10:9]) programs the edge that generates the interrupt. When RTEC is set to 0x00 and RTSEN = 1, the interrupt occurs on the rising edge (default). When RTEC is set to 0x01 and RTSEN = 1, the interrupt occurs on the falling edge. When RTEC is set to 0x1X and RTSEN = 1, the interrupt occurs on either edge.This is a synchronous interrupt. The RTSF bit is cleared by writing 1 to it. Writing 0 to RTSF has no effect.

Table 28-22. RTS Edge Triggered Interrupt Truth Table

RTS	RTSEN	RTEC [1]	RTEC [0]	RTSF	Interrupt Occurs On:	ipi_uart_mint_b
X	0	X	X	0	Interrupt disabled	1
1–>0	1	0	0	0	Rising edge	1
0–>1	1	0	0	1	Rising edge	0
1–>0	1	0	1	1	Falling edge	0
0–>1	1	0	1	0	Falling edge	1

Table 28-22. RTS Edge Triggered Interrupt Truth Table (continued)

RTS	RTSEN	RTEC [1]	RTEC [0]	RTSF	Interrupt Occurs On:	ipi_uart_mint_b
1–>0	1	1	X	1	Either edge	0
0–>1	1	1	X	1	Either edge	0

There is another RTS interrupt that is not programmable, however it asserts the RTS Delta (RTSD) bit when the RTS pin changes state. The status bit RTSD asserts the *ipi_uart_mint_b* interrupt when the RTS delta interrupt enable = 1. This is an asynchronous interrupt. The RTSD bit is cleared by writing 1 to it. Writing 0 to the RTSD bit has no effect.

28.4.3.3 Clear To Send ($\overline{\text{CTS}}$)

This output pin serves two purposes. Normally, the receiver indicates that it is ready to receive data by asserting this pin (low). When the CTS trigger level is programmed to trigger at 32 characters received and the receiver detects the valid start bit of the 33 character, it de-asserts this pin.

28.4.3.4 Programmable CTS Deassertion

The CTS output can also be programmed to deassert when the RxFIFO reaches a certain level. Setting the CTS trigger level (UCR4[15:10]) at any value less than 32 deasserts the CTS pin on detection of the valid start bit of the N + 1 character (where N is the trigger level setting). However, the receiver continues to receive characters until the RxFIFO is full.

28.4.3.5 TXD—UART Transmit

This is the transmitter serial output. When operating in normal mode, NRZ encoded data is output. When operating in infrared mode, a 3/16 bit-period pulse is output for each 0 bit transmitted, and no pulse is output for each 1 bit transmitted. For RS-232 applications, this pin must be connected to an RS-232 transmitter.

28.4.3.5.1 RXD—UART Receive

This is the receiver serial input. When operating in normal mode, NRZ encoded data is expected. When operating in infrared mode, a narrow pulse is expected for each 0 bit received and no pulse is expected for each 1 bit received. External circuitry must convert the IR signal to an electrical signal. RS-232 applications require an external RS-232 receiver to convert voltage levels.

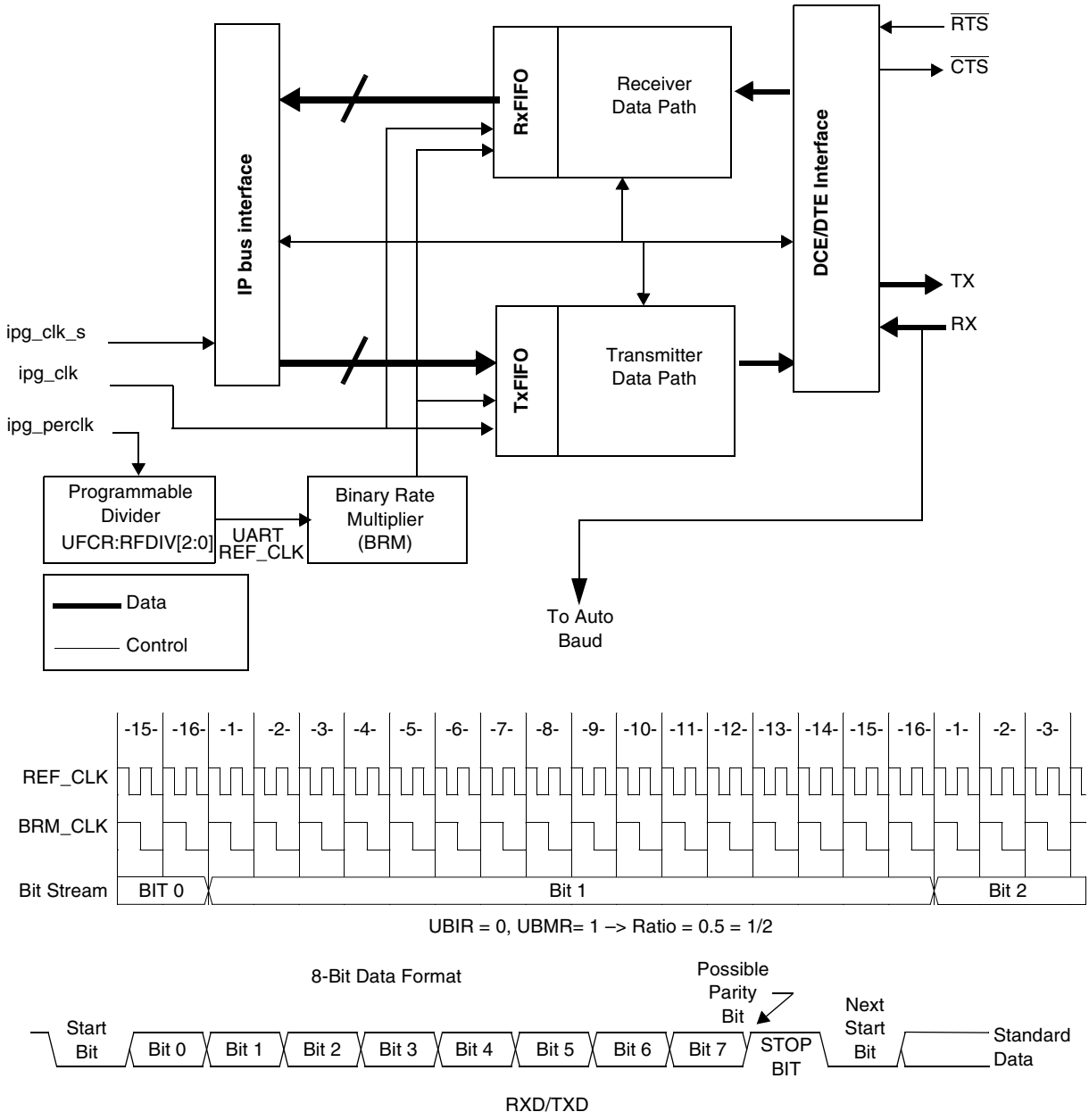


Figure 28-21. UART Simplified Block and Clock Generation Diagrams

28.4.4 Sub-Block Description

The UART module contains seven working registers separated in 2 status registers (USR1 and USR2) and five control registers (UCR1, UCR2, UCR3, UCR4, UFCR). A separate test register is provided for applications (test, verification, and so on) that require it. The binary rate multiplier registers (UBIR, UBMR) control the UART bit rate.

There is also a transmitter register (UTXD) and a receiver register (URXD). The registers are optimized for a 16-bit bus. All status bits associated with the received data are accessible along with the data in a single read. Except for the transmit data register (UTXD), all register bits are readable and most are read/write. The UART Baud Rate Count Register (UBRC) performs automatic baud rate detection. There are also two registers for the escape sequence detection, the UART Escape Character Register (UESC) and the UART Escape Timer Register (UTIM). And finally, the One Millisecond register (ONEMS) must be filled with appropriate value when the module needs to measure a duration (Escape detection mode or IR Special Case). The following sections describe the basic functionality of the mains blocks of UART module.

28.4.4.1 Transmitter

The transmitter accepts a parallel character from the MCU and transmits it serially. The start, stop, and parity (when enabled) bits are added to the character. When the ignore RTS bit (IRTS) is set, the transmitter sends a character as soon as it is ready to transmit. RTS can be used to provide flow-control of the serial data. When $\overline{\text{RTS}}$ is set to '1', the transmitter finishes sending the character in progress (if any), stops, and waits for $\overline{\text{RTS}}$ to be set to '0' again. Generation of BREAK characters and parity errors (for debugging purposes) is supported. The transmitter operates from the clock provided by the BRM. Normal NRZ encoded data is transmitted when the IR interface is disabled.

The transmitter FIFO (TxFIFO) contains 32 bytes. The data is written to TxFIFO by writing to the UTXD register with the byte data to the [7:0] bits. The data is written consecutively if the TxFIFO is not full. It is read (internally) consecutively if the TxFIFO is not empty. If the TxFIFO is full and data is again attempted to be written to the FIFO, a bus xfr_error is generated.

28.4.4.1.1 Transmitter FIFO Empty Interrupt Suppression

The transmitter FIFO empty interrupt suppression logic suppresses the TXFE interrupt between writes to the TxFIFO. When TxFIFO is empty, the software can either send one or several characters. If the software sends one character, it writes it into UTXD register and this character is immediately transferred to the transmitter shift register (when the transmitter is enabled). Without interrupt suppression logic, the TXFE interrupt would be set immediately. But, with this logic, the interrupt is set when the last bit of the character has been transmitted, that is, before the transmission of the parity bit (if exists) and the stop bit(s).

So, the suppression logic does not immediately send the TXFE interrupt. It allows the software to write another character to the TxFIFO before the interrupt is asserted.

When the transmitter shift register empties before another character is written to the TxFIFO, the interrupt is asserted. Writing data (even a single character) to the TxFIFO releases the interrupt. The interrupt is asserted on the following conditions:

- System Reset
- UART module reset
- When a single character has been written to Transmitter FIFO and then the Transmitter FIFO and the Transmitter Shift Register become empty until another character is written to the Transmitter FIFO

- The last character in the TxFIFO is transferred to the shift register, when TxFIFO contains two or more characters. See Figure 28-22.

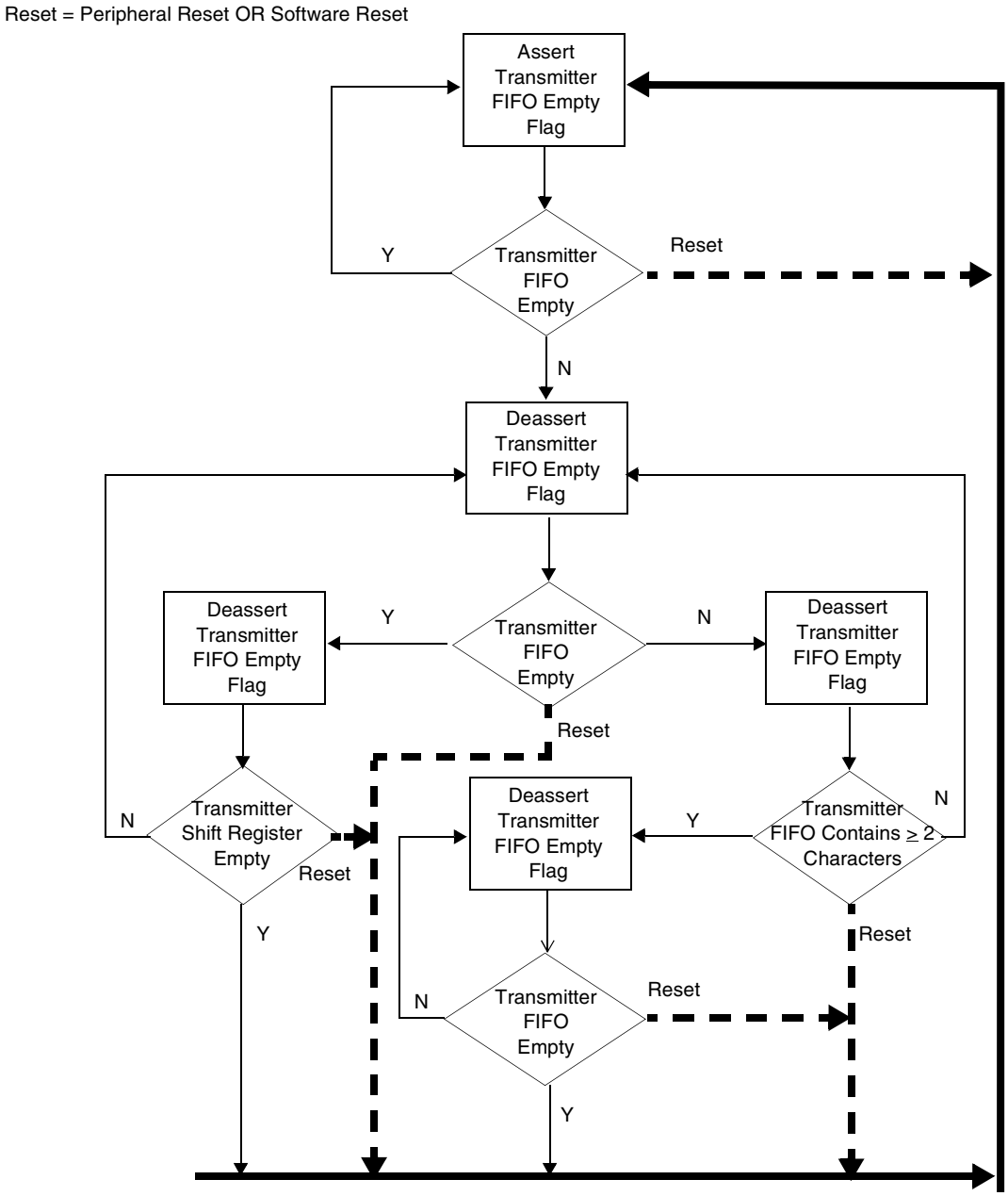


Figure 28-22. Transmitter FIFO Empty Interrupt Suppression Flow Chart

28.4.4.1.2 Transmitting a Break Condition

To send a break, bit 4 of the UCR1 reg (SNDBRK) should be asserted. This bit forces the transmitter to send a break character (continuous zeros). The transmitter will finish sending the character in progress (if any) then send break until this bit is reset. The user is responsible to ensure that this bit is high for long enough to generate a valid BREAK. The transmitter samples SNDBRK after every bit is transmitted.

Following completion of the BREAK transmission, the UART will transmit two mark bits. The user can continue to fill the FIFO and any character remaining will be transmitted when the break is terminated.

28.4.4.2 Receiver

The receiver accepts a serial data stream and converts it into parallel characters. When enabled, it searches for a start bit, qualifies it, and samples the following data bits at the bit-center. Jitter tolerance and noise immunity are provided by sampling at a 16x rate and using voting techniques to clean up the samples. Once the start bit is found, the data bits, parity bit (if enabled), and stop bits (either 1 or 2 depending on user selection) are shifted in. Parity is checked and its status reported in the URXD register when parity is enabled. Frame errors and BREAKs are also checked and reported. When a new character is ready to be read by the MCU from the RxFIFO, the receive data ready (RDR = USR2[0]) bit is asserted and an interrupt is posted (if DREN = UCR4[0] = 1). If the receiver trigger level is set to 2 (RXTL[5:0] = UFCR[5:0] = 2), and 2 chars have been received into RxFIFO, the receiver ready interrupt flag (RRDY = USR1[9]) is asserted and an interrupt is posted if the receiver ready interrupt enable bit is set (RRDYEN = UCR1[9] = 1). If the UART Receiver Register (URXD) is read once, and in consequence there is only 1 character in the RxFIFO, the interrupt generated by the RRDY bit is automatically cleared. The RRDY bit is cleared when the data in the RxFIFO falls below the programmed trigger level.

Normal NRZ encoded data is expected when the IR interface is disabled.

The RxFIFO contains 32 half-words. The data is read from the RxFIFO by reading the half-word data in the [15:0] bits in the URXD register. The data is written consecutively if the RxFIFO is not full, or is read consecutively if the RxFIFO is not empty. When additional data is written to the RxFIFO while it is full, the write operation cannot complete unless a read is performed. If a write is performed on the RxFIFO when it is full, the ORE bit (USR2[1]) register is set. The ORE bit is cleared by writing 1 to it.

28.4.4.2.1 Idle Line Detect

The receiver logic block includes the ability to detect an idle line. Idle lines indicate the end or the beginning of a message.

For an idle condition to occur:

- RxFIFO must be empty and
- RXD pin must be idle for more than a configured number of frames (ICD[1:0] = UCR1[11:10]).

When the idle condition detected interrupt enable (IDEN = UCR1[12]) is set and the line is idle for 4 (default), 8, 16, or 32 (maximum) frames, the detection of an idle condition flags an interrupt. When an idle condition is detected, the IDLE (USR2[12]) bit is set. Clear the IDLE bit by writing 1 to it. Writing 0 to the IDLE bit has no effect.

28.4.4.2.2 Idle Condition Detect Configuration

The idle condition detect ICD [1:0] field is located in the UCR1[11:10]. If the bits are set to 00b, RXD must be idle for more than 4 frames before the IDLE bit is asserted. If the bits are set to 01b, RXD must be idle for more than 8 frames before the IDLE bit is asserted. If the bits are set to 10b, RXD must be idle for more than 16 frames before the IDLE bit is asserted. If the bits are set to 11b, RXD must be idle for more than 32 frames before the IDLE bit is asserted (see Table 28-23).

Table 28-23. Detection Truth Table

IDEN	ICD [1]	ICD [0]	IDLE	ipi_uart_rx_b
0	X	X	0	1
1	0	0	asserted after 4 idle frames	asserted after 4 idle frames
1	0	1	asserted after 8 idle frames	asserted after 8 idle frames
1	1	0	asserted after 16 idle frames	asserted after 16 idle frames
1	1	1	asserted after 32 idle frames	asserted after 32 idle frames
Note: This table assumes that no other interrupt is set at the same time this interrupt is set for the ipi_uart_rx_b signal. This table shows how this interrupt affects the ipi_uart_rx_b signal.				

During a normal message there is no idle time between frames. When all of the information bits in a frame are logic 1s, the start bit ensures that at least one logic 0 bit time occurs for each frame so that the IDLE bit is not asserted.

28.4.4.2.3 Aging Character Detect

The receiver block also includes the possibility to detect when at least one character has been sitting into the RxFIFO for a time corresponding to 8 characters. This aging character capability allows the UART to inform the MCU that there is less character into the RxFIFO than the Rx trigger and, no new character has been detected on the RXD line. The aging capability is a timer which starts to count as soon as there is one character in RxFIFO. This counter is reset when either a RxFIFO read is performed or another character has been received in RxFIFO. If none of those two events occurs, the bit AGTIM (USR1[8]) is set when the counter has measured a time corresponding to 8 characters. AGTIM is cleared by writing a 1 to it. AGTIM can flag an interrupt to MCU on *ipi_uart_rx_b* if ATEN (UCR2[3]) has been set.

To summarize, AGTIM is set when:

- There is at least one character into RxFIFO.
- No read has occurred on RxFIFO and RXD line has stayed high, for a time corresponding to 8 characters.
- The RxFIFO trigger is not reached (RRDY=0).

28.4.4.2.4 Receiver Wake

The WAKE bit (USR2[7]) is set when the receiver detects a qualified the start bit, that is, which has lasted more than a half-bit duration. When the wake interrupt enable WKEN (UCR4[7]) bit is enabled, the receiver flags an interrupt (*ipi_uart_mint_b*) if the WAKE status bit is set. The WAKE bit is cleared by writing 1 to it. Writing 0 to the WAKE bit has no effect.

When the asynchronous wake interrupt (AWAKE) is enabled (AWAKEN = UCR3[4] = 1), and the MCU is in Sleep Mode, and UART clocks have been shut-off, then a falling edge detected on the receive pin (RXD) asserts the AWAKE bit (USR1[4]) and the *ipi_uart_mint_b* interrupt to wake the MCU from Sleep Mode. Re-enable UART clocks and clear the AWAKE bit by writing 1 to it. Writing 0 to the AWAKE bit has no effect.

If the asynchronous IR WAKE interrupt is enabled (AIRINTEN = UCR3[5] = 1), the UART is configured for IR mode, and if MCU is in Sleep Mode, and UART clocks have been shut-off, and a falling edge is detected on the receive pin (RXD_IR), then this asserts the AIRINT bit (USR1[5]), and the *ipi_uart_mint_b* interrupt wakes the MCU from Sleep Mode. Re-enable UART clocks and clear the AIRINT bit by writing 1 to it. Writing 0 to the AIRINT bit has no effect.

Recommended procedure for programming the asynchronous interrupts is to first clear them by writing 1 to the appropriate bit in the UART Status Register 1 (USR1). Poll or enable the interrupt for the Receiver IDLE Interrupt Flag (RXDS) in the USR1. When asserted, the RXDS bit indicates to the software that the receiver state machine is in the idle state, the next state is idle, and the RXD pin is idle (high). After following this procedure, enable the asynchronous interrupt and enter Sleep Mode.

28.4.4.2.5 Receiving a BREAK Condition

A BREAK condition is received when the receiver detects all 0s (including a 0 during the bit time of the stop bit) in a frame. The BREAK condition asserts the BRCD bit (USR2[2]) and writes only the first BREAK character to the RxFIFO. Clear the BRCD bit by writing 1 to it. Writing 0 to the BRCD bit has no effect.

When asserted BRCD can generate an interrupt on *ipi_uart_mint_b*. The interrupt generation can be masked using the control bit BKEN (UCR4[2]). Receiving a break condition will also effect the following bits in the receiver register URXD:

URXD(11) = BRK. While high this bit indicates that the current char was detected as a break.

URXD(12) = FRMERR. The frame error bit will always be set when BRK is set.

URXD(10) = PRERR. If odd parity was selected the parity error bit will also be set when BRK is set.

URXD(14) = ERR. The error detect bit indicates that the character present in the rx data field has an error status. This can be asserted by a break.

28.4.4.2.6 Vote Logic

The vote logic block provides jitter tolerance and noise immunity by sampling with respect to a 16x clock (BRM_CLK) and using voting techniques to clean up the samples. The voting is implemented by sampling the incoming signal constantly on the rising edge of the BRM_CLK. The receiver is provided with the majority vote value, which is 2 out of the 3 samples. Examples of the majority vote results of the vote logic are shown in Table 28-24.

Table 28-24. Majority Vote Results

Samples	Vote
000	0
101	1
001	0
111	1

The vote logic captures a sample on every rising edge of BRM_CLK, however the receiver uses 16x oversampling to take its value in the middle of the sample character.

The receiver starts to count when the Start bit is set however it does not capture the contents of the RxFIFO at the time the Start bit is set. The start bit is validated when 0s are received for 7 consecutive 1/16 of bit times following the 1-to-0 transition. Once the counter reaches 0xF, it starts counting on the next bit and captures it in the middle of the sampling frame (see Table 28-24). All data bits are captured in the same manner. Once the stop bit is detected, the receiver shift register (SIPO_OUT) data is parallel shifted to the RxFIFO.

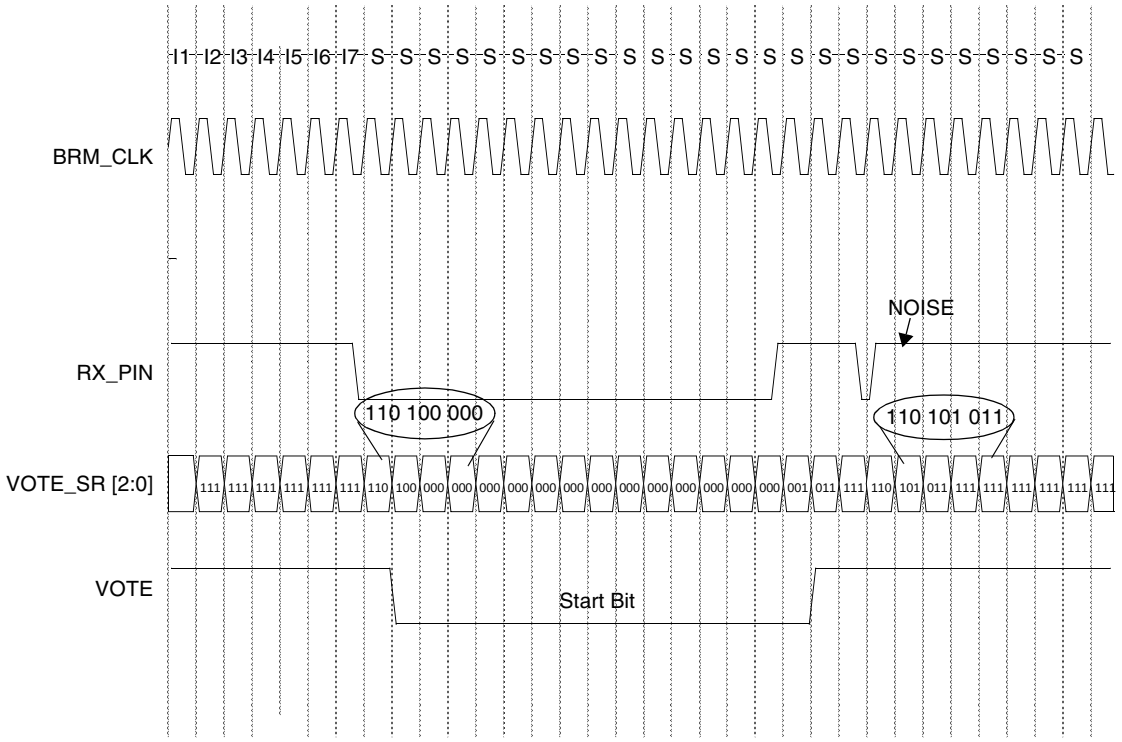


Figure 28-23. Majority Vote Results

A new feature has been recently implemented, it allows to re-synchronize the counter on each edge of RXD line. This is automatic and allows to improve the immunity of UART against signal distortion.

There is a special case when the BRM_CLK frequency is too low and is unable to capture a 0 pulse in IrDA. In this case, the software must set the IRSC bit so that the reference clock (after internal divider) is used for the voting logic. The pulse is validated by counting the length of the pulse.

28.4.5 Binary Rate Multiplier (BRM)

The BRM sub-module receives *ref_clk* (*ipg_perclk* clock after divider). Form this clock, and with integer and non-integer division, BRM generates all baud rates. The input and output frequency ratio is programmed in the UART BRM Incremental Register (UBIR) and UART BRM MOD Register (UBMR). The output frequency is divided by the input frequency to produce this ratio. For integer division, set the UBIR = 0x000F and write the divisor to the UBMR register. All values written to these registers must be

one less than the actual value to eliminate division by 0 (undefined), and to increase the maximum range of the registers.

Updating the BRM registers requires writing to both registers. The UBIR register must be written before writing to the UBMR register. If only one register is written to by the software, the BRM continues to use the previous values.

The following examples show how to determine what values are to be programmed into UBIR and UBMR for a given reference frequency and desired baud rate. The following equation can be used to help determine these values:

$$\text{BaudRate} = \frac{\text{RefFreq}}{\left(16 \times \frac{\text{UBMR} + 1}{\text{UBIR} + 1}\right)}$$

With

RefFreq (Hz): UART Reference Frequency (*ipg_perclk* after RFDIV divider).

BaudRate (bit/s): Desired baudrate.

Example 28-1. Integer Division ÷ 21

```
Reference Frequency = 19.44 MHz
UBIR = 0x000F
UBMR = 0x0014
Baudrate = 925.7 kbit/s
```

NOTE

Notice each value written to the registers is one less than the actual value.

Example 28-2. Non-Integer Division

```
Reference Frequency = 16 MHz
Desired Baudrate = 920 kbits/s
```

Eqn. 28-1

$$\frac{\text{UBMR} + 1}{\text{UBIR} + 1} = \frac{\text{RefFreq}}{16 \times \text{BaudRate}} = \frac{16 \times 10^6}{16 \times 920 \times 10^3} = 1.087$$

```
Ratio = 1.087 = 1087 / 1000
UBIR = 999 (decimal)= 0x3E7
UBMR = 1086 (decimal)= 0x43E
```

Example 28-3. Non-Integer Division

```
Reference Frequency = 25 MHz
Desired Baudrate = 920 kbit/s
```

```
Ratio = 1.69837 = 625 / 368
UBIR = 367 (decimal)= 0x16F
UBMR = 624 (decimal)= 0x270
```

Example 28-4. Non-Integer Division

```
Reference Frequency: 30 MHz
Desired Baudrate = 115.2 kbit/s

Ratio = 16.276043 = 65153 / 4003
UBIR = 4002 (decimal) = 0x0FA2
UBMR = 65152 (decimal) = 0xFE80
```

28.4.6 Baud Rate Automatic Detection Logic

When the baud rate automatic detection logic is enabled, the UART locks onto the incoming baud rate. To enable this feature, set the automatic detection of baud rate bit (ADBR = UCR1[14] = 1) and write 1 to the ADET bit (USR2[15]) to clear it. When ADET=0 and ADBR =1, the detection starts. Then, once the beginning of start bit (transition from 1-to-0 of RXD) has been detected, UART start a counter (UBRC) working at reference frequency. Once the end of start bit is detected (transition from 0-to-1 of RXD), the value of UBRC - 1 is directly copied into UBMR register. UBIR register is filled with 0x000F.

So, at the end of start bit, registers gets following values:

UBRC = number of reference clock periods (after divider) during Start bit.

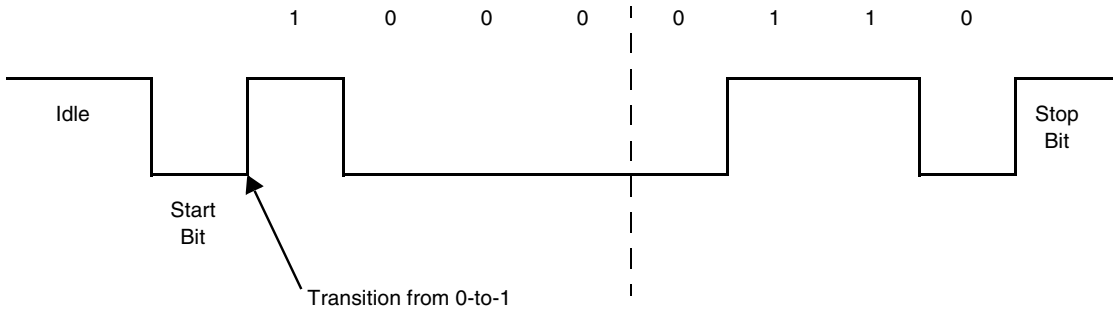
UBIR = 0x000F

UBMR = UBRC - 1

The updated values of the 3 registers can be read.

Table 28-25. Baud Rate Automatic Detection

ADBR	ADET	Baud Rate Detection	ipi_uart_mint_b
0	X	Manual Configuration	1
1	0	Auto Detection Started	1
1	1	Auto Detection Complete	0
Note: This table assumes that no other interrupt is set at the same time this interrupt is set for the ipi_uart_mint_b signal.			



Note: LSB transmitted first.

Figure 28-24. Baud Rate Detection Protocol Diagram

If any of the UART BRM registers are written to simultaneously by the baud rate automatic detection logic and peripheral data bus, the peripheral data bus has priority.

28.4.6.1 Baud Rate Automatic Detection Protocol

The receiver must receive an ASCII character “A” or “a” to verify proper detection of the incoming baud rate. When an ASCII character “A” (0x41) or “a” (0x61) is received and no error occurs, the Automatic Detect baud rate bit is set (ADET=1) and if the interrupt is enabled (ADEN=UCR1[15]=1), an interrupt *ipi_uart_mint_b* is generated.

When an ASCII character “A” or “a” is not received (because of a bit error or the reception of another character), the auto detection sequence restarts and waits for another 1-to-0 transition.

As long as ADET = 0 and ADBR = 1, the UART continues to try to lock onto the incoming baud rate. Once the ASCII character “A” or “a” is detected and the ADET bit is set, the receiver ignores the ADBR bit and continues normal operation with the calculated baudrate.

The UART interrupt is active (*ipi_uart_mint_b* = 0) as long as ADET = 1 and ADBR = 1. This can be disabled by clearing the automatic baud rate detection interrupt enable bit (ADEN = 0). Before starting an automatic baud rate detection sequence, set ADET = 0 and ADBR = 1.

The RxFIFO must contain the ASCII character “A” or “a” following the automatic baud rate detection interrupt.

The 16-bit UART Baud Rate Count Register (UBRC) is reset to 4 and stays at 0xFFFF when an overflow occurs. The UBRC register counts (measures) the duration of start bit. When the start bit is detected and counted, the UART Baud Rate Count Register retains its value until the next automatic baud rate detection sequence is initiated.

The read only Baud Rate Count Register counts only when auto detection is enabled.

28.4.6.2 Baud Rate Automatic Detection Protocol Improved

Several issues have been reported for ICs using the autobaud protocol like it is described above, especially for 57.6 kbit/s and 115.2 kbit/s. In consequence this protocol has been improved. The old one is still available in the current UART IP, but several modifications can also be used in order to make this autobaud

detection more reliable. If the user wants to keep with the old method, he has to set the bit ADNIMP (UCR3[7]) to 1. If this bit is not set (default), the autobaud improvements will be used. Those improvements are mainly grouped in two categories: the new baudrate measurement and the new ACST bit (and associated interrupt).

28.4.6.3 New Baudrate Determination

In order to fight against the problems caused by the distortion and the noise on the RXD line, the duration of the baudrate measurement has been extended. Previously, as described above, this determination was based on the measurement of the START bit duration. Now, this measurement is based on the duration of START bit + bit0. Bit0 is the first bit following the START bit. In fact, the counter which is started at the falling edge of START bit is no longer stopped at next rising edge (end of START bit), but it is stopped at the next falling edge (end of bit0). As the character sent is always a “A” (41h) or a “a” (61h), this second falling edge will be always be present and it will indicate the end of bit0. Once this counter is stopped, the result is divided by 2 and used by the BRM to determine the incoming baud rate.

Note: UBRC register contains the result of this division by two, in consequence it reflects the measurement of the duration of one bit.

28.4.6.3.1 New Autobaud Counter Stopped Bit and Interrupt

A new bit has been added in USR2 register: ACST (USR2[11]). This bit is set immediately after the determination of the baud rate, So,

- If ADNIMP is not set (default), ACST is set to 1 after the end of bit 0,
- If ADNIMP is set to 1, ACST is set to 1 at the end of START bit.

If ACIEN (UCR3[0]) is set to 1, ACST will flag an interrupt on *ipi_uart_mint_b* signal. This interrupt informs the MCU the BRM has just been set with the result of the bit length measurement. If needed, the MCU can perform a read of UBMR (or UBRC) register and determine by itself the baudrate measured. Then the MCU has the possibility to correct the BRM registers with the nearest standardized baudrate.

NOTE

- ACST is set only if ADBR is set to 1, that is, the UART is autobauding.
- Clear the ACST bit by writing 1 to it. Writing 0 to the ACST bit has no effect.

28.4.7 Escape Sequence Detection

An escape sequence typically consists of 3 characters entered in rapid succession (such as +++). Because these are valid characters by themselves, the time between characters determines if it is a valid escape sequence. Too much time between two of the “+” characters is interpreted as two “+” characters, and not part of an escape sequence.

The software chooses the escape character and writes its value to the UART Escape Character Register (UESC). The software must also enable escape detection feature by setting ESCEN (UCR2[11]) to 1. The hardware compares this value to incoming characters in the RxFIFO. When an escape character is detected, the internal escape timer starts to count. The software specifies a time-out value for the maximum

allowable time between 2 successive escape characters. The escape timer is programmable in intervals of 2 ms to a maximum interval of 8.192 seconds.

Table 28-26. Escape Timer Scaling

UTIM Register	Maximum Time Between Specified Escape Characters
0x000	2 ms
0x001	4 ms
0x002	6 ms
0x003	8 ms
0x004	10 ms
…	…
0F8	498 ms
0F9	500 ms
…	…
9C3	5 s
…	…
FFD	8.188 s
FFE	8.190 s
FFF	8.192 s
Note: To calculate the time interval: (UTIM_Value + 1) × 0.002 = Time_Interval Example: (09C3 + 1) × 0.002 = 5 sec.	

The escape sequence detection feature is available for all the reference frequencies. Before using Escape Sequence Detection, the user must fill the ONEMS register. This 16-bit register must contain the value of the UART internal frequency divided by 1000. The internal frequency is obtained after the UART internal divider which is applied on *ipg_perclk* clock.

Example:

- If the UART BRM (*ipg_perclk*) input clock frequency is 66.5 MHz.
- And if the UART input clock is divided by 2 with the internal divider: UFCR[9:7] = 3’b100

Eqn. 28-2

$$ONEMS = \frac{66.5\times10^{6}}{2\times1000} = 33250 = 81E2h$$

The escape sequence detection feature asserts the escape sequence interrupt flag (ESCF) bit when the escape sequence interrupt enable (ESCI) bit is set and an escape sequence is detected. Clear the ESCF bit by writing 1 to it. Writing 0 to the ESCF bit has no effect.

28.4.7.1 Generalities

The Infrared interface is selected when IREN (UCR1[7]) is set to 1.

The Infrared Interface is compatible with IrDA Serial Infrared Physical Layer Specification. In this specification, a “zero” is represented by a positive pulse, and a “one” is represented by no pulse (line remains low).

In the UART:

In TX: For each “zero” to be transmitted, a narrow positive pulse which is 3/16 of a bit time is generated. For each “one” to be transmitted no pulse is generated (output is low). External circuitry has to be provided to drive an Infrared LED.

In RX: When receiving, a narrow negative pulse is expected for each “zero” transmitted while no pulse is expected for each “one” transmitted (input is high). Note that Rx part of IR block expects to receive an inverted signal compared to IrDA specification. Circuitry external to the IC transforms the Infrared signal to an electrical signal.

The IR interface has an edge triggered interrupt (IRINT). This interrupt validates a zero bit being received. This interrupt is enabled by writing a “one” to ENIRI bit.

The behavior of Infrared Interface is determined by three bits INVT (UCR3[1]), INVR (UCR4[9]), and IRSC (UCR4[5]).

28.4.7.2 Inverted Transmission and Reception Bits (INVT and INVR)

The values of INVT and INVR depend of the IrDA transceiver connected on the TXD_IR and RXD_IR pins of the UART. If this transceiver is not inverting on both paths Tx and Rx (like in IrDA specification), that is, a Zero is represented as a positive pulse and a One is represented by no pulse (line remains low) for TX and RX, the bit INVT must be set to 0 and the bit INVR must be set to 1 (because Rx IR block expects an inverted signal). On the contrary user must set INVT=1 and INVR=0 if both paths of the transceiver are inverting (a Zero is represented as a negative pulse and a One is represented by no pulse (line remains high). The transceiver can also be inverting on only one path (Tx or Rx), in this case INVT and INVR must be together equal to 1 or to 0 (depending on which path is inverted).

28.4.7.3 InfraRed Special Case (IRSC) Bit

The value to apply to IRSC bit depends essentially of two parameters: the baudrate and the Minimum Pulse Duration (MPD) of the transceiver. As already written, in IrDA a Zero is represented by a positive pulse. The IrDA specification says that for SIR (Serial IR) baudrates (from 2.4 kbit/s to 115.2 kbit/s) this nominal pulse duration is equal to 3/16 of a bit duration (at the selected baudrate). But, for all the baudrates a Minimum Pulse Duration is also specified. For SIR, the MPD is constant and equal to 1.41 us.

In order to understand the meaning of bit IRSC, me must have an idea of how works the Rx path in IrDA.

When UART is in IrDA mode, a Zero is not only detected by the state of the RXD_IR line, but also with the duration of the pulse. This pulse duration can be measured with 2 different clocks. The choice of the clock is done with IRSC bit.

- If IRSC = 0, the clock used is the BRM clock.
- If IRSC = 1, the clock used is the UART internal clock (UART clock after the divider (RFDIV)).

In normal operation, IRSC=0. This means at any time, the user must be sure the frequency of BRM_clock is high enough to measure the pulse. In the UART and for IRSC=0, the pulse must last *at least 2 BRM clock cycles*.

If this condition is not fulfilled, IRSC must be set to 1.

Let's take two examples, with the Minimum Pulse Duration equals to the MPD of the IrDA specification (in SIR).

Example 28-5. Clock Example #1

The user wants to receive IrDA data at 115.2 kbit/s. The UBIR and UBMR registers are set in order to create the BRM_clock with a frequency of 16*baudrate = 16 * 115.2 = 1.843 MHz. But at the same time, in order to correctly detect the pulse, the user must be sure that 2* BRM_clock period is lower than 1.41us. Lets check:

BRM_clock period = 1/1843000 = 542 ns

So 2*BRM_clock period = 1.09 us < 1.41 us. It is fine.

Example 28-6. Clock Example #2

This time the user wants to receive at 19.2 kbit/s. So, the BRM_clock is set to 16*19200 = 307.2 kHz. Let's check if 2* BRM_clock period < 1.41 us:

BRM_clock period =1/307200 = 3.25 us

So 2*BRM_clock period = 6.50 us >> 1.41 us. It does not work.

So, in this case, the BRM clock can not be used to measure the pulse duration, and the user must select the UART internal clock by setting IRSC =1.

NOTE

Like for Escape character detection, when IR Special Case is enabled (IRSC=1), the UART must measure a duration. In order to do that, the user must fill the ONEMS register. See Section 28.4.7, "Escape Sequence Detection."

28.4.7.4 IrDA Interrupt

Serial infrared mode (SIR) uses an edge triggered interrupt flag IRINT (USR2[8]). When INVR =0, detection of a falling edge on the UART_RXD pin assets the IRINT bit. When INVR=1, detection of a rising edge on the UART_RXD pin assets the IRINT bit. When IRINT and ENIRI bits are both asserted,

the *ipi_uart_mint_b* interrupt is asserted. Clear the IRINT bit by writing 1 to it. Writing 0 to the IRINT bit has no effect.

28.4.7.5 Conclusion About IrDA

Before using the UART in IrDA, the baud rate limit must be calculated. This baud rate limit will inform the user if IRSC bit has to be set or not.

Let's determine this limit:

As already written, if IRSC = 0 the following condition must always be fulfilled:

$$2 \times \text{BRMClockPeriod} < \text{MinPulseDuration}$$

So,

$$\text{BRMClockFrequency} > \frac{2}{\text{MPD}}$$

So, knowing BRM_clock frequency = 16 * Baudrate, we get:

$$\text{BaudRate} > \frac{1}{8 \times \text{MinPulseDuration}}$$

So, the user needs to set IRSC = 0 when:

If Minimum Pulse Duration = 2.5 us and Baudrate > 50 kbit/s.

If Minimum Pulse Duration = 2.0 us and Baudrate > 62.5 kbit/s.

If Minimum Pulse Duration = 1.41 us and Baudrate > 88.6 kbit/s.

For baud rates lower than the limit, IRSC must be set to 1.

28.4.8 UART Operation in Low-Power System States

The UART's serial interface will operate as long as *ipg_clk* and *ipg_perclk* are provided. The RXEN (UCR2[1]), TXEN (UCR2[2]), and UARTEN (UCR1[0]) bits are set by the user and provide software control of low-power modes.

The table below shows the UART functionality while in hardware controlled low-power modes. These modes are controlled by the signals ipg_doze and ipg_stop.

Table 28-27. UART Low Power State Operation

	Normal State	Doze State		Stop State
		DOZE Bit = 0	DOZE Bit = 1	
UART-Clock	ON	ON	ON	OFF
UART Serial/IrDA	ON	ON	OFF	OFF

While in Doze State, the UART behavior depends on the DOZE (UCR1[1]) control bit. While the DOZE bit is negated, the UART serial interface is operational. While the system is in the Doze State, and the DOZE bit is asserted, the UART is disabled. If the Doze State is entered with the DOZE bit asserted while the UART serial interface was receiving or transmitting data, it will complete the receive/transmit of the current character and signal to the far-end transmitter/receiver to stop sending/receiving. The control/status/data registers will not change when getting into/out of low power modes.

The following UART interrupts wake the MCU processor from Sleep Mode:

- RTS (RTSD)
- IrDA Asynchronous WAKE (AIRINT)
- Asynchronous WAKE (AWAKE)

Setting the UARTEN (UCR1[0]) bit to 0 shuts off the receiver and transmitter logic and the associated clocks.

If the UART is used only in transmit mode, UARTEN and TXEN must be set to 1. If the UART is used only in receive mode, UARTEN and RXEN must be set to 1. Setting TXEN or RXEN to 0 allows to save a lot of power.

When an asynchronous WAKE interrupt exits the MCU from Sleep Mode, make sure that a dummy character is sent first because the first character may not be received correctly.

28.4.9 UART Operation in System Debug State

The bit UTS [11] controls whether the UART will respond to the input signal ipg_debug, or whether it will continue to run as normal.

If the UART is programmed to respond to ipg_debug:

1. The UART will halt all operations upon detecting the ipg_debug input.
2. A transfer in progress, either to/from a core (via the IP bus interface) or to/from an external device, will be completed before halting. This means a single byte/word transfer, not an entire FIFO. Reception of any further data from an external device will be disabled.
3. Internal registers will continue to be writable and readable via the IP bus interface. A read will leave the contents unaffected.
4. The RX FIFO is affected in debug mode in the following way:
   a) All writes into the RX FIFO are prevented.
   b) The bit RXDBG (UTS[9]) is used to select the readability of the RX FIFO during debug mode:

—RXDBG = 0: hold the read pointer at the location it had upon entering debug mode, and URXD register returns only the data value at that location, no matter how many reads attempted.

—RXDBG = 1, selectable at any time: Allow to read the characters received in Rx FIFO. It will not be possible to re-read previously read locations, nor will it be possible to readjust the read pointer to the value it had prior to entering debug mode.

Appendix A Programming IrDA Interface

A.1 High Speed

As an example, the following sequence can be used to program the IrDA interface in order to send and receive characters at 115.2 kbit/s.

Assumptions:

- Input UART clock = 90 MHz
- Internal clock divider = 3 (divide Input UART clock by 3)
- Baud rate = 115.2 kbit/s
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a Zero is represented by a positive pulse, and a One is represented by no pulse (line stays low).
- Interrupt: Sent to MCU when 1 char is received into the Rx FIFO (RDR)

Registers values and Programming orders:

```
UCR1 = 0x0085
             UCR1[7] = IREN = 1: Enable IR interface
             UCR1[0] = UARTEN = 1: Enable UART

UTS = 0x0000

UFCR = 0x0981
             TXTL[5:0] = 0x02: Default value
             RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock is 30
MHz)
             RXTL[5:0] = 0x01: Default value

UBIR = 0x0202
UBMR = 0x20BE Baud rate = 115.2 kbit/s with internal clock = 30 MHz

UCR2 = 0x4027
             UCR2[14] = IRTS = 1: Ignore level of RTS input signal
             UCR2[5] = WS = 1: Characters are 8-bit length
             UCR2[2] = TXEN = 1: Enable Rx path
             UCR2 [1] = RXEN = 1: Enable Tx path
             UCR2[0] = SRST_B = 1: No software reset

UCR3 = 0x0000

UCR4 = 0x8201
             CTSTL[5:0] = 0x20: Default value
             UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA  transceiver is
not inverting)
             UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)
```

The UART is ready to send a character as soon as there is a write into UTXD register. And an interrupt will be sent to MCU when a character is received.

A.2 Low Speed

This time, we keep the same assumptions but the speed is now 9.6 kbit/s. So, this baud rate is below the limit (even with a Min. Pulse Duration of 2.5 us) and thus IRSC and REF30 must be set to 1.

Assumptions:

- Input UART clock = 90 MHz
- Internal clock divider = 3 (divide Input UART clock by 3)
- Baud rate = 9.6 kbit/s
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a Zero is represented by a positive pulse, and a One is represented by no pulse (line stays low).
- Interrupt: Sent to MCU when 1 char is received into the Rx FIFO (RDR)

Registers values and Programming orders:

```
UCR1 = 0x0085
         UCR1[7] = IREN = 1: Enable IR interface
         UCR1[0] = UARTEN = 1: Enable UART


UFCR = 0x0981
         UFCR[15:10] = TXTL[5:0] = 0x02: Default value
         RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock is 30 MHz)
         UFCR[5:0] = RXTL[5:0] = 0x01: Default value

UBIR = 0x00FF
UBMR = 0xC354 Baud rate = 9.6 kbit/s with internal clock = 30 MHz

UCR2 = 0x4027
         UCR2[14] = IRTS = 1: Ignore level of RTS input signal
         UCR2[5] = WS = 1: Characters are 8-bit length
         UCR2[2] = TXEN = 1: Enable Rx path
         UCR2 [1] = RXEN = 1: Enable Tx path
         UCR2[0] = SRST_B = 1: No software reset

UCR3 = 0x0004
         UCR3[2] = REF30 = 1: Internal UART clock = 30 MHz

UCR4 = 0x8221
         UCR4[15:10] = CTSTL[5:0] = 0x20: Default value
         UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not
inverting)
                  UCR4[5] = IRSC = 1: Because data rate is below the limit and thus the UART
internal clock is used to measure the pulse duration.
                  UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)
```

The UART is now ready to send a character as soon as there is a write into UTXD register. And an interrupt will be sent to MCU when a character is received.

Chapter 29
Fast Ethernet Controller (FEC)

29.1 Introduction

This chapter provides a feature-set overview, a functional block diagram, and transceiver connection information for both the 10 and 100 Mbps MII (Media Independent Interface), as well as the 7-wire serial interface. Additionally, detailed descriptions of operation and the programming model are included.

29.2 Overview

The Ethernet Media Access Controller (MAC) is designed to support both 10 and 100 Mbps Ethernet/IEEE 802.3 networks. An external transceiver interface and transceiver function are required to complete the interface to the media. The FEC supports three different standard MAC-PHY (physical) interfaces for connection to an external Ethernet transceiver. The FEC supports the 10/100 Mbps MII and the 10 Mbps-only 7-wire interface, which uses a subset of the MII pins.

29.2.1 Features

The FEC incorporates the following features:

- Support for three different Ethernet physical interfaces:
  - 100-Mbps IEEE 802.3 MII
  - 10-Mbps IEEE 802.3 MII
  - 10-Mbps 7-wire interface (industry standard)
- IEEE 802.3 full duplex flow control
- Programmable max frame length supports IEEE 802.1 VLAN tags and priority
- Support for full-duplex operation (200Mbps throughput) with a minimum system clock rate of 50MHz
- Support for half-duplex operation (100Mbps throughput) with a minimum system clock rate of 25 MHz
- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
- Address recognition
  - Frames with broadcast address may be always accepted or always rejected
  - Exact match for single 48-bit individual (unicast) address
  - Hash (64-bit hash) check of individual (unicast) addresses

— Hash (64-bit hash) check of group (multicast) addresses
— Promiscuous mode

29.3 Modes of Operation

The primary operational modes are described in this section.

29.3.1 Full and Half Duplex Operation

Full duplex mode is intended for use on point to point links between switches or end node to switch. Half duplex mode is used in connections between an end node and a repeater or between repeaters. Selection of the duplex mode is controlled by TCR[FDEN].

When configured for full duplex mode, flow control may be enabled. Refer to the TCR[RFC_PAUSE] and TCR[TFC_PAUSE] bits, the RCR[FCE] bit, and Section 29.5.10, “Full Duplex Flow Control,” for more details.

29.3.2 Interface Options

The following interface options are supported. A detailed discussion of the interface configurations is provided in Section 29.5.5, “Network Interface Options.”

29.3.2.1 10 Mbps and 100 Mbps MII Interface

MII is the Media Independent Interface defined by the IEEE 802.3 standard for 10/100 Mbps operation. The MAC-PHY interface may be configured to operate in MII mode by asserting RCR[MII_MODE].

The speed of operation is determined by the FEC_TX_CLK and FEC_RX_CLK pins which are driven by the external transceiver. The transceiver will either auto-negotiate the speed or it may be controlled by software via the serial management interface (FEC_MDC/FEC_MDIO pins) to the transceiver. Refer to the MMFR and MSCR register descriptions as well as the section on the MII for a description of how to read and write registers in the transceiver via this interface.

29.3.2.2 10 Mpbs 7-Wire Interface Operation

The FEC supports a 7-wire interface as used by many 10 Mbps ethernet transceivers. The RCR[MII_MODE] bit controls this functionality. If this bit is deasserted, the MII mode is disabled and the 10 Mbps, 7-wire mode is enabled.

29.3.3 Address Recognition Options

The address options supported are promiscuous, broadcast reject, individual address (hash or exact match), and multicast hash match. Address recognition options are discussed in detail in Section 29.5.8, “Ethernet Address Recognition.”

29.3.4 Internal Loopback

Internal loopback mode is selected via RCR[LOOP]. Loopback mode is discussed in detail in Section 29.5.13, “Internal and External Loopback.”

29.4 FEC Top-Level Functional Diagram

The block diagram of the FEC is shown below. The FEC is implemented with a combination of hardware and microcode. The off-chip (Ethernet) interfaces are compliant with industry and IEEE 802.3 standards.

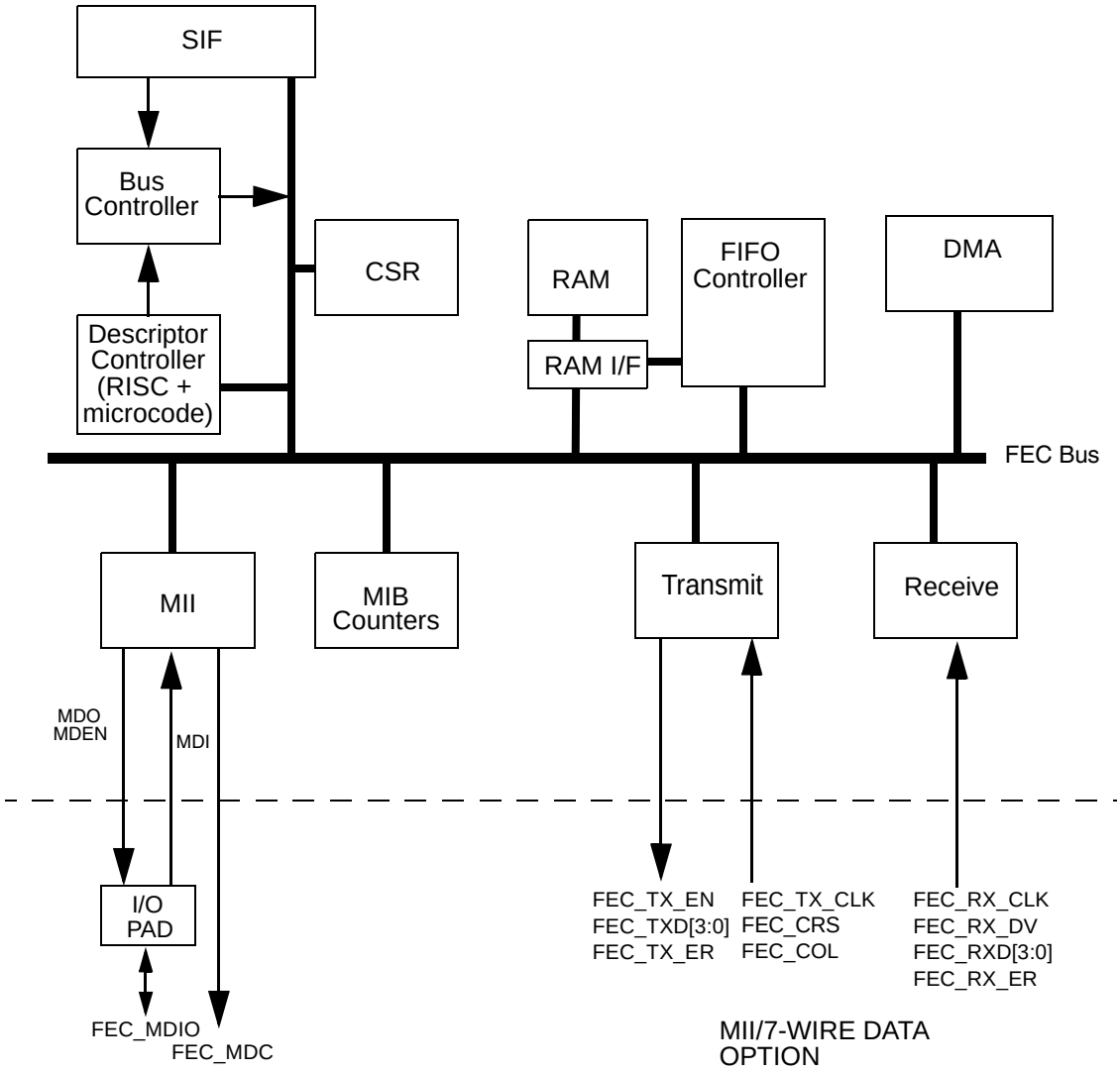


Figure 29-1. FEC Block Diagram

The descriptor controller is a RISC-based controller that provides the following functions in the FEC:

- Initialization (those internal registers not initialized by the user or hardware)
- High level control of the DMA channels (initiating DMA transfers)

- Interpreting buffer descriptors
- Address recognition for receive frames
- Random number generation for transmit collision back-off timer

NOTE

For DMA references in this section, refer to the FEC's DMA engine. This DMA engine is for the transfer of FEC data only, and is not related to the DMA controller described in Chapter 37, "Direct Memory Access Controller (DMAC)."

The RAM is the focal point of all data flow in the Fast Ethernet Controller and is divided into transmit and receive FIFOs. The FIFO boundaries are programmable using the FRSR register. User data flows to/from the DMA block from/to the receive/transmit FIFOs. Transmit data flows from the transmit FIFO into the transmit block and receive data flows from the receive block into the receive FIFO.

The user controls the FEC by writing, through the SIF (Slave Interface) module, into control registers located in each block. The CSR (Control and Status Register) block provides global control (for example, Ethernet reset and enable) and interrupt handling registers.

The MII block provides a serial channel for control/status communication with the external physical layer device (transceiver). This serial channel consists of the FEC_MDC (Management Data Clock) and FEC_MDIO (Management Data Input/Output) lines of the MII interface.

The DMA block provides multiple channels allowing transmit data, transmit descriptor, receive data and receive descriptor accesses to run independently.

The Transmit and Receive blocks provide the Ethernet MAC functionality (with some assist from microcode).

The Message Information Block (MIB) maintains counters for a variety of network events and statistics. It is not necessary for operation of the FEC but provides valuable counters for network management. The counters supported are the RMON (RFC 1757) Ethernet Statistics group and some of the IEEE 802.3 counters. See Section 29.6.3, "MIB Block Counters Memory Map" for more information.

29.5 Functional Description

This section describes the operation of the FEC, beginning with the hardware and software initialization sequence, then a detailed description of the functions of the FEC.

29.5.1 Initialization Sequence

This section describes which registers are reset due to hardware reset, which are reset by the FEC RISC, and what locations the user must initialize prior to enabling the FEC.

29.5.1.1 Hardware Controlled Initialization

In the FEC, registers and control logic that generate interrupts are reset by hardware. A hardware reset deasserts output signals and resets general configuration bits.

Other registers reset when the ECR[ETHER_EN] bit is cleared. ECR[ETHER_EN] is deasserted by a hard reset or may be deasserted by software to halt operation. By deasserting ECR[ETHER_EN], the configuration control registers such as the TCR and RCR will not be reset, but the entire data path will be reset.

Table 29-1. ECR[ETHER_EN] De-Assertion Effect on FEC

Register/Machine	Reset Value
XMIT block	Transmission is aborted (bad CRC appended)
RECV block	Receive activity is aborted
DMA block	All DMA activity is terminated
RDAR	Cleared
TDAR	Cleared
Descriptor Controller block	Halt operation

29.5.2 User Initialization (Prior to Asserting ECR[ETHER_EN])

The user needs to initialize portions the FEC prior to setting the ECR[ETHER_EN] bit. The exact values will depend on the particular application. The sequence is not important.

Ethernet MAC registers requiring initialization are defined in Table 29-2.

Table 29-2. User Initialization (Before ECR[ETHER_EN])

Description
Initialize EIMR
Clear EIR (write 0xFFFF_FFFF)
TFWR (optional)
IALR/IAUR
GAUR/GALR
PALR/PAUR
OPD (only needed for full duplex flow control)
RCR
TCR
MSCR (optional)
Clear MIB_RAM (locations $1002_B000+ 0x200-0x2FC)

FEC FIFO/DMA registers that require initialization are defined in Table 29-3.

Table 29-3. FEC User Initialization (Before ECR[ETHER_EN])

Description
Initialize FRSR (optional)
Initialize EMRBR
Initialize ERDSR

Table 29-3. FEC User Initialization (Before ECR[ETHER_EN]) (continued)

Description
Initialize ETDSR
Initialize (Empty) Transmit Descriptor ring
Initialize (Empty) Receive Descriptor ring

29.5.3 Microcontroller Initialization

In the FEC, the descriptor control RISC initializes some registers after ECR[ETHER_EN] is asserted. After the microcontroller initialization sequence is complete, the hardware is ready for operation.

Table 29-4 shows microcontroller initialization operations.

Table 29-4. Microcontroller Initialization

Description
Initialize BackOff Random Number Seed
Activate Receiver
Activate Transmitter
Clear Transmit FIFO
Clear Receive FIFO
Initialize Transmit Ring Pointer
Initialize Receive Ring Pointer
Initialize FIFO Count Registers

29.5.4 User Initialization (After Asserting ECR[ETHER_EN])

After asserting ECR[ETHER_EN], the user can set up the buffer/frame descriptors and write to the TDAR and RDAR. Refer to Section 29.6.5, “Buffer Descriptors” for more details.

29.5.5 Network Interface Options

The FEC supports both an MII interface for 10/100 Mbps Ethernet and a 7-wire serial interface for 10 Mbps Ethernet. The interface mode is selected by the RCR[MII_MODE] bit. In MII mode (RCR[MII_MODE] = 1), there are 18 signals defined by the IEEE 802.3 standard and supported by the EMAC. These signals are shown in Table 29-5 below.

Table 29-5. MII Mode

Signal Description	EMAC pin
Transmit Clock	FEC_TX_CLK
Transmit Enable	FEC_TX_EN
Transmit Data	FEC_TXD[3:0]

Table 29-5. MII Mode (continued)

Signal Description	EMAC pin
Transmit Error	FEC_TX_ER
Collision	FEC_COL
Carrier Sense	FEC_CRS
Receive Clock	FEC_RX_CLK
Receive Data Valid	FEC_RX_DV
Receive Data	FEC_RXD[3:0]
Receive Error	FEC_RX_ER
Management Data Clock	FEC_MDC
Management Data Input/Output	FEC_MDIO

The 7-wire serial mode interface (RCR[MII_MODE] = 0) operates in what is generally referred to as the "AMD" mode. 7-wire mode connections to the external transceiver are shown in Table 29-6.

Table 29-6. 7-Wire Mode Configuration

SIGNAL DESCRIPTION	EMAC PIN
Transmit Clock	FEC_TX_CLK
Transmit Enable	FEC_TX_EN
Transmit Data	FEC_TXD[0]
Collision	FEC_COL
Receive Clock	FEC_RX_CLK
Receive Data Valid	FEC_RX_DV
Receive Data	FEC_RXD[0]

29.5.6 FEC Frame Transmission

The Ethernet transmitter is designed to work with almost no intervention from software. Once ECR[ETHER_EN] is asserted and data appears in the transmit FIFO, the Ethernet MAC is able to transmit onto the network.

When the transmit FIFO fills to the watermark (defined by the TFWR), the MAC transmit logic will assert FEC_TX_EN and start transmitting the preamble (PA) sequence, the start frame delimiter (SFD), and then the frame information from the FIFO. However, the controller defers the transmission if the network is busy (FEC_CRS asserts). Before transmitting, the controller waits for carrier sense to become inactive, then determines if carrier sense stays inactive for 60 bit times. If so, the transmission begins after waiting an additional 36 bit times (96 bit times after carrier sense originally became inactive). See Section 29.5.14.1, "Transmission Errors" for more details.

If a collision occurs during transmission of the frame (half duplex mode), the Ethernet controller follows the specified backoff procedures and attempts to retransmit the frame until the retry limit is reached. The transmit FIFO stores at least the first 64 bytes of the transmit frame, so that they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data has been transmitted, the FCS (Frame Check Sequence or 32-bit Cyclic Redundancy Check, CRC) bytes are appended if the TC bit is set in the transmit frame control word. If the ABC bit is set in the transmit frame control word, a bad CRC will be appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the Ethernet controller writes the frame status information to the MIB block. Short frames are automatically padded by the transmit logic (if the TC bit in the transmit buffer descriptor for the end of frame buffer = 1).

Both buffer (TXB) and frame (TXF) interrupts may be generated as determined by the settings in the EIMR.

The transmit error interrupts are HBERR, BABT, LATE_COL, COL_RETRY_LIM, and XFIFO_UN. If the transmit frame length exceeds MAX_FL bytes the BABT interrupt will be asserted, however the entire frame will be transmitted (no truncation).

To pause transmission, set the GTS (graceful transmit stop) bit in the TCR register. When the TCR[GTS] is set, the FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame either finishes or terminates with a collision. After the transmitter has stopped the GRA (graceful stop complete) interrupt is asserted. If TCR[GTS] is cleared, the FEC resumes transmission with the next frame.

The Ethernet controller transmits bytes least significant bit first.

29.5.7 FEC Frame Reception

The FEC receiver is designed to work with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking, and maximum frame length checking.

When the driver enables the FEC receiver by asserting ECR[ETHER_EN], it will immediately start processing receive frames. When FEC_RX_DV asserts, the receiver will first check for a valid PA/SFD header. If the PA/SFD is valid, it will be stripped and the frame will be processed by the receiver. If a valid PA/SFD is not found, the frame will be ignored.

In serial mode, the first 16 bit times of RX_D0 following assertion of FEC_RX_DV are ignored. Following the first 16 bit times the data sequence is checked for alternating 1/0s. If a 11 or 00 data sequence is detected during bit times 17 to 21, the remainder of the frame is ignored. After bit time 21, the data sequence is monitored for a valid SFD (11). If a 00 is detected, the frame is rejected. When a 11 is detected, the PA/SFD sequence is complete.

In MII mode, the receiver checks for at least one byte matching the SFD. Zero or more PA bytes may occur, but if a 00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first 6 bytes of the frame have been received, the FEC performs address recognition on the frame.

Once a collision window (64 bytes) of data has been received and if address recognition has not rejected the frame, the receive FIFO is signalled that the frame is “accepted” and may be passed on to the DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to “reject” the frame. Thus, no collision fragments are presented to the user except late collisions, which indicate serious LAN problems.

During reception, the Ethernet controller checks for various error conditions and once the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, CR, OV and TR status bits, and the frame length. See Section 29.5.14.2, “Reception Errors” for more details.

Receive Buffer (RXB) and Frame Interrupts (RXF) may be generated if enabled by the EIMR register. A receive error interrupt is babbling receiver error (BABR). Receive frames are not truncated if they exceed the max frame length (MAX_FL); however, the BABR interrupt will occur and the LG bit in the Receive Buffer Descriptor (RxBD) will be set. See Section 29.6.5.2, “Ethernet Receive Buffer Descriptor (RxBD)” for more details.

When the receive frame is complete, the FEC sets the L-bit in the RxBD, writes the other frame status bits into the RxBD, and clears the E-bit. The Ethernet controller next generates a maskable interrupt (RXF bit in EIR, maskable by RXF bit in EIMR), indicating that a frame has been received and is in memory. The Ethernet controller then waits for a new frame.

The Ethernet controller receives serial data LSB first.

29.5.8 Ethernet Address Recognition

The FEC filters the received frames based on destination address (DA) type — individual (unicast), group (multicast), or broadcast (all-ones group address). The difference between an individual address and a group address is determined by the I/G bit in the destination address field. A flowchart for address recognition on received frames is illustrated in the figures below.

Address recognition is accomplished through the use of the receive block and microcode running on the microcontroller. The flowchart shown in Figure 29-2 illustrates the address recognition decisions made by the receive block, while Figure 29-3 illustrates the decisions made by the microcontroller.

If the DA is a broadcast address and broadcast reject (RCR[BC_REJ]) is deasserted, then the frame will be accepted unconditionally, as shown in Figure 29-2. Otherwise, if the DA is not a broadcast address, then the microcontroller runs the address recognition subroutine, as shown in Figure 29-3.

If the DA is a group (multicast) address and flow control is disabled, then the microcontroller will perform a group hash table lookup using the 64-entry hash table programmed in GAUR and GALR. If a hash match occurs, the receiver accepts the frame.

If flow control is enabled, the microcontroller will do an exact address match check between the DA and the designated PAUSE DA (01:80:C2:00:00:01). If the receive block determines that the received frame is a valid PAUSE frame, then the frame will be rejected. Note the receiver will detect a PAUSE frame with the DA field set to either the designated PAUSE DA or the unicast physical address.

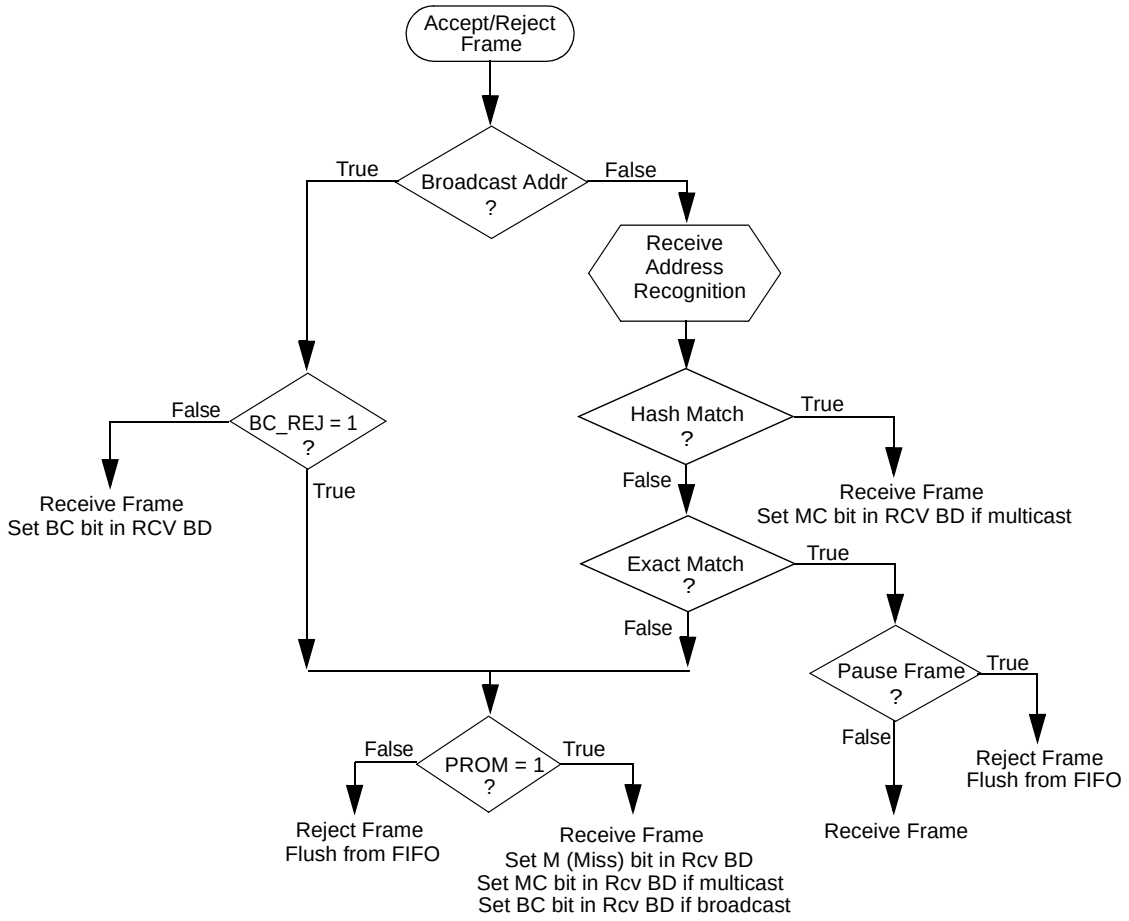
If the DA is the individual (unicast) address, the microcontroller performs an individual exact match comparison between the DA and 48-bit physical address that the user programs in the PALR and PAUR

registers. If an exact match occurs, the frame is accepted; otherwise, the microcontroller does an individual hash table lookup using the 64-entry hash table programmed in registers, IAUR and IALR. In the case of an individual hash match, the frame is accepted. Again, the receiver will accept or reject the frame based on PAUSE frame detection, shown in Figure 29-2.

If neither a hash match (group or individual), nor an exact match occur, then if promiscuous mode is enabled (RCR[PROM] = 1), then the frame will be accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame will be rejected.

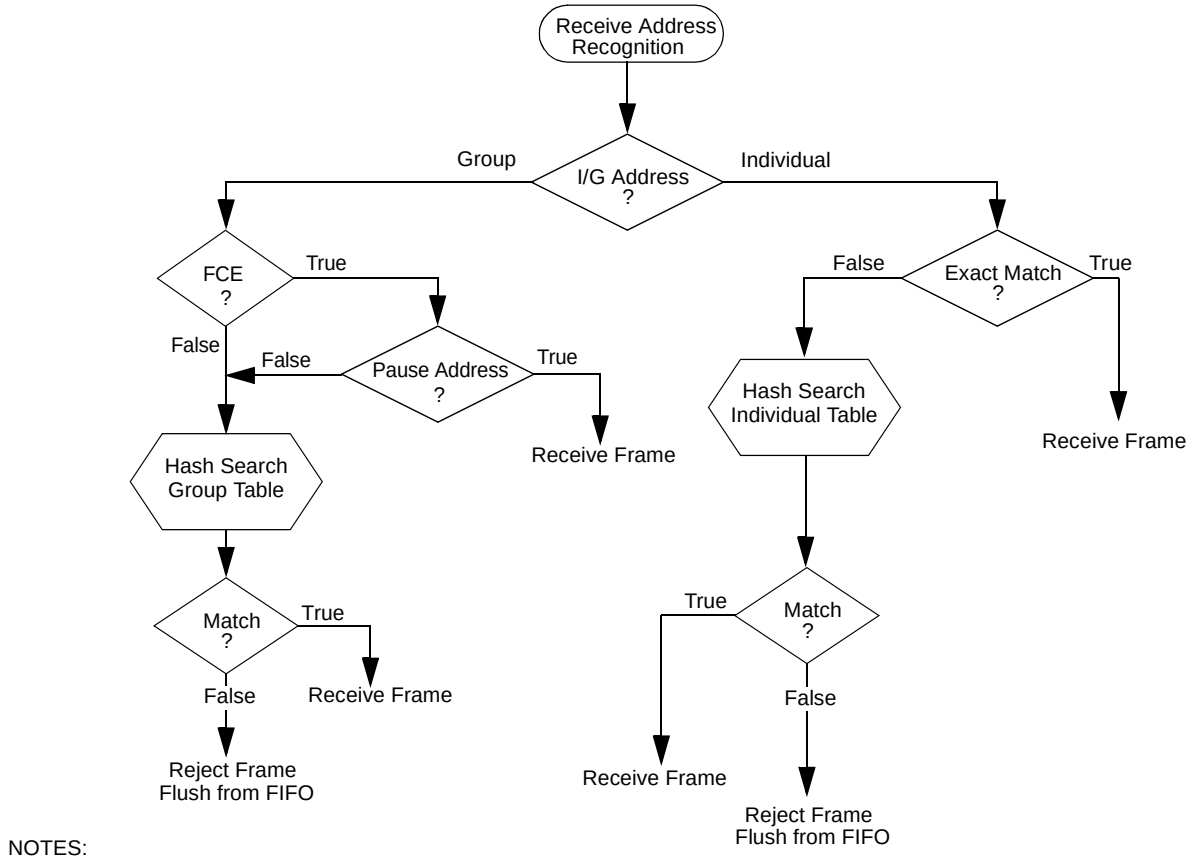
Similarly, if the DA is a broadcast address, broadcast reject (RCR[BC_REJ]) is asserted, and promiscuous mode is enabled, then the frame will be accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame will be rejected.

In general, when a frame is rejected, it is flushed from the FIFO.



NOTES:
BC_REJ - field in RCR register (BroadCast REJect)
PROM - field in RCR register
Pause Frame - valid PAUSE frame received

Figure 29-2. Ethernet Address Recognition—Receive Block Decisions



NOTES:
FCE - field in RCR register (Flow Control Enable)
I/G - Individual/Group bit in Destination Address (least significant bit in first byte received in MAC frame)

Figure 29-3. Ethernet Address Recognition—Microcode Decisions

29.5.9 Hash Algorithm

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, which are represented by 64 bits stored in GAUR, GALR (group address hash match) or IAUR, IALR (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the 6 most significant bits of the CRC-encoded result to generate a number between 0 and 63. The MSB of the CRC result selects GAUR (MSB = 1) or GALR (MSB = 0). The least significant 5 bits of the hash result select the bit within the selected register. If the CRC generator selects a bit that is set in the hash table, the frame is accepted; otherwise, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (or 87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The hash table registers must be initialized by the user. The CRC32 polynomial to use in computing the hash is:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$$

A table of example Destination Addresses and corresponding hash values is included below for reference.

Table 29-7. Destination Address to 6-Bit Hash

48-bit DA	6-bit Hash (in hex)	Hash Decimal Value
65:ff:ff:ff:ff:ff	0x0	0
55:ff:ff:ff:ff:ff	0x1	1
15:ff:ff:ff:ff:ff	0x2	2
35:ff:ff:ff:ff:ff	0x3	3
b5:ff:ff:ff:ff:ff	0x4	4
95:ff:ff:ff:ff:ff	0x5	5
d5:ff:ff:ff:ff:ff	0x6	6
f5:ff:ff:ff:ff:ff	0x7	7
db:ff:ff:ff:ff:ff	0x8	8
fb:ff:ff:ff:ff:ff	0x9	9
bb:ff:ff:ff:ff:ff	0xa	10
8b:ff:ff:ff:ff:ff	0xb	11
0b:ff:ff:ff:ff:ff	0xc	12
3b:ff:ff:ff:ff:ff	0xd	13
7b:ff:ff:ff:ff:ff	0xe	14
5b:ff:ff:ff:ff:ff	0xf	15
27:ff:ff:ff:ff:ff	0x10	16
07:ff:ff:ff:ff:ff	0x11	17
57:ff:ff:ff:ff:ff	0x12	18
77:ff:ff:ff:ff:ff	0x13	19
f7:ff:ff:ff:ff:ff	0x14	20
c7:ff:ff:ff:ff:ff	0x15	21
97:ff:ff:ff:ff:ff	0x16	22
a7:ff:ff:ff:ff:ff	0x17	23
99:ff:ff:ff:ff:ff	0x18	24
b9:ff:ff:ff:ff:ff	0x19	25

Table 29-7. Destination Address to 6-Bit Hash (continued)

48-bit DA	6-bit Hash (in hex)	Hash Decimal Value
f9:ff:ff:ff:ff:ff	0x1a	26
c9:ff:ff:ff:ff:ff	0x1b	27
59:ff:ff:ff:ff:ff	0x1c	28
79:ff:ff:ff:ff:ff	0x1d	29
29:ff:ff:ff:ff:ff	0x1e	30
19:ff:ff:ff:ff:ff	0x1f	31
d1:ff:ff:ff:ff:ff	0x20	32
f1:ff:ff:ff:ff:ff	0x21	33
b1:ff:ff:ff:ff:ff	0x22	34
91:ff:ff:ff:ff:ff	0x23	35
11:ff:ff:ff:ff:ff	0x24	36
31:ff:ff:ff:ff:ff	0x25	37
71:ff:ff:ff:ff:ff	0x26	38
51:ff:ff:ff:ff:ff	0x27	39
7f:ff:ff:ff:ff:ff	0x28	40
4f:ff:ff:ff:ff:ff	0x29	41
1f:ff:ff:ff:ff:ff	0x2a	42
3f:ff:ff:ff:ff:ff	0x2b	43
bf:ff:ff:ff:ff:ff	0x2c	44
9f:ff:ff:ff:ff:ff	0x2d	45
df:ff:ff:ff:ff:ff	0x2e	46
ef:ff:ff:ff:ff:ff	0x2f	47
93:ff:ff:ff:ff:ff	0x30	48
b3:ff:ff:ff:ff:ff	0x31	49
f3:ff:ff:ff:ff:ff	0x32	50
d3:ff:ff:ff:ff:ff	0x33	51
53:ff:ff:ff:ff:ff	0x34	52
73:ff:ff:ff:ff:ff	0x35	53
23:ff:ff:ff:ff:ff	0x36	54
13:ff:ff:ff:ff:ff	0x37	55
3d:ff:ff:ff:ff:ff	0x38	56
0d:ff:ff:ff:ff:ff	0x39	57

Table 29-7. Destination Address to 6-Bit Hash (continued)

48-bit DA	6-bit Hash (in hex)	Hash Decimal Value
5d:ff:ff:ff:ff:ff	0x3a	58
7d:ff:ff:ff:ff:ff	0x3b	59
fd:ff:ff:ff:ff:ff	0x3c	60
dd:ff:ff:ff:ff:ff	0x3d	61
9d:ff:ff:ff:ff:ff	0x3e	62
bd:ff:ff:ff:ff:ff	0x3f	63

29.5.10 Full Duplex Flow Control

Full-duplex flow control allows the user to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, MAC data frame transmission stops for a given pause duration.

To enable pause frame detection, the FEC must operate in full-duplex mode (TCR[FDEN] asserted) and flow control enable (RCR[FCE]) must be asserted. The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications, as shown in the table below. In addition, the receive status associated with the frame should indicate that the frame is valid.

48-bit Destination Address	0x0180_c200_0001 or Physical Address
48-bit Source Address	Any
16-bit Type	0x8808
16-bit Opcode	0x0001
16-bit PAUSE Duration	0x0000 to 0xFFFF

Pause frame detection is performed by the receiver and microcontroller modules. The microcontroller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the type and opcode pause frame fields. On detection of a pause frame, TCR[GTS] is asserted by the FEC internally. When transmission has paused, the EIR[GRA] interrupt is asserted and the pause timer begins to increment. Note that the pause timer makes use of the transmit back-off timer hardware, which is used for tracking the appropriate collision back-off timer in half-duplex mode. The pause timer increments once every slot time, until OPD[PAUSE_DUR] slot times have expired. On OPD[PAUSE_DUR] expiration, TCR[GTS] is deasserted allowing MAC data frame transmission to resume. Note that the receive flow control pause (TCR[RFC_PAUSE]) status bit is asserted while the transmitter is paused due to reception of a pause frame.

To transmit a pause frame, the FEC must operate in full-duplex mode and the user must assert flow control pause (TCR[TFC_PAUSE]). On assertion of transmit flow control pause (TCR[TFC_PAUSE]), the transmitter asserts TCR[GTS] internally. When the transmission of data frames stops, the EIR[GRA] (graceful stop complete) interrupt asserts. Following EIR[GRA] assertion, the pause frame is transmitted. On completion of pause frame transmission, flow control pause (TCR[TFC_PAUSE]) and TCR[GTS] are deasserted internally.

The user must specify the desired pause duration in the OPD register.

Note that when the transmitter is paused due to receiver/microcontroller pause frame detection, transmit flow control pause (TCR[TFC_PAUSE]) still may be asserted and will cause the transmission of a single pause frame. In this case, the EIR[GRA] interrupt will not be asserted.

29.5.11 Inter-Packet Gap (IPG) Time

The minimum inter-packet gap time for back-to-back transmission is 96 bit times. After completing a transmission or after the back-off algorithm completes, the transmitter waits for carrier sense to be negated before starting its 96 bit time IPG counter. Frame transmission may begin 96 bit times after carrier sense is negated if it stays negated for at least 60 bit times. If carrier sense asserts during the last 36 bit times, it will be ignored and a collision will occur.

The receiver receives back-to-back frames with a minimum spacing of at least 28 bit times. If an inter-packet gap between receive frames is less than 28 bit times, the following frame may be discarded by the receiver.

29.5.12 Collision Handling

If a collision occurs during frame transmission, the Ethernet controller will continue the transmission for at least 32 bit times, transmitting a JAM pattern consisting of 32 ones. If the collision occurs during the preamble sequence, the JAM pattern will be sent after the end of the preamble sequence.

If a collision occurs within 512 bit times, the retry process is initiated. The transmitter waits a random number of slot times. A slot time is 512 bit times. If a collision occurs after 512 bit times, then no retransmission is performed and the end of frame buffer is closed with a Late Collision (LC) error indication.

29.5.13 Internal and External Loopback

Both internal and external loopback are supported by the Ethernet controller. In loopback mode, both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Both internal and external loopback are configured using combinations of the LOOP and DRT bits in the RCR register and the FDEN bit in the TCR register.

For both internal and external loopback set FDEN = 1.

For internal loopback set RCR[LOOP] = 1 and RCR[DRT] = 0. FEC_TX_EN and FEC_TX_ER will not assert during internal loopback. During internal loopback, the transmit/receive data rate is higher than in normal operation because the internal system clock is used by the transmit and receive blocks instead of the clocks from the external transceiver. This will cause an increase in the required system bus bandwidth for transmit and receive data being DMA'd to/from external memory. It may be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underrun and receive FIFO overflow.

For external loopback set RCR[LOOP] = 0, RCR[DRT] = 0 and configure the external transceiver for loopback.

29.5.14 Ethernet Error-Handling Procedure

The Ethernet controller reports frame reception and transmission error conditions using the FEC RxBDs, the EIR register, and the MIB block counters.

29.5.14.1 Transmission Errors

29.5.14.1.1 Transmitter Underrun

If this error occurs, the FEC sends 32 bits that ensure a CRC error and stops transmitting. All remaining buffers for that frame are then flushed and closed. The UN bit is set in the EIR. The FEC will then continue to the next transmit buffer descriptor and begin transmitting the next frame.

The “UN” interrupt will be asserted if enabled in the EIMR register.

29.5.14.1.2 Retransmission Attempts Limit Expired

When this error occurs, the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and the RL bit is set in the EIR. The FEC will then continue to the next transmit buffer descriptor and begin transmitting the next frame.

The “RL” interrupt will be asserted if enabled in the EIMR register.

29.5.14.1.3 Late Collision

When a collision occurs after the slot time (512 bits starting at the Preamble), the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and the LC bit is set in the EIR register. The FEC will then continue to the next transmit buffer descriptor and begin transmitting the next frame.

The “LC” interrupt will be asserted if enabled in the EIMR register.

29.5.14.1.4 Heartbeat

Some transceivers have a self-test feature called “heartbeat” or “signal quality error.” To signify a good self-test, the transceiver indicates a collision to the FEC within 4 microseconds after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver still seems to be functioning properly. This is called the heartbeat condition.

If the HBC bit is set in the TCR register and the heartbeat condition is not detected by the FEC after a frame transmission, then a heartbeat error occurs. When this error occurs, the FEC closes the buffer, sets the HB bit in the EIR register, and generates the HBERR interrupt if it is enabled.

29.5.14.2 Reception Errors

29.5.14.2.1 Overrun Error

If the receive block has data to put into the receive FIFO and the receive FIFO is full, the FEC sets the OV bit in the RxBD. All subsequent data in the frame will be discarded and subsequent frames may also be discarded until the receive FIFO is serviced by the DMA and space is made available. At this point the receive frame/status word is written into the FIFO with the OV bit set. This frame must be discarded by the driver.

29.5.14.2.2 Non-Octet Error (Dribbling Bits)

The Ethernet controller handles up to seven dribbling bits when the receive frame terminates past an non-octet aligned boundary. Dribbling bits are not used in the CRC calculation. If there is a CRC error, then the frame non-octet aligned (NO) error is reported in the RxBD. If there is no CRC error, then no error is reported.

29.5.14.2.3 CRC Error

When a CRC error occurs with no dribble bits, the FEC closes the buffer and sets the CR bit in the RxBD. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

29.5.14.2.4 Frame Length Violation

When the receive frame length exceeds MAX_FL bytes the BABR interrupt will be generated, and the LG bit in the end of frame RxBD will be set. The frame is not truncated unless the frame length exceeds 2047 bytes).

29.5.14.2.5 Truncation

When the receive frame length exceeds 2047 bytes the frame is truncated and the TR bit is set in the receive BD.

29.6 Memory Map and Register Definition

This section gives an overview of the registers, followed by a description of the buffers.

The FEC is programmed by a combination of control/status registers (CSRs) and buffer descriptors. The CSRs are used for mode control and to extract global status information. The descriptors are used to pass data buffers and related buffer information between the hardware and software.

29.6.1 High-Level Module Memory Map

The FEC implementation requires a 1-Kbyte memory map space. This is divided into 2 sections of 512 bytes each. The first is used for control/status registers. The second contains event/statistic counters held in the MIB block. Table 29-8 defines the top level memory map.

Table 29-8. Module Memory Map

Address	Function
0x1002_B + 0x000-1FF	Control/Status Registers
0x1002_B + 0x200-3FF	MIB Block Counters

29.6.2 Detailed Memory Map (Control/Status Registers)

Table 29-9 shows the FEC register memory map with each register address, name, and a brief description.

Table 29-9. FEC Register Memory Map

Address	Register	Access	Reset Value	Section/Page
0x1002_B004 (EIR)	Interrupt Event Register	R/W	0x0000_0000	29.6.4.1/29-21
0x1002_B008 (EIMR)	Interrupt Mask Register	R/W	0x0000_0000	29.6.4.2/29-23
0x1002_B010 (RDAR)	Receive Descriptor Active Register	R/W	0x0000_0000	29.6.4.3/29-23
0x1002_B014 (TDAR)	Transmit Descriptor Active Register	R/W	0x0000_0000	29.6.4.4/29-24
0x1002_B024 (ECR)	Ethernet Control Register	R/W	0xF000_0000	29.6.4.5/29-25
0x1002_B040 (MMFR)	MII Management Frame Register	R/W	Undefined	29.6.4.6/29-25
0x1002_B044 (MSCR)	MII Speed Control Register	R/W	0x0000_0000	29.6.4.7/29-27
0x1002_B064 (MIBC)	MIB Control/Status Register	R/W	0x0000_0000	29.6.4.8/29-28
0x1002_B084 (RCR)	Receive Control Register	R/W	0x05EE_0001	29.6.4.9/29-29
0x1002_B0C4 (TCR)	Transmit Control Register	R/W	0x0000_0000	29.6.4.10/29-30
0x1002_B0E4 (PALR)	Physical Address Low Register	R/W	Undefined	29.6.4.11/29-30
0x1002_B0E8 (PAUR)	Physical Address High Register	R/W	See Section	29.6.4.12/29-31
0x1002_B0EC (OPD)	Opcode/Pause Duration	R/W	See Section	29.6.4.13/29-31
0x1002_B118 (IAUR)	Descriptor Individual Upper Address Register	R/W	Undefined	29.6.4.14/29-32
0x1002_B11C (IALR)	Descriptor Individual Lower Address Register	R/W	Undefined	29.6.4.15/29-32

Table 29-9. FEC Register Memory Map

Address	Register	Access	Reset Value	Section/Page
0x1002_B120 (GAUR)	Descriptor Group Upper Address Register	R/W	Undefined	29.6.4.16/29-33
0x1002_B124 (GALR)	Descriptor Group Lower Address Register	R/W	Undefined	29.6.4.17/29-33
0x1002_B144 (TFWR)	Transmit FIFO Watermark	R/W	0x0000_0001	29.6.4.18/29-34
0x1002_B14C (FRBR)	FIFO Receive Bound Register	R	0x0000_0600	29.6.4.19/29-34
0x1002_B150 (FRSR)	FIFO Receive FIFO Start Register	R	0x0000_0500	29.6.4.20/29-35
0x1002_B180 (ERDSR)	Pointer to Receive Descriptor Ring	R/W	Undefined	29.6.4.21/29-35
0x1002_B184 (ETDSR)	Pointer to Transmit Descriptor Ring	R/W	Undefined	29.6.4.22/29-36
0x1002_B188 (EMRBR)	Maximum Receive Buffer Size	R/W	Undefined	29.6.4.23/29-36

29.6.3 MIB Block Counters Memory Map

Table 29-10 defines the MIB Counters memory map which defines the locations in the MIB RAM space where hardware maintained counters reside. These fall in the 0x200-0x3FF address offset range. The counters are divided into two groups.

RMON counters are included which cover the Ethernet Statistics counters defined in RFC 1757. In addition to the counters defined in the Ethernet Statistics group, a counter is included to count truncated frames as the FEC only supports frame lengths up to 2047 bytes. The RMON counters are implemented independently for transmit and receive to insure accurate network statistics when operating in full duplex mode.

IEEE counters are included which support the Mandatory and Recommended counter packages defined in section 5 of ANSI/IEEE Std. 802.3 (1998 edition). The IEEE Basic Package objects are supported by the FEC but do not require counters in the MIB block. In addition, some of the recommended package objects which are supported do not require MIB counters. Counters for transmit and receive full duplex flow control frames are included as well.

Table 29-10. MIB Counters Memory Map

Offset	Mnemonic	Description
0x200	RMON_T_DROP	Count of frames not counted correctly
0x204	RMON_T_PACKETS	RMON Tx packet count
0x208	RMON_T_BC_PKT	RMON Tx Broadcast Packets
0x20C	RMON_T_MC_PKT	RMON Tx Multicast Packets

Table 29-10. MIB Counters Memory Map (continued)

Offset	Mnemonic	Description
0x210	RMON_T_CRC_ALIGN	RMON Tx Packets w CRC/Align error
0x214	RMON_T_UNDERSIZE	RMON Tx Packets < 64 bytes, good crc
0x218	RMON_T_OVERSIZE	RMON Tx Packets > MAX_FL bytes, good crc
0x21C	RMON_T_FRAG	RMON Tx Packets < 64 bytes, bad crc
0x220	RMON_T_JAB	RMON Tx Packets > MAX_FL bytes, bad crc
0x224	RMON_T_COL	RMON Tx collision count
0x228	RMON_T_P64	RMON Tx 64 byte packets
0x22C	RMON_T_P65TO127	RMON Tx 65 to 127 byte packets
0x230	RMON_T_P128TO255	RMON Tx 128 to 255 byte packets
0x234	RMON_T_P256TO511	RMON Tx 256 to 511 byte packets
0x238	RMON_T_P512TO1023	RMON Tx 512 to 1023 byte packets
0x23C	RMON_T_P1024TO2047	RMON Tx 1024 to 2047 byte packets
0x240	RMON_T_P_GTE2048	RMON Tx packets w > 2048 bytes
0x244	RMON_T_OCTETS	RMON Tx Octets
0x248	IEEE_T_DROP	Count of frames not counted correctly
0x24C	IEEE_T_FRAME_OK	Frames Transmitted OK
0x250	IEEE_T_1COL	Frames Transmitted with Single Collision
0x254	IEEE_T_MCOL	Frames Transmitted with Multiple Collisions
0x258	IEEE_T_DEF	Frames Transmitted after Deferral Delay
0x25C	IEEE_T_LCOL	Frames Transmitted with Late Collision
0x260	IEEE_T_EXCOL	Frames Transmitted with Excessive Collisions
0x264	IEEE_T_MACERR	Frames Transmitted with Tx FIFO Underrun
0x268	IEEE_T_CSERR	Frames Transmitted with Carrier Sense Error
0x26C	IEEE_T_SQE	Frames Transmitted with SQE Error
0x270	IEEE_T_FDXFC	Flow Control Pause frames transmitted
0x274	IEEE_T_OCTETS_OK	Octet count for Frames Transmitted w/o Error
0x284	RMON_R_PACKETS	RMON Rx packet count
0x288	RMON_R_BC_PKT	RMON Rx Broadcast Packets
0x28C	RMON_R_MC_PKT	RMON Rx Multicast Packets
0x290	RMON_R_CRC_ALIGN	RMON Rx Packets w CRC/Align error
0x294	RMON_R_UNDERSIZE	RMON Rx Packets < 64 bytes, good crc
0x298	RMON_R_OVERSIZE	RMON Rx Packets > MAX_FL bytes, good crc
0x29C	RMON_R_FRAG	RMON Rx Packets < 64 bytes, bad crc

Table 29-10. MIB Counters Memory Map (continued)

Offset	Mnemonic	Description
0x2A0	RMON_R_JAB	RMON Rx Packets > MAX_FL bytes, bad crc
0x2A4	RMON_R_RESVD_0	
0x2A8	RMON_R_P64	RMON Rx 64 byte packets
0x2AC	RMON_R_P65TO127	RMON Rx 65 to 127 byte packets
0x2B0	RMON_R_P128TO255	RMON Rx 128 to 255 byte packets
0x2B4	RMON_R_P256TO511	RMON Rx 256 to 511 byte packets
0x2B8	RMON_R_P512TO1023	RMON Rx 512 to 1023 byte packets
0x2BC	RMON_R_P1024TO2047	RMON Rx 1024 to 2047 byte packets
0x2C0	RMON_R_P_GTE2048	RMON Rx packets w > 2048 bytes
0x2C4	RMON_R_OCTETS	RMON Rx Octets
0x2C8	IEEE_R_DROP	Count of frames not counted correctly
0x2CC	IEEE_R_FRAME_OK	Frames Received OK
0x2D0	IEEE_R_CRC	Frames Received with CRC Error
0x2D4	IEEE_R_ALIGN	Frames Received with Alignment Error
0x2D8	IEEE_R_MACERR	Receive Fifo Overflow count
0x2DC	IEEE_R_FDXFC	Flow Control Pause frames received
0x2E0	IEEE_R_OCTETS_OK	Octet count for Frames Rcvd w/o Error

29.6.4 Register Descriptions

The following sections describe each register in detail.

29.6.4.1 Ethernet Interrupt Event Register (EIR)

When an event occurs that sets a bit in the EIR, an interrupt will be generated if the corresponding bit in the interrupt mask register (EIMR) is also set. The bit in the EIR is cleared if a one is written to that bit position; writing zero has no effect. This register is cleared upon hardware reset.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts which may occur in normal operation are GRA, TXF, TXB, RXF, RXB, and MII. Interrupts resulting from errors/problems detected in the network or transceiver are HBERR, BABR, BABT, LC and RL. Interrupts resulting from internal errors are EBERR and UN.

Some of the error interrupts are independently counted in the MIB block counters. Software may choose to mask off these interrupts since these errors will be visible to network management via the MIB counters.

- HBERR - IEEE_T_SQE
- BABR - RMON_R_OVERSIZE (good CRC), RMON_R_JAB (bad CRC)
- BABT - RMON_T_OVERSIZE (good CRC), RMON_T_JAB (bad CRC)

- LATE_COL - IEEE_T_LCOL
- COL_RETRY_LIM - IEEE_T_EXCOL
- XFIFO_UN - IEEE_T_MACERR

0x1002_B004 (EIR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HB ERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EB ERR	LC	RL	UN	0	0	0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 29-4. Ethernet Interrupt Event Register (EIR)

Table 29-11. EIR Field Descriptions

Field	Description
31 HBERR	Heartbeat error. Indicates TCR[HBC] is set and that the COL input was not asserted within the heartbeat window following a transmission.
30 BABR	Babbling receive error. Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
29 BABT	Babbling transmit error. Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused by a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
28 GRA	Graceful stop complete. Indicates the graceful stop is complete. During graceful stop the transmitter is placed into a pause state after completion of the frame currently being transmitted. This bit is set by one of three conditions: 1) A graceful stop initiated by the setting of the TCR[GTS] bit is now complete. 2) A graceful stop initiated by the setting of the TCR[TFC_PAUSE] bit is now complete. 3) A graceful stop initiated by the reception of a valid full duplex flow control pause frame is now complete. Refer to Section 29.5.10, “Full Duplex Flow Control.”
27 TXF	Transmit frame interrupt. Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
26 TXB	Transmit buffer interrupt. Indicates a transmit buffer descriptor has been updated.
25 RXF	Receive frame interrupt. Indicates a frame has been received and the last corresponding buffer descriptor has been updated.
24 RXB	Receive buffer interrupt. Indicates a receive buffer descriptor not the last in the frame has been updated.
23 MII	MII interrupt. Indicates the MII has completed the data transfer requested.
22 EBERR	Ethernet bus error. Indicates a system bus error occurred when a DMA transaction is underway. When the EBERR bit is set, ECR[ETHER_EN] is cleared, halting frame processing by the FEC. When this occurs, software needs to insure that the FIFO controller and DMA also soft reset.

Table 29-11. EIR Field Descriptions (continued)

Field	Description
21 LC	Late collision. Indicates a collision occurred beyond the collision window (slot time) in half duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.
20 RL	Collision retry limit. Indicates a collision occurs on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half duplex mode.
19 UN	Transmit FIFO underrun. Indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
18–0	Reserved, must be cleared.

29.6.4.2 Interrupt Mask Register (EIMR)

The EIMR controls which interrupt events are allowed to generate actual interrupts. All implemented bits in this CSR are read/write. This register is cleared upon a hardware reset. If the corresponding bits in both the EIR and EIMR are set, the interrupt will be signalled to the CPU. The interrupt signal will remain asserted until a 1 is written to the EIR bit (write 1 to clear) or a 0 is written to the EIMR bit.

0x1002_B008 (EIMR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HB ERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EB ERR	LC	RL	UN	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 29-5. EIMR Register

Table 29-12. EIMR Field Descriptions

Field	Description
31–19 See Figure 29-5 and Table 29-11	Interrupt mask. Each bit corresponds to an interrupt source defined by the EIR register. The corresponding EIMR bit determines whether an interrupt condition can generate an interrupt. At every processor clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR bit reflects the state of the interrupt signal even if the corresponding EIMR bit is set. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
18–0	Reserved, must be cleared.

29.6.4.3 Receive Descriptor Active Register (RDAR)

RDAR is a command register, written by the user, that indicates that the receive descriptor ring has been updated (empty receive buffers have been produced by the driver with the empty bit set).

Whenever the register is written, the RDAR bit is set. This is independent of the data actually written by the user. When set, the FEC will poll the receive descriptor ring and process receive frames (provided ECR[ETHER_EN] is also set). Once the FEC polls a receive descriptor whose empty bit is not set, then the FEC will clear the RDAR bit and cease receive descriptor ring polling until the user sets the bit again, signifying that additional descriptors have been placed into the receive descriptor ring.

The RDAR is cleared at reset and when ECR[ETHER_EN] is cleared.

0x1002_B010 (RDAR) Access: User read/write

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

Figure 29-6. RDAR Register

Table 29-13. RDAR Field Descriptions

Field	Description
31–25	Reserved, must be cleared.
24 RDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC device when no additional empty descriptors remain in the receive ring. Also cleared when ECR[ETHER_EN] is cleared.
23–0	Reserved, must be cleared.

29.6.4.4 Transmit Descriptor Active Register (TDAR)

The TDAR is a command register which should be written by the user to indicate that the transmit descriptor ring has been updated (transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor).

Whenever the register is written, the TDAR bit is set. This value is independent of the data actually written by the user. When set, the FEC will poll the transmit descriptor ring and process transmit frames (provided ECR[ETHER_EN] is also set). Once the FEC polls a transmit descriptor whose ready bit is not set, then the FEC will clear the TDAR bit and cease transmit descriptor ring polling until the user sets the bit again, signifying additional descriptors have been placed into the transmit descriptor ring.

The TDAR is cleared at reset, when ECR[ETHER_EN] is cleared, or when ECR[RESET] is set.

0x1002_B014 (TDAR) Access: User read/write

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

Figure 29-7. TDAR Register

Table 29-14. TDAR Field Descriptions

Field	Description
31–25	Reserved, must be cleared.
24 TDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHER_EN] is cleared.
23–0	Reserved, must be cleared.

29.6.4.5 Ethernet Control Register (ECR)

ECR is a read/write user register, though both fields in this register may be altered by hardware as well. The ECR is used to enable/disable the FEC.

0x1002_B024 (ECR) Access: User read/write

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

Figure 29-8. ECR Register

Table 29-15. ECR Field Descriptions

Field	Description
31–2	Reserved, must be cleared.
1 ETHER_EN	When this bit is set, FEC is enabled, and reception and transmission are possible. When this bit is cleared, reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. The buffer descriptor(s) for an aborted transmit frame are not updated after clearing this bit. When ETHER_EN is cleared, the DMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware alters the ETHER_EN bit under the following conditions: • ECR[RESET] is set by software, in which case ETHER_EN is cleared • An error condition causes the EIR[EBERR] bit to set, in which case ETHER_EN is cleared
0 RESET	When this bit is set, the equivalent of a hardware reset is performed but it is local to the FEC. ETHER_EN is cleared and all other FEC registers take their reset values. Also, any transmission/reception currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately 8 internal bus clock cycles after RESET is set.

29.6.4.6 MII Management Frame Register (MMFR)

The MMFR is accessed by the user and does not reset to a defined value. The MMFR is used to communicate with the attached MII compatible PHY device(s), providing read/write access to their MII registers. Performing a write to the MMFR will cause a management frame to be sourced unless the MSCR has been programmed to 0. In the case of writing to MMFR when MSCR = 0, if the MSCR is then written to a non-zero value, an MII frame will be generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

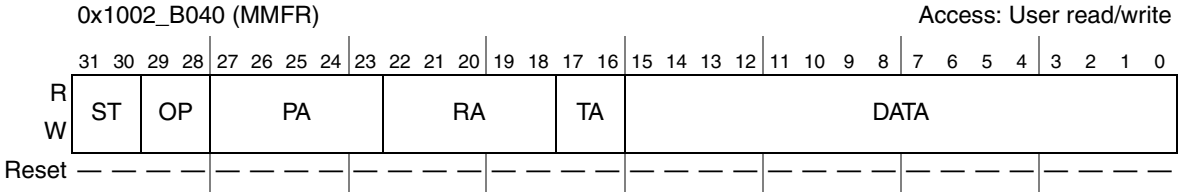


Figure 29-9. MMFR Register

Table 29-16. MMFR Field Descriptions

Field	Description
31–30 ST	Start of frame delimiter. These bits must be programmed to 0b01 for a valid MII management frame.
29–28 OP	Operation code. 00 Write frame operation, but not MII compliant. 01 Write frame operation for a valid MII management frame. 10 Read frame operation for a valid MII management frame. 11 Read frame operation, but not MII compliant.
27–23 PA	PHY address. This field specifies one of up to 32 attached PHY devices.
22–18 RA	Register address. This field specifies one of up to 32 registers within the specified PHY device.
17–16 TA	Turn around. This field must be programmed to 10 to generate a valid MII management frame.
15–0 DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

To perform a read or write operation on the MII Management Interface, the MMFR must be written by the user. To generate a valid read or write management frame, the ST field must be written with a 01 pattern, and the TA field must be written with a 10. If other patterns are written to these fields, a frame will be generated but will not comply with the IEEE 802.3 MII definition.

To generate an IEEE 802.3-compliant MII Management Interface write frame (write to a PHY register), the user must write {01 01 PHYAD REGAD 10 DATA} to the MMFR. Writing this pattern will cause the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time the contents of the MMFR will be altered as the contents are serially shifted and will be unpredictable if read by the user. Once the write management frame operation has completed, the MII interrupt will be generated. At this time the contents of the MMFR will match the original value written.

To generate an MII Management Interface read frame (read a PHY register) the user must write {01 10 PHYAD REGAD 10 XXXX} to the MMFR (the content of the DATA field is a don’t care). Writing this pattern will cause the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time the contents of the MMFR will be altered as the contents are serially shifted, and will be unpredictable if read by the user. Once the read management frame operation has completed, the MII interrupt will be generated. At this time the contents of the MMFR will match the

original value written except for the DATA field whose contents have been replaced by the value read from the PHY register.

If the MMFR is written while frame generation is in progress, the frame contents will be altered. Software should use the MII interrupt to avoid writing to the MMFR while frame generation is in progress.

29.6.4.7 MII Speed Control Register (MSCR)

The MSCR provides control of the MII clock (FEC_MDC pin) frequency, and allows a preamble drop on the MII management frame.

0x1002_B044 (MSCR) Access: User read/write

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

Figure 29-10. MSCR Register

Table 29-17. MSCR Field Descriptions

Field	Description
31–8	Reserved, must be cleared.
7 DIS_PRE	Asserting this bit causes preamble (32 1's) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY device(s) does not require it.
6–1 MII_SPEED	MII_SPEED controls the frequency of the MII management interface clock (FEC_MDC) relative to the internal bus clock. A value of 0 in this field turns off the FEC_MDC and leaves it in low voltage state. Any non-zero value results in the FEC_MDC frequency of 1/(MII_SPEED $\times$ 2) of the internal bus frequency.
0	Reserved, must be cleared.

The MII_SPEED field must be programmed with a value to provide an FEC_MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value in order to source a read or write management frame. After the management frame is complete the MSCR may optionally be set to zero to turn off the FEC_MDC. The FEC_MDC generated will have a 50% duty cycle except when MII_SPEED is changed during operation (change will take effect following either a rising or falling edge of FEC_MDC).

If the internal bus clock is 25 MHz, programming this register to 0x0000_0005 results in an FEC_MDC as stated the equation below.

$$25\ \text{MHz} \times \frac{1}{5 \times 2} = 2.5\ \text{MHz} \qquad \textit{Eqn. 29-1}$$

A table showing optimum values for MII_SPEED as a function of internal bus clock frequency is provided below.

Table 29-18. Programming Examples for MSCR

System Clock Frequency	MII_SPEED (field in reg)	FEC_MDC frequency
25 MHz	0x5	2.5 MHz
33 MHz	0x7	2.36 MHz
40 MHz	0x8	2.5 MHz
50 MHz	0xA	2.5 MHz
66 MHz	0xD	2.54 MHz

29.6.4.8 MIB Control Register (MIBC)

The MIB control register is a read/write register used to provide control of and to observe the state of the MIB block. This register is accessed by user software if there is a need to disable the MIB block operation. For example, in order to clear all MIB counters in RAM the user should disable the MIB block, then clear all the MIB RAM locations, then enable the MIB block. The MIB_DISABLE bit is reset to 1. See Table 29-10 for the locations of the MIB counters.

0x1002_B064 (MIBC) Access: User read/write

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

Figure 29-11. MIBC Register

Table 29-19. MIBC Field Descriptions

Field	Description
31 MIB_DIS	A read/write control bit. If set, the MIB logic halts and not update any MIB counters.
30 MIB_IDLE	A read-only status bit. If set the MIB block is not currently updating any MIB counters.
29–0	Reserved.

29.6.4.9 Receive Control Register (RCR)

The RCR is programmed by the user. The RCR controls the operational mode of the receive block and should be written only when ECR[ETHER_EN] = 0 (initialization time).

0x1002_B084 (RCR) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	FCE	BC_REJ	PROM	MII_MODE	DRT	LOOP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 29-12. RCR Register

Table 29-20. RCR Field Descriptions

Field	Description
31–27	Reserved, must be cleared.
26–16 MAX_FL	Maximum frame length. Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL causes the BABT interrupt to occur. Receive frames longer than MAX_FL causes the BABR interrupt to occur and sets the LG bit in the end of frame receive buffer descriptor. The recommended default value to be programmed by the user is 1518 or 1522 (if VLAN Tags are supported).
15–6	Reserved, must be cleared.
5 FCE	Flow control enable. If asserted, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter will stop transmitting data frames for a given duration.
4 BC_REJ	Broadcast frame reject. If asserted, frames with DA (destination address) equals FF_FF_FF_FF_FF_FF are rejected unless the PROM bit is set. If both BC_REJ and PROM equals 1, frames with broadcast DA are accepted and the M (MISS) is set in the receive buffer descriptor.
3 PROM	Promiscuous mode. All frames are accepted regardless of address matching.
2 MII_MODE	Media independent interface mode. Selects the external interface mode for both transmit and receive blocks. 0 7-wire mode (used only for serial 10 Mbps) 1 MII mode
1 DRT	Disable receive on transmit. 0 Receive path operates independently of transmit (use for full duplex or to monitor transmit activity in half duplex mode). 1 Disable reception of frames while transmitting (normally used for half duplex mode).
0 LOOP	Internal loopback. If set, transmitted frames are looped back internal to the device and transmit output signals are not asserted. The internal bus clock substitutes for the FEC_TXCLK when LOOP is asserted. DRT must be set to 0 when setting LOOP.

29.6.4.10 Transmit Control Register (TCR)

This register is read/write and is written by the user to configure the transmit block. This register is cleared at system reset. Bits 2 and 1 should be modified only when ECR[ETHER_EN] = 0.

0x1002_B0C4 (TCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RFC_PAUSE	TFC_PAUSE	FDEN	HBC	GTS
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-13. TCR Register

Table 29-21. TCR Field Descriptions

Field	Description
31–5	Reserved, must be cleared.
4 RFC_PAUSE	Receive frame control pause. This read-only status bit is asserted when a full duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This bit automatically clears when the pause duration is complete.
3 TFC_PAUSE	Transmit frame control pause. Transmits a PAUSE frame when asserted. When this bit is set, the MAC stops transmission of data frames after the current transmission is complete. At this time, GRA interrupt in the EIR register is asserted. With transmission of data frames stopped, MAC transmits a MAC Control PAUSE frame. Next, the MAC clears the TFC_PAUSE bit and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC Control PAUSE frame.
2 FDEN	Full duplex enable. If set, frames transmit independent of carrier sense and collision inputs. This bit should only be modified when ETHER_EN is cleared.
1 HBC	Heartbeat control. If set, the heartbeat check performs following end of transmission and the HB bit in the status register is set if the collision input does not assert within the heartbeat window. This bit should only be modified when ETHER_EN is cleared.
0 GTS	Graceful transmit stop. When this bit is set, MAC stops transmission after any frame currently transmitted is complete and GRA interrupt in the EIR register is asserted. If frame transmission is not currently underway, the GRA interrupt will be asserted immediately. Once transmission has completed, a restart can accomplish by clearing the GTS bit. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS equals 1, transmission stops after the collision. The frame is transmitted again once GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this, clear ECR[ETHER_EN] following the GRA interrupt.

29.6.4.11 Physical Address Low Register (PALR)

The PALR is written by the user. This register contains the lower 32 bits (bytes 0,1,2,3) of the 48-bit address used in the address recognition process to compare with the DA (Destination Address) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the 6-byte Source Address field when transmitting PAUSE frames. This register is not reset and must be initialized by the user.

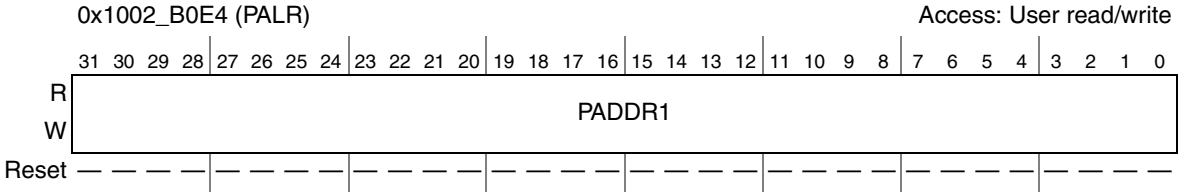


Figure 29-14. PALR Register

Table 29-22. PALR Field Descriptions

Field	Description
31–0 PADDR1	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

29.6.4.12 Physical Address High Register (PAUR)

The PAUR is written by the user. This register contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the DA (Destination Address) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the 6-byte Source Address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (0x8808) used for transmission of PAUSE frames. This register is not reset and bits 31:16 must be initialized by the user.

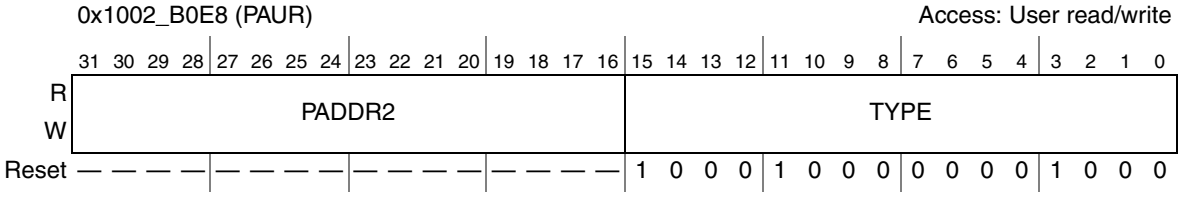


Figure 29-15. PAUR Register

Table 29-23. PAUR Field Descriptions

Field	Description
31–16 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
15–0 TYPE	Type field in PAUSE frames. These 16-bits are a constant value of 0x8808.

29.6.4.13 Opcode/Pause Duration Register (OPD)

The OPD register is read/write accessible. This register contains the 16-bit Opcode, and 16-bit pause duration fields used in transmission of a PAUSE frame. The Opcode field is a constant value, 0x0001. When another node detects a PAUSE frame, that node will pause transmission for the duration specified in the pause duration field. This register is not reset and must be initialized by the user.

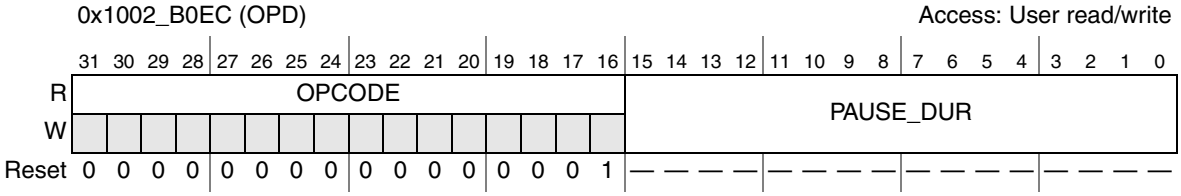


Figure 29-16. OPD Register

Table 29-24. OPD Field Descriptions

Field	Description
31–16 OPCODE	Opcode field used in PAUSE frames. These bits are a constant, 0x0001.
15–0 PAUSE_DUR	Pause Duration field used in PAUSE frames.

29.6.4.14 Descriptor Individual Upper Address Register (IAUR)

The IAUR is written by the user. This register contains the upper 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match with the DA field of receive frames with an individual DA. This register is not reset and must be initialized by the user.

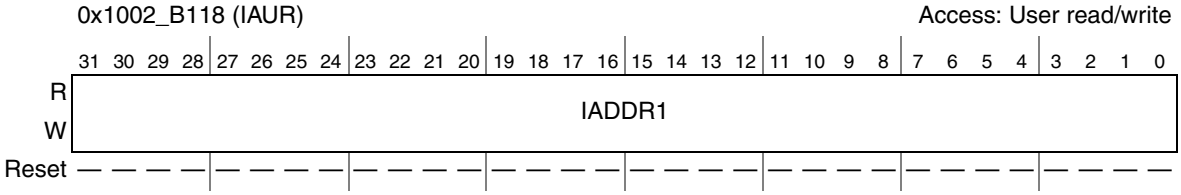


Figure 29-17. IAUR Register

Table 29-25. IAUR Field Descriptions

Field	Description
31–0 IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

29.6.4.15 Descriptor Individual Lower Address Register (IALR)

The IALR is written by the user. This register contains the lower 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match with the DA field of receive frames with an individual DA. This register is not reset and must be initialized by the user.

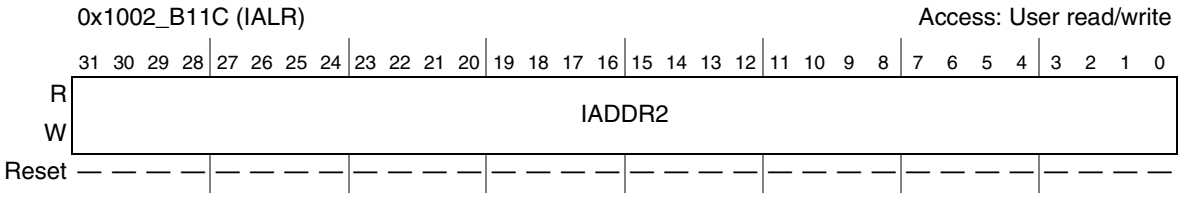


Figure 29-18. IALR Register

Table 29-26. IALR Field Descriptions

Field	Description
31–0 IADDR2	The lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

29.6.4.16 Descriptor Group Upper Address Register (GAUR)

The GAUR is written by the user. This register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register must be initialized by the user.

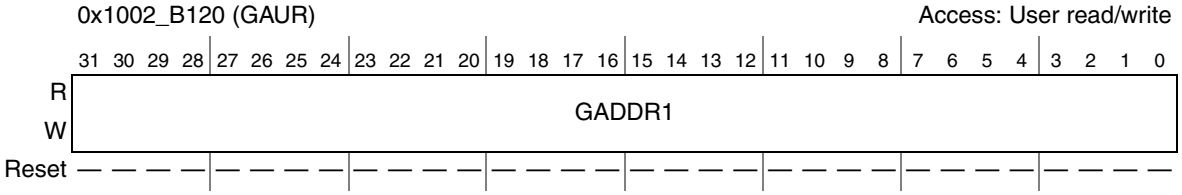


Figure 29-19. GAUR Register

Table 29-27. GAUR Field Descriptions

Field	Description
31–0 GADDR1	The GADDR1 register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

29.6.4.17 Descriptor Group Lower Address Register (GALR)

The GALR is written by the user. This register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register must be initialized by the user.

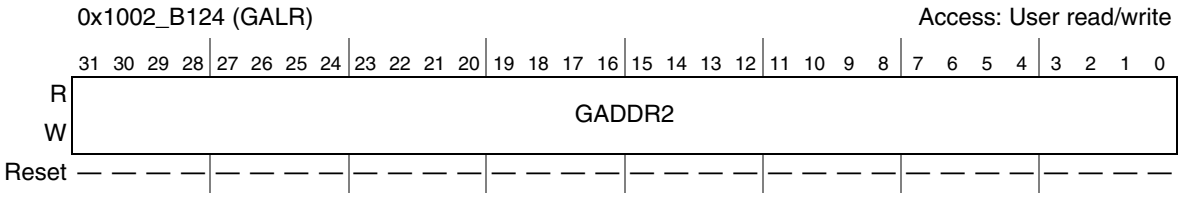


Figure 29-20. GALR Register

Table 29-28. GALR Field Descriptions

Field	Description
31–0 GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

29.6.4.18 Transmit FIFO Watermark Register (TFWR)

The TFWR is a 2-bit read/write register programmed by the user to control the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows the user to minimize transmit latency (TFWR = 0x) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value will minimize the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement (worst case bus access latency by the transmit data DMA channel).

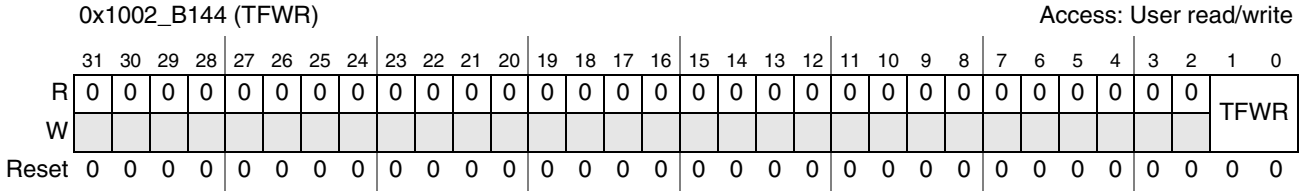


Figure 29-21. TFWR Register

Table 29-29. TFWR Field Descriptions

Field	Description
31–2	Reserved, must be cleared.
1–0 TFWR	Number of bytes written to transmit FIFO before transmission of a frame begins 00 64 bytes written 01 64 bytes written 10 128 bytes written 11 192 bytes written

29.6.4.19 FIFO Receive Bound Register (FRBR)

The FRBR is an 8-bit register that the user can read to determine the upper address bound of the FIFO RAM. Drivers can use this value, along with the FRSR to appropriately divide the available FIFO RAM between the transmit and receive data paths.

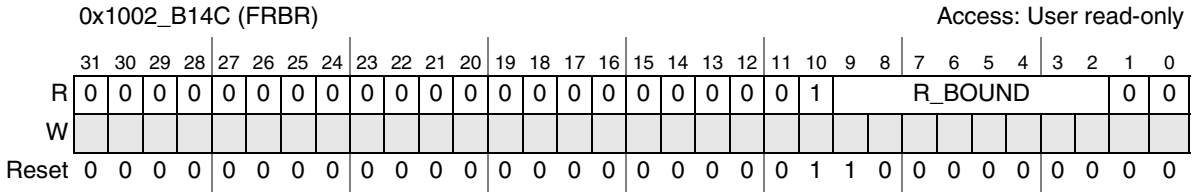


Figure 29-22. FRBR Register

Table 29-30. FRBR Field Descriptions

Field	Description
31–10	Reserved, read as 0 (except bit 10, which is read as 1).
9–2 R_BOUND	Read-only. Highest valid FIFO RAM address.
1–0	Reserved, read as 0.

29.6.4.20 FIFO Receive Start Register (FRSR)

The FRSR is an 8-bit register programmed by the user to indicate the starting address of the receive FIFO. FRSR marks the boundary between the transmit and receive FIFOs. The transmit FIFO uses addresses from the start of the FIFO to the location four bytes before the address programmed into the FRSR. The receive FIFO uses addresses from FRSR to FRBR inclusive.

The FRSR is initialized by hardware at reset. FRSR only needs to be written to change the default value.

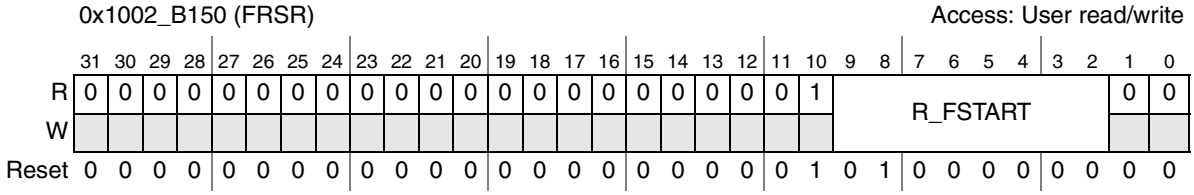


Figure 29-23. FRSR Register

Table 29-31. FRSR Field Descriptions

Field	Description
31–11	Reserved, must be cleared.
10	Reserved, must be set.
9–2 R_FSTART	Address of first receive FIFO location. Acts as delimiter between receive and transmit FIFOs. For proper operation, ensure that R_FSTART is set to 0x48 or greater.
1–0	Reserved, must be cleared.

29.6.4.21 Receive Buffer Descriptor Ring Start Register (ERDSR)

The ERDSR is written by the user. It provides a pointer to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16).

This register is not reset and must be initialized by the user prior to operation.

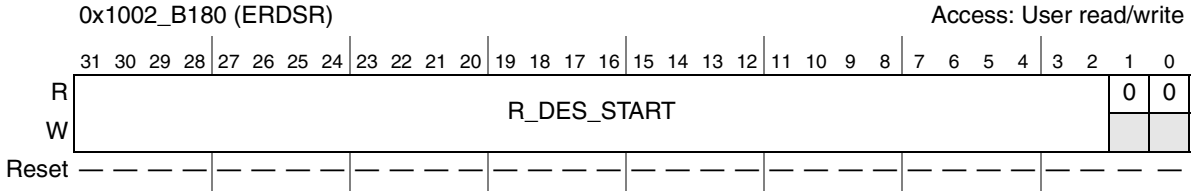


Figure 29-24. ERDSR Register

Table 29-32. ERDSR Field Descriptions

Field	Description
31–2 R_DES_START	Pointer to start of receive buffer descriptor queue.
1–0	Reserved, must be cleared.

29.6.4.22 Transmit Buffer Descriptor Ring Start Register (ETDSR)

The ETDSR is written by the user. It provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). Bits 1 and 0 should be written to 0 by the user. Non-zero values in these two bit positions are ignored by the hardware.

This register is not reset and must be initialized by the user prior to operation.

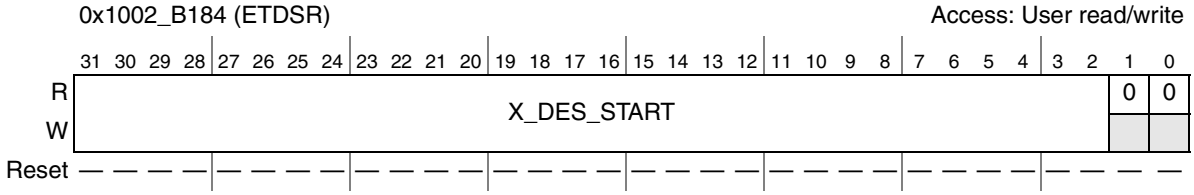


Figure 29-25. Transmit Buffer Descriptor Ring Start Register (ETDSR)

Field	Description
31–2 X_DES_START	Pointer to start of transmit buffer descriptor queue.
1–0	Reserved, must be cleared.

29.6.4.23 Receive Buffer Size Register (EMRBR)

The EMRBR is a 9-bit register programmed by the user. The EMRBR dictates the maximum size of all receive buffers. Note that because receive frames will be truncated at 2k-1 bytes, only bits 10–4 are used. This value should take into consideration that the receive CRC is always written into the last receive buffer. To allow one maximum size frame per buffer, EMRBR must be set to RCR[MAX_FL] or larger. The EMRBR must be evenly divisible by 16. To insure this, bits 3-0 are forced low. To minimize bus utilization (descriptor fetches) it is recommended that EMRBR be greater than or equal to 256 bytes.

The EMRBR does not reset, and must be initialized by the user.

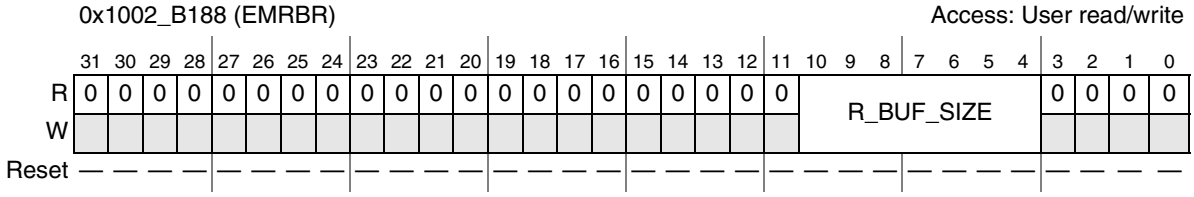


Figure 29-26. EMRBR Register

Table 29-33. EMRBR Field Descriptions

Field	Description
31–11	Reserved, must be cleared.
10–4 R_BUF_SIZE	Receive buffer size in bytes. 0x00 0 bytes 0x01 16 bytes 0x02 32 bytes ... 0x7F 2032 bytes
3–0	Reserved, must be cleared.

29.6.5 Buffer Descriptors

This section provides a description of the operation of the driver/DMA via the buffer descriptors. It is followed by a detailed description of the receive and transmit descriptor fields.

29.6.5.1 Driver/DMA Operation with Buffer Descriptors

The data for the FEC frames must reside in memory external to the FEC. The data for a frame is placed in one or more buffers. Associated with each buffer is a buffer descriptor (BD) which contains a starting address (pointer), data length, and status/control information (which contains the current state for the buffer). To permit maximum user flexibility, the BDs are also located in external memory and are read in by the FEC DMA engine.

Software “produces” buffers by allocating/initializing memory and initializing buffer descriptors. Setting the RxBD[E] or TxBD[R] bit “produces” the buffer. Software writing to either the TDAR or RDAR tells the FEC that a buffer has been placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and “consumes” the buffers after they have been produced. After the data DMA is complete and the buffer descriptor status bits have been written by the DMA engine, the RxBD[E] or TxBD[R] bit will be cleared by hardware to signal the buffer has been “consumed.” Software may poll the BDs to detect when the buffers have been consumed or may rely on the buffer/frame interrupts. These buffers may then be processed by the driver and returned to the free list.

The ECR[ETHER_EN] signal operates as a reset to the BD/DMA logic. When ECR[ETHER_EN] is deasserted the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive buffer descriptor must be initialized by software before the ECR[ETHER_EN] bit is set.

The buffer descriptors operate as two separate rings. ERDSR defines the starting address for receive BDs and ETDSR defines the starting address for transmit BDs. The last buffer descriptor in each ring is defined by the Wrap (W) bit. When set, W indicates that the next descriptor in the ring is at the location pointed to by ERDSR and ETDSR for the receive and transmit rings, respectively.

NOTE

Buffer descriptor rings must start on a 128-bit boundary.

29.6.5.1.1 Driver/DMA Operation with Transmit BDs

Typically a transmit frame will be divided between multiple buffers. An example is to have an application payload in one buffer, TCP header in a 2nd buffer, IP header in a 3rd buffer, Ethernet/IEEE 802.3 header in a 4th buffer. The Ethernet MAC does not prepend the Ethernet header (Destination Address, Source Address, Length/Type field(s)), so this must be provided by the driver in one of the transmit buffers. The Ethernet MAC can append the Ethernet CRC to the frame. Whether the CRC is appended by the MAC or by the driver is determined by the TC bit in the transmit BD which must be set by the driver.

The driver (TxBD software producer) should set up Tx BDs in such a way that a complete transmit frame is given to the hardware at once. If a transmit frame consists of three buffers, the BDs should be initialized with pointer, length and control (W, L, TC, ABC) and then the TxBD[R] bits should be set = 1 in reverse order (3rd, 2nd, 1st BD) to insure that the complete frame is ready in memory before the DMA begins. If the TxBDs are set up in order, the DMA Controller could DMA the first BD before the 2nd was made available, potentially causing a transmit FIFO underrun.

In the FEC, the DMA is notified by the driver that new transmit frame(s) are available by writing to the TDAR register. When this register is written to (data value is not significant) the FEC RISC will tell the DMA to read the next transmit BD in the ring. Once started, the RISC + DMA will continue to read and interpret transmit BDs in order and DMA the associated buffers, until a transmit BD is encountered with the R bit = 0. At this point the FEC will poll this BD one more time. If the R bit = 0 the second time, then the RISC will stop the transmit descriptor read process until software sets up another transmit frame and writes to TDAR.

When the DMA of each transmit buffer is complete, the DMA writes back to the BD to clear the R bit, indicating that the hardware consumer is finished with the buffer.

29.6.5.1.2 Driver/DMA Operation with Receive BDs

Unlike transmit, the length of the receive frame is unknown by the driver ahead of time. Therefore the driver must set a variable to define the length of all receive buffers. In the FEC, this variable is written to the EMRBR register.

The driver (RxBD software producer) should set up some number of “empty” buffers for the Ethernet by initializing the address field and the E and W bits of the associated receive BDs. The hardware (receive DMA) will consume these buffers by filling them with data as frames are received and clearing the E bit and writing to the L (1 indicates last buffer in frame) bit, the frame status bits (if L = 1) and the length field.

If a receive frame spans multiple receive buffers, the L bit is only set for the last buffer in the frame. For non-last buffers, the length field in the receive BD will be written by the DMA (at the same time the E bit is cleared) with the default receive buffer length value. For end of frame buffers the receive BD will be written with L = 1 and information written to the status bits (M, BC, MC, LG, NO, CR, OV, TR). Some of the status bits are error indicators which, if set, indicate the receive frame should be discarded and not given to higher layers. The frame status/length information is written into the receive FIFO following the end of the frame (as a single 32-bit word) by the receive logic. The length field for the end of frame buffer will be written with the length of the entire frame, not just the length of the last buffer.

For simplicity the driver may assign the default receive buffer length to be large enough to contain an entire frame, keeping in mind that a malfunction on the network or out of specification implementation could

result in giant frames. Frames of 2k (2048) bytes or larger are truncated by the FEC at 2047 bytes so software is guaranteed never to see a receive frame larger than 2047 bytes.

Similar to transmit, the FEC will poll the receive descriptor ring after the driver sets up receive BDs and writes to the RDAR register. As frames are received the FEC will fill receive buffers and update the associated BDs, then read the next BD in the receive descriptor ring. If the FEC reads a receive BD and finds the E bit = 0, it will poll this BD once more. If the BD = 0 a second time the FEC will stop reading receive BDs until the driver writes to RDAR.

29.6.5.2 Ethernet Receive Buffer Descriptor (RxBD)

In the RxBD, the user initializes the E and W bits in the first longword and the pointer in second longword. When the buffer has been DMA'd, the Ethernet controller will modify the E, L, M, BC, MC, LG, NO, CR, OV, and TR bits and write the length of the used portion of the buffer in the first longword. The M, BC, MC, LG, NO, CR, OV and TR bits in the first longword of the buffer descriptor are only modified by the Ethernet controller when the L bit is set.

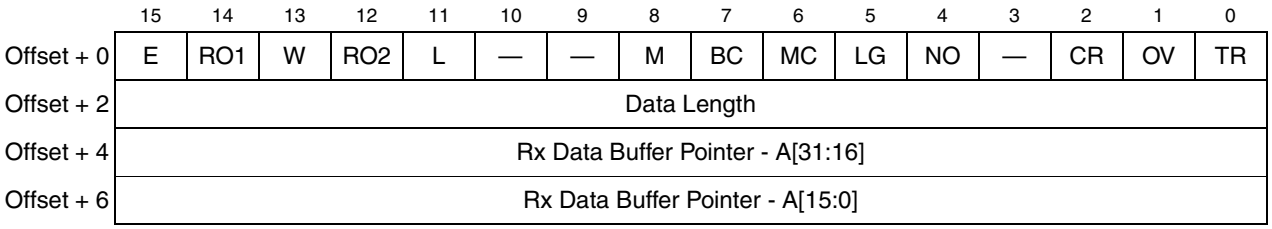


Figure 29-27. Receive Buffer Descriptor (RxBD)

Table 29-34. Receive Buffer Descriptor Field Definitions

Word	Field	Description
Offset + 0	15 E	Empty. Written by the FEC (=0) and user (=1). 0 The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	14 RO1	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	13 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ERDSR.
Offset + 0	12 RO2	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	11 L	Last in frame. Written by the FEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 0	10–9	Reserved, must be cleared.

Table 29-34. Receive Buffer Descriptor Field Definitions (continued)

Word	Field	Description
Offset + 0	8 M	Miss. Written by the FEC. This bit is set by the FEC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, the user can use the M-bit to quickly determine whether the frame was destined to this station. This bit is valid only if the L-bit is set and the PROM bit is set. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.
Offset + 0	7 BC	Set if the DA is broadcast (FF-FF-FF-FF-FF-FF).
Offset + 0	6 MC	Set if the DA is multicast and not BC.
Offset + 0	5 LG	Rx frame length violation. Written by the FEC. A frame length greater than RCR[MAX_FL] was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds 2032 bytes.
Offset + 0	4 NO	Receive non-octet aligned frame. Written by the FEC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L-bit is set. If this bit is set the CR bit will not be set.
Offset + 0	3	Reserved, must be cleared.
Offset + 0	2 CR	Receive CRC error. Written by the FEC. This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.
Offset + 0	1 OV	Overrun. Written by the FEC. A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR, and CL lose their normal meaning and are zero. This bit is valid only if the L-bit is set.
Offset + 0	0 TR	Set if the receive frame is truncated (frame length > 2032 bytes). If the TR bit is set, frame must be discarded and the other error bits must be ignored as they may be incorrect.
Offset + 2	15–0 Data Length	Data length. Written by the FEC. Data length is the number of octets written by the FEC into this BD's data buffer if L equals 0 (the value will be equal to EMRBR), or the length of the frame including CRC if L equals 1. It is written by the FEC once as the BD is closed.
0ffset + 4	15–0 A[31:16]	RX data buffer pointer, bits [31:16][1]
Offset + 6	15–0 A[15:0]	RX data buffer pointer, bits [15:0]

[1] The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 16. The buffer must reside in memory external to the FEC. The Ethernet controller never modifies this value.

NOTE

Whenever the software driver sets an E bit in one or more receive descriptors, the driver should follow that with a write to RDAR.

29.6.5.3 Ethernet Transmit Buffer Descriptor (TxBD)

Data is presented to the FEC for transmission by arranging it in buffers referenced by the channel's TxBDs. The Ethernet controller confirms transmission by clearing the ready bit (R bit) when DMA of the buffer is

complete. In the TxBD the user initializes the R, W, L, and TC bits and the length (in bytes) in the first longword, and the buffer pointer in the second longword.

The FEC will set the R bit = 0 in the first longword of the BD when the buffer has been DMA'd. Status bits for the buffer/frame are not included in the transmit buffer descriptors. Transmit frame status is indicated via individual interrupt bits (error conditions) and in statistic counters in the MIB block. See Section 29.6.3, "MIB Block Counters Memory Map" for more details.

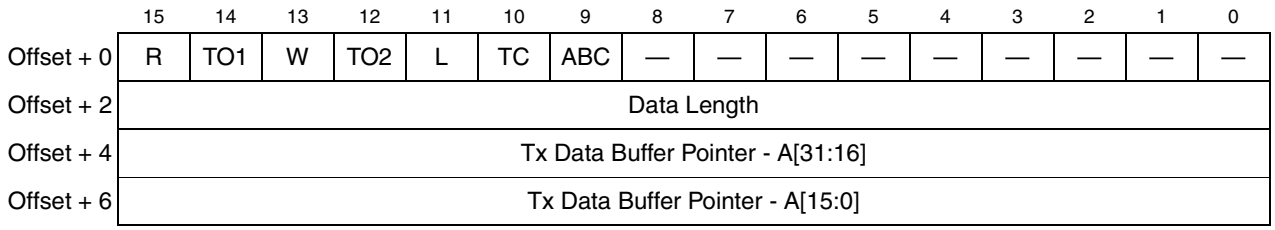


Figure 29-28. Transmit Buffer Descriptor (TxBD)

Table 29-35. Transmit Buffer Descriptor Field Definitions

Word	Field	Description
Offset + 0	15 R	Ready. Written by the FEC and the user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, prepared for transmission by the user, has not been transmitted or currently transmits. The user may write no fields of this BD once this bit is set.
Offset + 0	14 TO1	Transmit software ownership. This field is reserved for software use. This read/write bit will not be modified by hardware, nor will its value affect hardware.
Offset + 0	13 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.
Offset + 0	12 TO2	Transmit software ownership. This field is reserved for use by software. This read/write bit will not be modified by hardware, nor will its value affect hardware.
Offset + 0	11 L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame 1 The buffer is the last in the transmit frame
Offset + 0	10 TC	Transmit CRC. Written by user (only valid if L is set). 0 End transmission immediately after the last data byte 1 Transmit the CRC sequence after the last data byte
Offset + 0	9 ABC	Append bad CRC. Written by user (only valid if L is set). 0 No effect 1 Transmit the CRC sequence inverted after the last data byte (regardless of TC value)
Offset + 0	8–0	Reserved, must be cleared.
Offset + 2	15–0 Data Length	Data length, written by user. Data length is the number of octets the FEC should transmit from this BD's data buffer. It is never modified by the FEC. Bits [15:5] are used by the DMA engine; bits[4:0] are ignored.

Table 29-35. Transmit Buffer Descriptor Field Definitions (continued)

Word	Field	Description
Offset + 4	15–0 A[31:16]	Tx data buffer pointer, bits [31:16][1]
Offset + 6	15–0 A[15:0]	Tx data buffer pointer, bits [15:0]

[1] The transmit buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 4. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

NOTE

Once the software driver has set up the buffers for a frame, it should set up the corresponding BDs. The last step in setting up the BDs for a transmit frame should be to set the R bit in the first BD for the frame. The driver should follow that with a write to TDAR which will trigger the FEC to poll the next BD in the ring.

Chapter 30
High-Speed USB On-The-Go (HS USB-OTG)

The USB module contains all of the functionality required to support three independent USB ports, compatible with the USB 2.0 specification. In addition to the normal USB functionality, the module also provides support for direct connections to on-board USB peripherals, and supports multiple interface types for serial transceivers.

Figure 30-1 shows a block diagram of the USB module.

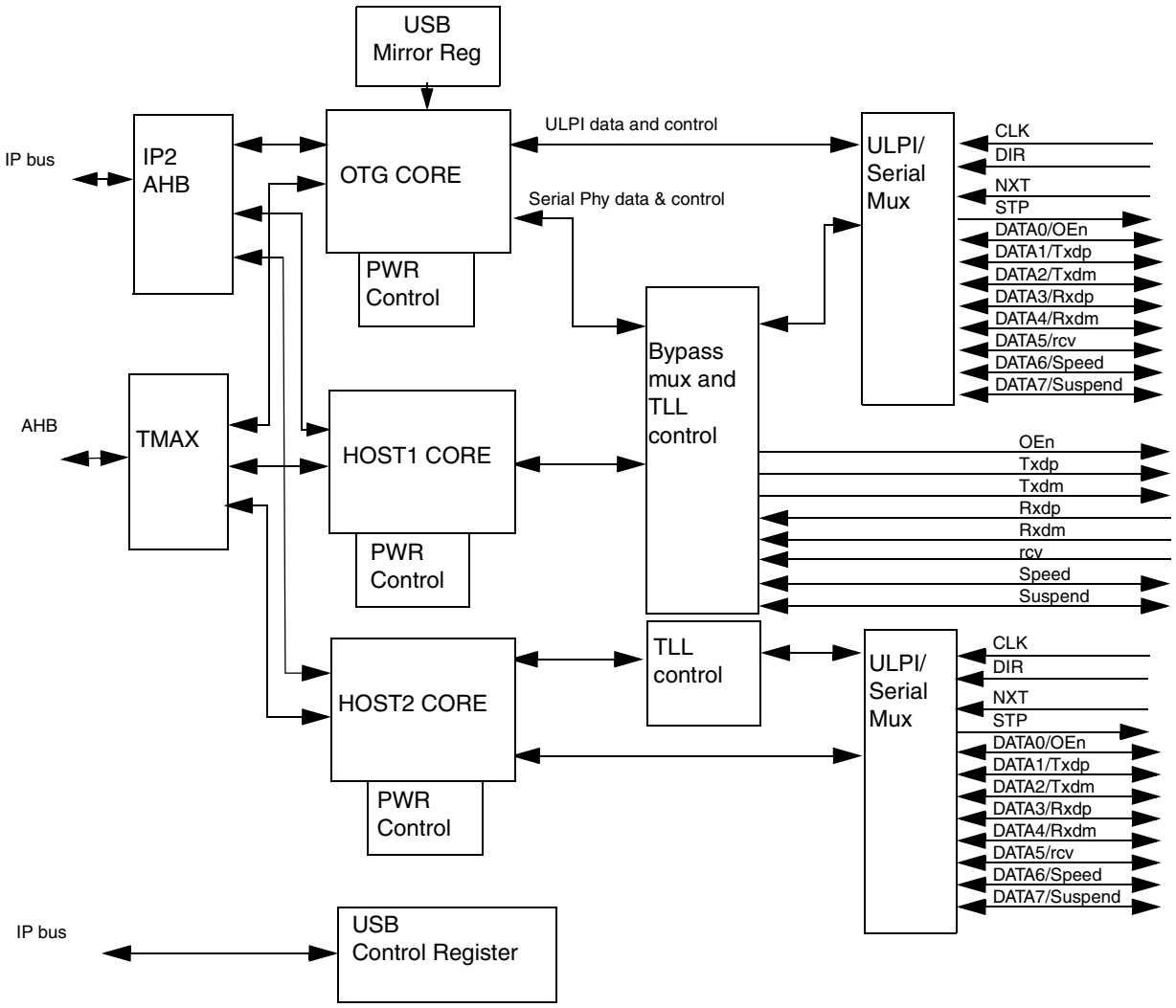


Figure 30-1. USB Block Diagram

30.1 Overview

The USB module provides high performance USB On-The-Go (OTG) functionality, compliant with the USB 2.0 specification, the OTG supplement and the ULPI 1.0 Low Pin Count specification. The module consists of 3 independent USB cores, each controlling 1 USB port.

In addition to the USB cores, the module provides for a Transceiver-less Link (TLL) operation on host ports 1 and 2 and allows for routing the OTG transceiver interface to HOST port 1 such that this transceiver can be used to communicate with a USB peripheral connected to host port 1.

30.2 Features

The USB module includes the following features:

- Full Speed/Low speed Host only core (HOST 1)
  - Transceiverless Link Logic (TLL) for on board connection to a FS/LS USB peripheral
  - Bypass mode to route Host Port 1 signals to OTG I/O port
- High Speed/Full Speed/Low Speed Host Only core (HOST2)
  - High Speed ULPI 1.0 compliant interface
  - Full Speed/Low Speed interface for Serial transceiver
  - TLL function for direct connection to USB peripheral in FS/LS (serial) operation
- High speed OTG core
  - High Speed ULPI 1.0 compliant interface
  - Software configurable for ULPI or Serial transceiver interface
  - High Speed (with ULPI transceiver), Full Speed and Low Speed operation in HOST mode
  - High Speed (with ULPI transceiver), and Full Speed operation in Peripheral mode
  - Hardware support for OTG signaling, Session Request Protocol and Host Negotiation Protocol
  - Up to 8 bidirectional endpoints
- Low power mode with local and remote wake-up capability
- Serial PHY interfaces configurable for Bidirectional/Unidirectional and Differential/Single Ended
- Embedded DMA controller

30.3 Modes of Operation

The USB module has two main modes of operation; Normal mode and Bypass mode. Furthermore, the USB interfaces can be configured for High Speed operation (480 Mbps) and/or Full/Low speed operation (12/1.5 Mbps).

This chapter details the configuration options.

30.3.1 Operational Modes

30.3.1.1 Normal Mode

In normal mode, each USB core controls its corresponding PORT. Each port can work in 1 or more modes

- Host Port 1:

  This port supports Full/Low speed and is used Serial transceiver only.

  — PHY mode:

  In this mode, an external serial transceiver is connected to the port. This is used for off-board USB connections

  — TLL mode:

  In TLL mode, internal logic is enabled to emulate the functionality of 2 back-to-back connected transceivers. This mode is typically used for on-board USB connections to USB-capable peripherals

- Host Port 2

  This port supports ULPI and Serial Transceivers.

  — Serial Interface mode

  PHY mode - for connections using transceivers

  TLL mode - for direct on-board connections to USB peripherals

  — ULPI interface

  ULPI is the low-pin count standard for connecting off-chip High-Speed USB transceivers to a USB device. When the port is configured for ULPI mode, only a ULPI compatible transceiver can be used

- OTG port:

  This Port requires a transceiver and is intended for off-board USB connections.

  — Serial Interface mode

  In serial mode, a serial OTG transceiver must be connected. The port does not support dedicated signals for OTG signaling. Instead, a transceiver with built-in OTG registers must be used. Typically, the Transceiver registers are accessible over an I2C or SPI interface

  — ULPI Mode

  It this mode, a ULPI transceiver is connected to the port pins to support High-speed off board USB connections. ULPI mode is activated by writing the relevant register

30.3.1.2 Bypass Mode

Bypass mode affects the operation of the OTG port and HOST port 1. This mode is only available when a serial transceiver is used on the OTG port, and the peripheral device on port 1 is using a TLL connection.

Bypass mode is activated by setting the bypass bit in the USBCONTROL register. In this mode, the USB OTG port connections are internally routed to the USB HOST 1 port, such that the transceiver on the OTG port connects to a peripheral USB device on HOST port 1. The OTG core and the HOST 1 core are

disconnected from their ports when bypass is active. The status of the DM and DP inputs for the cores is programmable in the USB Control Register (Table 30-2).

30.3.1.3 Low Power Mode

Each of the three USB cores has an associated power control module that is controlled by the USB core and clocked on a 32 KHz clock. When a USB bus is idle, the transceiver can be placed in low power mode (suspend), after which the clocks to the USB core can be stopped. The 32 KHz low power clock must remain active as it is needed for wakeup detection.

Either the local CPU or the remote USB Host/Peripheral can initiate a wake-up sequence to resume USB communication.

30.4 External Signal Description

30.4.1 Overview

See Table 30-3 for the list of signals entering and existing this module to peripherals within the chip.

30.4.2 Detailed Signal Descriptions

Detailed signal descriptions for each module lists in Section 30.6, “Functional Description.”

30.5 Memory Map and Register Definitions

Table 30-2 shows the USB module memory map.

Table 30-2. USB Module Memory Map

Address	Controller	Use	Access
Base + 0x000	OTG	ID (UOG_ID)	R
Base + 0x004	OTG	Hardware General (UOG_HWGENERAL)	R
Base + 0x008	OTG	Host Hardware Parameters (UOG_HWHOST)	R
Base + 0x010	OTG	TX Buffer Hardware Parameters (UOG_HWTXBUF)	R
Base +0x014	OTG	RX Buffer Hardware Parameters (UOG_HWRXBUF)	R
Base +0x080	OTG	General Purpose Timer #0 Load(GPTIMER0LD)	RW
Base +0x084	OTG	General Purpose Timer #0 Controller(GPTIMER0CTRL)	RW
Base +0x088	OTG	General Purpose Timer #1 Load(GPTIMER0LD)	RW
Base +0x08c	OTG	General Purpose Timer #1 Controller(GPTIMER0CTRL)	RW
Base +0x100	OTG	Capability Register Length (UOG_CAPLENGTH)	R
Base + 0x102	OTG	Host Interface Version (UOG_HCIVERSION)	R
Base +0x104	OTG	Host Control Structural Parameters (UOG_HCSPARAMS)	R
Base +0x108	OTG	Control Capability Parameters (UOG_HCCPARAMS)	R

Table 30-2. USB Module Memory Map (continued)

Address	Controller	Use	Access
Base + 0x120	OTG	Device Interface Version (UOG_DCIVERSION)	R
Base + 0x124	OTG	Device Controller Capability Parameters (UOG_DCCPARAMS)	R
Base +0x140	OTG	USB Command Register (UOG_USBCMD)	RW
Base +0x144	OTG	USB Status Register (UOG_USBSTS)	RW
Base +0x148	OTG	Interrupt Enable Register (UOG_USBINTR)	RW
Base + 0x14C	OTG	USB Frame Index (UOG_FRINDEX)	RW
Base +0x154	OTG	Host Controller Frame List Base Address (UOG_PERIODICLISTBASE)	RW (32-bit only)
Base +0x158	OTG	Host Controller Next Asynch. Address (UOG_ASYNCLISTADDR)	RW (32-bit only)
Base +0x160	OTG	Host Controller Embedded TT Asynch. Buffer Status (UOG_BURSTSIZE)	RW (32-bit only)
Base +0x164	OTG	TX FIFO Fill Tuning (UOG_TXFILLTUNING)	RW (32-bit only)
Base + 0x170	OTG	ULPI Viewport (ULPIVIEW)	RW
Base +0x180	OTG	Config Flag (UOG_CFGFLAG)	R
Base +0x184	OTG	Port Status and Control (UOG_PORTSC1)	RW
Base + 0x1A4	OTG	On-The-Go Status and control (UOG_OTGSC	RW
Base +0x1A8	OTG	USB Device Mode (UOG_USBMODE)	RW
Base + 0x1AC	OTG	Endpoint Setup Status (UOG_ENDPTSETUPSTAT)	RW
Base + 0x1B0	OTG	Endpoint Initialization (UOG_ENDPTPRIME)	RW
Base + 0x1B4	OTG	Endpoint De-Initialize (UOG_ENDPTFLUSH)	RW
Base + 0x1B8	OTG	Endpoint Status (UOG_ENDPTSTAT)	R
Base + 0x1BC	OTG	Endpoint Complete (UOG_ENDPTCOMPLETE)(	RW
Base + 0x1C0	OTG	Endpoint Control0 (ENDPTCTRL0)	RW
Base + 0x1C4	OTG	Endpoint Control1 (ENDPTCTRL1)	RW
Base + 0x1C8	OTG	Endpoint Control2 (ENDPTCTRL2)	RW
Base + 0x1CC	OTG	Endpoint Control3 (ENDPTCTRL3)	RW
Base + 0x1D0	OTG	Endpoint Control4 (ENDPTCTRL4)	RW
Base + 0x1D4	OTG	Endpoint Control5 (ENDPTCTRL5)	RW
Base + 0x1D8	OTG	Endpoint Control6 (ENDPTCTRL6)	RW
Base + 0x1DC	OTG	Endpoint Control07(ENDPTCTRL7)	RW
Base + 0x200	Host1	Host 1 ID (UH1_ID)	R
Base + 0x204	Host1	Hardware General (UH1_HWGENERAL)	R
Base + 0x208	Host1	Host Hardware Parameters (UH1_HWHOST)	R
Base + 0x210	Host1	TX Buffer Hardware Parameters (UH1_HWTXBUF)	R
Base +0x214	Host1	RX Buffer Hardware Parameters (UH1_HWRXBUF)	R

Table 30-2. USB Module Memory Map (continued)

Address	Controller	Use	Access
Base +0x280	Host1	General Purpose Timer #0 Load(GPTIMER0LD)	RW
Base +0x284	Host1	General Purpose Timer #0 Controller(GPTIMER0CTRL)	RW
Base +0x288	Host1	General Purpose Timer #1 Load(GPTIMER0LD)	RW
Base +0x28c	Host1	General Purpose Timer #1 Controller(GPTIMER0CTRL)	RW
Base +0x300	Host1	Capability Register Length (UH1_CAPLENGTH)	R
Base + 0x302	Host1	Host Interface Version (UH1_HCIVERSION)	R
Base +0x304	Host1	Host Control Structural Parameters (UH1_HCSPARAMS)	R
Base +0x308	Host1	Control Capability Parameters (UH1_HCCPARAMS)	R
Base +0x340	Host1	USB Command Register (UH1_USBCMD)	RW
Base +0x344	Host1	USB Status Register (UH1_USBSTS)	RW
Base +0x348	Host1	Interrupt Enable Register (UH1_USBINTR)	RW
Base + 0x34C	Host1	USB Frame Index (UH1_FRINDEX)	RW
Base +0x354	Host1	Host Controller Frame List Base Address (UH1_PERIODICLISTBASE)	RW (32-bit only)
Base +0x358	Host1	Host Controller Next Asynch. Address (UH1_ASYNCLISTADDR)	RW (32-bit only)
Base +0x360	Host1	Host Controller Embedded TT Asynch. Buffer Status (UH1_BURSTSIZE)	RW (32-bit only)
Base +0x364	Host1	TX FIFO Fill Tuning (UH1_TXFILLTUNING)	RW (32-bit only)
Base +0x380	Host1	Reserved	R
Base +0x384	Host1	Port Status and Control (UH1_PORTSC1)	RW
Base +0x3A8	Host1	USB Device Mode (UH1_USBMODE)	RW
Base + 0x400	Host2	ID (UH2_ID)	R
Base + 0x404	Host2	Hardware General (UH2_HWGENERAL)	R
Base + 0x408	Host2	Host Hardware Parameters (UH2_HWHOST)	R
Base + 0x410	Host2	TX Buffer Hardware Parameters (UH2_HWTXBUF)	R
Base +0x414	Host2	RX Buffer Hardware Parameters (UH2_HWRXBUF)	R
Base +0x480	Host2	General Purpose Timer #0 Load(GPTIMER0LD)	RW
Base +0x484	Host2	General Purpose Timer #0 Controller(GPTIMER0CTRL)	RW
Base +0x488	Host2	General Purpose Timer #1 Load(GPTIMER0LD)	RW
Base +0x48c	Host2	General Purpose Timer #1 Controller(GPTIMER0CTRL)	RW
Base +0x500	Host2	Capability Register Length (UH2_CAPLENGTH)	R
Base + 0x502	Host2	Host Interface Version (UH2_HCIVERSION)	R
Base +0x504	Host2	Host Control Structural Parameters (UH2_HCSPARAMS)	R
Base +0x508	Host2	Control Capability Parameters (UH2_HCCPARAMS)	R
Base +0x540	Host2	USB Command Register (UH2_USBCMD)	RW

Table 30-2. USB Module Memory Map (continued)

Address	Controller	Use	Access
Base +0x544	Host2	USB Status Register (UH2_USBSTS)	RW
Base +0x548	Host2	Interrupt Enable Register (UH2_USBINTR)	RW
Base + 0x54C	Host2	USB Frame Index (UH2_FRINDEX)	RW
Base +0x554	Host2	Host Controller Frame List Base Address (UH2_PERIODICLISTBASE)	RW (32-bit only)
Base +0x558	Host2	Host Controller Next Asynch. Address (UH2_ASYNCLISTADDR)	RW (32-bit only)
Base +0x560	Host2	Host Controller Embedded TT Asynch. Buffer Status (UH2_BURSTSIZE)	RW (32-bit only)
Base +0x564	Host2	TX FIFO Fill Tuning (UH2_TXFILLTUNING)	RW (32-bit only)
Base + 0x570	Host2	ULPI Viewport (ULPIVIEW)	RW
Base +0x580	Host2	Reserved	R
Base +0x584	Host2	Port Status and Control (UH2_PORTSC1)	RW
Base +0x5A8	Host2	USB Device Mode (UH2_USBMODE)	RW
Base + 0x600		USB Control Register (USB_CTRL)	RW
Base + 0x604		USB OTG Mirror Register (USB_OTG_MIRROR)	RW

30.5.1 Register Descriptions

The following sections describe the registers used to control the USB module.

30.5.1.1 USBCONTROL—USB Control Register (USB_CTRL)

The USB control register controls the integration specific features of the USB module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

Base + 0x600

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OWIR	OSIC		OUIE	OWIE	OBPVAL		OPM	H2WIR	H2SIC		H2UIE	H2WIE	0	0	H2PM
W																
Reset:	—	P	P	0	0	0	0	0	—	P	P	0	0	—	—	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	H1WIR	H1SIC		0	H1WIE	H1BPVAL		H1PM	0	0	H2DT	H1DT	0	0	0	BPE
W																
Reset:	—	P	P	—	0	0	0	0	—	—	0	0	—	0	0	0

P = Preset value taken from Preset input signals

Figure 30-2. USB Control Register

Table 30-3. USB Control Register Field Descriptions

Field	Description
31 OWIR	OTG Wake-up Interrupt Request. This bit indicates that a wake-up interrupt request is received on the OTG port. This bit is cleared by disabling the wake-up interrupt. 0 No wake-up detected 1 Wake-up Interrupt Request received
30–29 OSIC	OTG Serial Interface Configuration. Controls the interface type of the OTG port when used with a serial transceiver. This bit field allows for configuring the serial interface for Single Ended or Differential operation combined with Bidirectional or Unidirectional operation. The reset value of OSIC depends on the state of the signals "__ADD__" and "__ADD__" during reset. 00 Differential/Unidirectional (6-wire) 01 Differential/Bidirectional (4-wire) 10 Single Ended/Unidirectional (6-wire) 11 Single Ended/Bidirectional (3-wire)
28 OUIE	OTG ULPI interrupt enable. Controls whether or not interrupts from the ULPI transceiver will trigger the wake-up logic. This bit is only meaningful when a ULPI transceiver is selected. 0 ULPI transceiver interrupts are ignored by the wakeup logic. 1 ULPI transceiver interrupts activate the wake-up logic
27 OWIE	OTG Wake-up Interrupt Enable. This bit enables or disables the OTG wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 0 Interrupt Disabled 1 Interrupt Enabled
26–25 OBPVAL	OTG Bypass Value. This field contains the status of the RxDp and RxDm inputs to the OTG core when Bypass mode is enabled. Bit 26 controls RxDp, bit 25 controls RxDm.
24 OPM	OTG Power Mask. The power mask bit controls whether or not the external Vbus Power and OverCurrent detection are active for the OTG port. 0 The USBPWR pin will assert with the OTG core's Vbus power Enable and the assertion of the OC input will be reported to the OTG core. 1 The USBPWR and OC pins are not used by the OTG core.
23 H2WIR	Host 2 Wake-up Interrupt Request. Indicates a pending Wake-up request on Host port 2. This bit is cleared by disabling the interrupt. The interrupt must be disabled for at least 2 clock cycles of the standby clock. 0 No Wake-up interrupt received 1 Wake-up interrupt received
22–21 H2SIC	Host 2 Serial Interface Configuration. Controls the interface type of the Host 2 port when used with a serial transceiver. This bit field allows for configuring the serial interface for Single Ended or Differential operation combined with Bidirectional or Unidirectional operation. 00 Differential/Unidirectional (6-wire) 01 Differential/Bidirectional (4-wire) 10 Single Ended/Unidirectional (6-wire) 11 Single Ended/Bidirectional (3-wire)
20 H2UIE	Host 2 ULPI interrupt enable. Controls whether or not interrupts from the ULPI transceiver will trigger the wake-up logic. This bit is only meaningful when a ULPI transceiver is selected. 0 ULPI transceiver interrupts are ignored by the wakeup logic. 1 ULPI transceiver interrupts activate the wake-up logic

Table 30-3. USB Control Register Field Descriptions (continued)

Field	Description
19 H2WIE	Host 2 Wake-up Interrupt Enable. This bit enables or disables the Host 2 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode. 0 Interrupt Disabled 1 Interrupt Enabled
18–17	Reserved. These bits are reserved and should read 0.
16 H2PM	Host 2 Power Mask. The power mask bit controls whether or not the external Vbus Power and OverCurrent detection are active for the Host 2 port. 0 The USBPWR pin will assert with the Host 2 core's Vbus power Enable and the assertion of the OC input will be reported to the Host 2 core. 1 The USBPWR and OC pins are not used by the Host 2 core.
15 H1WIR	Host 1 Wake-up Interrupt Request. Indicates a pending Wake-up request on Host port 1. This bit is cleared by disabling the interrupt. The interrupt must be disabled for at least 2 clock cycles of the standby clock. 0 Wake-up interrupt received 1 No Wake-up interrupt received
14–13 H1SIC	Host 1 Serial Interface Configuration. Controls the interface type of the Host 1 port when used with a serial transceiver. This bit field allows for configuring the serial interface for Single Ended or Differential operation combined with Bidirectional or Unidirectional operation. 00 Differential/Unidirectional (6-wire) 01 Differential/Bidirectional (4-wire) 10 Single Ended/Unidirectional (6-wire) 11 Single Ended/Bidirectional (3-wire)
12	Reserved. This bit is reserved and should read 0.
11 H1WIE	Host 1 Wake-up Interrupt Enable. This bit enables or disables the Host 1 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 0 Interrupt Disabled 1 Interrupt Enabled
10–9 H1BPVAL	HOST 1 Bypass Value. This field contains the status of the RxDp and RxDm inputs to the HOST1 core when Bypass mode is enabled. Bit 10 controls RxDp, bit 9 controls RxDm.
8 H1PM	Host 1 Power Mask. The power mask bit controls whether or not the external Vbus Power and OverCurrent detection are active for the Host 1 port. 0 The USBPWR pin will assert with the Host 1core's Vbus power Enable and the assertion of the OC input will be reported to the Host 1 core. 1 The USBPWR and OC pins are not used by the Host 1 core.
7–6	Reserved. These bits are reserved and should read 0.
5 H2DT	Host 2 TLL Disable. This bit controls whether or not the Transceiver-less Link Logic is enabled for the serial interface of Host Port 2. 0 TLL is enabled 1 TLL is disabled
4 H1DT	Host 1 TLL disable. This bit controls the TLL logic for Host Port 1 0 TLL is enabled 1 TLL is disabled

Table 30-3. USB Control Register Field Descriptions (continued)

Field	Description
3–1	Reserved. These bits are reserved and should read 0.
0 BPE	Bypass Enable. This bit enables/disables the USB Bypass function. 0 Bypass inactive—Normal mode operation. 1 Bypass active—USB signals from Host port 1 are routed to the OTG port.

30.5.1.2 OTGMIRROR — OTG port Mirror Register

The OTG port is designed for operation with an external OTG transceiver. When a ULPI transceiver is in use, all OTG signaling is communicated over the ULPI data bus as described in the ULPI specification. However, when a serial transceiver is used, the interface for OTG signaling is not standardized. Most OTG transceivers use a serial interface like I2C or SPI to transfer the OTG signaling back to the CPU and/or USB core. In this case, the USB CORE has no direct connection the OTG signals in the transceiver.

The OTGMIRROR register provides a soft interface between the OTG signals in the transceiver and the OTG signal inputs to the USB core. The USB driver software is responsible for reading the OTG status registers in the transceiver over the serial interface and set the bits accordingly in the OTGMIRROR register (see Figure 30-3), such that the USB controller knows the status of the transceiver.

The USB driver should be designed such that the latency requirements as defined in the USB 2.0 OTG supplement specification are met.

Base + 0x604

	7	6	5	4	3	2	1	0
R	0	0	0	SESEND	VBUSVAL	BSESVLD	ASESVLD	IDDIG
W								
Reset:	—	—	—	0	0	0	0	0

= Unimplemented or Reserved

Figure 30-3. OTG Mirror Register (OTGMIRROR)

SESEND—B device Session End

This bit is set by the USB driver when the PHY reports a Session End condition.

1 = Session End (0.2V < Vbus < 0.8V)

0 = Session Active

VBUSVLD—Vbus Valid

The USB driver sets this bit when the transceiver reports Vbus Valid.

1 = Vbus is Valid (Vbus > 4.4V)

0 = Vbus Invalid (Vbus < 4.4V)

BSESVLD—B Session Valid

B session valid should be set when the transceiver reports B-session Valid.

1 = B Session is Valid (0.8V < Vbus < 4.0V)

0 = B Session is not valid (Vbus < 0.8V)

ASESVLD—A session Valid

This bit must be set when a valid ‘A-Session’ level is detected on Vbus.
1 = A Session is Valid (0.8V < Vbus < 2.0V)
0 = Session is not valid for A-device.

IDIDG—OTG ID-pin Status

This bit indicates to the USB core whether it should operate as A-device or as B-device
1 = ID pin is High—Operate as B-device
0 = ID pin is Low—Operate as A-device

30.5.1.3 USB Core Register

For detailed registers description of OTG Controller, Host 1 Controller, and Host 2 Controller, refer to Section 30.5.1, “Register Descriptions.” The value of registers that depended on implementation can be found based on the core configuration parameter file (OTG Controller refer to Section 30.6.3, “USB OTG Controller,” Section 30.6.1, “USB HOST Controller 1,” and Section 30.6.2, “USB Host Controller 2.”

30.6 Functional Description

This sections describes the functionality and the topology of the different building blocks of the USB module.

30.6.1 USB HOST Controller 1

30.6.1.1 Host Controller 1 to Host Port 1 Interface

The Host 1 core USB signals do not connect directly to the HOST1 I/O pins. Instead, the signals pass through the Bypass Mux to allow for additional functionality on Host Port1. See section Section 30.6.6, “USB Bypass Mode” for details.

In addition to Bypass Muxing, this mux also provides interface type conversion for Host core 1. This type conversion is independent of the mux state. Table 30-3 details the available type settings.

Depending on the selected interface type, the USB port signals have the functionality as described in table Table 30-4.

Table 30-4. Host Port 1 Pin Functions

	Single-Ended Mode			Differential Mode		
Signal	**Unidir**	**Bidir**		**Unidir**	**Bidir**	
OEn	OEn	OEn = 0	OEn = 1	OEn	OEn = 0	OEn = 1
TxDp	DAT			TxDp		
TxDm	SE0			TxDm		
RxDp	RxDp	DATO	DATI	RxDp	TxDp	RxDp

Table 30-4. Host Port 1 Pin Functions (continued)

	Single-Ended Mode			Differential Mode		
Signal	**Unidir**	**Bidir**		**Unidir**	**Bidir**	
RxDm	RxDm	SE0O	SE0I	RxDm	TxDm	RxDm
RCV	Rcv	—	—	Rcv	—	Rcv

NOTE

If the RCV signal is not available on the external part (for example, when using a direct connection with TLL mode), the RCV input must be tied to the RxVp input.

30.6.2 USB Host Controller 2

Host controller 2 is configured for operation with a FS/LS serial transceiver or a parallel ULPI transceiver (HS/FS/LS). The selection between transceiver type is software programmable. The module defaults to serial mode.

30.6.2.1 Host Port 2 Signal Connections and Signal Muxing

Table 30-5 details the I/O pad connections for the USB Host Port 2. The direction control is logic ‘1’ for output.

Table 30-5. Host Port 2 Signal Connections

I/O PAD	Type	Input	Output	Direction control
USB2_ULPI_CLK	I/O	ipp_ind_uh2_clk		
USB2_ULPI_DIR	IN	ipp_ind_uh2_dir		
USB2_ULPI_STP	OUT		ipp_do_uh2_stp	
USB2_ULPI_NXT	IN	ipp_ind_uh2_nxt		
USB2_ULPI_DATA0	I/O	ipp_ind_uh2_data0	ipp_do_uh2_data0	ipp_obe_uh2_data0
USB2_ULPI_DATA1	I/O	ipp_ind_uh2_data1	ipp_do_uh2_data1	ipp_obe_uh2_data1
USB2_ULPI_DATA2	I/O	ipp_ind_uh2_data2	ipp_do_uh2_data2	ipp_obe_uh2_data2
USB2_ULPI_DATA3	I/O	ipp_ind_uh2_data3	ipp_do_uh2_data3	ipp_obe_uh2_data3
USB2_ULPI_DATA4	I/O	ipp_ind_uh2_data4	ipp_do_uh2_data4	ipp_obe_uh2_data4
USB2_ULPI_DATA5	I/O	ipp_ind_uh2_data5	ipp_do_uh2_data5	ipp_obe_uh2_data5
USB2_ULPI_DATA6	I/O	ipp_ind_uh2_data6	ipp_do_uh2_data6	ipp_obe_uh2_data6
USB2_ULPI_DATA7	I/O	ipp_ind_uh2_data7	ipp_do_uh2_data7	ipp_obe_uh2_data7

Host port 2 can support either a ULPI transceiver or a Serial Transceiver. Depending on the selected type, signaling for one or the other is available at the top level. Table 30-6 gives the relation between the top level signal and the USB core signal in both modes.

Table 30-6. ULPI/ Serial Muxing

USB TOP Input signals			USB TOP Output Signals		
From I/O Mux	**Serial Mode**	**ULPI Mode**	**To I/O Mux**	**Serial Mode**	**ULPI Mode**
ipp_ind_uh2_data0	ipp_ind_uh2_oe_n	uuh2_data_in[0]	ipp_do_uh2_data0	ipp_do_uh2_oe_n	uuh2_data_out[0]
ipp_ind_uh2_data1		uuh2_data_in[1]	ipp_do_uh2_data1	ipp_do_uh2_txdp_dat	uuh2_data_out[1]
ipp_ind_uh2_data2		uuh2_data_in[2]	ipp_do_uh2_data2	ipp_do_uh2_txdm_se0	uuh2_data_out[2]
ipp_ind_uh2_data3	ipp_ind_uh2_rxvp_txdp_dat	uuh2_data_in[3]	ipp_do_uh2_data3	ipp_do_uh2_rxvp_txdp_dat	uuh2_data_out[3]
ipp_ind_uh2_data4	ipp_ind_uh2_rxvm_txdm_se0	uuh2_data_in[4]	ipp_do_uh2_data4	ipp_do_uh2_rxvm_txdm_se0	uuh2_data_out[4]
ipp_ind_uh2_data5	ipp_ind_uh2_xcvr_ser_rcv	uuh2_data_in[5]	ipp_do_uh2_data5	ipp_do_uh2_xcvr_ser_rcv	uuh2_data_out[5]
ipp_ind_uh2_data6		uuh2_data_in[6]	ipp_do_uh2_data6	ipp_do_uh2_speed	uuh2_data_out[6]
ipp_ind_uh2_data7		uuh2_data_in[7]	ipp_do_uh2_data7	ipp_do_uh2_suspend	uuh2_data_out[7]
USB Port 2 Direction Control					
			ipp_obe_uh2_data0	ipp_obe_uh2_oe_n	uuh2_data_out_enable
			ipp_obe_uh2_data1	1'b0	uuh2_data_out_enable
			ipp_obe_uh2_data2	1'b0	uuh2_data_out_enable
			ipp_obe_uh2_data3	ipp_obe_uh2_rxvp_txdp_dat	uuh2_data_out_enable
			ipp_obe_uh2_data4	ipp_obe_uh2_rxvm_txdm_se0	uuh2_data_out_enable
			ipp_obe_uh2_data5	ipp_obe_uh2_xcvr_ser_rcv	uuh2_data_out_enable
			ipp_obe_uh2_data6	1'b0	uuh2_data_out_enable
			ipp_obe_uh2_data7	1'b0	uuh2_data_out_enable

30.6.3 USB OTG Controller

The OTG controller offers HS/FS/LS capabilities in Host mode and HS/FS in device mode.

30.6.3.1 Host Mode

The controller supports direct connection of a FS/LS device (without external hub). Although there is no separate Transaction Translator block in the system. The transaction translator function normally associated with a USB 2.0 high speed hub has been implemented within the DMA and Protocol engine blocks to support connection to full and low speed devices.

30.6.3.2 Peripheral (Device) Mode

- Up to 8 bidirectional endpoints
- High/full speed operation
- Supports HNP, SRP.
- Remote wakeup capable

30.6.3.3 Special Considerations

The OTG port functions as gateway between the Host 1 Port and the OTG transceiver when in Bypass mode.

30.6.3.4 OTG Port Signal Connections and Signal Muxing

Table 30-7 describes the Signal connections from the USB core to the OTG port OUTPUT PADS.

Table 30-7. Signal Connections

I/O PAD	Type	Input	Output	Direction Control
USB_ULPI_CLK	I/O	ipp_ind_otg_clk		
USB_ULPI_DIR	IN	ipp_ind_otg_dir		
USB_ULPI_STP	OUT		ipp_do_otg_stp	
USB_ULPI_NXT	IN	ipp_ind_otg_nxt		
USB_ULPI_DATA0	I/O	ipp_ind_otg_data0	ipp_do_otg_data0	ipp_obe_otg_data0
USB_ULPI_DATA1	I/O	ipp_ind_otg_data1	ipp_do_otg_data1	ipp_obe_otg_data1
USB_ULPI_DATA2	I/O	ipp_ind_otg_data2	ipp_do_otg_data2	ipp_obe_otg_data2
USB_ULPI_DATA3	I/O	ipp_ind_otg_data3	ipp_do_otg_data3	ipp_obe_otg_data3
USB_ULPI_DATA4	I/O	ipp_ind_otg_data4	ipp_do_otg_data4	ipp_obe_otg_data4
USB_ULPI_DATA5	I/O	ipp_ind_otg_data5	ipp_do_otg_data5	ipp_obe_otg_data5
USB_ULPI_DATA6	I/O	ipp_ind_otg_data6	ipp_do_otg_data6	ipp_obe_otg_data6
USB_ULPI_DATA7	I/O	ipp_ind_otg_data7	ipp_do_otg_data7	ipp_obe_otg_data7

Table 30-8. OTG ULPI/Serial Muxing

USB TOP Input Signals			USB TOP Output Signals		
From I/O Mux	**Serial Mode**	**ULPI Mode**	**To I/O Mux**	**Serial Mode**	**ULPI Mode**
ipp_ind_otg_data 0		uotg_data_in [0]	ipp_do_otg_data0	ipp_do_otg_oe_n	uotg_data_out [0]
ipp_ind_otg_data 1		uotg_data_in [1]	ipp_do_otg_data1	ipp_do_otg_txdp_dat	uotg_data_out [1]
ipp_ind_otg_data 2		uotg_data_in [2]	ipp_do_otg_data2	ipp_do_otg_txdm_se0	uotg_data_out [2]
ipp_ind_otg_data 3	ipp_ind_otg_xcvr_ser_vp	uotg_data_in [3]	ipp_do_otg_data3	ipp_do_otg_xcvr_ser_ vp	uotg_data_out [3]
ipp_ind_otg_data 4	ipp_ind_otg_xcvr_ser_vm	uotg_data_in [4]	ipp_do_otg_data4	ipp_do_otg_xcvr_ser_ vm	uotg_data_out [4]
ipp_ind_otg_data 5	ipp_ind_otg_xcvr_ser_rcv	uotg_data_in [5]	ipp_do_otg_data5		uotg_data_out [5]
ipp_ind_otg_data 6		uotg_data_in [6]	ipp_do_otg_data6	ipp_do_otg_speed	uotg_data_out [6]
ipp_ind_otg_data 7		uotg_data_in [7]	ipp_do_otg_data7	ipp_do_otg_suspend	uotg_data_out [7]
OTG Port Direction Control					
			ipp_obe_otg_data0	1’b1	uotg_data_out _enable
			ipp_obe_otg_data1	1'b0	uotg_data_out _enable
			ipp_obe_otg_data2	1'b0	uotg_data_out _enable
			ipp_obe_otg_data3	ipp_obe_otg_xcvr_ser _vp	uotg_data_out _enable
			ipp_obe_otg_data4	ipp_obe_otg_xcvr_ser _vm	uotg_data_out _enable
			ipp_obe_otg_data5	1'b0	uotg_data_out _enable
			ipp_obe_otg_data6	1'b0	uotg_data_out _enable
			ipp_obe_otg_data7	1'b0	uotg_data_out _enable

30.6.4 USB Power Control Module

The USB module supports suspend and wakeup functionality, but the circuit is considered application specific and therefore not part of the IP. An external circuit has been designed to place external transceivers in suspend mode, and wake them up either on a local request (CPU initiated) or on remote request by

detecting activity on the USB line. The wake-up logic can optionally wake the CPU when it is in sleep mode at the time of the request. The power control mechanism is described in the VUSB-HS-SPH and VUSB-HS-OTG reference manuals. For details on the power control module, refer to the Power Module creation guide.

30.6.4.1 Entering Suspend Mode

Suspend mode is always entered under control of driver software by setting the appropriate bit INT PORTSC register. Once the controller is suspended, the clocks to the USB block can be stopped.

30.6.4.2 Wake-up Events

The power control module monitors the USB bus when the USB core is in the suspend state. Depending on whether the core is on Host or Device mode, a number of wakeup conditions are detected. Upon detection of a wakeup condition, an interrupt (asynchronous) is generated on the CPU complex. This interrupt will also re-activate the clocks if these were stopped during the suspend.

30.6.4.2.1 Host Mode Events

The Host controller wakes-up on the following events:

Remote Wakeup Request

A peripheral can request the host to re-activate the bus by driving wake-up signaling on the Dm/Dp lines. The power control module will detect a J-K transition on the Dm/Dp lines and signal the wakeup request to the core.

Wake on Over-Current

If wake on overcurrent is enabled in the PORTSC registers, the power control module will signal a wakeup condition to the USB core.

Wake on Disconnect

The Power Control module detects disconnect events by monitoring the Dp/Dm lines. When a disconnect event is detected (Dm = Dp = 0) and the Wake on Disconnect is enabled in the PORTSC register, the core will be notified.

Wake on Connect

Similar to the Wake on Disconnect, the power control module detects a connect event (Dm or Dp High) and signals this to the USB core by setting the pwrctl_wakeup signal if enabled in the PORTSC register.

30.6.4.2.2 Device mode events

When the OTG controller is configured for peripheral operation, the power control module will detected following events:

Detection of bus activity

Any non-idle condition on the USB bus will activate the wakeup output of the power control module to notify the USB core of the wakeup event.

30.6.5 TLL Mode

The Transceiver less Link Logic circuit allows two micro-controllers to use USB for an Inter-processor Communication Link (ICL) without using conventional USB transceivers. The TLL muxes support serial type interfacing only and are available on Host Port 1 and Host Port 2.

30.6.5.1 TLL Functional Description

The TLL logic is a logic representation of 2 serial transceivers connected by a USB cable. The USB bus DM/DP states are modeled internally in the TLL function, such that the USB I/O port acts as if it were a transceiver.

In a regular USB implementation with serial PHY's, the speed selection on the USB bus is done by means of a pull-up resistor on the Peripheral side either on the DM (low speed) or the DP (full Speed) line. This Pull-up pulls one of the USB lines high when the bus is idle.

This option cannot be modeled in logic as the serial USB interface does not provide for such a signal. The TLL mux is therefore configured for Full Speed only operation.

The IDLE condition of the bus is determined by the OEn signals and suspend signals on both sides of the mux. When both OEn signals are high, or when one or both suspend signals are high, the idle condition is assumed. The TLL block then drives TxDp high and TxDm low on both sides of the block.

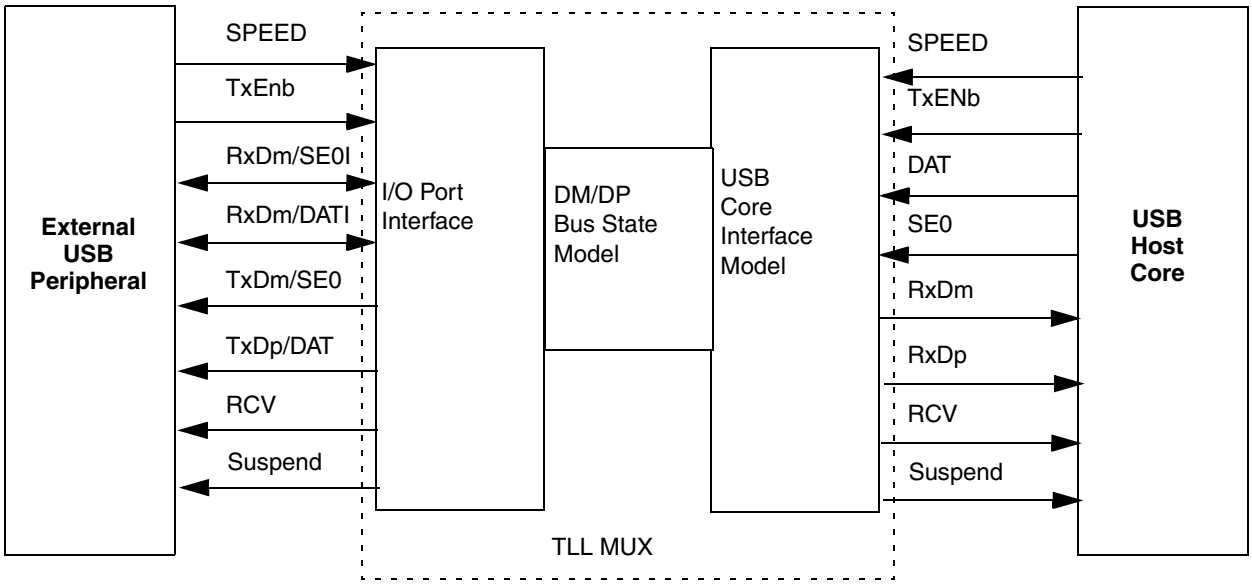


Figure 30-4. TLL Mux Functional Diagram

30.6.5.2 Host Port 1

On Host Port1, the TLL function is integrated with the Bypass function (see Section 30.6.6, "USB Bypass Mode") and the Transceiver Type conversion logic. TLL mode is the default mode for this port. It can be disabled by setting bit 4 in the USBCONTROL register.

Table 30-9. Port 1 TLL and PHY Mode Pin Connections

	TLL mode		PHY Mode	
Pin	**I/O**	**External Device Pin**	**I/O**	**External PHY Pin**
SPEED	I	speed	O	speed
TxEnb	I	TxEnb	O	TxEnb
RxVm	I	TxDm	I	RxVm
RxVp	I	TxDp	I	RxVp
TxDm	O	RxDm	O	TxDm
TxDp	O	RxDp	O	TxDp
RCV	O	RCV	I	RCV
Suspend	I	suspend	O	SuspendM

30.6.5.3 Host Port 2

The TLL module on Host Port 2 contains the TLL logic and the Serial Transceiver Type conversion logic. The Interface type conversion is available in both TLL and Non TLL modes.

TLL operation is the default mode. It can be turned off for operation with an external transceiver by setting bit 5 in the USBCONTROL register.

The USBCONTROL register.

Table 30-10. Port 2 TLL and PHY Mode Pin Connections

	TLL mode		PHY Mode	
Pin	**I/O**	**External Device Pin**	**I/O**	**External PHY Pin**
SPEED	I	speed	O	speed
TxEnb	I	TxEnb	O	TxEnb
RxVm	I	TxDm	I	RxVm
RxVp/	I	TxDp	O	RxVp
TxDm	O	RxDm	O	TxDm
TxDp	O	RxDp	O	TxDp
RCV	O	RCV	I	RCV
Suspend	I	suspend	O	SuspendM

30.6.6 USB Bypass Mode

The USB Bypass mode is a special mode that allows for the transceiver on the OTG port to be used as transceiver for a USB peripheral device connected to host port 1. This mode is only available for full/low speed serial transceivers.

The Bypass module is combination of:

- the bypass function (Host Port1 - OTG port pass through)
- TLL function for Host Port 1
- Serial Transceiver Interface Conversion

30.6.6.1 Bypass Mode operation

Bypass mode is enabled by writing a ‘1’ to bit 0 (BPE) in Figure 30-2.

Figure 30-5 shows the USB bypass mux functional diagram.

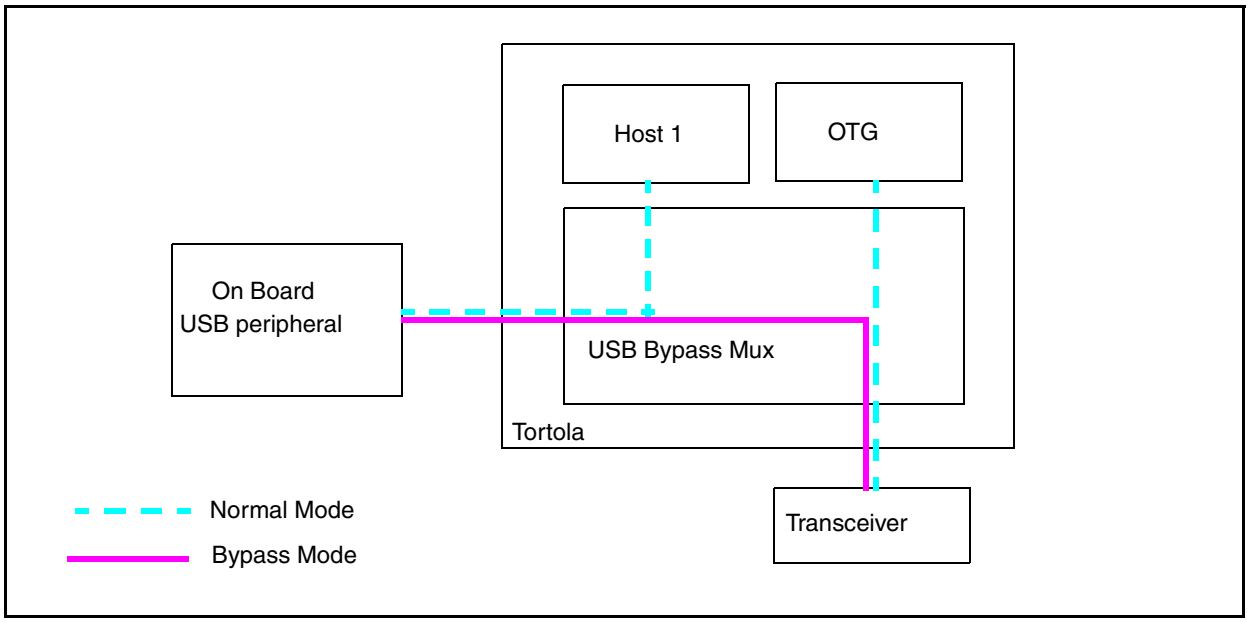


Figure 30-5. USB Bypass Mux Functional Diagram

In bypass mode, the serial interface signals from Host Port 1 are routed to the Serial Interface pins of the OTG port such that an external USB Peripheral device can use the OTG transceiver to connect to an external USB Host. As this function only works with USB peripherals directly connected to Host Port 1, the port is automatically set for TLL mode. Transceiver Interface Type conversion is available in Bypass mode such that the interface type of the OTG transceiver can be different from the Host 1 interface type.

The USB HOST1 core is disconnected from the port and the inputs RxDp, RCV and RxDm from the core are driven by bits 9 and 10 (H1BPVAL) in the Figure 30-2.The OTG core is also disconnected from its port and inputs RxDp, RCV and RxDm are driven by bits 25 and 26 (OBPVAL) from the Figure 30-2.

30.6.6.2 OTG and Host 1 pin functions

Table 30-11 list the pin functions of the Host 1 port when Bypass mode is enabled and the associated OTG pin. The Pin functions of the OTG port are not affected by Bypass mode.

Table 30-11. HOST1 Bypass Mode Pin Functions

	Unidirectional				Bidirectional				OTG PORT	
Pin	I/O	Single-Ended	I/O	Differential	I/O	Single- Ended	I/O	Differential	I/O	Pin
RxDm	I	RxDm	I	RxDm	I/O	SEOI/SE0O	I/O	RxDm/TxDm	O	TxDm
RxDp	I	RxDp	I	RxDp	I/O	DATI/DATO	I/O	RxDp/TxDP	O	TxDP
RCV	O	RCV	O	RCV		—	O	RCV	I	RCV
TxDm	O	SE0	O	TxDM		—		—	I	RxDM
TxDp	O	DAT	O	TxDP		—		—		RxDp
OEb	I	OEn	I	OEb	I	OEn	I	OEn		OEb
FS	I	Speed	I	FS	I	Speed	I	Speed		Speed
Suspend	I	Suspend	I	Suspend	I	Suspend	I	Suspend		Suspend

30.6.7 ULPI/Serial MUX

Both Host2 and OTG cores can be configured by software for ULPI or Serial PHY operation. The ULPI/Serial mux selects between ULPI interface signals and Serial PHY interface signals. The mux is controlled by the PHY Select signals from the USB core and is switched when the software selects the interface mode.

The default configuration for the mux is Serial mode. Switching to ULPI mode is done by writing the Parallel Transceiver Select (PTS) bits in the PORTSC register with 0b10.

30.6.8 Interrupts

30.6.8.1 USB Core Interrupts

Each USB core uses one dedicated vector in the Interrupt Table. The vector numbers associated with each of the cores can be found in the Interrupt section.

With the exception of the wake-up interrupts, all of the interrupt sources are controlled in the USB Cores. Refer to the USB Core documentation for details.

30.6.8.2 USB Wake-Up Interrupts

Each USB Core has an associated wake-up interrupt. The wake-up interrupts are generated outside the USB cores' but use the same vector as the corresponding Cores' interrupt. These interrupt are generated by the Power Control Modules which run on the 32KHz standby clock.The wake-up interrupt is designed to work even when the USB and CPU clocks are disabled, such that a wake-up condition on the USB bus

can re-activate the CPU clocks. Therefore, this interrupt request propagates through combinatorial logic to the CPU's interrupt module.

Because the wake-up interrupt is generated and cleared on a 32 KHz clock, this interrupt request will respond very slowly to clear actions. For this reason, the software must disable the wake-up interrupt to clear the request flag. Disabling the interrupt will mask the request instantaneously as this is clocked by the CPU clock. The software should than wait for at least 3 32 KHz clock cycles before re-enabling this interrupt to allow sufficient time for the request flag to clear. As this interrupt is only used during low-power modes of the USB, it is sufficient to enable the wake-up interrupt just prior to enter USB suspend mode.

30.7 Initialization/Application Information

This section described the detailed application knowledge for Host1, Host2 and OTG ports. It can be generally divided in two parts, one is for Host and the other is for Device. Host part is applied to three ports, Device part is only applied to OTG port. In following register description, Device related content is just for OTG, and Host related content is for all three ports.

30.7.1 Software Model

The Device API provides a framework of routines to control the USB-HS OTG High-Speed USB On-The-Go peripheral in USB device applications. It includes an application to respond to the device framework commands issued by a USB host.

The USB-HS OTG High-Speed USB On-The-Go Device API is designed to significantly simplify the software tasks required to develop a USB device application. The API presents a high-level data transfer interface to the user's application code. All the register, interrupt and DMA interactions with the USB-HS OTG High-Speed USB On-The-Go core are managed by the API. The API also includes routines that handle all the USB device framework commands which are required for all USB devices.

The Host Stack provides a layered software architecture to control all aspects of a USB bus system. The Host Controller Device (HCD) interface controls the functions of an embedded EHCI host controller. The USB driver layer provides all the USB driver functions to enumerate, manage and schedule a USB bus system, while the upper layers of the stack support standard USB device class interfaces to the device drives running on your embedded system.

For details on the USB-HS OTG High-Speed USB On-The-Go Software Stack refer the documentation provided with the software products.

- ANSI-C OTG software Stack provides Host and Device application support. USB software included with the USB-HS OTG High-Speed USB On-The-Go core is tested with the hardware.
- OTG Application Program Interface (API) handles OTG protocols. Connect and disconnect events are handled as well as the OTG Host Negotiation Protocol (HNP) and Session Request Protocol (SRP) state machines. The OTG code calls the Host or Device API functions based on the connection state of the OTG state machines.
- Host API to speed up Host software development. Simple API calls allow direct interaction with USB pipes. Additional layers support bus enumeration, bus management and a growing set of supported USB classes.

- Device API to speed up peripheral development. USB peripheral characteristics such as endpoints, configurations, interfaces, and alternate settings are controlled by supplied ANSI-C firmware. Device Frame work command set reduces software development time. Simple API interface allows quick coding of USB device applications.

30.7.1.1 Device Data Structure

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers.

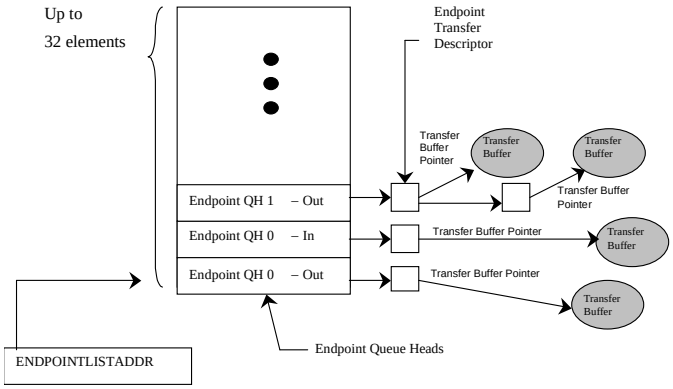


Figure 30-6. End Point Queue Head Organization

The USB-HS OTG High-Speed USB On-The-Go Device API incorporates and abstracts for the application developer all of the information contained in the device operational model.

30.7.1.2 Host Data Structure

The host data structures are used to communicate control, status, and data between software and the Host Controller. The Periodic Frame List is an array of pointers for the periodic schedule. A sliding window on the Periodic Frame List is used. The Asynchronous Transfer List is where all the control and bulk transfers are managed. The USB-HS OTG High-Speed USB On-The-Go Host API incorporates and abstracts for the application developer all of the information contained in the host operational model.

30.7.2 Register Interface

Slave accesses from the controlling processor enables access to the configuration, control, and status registers. One function of the system address map is the registers base address, which must begin on a **DWord** (32-bit) boundary. Register offset definitions are listed in the table below.

Configuration, control and status registers are divided into three categories, identification, capability, and operational registers.

- Identification registers are used to declare the slave interface presence along with the complete set of the hardware configuration parameters.

- Static, read only capability registers define the software limits, restrictions, and capabilities of the host/device controller.
- Operational registers are comprised of dynamic control or status registers that may be read only, read/write, or read/write to clear. The following sections define the use of these registers.

EHCI registers are listed alongside device registers to show the complementary nature of host and device control.

Note: Host mode EHCI compatibility begins at offset 0x100. If it is necessary to begin the EHCI register set at offset 0x000, the identification registers are disabled from the address map by connecting the upper most address bit of the slave interface to a logic level '1' and adjusting the offsets below accordingly.

Table 30-12. Interface Register Sets

Offset	Register Set	Explanation
000h to 0fch	Identification Registers	Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.
100h to 124h	Capability Registers	Capability registers specify the limits, restrictions, and capabilities of a host/device controller implementation. These values are used as parameters to the host/device controller driver.
140h to 1FCh	Operational Registers	Operational registers are used by the system software to control and monitor the operational state of the host/device controller.

30.7.2.1 Configuration, Control and Status Register Set

Table 30-13. Configuration, Control, and Status Register Set

Offset	Size (bytes)	Mnemonic	Register Name	DEV	OTG	SPH	MPH
000h	4	ID	Identification Register	÷	÷	÷	÷
004h	4	HWGENERAL	General Hardware Parameters	÷	÷	÷	÷
008h	4	HWHOST	Host Hardware Parameters		÷	÷	÷
00Ch	4	HWDEVICE	Device Hardware Parameters	÷	÷		
010h	4	HWTXBUF	TX Buffer Hardware Parameters	÷	÷	÷	÷

Table 30-13. Configuration, Control, and Status Register Set (continued)

Offset	Size (bytes)	Mnemonic	Register Name	DEV	OTG	SPH	MPH
014h	4	HWRXBUF	RX Buffer Hardware Parameters	÷	÷	÷	÷
018h	4	HWTTTXBUF	TT-TX Buffer Hardware Parameters				÷
01Ch	4	HWTTRXBUF	TT-RX Buffer Hardware Parameters				÷
020h-0FCh	232	Reserved	N/A				
080h	4	GPTIMER0LD	General Purpose Timer #0 Load Register				
084h	4	GPTIMER0CTRL	General Purpose Timer #0 Control Register				
088h	4	GPTIMER1LD	General Purpose Timer #1 Load Register				
08ch	4	GPTIMER1CTRL	General Purpose Timer #1 Control Register				
100h	1	CAPLENGTH	Capability Register Length	÷	÷	÷	÷
101h	1	Reserved	N/A				
102h	2	HCIVERSION	Host Interface Version Number		÷	÷	÷
104h	4	HCSPARAMS	Host Ctrl. Structural Parameters		÷	÷	÷
108h	4	HCCPARAMS	Host Ctrl. Capability Parameters		÷	÷	÷
10Ch–11Fh	20	Reserved	N/A				
120h	2	DCIVERSION	Dev. Interface Version Number	÷	÷		
122h	2	Reserved	N/A	÷			
124h	4	DCCPARAMS	Device Ctrl. Capability Parameters	÷	÷		
128h–13Ch	24	Reserved	N/A				

Table 30-13. Configuration, Control, and Status Register Set (continued)

Offset	Size (bytes)	Mnemonic	Register Name	DEV	OTG	SPH	MPH
140h	4	USBCMD	USB Command	÷	÷	÷	÷
144h	4	USBSTS	USB Status	÷	÷	÷	÷
148h	4	USBINTR	USB Interrupt Enable	÷	÷	÷	÷
14Ch	4	FRINDEX	USB Frame Index	÷	÷	÷	÷
150h	4	Reserved	4G Segment Selector				
154h	4	PERIODICLISTBASE	Frame List Base Address		÷	÷	÷
		Device Addr	USB Device Address	÷	÷		
158h	4	ASYNCLISTADDR	Next Asynchronous List Address		÷	÷	÷
		Endpointlist Addr	Address at Endpoint list in memory	÷	÷		
15Ch	4	ASYNCTTSTS	Asynchronous Buffer Status For Embedded TT.				÷
160h	4	BURSTSIZE	Programmable Burst Size	÷	÷	÷	÷
164h	4	TXFILLTUNING	Host Transmit Pre-Buffer Packet Tuning		÷	÷	÷
168h	4	TXTTFILLTUNING	Host TT Transmit Pre-Buffer Packet Tuning				÷
16Ch	4	N/A	Reserved				
170-17Ch	16	N/A	Reserved				
180h	4	CONFIGFLAG	Configured Flag Register		÷	÷	÷
184h	4	PORTSC1	Port Status/Control 1	÷	÷	÷	÷
188h	4	PORTSC2	Port Status/Control 2				÷
…	4	PORTSCx	Port Status/Control x				÷
1A0h	4	PORTSC8	Port Status/Control 8				÷

Table 30-13. Configuration, Control, and Status Register Set (continued)

Offset	Size (bytes)	Mnemonic	Register Name	DEV	OTG	SPH	MPH
1A4h	4	OTGSC	On-The-Go (OTG) Status and Control		÷		
1A8h	4	USBMODE	USB Device Mode	÷	÷	÷	÷
1ACh	4	ENPDTSETUPSTAT	Endpoint Setup Status	÷	÷		
1B0h	4	ENDPTPRIME	Endpoint Initialization	÷	÷		
1B4h	4	ENDPTFLUSH	Endpoint De-Initialize	÷	÷		
1B8h	4	ENDPTSTATUS	Endpoint Status	÷	÷		
1BCh	4	ENDPTCOMPLETE	Endpoint Complete	÷	÷		
1C0h	4	ENDPTCTRL0	Endpoint Control 0	÷	÷		
1C4h	4	ENDPTCTRL1	Endpoint Control 1	÷	÷		
…	4	ENDPTCTRLx	Endpoint Control x	÷	÷		
1FCh	4	ENDPTCTRL15	Endpoint Control 15	÷	÷		

NOTE	*Italic Text* indicates a deviation from EHCI for Device

30.8 Summary of Register Layouts

Table 30-14. HS-USB Register Summary

Name	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
000h ID	reserved								REVISION								NID								ID							
004h HWGENERAL	reserved																						SM	PHYM			PHYW	BWT	CLKC		RT
008h HWHOST	TTPER								TTASY								reserved												NPORT			HC
00Ch HWDEVICE	reserved																										DEVP					DC
010h HWTXBUF	TXLC	reserved							TXCHANADD								TXADD								TXBURST							
014h HWRXBUF	reserved																RXADD								RXBURST							
020h Reserved	reserved																															
… Reserved	reserved																															
0FCh Reserved	reserved																															
100h CAPLENGTH																									CAPLENGTH							
101h Reserved																									reserved							
102h HCIVERSION																	HCIVERSION															
104h HCSPARAMS	reserved			N_TT			N_PTT			reserved			PI				N_CC			N_PCC				reserved		PPC			N_PORTS			
108h HCCPARAMS	reserved																EEC{[7:0]								IST[7:4]				R	R	PFL	ADC
10Ch Reserved	reserved																															
… Reserved	reserved																															
11Fh Reserved	reserved																															
120h DCIVERSION																	DCIVERSION															
122h Reserved																	reserved															

Table 30-14. HS-USB Register Summary (continued)

Name	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
124h DCCPARAMS	reserved																							HC	DC	R	R	DEN				
128h Reserved	reserved																															
… Reserved	reserved																															
13Ch Reserved	reserved																															
140h USBCMD	reserved								ITC								FS2	R	SUTW	ATDTW	ASPE	R	ASP1	ASP0	LR	IAA	ASE	PSE	FS1	FS0	RST	RS
144h USBSTS	reserved																AS	PS	RCL	HCH	reserved			SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	UI
148h USBINTR	reserved																							SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE
14Ch FRINDEX	reserved																FRINDEX[13:0]															
150h Reserved	reserved																															
154h PERIODICLISTBASE	PERBASE[31:12]																							reserved								
Device Addr	USBADR[31:25]								reserved																							
158h ASYNCLISTADDR	ASYBASE[31:5]																							reserved								
Endpointlist Addr	EPBASE[31:11]																							reserved								
15Ch ASYNCTTSTS	reserved																													TTAC	TTAS	
160h BURSTSIZE	reserved																TXPBURST									RXPBURST						
164h TXFILLTUNING	reserved										TXFIFOTHRES						R		TXSCHHEALTH					TXSCHOH							
168h TXTTFILLTUNING	reserved																			TXTTSCHHEALTH				R	TXTTSCHOH						
16Ch Reserved	reserved																															
170 ULPI Viewport	ULPI Viewport [optional]																															

Table 30-14. HS-USB Register Summary (continued)

Name	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
174 Reserved	reserved																															
… Reserved	reserved																															
17Ch Reserved	reserved																															
180h CONFIGFLAG	set to zero																															1
184h PORTSC1	PTS		STS	PTW	PSPD		R	PFSC	PHCD	WKOC	WKDS	WKCN	PTC				PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
188h PORTSC2	PTS		STS	PTW	PSPD		R	PFSC	PHCD	WKOC	WKDS	WKCN	PTC				PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
… PORTSCx	PTS		STS	PTW	PSPD		R	PFSC	PHCD	WKOC	WKDS	WKCN	PTC				PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
1A0h PORTSC8	PTS		STS	PTW	PSPD		R	PFSC	PHCD	WKOC	WKDS	WKCN	PTC				PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
1A4h OTGSC	R	DPIE	1msE	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	R	DPIS	1msS	BSEIS	BSVIS	ASVIS	AVVIS	IDIS	R	DPS	1msT	BSE	BSV	ASV	AVV	ID	reserved		DP	OT	R	VC	VD	
1A8h USBMODE	reserved																									SDIS	SLOM	ES	CM			
1ACh ENPDTSETUPSTAT	reserved																ENDPTSETUPSTAT															
1B0h ENDPTPRIME	PETB[15:0]																PERB[15:0]															
1B4h ENDPTFLUSH	FETB[15:0]																FERB[15:0]															
1B8h ENDPTSTATUS	ETBR[15:0]																ERBR[15:0]															
1BCh ENDPTCOMPLETE	ETCE[15:0]																ERCE[15:0]															
1C0h ENDPTCTRL0	reserved								TXE	reserved			TXT		R	TXS	reserved								RXE	reserved			RXT		R	RXS
1C4h ENDPTCTRL1	reserved								TXE	TXR	TXI	R	TXT		TXD	TXS	reserved								RXE	RXR	RXI	R	RXT		RXD	RXS

Table 30-14. HS-USB Register Summary (continued)

Name	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
… ENDPTCTRLx	reserved								TXE	TXR	TXI	R	TXT		TXD	TXS	reserved								RXE	RXR	RXI	R	RXT		RXD	RXS
1FCh ENDPTCTRL15	reserved								TXE	TXR	TXI	R	TXT		TXD	TXS	reserved								RXE	RXR	RXI	R	RXT		RXD	RXS

30.8.1 Identification Registers

Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.

30.8.1.1 ID

Address:Base + 000h

Default Value:Implementation Dependent

Attribute:Read Only

Size:32 bits

The Identification register (ID) provides a simple way to determine if the USB-HS OTG High-Speed USB On-The-Go USB 2.0 core is provided in the system. The ID register identifies the USB-HS OTG High-Speed USB On-The-Go USB 2.0 core and its revision.

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
Reserved								REVISION								1	1	NID						0	0	ID					

Figure 30-7. ID - Identification Register

The register fields are described in the following table.

Field	Description
ID[5:0]	Configuration number. This number is set to 0x05 and indicates that the peripheral is the USB-HS OTG High-Speed USB On-The-Go USB 2.0 core.
NID[5:0]	Ones complement version of ID[5:0].
REVISION[7:0]	Revision number of the core
Reserved	These bits are reserved and should be set to zero

30.8.1.2 HWGENERAL

Address:Base + 004h

Default Value:Implementation Dependent

Attribute:Read Only

Size:32 bits

General hardware parameters as defined in System Level Issues and Core Configuration.

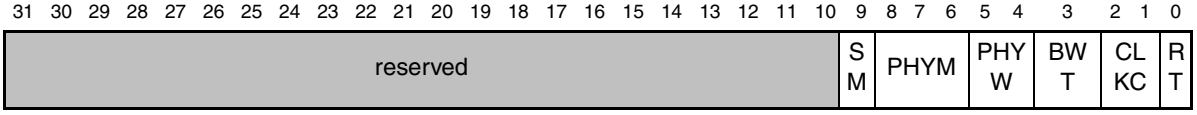


Figure 30-8. HWGENERAL—General Hardware Parameters

The register fields are described in the following table.

Table 30-15. HWGENERAL

Field	Description
reserved	Reserved. These bits are reserved and should be set to zero.
SM	VUSB_HS_PHY_SERIAL
PHYM	VUSB_HS_PHY_TYPE
PHYW	VUSB_HS_PHY16_8
BWT	Reserved for internal testing.
CLKC	VUSB_HS_CLOCK_CONFIGURATION
RT	VUSB_HS_RESET_TYPE

30.8.1.2.1 HWHOST

Address:Base + 008h

Default Value:Implementation Dependent

Attribute:Read Only

Size:32 bits

Host hardware parameters as defined in System Level Issues and Core Configuration.

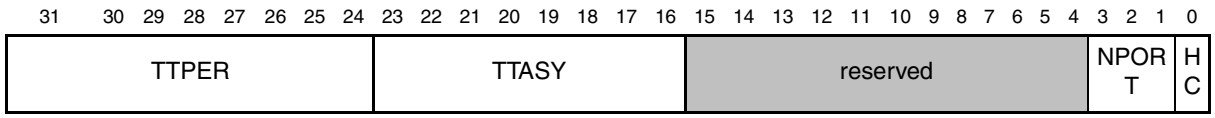


Figure 30-9. HWHOST—Host Hardware Parameters

The register fields are described in the following table.

Table 30-16. HWHOST Field Descriptions

Field	Description
TTPER	VUSB_HS_TT_PERIODIC_CONTEXTS
TTASY	VUSB_HS_TT_ASYNC_CONTEXTS
reserved	Reserved. These bits are reserved and should be set to zero.
NPORT	VUSB_HS_NUM_PORT-1
HC	VUSB_HS_HOST

30.8.1.2.2 HWDEVICE

Address:Base + 00Ch

Default Value:Implementation Dependent

Attribute:Read Only

Size:32 bits

Device hardware parameters as defined in System Level Issues and Core Configuration.

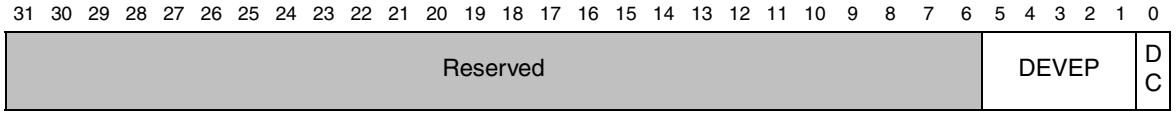


Figure 30-10. HWDEVICE—Device Hardware Parameters

The register fields are described in the following table.

Field	Description
Reserved	Reserved. These bits are reserved and should be set to zero.
DEVEP	VUSB_HS_DEV_EP
DC	device capable; [VUSB_HS_DEV /= 0]

30.8.1.2.3 HWTXBUF

Address:Base + 010h

Default Value:Implementation Dependent

Attribute:Read Only

Size:32 bits

TX buffer hardware parameters as defined in System Level Issues and Core Configuration.

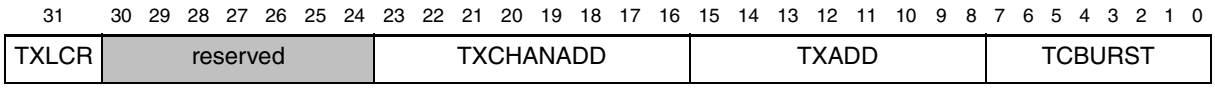


Figure 30-11. HWTXBUF—TX Buffer Hardware Parameters

The register fields are described in the following table.

Field	Description
TXLC	VUSB_HS_TX_LOCAL_CONTEXT_REGISTERS
reserved	Reserved. These bits are reserved and should be set to zero.
TXCHANADD	VUSB_HS_TX_CHAN_ADD
TXADD	VUSB_HS_TX_ADD
TCBURST	VUSB_HS_TX_BURST

30.8.1.2.4 HWRXBUF

Address:Base + 014h

Default Value:Implementation Dependent

Attribute:Read Only

Size:32 bits

RX buffer hardware parameters as defined in System Level Issues and Core Configuration.

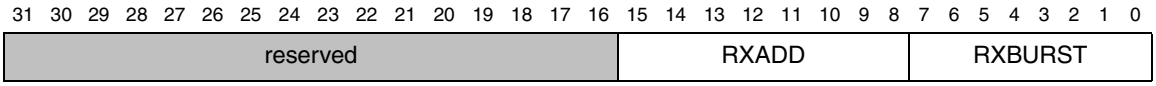


Figure 30-12. WRXBUF—RX Buffer Hardware Parameters

The register fields are described in the following table.

Field	Description
reserved	Reserved. These bits are reserved and should be set to zero.
RXADD	VUSB_HS_RX_ADD
RXBURST	VUSB_HS_RX_BURST

30.8.1.3 Device/Host Capability Registers

Device/Host Capability registers specify the software limits, restrictions, and capabilities of the host/device controller implementation.

30.8.1.3.1 CAPLENGTH—EHCI Compliant

Address:Base + 100h

Default Value:40h

Attribute:Read Only

Size:8 bits

This register is used to indicate which offset to add to the register base address at the beginning of the Operational Register.

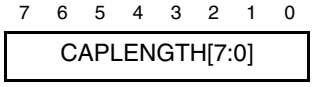


Figure 30-13. CAPLENGTH—Capability Register Length

30.8.1.3.2 HCIVERSION—EHCI Compliant

Address:Base + 102h

Default Value:0100h

Attribute:Read Only

Size:16 bits

This is a two-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.

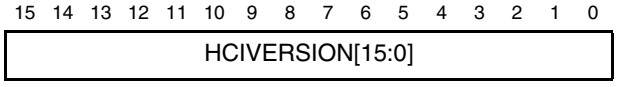


Figure 30-14. HCIVERSION—Host Interface version number

30.8.1.3.3 HCSPARAMS—EHCI Compliant with extensions

Address:Base + 104h

Default Value:Implementation Dependent

Attribute:Read Only

Size:32 bits

Port steering logic capabilities are described in this register.

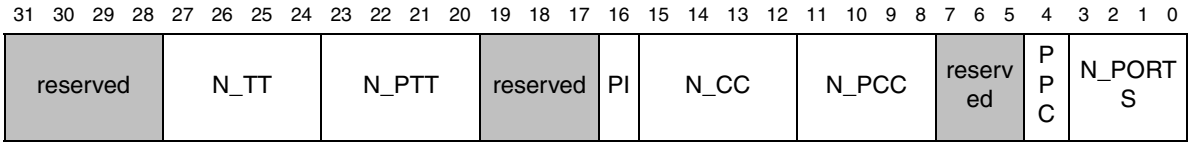


Figure 30-15. HCSPARAMS—Host Control Structural Parameters

The register fields are described in the following table.

Table 30-17. HCSPARAMS Field Descriptions

Field	Description
Reserved	Reserved. These bits are reserved and should be set to zero.
Pi	Port Indicators (P INDICATOR). This bit indicates whether the ports support port indicator control. When set to one, the port status and control registers include a read/writeable field for controlling the state of the port indicator. This field will always be “1”.
N_TT[3:0]	Number of Transaction Translators (N_TT). This field indicates the number of embedded transaction translators associated with the USB2.0 host controller. For Multi-Port Host this field will always equal “0001”. For all other implementations, N_TT = “0000”. This in a non-EHCI field to support embedded TT.
N_PTT[3:0]	Number of Ports per Transaction Translator (N_PTT). This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. For Multi-Port Host this field will always equal N_PORTS. For all other implementations, N_PTT = “0000”. This in a non-EHCI field to support embedded TT.
N_CC[3:0]	Number of Companion Controller (N_CC). This field indicates the number of companion controllers associated with this USB2.0 host controller. A zero in this field indicates there are no internal Companion Controllers. Port-ownership hand-off is not supported. A value larger than zero in this field indicates there are companion USB1.1 host controller(s). Port-ownership hand-offs are supported. High, Full- and Low-speed devices are supported on the host controller root ports. In this implementation this field will always be “0”.

Table 30-17. HCSPARAMS Field Descriptions (continued)

Field	Description
N_PCC[3:0]	Number of Ports per Companion Controller. This field indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC. In this implementation this field will always be “0”.
PPC	Port Power Control. This field indicates whether the host controller implementation includes port power control. A one indicates the ports have port power switches. A zero indicates the ports do not have port power switches. The value of this field affects the functionality of the Port Power field in each port status and control register. This field will always be “0” for a device only implementation.
N_PORTS[3:0]	Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register. Valid values are in the range of 1h to Fh. A zero in this field is undefined. The number of ports for a host implementation is parameterizable from 1 to 8. This field will always be 1 for device only implementation.

30.8.1.3.4 HCCPARAMS—EHCI Compliant

Address:Base + 108h

Default Value:0006h

Attribute:Read Only

Size:32 bits

This register identifies multiple mode control (time-base bit functionality) addressing capability.

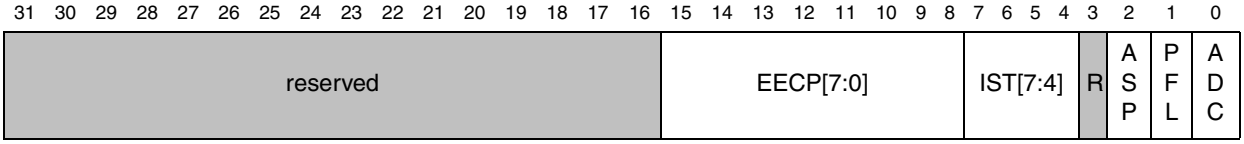


Figure 30-16. HCCPARAMS—Host Control Capability Parameters

The register fields are described in the following table.

Table 30-18. HCCPARAMS Field Descriptions

Field	Description
Reserved	Reserved. These bits are reserved and should be set to zero.
EECP[7:0]	EHCI Extended Capabilities Pointer. Default = 0. This optional field indicates the existence of a capabilities list. A value of 00h indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 40h or greater if implemented to maintain the consistency of the PCI header defined for this class of device. For this implementation this field is always “0”.
IST[7:4]	Isochronous Scheduling Threshold. Default = implementation dependent. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be “0”.
R	Reserved. These bits are reserved and should be set to zero.
ASP	Asynchronous Schedule Park Capability. Default = 1. If this bit is set to a one, then the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USBCMD register. This field will always be “1”
PFL	Programmable Frame List Flag. If this bit is set to zero, then the system software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. If set to a one, then the system software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This field will always be “1”.
ADC	64-bit Addressing Capability. This field will always be “0”. No 64-bit addressing capability is supported.

DCIVERSION (Non-EHCI)

Address:Base + 120h

Default Value:Implementation Dependent

Attribute:Read Only

Size:16 bits

The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

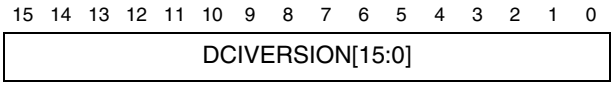


Figure 30-17. DCIVERSION—Device Interface Version Number

30.8.1.3.5 DCCPARAMS (Non-EHCI)

Address:Base + 124h

Default Value:Implementation Dependant

Attribute:Read Only

Size:32 bits

These fields describe the overall host/device capability of the controller.

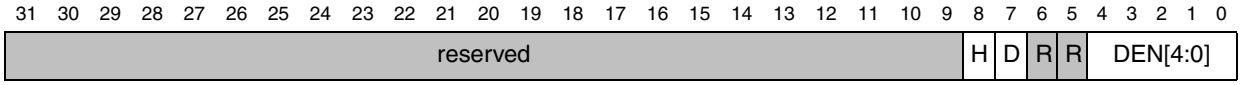


Figure 30-18. DCCPARAMS—Device Control Capability Parameters

The register fields are described in the following table.

Table 30-19. DCCPARAMS Field Descriptions

Field	Description
Reserved	Reserved. These bits are reserved and should be set to zero.
HC	Host Capable. When this bit is 1, this controller is capable of operating as an EHCI compatible USB 2.0 host controller.
DC	Device Capable. When this bit is 1, this controller is capable of operating as a USB 2.0 device.
R	Reserved. These bits are reserved and should be set to zero.
DEN[4:0]	Device Endpoint Number. This field indicates the number of endpoints built into the device controller. If this controller is not device capable, then this field will be zero. Valid values are 0–16.

30.8.1.4 Device/Host Timer Registers (Non-EHCI)

The host/device controller drivers can measure time related activities using these timer registers.

These registers are not part of the standard EHCI controller.

30.8.1.4.1 GPTIMER0LD (Non-EHCI)

Address: Base + 80h

Default Value: 00000000h

Attribute: Read/Write

Size: 32 bits

This register contains the timer duration or load value. See the GPTIMER0CTRL (Non-EHCI) for a description of the timer functions.

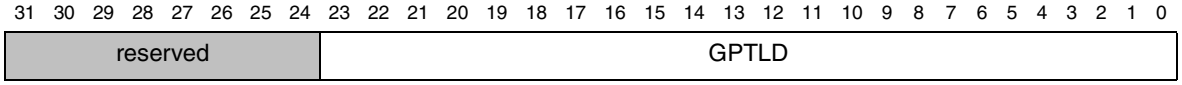


Figure 30-19. GPTIMER0LD-General Purpose Timer #0 Load Register

The register fields are described in the following table.

Table 30-20. GPTIMER0LD Field Descriptions

Field	Description
Reserved	Reserved. These bits are reserved and should be set to zero.
H\GPTLD	General Purpose Timer Load Value. This field is the value to be loaded into the GPTCNT countdown timer on a reset action. This value represents the time in microseconds minus 1 for the timer duration.

30.8.1.4.2 GPTIMER0CTRL (Non-EHCI)

Address: Base + 84h

Default Value: 00000000h

Attribute: Read Only, Write Only, Read/Write

Size: 32 bits

This register contains the control for the timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two modes supported by this timer, the first is a one shot and the second is a looped count which is described in the register table below. When the timer counter value transitions to zero, an interrupt can be generated though the use of the timer interrupts in the USBSTS and USBINTR registers.

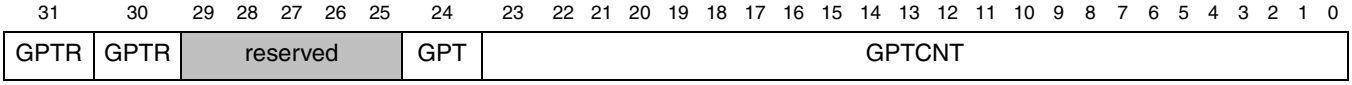


Figure 30-20. GPTIMER0LD-General Purpose Timer #0 Controller

The register fields are described in the following table.

Table 30-21. GPTIMER0LD Field Descriptions

Field	Description
GPTRUN	General Purpose Timer Run (0)-Timer Stop; (1)-Timer Run. This bit enable the GPT to run. Setting or Clearing this bit will not have an effect on the GPTCNT except.
GPTRST	General Purpose Timer Reset (0)-No action; (1)-Load Counter Value. This bit will reload the GPTCNT with the value in GPTLD.
reserved	Reserved. These bits are reserved and should be set to zero.
GPTMOD	General Purpose Timer Mode. (0)-One Shot; (1)-Repeat.In one-shot mode the timer will count to zero, generate an interrupt and stop until the timer is reset by software. In repeat mode the timer will count to zero, generate an interrupt and automatically reload the counter to begin again.
GPTCNT	General Purpose Timer Counter. This field is the value of running timer.

30.8.1.4.3 GPTIMER1LD (Non-EHCI)

Address: Base + 88h

Default Value: 00000000h

High-Speed USB Controller Core Reference

Attribute: Read/Write

Size: 32 bits

Same as GPTIMER0LD.

30.8.1.4.4 GPTIMER1CTRL (Non-EHCI)

Address: Base + 8Ch

Default Value: 00000000h

Attribute: Read Only, Write Only, Read/Write

Size: 32 bits

Same as GPTIMER0CTRL.

30.8.1.5 Device/Host Operational Registers

Operational registers are comprised of dynamic control or status registers that may be read only, read/write, or read/write to clear. The following sections define the use of these registers.

30.8.1.5.1 USBCMD

Address:Base + 140h

Default Value:00080B00h (host mode)

00080000h (device mode)

Attribute:Read Only, Read/Write, Write Only (field dependent)

Size:32 bits

The serial bus host/device controller executes the command indicated in this register.

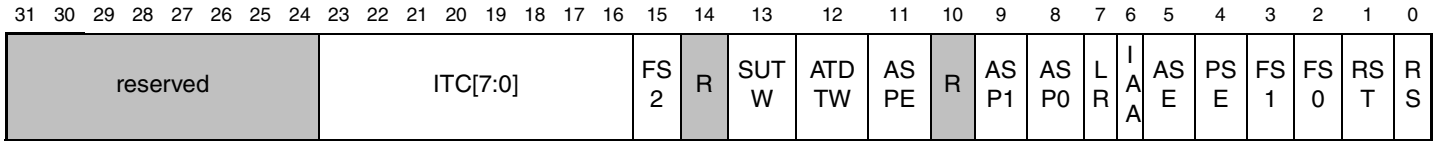


Figure 30-21. USBCMD—USB Command Register

The register fields are described in the following table.

Table 30-22. USBCMD—USB Command Register Field Descriptions

Field	Description
R, reserved	Reserved. These bits are reserved and should be set to zero.
ITC[7:0]	Interrupt Threshold Control. Read/Write. Default 08h. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 00h Immediate (no threshold) 01h 1 micro-frame 02h 2 micro-frames 04h 4 micro-frames 08h 8 micro-frames 10h 16 micro-frames 20h 32 micro-frames 40h 64 micro-frames
SUTW	Setup TripWire. Read/Write [device mode only].This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (See USBMODE) then there exists a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists. For more information on the use of this bit, see the Device Operational Model section of the USB-HS OTG High-Speed USB On-The-Go DEV reference manual.

Table 30-22. USBCMD—USB Command Register Field Descriptions (continued)

Field	Description
ATDTW	Add dTD TripWire. Read/Write[device mode only]. This bit is used as a semaphore to ensure the to proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit shall also be cleared by hardware when is state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized. For more information on the use of this bit, see the Device Operational Model section of the USB-HS OTG High-Speed USB On-The-Go DEV reference manual.
ASPE	Asynchronous Schedule Park Mode Enable (OPTIONAL). Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this bit defaults to a 1h and is R/W. Otherwise the bit must be a zero and is read-only. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. This field is set to "1" in this implementation.
ASP[1:0]	Asynchronous Schedule Park Mode Count (OPTIONAL). Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is RO. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. See Section 4.10.3.2 for full operational details. Valid values are 1h to 3h. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior. This field is set to 3h in this implementation.
LR	Light Host/Device Controller Reset (OPTIONAL). Read Only. Not Implemented. This field will always be "0".
IAA	Interrupt on Async Advance Doorbell. Read/Write. This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results.
ASE	Asynchronous Schedule Enable. Read/Write. Default 0b. This bit controls whether the host controller skips processing the Asynchronous Schedule. Valuesmeaning 0 Do not process the Asynchronous Schedule. 1 Use the ASYNCLISTADDR register to access theAsynchronous Schedule. Only the host controller uses this bit.
PSE	Periodic Schedule Enable. Read/Write. Default 0b. This bit controls whether the host controller skips processing the Periodic Schedule. Values meaning 0 Do not process the Periodic Schedule 1 Use the PERIODICLISTBASE register to access the PeriodicSchedule. Only the host controller uses this bit.

Table 30-22. USBCMD—USB Command Register Field Descriptions (continued)

Field	Description
FS[2:0]	Frame List Size. (Read/Write or Read Only). Default 000b. This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3 and 2. Values meaning 000 1024 elements (4096 bytes) Default value 001 512 elements (2048 bytes) 010 256 elements (1024 bytes) 011 128 elements (512 bytes) 100 64 elements (256 bytes) 101 32 elements (128 bytes) 110 16 elements (64 bytes) 111 8 elements (32 bytes) Only the host controller uses this field.
RST	Controller Reset (RESET) — Read/Write. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host Controller: When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior. Device Controller: When software writes a one to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a one to this bit when the device is in the attached state is not recommended, since the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.
RS	Run/Stop (RS)—Read/Write. Default 0b. 1=Run. 0=Stop. Host Controller: When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the host controller is in the Halted state (that is, HCHalted in the USBSTS register is a one). Device Controller: Writing a one to this bit will cause the device controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this will cause a detach event.

30.8.1.5.2 USBSTS

Address:Base + 144h

Default Value:00001000h (host mode)

00000000h (device mode)

Attribute:Read Only, Read/Write, Read/Write-Clear (field dependant)

Size:32 bits

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus. Software clears certain bits in this register by writing a 1 to them.

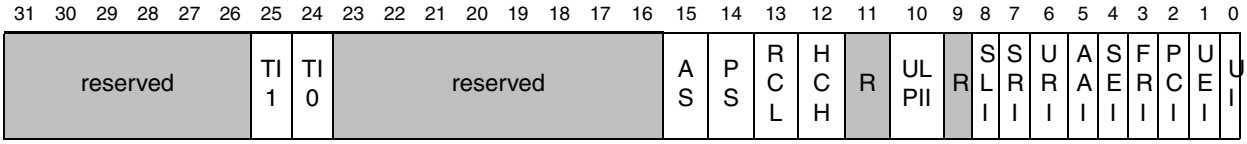


Figure 30-22. USBSTS—USB Status

The register fields are described in the following table.

Table 30-23. USBSTS Field Descriptions

Field	Description
Reserved	Reserved. These bits are reserved and should be set to zero.
TI1	General Purpose Timer Interrupt 1(GPTINT1)--R/WC. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero, writing a one to this bit will clear it.
TI0	General Purpose Timer Interrupt 0(GPTINT0)--R/WC. This bit is set when the counter in the GPTIMER0CTRL register transitions to zero, writing a one to this bit will clear it.
reserved	Reserved. These bits are reserved and should be set to zero.
AS	Asynchronous Schedule Status — Read Only. 0=Default. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). Only used by the host controller.
PS	Periodic Schedule Status — Read Only. 0=Default. This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). Only used by the host controller.
RCL	Reclamation — Read Only. 0=Default. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller.

Table 30-23. USBSTS Field Descriptions (continued)

Field	Description
HCH	HCHalted — Read Only. 1=Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (for example, internal error). Only used by the host controller.
R	Reserved. These bits are reserved and should be set to zero.
ULPII	ULPI Interrupt—R/WC. 0=Default. When the ULPI Viewport is present in the design, an event completion will set this interrupt. Used by both host and device controller. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1.
R	Reserved. These bits are reserved and should be set to zero.
SLI	DCSuspend—R/WC. 0=Default. When a device controller enters a suspend state from an active state, this bit will be set to a one. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller.
SRI	SOF Received—R/WC. 0=Default. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125ms in HS mode and will be synchronized to the actual SOF that is received. Since the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit.
URI	USB Reset Received—R/WC. 0=Default. When the device controller detects a USB Reset and enters the default state, this bit will be set to a one. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller.
AAI	Interrupt on Async Advance — R/WC. 0=Default. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller.
SEI	System Error— R/WC. This bit is not used in this implementation and will always be set to “0”.
FRI	Frame List Rollover — R/WC. The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [1 3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FHINDEX [12] toggles. Only used by the host controller.

Table 30-23. USBSTS Field Descriptions (continued)

Field	Description
PCI	Port Change Detect — R/WC. The Host Controller sets this bit to a one when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively. This bit is not EHCI compatible.
UEI	USB Error Interrupt (USBERRINT) — R/WC. When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set See Section (Reference Host Operation Model: Transfer/Transaction Based Interrupt—that is, 4.15.1 in EHCI Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com) for a complete list of host error interrupt conditions. See section Device Error Matrix in the USB-HS OTG High-Speed USB On-The-Go DEV reference manual. The device controller detects resume signaling only.
UI	USB Interrupt (USBINT)—R/WC. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

30.8.1.5.3 USBINTR

Address:Base + 148h

Default Value:00000000h

Attribute:Read/Write

Size:32 bits

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt is active. The USB Status register (USBSTS) still shows interrupt sources even if they are disabled by the USBINTR register, allowing polling of interrupt events by the software.

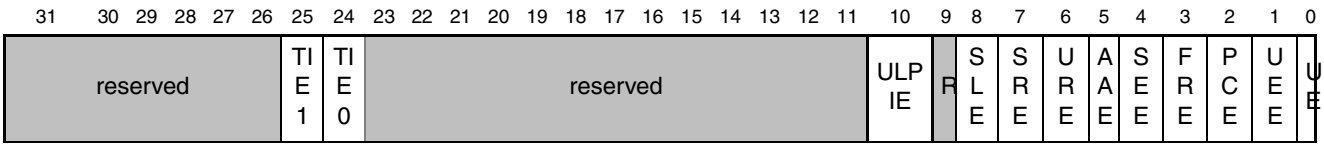


Figure 30-23. USBINTR—USB Interrupt Enable

The register fields are described in the following table.

Table 30-24. USBINTR Field Descriptions

Field	Interrupt Source	Description
Reserved	Reserved	These bits are reserved and should be set to zero.
TIE1	GPT Interrupt Enable1	When this bit is one and the GPTINT1 in USBSTS register is a one the controller will issue an interrupt. The interrupt is acknowledge by software clear the GPTINT1 bit.
TIE0	GPT Interrupt Enable0	When this bit is one and the GPTINT0 in USBSTS register is a one the controller will issue an interrupt. The interrupt is acknowledge by software clear the GPTINT0 bit.
reserved	Reserved.	These bits are reserved and should be set to zero.
ULPIE	ULPI Enable	When this bit is a one, and the ULPI Interrupt bit in the USBSTS register transitions, the controller will issue and interrupt. The interrupt is acknowledged by software writing a one to the ULPI Interrupt bit. Used by both host and device controller. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1.
reserved	Reserved.	These bits are reserved and should be set to zero.
SLE	Sleep Enable	When this bit is a one, and the *DCSuspend* bit in the USBSTS register transitions, the device controller will issue an interrupt. The interrupt is acknowledged by software writing a one to the *DCSuspend* bit. Only used by the device controller.
SRE	SOF Received Enable	When this bit is a one, and the *SOF Received* bit in the USBSTS register is a one, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the *SOF Received* bit.
URE	USB Reset Enable	When this bit is a one, and the *USB Reset Received* bit in the USBSTS register is a one, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the *USB Reset Received* bit. Only used by the device controller.
AAE	Interrupt on Async Advance Enable	When this bit is a one, and the Interrupt on *Async Advance* bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on *Async Advance* bit. Only used by the host controller.
SEE	System Error Enable	When this bit is a one, and the *System Error* bit in the USBSTS register is a one, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the *System Error* bit.

Table 30-24. USBINTR Field Descriptions (continued)

Field	Interrupt Source	Description
FRE	Frame List Rollover Enable	When this bit is a one, and the *Frame List Rollover* bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the *Frame List Rollover* bit. Only used by the host controller.
PCE	Port Change Detect Enable	When this bit is a one, and the *Port Change Detect* bit in the USBSTS register is a one, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the *Port Change Detect* bit.
UEE	USB Error Interrupt Enable	When this bit is a one, and the *USBERRINT* bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the *USBERRINT* bit in the USBSTS register.
UE	USB Interrupt Enable	When this bit is a one, and the *USBINT* bit in the USBSTS register is a one, the host/device controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the *USBINT* bit.

30.8.1.5.4 FRINDEX

Address:Base + 14Ch

Default Value:Undefined (free running counter)

Attribute:Read/Write in host mode, Read in device mode

Size:32 bits

This register is used by the host controller to index the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the USBCMD register.

This register must be written as a DWord. Byte writes produce-undefined results. This register cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit. A write to this register while the Run/Stop hit is set to a one produces undefined results. Writes to this register also affect the SOF value.

In device mode this register is read only and, the device controller updates the FRINDEX [13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX [13:3] will be checked against the SOF marker. If FRINDEX [13:3] is different from the SOF marker, FRINDEX [13:3] will be set to the SOF value and FRINDEX [2:0] will be set to zero (that is, SOF for 1 ms frame). If FRINDEX [13:3] is equal to the SOF value, FRINDEX [2:0] will be increment (that is, SOF for 125 us micro-frame.)

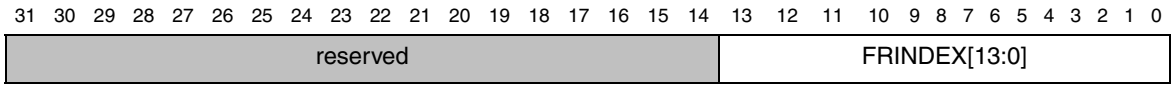


Figure 30-24. FRINDEX—USB Frame Index

The register fields are described in the following table.

Table 30-25. FRINDEX Register Field Descriptions

Field	Description
Reserved	Reserved. These bits are reserved and should be set to zero.
FRINDEX	Frame Index. The value, in this register, increments at the end of each time frame (for example, micro-frame). Bits [N: 3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD[Frame List Size] Number Elements N 000b (1024)12 001b (512)11 010b (256)10 011b (128)9 100b (64)8 101b (32)7 110b (16)6 111b (8)5 In device mode the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current microframe.

30.8.1.5.5 CTRLDSSEGMENT

Address:Base + 150h

Default Value:00000000h

Attribute:Read Only

Size:32 bits

This register is not used in this implementation.

30.8.1.5.6 PERIODICLISTBASE; DEVICEADDR

Address:Base + 154h

Default Value:00000000h

Attribute:Read/Write (Writes must be DWord Writes)

Size:32 bits

This register is shared between the host controller and the device controller operation.

Host Controller (PERIODICLISTBASE)

This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence.

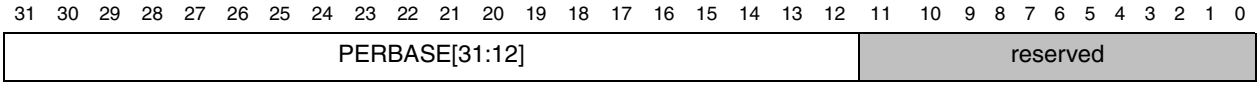


Figure 30-25. PERIODICLISTBASE - Host Controller Frame List Base Address

The register fields are described in the following table.

Table 30-26. PERIODICLISTBASE Register Field Descriptions

Field	Description
BASEADR	Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.
Reserved	Reserved. Must be written as zeros. During runtime, the values of these bits are undefined.

Device Controller (USB DEVICEADDR)

The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor.

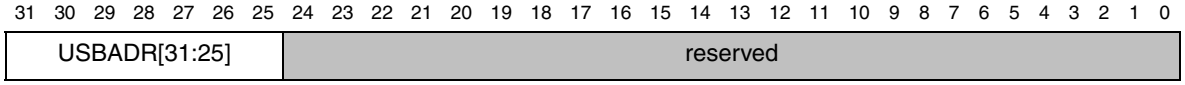


Figure 30-26. DEVICEADDR - Device Controller USB Device Address

The register fields are described in the following table.

Table 30-27. DEVICEADDR Field Descriptions

Field	Description
USBADR	Device Address. These bits correspond to the USB device address
reserved	Reserved. Must be written as zeros. During runtime, the values of these bits are undefined.

30.8.1.5.7 ASYNCLISTADDR; ENDPOINTLISTADDR

Address:Base + 158h

Default Value:00000000h

Attribute:Read/Write (Writes must be DWord Writes)

Size:32 bits

This register is shared between the host controller and the device controller operation.

Host Controller (ASYNCLISTADDR)

This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a zero when read.

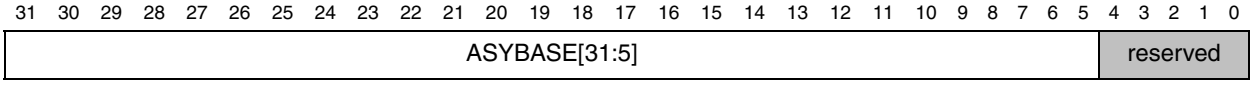


Figure 30-27. ASYNCLISTADDR - Host Controller Next Asynch. Address

The register fields are described in the following table.

Table 30-28. ASYNCLISTADDR Field Descriptions

Field	Description
ASYBASE[31:5]	Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.
Reserved	Reserved. These bits are reserved and their value has no effect on operation.

Device Controller (ENDPOINTLISTADDR)

In device mode, this register contains the address of the top of the endpoint list in system memory. Bits [10:0] of this register cannot be modified by the system software and will always return a zero when read.

The memory structure referenced by this physical memory pointer is assumed 64-byte.

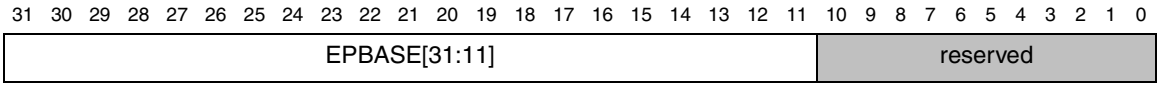


Figure 30-28. ENDPOINTLISTADDR - Device Controller Endpoint List Address

The register fields are described in the following table.

Table 30-29. ENDPOINTLISTADDR Field Descriptions

Field	Description
EPBASE[31:11]	Endpoint List Pointer(Low). These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Head (OH) (for example, one queue head per endpoint and direction).
Reserved	Reserved. These bits are reserved and their value has no effect on operation.

30.8.1.5.8 BURSTSIZE

Address:Base + 160h

Default Value:Implementation Dependent

Attribute:Read/Write (Writes must be DWord Writes)

Size:32 bits

This register is used to control dynamically change the burst size used during data movement on the initiator (master) interface.

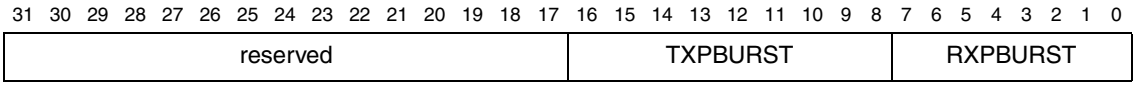


Figure 30-29. BURSTSIZE - Host Controller Embedded TT Async. Buffer Status

The register fields are described in the following table.

Table 30-30. BURSTSIZE Field Descriptions

Field	Description
Reserved	Reserved. These bits are reserved and their value has no effect on operation.
TXPBURST	Programmable TX Burst Length. (Read/Write) Default is the constant VUSB_HS_TX_BURST. This register represents the maximum length of a the burst in 32-bit words while moving data from system memory to the USB bus.
RXPBURST	Programmable RX Burst Length. (Read/Write) Default is the constant VUSB_HS_RX_BURST. This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory.

30.8.1.5.9 TXFILLTUNING

Address:Base + 164h

Default Value:00020000h

Attribute:Read/Write (Writes must be DWord Writes)

Size:32 bits

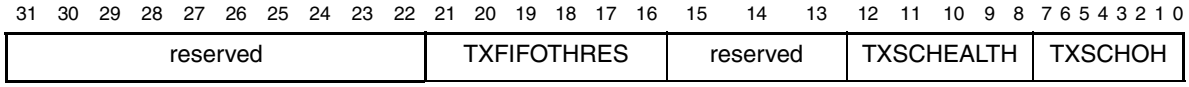


Figure 30-30. TXFILLTUNING

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Definitions:

T_0 = Standard packet overhead

T_1 = Time to send data payload

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

T_s = Total Packet Flight Time (send-only) packet

$T_s = T_0 + T_1$

T_p = Total Packet Time (fetch and send) packet

$T_p = T_{ff} + T_0 + T_1$

Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the [micro]frame is < T_s then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a mark will be made the scheduler health counter to note the occurrence of a “back-off” event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the TSCHHEALTH (T_{ff}) described below.

The register fields are described in the following table.

Table 30-31. TXFILLTUNING Field Descriptions

Field	Description
Reserved	Reserved. These bits are reserved and their value has no effect on operation.
TXFIFOTHRES	FIFO Burst Threshold. (Read/Write) [Default = 2] This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
TXSCHHEALTH	Scheduler Health Counter. (Read/Write To Clear) [Default = 0] This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31.
Reserved	Reserved. These bits are reserved and their value has no effect on operation.
TXSCHOH	Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode for OTG and SPH. The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode for OTG and SPH. The time unit represented in this register is always 1.267 the MPH product.

30.8.1.5.10 ULPI VIEWPORT (Optional)

Address:Base + 170h

Default Value:00000000h

Attribute:Read/Write

Size:32 bits

The register provides indirect access to the ULPI PHY register set. Although the core performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.

- CAUTION: Writes to the ULPI through the VIEWPORT can substantially harm standard USB operations. currently no usage model

has been defined where software should need to execute writes directly to the ULPI.

- NOTE: Executing read operations though the ULPI Viewport should have no harmful side effects to standard USB operations.
- Note: the ULPI Viewport is only synthesized in the design if the ULPI option has been purchased and installed and the constant `VUSB_HS_PHY_ULPI` is set to `1`. If the ULPI interface is not enabled, this register will always read zeros.

There are two operations that can be performed with the ULPI Viewport, wakeup and read /write operations. The wakeup operation is used to put the ULPI interface into normal operation mode and re-enable the clock if necessary. A wakeup operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or carkit mode. The ULPI state can be determined by reading the sync. state bit (ULPISS). If this bit is a one, then ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPISS indicates a ‘0’ then read/write operations will not be able execute. Undefined behavior will result if ULPISS = 0 and a read or write operation is performed. To execute a wakeup operation, write all 32-bits of the ULPI Viewport where ULPIPORT is constructed appropriately and the ULPIWU bit is a ‘1’ and ULPIRUN bit is a ‘0’. Poll the ULPI Viewport until ULPIWU is zero for the operation to complete.

To execute a read or write operation, write all 32-bits of the ULPI Viewport where ULPIDATWR, ULPIADDR, ULPIPORT, ULPIRW are constructed appropriately and the ULPIRUN bit is a ‘1’. Poll the ULPI Viewport until ULPIRUN is zero for the operation to complete. Once ULPIRUN is zero, the ULPIDATRD will be valid if the operation was a read.

The polling method above could also be replaced and interrupt driven using the ULPI interrupt defined in the USBSTS and USBINTR registers. When a wakeup or read/write operation complete, the ULPI interrupt will be set.

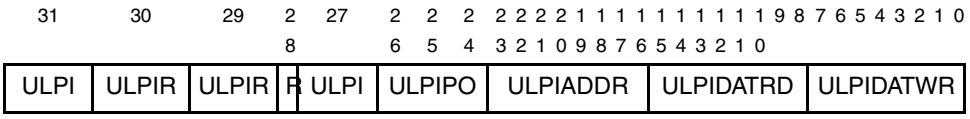


Figure 30-31. ULPI VIEWPORT

Table 30-32. ULPI VIEWPORT Field Descriptions

Field	Description
ULPIWU	ULPI Wakeup—Read/Write. Writing the '1' to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver can not set it back to '0'. Note: The driver must never execute a wakeup and a read/write operation at the same time.
ULPIRUN	ULPI Read/Write Run—Read/Write. Writing the '1' to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver can not set it back to '0'. Note: The driver must never execute a wakeup and a read/write operation at the same time.
ULPIRW	ULPI Read/Write Control—Read/Write. (0)—Read; (1)—Write. This bit selects between running a read or write operation.
reserved	Reserved. This bit is reserved and its value has no effect on operation.
ULPISS	ULPI Sync State—Read Only. (1)—Normal Sync. State. (0) In another state (that is, carkit, serial, low power) This bit represents the state of the ULPI interface. Before reading this bit, the ULPIPORT field should be set accordingly if used with the multi-port host. Otherwise, this field should always remain 0.
ULPIPORT	ULPI Port Number—Read/Write. For the wakeup or read/write operation to be executed, this value selects the port number the ULPI PHY is attached to in the multi-port host. The range is 0 to 7. This field should always be written as a 0 for the non-multi port products.
ULPIADDR	ULPI Data Address—Read/Write. When a read or write operation is commanded, the address of the operation is written to this field.
ULPIDATRD	ULPI Data Read—Read Only. After a read operation completes, the result is placed in this field.
ULPIDATWR	ULPI Data Write—Read/Write. When a write operation is commanded, the data to be sent is written to this field.

30.8.1.5.11 CONFIGFLAG

Address:Base + 180h

Default Value:00000001h

Attribute:Read Only

Size:32 bits

This register is not used in this implementation. A read from this register returns a constant of a 00000001h to indicate that all port routines default to this host controller.

30.8.1.5.12 PORTSCx

Address:Base + 184h + (4*(Port Number –1))

where Port Number =1,2,3,…8

Default Value:

IIII0000000000000000XX0000000000b (host mode)

IIII0000000000000001XX0000000100b (device mode)

I = Implementation dependent

X = Unknown

Attribute:RO, Read/Write, R/WC (field dependent)

Size:32 bits

Host Controller

A host controller must implement one to eight port registers. The number of port registers implemented by a particular instantiation of a host controller is documented in the HCSPARAMs register. Software uses this information as an input parameter to determine how many ports need service.

This register is only reset when power is initially applied or in response to a controller reset. The initial conditions of a port are:

No device connected

Port disabled

If the port has port power control, this state remains until software applies power to the port by setting port power to one.

Device Controller

A device controller must implement only port register one and it does not support power control. Port control in device mode is only used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling and allows software to put the PHY into low power suspend mode and disable the PHY clock.

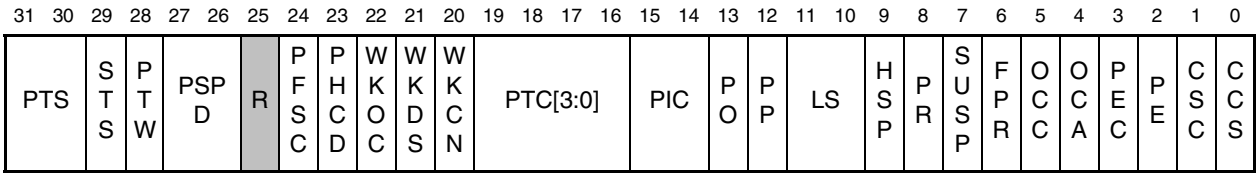


Figure 30-32. PORTSCx - Port Status Control[1:8]

The register fields are described in the following table.

Table 30-33. PORTSCx Field Descriptions

Field	Description
PTS	Parallel Transceiver Select—Read/Write. This register bit pair is used in conjunction with the configuration constant VUSB_HS_PHY_TYPE to control which parallel transceiver interface is selected. If VUSB_HS_PHY_TYPE is set for 0,1,2, or 3 then this bit is read only. If VUSB_HS_PHY_TYPE is 3,4,5, or 6 then this bit is read/write. This field is reset to: “00” if VUSB_HS_PHY_TYPE = 0,4—UTMI/UTMI+ “01” if VUSB_HS_PHY_TYPE = 1,5—Reserved “10” if VUSB_HS_PHY_TYPE = 2,6—ULPI “11” if VUSB_HS_PHY_TYPE = 3,7—Serial/1.1 PHY (FS Only) This bit is not defined in the EHCI specification.
STS	Serial Transceiver Select—Read/Write. This register bit is used in conjunction with the configuration constant VUSB_HS_PHY_SERIAL to control whether the parallel or serial transceiver interface is selected for FS and LS operation. The Serial Interface Engine can be used in combination with the UTMI+ or ULPI physical interface to provide FS/LS signaling instead of the parallel interface. If VUSB_HS_PHY_SERIAL is set for 0 or 1 then this bit is read only. If VUSB_HS_PHY_SERIAL is 3 or 4 then this bit is read/write. This bit has no effect unless Parallel Transceiver Select is set to UTMI+ or ULPI. The Serial/1.1 physical interface will use the Serial Interface Engine for FS/LS signaling regardless of this bit value. Note: This bit is reserved for future operation and is a placeholder adding dynamic use of the serial engine in accord with UMTI+ and ULPI characterization logic. This bit is not defined in the EHCI specification.
PTW	Parallel Transceiver Width—Read/Write. This register bit is used in conjunction with the configuration constant VUSB_HS_PHY8_16 to control whether the data bus width of the UTMI transceiver interface. If VUSB_HS_PHY8_16 is set for 0 or 1 then this bit is read only. If VUSB_HS_PHY8_16 is 2 or 3 then this bit is read/write. This bit is reset to 1 if VUSB_HS_PHY8_16 selects a default UTMI interface width of 16-bits else it is reset to 0. Writing this bit to 0 selects the 8-bit [60MHz] UTMI interface. Writing this bit to 1 selects the 16-bit [30MHz] UTMI interface. This bit has no effect if the Serial interfaces are selected. This bit is not defined in the EHCI specification.
PSPD	Port Speed—Read Only. This register field indicates the speed at which the port is operating. For HS mode operation in the host controller and HS/FS operation in the device controller the port routing steers data to the Protocol engine. For FS and LS mode operation in the host controller, the port routing steers data to the Protocol Engine w/ Embedded Transaction Translator. 00—Full Speed 01—Low Speed 10—High Speed This bit is not defined in the EHCI specification.
PFSC	Port Force Full Speed Connect—Read/Write. Default = 0b. Writing this bit to a 1b will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with a HS host, hub or device. This bit is not defined in the EHCI specification. This bit is for debugging purposes.

Table 30-33. PORTSCx Field Descriptions (continued)

Field	Description
PHCD	PHY Low Power Suspend - Clock Disable (PLPSCD)—Read/Write. Default = 0b. Writing this bit to a 1b will disable the PHY clock. Writing a 0b enables it. Reading this bit will indicate the status of the PHY clock. NOTE: The PHY clock cannot be disabled if it is being used as the system clock. In device mode, The PHY can be put into Low Power Suspend—Clock Disable when the device is not running (USBCMD Run/Stop=0b) or the host has signaled suspend (PORTSC SUSPEND=1b). Low power suspend will be cleared automatically when the host has signaled resume if using a circuit similar to that in. Before forcing a resume from the device, the device controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend—Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software. See for more discussion on clock disable and power down issues. This bit is not defined in the EHCI specification.
WKOC	Wake on Over-current Enable (WKOC_E) — Read/Write. Default = 0b. Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if *Port Power(PP)* is zero. This bit is output from the controller as signal pwrctl_wake_ovrcurr_en (OTG/host core only) for use by an external power control circuit.
WKDC	Wake on Disconnect Enable (WKDSCNNT_E) — Read/Write. Default=0b. Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if *Port Power(PP)* is zero or in device mode. This bit is output from the controller as signal pwrctl_wake_dscnnt_en (OTG/host core only) for use by an external power control circuit.
WKCN	Wake on Connect Enable (WKCNNT_E) — Read/Write. Default=0b. Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if *Port Power(PP)* is zero or in device mode. This bit is output from the controller as signal pwrctl_wake_dscnnt_en (OTG/host core only) for use by an external power control circuit.
PTC[3:0]	Port Test Control — Read/Write. Default = 0000b. Any other value than zero indicates that the port is operating in test mode. Value Specific Test 0000b TEST_MODE_DISABLE 0001b J_ STATE 0010b K_STATE 0011b SE0 (host)/NAK (device) 0100b Packet 0101b FORCE_ENABLE_HS 0110b FORCE_ENABLE_FS 0111b FORCE_ENABLE_LS 1000b to 1111bReserved Refer to the *USB Specification Revision 2.0 Universal Serial Bus Specification, Revision 2.0,* April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org for details on each test mode. The FORCE_ENABLE_FS and FORCE ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point. **Note:** Low-speed operations are not supported as a peripheral device.

Table 30-33. PORTSCx Field Descriptions (continued)

Field	Description
PIC[1:0]	Port Indicator Control — Read/Write. Default = Ob. Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is a zero. If P_INDICATOR bit is a one, then the bit is: Bit ValueMeaning 00b Port indicators are off 01b Amber 10b Green 11b Undefined Refer to the USB Specification Revision 2.0 Universal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org for a description on how these bits are to be used. This field is output from the controller as signals port_ind_ctl_1 and port_ind_ctl_0 for use by an external led driving circuit.
PO	Port Owner—Read/Write. Default = 0. This bit unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is zero System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that an internal companion controller owns and controls the port. Port owner handoff is not implemented in this design, therefore this bit will always be 0.
PP	Port Power (PP)—Read/Write or Read Only. The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0b 0b Read Only— A device controller with no OTG capability does not have port power control switches. 1b 1b/0b—RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (that is, PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitional by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in the host/OTG controller (PPC = 1). In a device only implementation port power control is not necessary, thus PPC and PP = 0.
LS[1:0]	Line Status—Read Only. These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The encoding of the bits are: Bits [11:10]Meaning 00b SE0 10b J-state 01b K-state 11b Undefined In host mode, the use of linestate by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of linestate by the device controller driver is not necessary.
HSP	High-Speed Port — Read Only. Default = 0b. When the bit is one, the host/device connected to the port is in high-speed mode and if set to zero, the host/device connected to the port is not in a high-speed mode. Note: HSP is redundant with PSPD(27:26) but will remain in the design for compatibility. This bit is not defined in the EHCI specification.

Table 30-33. PORTSCx Field Descriptions (continued)

Field	Description
PR	Port Reset This field is zero if *Port Power(PP)* is zero. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. Default 0. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. *This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver.* In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register.
SUSP	Suspend In Host Mode: Read/Write. 1=Port in suspend state. 0=Port not in suspend state. Default=0. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend]Port State 0x Disable 10 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the *Force Port Resume* bit to zero. The host controller ignores a write of zero to this bit. If host software sets this bit to a one when the port is not enabled (that is, *Port enabled* bit is a zero) the results are undefined. This field is zero if *Port Power(PP)* is zero in host mode. In Device Mode: Read Only. 1=Port in suspend state. 0=Port not in suspend state. Default=0. In device mode this bit is a read only status bit.
FPR	Force Port Resume —Read/Write. 1= Resume detected/driven on port. 0=No resume (K-state) detected/driven on port. Default = 0. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the *Port Change Detect* bit in the USBSTS register is also set to one. *This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver.* Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit will remain a one until the port has switched to the high-speed idle. Writing a zero has no affect because the port controller will time the resume operation clear the bit the port control state switches to HS or FS idle. This field is zero if *Port Power(PP)* is zero in host mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition detected, the *Port Change Detect* bit in the USBSTS register is also set to one.

Table 30-33. PORTSCx Field Descriptions (continued)

Field	Description
OCC	Over-current Change—R/WC. Default=0. 1=This bit gets set to one when there is a change to Over-current Active. Software clears this bit by writing a one to this bit position. For host/OTG implementations the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device-only implementations this bit shall always be 0.
OCA	Over-current Active—Read Only. Default 0. 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 one to zero when the over current condition is removed. For host/OTG implementations the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device-only implementations this bit shall always be 0.
PEC	Port Enable/Disable Change—R/WC. 1=Port enabled/disabled status has changed. 0=No change. Default = 0. In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point. Software clears this by writing a one to it. This field is zero if *Port Power(PP)* is zero. In Device mode: The device port is always enabled. (This bit will be zero)
PE	Port Enabled/Disabled—Read/Write. 1=Enable. 0=Disable. Default 0. In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if *Port Power(PP)* is zero in host mode. In Device Mode: The device port is always enabled. (This bit will be one)
CSC	Connect Status Change—R/WC. 1 =Change in Current Connect Status. 0=No change. Default 0. In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (that is, the bit will remain set). Software clears this bit by writing a one to it. This field is zero if *Port Power(PP)* is zero in host mode. In Device Mode: This bit is undefined in device controller mode.
CCS	Current Connect Status—Read Only. In Host Mode: 1=Device is present on port. 0=No device is present. Default = 0. This value reflects the current state of the port, and may not correspond directly to the event that caused the *Connect Status Change* bit (Bit 1) to be set. This field is zero if *Port Power(PP)* is zero in host mode. In Device Mode: 1=Attached. 0=Not Attached. Default=0. A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended.

30.8.1.5.13 OTGSC

Address:Base + 1A4h

Default Value:00000020h

Attribute:RO, Read/Write, R/WC (field dependent)

Size:32 bits

Host Controller

A host controller implements one On-The-Go (OTG) Status and Control register corresponding to Port 0 of the host controller.

The OTGSC register has four sections

OTG Interrupt enables (Read/Write)

OTG Interrupt status (Read/Write to Clear)

OTG Status inputs (Read Only)

OTG Controls(Read/Write)

The status inputs are debounced using a 1Msec time constant. Values on the status inputs that do not persist for more than 1Msec will not cause an update of the status input register, or cause an OTG interrupt.

See also USBMODE register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DPIE	1msE	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	R	DPIS	1mss	BSEIS	BSVIS	ASVIS	AVVIS	IDIS	R	DPS	1mst	BSE	BSV	ASV	AVV	ID	R		IDPU	DP	OT	R	VC	VD

Figure 30-33. OTGSC—OTG Status Control

The register fields are described in the following table.

Table 30-34. OTGSC Field Descriptions

Field	Description
DPIE	Data Pulse Interrupt Enable
1msE	1 millisecond timer Interrupt Enable—Read/Write
BSEIE	B Session End Interrupt Enable—Read/Write. Setting this bit enables the B session end interrupt.
BSVIE	B Session Valid Interrupt Enable—Read/Write. Setting this bit enables the B session valid interrupt.
ASVIE	A Session Valid Interrupt Enable—Read/Write. Setting this bit enables the A session valid interrupt.

Table 30-34. OTGSC Field Descriptions (continued)

Field	Description
AVVIE	A VBus Valid Interrupt Enable—Read/Write. Setting this bit enables the A VBus valid interrupt.
IDIE	USB ID Interrupt Enable—Read/Write. Setting this bit enables the USB ID interrupt.
DPIS	Data Pulse Interrupt Status—Read/Write to Clear. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0).PortPower = Off (0). Software must write a one to clear this bit.
1msS	1 millisecond timer Interrupt Status—Read/Write to Clear. This bit is set once every millisecond. Software must write a one to clear this bit.
BSEIS	B Session End Interrupt Status—Read/Write to Clear. This bit is set when VBus has fallen below the B session end threshold. Software must write a one to clear this bit
BSVIS	B Session Valid Interrupt Status—Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software must write a one to clear this bit.
ASVIS	A Session Valid Interrupt Status—Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software must write a one to clear this bit.
AVVIS	A VBus Valid Interrupt Status—Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software must write a one to clear this bit.
IDIS	USB ID Interrupt Status—Read/Write. This bit is set when a change on the ID input has been detected. Software must write a one to clear this bit.
DPS	Data Bus Pulsing Status—Read Only. A ‘1’ indicates data bus pulsing is being detected on the port.
1msT	1 millisecond timer toggle - Read Only. This bit toggles once per millisecond.
BSE	B Session End—Read Only. Indicates VBus is below the B session end threshold.
BSV	B Session Valid—Read Only. Indicates VBus is above the B session valid threshold.
ASV	A Session Valid—Read Only. Indicates VBus is above the A session valid threshold.
AVV	A VBus Valid—Read Only. Indicates VBus is above the A VBus valid threshold.
ID	USB ID—Read Only. 0 = A device, 1 = B device

Table 30-34. OTGSC Field Descriptions (continued)

Field	Description
IDPU	ID Pullup—Read/Write This bit provide control over the ID pull-up resister; 0 = off, 1 = on [default]. When this bit is 0, the ID input will not be sampled.
DP	Data Pulsing—Read/Write. Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
OT	OTG Termination—Read/Write. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM.
VC	VBUS Charge—Read/Write. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
VD	VBUS_Discharge—Read/Write. Setting this bit causes VBus to discharge through a resistor.

30.8.1.5.14 USBMODE

Address:Base + 1A8h

Default Value:00000000h (otg implementation—mode not selected)

00000003h (host mode)

00000002h (device mode))

Attribute:R/WO, Read Only

Size:32 bits

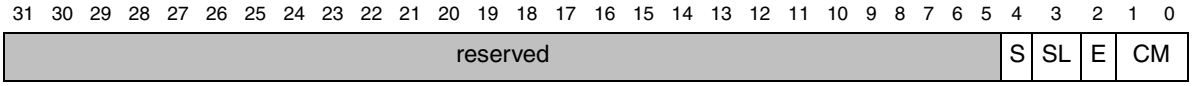


Figure 30-34. USBMODE -USB Device Mode

The register fields are described in the following table.

Field	Description
reserved	Reserved. These bits are reserved and should be set to zero.
SDIS	Stream Disable Mode. (0—Inactive [default]; 1—Active) Device Mode: Setting to a '1' disables double priming on both RX and TX for low bandwidth systems. This mode ensures that when the RX and TX buffers are sufficient to contain an entire packet that the standard double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note: In High Speed Mode, all packets received will be responded to with a NYET handshake when stream disable is active. Host Mode: Setting to a '1' ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB. Note: Time duration to pre-fill the FIFO becomes significant when stream disable is active. See TXFILLTUNING and TXTTFILLTUNING [MPH Only] to characterize the adjustments needed for the scheduler when using this feature. Note: The use of this feature substantially limits of the overall USB performance that can be achieved.
SLOM	Setup Lockout Mode. In device mode, this bit controls behavior of the setup lock mechanism. See Control Endpoint Operation Model. 0—Setup Lockouts On (default); 1—Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in USBCMD)
ES	Endian Select—Read/Write. This bit can change the byte alignment of the transfer buffers to match the host microprocessor. The bit fields in the microprocessor interface and the data structures are unaffected by the value of this bit because they are based upon the 32-bit word. Bit Meaning 0 Little Endian [Default] 1 Big Endian
CM[1:0]	Controller Mode—R/WO. Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host and device capability, the controller will default to an idle state and will need to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the *RESET* bit in the USBCMD register before reprogramming this register. Bit Meaning 00 Idle [Default for combination host/device] 01 Reserved 10 Device Controller [Default for device only controller] 11 Host Controller [Default for host only controller]

30.8.1.5.15 ENDPTSETUPSTAT

Address:Base + 1ACh

Default Value:00000000h

Attribute:R/WC

Size:32 bits

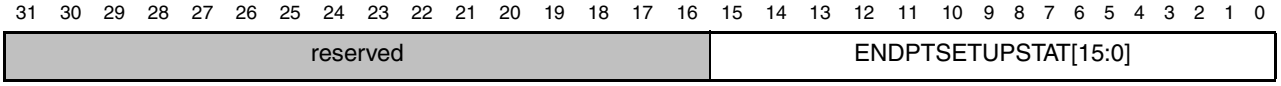


Figure 30-35. ENDPTSETUPSTAT—Endpoint Setup Status

The register fields are described in the following table.

Field	Description
reserved	Reserved. These bits are reserved and should be set to zero.
ENDPTSETUPSTAT [15:0]	Setup Endpoint Status. For every setup transaction that is received, a corresponding bit in this register is set to one. Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read the setup data from Queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lock our mechanism is engaged. See Managing Endpoints in the Device Operational Model. This register is only used in device mode.

30.8.1.5.16 ENDPTPRIME

Address:Base + 1B0h

Default Value:00000000h

Attribute:R/WS

Size:32 bits

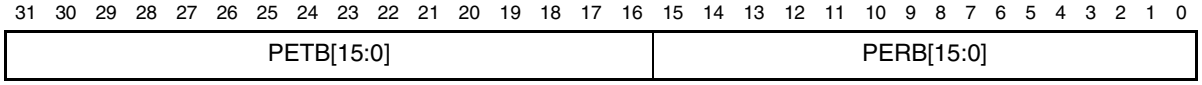


Figure 30-36. ENDPTPRIME—Endpoint Initialization

This register is only used in device mode. The register fields are described in the following table.

Field	Description
PETB [15:0]	Prime Endpoint Transmit Buffer—R/WS. For each endpoint a corresponding bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed. Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PETB[15] – Endpoint #15 PETB[1] – Endpoint #1 PETB[0] – Endpoint #0
PERB [15:0]	Prime Endpoint Receive Buffer—R/WS. For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed. Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. Bit 15 – Endpoint #15 Bit 1 – Endpoint #1 Bit 0 – Endpoint #0

30.8.1.5.17 ENDPTFLUSH

Address:Base + 1B4h

Default Value:00000000h

Attribute:R/WS

Size:32 bits

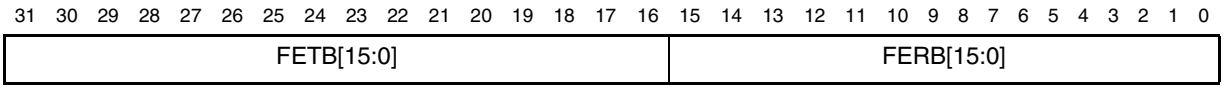


Figure 30-37. ENDPTFLUSH—Endpoint De-Initialize

This register is only used in device mode. The register fields are described in the following table.

Field	Description
FETB [15:0]	Flush Endpoint Transmit Buffer—R/WS. Writing a one to a bit(s) in this register will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful. FETB[15] – Endpoint #15 FETB[1] – Endpoint #1 FETB[0] – Endpoint #0
FERB [15:0]	Flush Endpoint Receive Buffer—R/WS. Writing a one to a bit(s) will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful. Bit 15 – Endpoint #15 Bit 1 – Endpoint #1 Bit 0 – Endpoint #0

30.8.1.5.18 ENDPTSTAT

Address:Base + 1B8h

Default Value:00000000h

Attribute:Read Only

Size:32 bits

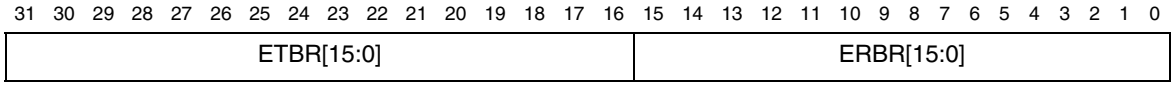


Figure 30-38. ENDPTSTAT—Endpoint Status

This register is only used in device mode. The register fields are described in the following table.

Field	Description
ETBR [15:0]	Endpoint Transmit Buffer Ready—Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ETBR[15]– Endpoint #15 ETBR[1]– Endpoint #1 ETBR[0]– Endpoint #0
ERBR [15:0]	Endpoint Receive Buffer Ready—Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ERBR[15]– Endpoint #15 ERBR[1]– Endpoint #1 ERBR[0]– Endpoint #0

30.8.1.5.19 ENDPTCOMPLETE

Address:Base + 1BCh

Default Value:00000000h

Attribute:R/WC

Size:32 bits

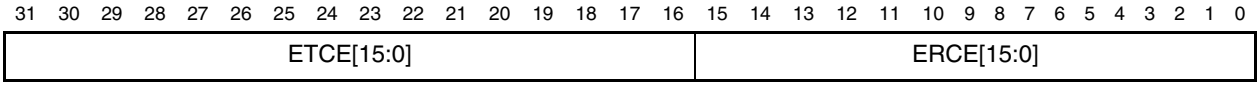


Figure 30-39. ENDPTCOMPLETE—Endpoint Compete

This register is only used in device mode. The register fields are described in the following table.

Field	Description
ETCE [15:0]	Endpoint Transmit Complete Event—R/WC. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the *USBINT*. Writing a one will clear the corresponding bit in this register. ETCE[15]– Endpoint #15 ETCE[1]– Endpoint #1 ETCE[0]– Endpoint #0
ERCE [15:0]	Endpoint Receive Complete Event—RW/C. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the *USBINT*. Writing a one will clear the corresponding bit in this register. ERCE[15]– Endpoint #15 ERCE[1]– Endpoint #1 ERCE[0]– Endpoint #0

30.8.1.5.20 ENDPTCTRL0

Address:Base + 1C0h

Default Value:0080008h

Attribute:Read Only, Read/Write, R/WC (field dependent)

Size:32 bits

Every Device will implement Endpoint0 as a control endpoint.

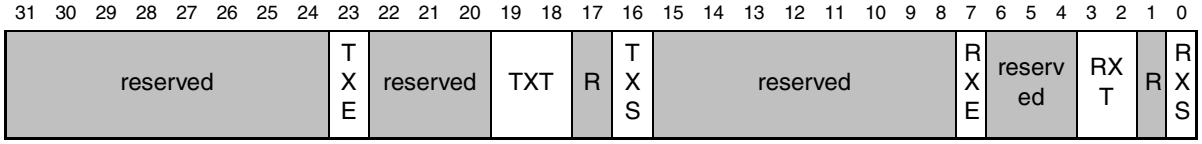


Figure 30-40. ENDPTCTRL0—Endpoint Control 0

The register fields are described in the following table.

Field	Description
reserved	Reserved. These bits are reserved and should be set to zero.
TXE	TX Endpoint Enable 1—Enabled Endpoint0 is always enabled.

Field	Description
TXT[1:0]	TX Endpoint Type—Read/Write 00—Control Endpoint0 is fixed as a Control End Point.
R	Reserved. Bit reserved and should be set to zero.
TXS	TX Endpoint Stall—Read/Write 0—End Point OK [Default] 1—End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request.
RXE	RX Endpoint Enable 1—Enabled Endpoint0 is always enabled.
RXT[1:0]	RX Endpoint Type—Read/Write 00—Control Endpoint0 is fixed as a Control End Point.
RXS	RX Endpoint Stall—Read/Write 0—End Point OK. [Default] 1—End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request.

30.8.1.5.21 ENDPTCTRL1—ENDPTCTRL15

Address:Base + 1C0h+(4*(EndPoint Number))

Default Value:00000000h

Attribute:Read/Write

Size:32 bits

There is an ENDPTCTRLx register for each endpoint in a device.

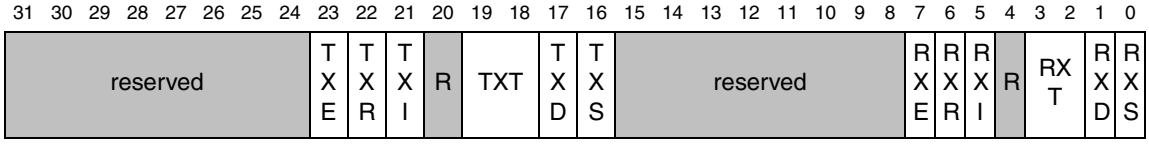


Figure 30-41. ENDPTCTRL1—ENDPTCTRL15—Endpoint Control 1 to 15

CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (IE. Bulk-type). leaving an unconfigured endpoint control will cause undefined behavior for the data pid tracking on the active endpoint/direction.

The register fields are described in the following table.

Field	Description
Reserved	Reserved. These bits are reserved and should be set to zero.
TXE	TX Endpoint Enable 0—Disabled [Default] 1—Enabled An Endpoint should be enabled only after it has been configured.
TXR	TX Data Toggle Reset (WS) Write 1—Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
TXI	TX Data Toggle Inhibit 0—PID Sequencing Enabled. [Default] 1—PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
R	Reserved. Bit reserved and should be set to zero.
TXT[1:0]	TX Endpoint Type—Read/Write 00—Control 01—Isochronous 10—Bulk 11—Interrupt
TXD	TX Endpoint Data Source—Read/Write 0—Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
TXS	TX Endpoint Stall—Read/Write 0—End Point OK 1—End Point Stalled This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above.
RXE	RX Endpoint Enable 0—Disabled [Default] 1—Enabled An Endpoint should be enabled only after it has been configured.

Field	Description
RXR	RX Data Toggle Reset (WS) Write 1—Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
RXT[1:0]	RX Endpoint Type—Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved
RXD	RX Endpoint Data Sink—Read/Write—TBD 0—Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
RXS	RX Endpoint Stall—Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above,

30.8.1.6 OTG Operations

30.8.1.6.1 Register Bits

In the previous section, the Register interface has behaviors described for device mode and behaviors described for host mode. However, during OTG operations it is necessary to perform tasks independent of the controller mode.

Note also from ENDPTCTRL1—ENDPTCTRL15—Endpoint Control 1 to 1542 that the only way to transition the controller mode out of host or device mode is with the controller reset bit. Therefore, it is also necessary for the OTG tasks to be performed independent of a controller reset as well as independent of the controller mode.

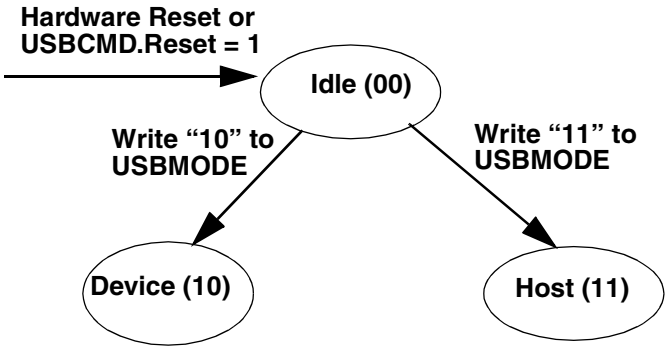


Figure 30-42. Controller Mode

To this end, the listed below are the register bits that are used for OTG operations, which are independent of the controller mode and are also not affected by a write to the reset bit in the USBCMD register:

All Identification Registers

All Device/Host Capability Registers

OTGSC: All bits

PORTSC:

Physical Interface Select
Physical Interface Serial Select
Physical Interface Data Width
Physical Interface Low Power
Physical Interface Wake Signals
Port Indicators
Port Power

30.8.1.6.2 Hardware Assist

The hardware assist provides automated response and sequencing that may not be possible to software with significant interrupt latency response times. The use of this additional circuitry is optional and can be used to assist the 3 sequences below.

Auto-Reset

When the HAAR is set to one, the host will automatically start a reset after a connect event. This shortcuts the normal process where software is notified of the connect event and starts the reset. Software will still receive notification of the connect event but should not write the reset bit when the HAAR is set. Software will be notified again after the reset is complete via the enable change bit in the PORTSC register which cause a port change interrupt.

This assist will ensure the OTG parameter TB_ACON_BSE0_MAX = 1ms is met.

Data-Pulse

Writing a one to HADP will start a data pulse of approximately 7ms in duration and then automatically cease the data pulsing. During the data pulse, the DP will be set and then cleared. This automation relieves software from accurately controlling the data-pulse duration. During the data pulse, the HCD can poll to see that the HADP and DP bit have returned low to recognize the completion or simply launch the data pulse and wait to see if a VBUS Valid interrupt occurs when the A-side supplies bus power.

This assist will ensure data pulsing meets the OTG requirement of > 5ms and < 10ms.

B-Disconnect to A-Connect

During HNP, the B-disconnect occurs from the OTG A_suspend state and within 3 ms, the device must enable the pullup on the DP leg in the A-peripheral state. When HABA is set, the Host Controller port is in suspend mode, and the device disconnects, then this hardware assist begins.

1. Reset the OTG core.
2. Write the OTG core into device mode.
3. Write the device run bit to a 1 and enable necessary interrupts including:
    - USB Reset Enable (URE); enables interrupt on USB bus reset to device
    - Sleep Enable (SLE); enables interrupt on device suspend
    - Port Change Detect Enable (PCE); enables interrupt on device connect

When software has enabled this hardware assist, it must not interfere during the transition and should not write any register in the core until it gets an interrupt from the device controller signifying that a reset interrupt has occurred or at least first verify that the core has entered device mode. HCD/DCD must not activate the core soft reset at any time since this action is performed by hardware. During the transition, the software may see an interrupt from the disconnect and/or other spurious interrupts (i.e. SOF/etc.) that may or may not cascade and my be cleared by the soft reset depending on the software response time.

After the core has entered device mode by the hardware assist, the DCD must ensure that the ENDPTLISTADDR is programmed properly before the host sends a setup packet. Since the end of the reset duration, which may be initiated quickly (a few microseconds) after connect, will require at a minimum 50 ms, this is the time for which the DCD must be ready to accept setup packets after having received notification that the reset has been detected or simply that the OTG is in device mode which ever occurs first.

In the case where the A-peripheral fails to see a reset after the controller enters device mode and engages the DP-pullup, the device controller interrupt the DCD signifying that a suspend has occurred.

This assist will ensure the parameter TA_BDIS_ACON_MAX = 3ms is met.

30.8.2 Host Data Structures

This section defines the interface data structures used to communicate control, status, and data between HCD (software) and the Enhanced Host Controller (hardware). The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of a Periodic Schedule,

Periodic Frame List, Asynchronous Schedule, Isochronous Transaction Descriptors, Split-transaction Isochronous Transfer Descriptors, Queue Heads, and Queue Element Transfer Descriptors.

The periodic frame list is the root of all periodic (isochronous and interrupt transfer type) support for the host controller interface. The asynchronous list is the root for all the bulk and control transfer type support. Isochronous data streams are managed using Isochronous Transaction Descriptors. Isochronous split-transaction data streams are managed with Split-transaction Isochronous Transfer Descriptors. All Interrupt, Control, and Bulk data streams are managed via queue heads and Queue Element Transfer Descriptors. These data structures are optimized to reduce the total memory footprint of the schedule and to reduce (on average) the number of memory accesses needed to execute a USB transaction.

Note that software must ensure that no interface data structure reachable by the EHCI host controller spans a 4K-page boundary.

The data structures defined in this section are (from the host controller's perspective) a mix of read-only and read/writeable fields. The host controller must preserve the read-only fields on all data structure writes.

30.8.2.1 Periodic Frame List

This schedule is for all periodic transfers (isochronous and interrupt). The periodic schedule is referenced from the operational registers space using the *PERIODICLISTBASE* address register and the *FRINDEX* register. The periodic schedule is based on an array of pointers called the Periodic Frame List. The *PERIODICLISTBASE* address register is combined with the *FRINDEX* register to produce a memory pointer into the frame list. The Periodic Frame List implements a *sliding window* of work over time.

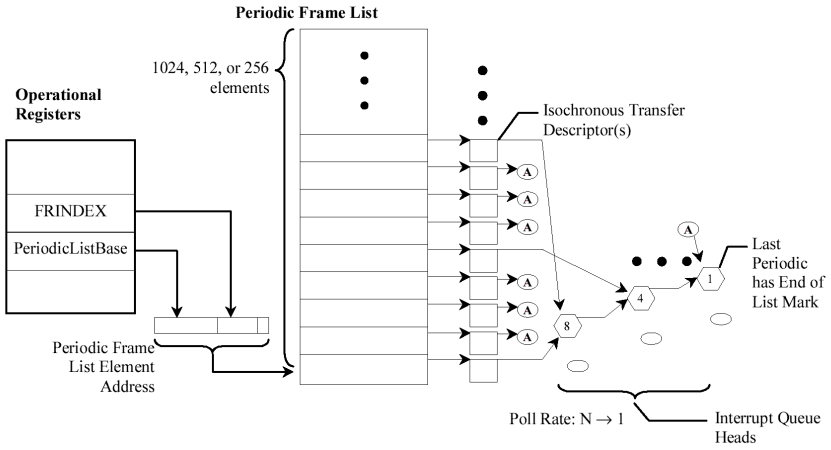


Figure 30-43. Periodic Schedule Organization

[1] Split transaction Interrupt, Bulk and Control are also managed using queue heads and queue element transfer descriptors.

The periodic frame list is a 4K-page aligned array of Frame List Link pointers. The length of the frame list may be programmable. The programmability of the periodic frame list is exported to system software via the HCCPARAMS register. If non-programmable, the length is 1024 elements. If programmable, the length can be selected by system software as one of 256, 512, or 1024 elements. An implementation must

support all three sizes. Programming the size (that is, the number of elements) is accomplished by system software writing the appropriate value into *Frame List Size* field in the USBCMD register.

Frame List Link pointers direct the host controller to the first work item in the frame's periodic schedule for the current micro-frame. The link pointers are aligned on DWord boundaries within the Frame List.

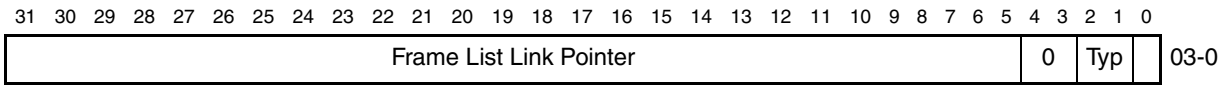


Figure 30-44. Format of Frame List Element Pointer

Frame List Link pointers always reference memory objects that are 32-byte aligned. The referenced object may be an isochronous transfer descriptor for high-speed devices, a split-transaction isochronous transfer descriptor (for full-speed isochronous endpoints), or a queue head (used to support high-, full- and low-speed interrupt). System software should not place non-periodic schedule items into the periodic schedule. The least significant bits in a frame list pointer are used to key the host controller as to the type of object the pointer is referencing.

The least significant bit is the *T-Bit* (bit 0). When this bit is set to a one, the host controller will never use the value of the frame list pointer as a physical memory pointer. The *Typ* field is used to indicate the exact type of data structure being referenced by this pointer. The value encodings are:

Table 30-35. Typ Field Value Definitions

Value	Meaning
00b	Isochronous Transfer Descriptor
01b	Queue Head
10b	Split Transaction Isochronous Transfer Descriptor.
11b	Frame Span Traversal Node.

30.8.2.2 Asynchronous List Queue Head Pointer

The Asynchronous Transfer List (based at the ASYNCLISTADDR register) is where all the control and bulk transfers are managed. Host controllers use this list only when it reaches the end of the periodic list, the periodic list is disabled, or the periodic list is empty.

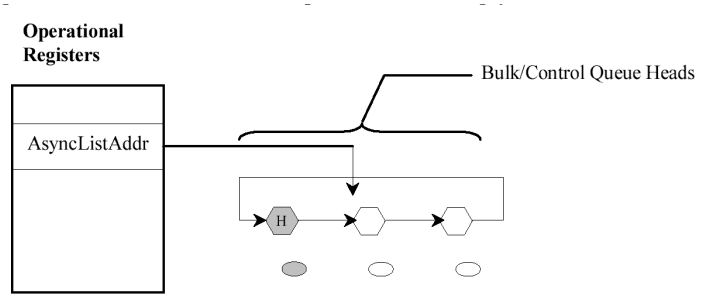


Figure 30-45. Asynchronous Schedule Organization

The Asynchronous list is a simple circular list of queue heads. The ASYNCLISTADDR register is simply pointer to the *next* queue head. This implements a pure round-robin service for all queue heads linked into the asynchronous list.

30.8.2.3 Isochronous (High-Speed) Transfer Descriptor (iTD)

The format of an isochronous transfer descriptor is illustrated in Table 30-36. This structure is used only for high-speed isochronous endpoints. All other transfer types should use queue structures. Isochronous TDs must be aligned on a 32-byte boundary.

Table 30-36. Isochronous Transfer Descriptor

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Next Link Pointer	0 Typ T	03-0
Status	Transaction 0 Length io PG* Transaction 0 Offset*	07-0
Status	Transaction 1 Length io PG* Transaction 1 Offset*	0B-0
Status	Transaction 2 Length io PG* Transaction 2 Offset*	0F-0
Status	Transaction 3 Length io PG* Transaction 3 Offset*	13-1
Status	Transaction 4 Length io PG* Transaction 4 Offset*	17-1
Status	Transaction 5 Length io PG* Transaction 5 Offset*	1B-1
Status	Transaction 6 Length io PG* Transaction 6 Offset*	1F-1
Status	Transaction 7 Length io PG* Transaction 7 Offset*	23-2
Buffer Pointer (Page 0)	EndPt R Device Address	27-2
Buffer Pointer (Page 1)	I/ Maximum Packet Size	2B-2
Buffer Pointer (Page 2)	Reserved Mult	2F-2
Buffer Pointer (Page 3)	Reserved	33-3

Table 30-36. Isochronous Transfer Descriptor (continued)

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		
Buffer Pointer (Page 4)	Reserved	37-3
Buffer Pointer (Page 5)	Reserved	3B-3
Buffer Pointer (Page 6)	Reserved	3F-3

	Host Controller Read/Write		Host Controller Read Only.

Note: *Note: these fields may be modified by the host controller if the I/O field indicates an OUT.

30.8.2.3.1 Next Link Pointer

The first DWord of an iTD is a pointer to the next schedule data structure.

Table 30-37. Next Schedule Element Pointer

Bit	Description
31:5	Link Pointer (LP). These bits correspond to memory address signals [31:5], respectively. This field points to another Isochronous Transaction Descriptor (iTD/siTD) or Queue Head (QH).
4:3	Reserved. These bits are reserved and their value has no effect on operation. Software should initialize this field to zero.
2:1	QH/(s)iTD Select (Typ). This field indicates to the Host Controller whether the item referenced is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: ValueMeaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0	Terminate (T). 1= Link Pointer field is not valid. 0= Link Pointer field is valid.

30.8.2.3.2 iTD Transaction Status and Control List

DWords 1 through 8 are eight slots of transaction control and status. Each transaction description includes:

- Status results field
- Transaction length (bytes to send for OUT transactions and bytes received for IN transactions).
- Buffer offset. The *PG* and *Transaction X Offset* fields are used with the buffer pointer list to construct the starting buffer address for the transaction.

The host controller uses the information in each transaction description plus the endpoint information contained in the first three DWords of the Buffer Page Pointer list, to execute a transaction on the USB.

Table 30-38. iTD Transaction Status and Control

Bit	Description	
31:28	Status. This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:	
	Bit	**Definition**
	31	Active. Set to one by software to enable the execution of an isochronous transaction by the Host Controller. When the transaction associated with this descriptor is completed, the Host Controller sets this bit to zero indicating that a transaction for this element should not be executed when it is next encountered in the schedule.
	30	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, no action is necessary.
	29	Babble Detected. Set to one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.
	28	Transaction Error (XactErr). Set to one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit may only be set for isochronous IN transactions.
27:16	Transaction X Length. For an OUT, this field is the number of data bytes the host controller will send during the transaction. The host controller is not required to update this field to reflect the actual number of bytes transferred during the transfer. For an IN, the initial value of the endpoint to deliver. During the status update, the host controller writes back the field is the number of bytes the host expects the number of bytes successfully received. The value in this register is the actual byte count (for example, 0‡zero length data, 1‡one byte, 2‡two bytes, etc.). The maximum value this field may contain is 0xC00 (3072).	
15	Interrupt On Complete (IOC). If this bit is set to one, it specifies that when this transaction completes, the Host Controller should issue an interrupt at the next interrupt threshold.	
14:12	Page Select (PG). These bits are set by software to indicate which of the buffer page pointers the offset field in this slot should be concatenated to produce the starting memory address for this transaction. The valid range of values for this field is 0 to 6.	
11:0	Transaction X Offset. This field is a value that is an offset, expressed in bytes, from the beginning of a buffer. This field is concatenated onto the buffer page pointer indicated in the adjacent *PG* field to produce the starting buffer address for this transaction.	

30.8.2.3.3 iTD Buffer Page Pointer List (Plus)

DWords 9-15 of an isochronous transaction descriptor are nominally page pointers (4K aligned) to the data buffer for this transfer descriptor. This data structure requires the associated data buffer to be contiguous (relative to virtual memory), but allows the physical memory pages to be non-contiguous. Seven page

pointers are provided to support the expression of eight isochronous transfers. The seven pointers allow for 3 (transactions) * 1024 (maximum packet size) * 8 (transaction records) (24576 bytes) to be moved with this data structure, regardless of the alignment offset of the first page.

Since each pointer is a 4K aligned page pointer, the least significant 12 bits in several of the page pointers are used for other purposes.

Table 30-39. it Buffer Pointer Page 0 (Plus)

Bit	Description
31:12	Buffer Pointer (Page 0). This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11:8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved. Bit reserved for future use and should be initialized by software to zero.
6:0	Device Address. This field selects the specific device serving as the data source or sink.

Table 30-40. iTD Buffer Pointer Page 1 (Plus)

Bit	Description
31:12	Buffer Pointer (Page 1). This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11	Direction (I/O). 0 = OUT; 1 = IN. This field encodes whether the high-speed transaction should use an IN or OUT PID.
10:0	Maximum Packet Size. This directly corresponds to the maximum packet size of the associated endpoint (*wMaxPacketSize*). This field is used for high-bandwidth endpoints where more than one transaction is issued per transaction description (for example, per micro-frame). This field is used with the *Multi* field to support high-bandwidth pipes. This field is also used for all IN transfers to detect packet babble. Software should not set a value larger than 1024 (400h). Any value larger yields undefined results.

Table 30-41. iTD Buffer Pointer Page 2 (Plus)

Bit	Description
31:12	Buffer Pointer. This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].

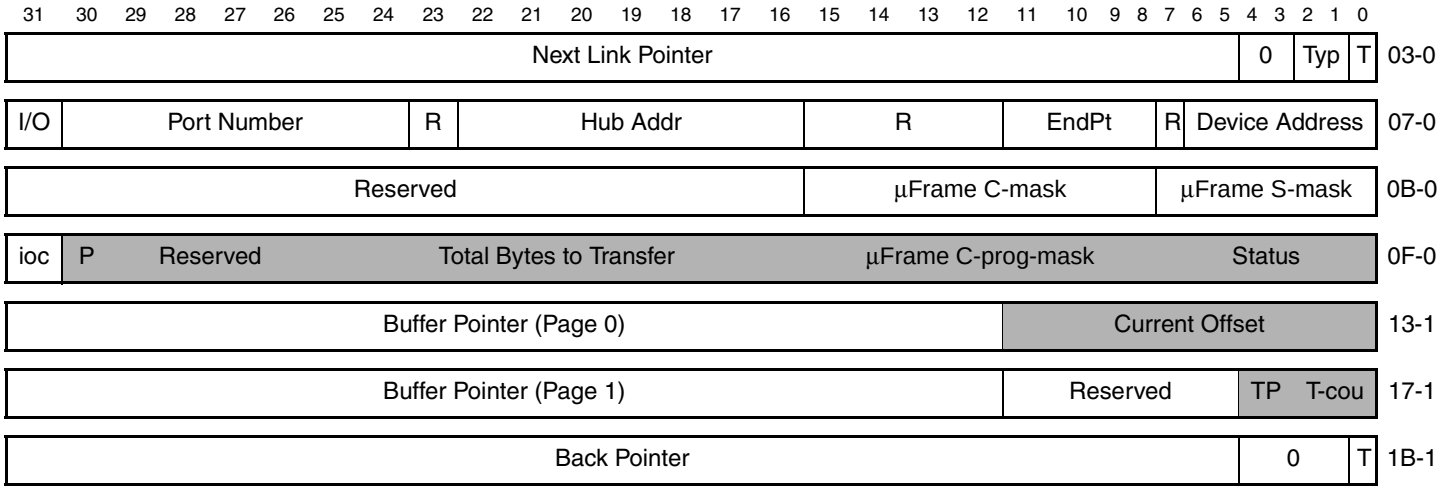
Bit	Description
11:2	Reserved. This bit reserved for future use and should be set to zero.
1:0	Multi. This field is used to indicate to the host controller the number of transactions that should be executed per transaction description (for example, per micro-frame). The valid values are: ValueMeaning 00bReserved. A zero in this field yields undefined results. 01bOne transaction to be issued for this endpoint per micro-frame 10bTwo transactions to be issued for this endpoint per micro-frame 11bThree transactions to be issued for this endpoint per micro-frame

Table 30-42. iTD Buffer Pointer Page 3-6

Bit	Description
31:12	Buffer Pointer. This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11:0	Reserved. These bits reserved for future use and should be set to zero.

30.8.2.4 Split Transaction Isochronous Transfer Descriptor (siTD)

All Full-speed isochronous transfers through the internal transaction translator are managed using the siTD data structure. This data structure satisfies the operational requirements for managing the split transaction protocol.



	Host Controller Read/Write		Host Controller Read Only.

Figure 30-46. Split-Transaction Isochronous Transaction Descriptor (siTD)

30.8.2.4.1 Next Link Pointer

DWord0 of a siTD is a pointer to the next schedule data structure.

Table 30-43. Next Link Pointer

Bit	Description
31:5	Next Link Pointer (LP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4:3	Reserved. These bits must be written as zeros.
2:1	QH/(s)iTD Select (Typ). This field indicates to the Host Controller whether the item referenced is an iTD/siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: ValueMeaning 00biTD (isochronous transfer descriptor) 01bQH (queue head) 10bsiTD (split transaction isochronous transfer descriptor 11bFSTN (frame span traversal node)
0	Terminate (T). 1=Link Pointer field is not valid. 0=Link Pointer is valid.

30.8.2.4.2 siTD Endpoint Capabilities/Characteristics

DWords 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent Companion Controller, and micro-frame scheduling control.

Table 30-44. Endpoint and Transaction Translator Characteristics

Bit	Description
31	Direction (I/O).0 = OUT; 1 = IN. This field encodes whether the full-speed transaction should be an IN or OUT.
30:24	Port Number. This field is the port number of the recipient Transaction Translator.
23	Reserved. Bit reserved and should be set to zero.
22:16	Hub Address. This field holds the device address of the Companion Controllers' hub.
15:12	Reserved. Field reserved and should be set to zero.
11:8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.

Bit	Description
7	Reserved. Bit is reserved for future use. It should be set to zero.
6:0	Device Address. This field selects the specific device serving as the data source or sink.

Table 30-45. Micro-frame Schedule Control

Bit	Description
31:16	Reserved. This field reserved for future use. It should be set to zero.
15:8	Split Completion Mask (μFrame C-Mask). This field (along with the *Active* and *SplitX- state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute complete-split transactions. When the criteria for using this field is met, an all zeros value has undefined behavior. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the μ*Frame C-Mask* field is a one, then this siTD is a candidate for transaction execution. There may be more than one bit in this mask set.
7:0	Split Start Mask (μFrame S-mask). This field (along with the *Active* and *SplitX-state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute start-split transactions. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the μ*Frame S-mask* field is a one, then this siTD is a candidate for transaction execution. An all zeros value in this field, in combination with existing periodic frame list has undefined results.

30.8.2.4.3 siTD Transfer State

DWords 3-6 are used to manage the state of the transfer.

Table 30-46. siTD Transfer Status and Control

Bit	Description
31	Interrupt On Complete (ioc). 0 = Do not interrupt when transaction is complete. 1 = Do interrupt when transaction is complete. When the host controller determines that the split transaction has completed it will assert a hardware interrupt at the next interrupt threshold.
30	Page Select (P). Used to indicate which data page pointer should be concatenated with the *CurrentOffset* field to construct a data buffer pointer (0 selects *Page 0* pointer and 1 selects *Page 1*). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero).
29:26	Reserved. This field reserved for future use and should be set to zero.
25:16	Total Bytes To Transfer. This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh)

Bit	Description	
15:8	μFrame Complete-split Progress Mask (C-prog-Mask). This field is used by the host controller to record which split-completes has been executed.	
7:0	Status. This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:	
	Bit	Definition
	7	Active. Set to one by software to enable the execution of an isochronous split transaction by the Host Controller.
	6	ERR. Set to a one by the Host Controller when an ERR response is received from the Companion Controller.
	5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). In the case of an under run, the Host Controller will transmit an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary.
	4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.
	3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit will only be set for IN transactions.
	2	Missed Micro-Frame. The host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction.
	1	Split Transaction State (SplitXstate). The bit encodings are: ValueMeaning 00bDo Start Split. This value directs the host controller to issue a Start split transaction to the endpoint when a match is encountered in the S-mask. 01bDo Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint when a match is encountered in the C-mask.
	0	Reserved. Bit reserved for future use and should be set to zero.

30.8.2.4.4 siTD Buffer Pointer List (plus)

DWords 4 and 5 are the data buffer page pointers for the transfer. This structure supports one physical page cross. The most significant 20 bits of each DWord in this section are the 4K (page) aligned buffer pointers. The least significant 12 bits of each DWord are used as additional transfer state.

Table 30-47. Buffer Page Pointer List (plus)

Bit	Description		
31:12	Buffer Pointer List. Bits [31:12] of DWords 4 and 5 are 4K paged aligned, physical memory addresses. These bits correspond to physical address bits [31:12] respectively. The lower 12 bits in each pointer are defined and used as specified below. The field *P* specifies the *current* active pointer		
11:0	Page 0: Current Offset. The 12 least significant bits of the Page 0 pointer is the current byte offset for the current page pointer (as selected with the page indicator bit (*P* field)). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero). The least significant bits of Page 1 pointer is split into three sub-fields Page 1:		
	Bits	Description	
	11:5	Reserved.	
	4:3	Transaction position (TP). This field is used with T-count to determine whether to send *all*, *first*, *middle*, or *last* with each outbound transaction payload. System software must initialize this field with the appropriate starting value. The host controller must correctly manage this state during the lifetime of the transfer. The bit encodings are: Value Meaning 00b All. The entire full-speed transaction data payload is in this transaction (that is, less than or equal to 188 bytes). 01b Begin. This is the first data payload for a full-speed that is greater than 188 bytes.transaction 10B Mid. This is the *middle* payload for a full-speed OUT transaction that is larger than 188 bytes. 11b End. This is the *last* payload for a full-speed OUT transaction that was larger than 188 bytes.	
	2:0	Transaction count (T-Count). Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.	

30.8.2.4.5 siTD Back Link Pointer

DWord 6 of a siTD is simply another schedule link pointer. This pointer is always zero, or references a siTD. This pointer cannot reference any other schedule data structure.

Table 30-48. siTD Back Link Pointer

Bit	Description
31:5	siTD Back Pointer. This field is a physical memory pointer to a siTD.
4:1	Reserved. This field is reserved for future use. It should be set to zero.
0	Terminate (T). 1 = siTD Back Pointer field is not valid. 0 = siTD Back Pointer field is valid.

30.8.2.5 Queue Element Transfer Descriptor (qTD)

This data structure is only used with a queue head. This data structure is used for one or more USB transactions. This data structure is used to transfer up to 20480 (5*4096) bytes. The structure contains two structure pointers used for queue advancement, a DWord of transfer state, and a five-element array of data buffer pointers. This structure is 32 bytes (or one 32-byte cache line). This data structure must be physically contiguous.

The buffer associated with this transfer must be virtually contiguous. The buffer may start on any byte boundary. A separate buffer pointer list element must be used for each physical page in the buffer, regardless of whether the buffer is physically contiguous.

Host controller updates (host controller writes) to stand-alone qTDs only occur during transfer retirement. References in the following bit field definitions of updates to the qTD are to the qTD portion of a queue head.

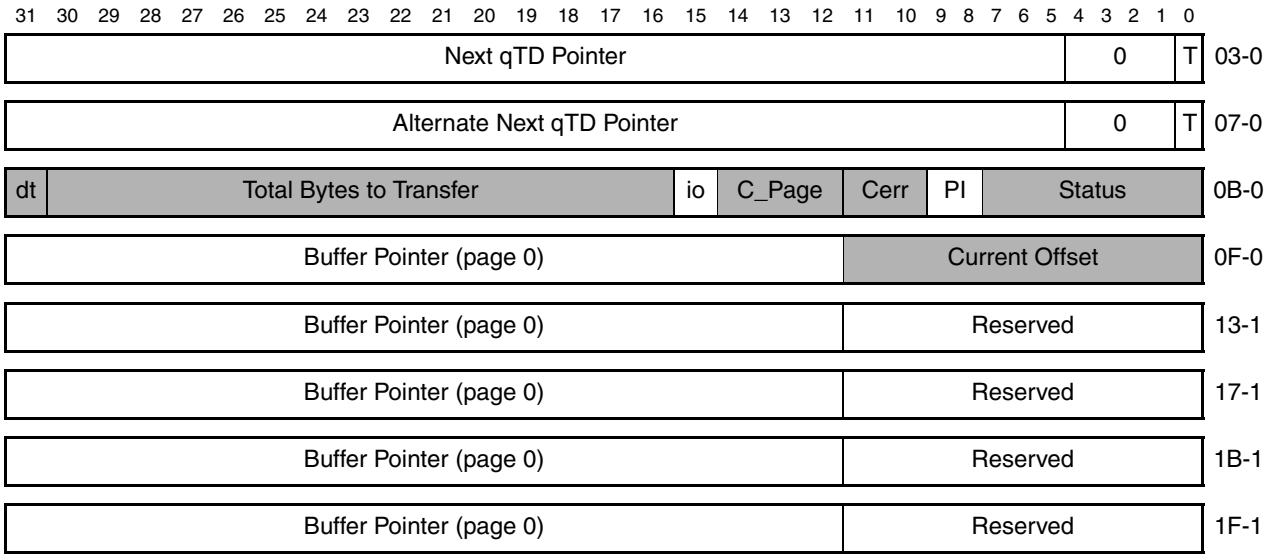


Figure 30-47. Queue Element Transfer Descriptor Block Diagram

Queue Element Transfer Descriptors must be aligned on 32-byte boundaries.

30.8.2.5.1 Next qTD Pointer

The first DWord of an element transfer descriptor is a pointer to another transfer element descriptor.

Table 30-49. D Next Element Transfer Pointer (DWord 0)

Bit	Description
31:5	Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed. The field corresponds to memory address signals[31:5], respectively.
4:1	Reserved. These bits are reserved and their value has no effect on operation.
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

30.8.2.5.2 Alternate Next qTD Pointer

The second DWord of a queue element transfer descriptor is used to support hardware-only advance of the data stream to the next client buffer on short packet. To be more explicit the host controller will always use this pointer when the current qTD is retired due to short packet.

Table 30-50. TD Alternate Next Element Transfer Pointer (DWord 1)

Bit	Description
31:5	Alternate Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed in the event that the current qTD execution encounters a short packet (for an IN transaction). The field corresponds to memory address signals [31:5], respectively.
4:1	Reserved. These bits are reserved and their value has no effect on operation.
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

30.8.2.5.3 qTD Token

The third DWord of a queue element transfer descriptor contains most of the information the host controller requires to execute a USB transaction (the remaining endpoint-addressing information is specified in the queue head).

NOTE	The field descriptions forward reference fields defined in the queue head. Where necessary, these forward references are preceded with a QH notation.

Table 30-51. qTD Token (DWord 2)

Bit	Description
31	Data Toggle. This is the data toggle sequence bit. The use of this bit depends on the setting of the *Data Toggle Control* bit in the queue head.
30:16	Total Bytes to Transfer. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction, only on the successful completion of the transaction. The maximum value software may store in this field is 5 * 4K (5000H). This is the maximum number of bytes 5 page pointers can access. If the value of this field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the host controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for OUT transfers that *Total Bytes To Transfer* be an even multiple of QHD.Maximum Packet Length. If software builds such a transfer descriptor for an OUT transfer, the last transaction will always be less than QHD.Maximum Packet Length. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16K(4000H).
15	Interrupt On Complete (IOC). If this bit is set to a one, it specifies that when this qTD is completed, the Host Controller should issue an interrupt at the next interrupt threshold.
14:12	Current Page (C_Page). This field is used as an index into the qTD buffer pointer list. Valid values are in the range 0H to 4H. The host controller is not required to write this field back when the qTD is retired.
11:10	Error Counter (CERR). This field is a 2-bit down counter that keeps track of the number of consecutive Errors detected while executing this qTD. If this field is programmed with a non-zero value during set-up, the Host Controller decrements the count and writes it back to the qTD if the transaction fails. If the counter counts from one to zero, the Host Controller marks the qTD inactive, sets the *Halted* bit to a one, and error status bit for the error that caused *CERR* to decrement to zero. An interrupt will be generated if the *USB Error Interrupt Enable* bit in the USBINTR register is set to a one. If HCD programs this field to zero during set-up, the Host Controller will not count errors for this qTD and there will be no limit on the retries of this qTD. Note that write-backs of intermediate execution state are to the queue head overlay area, not the qTD. Error Decrement Counter Transaction Error Yes Data Buffer Error No[3] Stalled No[1] Babble Detected No[1] No Error No[2]

	Error	Decrement Counter Error Decrement Counter
	1	Detection of Babble or Stall automatically halts the queue head. Thus, count is not decremented

Bit	Description	
	2	If the EPS field indicates a HS device or the queue head is in the Asynchronous Schedule (and *PIDCode* indicates an IN or OUT) and a bus transaction completes and the host controller does not detect a transaction error, then the host controller should reset *CERR* to extend the total number of errors for this transaction. For example, *CERR* should be reset with maximum value (3) on each successful completion of a transaction. The host controller must never reset this field if the value at the start of the transaction is 00b. See section Split Transaction Execution State Machine for Interrupt for CERR adjustment rules when the EPS field indicates a FS or LS device and the queue head is in the Periodic Schedule. See section Asynchronous - Do Complete Split for CERR adjustment rules when the EPS field indicates a FS or LS device, the queue head is in the Asynchronous schedule and the *PIDCode* indicates a SETUP.
	3	Data buffer errors are host problems. They don't count against the device's retries.
	Note: Software must not program CERR to a value of zero when the EPS field is programmed with a value indicating a Full- or Low-speed device. This combination could result in undefined behavior.	
9:8	PID Code. This field is an encoding of the token, which should be used for transactions associated with this transfer descriptor. Encodings are:	
	00b	OUT Token generates token (E1H)
	01b	IN Token generates token (69H)
	10b	SETUP Token generates token (2DH) (undefined if endpoint is an Interrupt the queue head is non-zero.) transfer type, for example, μ*Frame S-mask* field in
	11b	Reserved
7:0	Status. This field is used by the Host Controller to communicate individual command execution states back to HCD. This field contains the status of the last transaction performed on this qTD. The bit encodings are:	
	Bit	**Status Field Description**
	7	Active. Set to one by software to enable the execution of transactions by the Host Controller.
	6	Halted. Set to a one by the Host Controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this qTD. This can be caused by babble, the error counter counting down to zero, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set to a one, the Active bit is also set to zero.

Bit	Description	
	5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, the Host Controller will force a timeout condition on the USB, invalidating the transaction at the source. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction. In addition to setting this bit, the Host Controller also sets the *Halted* bit to a one. Since "babble" is considered a fatal error for the transfer, setting the Halted bit to a one insures that no more transactions occur because of this descriptor.
	3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	2	Missed Micro-Frame. This bit is ignored unless the *QH.EPS* field indicates a full- or low-speed endpoint and the queue head is in the periodic list. This bit is set when the host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.

Bit	Description	
	1	Split Transaction State (SplitXstate). This bit is ignored by the host controller unless the *QH.EPS* field indicates a full- or low-speed endpoint. When a Full- or Low-speed device, the host controller uses this bit to track the state of the split- transaction. The functional requirements of the host controller for managing this state bit and the split transaction protocol depends on whether the endpoint is in the periodic or asynchronous schedule. The bit encodings are: ValueMeaning 0b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint. 1b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint.
	0	Ping State (P)/ERR. If the *QH.EPS* field indicates a High-speed device and the *PID_Code* indicates an OUT endpoint, then this is the state bit for the Ping protocol. The bit encodings are: ValueMeaning 0b Do OUT. This value directs the host controller to issue an OUT PID to the endpoint. 1b Do Ping. This value directs the host controller to issue a PING PID to the endpoint. If the *QH.EPS* field does not indicate a High-speed device, then this field is used as an error indicator bit. It is set to a one by the host controller whenever a periodic split-transaction receives an ERR handshake.

30.8.2.5.4 qTD Buffer Page Pointer List

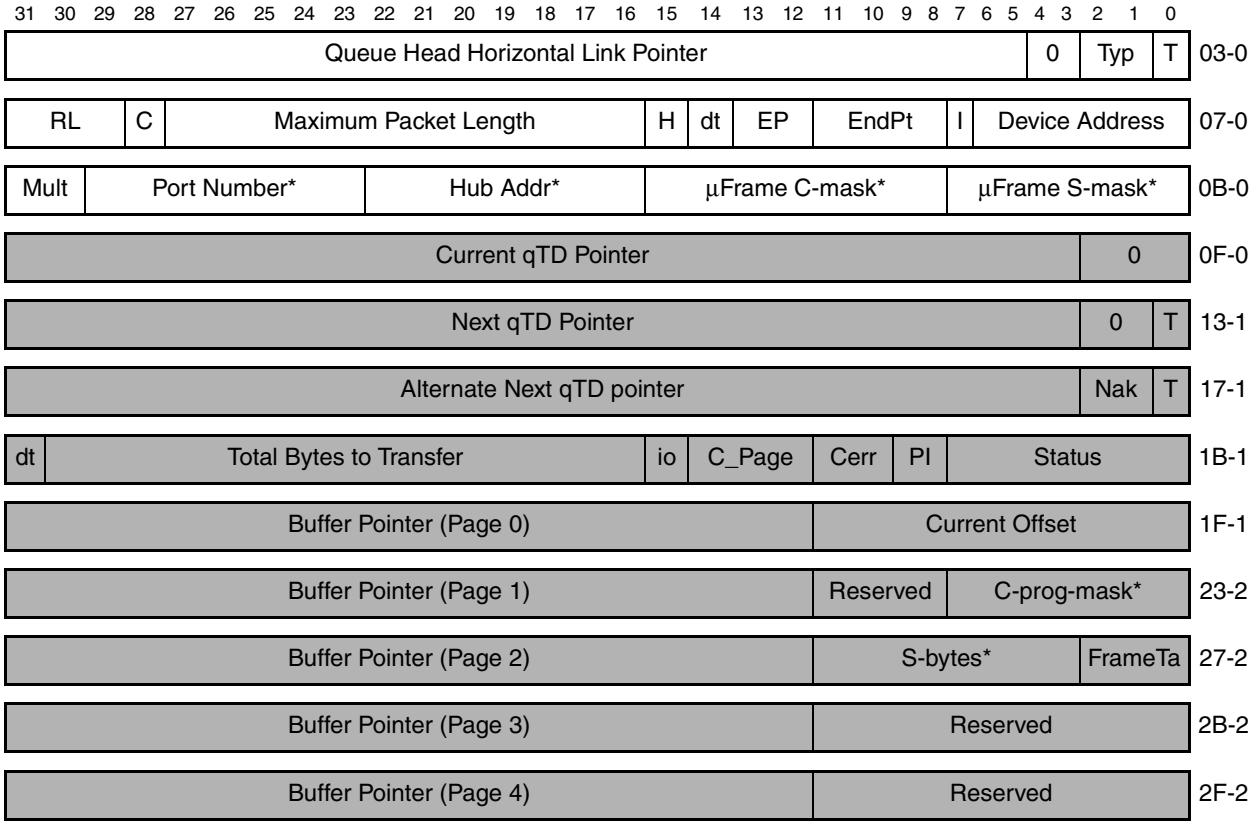
The last five DWords of a queue element transfer descriptor is an array of physical memory address pointers. These pointers reference the individual pages of a data buffer.

System software initializes *Current Offset* field to the starting offset into the current page, where current page is selected via the value in the *C_Page* field.

Table 30-52. qTD Buffer Pointer(s) (DWords 3-7)

Bit	Description
31:12	Buffer Pointer List. Each element in the list is a 4K page aligned physical memory address. The lower 12 bits in each pointer are reserved (except for the first one), as each memory pointer must reference the start of a 4K page. The field C_Page specifies the current active pointer. When the transfer element descriptor is fetched, the starting buffer address is selected using C_Page (similar to an array index to select an array element). If a transaction spans a 4K buffer boundary, the host controller must detect the page-span boundary in the data stream, increment C_Page and advance to the next buffer pointer in the list, and conclude the transaction via the new buffer pointer.
11:0	Current Offset (Reserved). This field is reserved in all pointers except the first one (for example, Page 0). The host controller should ignore all reserved bits. For the page 0 current offset interpretation, this field is the byte offset into the current page (as selected by C_Page). The host controller is not required to write this field back when the qTD is retired. Software should ensure the Reserved fields are initialized to zeros.

30.8.2.6 Queue Head



Transfer OverlayTransfer Results

Static Endpoint State

*These fields are used exclusively to support split transactions to USB 2.0 Hubs

	Host Controller Read/Write		Host Controller Read Only.

Figure 30-48. Queue Head Structure Layout

Queue Head Horizontal Link Pointer

The first DWord of a Queue Head contains a link pointer to the next data object to be processed after any required processing in this queue has been completed, as well as the control bits defined below.

This pointer may reference a queue head or one of the isochronous transfer descriptors. It must not reference a queue element transfer descriptor.

Table 30-53. Queue Head DWord 0

Bit	Description
31:5	Queue Head Horizontal Link Pointer (QHLP). This field contains the address of the next data object to be processed in the horizontal list and corresponds to memory address signals [31:5], respectively.
4:3	Reserved. These bits must be written as zeros.
2:1	QH/(s)iTD Select (Typ). This field indicates to the hardware whether the item referenced by the link pointer is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: ValueMeaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor) 11b FSTN (frame span traversal node)
0	Terminate (T). 1=Last QH (pointer is invalid). 0=Pointer is valid. If the queue head is in the context of the periodic list, a one bit in this field indicates to the host controller that this is the end of the periodic list. This bit is ignored by the host controller when the queue head is in the Asynchronous schedule. Software must ensure that queue heads reachable by the host controller always have valid horizontal link pointers.

30.8.2.6.1 Endpoint Capabilities/Characteristics

The second and third DWords of a Queue Head specifies static information about the endpoint. This information does not change over the lifetime of the endpoint. There are three types of information in this region:

- Endpoint Characteristics. These are the USB endpoint characteristics including addressing, maximum packet size, and endpoint speed.

- Endpoint Capabilities. These are adjustable parameters of the endpoint. They effect how the endpoint data stream is managed by the host controller.
- Split Transaction Characteristics. This data structure is used to manage full- and low-speed data streams for bulk, control, and interrupt via split transactions to USB2.0 Hub Transaction Translator. There are additional fields used for addressing the hub and scheduling the protocol transactions (for periodic).

The host controller must not modify the bits in this region.

Table 30-54. Endpoint Characteristics: Queue Head DWord 1

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31:28</td><td>Nak Count Reload (RL). This field contains a value, which is used by the host controller to reload the Nak Counter field.</td></tr>
<tr><td>27</td><td>Control Endpoint Flag (C). If the QH.EPS field indicates the endpoint is not a high-speed device, and the endpoint is a control endpoint, then software must set this bit to a one. Otherwise, it should always set this bit to a zero.</td></tr>
<tr><td>26:16</td><td>Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).</td></tr>
<tr><td>15</td><td>Head of Reclamation List Flag (H). This bit is set by System Software to mark a queue head as being the head of the reclamation list.</td></tr>
<tr><td>14</td><td>Data Toggle Control (DTC). This bit specifies where the host controller should get the initial data toggle on an overlay transition.
0b Ignore DT bit from incoming qTD.
Host controller preserves DT bit in
the queue head.
1b Initial data toggle comes from incoming qTD DT bit.
Host controller
replaces DT bit in the queue head from the DT bit in the qTD.</td></tr>
<tr><td>13:12</td><td>Endpoint Speed (EPS). This is the speed of the associated endpoint. Bit combinations are:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>00b</td><td>Full-Speed (12Mbs)</td></tr>
<tr><td>01b</td><td>Low-Speed (1.5Mbs)</td></tr>
<tr><td>10b</td><td>High-Speed (480 Mb/s)</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table>
This field must not be modified by the host controller.</td></tr>
<tr><td>11:8</td><td>Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.</td></tr>
</table>

Bit	Description
7	Inactivate on Next Transaction (I). This bit is used by system software to request that the host controller set the Active bit to zero. See Section Rebalancing the Periodic Schedule for full operational details. This field is only valid when the queue head is in the Periodic Schedule and the EPS field indicates a Full or Low-speed endpoint. Setting this bit to a one when the queue head is in the Asynchronous Schedule or the EPS field indicates a high-speed device yields undefined results.
6:0	Device Address. This field selects the specific device serving as the data source or sink.

Table 30-55. Endpoint Capabilities: Queue Head DWord 2

Bit	Description
31:30	High-Bandwidth Pipe Multiplier (Mult). This field is a multiplier used to key the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution. The host controller makes the simplifying assumption that software properly initializes this field (regardless of location of queue head in the schedules or other run time parameters). The valid values are: ValueMeaning 00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro-frame 10b Two transactions to be issued for this endpoint per micro-frame 11b Three transactions to be issued for this endpoint per micro-frame
29:23	Port Number. This field is ignored by the host controller unless the *EPS* field indicates a full- or low-speed device. The value is the port number identifier on the USB 2.0 Hub (for hub at device address *Hub Addr* below), below which the full- or low-speed device associated with this endpoint is attached. This information is used in the split-transaction protocol.
22:16	Hub Addr. This field is ignored by the host controller unless the *EPS* field indicates a full-or low-speed device. The value is the USB device address of the USB 2.0 Hub below which the full- or low-speed device associated with this endpoint is attached. This field is used in the split-transaction protocol.

Bit	Description
15:8	Split Completion Mask (μFrame C-Mask). This field is ignored by the host controller unless the *EPS* field indicates this device is a low- or full-speed device and this queue head is in the periodic list. This field (along with the *Active* and *SplitX-state* fields) is used to determine during which micro-frames the host controller should execute a complete-split transaction. When the criteria for using this field are met, a zero value in this field has undefined behavior. This field is used by the host controller to match against the three low-order bits of the FRINDEX register. If the FRINDEX register bits decode to a position where the μ*Frame C- Mask* field is a one, then this queue head is a candidate for transaction execution. There may be more than one bit in this mask set.
7:0	Interrupt Schedule Mask (μFrame S-mask). This field is used for all endpoint speeds. Software should set this field to a zero when the queue head is on the asynchronous schedule. A non-zero value in this field indicates an interrupt endpoint. The host controller uses the value of the three low-order bits of the FRINDEX register as an index into a bit position in this bit vector. If the μ*Frame S-mask* field has a one at the indexed bit position then this queue head is a candidate for transaction execution. If the *EPS* field indicates the endpoint is a high-speed endpoint, then the transaction executed is determined by the *PID_Code* field contained in the execution area. This field is also used to support split transaction types: Interrupt (IN/OUT). This condition is true when this field is non-zero and the *EPS* field indicates this is either a full- or low-speed device. A zero value in this field, in combination with existing in the periodic frame list has undefined results.

30.8.2.6.2 Transfer Overlay

The nine DWords in this area represent a *transaction working space* for the host controller. The general operational model is that the host controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it follows the *Queue Head Horizontal Link Pointer* to the next queue head. The host controller will never follow the *Next Transfer Queue Element or Alternate Queue Element* pointers unless it is actively attempting to advance the queue. For the duration of the transfer, the host controller keeps the incremental status of the transfer in the overlay area. When the transfer is complete, the results are written back to the original queue element.

The DWord3 of a Queue Head contains a pointer to the source qTD currently associated with the overlay. The host controller uses this pointer to write back the overlay area into the source qTD after the transfer is complete.

Table 30-56. Current qTD Link Pointer

Bit	Description
31:5	Current Element Transaction Descriptor Link Pointer. This field contains the address Of the current transaction being processed in this queue and corresponds to memory address signals [31:5], respectively.
4:0	Reserved (R). These bits are ignored by the host controller when using the value as an address to write data. The actual value may vary depending on the usage.

The DWords 4-11 of a queue head are the transaction overlay area. This area has the same base structure as a Queue Element Transfer Descriptor. The queue head utilizes the reserved fields of the page pointers to implement tracking the state of split transactions.

This area is characterized as an *overlay* because when the queue is advanced to the next queue element, the source queue element is *merged* onto this area. This area serves an execution cache for the transfer.

Table 30-57. Host-Controller Rules for Bits in Overlay (DWords 5, 6, 8 and 9)

DWord	Bit	Description
5	4:1	Nak Counter (NakCnt)µRW. This field is a counter the host controller decrements whenever a transaction for the endpoint associated with this queue head results in a Nak or Nyet response. This counter is reloaded from *RL* before a transaction is executed during the first pass of the reclamation list (relative to an Asynchronous List Restart condition). It is also loaded from RL during an overlay.
6	31	Data Toggle. The *Data Toggle Control* controls whether the host controller preserves this bit when an overlay operation is performed.
6	15	Interrupt On Complete (IOC). The IOC control bit is always inherited from the source qTD when the overlay operation is performed.
6	11:10	Error Counter (C_ERR). This two-bit field is copied from the qTD during the overlay and written back during queue advancement.
6	0	Ping State (P)/ERR. If the *EPS* field indicates a high-speed endpoint, then this field should be preserved during the overlay operation.
8	7:0	Split-transaction Complete-split Progress (C-prog-mask). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	4:0	Split-transaction Frame Tag (Frame Tag). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	11:5	S-bytes. Software must ensure that the *S-bytes* field in a *qTD* is zero before activating the *qTD*. This field is used to keep track of the number of bytes sent or received during an IN or OUT split transaction.

30.8.2.7 Periodic Frame Span Traversal Node (FSTN)

This data structure is to be used only for managing Full- and Low-speed transactions that span a Host-frame boundary. See section Host Controller Operational Model for FSTNs for full operational details. Software must not use an FSTN in the Asynchronous Schedule. An FSTN in the Asynchronous schedule results in undefined behavior. Software must not use the FSTN feature with a host controller whose HCIVERSION register indicates a revision implementation below 0096h. FSTNs are not defined for implementations before 0.96 and their use will yield undefined results.

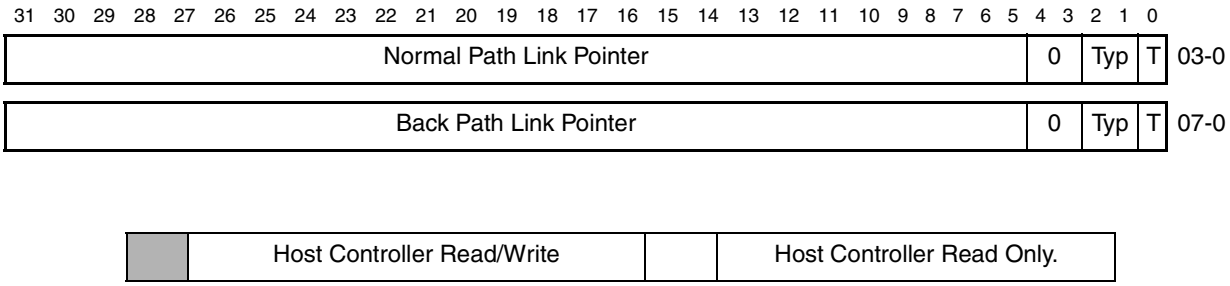


Figure 30-49. Frame Span Traversal Node Structure Layout

30.8.2.7.1 FSTN Normal Path Pointer

The first DWord of an FSTN contains a link pointer to the next schedule object. This object can be of any valid periodic schedule data type.

Bit	Description
31:5	Normal Path Link Pointer (NPLP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4:3	Reserved. These bits must be written as 0s.
2:1	QH/(s)iTD/FSTN Select (Typ). This field indicates to the Host Controller whether the item referenced is a iTD/siTD, a QH or an FSTN. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: ValueMeaning 00biTD (isochronous transfer descriptor) 01bQH (queue head) 10bsiTD (split transaction isochronous transfer descriptor) 11bFSTN (Frame Span Traversal Node)
0	Terminate (T). 1=Link Pointer field is not valid. 0=Link Pointer is valid.

30.8.2.7.2 FSTN Back Path Link Pointer

The second DWord of an FTSN node contains a link pointer to a queue head. If the *T-bit* in this pointer is a zero, then this FSTN is a *Save-Place* indicator. Its *Typ* field must be set by software to indicate the target data structure is a queue head. If the *T-bit* in this pointer is set to a one, then this FSTN is the *Restore* indicator. When the *T-bit* is a one, the host controller ignores the *Typ* field.

Bit	Description
31:5	Back Path Link Pointer (BPLP). This field contains the address of a Queue Head. This field corresponds to memory address signals [31:5], respectively.
4:3	Reserved. These bits must be written as 0s.

Bit	Description
2:1	Typ. Software must ensure this field is set to indicate the target data structure is a Queue Head. Any other value in this field yields undefined results.
0	Terminate (T). 1=Link Pointer field is not valid (that is, the host controller must not use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Restore indicator. 0=Link Pointer is valid (that is, the host controller may use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Save-Place indicator.

30.8.3 Host Operational Model

The general operational model is for the enhanced interface host controller hardware and enhanced interface host controller driver (generally referred to as system software). Each significant operational feature of the EHCI host controller is discussed in a separate section. Each section presents the operational model requirements for the host controller hardware. Where appropriate, recommended system software operational models for features are also presented.

30.8.3.1 Host Controller Initialization

When the system boots, the host controller is enumerated, assigned a base address for the register space and BIOS sets the FLADJ register to a system-specific value. After initial power-on or *HCReset* (hardware or via *HCReset* bit in the USBCMD register), all of the operational registers will be at their default values, as illustrated in Table 30-58. After a hardware reset, only the operational registers not contained in the Auxiliary power well will be at their default values.

Table 30-58. Default Values of Operational Register Space

Operational Register	Default Value (after Reset)
USBCMD	00080000h (00080B00h if *Asynchronous Schedule Park Capability is a one*)
USBSTS	00001000h
USBINTR	00000000h
FRINDEX	00000000h
CTRLDSSEGMENT	00000000h
PERIODICLISTBASE	Undefined
ASYNCLISTADDR	Undefined
CONFIGFLAG	00000000h
PORTSC	00002000h (w/*PPC* set to one); 00003000h (w/*PPC* set to a zero)

In order to initialize the host controller, software should perform the following steps

- Program the CTRLDSSEGMENT register with 4-Gigabyte segment where all of the interface data structures are allocated.
- Write the appropriate value to the USBINTR register to enable the appropriate interrupts.
- Write the base address of the Periodic Frame List to the PERIODICLIST BASE register. If there are no work items in the periodic schedule, all elements of the Periodic Frame List should have their *T-Bits* set to a one.
- Write the USBCMD register to set the desired interrupt threshold, frame list size (if applicable) and turn the host controller *ON* via setting the *Run/Stop* bit.
- Write a 1 to CONFIGFLAG register to route all ports to the EHCI controller (see Section Any time the USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.).

At this point, the host controller is up and running and the port registers will begin reporting device connects, etc. System software can enumerate a port through the reset process (where the port is in the enabled state). At this point, the port is active with SOFs occurring down the enabled port enabled High-speed ports, but the schedules have not yet been enabled. The EHCI Host controller will not transmit SOFs to enabled Full- or Low-speed ports. In order to communicate with devices via the asynchronous schedule, system software must write the ASYNDLISTADDR register with the address of a control or bulk queue head. Software must then enable the asynchronous schedule by writing a one to the *Asynchronous Schedule Enable* bit in the USBCMD register. In order to communicate with devices via the periodic schedule, system software must enable the periodic schedule by writing a one to the *Periodic Schedule Enable* bit in the USBCMD register. Note that the schedules can be turned on before the first port is reset (and enabled).

Any time the USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.

30.8.3.2 Port Routing and Control

A USB 2.0 Host controller is comprised of one high-speed host controller, which implements the EHCI programming interface and 0 to N USB 1.1 companion host controllers. Companion host controllers (cHCs) may be implementations of either Universal or Open host controller specifications. This configuration is used to deliver the required full USB 2.0-defined port capability; for example, Low-, Full-, and High-speed capability for every port. Figure 30-50 illustrates a simple block diagram of the port routing logic and its relationship to the high-speed and companion host controllers within a USB 2.0 host controller.

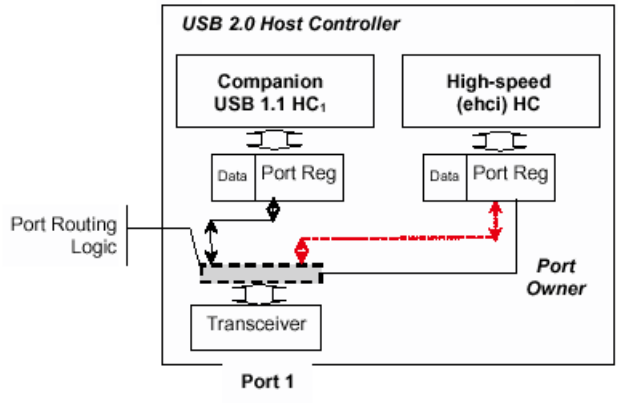


Figure 30-50. Example USB 2.0 Host Controller Port Routing Block Diagram

There exists one transceiver per physical port and each host controller module has its own port status and control registers. The EHCI controller has port status and control registers for every port. Each companion host controller has only the port control and status registers it is required to operate. Each transceiver can be controlled by either the EHCI host controller or one companion host controller. Routing logic lies between the transceiver and the port status and control registers.[1]

The port routing logic is controlled from signals originating in the EHCI host controller. The EHCI host controller has a *global* routing policy control field and per-port *ownership* control fields. The *Configured Flag* (*CF*) bit (defined in section BURSTSIZE) is the global routing policy control. At power-on or reset, the default routing policy is to the companion controllers (if they exist). If the system does not include a driver for the EHCI host controller and the host controller includes Companion Controllers, then the ports will still work in Full- and Low-speed mode (assuming the system includes a driver for the companion controllers). In general, when the EHCI owns the ports, the companion host controllers' port registers do not see a connect indication from the transceiver. Similarly, when a companion host controller owns a port, the EHCI controller's port registers do not see a connect indication from the transceiver. The details on the rules for the port routing logic are described in the following sections. The USB 2.0 host controller must be implemented as a multi-function PCI device if the implementation includes companion controllers. The companion host controllers' function numbers must be less than the EHCI host controller function number. The EHCI host controller must be a larger function number with respect to the companion host controllers associated with this EHCI host controller. If a PCI device implementation contains only an EHCI controller (that is, no companion controllers or other PCI functions), then the EHCI host controller must be function zero, in accordance with the PCI Specification. The *N_CC* field in the Structural Parameter register (*HCSPARAMS*) indicates whether the controller implementation includes companion host controllers. When *N_CC* has a non-zero value there exists companion host controllers. If N_CC has a value of zero, then the host controller implementation does not include companion host controllers. If the host controller root ports are exposed to attachment of full- or low-speed devices, the ports will always fail the high-speed chirp during reset and the ports will not be enabled. System software can notify the user of the illegal condition. This type of implementation requires a USB 2.0 hub be connected to a root port to provide full and low-speed device connectivity.

1.The routing logic should not be implemented in the 480 MHz clock domain of the transceiver.

System software uses information in the host controller capability registers to determine how the ports are routed to the companion host controllers. See Section HCSPARAMS—EHCI Compliant with extensions.[1]

30.8.3.2.1 Port Routing Control via EHCI *Configured* (*CF*) Bit

Each port in the USB 2.0 host controller can be routed either to a single companion host controller or to the EHCI host controller. The port routing logic is controlled by two mechanisms in the EHCI HC: a host controller global flag and per-port control. The *Configured Flag* (*CF*) bit (defined in Section BURSTSIZE), is used to globally set the policy of the routing logic. Each port register has a *Port Owner* control bit which allows the EHCI Driver to explicitly control the routing of individual ports. Whenever the *CF bit* transitions from a zero to a one (this transition is only available under program control) the port routing unconditionally routes all of the port registers to the EHCI HC (all *Port Owner* bits go to zero). While the *CF-bit* is a one, the EHCI Driver can control individual ports' routing via the *Port Owner* control bit. Likewise, whenever the *CF bit* transitions from a one to a zero (as a result of Aux power application, *HCRESET*, or software writing a zero to *CF-bit*), the port routing unconditionally routes all of the port registers to the appropriate companion HC. The default value for the EHCI HC's *CF bit* (after Aux power application or *HCRESET*) is zero. Table 30-59 summarizes the default routing for all the ports, based on the value of the EHCI HC's *CF bit*.

The *view* of the port depends on the current owner. A Universal or Open companion host controller will see port register bits consistent with the appropriate specification. Port bit definitions that are required for EHCI host controllers are not visible to companion host controllers.

Table 30-59. Default Port Routing Depending on EHCI HC CF Bit

HS CF Bit	Default Port Ownership	Explanation
0B	Companion HCs	The companion host controllers own the ports and only Full- and Low-speed devices are supported in the system. The exact port assignments are implementation dependent. The ports behave only as Full- and Low-speed ports in this configuration
1B	EHCI HC	The EHCI host controller has default ownership over all of the ports. The routing logic inhibits device connect events from reaching the companion HCs' port status and control registers when the port owner is the EHCI HC.The EHCI HC has access to the additional port status and control bits defined in this specification (see Section PORTSCx). The EHCI HC can temporarily release control of the port to a companion HC by setting the *PortOwner* bit in the PORTSC register to a one.

1.If an implementation includes more than one set of companion and EHCI host controllers, they are organized as groups of companion host controllers with intermixed EHCI controllers.

30.8.3.2.2 Port Routing Control via *PortOwner* and Disconnect Event

Manipulating the port routing via the *CF-bit* is an extreme process and not intended to be used during normal operation. The normal mode of port ownership transferal is on the granularity of individual ports using the *Port Owner* bit in the EHCI HC's PORTSC register (for hand-offs from EHCI to companion host controllers). Individual port ownership is returned to the EHCI controller when the port registers a device disconnect. When the disconnect is detected, the port routing logic immediately returns the port ownership to the EHCI controller. The companion host controller port register detects the device disconnect and operates normally.

Under normal operating conditions (assuming all HC drivers loaded and operational and the EHCI *CF-bit* is set to a one), the typical port enumeration sequence proceeds as illustrated below:

- Initial condition is that EHCI is port owner. A device is connected causing the port to detect a connect, set the port connect change bit and issue a port-change interrupt (if enabled).
- EHCI Driver identifies the port with the new connect change bit asserted and sends a change report to the hub driver. Hub driver issues a GetPortStatus() request and identifies the connect change. It then issues a request to clear the connect change, followed by a request to reset and enable the port.
- When the EHCI Driver receives the request to reset and enable the port, it first checks the value reported by the *LineStatus* bits in the PORTSC register. If they indicate the attached device is a full-speed device (for example, D+ is asserted), then the EHCI Driver sets the *PortReset* control bit to a one (and sets the *PortEnable* bit to a zero) which begins the reset-process. Software times the duration of the reset, then terminates reset signaling by writing a zero to the port reset bit. The reset process is actually complete when software reads a zero in the *PortReset* bit. The EHCI Driver checks the PortOwner bit in the PORTSC register. If set to a one, the connected device is a high-speed device and EHCI Driver (root hub emulator) issues a change report to the hub driver and the hub driver continues to enumerate the attached device.
- At the time the EHCI Driver receives the port reset and enable request the *LineStatus* bits might indicate a low-speed device. Additionally, when the port reset process is complete, the *PortEnable* field may indicate that a full-speed device is attached. In either case the EHCI driver sets the PortOwner bit in the PORTSC register to a one to release port ownership to a companion host controller.
- When the EHCI Driver sets *PortOwner* bit to a one, the port routing logic makes the connection state of the transceiver available to the companion host controller port register and removes the connection state from the EHCI HC port. The EHCI PORTSC register observes and reports a disconnect event via the disconnect change bit. The EHCI Driver detects the connection status change (either by polling or by port change interrupt) and then sends a change report to the hub driver. When the hub driver requests that port-state, the EHCI Driver responds with a reset complete change set to a one, a connect change set to a one and a connect status set to a zero. This information is derived directly from the EHCI port register. This will allow the hub driver to assume the device was disconnected during reset. It will acknowledge the change bits and wait for the next change event. While the EHCI controller does not own the port, it simply remains in a state where the port reports no device connected.
  The device-connect evaluation circuitry of the companion HC activates and detects the device, the companion Driver detects the connection and enumerates the port.

When a port is routed to a companion HC, it remains under the control of the companion HC until the device is disconnected from the root port (ignoring for now the scenario where EHCI's *CF-bit* transitions from a 1b to a 0b). When a disconnect occurs, the disconnect event is detected by both the companion HC port control and the EHCI port ownership control. On the event, the port ownership is returned immediately to the EHCI controller. The companion HC stack detects the disconnect and acknowledges as it would in an ordinary standalone implementation. Subsequent connects will be detected by the EHCI port register and the process will repeat.

30.8.3.2.3 Example Port Routing State Machine

Figure 30-51 illustrates an example of how the port ownership should be managed. The following sections describe the entry conditions to each state.

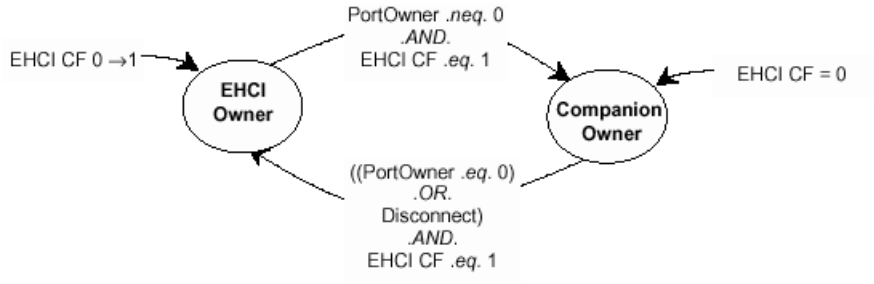


Figure 30-51. Port Owner Handoff State Machine

EHCI HC Owner

Entry to this state occurs whenever one of the following events occur:

- When the EHCI HC's *Configure Flag* (*CF*) bit in the CONFIGFLAG register transitions from a zero to a one. This signals the fact that the system has a host controller driver for the EHCI HC and that all ports in the USB 2.0 host controller must default route to the EHCI controller.
- When the port is owned by a companion HC and the device is disconnected from the port. The EHCI port routing control logic is notified of the disconnect, and returns port routing to the EHCI controller. The connection state of the companion HC goes immediately to the disconnected state (with appropriate side effect to connect change, enable and enable change). The companion HC driver will acknowledge the disconnect by setting the connect status change bit to a zero. This allows the companion HC's driver to interact with the port completely through the disconnect process.
- When system software writes a zero to the *PortOwner* bit in the PORTSC register. This allows software to take ownership of a port from a companion host controller. When this occurs, the routing logic to the companion HC effectively signals a disconnect to the companion HC's port status and control register.

Companion HC Owner

Entry to this state occurs whenever one of the following events occur:

- When the *Port Owner* field transitions from a zero to a one.
- When the HS-mode HC's *Configure Flag* (*CF*) is equal to zero.

On entry to this state, the routing logic allows the companion HC port register to detect a device connect. Normal port enumeration proceeds.

30.8.3.2.4 Port Power

The *Port Power Control* (*PPC*) bit in the HCSPARAMS register indicates whether the USB 2.0 host controller has port power control (See section HCSPARAMS—EHCI Compliant with extensions). When this bit is a zero, then the host controller does not support software control of port power switches. When in this configuration, the port power is always available and the companion host controllers must implement functionality consistent with port power always on. When the *PPC* bit is a one, then the host controller implementation includes port power switches. Each available switch has an output enable, which is referred to in this discussion as *PortPowerOutputEnable* (*PPE*). *PPE* is controlled based on the state of the combination bits *PPC* bit, EHCI *Configured* (*CF*)-bit and individual *Port Power* (*PP*) bits. Table 30-60 illustrates the summary behavioral model.

Table 30-60. Port Power Enable Control Rules

CF	CHC[2] (PP)	EHC[3] (PP)	Owner	PPE[1]	Description
0	0	X	CHC	0	When the EHCI controller has not been configured, the port is owned by the companion host controller. When the companion HC's port power select is off, then the port power is off.
0	1	X	CHC	1	Similar to previous entry. When the companion HC's port power select is on, then the port power is on.
1	0	0	CHC	0	Port owner has port power turned off, the power to port is off.
1	0	0	EHC	0	Port owner has port power turned off, the power to port is off.
1	0	1	EHC	1	Port owner has port power on, so power to port is on.
1	0	1	CHC	1	If either HC has port power turned on, the power to the port is on.
1	1	0	EHC	1	If either HC has port power turned on, the power to the port is on.
1	1	0	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	EHC	1	Port owner has port power on, so power to port is on.

a) [1]PPE (Port Power Enable). This bit actually turns on the port power switch (if one exists).

a) [2]CHC (Companion Host Controller).

a) [3]EHC (EHCI Host Controller).

30.8.3.2.5 Port Reporting Over-Current

Host controllers are by definition power providers on USB. Whether the ports are considered high- or low-powered is a platform implementation issue. Each EHCI PORTSC register has an over-current status and over-current change bit. The functionality of these bits is specified in the USB Specification Revision 2.0.

The over current detection and limiting logic usually resides outside the host controller logic. This logic may be associated with one or more ports. When this logic detects an over-current condition it is made available to both the companion and EHCI ports. The effect of an over-current status on a companion host controller port is beyond the scope of this document. The over-current condition effects the following bits in the PORTSC register on the EHCI port:

- *Over-current Active* bits are set to a one. When the over-current condition goes away, the *Over-current Active* bit will transition from a one to a zero.
- *Over-current Change* bits are set to a one. On every transition of the *Over-current Active* bit the host controller will set the *Over-current Change* bit to a one. Software sets the *Over-current Change* bit to a zero by writing a one to this bit.
- *Port Enabled/Disabled* bit is set to a zero. When this change bit gets set to a one, then the *Port Change Detect* bit in the USBSTS register is set to a one.
- *Port Power (PP)* bits may optionally be set to a zero. There is no requirement in USB that a power provider shut off power in an over current condition. It is sufficient to *limit* the current and leave power applied. When the *Over-current Change* bit transitions from a zero to a one, the host controller also sets the *Port Change Detect* bit in the USBSTS register to a one. In addition, if the *Port Change Interrupt Enable* bit in the USBINTR register is a one, then the host controller will issue an interrupt to the system. Refer to Table 30-61 for summary behavior for over-current detection when the host controller is halted (suspended from a device component point of view).

30.8.3.3 Suspend/Resume

The EHCI host controller provides an equivalent suspend and resume model as that defined for individual ports in a USB 2.0 Hub. Control mechanisms are provided to allow system software to suspend and resume individual ports. The mechanisms allow the individual ports to be resumed completely via software initiation. Other control mechanisms are provided to parameterize the host controller's response (or sensitivity) to external resume events. In this discussion, host-initiated, or software initiated resumes are called Resume Events/Actions. Bus-initiated resume events are called wake-up events. The classes of wakeup events are:

- Remote-wakeup enabled device asserts resume signaling. In similar kind to USB 2.0 Hubs, EHCI controllers must always respond to explicit device resume signaling and wake up the system (if necessary).
- Port connect and disconnect and over-current events. Sensitivity to these events can be turned on or off by using the per-port control bits in the PORTSC registers.

Selective suspend is a feature supported by every PORTSC register. It is used to place specific ports into a suspend mode. This feature is used as a functional component for implementing the appropriate power management policy implemented in a particular operating system. When system software intends to suspend the entire bus, it should selectively suspend all enabled ports, then shut off the host controller by setting the *Run/Stop* bit in the USBCMD register to a zero. The EHCI module can then be placed into a lower device state via the PCI power management interface (See Appendix A, Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com).

When a wake event occurs the system will resume operation and system software will eventually set the *Run/Stop* bit to a one and resume the suspended ports. Software must not set the *Run/Stop* bit to a one until it is confirmed that the clock to the host controller is stable. This is usually confirmed in a system implementation in that all of the clocks in the system are stable before the CPU is restarted. So, by definition, if software is running, clocks in the system are stable and the *Run/Stop* bit in the USBCMD

register can be set to a one. There are also minimum system software delays defined in the PCI Power Management Specification. Refer to this specification for more information.

30.8.3.3.1 Port Suspend/Resume

System software places individual ports into suspend mode by writing a one into the appropriate PORTSC *Suspend* bit. Software must only set the *Suspend* bit when the port is in the enabled state (*Port Enabled* bit is a one) and the EHCI is the port owner (*Port Owner* bit is a zero).

The host controller may evaluate the *Suspend* bit immediately or wait until a micro-frame or frame boundary occurs. If evaluated immediately, the port is not suspended until the current transaction (if one is executing) completes. Therefore, there may be several micro-frames of activity on the port until the host controller evaluates the *Suspend* bit. The host controller must evaluate the *Suspend* bit at least every frame boundary.

System software can initiate a resume on a selectively suspended port by writing a one to the *Force Port Resume* bit. Software should not attempt to resume a port unless the port reports that it is in the suspended state (see Section PORTSCx). If system software sets *Force Port Resume* bit to a one when the port is not in the suspended state, the resulting behavior is undefined. In order to assure proper USB device operation, software must wait for at least 10 milliseconds after a port indicates that it is suspended (*Suspend* bit is a one) before initiating a port resume via the *Force Port Resume* bit. When *Force Port Resume* bit is a one, the host controller sends resume signaling down the port. System software times the duration of the resume (nominally 20 milliseconds) then sets the *Force Port Resume* bit to a zero. When the host controller receives the write to transition *Force Port Resume* to zero, it completes the resume sequence as defined in the USB specification, and sets both the *Force Port Resume* and *Suspend* bits to zero. Software-initiated port resumes do not affect the *Port Change Detect* bit in the USBSTS register nor do they cause an interrupt if the *Port Change Interrupt Enable* bit in the USBINTR register is a one. An external USB event may also initiate a resume. The wake events are defined above. When a wake event occurs on a suspended port, the resume signaling is detected by the port and the resume is reflected downstream within 100 µsec. The port's *Force Port Resume* bit is set to a one and the *Port Change Detect* bit in the USBSTS register is set to a one. If the *Port Change Interrupt Enable* bit in the USBINTR register is a one the host controller will issue a hardware interrupt.

System software observes the resume event on the port, delays a port resume time (nominally 20 msec), then terminates the resume sequence by writing zero to the *Force Port Resume* bit in the port. The host controller receives the write of zero to *Force Port Resume,* terminates the resume sequence and sets *Force Port Resume* and *Suspend* port bits to zero. Software can determine that the port is enabled (not suspended) by sampling the PORTSC register and observing that the *Suspend* and *Force Port Resume* bits are zero. Software must ensure that the host controller is running (that is, *HCHalted* bit in the USBSTS register is a zero), before terminating a resume by writing a zero to a port's *Force Port Resume* bit. If *HCHalted* is a one when *Force Port Resume* is set to a zero, then SOFs will not occur down the enabled port and the device will return to suspend mode in a maximum of 10 milliseconds.

Table 30-61 summarizes the wake-up events. Whenever a resume event is detected, the *Port Change Detect* bit in the USBSTS register is set to a one. If the *Port Change Interrupt Enable* bit is a one in the USBINTR register, the host controller will also generate an interrupt on the resume event. Software

acknowledges the resume event interrupt by clearing the *Port Change Detect* status bit in the USBSTS register.

Table 30-61. Behavior During Wake-up Events

Port Status and Signaling Type	Signaled Port Response	Device State	
		D0	not D0
Port disabled, resume K-State received	No Effect	N/A	N/A
Port suspended, Resume K-State received	Resume reflected downstream on signaled port. Force Port Resume status bit in PORTSC register is set to a one. Port Change Detect bit in USBSTS register set to a one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is a one. A disconnect is detected.	Depending in the initial port state, the PORTSC Connected Enable status bits are set to zero, and the Connect Change status bit is set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is a zero. A disconnect is detected.	Depending on the initial port state, the PORTSC Connect and Enable status bits are set to zero, and the Connect Change status bit is set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [3]	[3]
Port is not connected and the port's WKCNNT_E bit is a one. A connect is detected.	PORTSC Connect Status and Connect Status Change bits are set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [2]	[2]
Port is not connected and the port's WKCNNT_E bit is a zero. A connect is detected.	PORTSC Connect Status and Connect Status Change bits are set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [3]	[3]
Port is connected and the port's WKOC_E bit is a one. An over-current condition occurs.	PORTSC Over-current Active, Over-current Change bits are set to a one. If Port Enable/Disable bit is a one, it is set to a zero. Port Change Detect bit in the USBSTS register is set to a one	[1], [2]	[2]
Port is connected and the port's WKOC_E bit is a zero. An over-current condition occurs.	PORTSC Over-current Active, Over-current Change bits are set to a one. If Port Enable/Disable bit is a one, it is set to a zero. Port Change Detect bit in the USBSTS register is set to a one.	[1], [3]	[3]

[1] Hardware interrupt issued if Port Change Interrupt Enable bit in the USBINTR register is a one.
[2] PME# asserted if enabled (Note: PME Status must always be set to a one).
[3] PME# not asserted.

30.8.3.4 Schedule Traversal Rules

The host controller executes transactions for devices using a simple, shared-memory schedule. The schedule is comprised of a few data structures, organized into two distinct lists. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic and hardware/software complexity.

System software maintains two schedules for the host controller: a periodic schedule and an asynchronous schedule. The root of the periodic schedule is the *PERIODICLISTBASE* register (see Section PERIODICLISTBASE; DEVICEADDR). The *PERIODICLISTBASE* register is the physical memory base address of the periodic frame list. The periodic frame list is an array of physical memory pointers. The objects referenced from the frame list must be valid schedule data structures as defined in Host Data Structures. In each micro-frame, if the periodic schedule is enabled (see Section Periodic Schedule) then the host controller must execute from the periodic schedule before executing from the asynchronous schedule. It will only execute from the asynchronous schedule after it encounters the end of the periodic schedule. The host controller traverses the periodic schedule by constructing an array offset reference from the *PERIODICLISTBASE* and the *FRINDEX* registers (see Figure 30-52). It fetches the element and begins traversing the graph of linked schedule data structures.

The end of the periodic schedule is identified by a *next* link pointer of a schedule data structure having its *T-bit* set to a one. When the host controller encounters a *T-Bit* set to a one during a horizontal traversal of the periodic list, it interprets this as an End-Of-Periodic-List mark. This causes the host controller to cease working on the periodic schedule and transitions immediately to traversing the asynchronous schedule. Once this transition is made, the host controller executes from the asynchronous schedule until the end of the micro-frame.

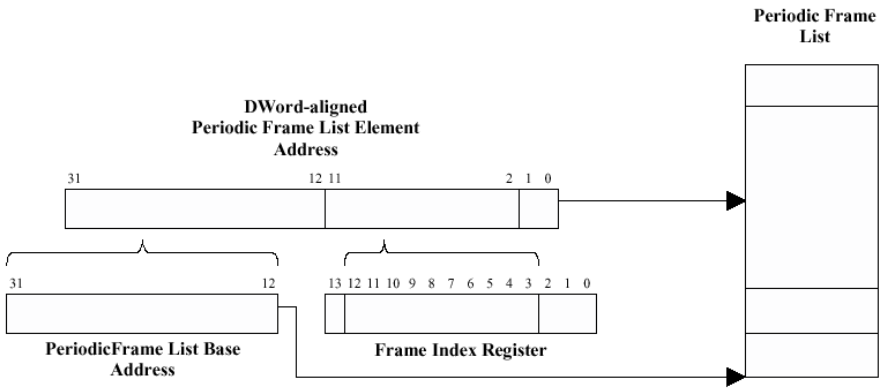


Figure 30-52. Derivation of Pointer into Frame List Array

When the host controller determines that it is time to execute from the asynchronous list, it uses the operational register *ASYNCLISTADDR* to access the asynchronous schedule, see Figure 30-53.

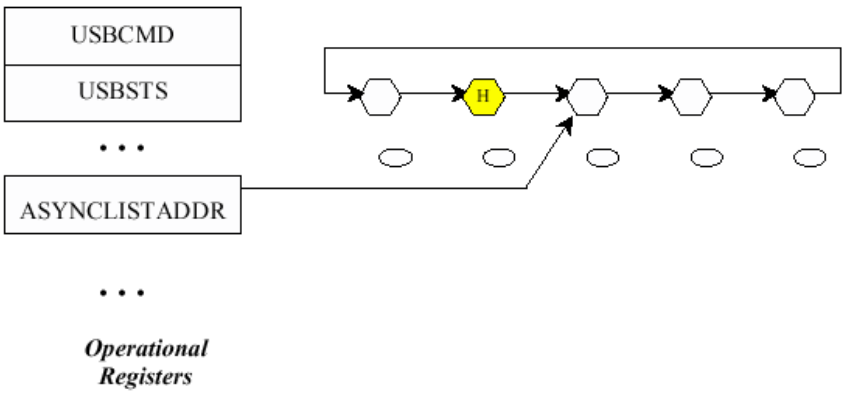


Figure 30-53. General Format of Asynchronous Schedule List

The *ASYNCLISTADDR* register contains a physical memory pointer to the *next* queue head. When the host controller makes a transition to executing the asynchronous schedule, it begins by reading the queue head referenced by the *ASYNCLISTADDR* register. Software must set queue head *horizontal* pointer *T-bits* to a zero for queue heads in the asynchronous schedule. See Section Asynchronous Schedule for complete operational details.

30.8.3.4.1 Example—Preserving Micro-Frame Integrity

One of the requirements of a USB host controller is to maintain *Frame Integrity.* This means that the HC must preserve the micro-frame boundaries. For example: SOF packets must be generated on time (within the specified allowable jitter), and High-speed EOF1,2 thresholds must be enforced. The end of micro-frame timing points EOF1 and EOF2 are clearly defined in the USB Specification Revision 2.0. One implication of this responsibility is that the HC must ensure that it does not start transactions that will not be completed before the end of the micro-frame. More precisely, no transactions should be started by the host controller, which cannot be completed in their entirety before the EOF1 point. In order to enforce this rule, the host controller must check each transaction before it starts to ensure that it will complete before the end of the micro-frame.

So, what exactly needs to be involved in this check? Fundamentally, the transaction data payload, plus bit stuffing, plus transaction overhead must be taken into consideration. It is possible to be extremely accurate on how much time the next transaction will take. Take OUTs for an example. The host controller must fetch all of the OUT data from memory in order to send it onto the USB bus. A host controller implementation could pre-fetch *all* of the OUT data, and pre-compute the actual number of bits in the token and data packets. In addition, the system knows the depth of the target endpoint, so it could closely estimate turnaround time for handshake. In addition, the host controller knows the size of a handshake packet. Pre-computing effects of bit stuffing and summing up the other overhead numbers could allow the host controller to know exactly whether there was enough bus time, before EOF1 to complete the OUT transaction. To accomplish this particular approach takes an inordinate amount of time and hardware complexity.

The alternative is to make a reasonable guess whether the next transaction can be started. An example approximation algorithm is described below. This example algorithm relies on the EHCI policy that periodic transactions are scheduled first in the micro-frame. It is a reasonable assumption that software will

never over-commit the micro-frame to periodic transactions greater than the specification allowable 80%. In the available remaining 20% bandwidth, the host controller has some ability (in this example) to decide whether or not to execute a transaction. The result of this algorithm is that sometimes, under some circumstances a transaction will not be executed that *could* have been executed. However, under all circumstances, a transaction will never be started unless there is enough time in the frame to complete the transaction.

Transaction Fit - A Best-Fit Approximation Algorithm

A curve is calculated which represents the *latest* start time for every packet size, at which software will schedule the start of a periodic transaction. This curve is the 80% bandwidth curve. Another curve is calculated which is the absolute, latest permitted start time for every packet size. This curve represents the absolute latest time, that a transaction of each packet size can be started and completed, in the micro-frame. A plot of these two curves is illustrated in Figure 30-54. The plot Y-axis represents the number of byte-times left in a frame.

The space between the *80%* and the *Last Start* plots is bandwidth reclamation area. In this algorithm the host controller may skip transactions during this time if it is prudent.

The Best-Fit Approximation method plots a function ($f(x)$) between the 80% and *Last Start* curves. The function $f(x)$ adds a constant to every transaction's maximum packet size and the result compared with the number of bytes left in the frame. The constant represents an approximation of the effects of bit stuffing and protocol overhead. The host controller starts transactions whose results land *above* the function curve. The host controller will not start transactions whose results land below the function curve.

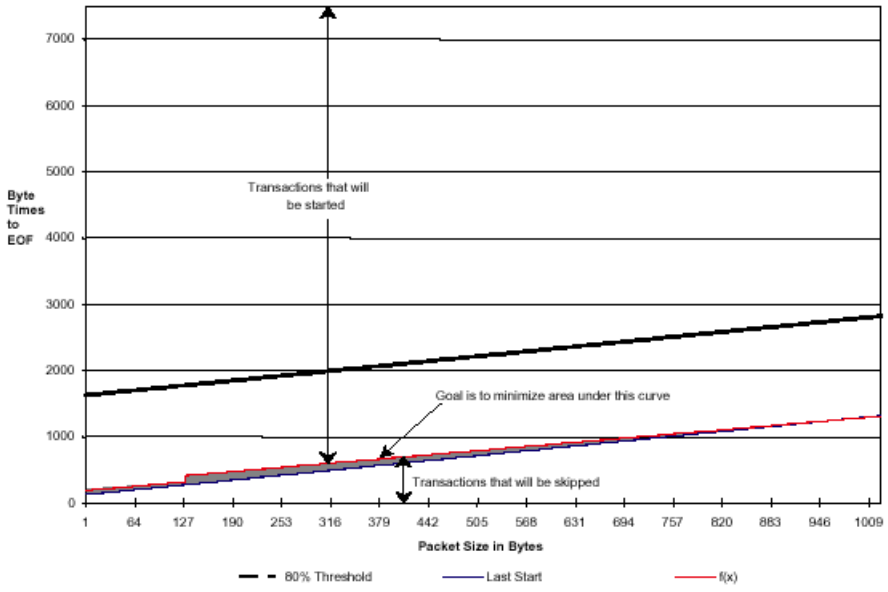


Figure 30-54. Best Fit Approximation

The *LastStart* line was calculated in this example to assume the absolute worst-case bus overhead per transaction. The particular transaction used was a start-split, zero-length OUT transaction with a

handshake. Summaries of the component parts are listed in Figure 30-62. The component times were derived from the protocol timings defined in the USB Specification Revision 2.0.

Table 30-62. Example Worse-case Transaction Timing Components

Component	Bit time	Byte Time	Explanation
Split Token	76	9.5	Split token as defined in USB core specification. Includes sync, token, eop, etc.
Host 2 Host IPG	88	11	Number of bit times required between consecutive host packets.
Token	67	8.375	Token as defined in USB core specification. Includes sync, token, eop, etc.
Host 2 Host IPG	88	11	Same as above
Data Packet (0 data bytes)	66.7	8.34	Zero-length data packet. Includes sync, PID, crc16, eop, etc.
Turnaround time	721	90.125	Time for packet initiator (Host) to see the beginning of a response to a transmitted packet.
Handshake packet	48	6	Handshake packet as defined in USB core specification. Includes sync, PID, eop, etc.
		144	Total

The exact details of the function (f(x)) are up to the particular implementation. However, it should be obvious that the goal is to minimize the area under the curve between the approximation function and the *Last Start* curve, without dipping below the *LastStart* line, while at the same time keeping the check as simple as possible for hardware implementation. The f(x) in Figure was constructed using the following pseudo-code test on each transaction size data point. This algorithm assumes that the host controller keeps track of the remaining bits in the frame.

```
Algorithm CheckTransactionWillFit (MaximumPacketSize, HC_BytesLeftInFrame)

Begin

        Local Temp = MaximumPacketSize + 192

        Local rvalue = TRUE


        If MaximumPacketSize >= 128 then

                Temp += 128

        End If


        If Temp > HC_BytesLeftInFrame then

                Rvalue = FALSE

        End If
```

```
        Return rvalue
End
```

This algorithm takes two inputs, the current maximum packet size of the transaction and a hardware counter of the number of bytes left in the current micro-frame. It unconditionally adds a simple constant of 192 to the maximum packet size to account for a first-order effect of transaction overhead and bit stuffing. If the transaction size is greater than or equal to 128 bytes, then an additional constant of 128 is added to the running sum to account for the additional worst-case bit stuffing of payloads larger than 128. An inflection point was inserted at 128 because the *f*(*x*) plot was getting *close* to the *LastStart* line.

30.8.3.5 Periodic Schedule Frame Boundaries vs Bus Frame Boundaries

The USB Specification Revision 2.0 requires that the frame boundaries (SOF frame number changes) of the high-speed bus and the full- and low-speed bus(s) below USB 2.0 Hubs be strictly aligned. Super-imposed on this requirement is that USB 2.0 Hubs manage full- and low-speed transactions via a micro-frame pipeline (see start- (SS) and complete- (CS) splits illustrated in Figure 30-55). A simple, direct projection of the frame boundary model into the host controller interface schedule architecture creates tension (complexity for both hardware and software) between the frame boundaries and the scheduling mechanisms required to service the full- and low-speed transaction translator periodic pipelines.

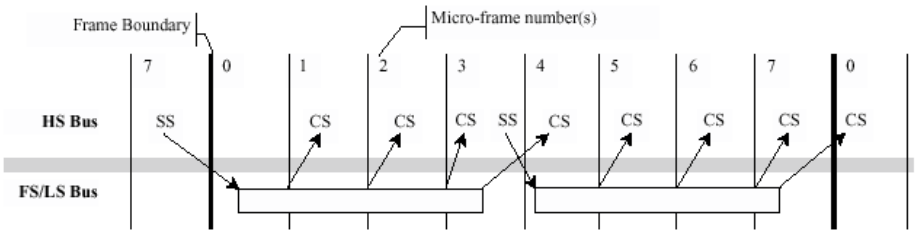


Figure 30-55. Frame Boundary Relationship between HS bus and FS/LS Bus

The simple projection, as Figure 30-55 illustrates, introduces frame-boundary wrap conditions for scheduling on both the beginning and end of a frame. In order to reduce the complexity for hardware and software, the host controller is required to implement a one micro-frame phase shift for its view of frame boundaries. The phase shift eliminates the beginning of frame and frame-wrap scheduling boundary conditions.

The implementation of this phase shift requires that the host controller use one register value for accessing the periodic frame list and another value for the frame number value included in the SOF token. These two values are separate, but tightly coupled. The periodic frame list is accessed via the Frame List Index Register (FRINDEX) documented in Section FRINDEX and initially illustrated in Section Schedule Traversal Rules . Bits FRINDEX[2:0], represent the micro-frame number. The SOF value is coupled to the value of FRINDEX[13:3]. Both FRINDEX[13:3] and the SOF value are increment based on FRINDEX[2:0]. It is required that the SOF value be delayed from the FRINDEX value by one micro-frame. The one micro-frame delay yields host controller periodic schedule and bus frame boundary relationship as illustrated in Figure 30-56. This adjustment allows software to trivially schedule the

periodic start and complete-split transactions for full-and low-speed periodic endpoints, using the natural alignment of the periodic schedule interface. The reasons for selecting this phase-shift are beyond the scope of this specification.

Figure 30-56 illustrates how periodic schedule data structures relate to schedule frame boundaries and bus frame boundaries. To aid the presentation, two terms are defined. The host controller's view of the 1-millisecond boundaries is called *H-Frames*. The high-speed bus's view of the 1-millisecond boundaries is called *B-Frames*.

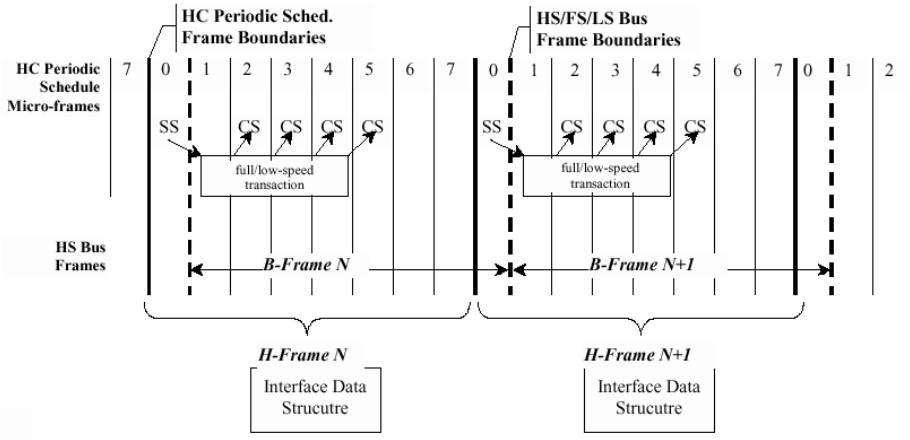


Figure 30-56. Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

H-Frame boundaries for the host controller correspond to increments of FRINDEX[13:3]. Micro-frame numbers for the *H-Frame* are tracked by FRINDEX[2:0]. *B-Frame* boundaries are visible on the high-speed bus via changes in the SOF token's frame number. Micro-frame numbers on the high-speed bus are only derived from the SOF token's frame number (that is, the high-speed bus will see eight SOFs with the same frame number value). *H-Frames* and *B-Frames* have the fixed relationship (that is, *B-Frames* lag *H-Frames* by one micro-frame time) illustrated in Figure 30-56 . The host controller's periodic schedule is naturally aligned to *H-Frames*. Software schedules transactions for full- and low-speed periodic endpoints relative the *H-Frames*. The result is these transactions execute on the high-speed bus at exactly the right time for the USB 2.0 Hub periodic pipeline. As described in Section FRINDEX, the SOF Value can be implemented as a shadow register (in this example, called SOFV), which lags the FRINDEX register bits [13:3] by one micro-frame count. Table 30-63 illustrates the required relationship between the value of FRINDEX and the value of SOFV. This lag behavior can be accomplished by incrementing FRINDEX[13:3] based on carry-out on the 7 to 0 increment of FRINDEX[2:0] and incrementing SOFV based on the transition of 0 to 1 of FRINDEX[2:0].

Software is allowed to write to FRINDEX. Section FRINDEX provides the requirements that software should adhere when writing a new value in FRINDEX.

Table 30-63. Operation of FRINDEX and SOFV (SOF Value Register)

	Current			Next	
FRINDEX[F]	SOFV	FRINDEX[mF]	FRINDEX[F]	SOFV	FRINDEX[mF]
N	N	111b	N+1	N	000b
N+1	N	000b	N+1	N+1	001b
N+1	N+1	001b	N+1	N+1	010b
N+1	N+1	010b	N+1	N+1	011b
N+1	N+1	011b	N+1	N+1	100b
N+1	N+1	100b	N+1	N+1	101b
N+1	N+1	101b	N+1	N+1	110b
N+1	N+1	110b	N+1	N+1	111b

Where [F] = [13:3]; [μF] = [2:0]

30.8.3.6 Periodic Schedule

The periodic schedule traversal is enabled or disabled via the *Periodic Schedule Enable* bit in the USBCMD register. If the *Periodic Schedule Enable* bit is set to a zero, then the host controller simply does not try to access the periodic frame list via the *PERIODICLISTBASE* register. Likewise, when the *Periodic Schedule Enable* bit is a one, then the host controller does use the *PERIODICLISTBASE* register to traverse the periodic schedule. The host controller will not react to modifications to the *Periodic Schedule Enable* immediately. In order to eliminate conflicts with split transactions, the host controller evaluates the *Periodic Schedule Enable* bit only when FRINDEX[2:0] is zero. System software must not disable the periodic schedule if the schedule contains an active split transaction work item that spans the 000b micro-frame. These work items must be removed from the schedule before the *Periodic Schedule Enable* bit is written to a zero. The *Periodic Schedule Status* bit in the USBSTS register indicates status of the periodic schedule. System software enables (or disables) the periodic schedule by writing a one (or zero) to the *Periodic Schedule Enable* bit in the USBCMD register. Software then can poll the *Periodic Schedule Status* bit to determine when the periodic schedule has made the desired transition. Software must not modify the *Periodic Schedule Enable* bit unless the value of the *Periodic Schedule Enable* bit equals that of the *Periodic Schedule Status* bit.

The periodic schedule is used to manage all isochronous and interrupt transfer streams. The base of the periodic schedule is the periodic frame list. Software links schedule data structures to the periodic frame list to produce a graph of scheduled data structures. The graph represents an appropriate sequence of transactions on the USB. Figure illustrates isochronous transfers (using iTDs and siTDs) with a period of one are linked directly to the periodic frame list. Interrupt transfers (are managed with queue heads) and isochronous streams with periods other than one are linked following the period-one iTD/siTDs. Interrupt queue heads are linked into the frame list ordered by poll rate. Longer poll rates are linked first (for example, closest to the periodic frame list), followed by shorter poll rates, with queue heads with a poll rate of one, on the very end.

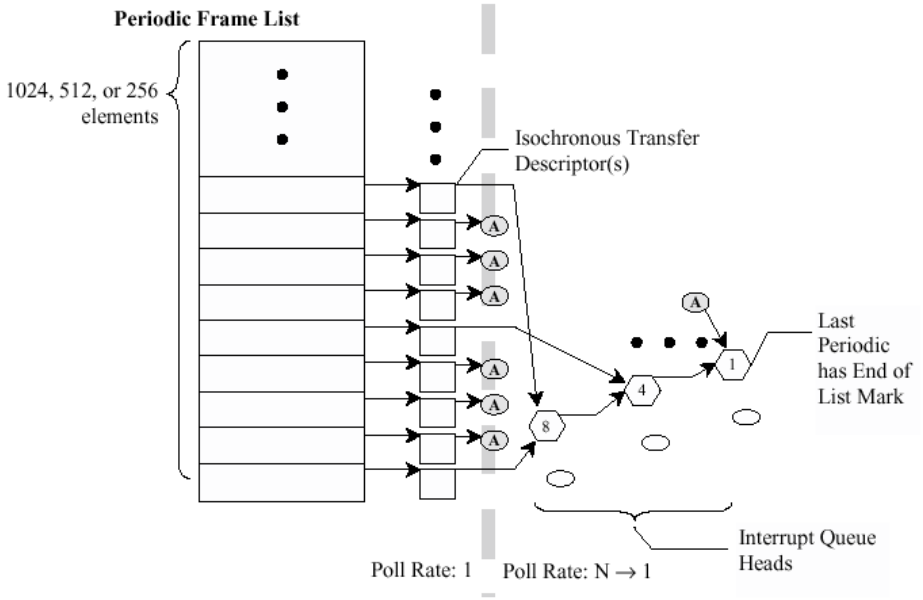


Figure 30-57. Example Periodic Schedule

30.8.3.7 Managing Isochronous Transfers Using iTDs

The structure of an iTD is presented in Isochronous (High-Speed) Transfer Descriptor (iTD). There are four distinct sections to an iTD:

- The first field is the *Next Link Pointer.* This field is for schedule linkage purposes only;
- Transaction description array. This area is an eight-element array. Each element represents control and status information for one micro-frame's worth of transactions for a single high-speed isochronous endpoint.
- The buffer page pointer array is a 7-element array of physical memory pointers to data buffers. These are 4K aligned pointers to physical memory.
- Endpoint capabilities. This area utilizes the unused low-order 12 bits of the buffer page pointer array. The fields in this area are used across all transactions executed for this iTD, including endpoint addressing, transfer direction, maximum packet size and high-bandwidth multiplier.

30.8.3.7.1 Host Controller Operational Model for iTDs

The host controller uses FRINDEX register bits [12:3] to index into the periodic frame list. This means that the host controller visits each frame list element eight consecutive times before incrementing to the next periodic frame list element. Each iTD contains eight transaction descriptions, which map directly to FRINDEX register bits [2:0]. Each iTD can span 8 micro-frames worth of transactions. When the host controller fetches an iTD, it uses FRINDEX register bits [2:0] to index into the transaction description array. If the *active* bit in the *Status* field of the indexed transaction description is set to zero, the host controller ignores the iTD and follows the *Next* pointer to the next schedule data structure.

When the indexed *active* bit is a one the host controller continues to parse the iTD. It stores the indexed transaction description and the general endpoint information (device address, endpoint number, maximum

packet size, etc.). It also uses the *Page Select (PG)* field to index the *buffer pointer* array, storing the selected *buffer pointer* and the next sequential *buffer pointer.* For example, if *PG* field is a 0, then the host controller will store Page 0 and Page 1.

The host controller constructs a physical data buffer address by concatenating the current buffer pointer (as selected using the current transaction description's *PG* field) and the transaction description's *Transaction Offset* field. The host controller uses the endpoint addressing information and *I/O-bit* to execute a transaction to the appropriate endpoint. When the transaction is complete, the host controller clears the active bit and writes back any additional status information to the *Status* field in the currently selected transaction description.

The data buffer associated with the iTD must be virtually contiguous memory. Seven page pointers are provided to support eight high-bandwidth transactions regardless of the starting packet's offset alignment into the first page. A starting buffer pointer (physical memory address) is constructed by concatenating the page pointer (example: page 0 pointer) selected by the active transaction descriptions' *PG* (example value: 00B) field with the transaction offset field. As the transaction moves data, the host controller must detect when an increment of the current buffer pointer will cross a page boundary. When this occurs the host controller simply replaces the current buffer pointer's page portion with the next page pointer (example: page 1 pointer) and continues to move data. The size of each bus transaction is determined by the value in the *Maximum Packet Size* field. An iTD supports high-bandwidth pipes via the *Mult* (multiplier) field. When the *Mult* field is 1, 2, or 3, the host controller executes the specified number of *Maximum Packet* sized bus transactions for the endpoint in the current micro-frame. In other words, the *Mult* field represents a transaction count for the endpoint in the current micro-frame. If the *Mult* field is zero, the operation of the host controller is undefined. The transfer description is used to service all transactions indicated by the *Mult* field.

For OUT transfers, the value of the *Transaction X Length* field represents the total bytes to be sent during the micro-frame. The *Mult* field must be set by software to be consistent with *Transaction X Length* and *Maximum Packet Size.* The host controller will send the bytes in *Maximum Packet Size*d portions. After each transaction, the host controller decrements it's local copy of *Transaction X Length* by *Maximum Packet Size*. The number of bytes the host controller sends is always *Maximum Packet Size* or *Transaction X Length*, whichever is less. The host controller advances the transfer state in the transfer description, updates the appropriate record in the iTD and moves to the next schedule data structure. The maximum sized transaction supported is 3 x 1024 bytes.

For IN transfers, the host controller issues *Mult* transactions. It is assumed that software has properly initialized the iTD to accommodate all of the possible data. During each IN transaction, the host controller must use *Maximum Packet Size* to detect packet babble errors. The host controller keeps the sum of bytes received in the *Transaction X Length* field. After all transactions for the endpoint have completed for the micro-frame, *Transaction X Length* contains the total bytes received. If the final value of *Transaction X Length* is less than the value of *Maximum Packet Size*, then less data than was allowed for was received from the associated endpoint. This *short packet* condition does not set the *USBINT* bit in the USBSTS register to a one. The host controller will not detect this condition. If the device sends more than *Transaction X Length* or *Maximum Packet Size* bytes (whichever is less), then the host controller will set the *Babble Detected* bit to a one and set the *Active* bit to a zero. Note, that the host controller is not required to update the iTD field *Transaction X Length* in this error scenario. If the *Mult* field is greater than one,

then the host controller will automatically execute the value of *Mult* transactions. The host controller will not execute all *Mult* transactions if:

- The endpoint is an OUT and *Transaction X Length* goes to zero before all the *Mult* transactions have executed (ran out of data), or
- The endpoint is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before *Mult* transactions have been executed. The end of micro-frame may occur before all of the transaction opportunities have been executed. When this happens, the transfer state of the transfer description is advanced to reflect the progress that was made, the result written back to the iTD and the host controller proceeds to processing the next micro-frame. Refer to Appendix D for a table summary of the host controller required behavior for all the high-bandwidth transaction cases.

30.8.3.7.2 Software Operational Model for iTDs

A client buffer request to an isochronous endpoint may span 1 to N micro-frames. When N is larger than one, system software may have to use multiple iTDs to read or write data with the buffer (if N is larger than eight, it must use more than one iTD).

Figure illustrates the simple model of how a client buffer is mapped by system software to the periodic schedule (that is, the periodic frame list and a set of iTDs). On the right is the client description of its request. The description includes a buffer base address plus additional annotations to identify which portions of the buffer should be used with each bus transaction. In the middle is the iTD data structures used by the system software to service the client request. Each iTD can be initialized to service up to 24 transactions, organized into eight groups of up to three transactions each. Each group maps to one micro-frame's worth of transactions. The EHCI controller does not provide per-transaction results within a micro-frame. It treats the per-micro-frame transactions as a single logical transfer. On the left is the host controller’s frame list. System software establishes references from the appropriate locations in the frame list to each of the appropriate iTDs. If the buffer is large, then system software can use a small set of iTDs to service the entire buffer. System software can activate the transaction description records (contained in each iTD) in any pattern required for the particular data stream.

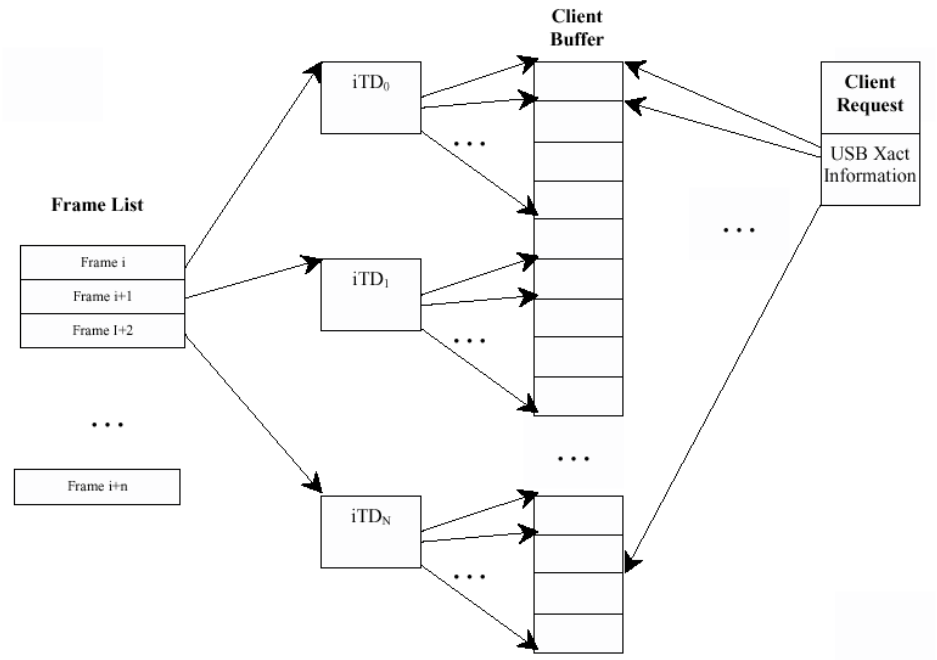


Figure 30-58. Example Association of iTDs to Client Request Buffer

As noted above, the client request includes a pointer to the base of the buffer and offsets into the buffer to annotate which buffer sections are to be used on each bus transaction that occurs on this endpoint. System software must initialize each transaction description in an iTD to ensure it uses the correct portion of the client buffer. For example, for each transaction description, the *PG* field is set to index the correct physical buffer page pointer and the *Transaction Offset* field is set relative to the correct buffer pointer page (for example, the same one referenced by the *PG* field). When the host controller executes a transaction it selects a transaction description record based on FRINDEX[2:0]. It then uses the current *Page Buffer Pointer* (as selected by the *PG* field) and concatenates to the *transaction offset* field. The result is a starting buffer address for the transaction. As the host controller moves data for the transaction, it must watch for a page wrap condition and properly advance to the next available *Page Buffer Pointer.* System software must not use the Page 6 buffer pointer in a transaction description where the length of the transfer will wrap a page boundary. Doing so will yield undefined behavior. The host controller hardware is not required to 'alias' the page selector to page zero. USB 2.0 isochronous endpoints can specify a period greater than one. Software can achieve the appropriate scheduling by linking iTDs into the appropriate frames (relative to the frame list) and by setting appropriate transaction description elements active bits to a one.

Periodic Scheduling Threshold

The *Isochronous Scheduling Threshold* field in the HCCPARAMS capability register is an indicator to system software as to how the host controller pre-fetches and effectively caches schedule data structures. It is used by system software when adding isochronous work items to the periodic schedule. The value of this field indicates to system software the minimum distance it can update isochronous data (relative to the current location of the host controller execution in the periodic list) and still have the host controller process them.

The iTD and siTD data structures each describe 8 micro-frames worth of transactions. The host controller is allowed to cache one (or more) of these data structures in order to reduce memory traffic. There are three basic caching models that account for the fact the isochronous data structures span 8 micro-frames. The three caching models are: no caching, micro-frame caching and frame caching.

When software is adding new isochronous transactions to the schedule, it always performs a read of the FRINDEX register to determine the current frame and micro-frame the host controller is currently executing. Of course, there is no information about where in the micro-frame the host controller is, so a constant uncertainty-factor of one micro-frame has to be assumed. Combining the knowledge of where the host controller is executing with the knowledge of the caching model allows the definition of simple algorithms for how closely software can reliably work to the executing host controller.

No caching is indicated with a value of zero in the *Isochronous Scheduling Threshold* field. The host controller may pre-fetch data structures during a periodic schedule traversal (per micro-frame) but will always dump any accumulated schedule state at the end of the micro-frame. At the appropriate time relative to the beginning of every micro-frame, the host controller always begins schedule traversal from the frame list. Software can use the value of the FRINDEX register (plus the constant 1 uncertainty-factor) to determine the approximate position of the executing host controller. When no caching is selected, software can add an isochronous transaction as near as 2 micro-frames in front of the current executing position of the host controller.

Frame caching is indicated with a non-zero value in bit [7] of the *Isochronous Scheduling Threshold* field. In the frame-caching model, system software assumes that the host controller caches one (or more) isochronous data structures for an entire frame (8 micro-frames). Software uses the value of the FRINDEX register (plus the constant 1 uncertainty) to determine the *current* micro-frame/frame (assume modulo 8 arithmetic in adding the constant 1 to the micro-frame number). For any current frame N, if the current micro-frame is 0 to 6, then software can safely add isochronous transactions to Frame N + 1. If the current micro-frame is 7, then software can add isochronous transactions to Frame N + 2.

Micro-frame caching is indicated with a non-zero value in the least-significant 3 bits of the *Isochronous Scheduling Threshold* field. System software assumes the host controller caches one or more periodic data structures for the number of micro-frames indicated in the *Isochronous Scheduling Threshold* field. For example, if the count value were 2, then the host controller keeps a *window* of 2 micro-frames worth of state (current micro-frame, plus the next) on-chip. On each micro-frame boundary, the host controller releases the current micro-frame state and begins accumulating the next micro-frame state.

30.8.3.8 Asynchronous Schedule

The Asynchronous schedule traversal is enabled or disabled via the *Asynchronous Schedule Enable* bit in the USBCMD register. If the *Asynchronous Schedule Enable* bit is set to a zero, then the host controller simply does not try to access the asynchronous schedule via the *ASYNCLISTADDR* register. Likewise, when the *Asynchronous Schedule Enable* bit is a one, then the host controller does use the *ASYNCLISTADDR* register to traverse the asynchronous schedule. Modifications to the *Asynchronous Schedule Enable* bit are not necessarily immediate. Rather the new value of the bit will only be taken into consideration the next time the host controller needs to use the value of the *ASYNCLISTADDR* register to get the next queue head.

The *Asynchronous Schedule Status* bit in the USBSTS register indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing a one (or zero) to the *Asynchronous Schedule Enable* bit in the USBCMD register. Software then can poll the *Asynchronous Schedule Status* bit to determine when the asynchronous schedule has made the desired transition. Software must not modify the *Asynchronous Schedule Enable* bit unless the value of the *Asynchronous Schedule Enable* bit equals that of the *Asynchronous Schedule Status* bit.

The asynchronous schedule is used to manage all Control and Bulk transfers. Control and Bulk transfers are managed using queue head data structures. The asynchronous schedule is based at the *ASYNCLISTADDR* register. The default value of the *ASYNCLISTADDR* register after reset is undefined and the schedule is disabled when the *Asynchronous Schedule Enable* bit is a zero.

Software may only write this register with defined results when the schedule is disabled. for example, *Asynchronous Schedule Enable* bit in the USBCMD and the *Asynchronous Schedule Status* bit in the USBSTS register are zero. System software enables execution from the asynchronous schedule by writing a valid memory address (of a queue head) into this register. Then software enables the asynchronous schedule by setting the *Asynchronous Schedule Enable* bit is set to one. The asynchronous schedule is actually enabled when the *Asynchronous Schedule Status* bit is a one.

When the host controller begins servicing the asynchronous schedule, it begins by using the value of the *ASYNCLISTADDR* register. It reads the first referenced data structure and begins executing transactions and traversing the linked list as appropriate. When the host controller “completes” processing the asynchronous schedule, it retains the value of the last accessed queue head's horizontal pointer in the *ASYNCLISTADDR* register. Next time the asynchronous schedule is accessed, this is the first data structure that will be serviced. This provides round-robin fairness for processing the asynchronous schedule.

A host controller “completes” processing the asynchronous schedule when one of the following events occur:

- The end of a micro-frame occurs.
- The host controller detects an empty list condition (that is, see Section Empty Asynchronous Schedule Detection)
- The schedule has been disabled via the *Asynchronous Schedule Enable* bit in the USBCMD register.

The queue heads in the asynchronous list are linked into a simple circular list as shown in Figure 30-53. Queue head data structures are the only valid data structures that may be linked into the asynchronous schedule. An isochronous transfer descriptor (iTD or siTD) in the asynchronous schedule yields undefined results.

The maximum packet size field in a queue head is sized to accommodate the use of this data structure for all non-isochronous transfer types. The USB Specification, Revision 2.0 specifies the maximum packet sizes for all transfer types and transfer speeds. System software should always parameterize the queue head data structures according to the core specification requirements.

30.8.3.8.1 Adding Queue Heads to Asynchronous Schedule

This is a software requirement section. There are two independent events for adding queue heads to the asynchronous schedule. The first is the initial activation of the asynchronous list. The second is inserting a new queue head into an activated asynchronous list.

Activation of the list is simple. System software writes the physical memory address of a queue head into the *ASYNCLISTADDR* register, then enables the list by setting the *Asynchronous Schedule Enable* bit in the USBCMD register to a one.

When inserting a queue head into an active list, software must ensure that the schedule is always coherent from the host controllers' point of view. This means that the system software must ensure that all queue head pointer fields are valid. For example qTD pointers have *T-Bit*s set to a one or reference valid qTDs and the *Horizontal Pointer* references a valid queue head data structure. The following algorithm represents the functional requirements:

```
InsertQueueHead (pQHeadCurrent, pQueueHeadNew)

        --
        -- Requirement: all inputs must be properly initialized.
        --
        -- pQHeadCurrent is a pointer to a queue head that is
        -- already in the active list
        -- pQHeadNew is a pointer to the queue head to be added
        --
        -- This algorithm links a new queue head into a existing
        -- list
        --
        pQueueHeadNew.HorizontalPointer = pQueueHeadCurrent.HorizontalPointer


        pQueueHeadCurrent.HorizontalPointer = physicalAddressOf(pQueueHeadNew)


End InsertQueueHead
```

30.8.3.8.2 Removing Queue Heads from Asynchronous Schedule

This is a software requirement section. There are two independent events for removing queue heads from the asynchronous schedule. The first is shutting down (deactivating) the asynchronous list. The second is extracting a single queue head from an activated list. Software deactivates the asynchronous schedule by setting the *Asynchronous Schedule Enable* bit in the USBCMD register to a zero. Software can determine when the list is idle when the *Asynchronous Schedule Status* bit in the USBSTS register is a zero. The normal mode of operation is that software removes queue heads from the asynchronous schedule without

shutting it down. Software must not remove an active queue head from the schedule. Software should first deactivate all active qTDs, wait for the queue head to go inactive, then remove the queue head from the asynchronous list. Software removes a queue head from the asynchronous list via the following algorithm. As illustrated, the unlinking is quite easy. Software merely must ensure all of the link pointers reachable by the host controller are kept consistent.

```
UnlinkQueueHead (pQHeadPrevious, pQueueHeadToUnlink, pQHeadNext)

        --
        -- Requirement: all inputs must be properly initialized.
        --
        -- pQHeadPrevious is a pointer to a queue head that
        -- references the queue head to remove
        -- pQHeadToUnlink is a pointer to the queue head to be
        -- removed
        -- pQheadNext is a pointer to a queue head still in the
        -- schedule. Software provides this pointer with the
        -- following strict rules:
        --      if the host software is one queue head, then
        --      pQHeadNext must be the same as
        --      QueueheadToUnlink.HorizontalPointer. If the host
        --      software is unlinking a consecutive series of
        --      queue heads, QHeadNext must be set by software to
        --      the  queue head remaining in the schedule.
        --
        -- This algorithm unlinks a queue head from a circular list
        --
        pQueueHeadPrevious.HorizontalPointer = pQueueHeadToUnlink.HorizontalPointer
pQueueHeadToUnlink.HorizontalPointer = pQHeadNext

End UnlinkQueueHead
```

If software removes the queue head with the *H-bit* set to a one, it must select another queue head still linked into the schedule and set its *H-bit* to a one. This should be completed before removing the queue head. The requirement is that software keep one queue head in the asynchronous schedule, with its *H-bit* set to a one. At the point software has removed one or more queue heads from the asynchronous schedule, it is unknown whether the host controller has a cached pointer to them. Similarly, it is unknown how long the host controller might retain the cached information, as it is implementation dependent and may be affected by the actual dynamics of the schedule load. Therefore, once software has removed a queue head from the

asynchronous list, it must retain the coherency of the queue head (link pointers, etc.). It cannot disturb the removed queue heads until it knows that the host controller does not have a local copy of a pointer to any of the removed data structures.

The method software uses to determine when it is safe to modify a removed queue head is to handshake with the host controller. The handshake mechanism allows software to remove items from the asynchronous schedule, then execute a simple, lightweight handshake that is used by software as a key that it can free (or reuse) the memory associated the data structures it has removed from the asynchronous schedule.

The handshake is implemented with three bits in the host controller. The first bit is a command bit (*Interrupt on Async Advance Doorbell* bit in the USBCMD register) that allows software to inform the host controller that something has been removed from its asynchronous schedule. The second bit is a status bit (*Interrupt on Async Advance* bit in the USBSTS register) that the host controller sets after it has released all on-chip state that may potentially reference one of the data structures just removed. When the host controller sets this status bit to a one, it also sets the command bit to a zero. The third bit is an interrupt enable (*Interrupt on Async Advance* bit in the USBINTR register) that is matched with the status bit. If the status bit is a one and the interrupt enable bit is a one, then the host controller will assert a hardware interrupt.

Figure 30-59 illustrates a general example. In this example, consecutive queue heads (B and C) are unlinked from the schedule using the algorithm above. Before the unlink operation, the host controller has a copy of queue head A.

The unlink algorithm requires that as software unlinks each queue head, the unlinked queue head is loaded with the address of a queue head that will remain in the asynchronous schedule.

When the host controller observes that doorbell bit being set to a one, it makes a note of the local reachable schedule information. In this example, the local reachable schedule information includes both queue heads (A and B). It is sufficient that the host controller can set the status bit (and clear the doorbell bit) as soon as it has traversed beyond current reachable schedule information (that is, traversed beyond queue head (B) in this example).

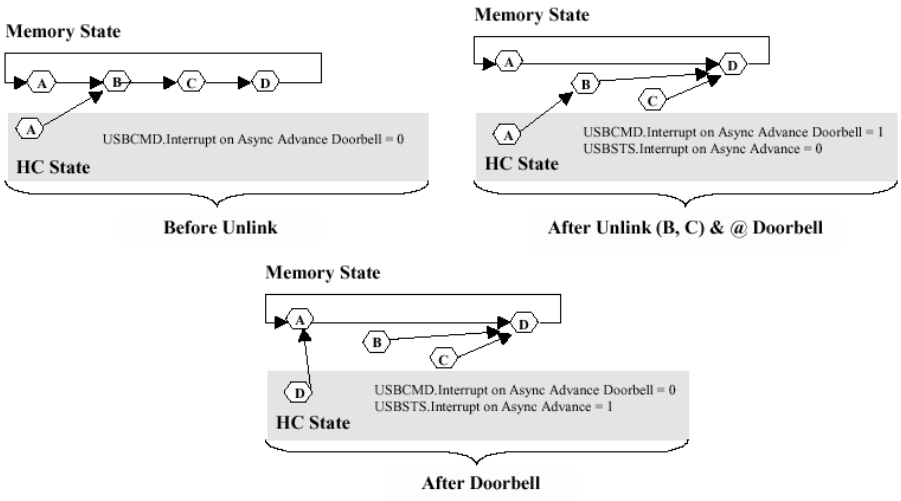


Figure 30-59. Generic Queue Head Unlink Scenario

Alternatively, a host controller implementation is allowed to traverse the entire asynchronous schedule list (for example, observed the head of the queue (twice)) before setting the *Advance on Async* status bit to a one.

Software may re-use the memory associated with the removed queue heads after it observes the *Interrupt on Async Advance* status bit is set to a one, following assertion of the doorbell. Software should acknowledge the *Interrupt on Async Advance* status as indicated in the USBSTS register, before using the doorbell handshake again.

30.8.3.8.3 Empty Asynchronous Schedule Detection

The Enhanced Host Controller Interface uses two bits to detect when the asynchronous schedule is empty. The queue head data structure (see Figure 30-48) defines an *H-bit* in the queue head, which allows software to mark a queue head as being the *head* of the reclaim list. The Enhanced Host Controller Interface also keeps a 1-bit flag in the USBSTS register (*Reclamation*) that is set to a zero when the Enhanced Interface Host Controller observes a queue head with the H-bit set to a one. The reclamation flag in the status register is set to one when any USB transaction from the asynchronous schedule is executed (or whenever the asynchronous schedule starts, see Section Asynchronous Schedule Traversal : Start Event).

If the Enhanced Host Controller Interface ever encounters an *H-bit* of one and a *Reclamation* bit of zero, the EHCI controller simply stops traversal of the asynchronous schedule.

A example illustrating the H-bit in a schedule is illustrated in Figure 30-60.

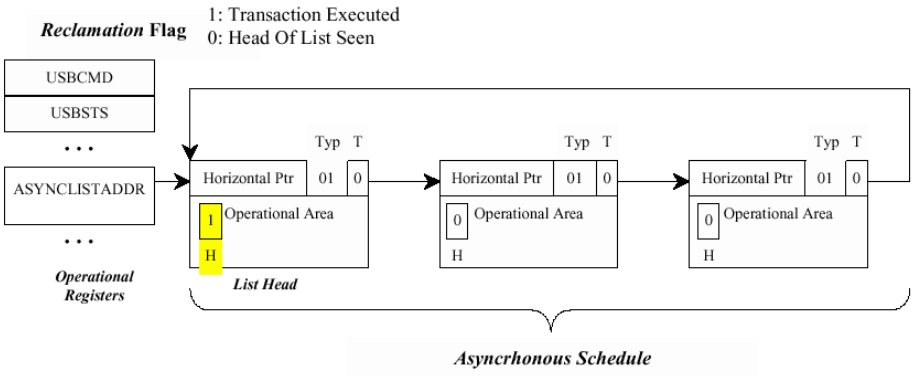


Figure 30-60. Asynchronous Schedule List w/Annotation to Mark Head of List

Software must ensure there is at most one queue head with the *H-bit* set to a one, and that it is always coherent with respect to the schedule.

30.8.3.8.4 Restarting Asynchronous Schedule Before EOF

There are many situations where the host controller will detect an empty list *long* before the end of the micro-frame. It is important to remember that under many circumstances the schedule traversal has stopped due to Nak/Nyet responses from all endpoints.

An example of particular interest is when a start-split for a bulk endpoint occurs early in the micro-frame. Given the EHCI simple traversal rules, the complete-split for that transaction may Nak/Nyet out very

quickly. If it is the only item in the schedule, then the host controller will cease traversal of the Asynchronous schedule very early in the micro-frame. In order to provide reasonable service to this endpoint, the host controller should issue the complete-split before the end of the current micro-frame, instead of waiting until the next micro-frame. When the reason for host controller idling asynchronous schedule traversal is because of empty list detection, it is mandatory the host controller implement a 'waking' method to resume traversal of the asynchronous schedule. An example method is described below.

Example Method for Restarting Asynchronous Schedule Traversal

The reason for idling the host controller when the list is empty is to keep the host controller from unnecessarily occupying too much memory bandwidth. The question is: *how long should the host controller stay idle before restarting*?

The answer in this example is based on deriving a manifest constant, which is the amount of time the host controller will stay idle before restarting traversal. In this example, the manifest constant is called *AsyncSchedSleepTime*, and has a value of 10µsec. The value is derived based on the analysis in Section Example Derivation for AsyncSchedSleepTime , The traversal algorithm is simple:

- Traverse the Asynchronous schedule until the either an End-Of-micro-Frame event occurs, or an empty list is detected. If the event is an End-of-micro-Frame, go attempt to traverse the Periodic schedule. If the event is an empty list, then set a sleep timer and go to a *schedule sleep* state.
- When the sleep timer expires, set working context to the Asynchronous Schedule start condition and go to *schedule active* state. The start context allows the HC to reload *Nakcnt* fields, etc. so the HC has a chance to run for more than one iteration through the schedule.

This process simply repeats itself each micro-frame. Figure 30-61 illustrates a sample state machine to manage the active and sleep states of the Asynchronous Schedule traversal policy. There are three states: Actively traversing the Asynchronous schedule, Sleeping, and Not Active. The last two are similar in terms of interaction with the Asynchronous schedule, but the Not Active state means that the host controller is busy with the Periodic schedule or the Asynchronous schedule is not enabled. The Sleeping state is specifically a special state where the host controller is just waiting for a period of time before resuming execution of the Asynchronous schedule.

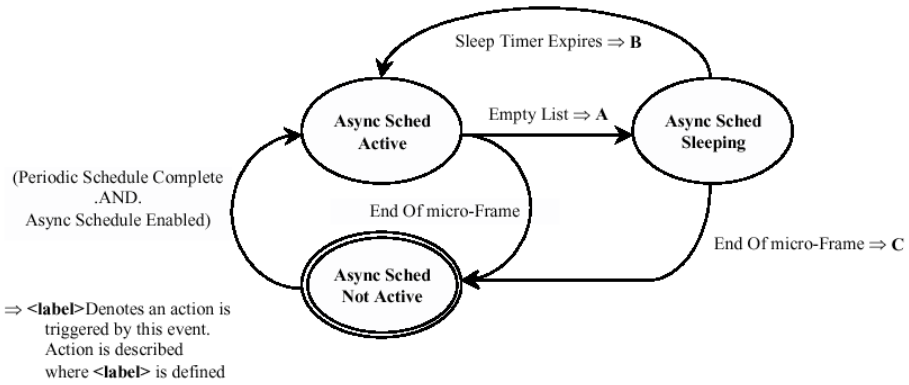


Figure 30-61. Example State Machine for Managing Asynchronous Schedule Traversal

The actions referred to in Figure 30-61 are defined in Table 30-64.

Table 30-64. Asynchronous Schedule SM Transition Actions

Action	Action Description Label
A	On detection of the empty list, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.
B	When the *AsynchronousTraversalSleepTimer* expires, the host controller sets the *Reclamation* bit in the USBSTS register to a one and moves the Nak Counter reload state machine to WaitForListHead (see Section Operational Model for Nak Counter).
C	The host controller cancels the sleep timer (*AsynchronousTraversalSleepTimer*).

Async Sched Not Active

This is the initial state of the traversal state machine after a host controller reset. The traversal state machine will not leave this state when the *Asynchronous Schedule Enable* bit in the USBCMD register is a zero.

This state is entered from Async Sched Active or Async Sched Sleeping states when the end-of-micro-frame event is detected.

Async Sched Active

This state is entered from the **Async Sched Not Active** state when the periodic schedule is not active. It is also entered from the **Async Sched Sleeping** states when the *AsyncrhonousTraversalSleepTimer* expires. On every transition into this state, the host controller sets the *Reclamation* bit in the USBSTS register to a one.

While in this state, the host controller will continually traverse the asynchronous schedule until either the end of micro-frame or an empty list condition is detected.

Async Sched Sleeping

The state is entered from the **Async Sched Active** state when a schedule empty condition is detected. On entry to this state, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.

Example Derivation for *AsyncSchedSleepTime*

The derivation is based on analysis of what work the host controller could be doing next. It assumes the host controller does not keep any state about what work is possibly pending in the asynchronous schedule. The schedule could contain any mix of the possible combinations of high- full- or low-speed control and bulk requests. Table 30-65 summarizes some of the typical 'next transactions' that could be in the schedule, and the amount of time (for example, *footprint* or *wall clock*) the transaction will take to complete.

Table 30-65. Typical Low-/Full-speed Transaction Times

Transaction Attributes		Footprint (time)	Description
Speed	HS	11.9 ms	Maximum foot print for a worst-case, full-sized bulk data transaction.
Size	512	9.45 ms	Maximum footprint for an approximate best-case, full-sized bulk data transaction.
Type	Bulk		
Speed	FS	~50 ms	Approximate typical for full-sized bulk data. An 8-byte low-speed is about 2x, or between 90 and 100 ms.
Size	64		
Type	Bulk		
Speed	FS	~12 ms	Approximate typical for 8-byte bulk/control (that is, setup)
Size	8		
Type	Cntrl		

A *AsyncSchedSleepTime* value of 10 µs provides a reasonable relaxation of the system memory load and still provides a good level of service for the various transfer types and payload sizes. For example, say we detect an empty list after issuing a start-split for a 64-byte full-speed bulk request. Assuming this is the only thing in the list, the host controller will get the results of the full-speed transaction from the hub during the fifth complete-split request. If the full-speed transaction was an IN and it nak'd, the 10µs sleep period would allow the host controller to get the NAK results on the first complete-split.

30.8.3.8.5 Asynchronous Schedule Traversal : *Start Event*

Once the HC has *idled* itself via the empty schedule detection (Section 0), it will naturally *activate* and begin processing from the Periodic Schedule at the beginning of each micro-frame. In addition, it may have idled itself early in a micro-frame. When this occurs (idles early in the micro-frame) the HC must occasionally *re-activate* during the micro-frame and traverse the asynchronous schedule to determine whether any progress can be made. The requirements and method for this restart are described in Section Restarting Asynchronous Schedule Before EOF . Asynchronous schedule *Start Event*s are defined to be:

- Whenever the host controller transitions from the periodic schedule to the asynchronous schedule. If the periodic schedule is disabled and the asynchronous schedule is enabled, then the beginning of the micro-frame is equivalent to the transition from the periodic schedule, or
- The asynchronous schedule traversal restarts from a sleeping state (see Section Restarting Asynchronous Schedule Before EOF).

30.8.3.8.6 Reclamation Status Bit (USBSTS Register)

The operation of the empty asynchronous schedule detection feature (Section Empty Asynchronous Schedule Detection) depends on the proper management of the *Reclamation* bit in the USBSTS register.

The host controller tests for an empty schedule just after it fetches a new queue head while traversing the asynchronous schedule (See Section Fetch Queue Head).

It is required that the host controller sets the *Reclamation* bit to a one whenever an asynchronous schedule traversal *Start Event*, as documented in Section Asynchronous Schedule Traversal : Start Event , occurs. The *Reclamation* bit is also set to a one whenever the host controller executes a transaction while traversing the asynchronous schedule (see Section Execute Transaction). The host controller sets the *Reclamation* bit to a zero whenever it finds a queue head with its *H-bit* set to a one. Software should only set a queue head's *H-bit* if the queue head is in the asynchronous schedule. If software sets the *H-bit* in an interrupt queue head to a one, the resulting behavior is undefined. The host controller may set the *Reclamation* bit to a zero when executing from the periodic schedule.

30.8.3.9 Operational Model for Nak Counter

This section describes the operational model for the *NakCnt* field defined in a queue head (see Section Queue Head). Software should not use this feature for interrupt queue heads. This rule is not required to be enforced by the host controller.

USB protocol has built-in flow control via the Nak response by a device. There are several scenarios, beyond the Ping feature, where an endpoint may naturally Nak or Nyet the majority of the time. An example is the host controller management of the split transaction protocol for control and bulk endpoints. All bulk endpoints (High- or Full-speed) are serviced via the same asynchronous schedule. The time between the *Start-split* transaction and the first *Complete-split* transaction could be very short (that is, like when the endpoint is the only one in the asynchronous schedule). The hub NYETs (effectively Naks) the *Complete-split* transaction until the classic transaction is complete. This could result in the host controller thrashing memory, repeatedly fetching the queue head and executing the transaction to the Hub, which will not complete until after the transaction on the classic bus completes.

There are two component fields in a queue head to support the throttling feature: a counter field (*NakCnt*), and a counter reload field (*RL*). *NakCnt* is used by the host controller as one of the criteria to determine whether or not to execute a transaction to the endpoint. There are two operational modes associated with this counter:

- **Not Used.** This mode is set when the *RL* field is zero. The host controller ignores the *NakCnt* field for any execution of transactions through a queue head with an *RL* field of zero. Software must use this selection for interrupt endpoints.
- **Nak Throttle Mode.** This mode is selected when the *RL* field is non-zero. In this mode, the value in the *NakCnt* field represents the maximum number of Nak or Nyet responses the host controller will tolerate on each endpoint. In this mode, the HC will decrement the *NakCnt* field based on the token/handshake criteria listed in Table 30-66 The host controller must reload *NakCnt* when the endpoint successfully moves data (for example, policy to reward device for moving data).

Table 30-66. NakCnt Field Adjustment Rules

Token	Handshake	
	Handshake NAK	NYET
IN/PING	decrement *NakCnt*	N/A (protocol error)
OUT	decrement *NakCnt*	No Action[1]Start
Split	decrement *NakCnt*	N/A (protocol error)
Complete Split	No Action	Decrement *NakCnt*

[1] Recommended behavior on this response is to reload NakCnt.

In summary, system software enables the counter by setting the reload field (*RL*) to a non-zero value. The host controller may execute a transaction if *NakCnt* is non-zero. The host controller will not execute a transaction if *NakCnt* is zero. The reload mechanism is described in detail in Section Nak Count Reload Control .

Note that when all queue heads in the Asynchronous Schedule either exhausts all transfers or all NakCnt's go to zero, then the host controller will detect an empty Asynchronous Schedule and idle schedule traversal (see Section Empty Asynchronous Schedule Detection).

Any time the host controller begins a new traversal of the Asynchronous Schedule, a *Start Event* is assumed, see Section Asynchronous Schedule Traversal : Start Event . Every time a Start-Event occurs, the Nak Count reload procedure is enabled.

30.8.3.9.1 Nak Count Reload Control

When the host controller reaches the *Execute Transaction* state for a queue head (meaning that it has an active operational state), it checks to determine whether the *NakCnt* field should be reloaded from *RL* (see Section Execute Transaction). If the answer is yes, then *RL* is copied into *NakCnt*. After the reload or if the reload is not active, the host controller evaluates whether to execute the transaction.

The host controller must reload nak counters (*NakCnt* see Figure 30-48) in queue heads during the first pass through the reclamation list after an asynchronous schedule Start Event (see Section Asynchronous Schedule Traversal : Start Event for the definition of the Start Event). The Asynchronous Schedule should have at most one queue head marked as the head (see Figure 30-60). Figure 30-62 illustrates an example state machine that satisfies the operational requirements of the host controller detecting the first pass through the Asynchronous Schedule. This state machine is maintained internal to the host controller and is only used to gate reloading of the nak counter during the queue head traversal state: **Execute Transaction** (Figure 30-62). The host controller does not perform the nak counter reload operation if the RL field (see Figure 30-48) is set to zero.

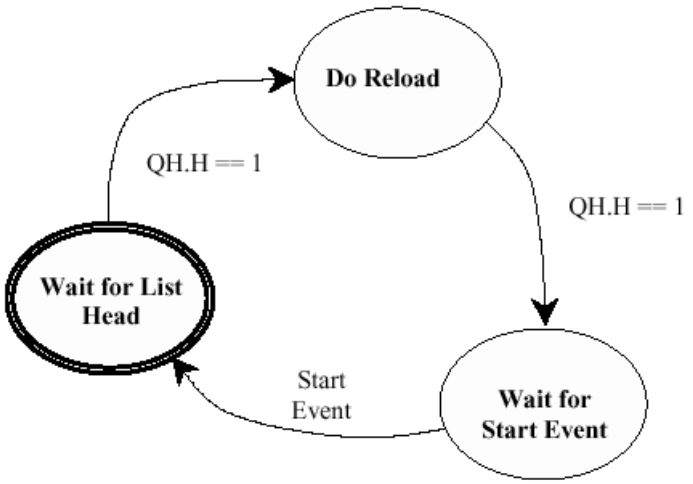


Figure 30-62. Example HC State Machine for Controlling Nak Counter Reloads

Wait for List Head

This is the initial state. The state machine enters this state from **Wait for Start Event** when a start event as defined in Section Asynchronous Schedule Traversal : Start Event occurs. The purpose of this state is to wait for the first observation of the head of the Asynchronous Schedule. This occurs when the host controller fetches a queue head whose *H-bit* is set to a one.

Do Reload

This state is entered from the **Wait for List Head** state when the host controller fetches a queue head with the *H-bit* set to a one. While in this state, the host controller will perform nak counter reloads for every queue head visited that has a non-zero nak reload value (*RL*) field.

Wait for Start Event

This state is entered from the *Do Reload* state when a queue head with the *H-bit* set to a one is fetched. While in this state, the host controller will not perform nak counter reloads.

30.8.3.10 Managing Control/Bulk/Interrupt Transfers via Queue Heads

This section presents an overview of how the host controller interacts with queuing data structures.

Queue heads use the Queue Element Transfer Descriptor (qTD) structure. One queue head is used to manage the data stream for one endpoint. The queue head structure contains static endpoint characteristics and capabilities. It also contains a working area from where individual bus transactions for an endpoint are executed (see Overlay area defined in Figure 30-48). Each qTD represents one or more bus transactions, which is defined in the context of this specification as a *transfer.*

The general processing model for the host controller's use of a queue head is simple:

- read a queue head,
- execute a transaction from the overlay area,

- write back the results of the transaction to the overlay area
- move to the next queue head.

If the host controller encounters errors during a transaction, the host controller will set one (or more) of the error reporting bits in the queue head's *Status* field. The *Status* field accumulates all errors encountered during the execution of a qTD (for example, the error bits in the queue head *Status* field are 'sticky' until the transfer (qTD) has completed). This state is always written back to the source qTD when the transfer is complete. On transfer (for example, buffer or halt conditions) boundaries, the host controller must auto-advance (without software intervention) to the next qTD. Additionally, the hardware must be able to halt the queue so no additional bus transactions will occur for the endpoint and the host controller will not advance the queue.

An example host controller operational state machine of a queue head traversal is illustrated in Figure 30-63. This state machine is a model for how a host controller should traverse a queue head. The host controller must be able to advance the queue from the *Fetch QH* state in order to avoid all hardware/software race conditions. This simple mechanism allows software to simply link qTDs to the queue head and *activate* them, then the host controller will always *find* them if/when they are reachable.

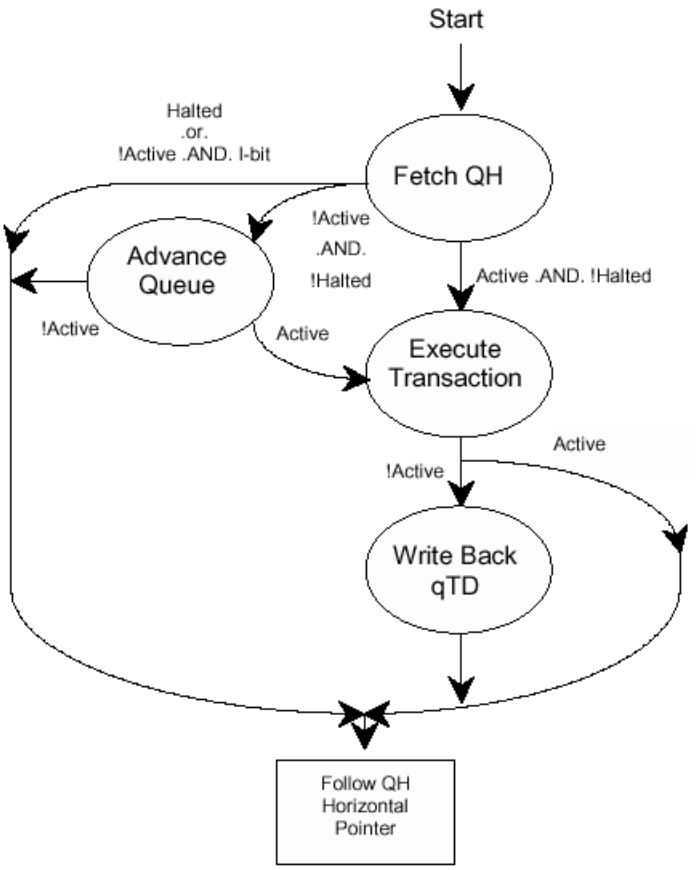


Figure 30-63. Host Controller Queue Head Traversal State Machine

This traversal state machine applies to all queue heads, regardless of transfer type or whether split transactions are required. The following sections describe each state. Each state description describes the entry criteria. The **Execute Transaction** state (Section Execute Transaction) describes the basic requirements for all endpoints. Sections Split Transactions for Asynchronous Transfers and Split Transaction Interrupt describe details of the required extensions to the **Execute Transaction** state for endpoints requiring split transactions.

Note: Prior to software placing a queue head into either the periodic or asynchronous list, software must ensure the queue head is properly initialized. Minimally, the queue head should be initialized to the following (see Section Queue Head for layout of a queue head):

- Valid static endpoint state
- For the very first use of a queue head, software may zero-out the queue head transfer overlay, then set the *Next qTD Pointer* field value to reference a valid qTD.

30.8.3.10.1 Fetch Queue Head

A queue head can be referenced from the physical address stored in the ASYNCLISTADDR Register (Section ASYNCLISTADDR; ENDPOINTLISTADDR). Additionally, it may be referenced from the *Next Link Pointer* field of an iTD, siTD, FSTN or another Queue Head. If the referencing link pointer has the *Typ* field set to indicate a queue head, it is assumed to reference a queue head structure as defined in Figure 30-48.

While in this state, the host controller performs operations to implement empty schedule detection (Section Empty Asynchronous Schedule Detection) and Nak Counter reloads (Section Operational Model for Nak Counter). After the queue head has been fetched, the host controller conducts the following queries for empty schedule detection:

- If queue head is not an interrupt queue head (that is, *S-mask* is a zero), and
- The *H-bit* is a one, and
- The *Reclamation* bit in the USBSTS register is a zero.

When these criteria are met, the host controller will stop traversing the asynchronous list (as described in Section Empty Asynchronous Schedule Detection). When the criteria are not met, the host controller continues schedule traversal. If the queue head is not an interrupt and the *H-bit* is a one and the *Reclamation* bit is a one, then the host controller sets the *Reclamation* bit in the USBSTS register to a zero before completing this state. The operations for reloading of the Nak Counter are described in detail in Section Operational Model for Nak Counter.

This state is complete when the queue head has been read on-chip.

30.8.3.10.2 Advance Queue

To advance the queue, the host controller must find the next qTD, adjust pointers, perform the overlay and write back the results to the queue head.

This state is entered from the **FetchQHD** state if the overlay *Active* and *Halt* bits are set to zero. On entry to this state, the host controller determines which next pointer to use to fetch a qTD, fetches a qTD and determines whether or not to perform an overlay. Note that if the *I-bit* is a one and the *Active* bit is a zero,

the host controller immediately skips processing of this queue head, exits this state and uses the horizontal pointer to the next schedule data structure. If the field *Bytes to Transfer* is not zero and the *T-bit* in the *Alternate Next qTD Pointer* is set to zero, then the host controller uses the *Alternate Next qTD Pointer.* Otherwise, the host controller uses the *Next qTD Pointer.* If *Next qTD Pointer*'s *T-bit* is set to a one, then the host controller exits this state and uses the horizontal pointer to the next schedule data structure.

Using the selected pointer the host controller fetches the referenced qTD. If the fetched qTD has it's *Active* bit set to a one, the host controller moves the pointer value used to reach the qTD (*Next* or *Alternate Next*) to the *Current qTD Pointer* field, then performs the overlay. If the fetched qTD has its *Active* bit set to a zero, the host controller aborts the queue advance and follows the queue head's horizontal pointer to the next schedule data structure. The host controller performs the overlay based on the following rules:

- The value of the data toggle (*dt*) field in the overlay area depends on the value of the *data toggle control* (*dtc*) bit (see Table 30-54).
- If the *EPS* field indicates the endpoint is a high-speed endpoint, the *Ping* state field is preserved by the host controller. The value of this field is not changed as a result of the overlay.
- *C-prog-mask* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *Frame Tag* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *NakCnt* field in the overlay area is loaded from the *RL* field in the queue head's Static Endpoint State.
- All other areas of the overlay are set by the incoming qTD.

The host controller exits this state when it has committed the write to the queue head.

30.8.3.10.3 Execute Transaction

The host controller enters this state from the **Fetch Queue Head** state only if the *Active* bit in *Status* field of the queue head is set to a one.

On entry to this state, the host controller executes a few pre-operations, then checks some pre-condition criteria before committing to executing a transaction for the queue head.

The pre-operations performed and the pre-condition criteria depend on whether the queue head is an interrupt endpoint. The host controller can determine that a queue head is an interrupt queue head when the queue head's *S-mask* field contains a non-zero value. It is the responsibility of software to ensure the *S-mask* field is appropriately initialized based on the transfer type. There are other criteria that must be met if the *EPS* field indicates that the endpoint is a low- or full-speed endpoint, see Sections Split Transactions for Asynchronous Transfers and Split Transaction Interrupt .

Interrupt Transfer Pre-condition Criteria

If the queue head is for an interrupt endpoint (for example, non-zero *S-mask* field), then the FRINDEX[2:0] field must identify a bit in the *S-mask* field that has a one in it. For example, an *S-mask* value of 00100000b would evaluate to true only when FRINDEX[2:0] is equal to 101b. If this condition is met then the host controller considers this queue head for a transaction**.**

Asynchronous Transfer Pre-Operations and Pre-Condition Criteria

If the queue head is not for an interrupt endpoint (for example, a zero *S-mask* field), then the host controller performs one pre-operation and then evaluates one pre-condition criteria: The pre-operation is:

Checks the Nak counter reload state (Section Operational Model for Nak Counter). It may be necessary for the host controller to reload the Nak Counter field. The reload is performed at this time.

The pre-condition evaluated is:

- Whether or not the *NakCnt* field has been reloaded, the host controller checks the value of the *NakCnt* field in the queue head. If *NakCnt* is non-zero, or if the *Reload Nak Counter* field is zero, then the host controller considers this queue head for a transaction.

Transfer Type Independent Pre-Operations

Regardless of the transfer type, the host controller always performs at least one pre-operation and evaluates one pre-condition. The pre-operation is:

- A host controller internal transaction (down) counter *qHTransactionCounter* is loaded from the queue head's *Mult* field. A host controller implementation is allowed to ignore this for queue heads on the asynchronous list. It is mandatory for interrupt queue heads. Software should ensure that the *Mult* field is set appropriately for the transfer type.

The pre-conditions evaluated are:

- The host controller determines whether there is enough time in the micro-frame to complete this transaction (see Section Transaction Fit - A Best-Fit Approximation Algorithm for an example evaluation method). If there is not enough time to complete the transaction, the host controller exits this state.
- If the value of *qHTransactionCounter* for an interrupt endpoint is zero, then the host controller exits this state.

When the pre-operations are complete and pre-conditions are met, the host controller sets the *Reclamation* bit in the USBSTS register to a one and then begins executing one or more transactions using the endpoint information in the queue head. The host controller iterates *qHTransactionCounter* times in this state executing transactions. After each transaction is executed, *qHTransactionCounter* is decremented by one. The host controller will exit this state when one of the following events occurs:

- The *qHTransactionCounter* decrements to zero, or
- The endpoint responds to the transaction with any handshake other than an ACK,[4] or
- The transaction experiences a transaction error, or
- The *Active* bit in the queue head goes to a zero, or
- There is not enough time in the micro-frame left to execute the next transaction (see Section Transaction Fit - A Best-Fit Approximation Algorithm for example method for implementing the frame boundary test).

[4] Note that for a high-bandwidth interrupt OUT endpoint, the host controller may optionally immediately retry the transaction if it fails.

The results of each transaction is recorded in the on-chip overlay area. If data was successfully moved during the transaction, the transfer state in the overlay area is advanced. To advance queue head's transfer state, the *Total Bytes to Transfer* field is decremented by the number of bytes moved in the transaction, the data toggle bit (*dt*) is toggled, the current page offset is advanced to the next appropriate value (for example, advanced by the number of bytes successfully moved), and the *C_Page* field is updated to the appropriate value (if necessary). See Section Buffer Pointer List Use for Data Streaming with qTDs .

Note that the *Total Bytes To Transfer* field may be zero when all the other criteria for executing a transaction are met. When this occurs, the host controller will execute a zero-length transaction to the endpoint. If the *PID_Code* field indicates an IN transaction and the device delivers data, the host controller will detect a packet babble condition, set the *babble* and *halted* bits in the *Status* field, set the *Active* bit to a zero, write back the results to the source qTD, then exit this state.

In the event an IN token receives a data PID mismatch response, the host controller must ignore the received data (for example, not advance the transfer state for the bytes received). Additionally, if the endpoint is an interrupt IN, then the host controller must record that the transaction occurred (for example, decrement *qHTransactionCounter*). It is recommended (but not required) the host controller continue executing transactions for this endpoint if the resultant value of *qHTransactionCounter* is greater than one.

If the response to the IN bus transaction is a Nak (or Nyet) and *RL* is non-zero, *NakCnt* is decremented by one. If *RL* is zero, then no write-back by the host controller is required (for a transaction receiving a Nak or Nyet response and the value of *CErr* did not change). Software should set the *RL* field to zero if the queue head is an interrupt endpoint. Host controller hardware is not required to enforce this rule or operation.

After the transaction has finished and the host controller has completed the post processing of the results (advancing the transfer state and possibly *NakCnt*, the host controller writes back the results of the transaction to the queue head's overlay area in main memory.

The number of bytes moved during an IN transaction depends on how much data the device endpoint delivers. The maximum number of bytes a device can send is *Maximum Packet Size*. The number of bytes moved during an OUT transaction is either *Maximum Packet Length* bytes or *Total Bytes to Transfer*, whichever is less.

If there was a transaction error during the transaction, the transfer state (as defined above) is not advanced by the host controller. The *CErr* field is decremented by one and the status field is updated to reflect the type of error observed. Transaction errors are summarized in Section Transaction Error .

The following events will cause the host controller to clear the *Active* bit in the queue head's overlay status field. When the *Active* bit transitions from a one to a zero, the transfer in the overlay is considered complete. The reason for the transfer completion (clearing the *Active* bit) determines the next state.

- *CErr* field decrements to zero. When this occurs the *Halted* bit is set to a one and *Active* is set to a zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The device responds to the transaction with a STALL PID. When this occurs, the *Halted* bit is set to a one and the *Active* bit is set to a zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.

- The *Total Bytes to Transfer* field is zero after the transaction completes. Note that for a zero length transaction, it was zero before the transaction was started. When this condition occurs, the *Active* bit is set to zero.
- The PID code is an IN, and the number of bytes moved during the transaction is less than the *Maximum Packet Length*. When this occurs, the *Active* bit is set to zero and a short packet condition exists. The short-packet condition is detected during the **Advance Queue** state. Refer to Section Split Transactions for additional rules for managing low- and full-speed transactions.

With the exception of a NAK response (when *RL* field is zero), the host controller always writes the results of the transaction back to the overlay area in main memory. This includes when the transfer completes. For a high-speed endpoint, the queue head information written back includes minimally the following fields: The *PID Code* field indicates an IN and the device sends more than the expected number of bytes (e.g. *Maximum Packet Length* or *Total Bytes to Transfer* bytes, whichever is less) (e.g. a packet babble). This results in the host controller setting the *Halted* bit to a one.

- NakCnt, dt, Total Bytes to Transfer, C_Page, Status, CERR, and Current Offset

For a low- or full-speed device the queue head information written back also includes the fields:

- C-prog-mask, FrameTag and S-bytes.

The duration of this state depends on the time it takes to complete the transaction(s) and the status write to the overlay is committed.

Halting a Queue Head

A halted endpoint is defined only for the transfer types that are managed via queue heads (control, bulk and interrupt). The following events indicate that the endpoint has reached a condition where no more activity can occur without intervention from the driver:

- An endpoint may return a STALL handshake during a transaction,
- A transaction had three consecutive error conditions, or
- A Packet Babble error occurs on the endpoint.

When any of these events occur (for a queue head) the Host Controller halts the queue head and set the USBERRINT status bit in the USBSTS register to a one. To halt the queue head, the *Active* bit is set to a zero and the *Halted* bit is set to a one. There may be other error status bits that are set when a queue is halted. The host controller always writes back the overlay area to the source qTD when the transfer is complete, regardless of the reason (normal completion, short packet or halt). The host controller will not advance the transfer state on a transaction that results in a *Halt* condition (e.g. no updates necessary for *Total Bytes to Transfer*, *C_Page*, *Current Offset*, and *dt*). The host controller must update *CErr* as appropriate. When a queue head is halted, the *USB Error Interrupt* bit in the USBSTS register is set to a one. If the *USB Error Interrupt Enable* bit in the USBINTR register is set to a one, a hardware interrupt is generated at the next interrupt threshold.

Asynchronous Schedule Park Mode

Asynchronous Schedule Park mode is a special execution mode that can be enabled by system software, where the host controller is permitted to execute more than one bus transaction from a high-speed queue head in the Asynchronous schedule before continuing horizontal traversal of the Asynchronous schedule.

This feature has no effect on queue heads or other data structures in the Periodic schedule. This feature is similar in intent as the *Mult* feature that is used in the Periodic schedule. Where-as the *Mult* feature is a characteristic that is tunable for each endpoint; park-mode is a policy that is applied to all high-speed queue heads in the asynchronous schedule. It is essentially the specification of an iterator for consecutive bus transactions to the same endpoint. All of the rules for managing bus transactions and the results of those as defined in Section Execute Transaction apply. This feature merely specifies how many consecutive times the host controller is permitted to execute from the same queue head before moving to the next queue head in the Asynchronous List. This feature should allow the host controller to attain better bus utilization for those devices that are capable of moving data at maximum rate, while at the same time providing a fair service to all endpoints.

A host controller exports its capability to support this feature to system software by setting the *Asynchronous Schedule Park Capability* bit in the HCCPARAMs register to a one. This information keys system software that the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USBCMD register are modifiable. System software enables the feature by writing a one to the *Asynchronous Schedule Park Mode Enable* bit.

When park-mode is not enabled (e.g. *Asynchronous Schedule Park Mode Enable* bit in the USBCMD register is a zero), the host controller must not execute more than one bus transaction per high-speed queue head, per traversal of the asynchronous schedule. When park-mode is enabled, the host controller must not apply the feature to a queue head whose *EPS* field indicates a Low/Full-speed device (i.e. only one bus transaction is allowed from each Low/Full-speed queue head per traversal of the asynchronous schedule). Park-mode may only be applied to queue heads in the Asynchronous schedule whose *EPS* field indicates that it is a high-speed device.

The host controller must apply park mode to queue heads whose *EPS* field indicates a high-speed endpoint. The maximum number of consecutive bus transactions a host controller may execute on a high-speed queue head is determined by the value in the *Asynchronous Schedule Park Mode Count* field in the USBCMD register. Software must not set *Asynchronous Schedule Park Mode Enable* bit to a one and also set *Asynchronous Schedule Park Mode Count* field to a zero. The resulting behavior is not defined. An example behavioral example describes the operational requirements for the host controller implementing park-mode. This feature does not affect how the host controller handles the bus transaction as defined in Section Execute Transaction . It only effects how many consecutive bus transactions for the current queue head can be executed. All boundary conditions, error detection and reporting applies as usual. This feature is similar in concept to the use of the *Mult* field for high-bandwidth Interrupt for queue heads in the Periodic Schedule.

The host controller effectively loads an internal down-counter *PM-Count* from *Asynchronous Schedule Park Mode Count* when *Asyncrhonous Schedule Park Mode Enable* bit is a one, and a high-speed queue head is first fetched and meets all the criteria for executing a bus transaction. After the bus transaction, *PM-Count* is decremented. The host controller may continue to execute bus transactions from the current queue head until *PM-Count* goes to zero, an error is detected, the buffer for the current transfer is exhausted or the endpoint responds with a flow-control or STALL handshake. Table 30-67 summarizes the responses that effect whether the host controller continues with another bus transaction for the current queue head.

Table 30-67. Actions for Park Mode, based on Endpoint Response and Residual Transfer State

PID	Endpoint Response	Transfer State after Transaction		Action
		PM-Count	Bytes to Transfer	
IN	DATA[0,1] w/Maximum Packet sized data	Not zero	Not Zero	Allowed to perform another bus transaction.[1,2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't care	Move to next QH.
	DATA[0,1] w/short packet	Don't care	Don't care	Retire qTD and move to next QH.
	NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH.
OUT	ACK	Not zero	Not Zero	Allowed to perform another bus transaction. [2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't' care	Move to next QH.
	NYET, NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH
PING	ACK	Not Zero	Not Zero	Allowed to perform another bus transaction. [2]
	NAK	Don't care	Don't care	Move to next QH
	STALL, XactErr	Don't care	Don't care	Move to next QH

1 Note, the host controller may continue to execute bus transactions from the current high-speed queue head (if *PM-Count* is not equal to zero), if a PID mismatch is detected (e.g. expected DATA1 and received DATA0, or visa-versa).,

2 Note, this specification does not *require* that the host controller execute another bus transaction when *PM-Count* is non-zero. Implementations are encouraged to make appropriate complexity and performance trade-offs.

30.8.3.10.4 Write Back qTD

This state is entered from the **Execute Transaction** state when the *Active* bit is set to a zero. The source data for the write-back is the transfer results area of the queue head overlay area (see Figure 30-48). The host controller uses the *Current qTD Pointer* field as the target address for the qTD. The queue head transfer result area is written back to the transfer result area of the target qTD. This state is also referred to as: qTD retirement. The fields that must be written back to the source qTD include *Total Bytes to Transfer*, *Cerr*, and *Status*.

The duration of this state depends on when the qTD write-back has been committed.

30.8.3.10.5 Follow Queue Head Horizontal Pointer

The host controller must use the horizontal pointer in the queue head to the next schedule data structure when any of the following conditions exist:

- If the *Active* bit is a one on exit from the **Execute Transaction** state, or
- When the host controller exits the **Write Back qTD** state, or
- If the **Advance Queue** state fails to advance the queue because the target qTD is not active, or
- If the *Halted* bit is a one on exit from the **Fetch QH** state.

There is no functional requirement that the host controller wait until the current transaction is complete before using the horizontal pointer to read the next linked data structure. However, it must wait until the current transaction is complete before executing the next data structure.

30.8.3.10.6 Buffer Pointer List Use for Data Streaming with qTDs

A qTD has an array of buffer pointers, which is used to reference the data buffer for a transfer. This specification requires that the buffer associated with the transfer be *virtually contiguous*. This means: if the buffer spans more than one physical page, it must obey the following rules (Figure 30-64 illustrates an example):

- The first portion of the buffer must begin at some offset in a page and extend through the end of the page.
- The remaining buffer cannot be allocated in small chunks scattered around memory. For each 4K chunk beyond the first page, each buffer portion matches to a full 4K page. The final portion, which may only be large enough to occupy a portion of a page, must start at the top of the page and be contiguous within that page.

The buffer pointer list in the qTD is long enough to support a maximum transfer size of 20K bytes. This case occurs when all five buffer pointers are used and the first offset is zero. A qTD handles a 16Kbyte buffer with any starting buffer alignment.

The host controller uses the field *C_Page* field as an index value to determine which buffer pointer in the list should be used to start the current transaction. The host controller uses a different buffer pointer for each physical page of the buffer. This is always true, even if the buffer is physically contiguous.

The host controller must detect when the current transaction will span a page boundary and automatically move to the next available buffer pointer in the page pointer list. The next available pointer is reached by incrementing *C_Page* and pulling the next page pointer from the list. Software must ensure there are sufficient buffer pointers to move the amount of data specified in the *Bytes to Transfer* field.

Figure 30-64 illustrates a nominal example of how System software would initialize the buffer pointers list and the *C_Page* field for a transfer size of 16383 bytes. *C_Page* is set to zero. The upper 20-bits of Page 0 references the start of the physical page. *Current Offset* (the lower 12-bits of queue head Dword 7) holds the offset in the page e.g. 2049 (e.g. 4096-2047). The remaining page pointers are set to reference the beginning of each subsequent 4K page.

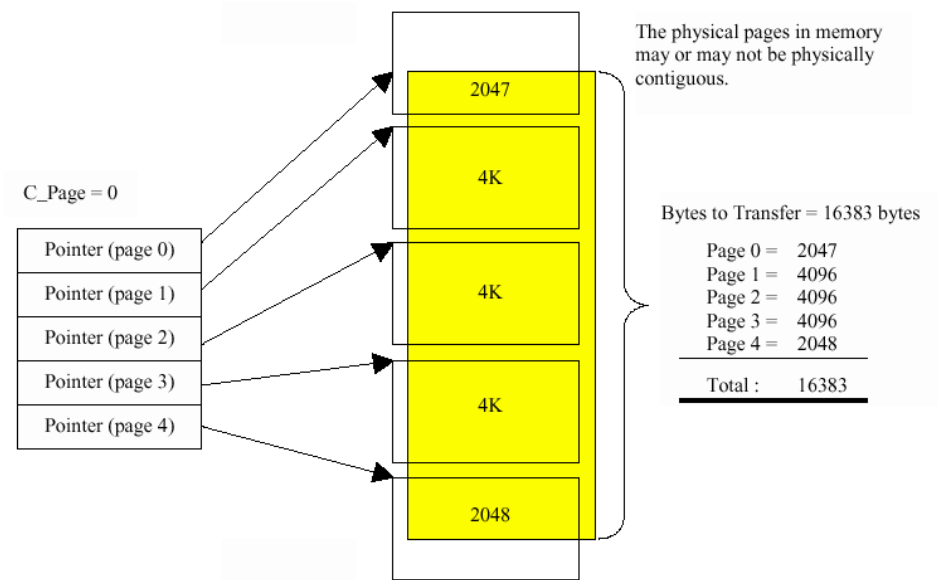


Figure 30-64. Example Mapping of qTD Buffer Pointers to Buffer Pages

For the first transaction on the qTD (assuming a 512-byte transaction), the host controller uses the first buffer pointer (page 0 because *C_Page* is set to zero) and concatenates the *Current Offset* field. The 512 bytes are moved during the transaction, the *Current Offset* and *Total Bytes to Transfer* are adjusted by 512 and written back to the queue head working area.

During the 4th transaction, the host controller needs 511 bytes in page 0 and one byte in page 1. The host controller will increment *C_Page* (to 1) and use the page 1 pointer to move the final byte of the transaction. After the 4th transaction, the active page pointer is the page 1 pointer and *Current Offset* has rolled to one, and both are written back to the overlay area. The transactions continue for the rest of the buffer, with the host controller automatically moving to the next page pointer (i.e. *C_Page*) when necessary. There are three conditions for how the host controller handles *C_Page*.

- The current transaction does not span a page boundary. The value of *C_Page* is not adjusted by the host controller.
- The current transaction does span a page boundary. The host controller must detect the page cross condition and advance to the next buffer while streaming data to/from the USB.
- The current transaction completes on a page boundary (i.e. the last byte moved for the current transaction is the last byte in the page for the current page pointer). The host controller must increment *C_Page* before writing back status for the transaction.

Note that the only valid adjustment the host controller may make to *C_Page* is to increment by one.

30.8.3.10.7 Adding Interrupt Queue Heads to the Periodic Schedule

The link path(s) from the periodic frame list to a queue head establishes in which frames a transaction can be executed for the queue head. Queue heads are linked into the periodic schedule so they are polled at the appropriate rate. System software sets a bit in a queue head's *S-Mask* to indicate which micro-frame with-in a 1 millisecond period a transaction should be executed for the queue head. Software must ensure

that all queue heads in the periodic schedule have *S-Mask* set to a non-zero value. An *S-mask* with a zero value in the context of the periodic schedule yields undefined results.

If the desired poll rate is greater than one frame, system software can use a combination of queue head linking and *S-Mask* values to spread interrupts of equal poll rates through the schedule so that the periodic bandwidth is allocated and managed in the most efficient manner possible. Some examples are illustrated in Table 30-68.

Table 30-68. Example Periodic Reference Patterns for Interrupt Transfers with 2ms Poll Rate

Frame # Reference Sequence	Description
0, 2, 4, 6, 8, etc. *S-Mask* = 01h	A queue head for the *bInterval* of 2 milliseconds (16 micro-frames) is linked into the periodic schedule so that it is reachable from the periodic frame list locations indicated in the previous column. In addition, the *S-Mask* field in the queue head is set to 01h, indicating that the transaction for the endpoint should be executed on the bus during micro-frame 0 of the frame.
0, 2, 4, 6, 8, etc. *S-Mask* = 02h	Another example of a queue head with a *bInterval* of 2 milliseconds is linked into the periodic frame list at exactly the same interval as the previous example. However, the *S-Mask* is set to 02h indicating that the transaction for the endpoint should be executed on the bus during micro-frame 1 of the frame.

30.8.3.10.8 Managing Transfer Complete Interrupts from Queue Heads

The host controller will set an interrupt to be signaled at the next interrupt threshold when the completed transfer (qTD) has an *Interrupt on Complete* (*IOC*) bit set to a one, or whenever a transfer (qTD) completes with a short packet. If system software needs multiple qTDs to complete a client request (i.e. like a control transfer) the intermediate qTDs do not require interrupts. System software may only need a single interrupt to notify it that the complete buffer has been transferred. System software may set IOC's to occur more frequently. A motivation for this may be that it wants early notification so that interface data structures can be re-used in a timely manner.

30.8.3.11 Ping Control

USB 2.0 defines an addition to the protocol for high-speed devices called Ping. Ping is required for all USB 2.0 High-speed bulk and control endpoints. Ping is not allowed for a split-transaction stream. This extension to the protocol eliminates the bad side-effects of Naking OUT endpoints. The *Status* field has a *Ping State* bit, which the host controller uses to determine the *next* actual PID it will use in the next transaction to the endpoint (see Table 30-51). The Ping State bit is only managed by the host controller for queue heads that meet the following criteria:

- Queue head is not an interrupt and
- *EPS* field equals High-Speed and
- *PIDCode* field equals OUT

Table 30-69 illustrates the state transition table for the host controller's responsibility for maintaining the PING protocol. Refer to Chapter 8 in the USB Specification Revision 2.0 for detailed description on the Ping protocol.

Table 30-69. Ping Control State Transition Table

	Event		
Current	**Host**	**Device**	**Next**
Do Ping	PING	Nak	Do Ping
Do Ping	PING	Ack	Do OUT
Do Ping	PING	XactErr[1]	Do Ping
Do Ping	PING	Stall	N/C[2] Do
OUT	OUT	Nak	Do Ping
Do OUT	OUT	Nyet	Do Ping
Do OUT	OUT	Ack	Do OUT
Do OUT	OUT	XactErr[1]	Do Ping
Do OUT	OUT	Stall	N/C[2]

1 Transaction Error (XactErr) is any time the host misses the handshake.

2 No transition change required for the Ping State bit. The Stall handshake results in the endpoint being halted (e.g. Active set to zero and Halt set to a one). Software intervention is required to restart queue. 3 A Nyet response to an OUT means that the device has accepted the data, but cannot receive any more at this time. Host must advance the transfer state and additionally, transition the Ping State bit to **Do Ping**. The Ping State bit has the following encoding:

Value	Meaning
0B	Do OUT The host controller will use an OUT PID during the next bus transaction to this endpoint.
1B	Do Ping The host controller will use a PING PID during the next bus transaction to this endpoint.

The defined ping protocol (see USB 2.0 Specification, Chapter 8) allows the host to be *imprecise* on the initialization of the ping protocol (i.e. start in *Do OUT* when we don't know whether there is space on the device or not). The host controller manages the *Ping State* bit. System software sets the initial value in the queue head when it initializes a queue head. The host controller preserves the *Ping State* bit across all queue advancements. This means that when a new qTD is written into the queue head overlay area, the previous value of the *Ping State* bit is preserved.

30.8.3.12 Split Transactions

USB 2.0 defines extensions to the bus protocol for managing USB 1.x data streams through USB 2.0 Hubs. This section describes how the host controller uses the interface data structures to manage data streams with full- and low-speed devices, connected below USB 2.0 hub, utilizing the split transaction protocol. Refer to USB 2.0 Specification for the complete definition of the split transaction protocol. Full- and Low-speed devices are enumerated identically as high-speed devices, but the transactions to the Full- and Low-speed endpoints use the split-transaction protocol on the high-speed bus. The split transaction protocol is an encapsulation of (or wrapper around) the Full- or Low-speed transaction. The high-speed wrapper portion of the protocol is addressed to the USB 2.0 Hub and Transaction Translator below which the Full- or Low-speed device is attached.

The EHCI interface uses dedicated data structures for managing full-speed isochronous data streams (see Section Split Transaction Isochronous Transfer Descriptor (siTD)). Control, Bulk and Interrupt are managed using the queuing data structures (see Sections Queue Head). The interface data structures need to be programmed with the device address and the Transaction Translator number of the USB 2.0 Hub operating as the Low-/Full-speed host controller for this link. The following sections describe the details of how the host controller must process and manage the split transaction protocol.

30.8.3.12.1 Split Transactions for Asynchronous Transfers

A queue head in the asynchronous schedule with an *EPS* field indicating a full-or low-speed device indicates to the host controller that it must use split transactions to stream data for this queue head. All full-speed bulk and full-, low-speed control are managed via queue heads in the asynchronous schedule.

Software must initialize the queue head with the appropriate device address and port number for the transaction translator that is serving as the full/low-speed host controller for the links connecting the endpoint. Software must also initialize the split transaction state bit (*SplitXState*) to **Do-Start-Split**. Finally, if the endpoint is a control endpoint, then system software must set the *Control Transfer Type* (*C*) bit in the queue head to a one. If this is not a control transfer type endpoint, the *C* bit must be initialized by software to be a zero. This information is used by the host controller to properly set the Endpoint Type (ET) field in the split transaction bus token. When the *C* bit is a zero, the split transaction token's ET field is set to indicate a bulk endpoint. When the *C* bit is a one, the split transaction token's ET field is set to indicate a control endpoint. Refer to Chapter 8 of USB Specification Revision 2.0 for details.

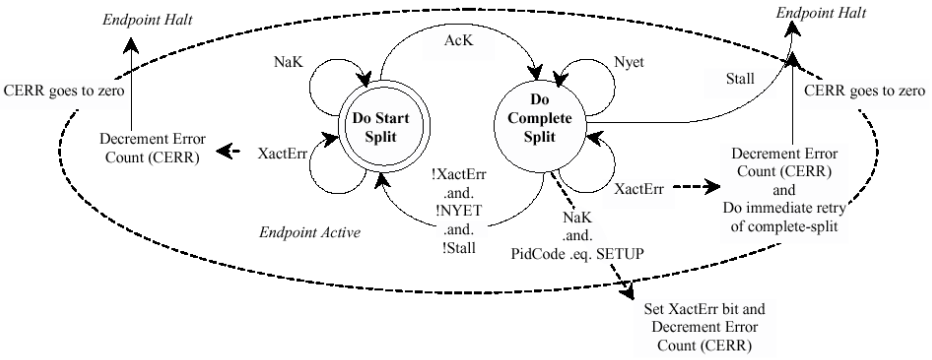


Figure 30-65. Host Controller Asynchronous Schedule Split-Transaction State Machine

Asynchronous - Do Start Split

This is the state which software must initialize a full- or low-speed asynchronous queue head. This state is entered from the **Do Complete Split** state only after a complete-split transaction receives a valid response from the transaction translator that is not a Nyet handshake.

For queue heads in this state, the host controller will execute a start-split transaction to the appropriate transaction translator. If the bus transaction completes without an error and *PidCode* indicates an IN or OUT transaction, then the host controller will reload the error counter (*CErr*). If it is a successful bus transaction and the *PidCode* indicates a SETUP, the host controller will not reload the error counter. If the transaction translator responds with a Nak, the queue head is left in this state, and the host controller proceeds to the next queue head in the asynchronous schedule.

If the host controller times out the transaction (no response, or bad response) the host controller decrements *Cerr* and proceeds to the next queue head in the asynchronous schedule.

Asynchronous - Do Complete Split

This state is entered from the **Do Start Split** state only after a start-split transaction receives an Ack handshake from the transaction translator.

For queue heads in this state, the host controller will execute a complete-split transaction to the appropriate transaction translator. If the transaction translator responds with a Nyet handshake, the queue head is left in this state, the error counter is reset and the host controller proceeds to the next queue head in the asynchronous schedule. When a Nyet handshake is received for a bus transaction where the queue head's *PidCode* indicates an IN or OUT, the host controller will reload the error counter (*CErr*). When a Nyet handshake is received for a complete-split bus transaction where the queue head's *PidCode* indicates a SETUP, the host controller must not adjust the value of *CErr.*

Independent of *PIDCode*, the following responses have the effects:

- Transaction Error (XactErr). Timeout or data CRC failure, etc. The error counter (*Cerr*) is decremented by one and the complete split transaction is *immediately* retried (if possible). If there is not enough time in the micro-frame to execute the retry, the host controller MUST ensure that the next time the host controller begins executing from the Asynchronous schedule, it must begin executing from this queue head. If another start-split (for some other endpoint) is sent to the transaction translator before the complete-split is really completed, the transaction translator could dump the results (which were never delivered to the host). This is why the core specification states the retries must be immediate. A method to accomplish this behavior is to not advance the asynchronous schedule. When the host controller returns to the asynchronous schedule in the next micro-frame, the first transaction from the schedule will be the retry for this endpoint. If *Cerr* went to zero, the host controller must halt the queue.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced and the state is exited. If the *PidCode* is a SETUP, then the Nak response is a protocol error. The *XactErr* status bit is set to a one and the *CErr* field is decremented.
- STALL. The target endpoint responded with a STALL handshake. The host controller sets the *halt* bit in the status byte, retires the qTD but does not attempt to advance the queue.
- If the *PidCode* indicates an IN, then any of following responses are expected:

- DATA0/1. On reception of data, the host controller ensures the PID matches the expected data toggle and checks CRC. If the packet is *good*, the host controller will advance the state of the transfer, e.g. move the data pointer by the number of bytes received, decrement *BytesToTransfer* field by the number of bytes received, and toggle the *dt* bit. The host controller will then exit this state. The response and advancement of transfer may trigger other processing events, such as retirement of the qTD and advancement of the queue.
- If the data sequence PID does not match the expected, the data is ignored, the transfer state is not advanced and this state is exited. If the *PidCode* indicates an OUT/SETUP, then any of following responses are expected:
- ACK. The target endpoint accepted the data, so the host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount and the data toggle bit (*dt*) is toggled. The host controller will then exit this state.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers via Queue Heads).

30.8.3.12.2 Split Transaction Interrupt

Split-transaction Interrupt-IN/OUT endpoints are managed via the same data structures used for high-speed interrupt endpoints. They both co-exist in the periodic schedule. Queue heads/qTDs offer the set of features required for reliable data delivery, which is characteristic to interrupt transfer types. The split-transaction protocol is managed completely within this defined functional transfer framework. For example, for a high-speed endpoint, the host controller will visit a queue head, execute a high-speed transaction (if criteria are met) and advance the transfer state (or not) depending on the results of the entire transaction. For low- and full-speed endpoints, the details of the *execution* phase are different (i.e. takes more than one bus transaction to complete), but the remainder of the operational framework is intact. This means that the transfer advancement, etc. occurs as defined in Section Managing Control/Bulk/Interrupt Transfers via Queue Heads, but only occurs on the completion of a split transaction.

Split Transaction Scheduling Mechanisms for Interrupt

Full- and low-speed Interrupt queue heads have an *EPS* field indicating full- or low-speed and have a non-zero *S-mask* field. The host controller can detect this combination of parameters and assume the endpoint is a periodic endpoint. Low- and full-speed interrupt queue heads require the use of the split transaction protocol. The host controller sets the Endpoint Type (ET) field in the split token to indicate the transaction is an interrupt. These transactions are managed through a transaction translator's periodic pipeline. Software should not set these fields to indicate the queue head is an interrupt unless the queue head is used in the periodic schedule.

System software manages the per/transaction translator periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each endpoint will occur. The characteristics of the transaction translator are such that the high-speed transaction protocol must execute during explicit micro-frames, or the data or response information in the pipeline is lost. Figure illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule and queue

head data structure. The **S** and ^{C}x labels indicate micro-frames where software can schedule strt-splits and complete splits (respectively).

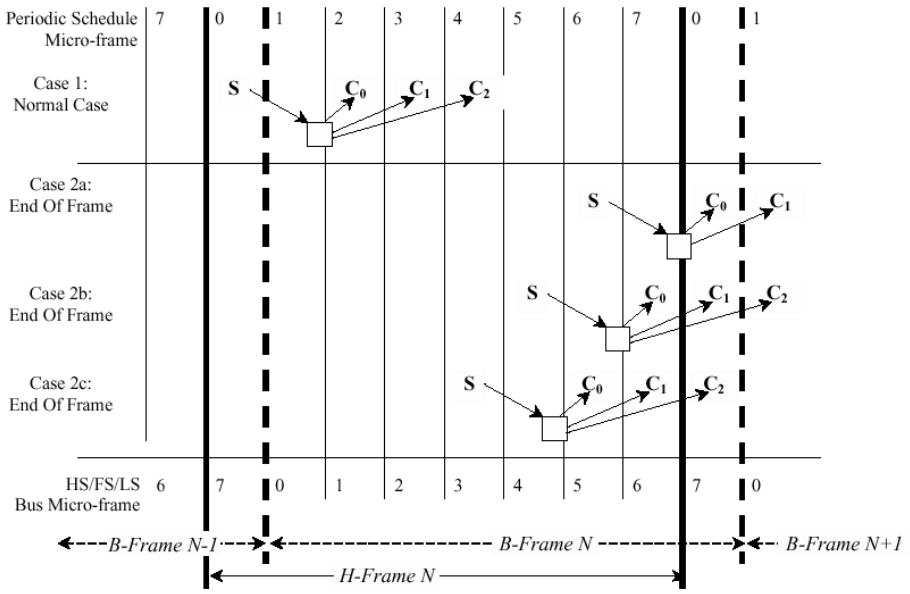


Figure 30-66. Split Transaction, Interrupt Scheduling Boundary Conditions

The scheduling cases are:

- Case 1: The normal scheduling case is where the entire split transaction is completely bounded by a frame (*H-Frame* in this case).
- Case 2a through Case 2c: The USB 2.0 Hub pipeline rules states clearly, when and how many complete-splits must be scheduled to account for earliest to latest execution on the full/low-speed link. The complete-splits may span the *H-Frame* boundary when the start-split is in micro-frame 4 or later. When this occurs, the *H-Frame* to *B-Frame* alignment requires that the queue head be reachable from consecutive periodic frame list locations. System software cannot build an efficient schedule that satisfies this requirement unless it uses FSTNs. Figure 30-67 illustrates the general layout of the periodic schedule.

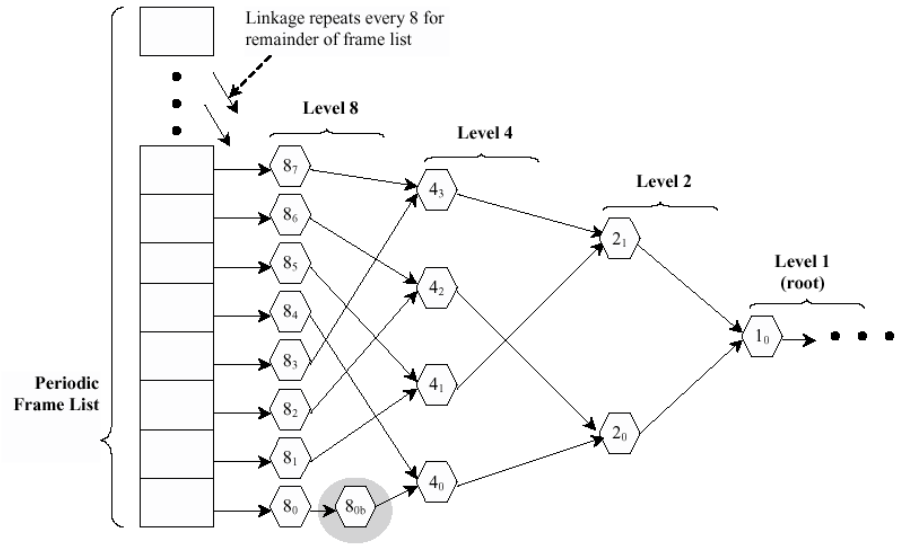


Figure 30-67. General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading

The periodic frame list is effectively the leaf level a binary tree, which is always traversed leaf to root. Each level in the tree corresponds to a 2^N poll rate. Software can efficiently manage periodic bandwidth on the USB by *spreading* interrupt queue heads that have the same poll rate requirement across all the available paths from the frame list. For example, system software can schedule eight poll rate 8 queue heads and account for them once in the high-speed bus bandwidth allocation.

When an endpoint is allocated an execution footprint that spans a frame boundary, the queue head for the endpoint must be reachable from consecutive locations in the frame list. An example would be if 8_{0b} where such an endpoint. Without additional support on the interface, to get 8_{0b} reachable at the correct time, software would have to link 8_1 to 8_{0b}. It would then have to move 4_1 and everything linked after into the same path as 4_0. This upsets the integrity of the binary tree and disallows the use of the spreading technique.

FSTN data structures are used to preserve the integrity of the binary-tree structure and enable the use of the spreading technique. Section Host Controller Operational Model for FSTNs defines the hardware and software operational model requirements for using FSTNs.

The following queue head fields are initialized by system software to instruct the host controller when to execute portions of the split-transaction protocol.

- *SplitXState*. This is a single bit residing in the *Status* field of a queue head (see Table 30-51). This bit is used to track the current state of the split transaction.
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure , case one, the *S-mask* would have a value of 00000001b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates **Do_Start**, and the current micro-frame as indicated by FRINDEX[2:0] is 0, then execute a start-split transaction.

- *Frame C-mask.* This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure , case one, the *C-mask* would have a value of 00011100b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates **Do_Complete**, and the current micro-frame as indicated by FRINDEX[2:0] is 2, 3, or 4, then execute a complete-split transaction. It is software's responsibility to ensure that the translation between *H-Frames* and *B-Frames* is correctly performed when setting bits in *S-mask* and *C-mask*

Host Controller Operational Model for FSTNs

The FSTN data structure is used to manage Low/Full-speed interrupt queue heads that need to be reached from consecutive frame list locations (i.e. boundary cases 2a through 2c). An FSTN is essentially a *back pointer*, similar in intent to the back pointer field in the siTD data structure (see Section siTD Back Link Pointer).

This feature provides software a simple primitive to save a schedule position, redirect the host controller to traverse the necessary queue heads in the previous frame, then restore the original schedule position and complete normal traversal.

There are four components to the use of FSTNs:

- FSTN data structure.
- A *Save Place* indicator. This is always an FSTN with its *Back Path Link Pointer. T-bit* set to zero.
- A *Restore* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to a one.
- Host controller FSTN traversal rules.

Host Controller Operational Model for FSTNs

When the host controller encounters an FSTN during micro-frames 2 through 7 it simply follows the node's *Normal Path Link Pointer* to access the next schedule data structure. Note that the FSTN's *Normal Path Link Pointer.T-bit* may set to a one, which the host controller must interpret as the end of periodic list mark.

When the host controller encounters a *Save-Place* FSTN in micro-frames 0 or 1, it will save the value of the *Normal Path Link Pointer* and set an internal flag indicating that it is executing in *Recovery Path* mode. *Recovery Path* mode modifies the host controller's rules for how it traverses the schedule and limits which data structures will be considered for execution of bus transactions. The host controller continues executing in *Recovery Path* mode until it encounters a *Restore* FSTN or it determines that it has reached the end of the micro-frame (see details in the list below).

The rules for schedule traversal and limited execution while in *Recovery Path* mode are:

- Always follow the *Normal Path Link Pointer* when it encounters an FSTN that is a *Save-Place* indicator. The host controller must not recursively follow *Save-Place* FSTNs. Therefore, while executing in *Recovery Path* mode, it must never follow an FSTN's *Back Path Link Pointer.*
- Do not process an siTD or, iTD data structure. Simply follow its *Next Link Pointer.*

- Do not process a QH (Queue Head) whose *EPS* field indicates a high-speed device. Simply follow its *Horizontal Link Pointer.*
- When a QH's *EPS* field indicates a Full/Low-speed device, the host controller will only consider it for execution if its *SplitXState* is **DoComplete** (note: this applies whether the *PID Code* indicates an IN or an OUT). See Sections Execute Transaction and Tracking Split Transaction Progress for Interrupt Transfers for a complete list of additional conditions that must be met in general for the host controller to issue a bus transaction. Note that the host controller must not execute a Start-split transaction while executing in *Recovery Path* mode. See Section Periodic Interrupt - Do Complete Split for special handling when in *Recovery Path* mode.
- Stop traversing the *recovery path* when it encounters an FSTN that is a *Restore* indicator. The host controller unconditionally uses the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* when returning to the normal path traversal. The host controller must clear the context of executing a *Recovery Path* when it restores schedule traversal to the *Save-Place* FSTN's *Normal Path Link Pointer.*
- If the host controller determines that there is not enough time left in the micro-frame to complete processing of the periodic schedule, it abandons traversal of the recovery path, and clears the context of executing a recovery path. The result is that at the start of the next consecutive micro-frame, the host controller starts traversal at the frame list.

An example traversal of a periodic schedule that includes FSTNs is illustrated in Figure 30-68.

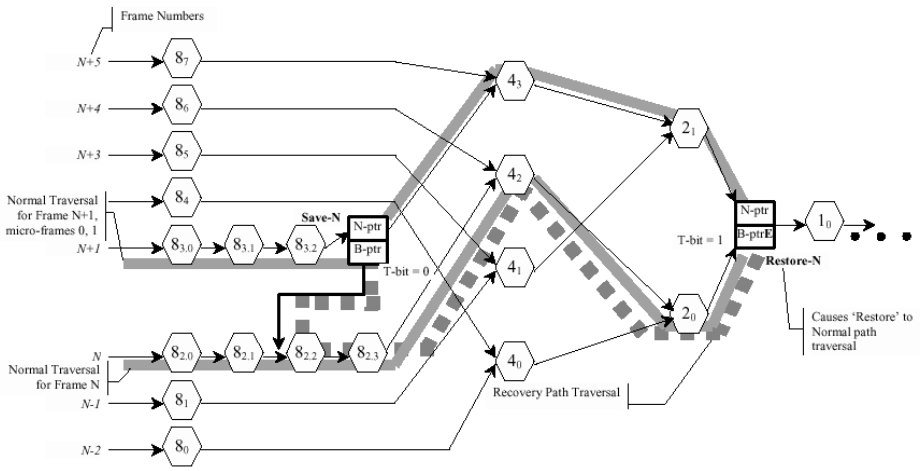


Figure 30-68. Example Host Controller Traversal of Recovery Path via FSTNs

In frame N+1 (micro-frames 0 and 1), when the host controller encounters Save-Path FSTN (Save-N), it observes that Save-N.Back Path Link Pointer.T-bit is zero (definition of a Save-Path indicator). The host controller saves the value of Save-N.Normal Path Link Pointer and follows Save-N.Back Path Link Pointer. At the same time, it sets an internal flag indicating that it is now in *Recovery Path* mode (the recovery path is annotated in Figure 30-9 with a large dashed line). The host controller continues traversing data structures on the recovery path and executing only those bus transactions as noted above, on the recovery path until it reaches Restore FSTN (Restore-N). Restore-N.Back Path Link Pointer.T-bit is set to a one (definition of a Restore indicator), so the host controller exits *Recovery Path* mode by clearing the internal *Recovery Path* mode flag and commences (restores) schedule traversal using the

saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* (e.g. Save-N.Normal Path Link Pointer). The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, **$8_{2.2}$**, **$8_{2.3}$**, **4_2**, **2_0**, **Restore-N**, 4_3, 2_1, Restore-N, 1_0 …}. The nodes on the recovery-path are bolded.In frame N (micro-frames 0-7), for this example, the host controller will traverse all of the schedule data structures utilizing the *Normal Path Link Pointers* in any FSTNs it encounters. This is because the host controller has not yet encountered a *Save-Place* FSTN so it not executing in *Recovery Path* mode. When it encounters the *Restore* FSTN, (Restore-N), during micro-frames 0 and 1, it uses Restore-N.Normal Path Link Pointer to traverse to the next data structure (i.e. normal schedule traversal). This is because the host controller must use a Restore FSTN's *Normal Path Link Pointer* when not executing in a *Recovery-Path* mode. The nodes traversed during frame N include: {$8_{2.0}$, $8_{2.1}$, $8_{2.2}$, $8_{2.3}$, 4_2, 2_0, Restore-N, 1_0 …}.

In frame N+1 (micro-frames 2-7), when the host controller encounters Save-Path FSTN Save-N, it will unconditionally follow Save-N.Normal Path Link Pointer. The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, 4_3, 2_1, Restore-N, 1_0 …}.

Software Operational Model for FSTNs

Software must create a consistent, coherent schedule for the host controller to traverse. When using FSTNs, system software must adhere to the following rules:

- Each *Save-Place* indicator requires a matching *Restore* indicator.
  - The *Save-Place* indicator is an FSTN with a valid *Back Path Link Pointer and T-bit* equal to zero. Note that *Back Path Link Pointer.Typ* field must be set to indicate the referenced data structure is a queue head. The *Restore* indicator is an FSTN with its *Back Path Link Pointer.T-bit* set to a one.
  - A *Restore* FSTN may be matched to one or more *Save-Place* FSTNs. For example, if the schedule includes a poll-rate 1 level, then system software only needs to place a *Restore* FSTN at the beginning of this list in order to match all possible *Save-Place* FSTNs.
- If the schedule does not have elements linked at a poll-rate level of one, and one or more *Save-Place* FSTNs are used, then System Software must ensure the *Restore* FSTN's *Normal Path Link Pointer*'s *T-bit* is set to a one, as this will be use to mark the end of the periodic list.
- When the schedule does have elements linked at a poll rate level of one, a *Restore* FSTN must be the first data structure on the poll rate one list. All traversal paths from the frame list converge on the poll-rate one list. System software must ensure that *Recovery Path* mode is exited before the host controller is allowed to traverse the poll rate level one list.
- A *Save-Place* FSTN's *Back Path Link Pointer* must reference a queue head data structure. The referenced queue head must be reachable from the previous frame list location. In other words, if the *Save-Place* FSTN is reachable from frame list offset N, then the FSTN's *Back Path Link Pointer* must reference a queue head that is reachable from frame list offset N-1.

Software should make the schedule as efficient as possible. What this means in this context is that software should have no more than one *Save-Place* FSTN reachable in any single frame. Note there will be times when two (or more, depending on the implementation) could exist as full/low-speed footprints change with bandwidth adjustments. This could occur, for example when a bandwidth rebalance causes system software to move the *Save-Place* FSTN from one poll rate level to another. During the transition, software must preserve the integrity of the previous schedule until the new schedule is in place.

Tracking Split Transaction Progress for Interrupt Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where data is lost. For interrupt-IN transfers, data is lost when it makes it into the USB 2.0 hub, but the USB 2.0 host system is unable to get it from the USB 2.0 Hub and into the system before it expires from the transaction translator pipeline. When a lost data condition is detected, the queue must be halted, thus signaling system software to recover from the error. A data-loss condition exists whenever a start-split is issued, accepted and successfully executed by the USB 2.0 Hub, but the complete-splits get unrecoverable errors on the high-speed link, or the complete-splits do not occur at the correct times. One reason complete-splits might not occur at the right time would be due to host-induced system hold-offs that cause the host controller to miss bus transactions because it cannot get timely access to the schedule in system memory.

The same condition can occur for an interrupt-OUT, but the result is not an endpoint halt condition, but rather effects only the progress of the transfer. The queue head has the following fields to track the progress of each split transaction. These fields are used to keep incremental state about which (and when) portions have been executed.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets one of the *C-prog*-mask bits for each complete-split executed. The bit position is determined by the micro-frame number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a complete-split transaction. If the previous complete-splits have not been executed then it means one (or more) have been skipped and data has potentially been lost.
- *FrameTag*. This field is used by the host controller during the complete-split portion of the split transaction to tag the queue head with the frame number (*H-Frame* number) when the next complete split must be executed.
- *S-bytes*. This field can be used to store the number of data payload bytes sent during the start-split (if the transaction was an OUT). The *S-bytes* field must be used to accumulate the data payload bytes received during the complete-splits (for an IN).

Split Transaction Execution State Machine for Interrupt

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame*.

As with asynchronous Full- and Low-speed endpoints, a split-transaction state machine is used to manage the split transaction sequence. Aside from the fields defined in the queue head for scheduling and tracking the split transaction, the host controller calculates one internal mechanism that is also used to manage the split transaction. The internal calculated mechanism is:

- *cMicroFrameBit*. This is a single-bit encoding of the current micro-frame number. It is an eight-bit value calculated by the host controller at the beginning of every micro-frame. It is calculated from the three least significant bits of the *FRINDEX* register (i.e. *cMicroFrameBit* = (1

shifted-left(*FRINDEX*[2:0]))). The *cMicroFrameBit* has at most one bit asserted, which always corresponds to the current micro-frame number. For example, if the current micro-frame is 0, then *cMicroFrameBit* will equal 00000001b. The variable *cMicroFrameBit* is used to compare against the *S-mask* and *C-mask* fields to determine whether the queue head is marked for a start- or complete-splt transaction for the current micro-frame.

Figure 30-69 illustrates the state machine for managing a complete interrupt split transaction. There are two phases to each split transaction. The first is a single start-split transaction, which occurs when the *SplitXState* is at **Do_Start** and the single bit in *cMicroFrameBit* has a corresponding bit active in *QH.S-mask*. The transaction translator does not acknowledge the receipt of the periodic start-split, so the host controller unconditionally transitions the state to **Do_Complete**. Due to the available jitter in the transaction translator pipeline, there will be more than one complete-split transaction scheduled by software for the **Do_Complete** state. This translates simply to the fact that there are multiple bits set to a one in the *QH.C-mask* field.

The host controller keeps the queue head in the **Do_Complete** state until the split transaction is complete (see definition below), or an error condition triggers the *three-strikes-rule* (e.g. after the host tries the same transaction three times, and each encounters an error, the host controller will stop retrying the bus transaction and halt the endpoint, thus requiring system software to detect the condition and perform system-dependent recovery).

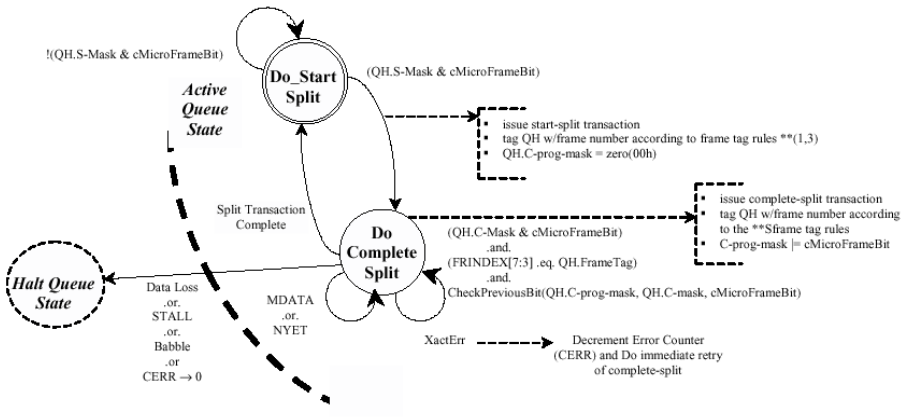


Figure 30-69. Split Transaction State Machine for Interrupt

**See Previous Section for the frame tag management rules.

Periodic Interrupt - Do Start Split

This is the state software must initialize a full- or low-speed interrupt queue head *StartXState* bit. This state is entered from the **Do_Complete Split** state only after the split transaction is complete. This occurs when one of the following events occur: The transaction translator responds to a complete-split transaction with one of the following:

- NAK. A NAK response is a propagation of the full- or low-speed endpoint's NAK response.
- ACK. An ACK response is a propagation of the full- or low-speed endpoint's ACK response. Only occurs on an OUT endpoint.

- DATA 0/1. Only occurs for INs. Indicates that this is the last of the data from the endpoint for this split transaction.
- ERR. The transaction on the low-/full-speed link below the transaction translator had a failure (e.g. timeout, bad CRC, etc.).
- NYET (and Last). The host controller issued the last complete-split and the transaction translator responded with a NYET handshake. This means that the start-split was not correctly received by the transaction translator, so it never executed a transaction to the full- or low-speed endpoint, see Section Periodic Interrupt - Do Complete Split for the definition of 'Last'.

Each time the host controller visits a queue head in this state (once within the **Execute Transaction** state), it performs the following test to determine whether to execute a start-split.

- *QH.S-mask* is bit-wise anded with *cMicroFrameBit*.

If the result is non-zero, then the host controller will issue a start-split transaction. If the *PIDCode* field indicates an IN transaction, the host controller must zero-out the *QH.S-bytes* field. After the split-transaction has been executed, the host controller sets up state in the queue head to track the progress of the complete-split phase of the split transaction. Specifically, it records the expected frame number into *QH.FrameTag* field (see Section Managing QH.FrameTag Field), set *C-prog-mask* to zero (00h), and exits this state. Note that the host controller must not adjust the value of *CErr* as a result of completion of a start-split transaction.

Periodic Interrupt - Do Complete Split

This state is entered unconditionally from the **Do Start Split** state after a start-split transaction is executed on the bus. Each time the host controller visits a queue head in this state (once within the **Execute Transaction** state), it checks to determine whether a complete-split transaction should be executed now.

There are four tests to determine whether a complete-split transaction should be executed.

- **Test A**. *cMicroFrameBit* is bit-wise anded with *QH.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame.
- **Test B**. *QH.FrameTag* is compared with the current contents of *FRINDEX*[7:3]. An equal indicates a match.
- **Test C**. The complete-split progress bit vector is checked to determine whether the previous bit is set, indicating that the previous complete-split was appropriately executed. An example algorithm for this test is provided below:

Algorithm Boolean CheckPreviousBit(QH.C-prog-mask, QH.C-mask, cMicroFrameBit)

```
Begin

-- Return values:

-- TRUE - no error

-- FALSE - error

--

Boolean rvalue = TRUE;

previousBit = cMicroframeBit logical-rotate-right(1)
```

```
-- Bit-wise anding previousBit with C-mask indicates
-- whether there was an intent
-- to send a complete split in the previous micro-frame. So,
-- if the
-- 'previous bit' is set in C-mask, check C-prog-mask to
-- make sure it
-- happened.
If (previousBit bitAND QH.C-mask)then
                          If not(previousBit bitAND QH.C-prog-mask) then
                 rvalue = FALSE;
         End if
End If
-- If the C-prog-mask already has a one in this bit position,
-- then an aliasing
-- error has occurred. It will probably get caught by the
-- FrameTag Test, but
-- at any rate it is an error condition that as detectable here
-- should not allow
-- a transaction to be executed.
If (cMicroFrameBit bitAND QH.C-prog-mask) then
 rvalue = FALSE;
End if
return (rvalue)
End Algorithm
```

- **Test D.** Check to see if a start-split should be executed in this micro-frame. Note this is the same test performed in the **Do Start Split** state (see Section Periodic Interrupt - Do Start Split). Whenever it evaluates to TRUE and the controller is NOT processing in the context of a *Recovery Path* mode, it means a start-split should occur in this micro-frame. Test D and Test A evaluating to TRUE at the same time is a system software error. Behavior is undefined.

If (A .and. B .and. C .and. not(D)) then the host controller will execute a complete-split transaction. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. On completion of the complete-split transaction, the host controller records the result of the transaction in the queue head and sets *QH.FrameTag* to the expected *H-Frame* number (see Section Managing QH.FrameTag Field). The effect to the state of the queue head and thus the state of the transfer depends on the response by the transaction translator to the complete-split

transaction. The following responses have the effects (note that any responses that result in decrementing of the *CErr* will result in the queue head being halted by the host controller if the result of the decrement is zero):

- NYET (and Last). On each NYET response, the host controller checks to determine whether this is the last complete-split for this split transaction. Last is defined in this context as the condition where all of the scheduled complete-splits have been executed. If it is the last complete-split (with a NYET response), then the transfer state of the queue head is not advanced (never received any data) and this state exited. The transaction translator must have responded to all the clompete-splits with NYETs, meaning that the start-split issued by the host controller was not received. The start-split should be retried at the next poll period.
- The test for whether this is the **Last** complete split can be performed by XOR *QH.C-mask* with *QH.C-prog-mask*. If the result is all zeros then all complete-splits have been executed. When this condition occurs, the *XactErr* status bit is set to a one and the *CErr* field is decremented.
- NYET (and not Last). See above description for testing for **Last**. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask* and *FrameTag*) and stay in this state. The host controller must not adjust *CErr* on this response.
- Transaction Error (XactErr). Timeout, data CRC failure, etc. The *CErr* field is decremented and the *XactErr* bit in the *Status* field is set to a one. The complete split transaction is *immediately* retried (if *Cerr* is non-zero).If there is not enough time in the micro-frame to complete the retry and the endpoint is an IN, or *CErr* is decremented to a zero from a one, the queue is halted. If there is not enough time in the micro-frame to complete the retry and the endpoint is an OUT and *CErr* is not zero, then this state is exited (i.e. return to **Do Start Split**). This results in a retry of the entire OUT split transaction, at the next poll period. Refer to Chapter 11 Hubs (specifically the section full- and low-speed Interrupts) in the USB Specification Revision 2.0 for detailed requirements on why these errors must be immediately retried.
- ACK. This can only occur if the target endpoint is an OUT. The target endpoint ACK'd the data and this response is a propagation of the endpoint ACK up to the host controller. The host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount. And the data toggle bit (*dt*) is toggled. The host controller will then exit this state for this queue head. The host controller must reload *CErr* with maximum value on this response. Advancing the transfer state may cause other process events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers via Queue Heads).
- MDATA. This response will only occur for an IN endpoint. The transaction translator responded with zero or more bytes of data and an MDATA PID. The incremental number of bytes received is accumulated in *QH.S-bytes*. The host controller must not adjust *CErr* on this response.
- DATA0/1. This response may only occur for an IN endpoint. The number of bytes received is added to the accumulated byte count in *QH.S-bytes*. The state of the transfer is advanced by the result and the host controller will exit this state for this queue head.

- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers via Queue Heads).
- If the data sequence PID does not match the expected, the entirety of the data received in this split transaction is ignored, the transfer state is not advanced and this state is exited.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced, and this state is exited. The host controller must reload *CErr* with maximum value on this response.
- ERR. There was an error during the full- or low-speed transaction. The ERR status bit is set to a one, *Cerr* is decremented, the state of the transfer is not advanced, and this state is exited.
- STALL. The queue is halted (an exit condition of the **Execute Transaction** state). The status field bits: *Active* bit is set to zero and the *Halted* bit is set to a one and the qTD is retired. Responses which are not enumerated in the list or which are received out of sequence are illegal and may result in undefined host controller behavior. The other possible combinations of tests A, B, C, and D may indicate that data or response was lost. Table 30-70 lists the possible combinations and the appropriate action.

Table 30-70. Interrupt IN/OUT Do Complete Split State Execution Criteria

Condition	Action	Description
not(A) not(D)	Ignore QHD	Neither a start nor complete-split is scheduled for the current micro-frame.Host controller should continue walking the schedule.
A not(C)	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	Progress bit check failed. These means a complete-split has been missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (e.g. start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A not(B) C	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	*QH.FrameTag* test failed. This means that exactly one or more *H-Frames* have been skipped. This means complete-splits and have missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (e.g. start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.

Condition	Action	Description
A B C not(D)	Execute complete-split	This is the non-error case where the host controller executes a complete-split transaction.
D	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	This is a degenerate case where the start-split was issued, but all of the complete-splits were skipped and all possible intervening opportunities to detect the missed data failed to fire. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (e.g. start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one. Note: When executing in the context of a *Recovery Path* mode, the host controller is allowed to process the queue head and take the actions indicated above, or it may wait until the queue head is visited in the normal processing mode. Regardless, the host controller must not execute a start-split in the context of a executing in a *Recovery Path* mode.

Managing QH.FrameTag Field

The *QH.FrameTag* field in a queue head is completely managed by the host controller. The rules for setting *QH.FrameTag* are simple:

- **Rule 1:** If transitioning from **Do Start Split** to **Do Complete Split** and the current value of *FRINDEX*[2:0] is 6 *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates split transactions whose start-split and complete-splits are in different *H-Frames* (case 2a, see Figure 30-70).
- **Rule 2:** If the current value of *FRINDEX*[2:0] is 7, *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates staying in **Do Complete Split** for cases 2a, 2b, and 2c (Figure 30-70).
- **Rule 3:** If transitioning from **Do_Start Split** to **Do Complete Split** and the current value of *FRINDEX*[2:0] is not 6, or currently in **Do Complete Split** and the current value of (*FRINDEX*[2:0]) is not 7, *FrameTag* is set to *FRINDEX*[7:3]. This accommodates all other cases (Figure 30-70).

Rebalancing the Periodic Schedule

System software must occasionally adjust a periodic queue head's S-mask and C-mask fields during operation. This need occurs when adjustments to the periodic schedule create a new bandwidth budget and one or more queue head's are assigned new execution footprints (i.e. new S-mask and C-mask values).

It is imperative that System software must not update these masks to new values in the midst of a split transaction. In order to avoid any race conditions with the update, the EHCI host controller provides a simple assist to system software. System software sets the *Inactivate-on-next-Transaction* (*I*) bit to a one to signal the host controller that it intends to update the S-mask and C-mask on this queue head. System

software will then wait for the host controller to observe the *I-bit* is a one and transition the *Active* bit to a zero. The rules for how and when the host controller sets the *Active* bit to zero are enumerated below:

- If the *Active* bit is a zero, no action is taken. The host controller does not attempt to advance the queue when the *I-bit* is a one.
- If the *Active* bit is a one and the *SplitXState* is **DoStart** (regardless of the value of *S-mask*), the host controller will simply set *Active* bit to a zero. The host controller is not required to write the transfer state back to the *current* qTD. Note that if the *S-mask* indicates that a start-split is scheduled for the current micro-frame, the host controller must not issue the start-split bus transaction. It must set the *Active* bit to zero.

System software must save transfer state before setting the *I-bit* to a one. This is required so that it can correctly determine what transfer progress (if any) occurred after the *I-bit* was set to a one and the host controller executed it's final bus-transaction and set *Active* to a zero.

After system software has updated the S-mask and C-mask, it must then reactivate the queue head. Since the *Active* bit and the *I-bit* cannot be updated with the same write, system software needs to use the following algorithm to coherently re-activate a queue head that has been stopped via the *I-bit*.

4. Set the *Halted* bit to a one, then
5. Set the *I-bit* to a zero, then
6. Set the *Active* bit to a one and the *Halted* bit to a zero in the same write.

Setting the *Halted* bit to a one inhibits the host controller from attempting to advance the queue between the time the *I-bit* goes to a zero and the *Active* bit goes to a one.

30.8.3.12.3 Split Transaction Isochronous

Full-speed isochronous transfers are managed using the split-transaction protocol through a USB 2.0 transaction translator in a USB2.0 Hub. The EHCI controller utilizes siTD data structure to support the special requirements of isochronous split-transactions. This data structure uses the scheduling model of isochronous TDs (iTD, Section Isochronous (High-Speed) Transfer Descriptor (iTD)) (see Section Managing Isochronous Transfers Using iTDs for the operational model of iTDs) with the contiguous data feature provided by queue heads. This simple arrangement allows a single isochronous scheduling model and adds the additional feature that all data received from the endpoint (per split transaction) must land into a contiguous buffer.

Split Transaction Scheduling Mechanisms for Isochronous

Full-speed isochronous transactions are managed through a transaction translator's periodic pipeline. As with full- and low-speed interrupt, system software manages each transaction translator's periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each full-speed isochronous endpoint occur. The requirements described in Section Split Transaction Scheduling Mechanisms for Interrupt apply. Figure 30-70 illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule. The ^{s}x and ^{c}x labels indicate micro-frames where software can schedule start- and complete-splits (respectively). The *H-Frame* boundaries are marked with a large, solid bold vertical line. The *B-Frame* boundaries are marked with a large, bold, dashed line. The bottom of the figure illustrates the relationship of an siTD to the *H-Frame*.

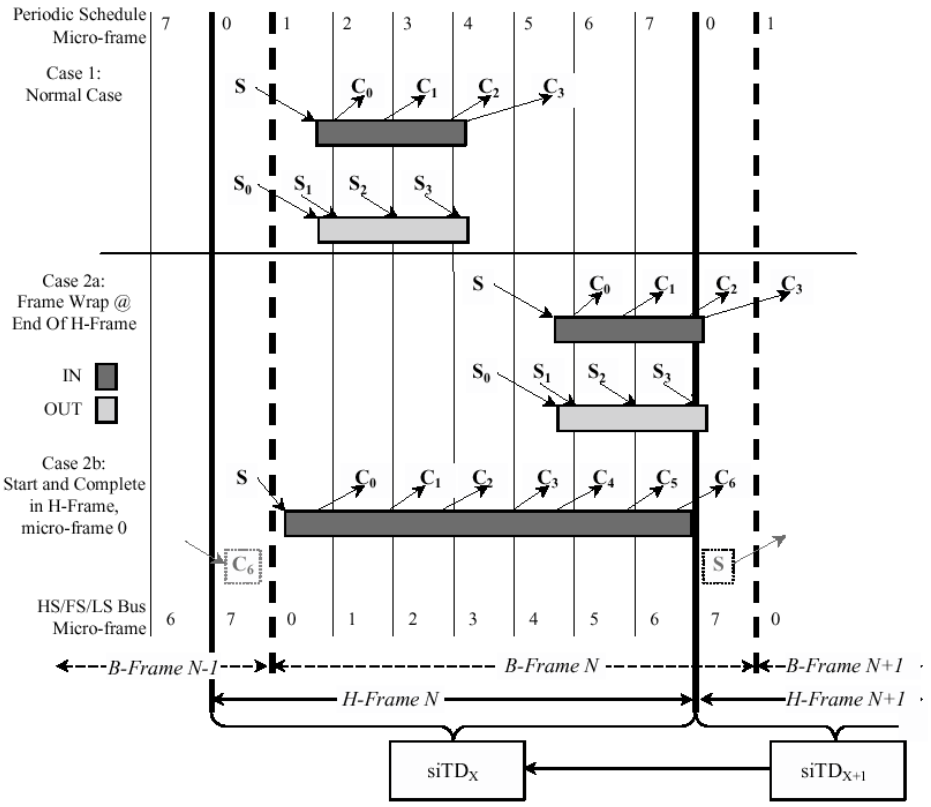


Figure 30-70. Split Transaction, Isochronous Scheduling Boundary Conditions

When the endpoint is an isochronous OUT, there are only start-splits, and no complete-splits. When the endpoint is an isochronous IN, there is at most one start-split and one to *N* complete-splits. The scheduling boundary cases are:

- *Case 1*: The entire split transaction is completely bounded by an *H-Frame.* For example: the start-splits and complete-splits are all scheduled to occur in the same *H-Frame*.
- *Case 2a*: This boundary case is where one or more (at most two) complete-splits of a split transaction IN are scheduled across an *H-Frame* boundary. This can only occur when the split transaction has the possibility of moving data in *B-Frame,* micro-frames 6 or 7 (*H-Frame* micro-frame 7 or 0). When an *H-Frame* boundary wrap condition occurs, the scheduling of the split transaction spans more than one location in the periodic list.(e.g. it takes two siTDs in adjacent periodic frame list locations to fully describe the scheduling for the split transaction).
- Although the scheduling of the split transaction may take two data structures, all of the complete-splits for each full-speed IN isochronous transaction must use only one data pointer. For this reason, siTDs contain a back pointer, the use of which is described below.
- Software must never schedule full-speed isochronous OUTs across an *H-Frame* boundary.
- *Case 2b*: This case can only occur for a very large isochronous IN. It is the only allowed scenario where a start-split and complete-split for the same endpoint can occur in the same micro-frame. Software must enforce this rule by scheduling the large transaction first. Large is defined to be anything larger than 579 byte maximum packet size.

A subset of the same mechanisms employed by full- and low-speed interrupt queue heads are employed in siTDs to schedule and track the portions of isochronous split transactions. The following fields are initialized by system software to instruct the host controller when to execute portions of the split transaction protocol.

- *SplitXState*. This is a single bit residing in the *Status* field of an siTD (see Table 30-46). This bit is used to track the current state of the split transaction. The rules for managing this bit are described in Section In the following presentation, all references to micro-frame are in the context of a micro-frame within an H-Frame.Split Transaction Execution State Machine for Isochronous .
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 30-70, case one, the *S-mask* would have a value of 00000001b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates **Do Start Split**, and the current micro-frame as indicated by FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask.* This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 30-70, case one, the *C-mask* would have a value of 00111100b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates **Do Complete Split**, and the current micro-frame as indicated by *FRINDEX*[2:0] is 2, 3, 4, or 5, then execute a complete-split transaction.
- *Back Pointer.* This field in a siTD is used to complete an IN split-transaction using the previous *H-Frame*'s siTD. This is only used when the scheduling of the complete-splits span an *H-Frame* boundary.

There exists a one-to-one relationship between a high-speed isochronous split transaction (including all start- and complete-splits) and one full-speed isochronous transaction. An siTD contains (amongst other things) buffer state and split transaction scheduling information. An siTD's buffer state always maps to one full-speed isochronous data payload. This means that for any full-speed transaction payload, a single siTD's data buffer must be used. This rule applies to both IN an OUTs. An siTD's scheduling information usually also maps to one high-speed isochronous split transaction. The exception to this rule is the *H-Frame* boundary wrap cases mentioned above.

The siTD data structure describes at most, one frame's worth of high-speed transactions and that description is strictly bounded within a frame boundary. Figure 30-71 illustrates some examples. On the top are examples of the full-speed transaction footprints for the boundary scheduling cases described above. In the middle are time-frame references for both the *B-Frames* (HS/FS/LS Bus) and the *H-Frames*. On the bottom is illustrated the relationship between the scope of an siTD description and the time references. Each *H-Frame* corresponds to a single location in the periodic frame list. The implication is that each siTD is reachable from a single periodic frame list location at a time.

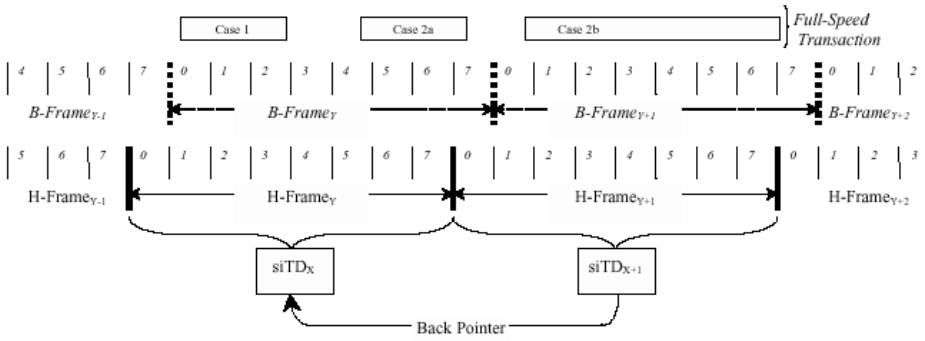


Figure 30-71. siTD Scheduling Boundary Examples

Each case is described below:

- *Case 1*: One siTD is sufficient to describe and complete the isochronous split transaction because the whole isochronous split transaction is tightly contained within a single *H-Frame*.
- *Case 2a, 2b*: Although both INs and OUTs can have these footprints, OUTs always take only one siTD to schedule. However, INs (for these boundary cases) require two siTDs to complete the scheduling of the isochronous split transaction. siTD$_X$ is used to always issue the start-split and the first *N* complete-splits. The full-speed transaction (for these cases) can deliver data on the full-speed bus segment during micro-frame 7 of *H-Frame*$_{Y+1}$, or micro-frame 0 of *H-Frame*$_{Y+2}$. The complete splits are scheduled using siTD$_{X+2}$ (not shown). The complete-splits to extract this data must use the buffer pointer from siTD$_{X+1}$. The only way for the host controller to reach siTD$_{X+1}$ from *H-Frame*$_{Y+2}$ is to use siTD$_{X+2}$'s back pointer. The host controller rules for when to use the back pointer are described is Section Periodic Isochronous - Do Complete Split .

Software must apply the following rules when calculating the schedule and linking the schedule data structures into the periodic schedule:

- Software must ensure that an isochronous split-transaction is started so that it will complete before the end of the *B-Frame*.
- Software must ensure that for a single full-speed isochronous endpoint, there is never a start-split and complete-split in *H-Frame, micro-frame 1*. This is mandated as a rule so that case 2a and case 2b can be discriminated. According to the core USB specification, the long isochronous transaction illustrated in Case 2b, could be scheduled so that the start-split was in micro-frame 1 of *H-Frame* N and the last complete-split would need to occur in micro-frame 1 of H-Frame N+1. However, it is impossible to discriminate between cases 2a and case 2b, which has significant impact on the complexity of the host controller.

Tracking Split Transaction Progress for Isochronous Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where device to host data is lost. Isochronous endpoints do not employ the concept of a halt on error, however the host is required to identify and report per-packet errors observed in the data stream. This includes schedule traversal problems (skipped micro-frames), timeouts and corrupted data received.

In similar kind to interrupt split-transactions, the portions of the split transaction protocol must execute in the micro-frames they are scheduled. The queue head data structure used to manage full- and low-speed interrupt has several mechanisms for tracking when portions of a transaction have occurred. Isochronous transfers use siTDs, for their transfers, and the data structures are only reachable via the schedule in the exact micro-frame in which they are required (so all the mechanism employed for tracking in queue heads is not required for siTDs). Software has the option of reusing siTD several times in the complete periodic schedule. However, it must ensure that the results of split transaction N are consumed and the siTD reinitialized (activated) before the host controller gets back to the siTD (in a future micro-frame).

Split-transaction isochronous OUTs utilize a low-level protocol to indicate which portions of the split transaction data have arrived. Control over the low-level protocol is exposed in an siTD via the fields *Transaction Position* (*TP*) and *Transaction Count* (*T-count*). If the entire data payload for the OUT split transaction is larger than 188 bytes, there will be more than one start-split transaction, each of which require proper annotation. If host hold-offs occur, then the sequence of annotations received from the host will not be complete, which is detected and handled by the transaction translator. See Section Periodic Isochronous - Do Start Split for a description on how these fields are used during a sequence of start-split transactions.

The fields *siTD.T-Count* and *siTD.TP* are used by the host controller to drive and sequence the transaction position annotations. It is the responsibility of system software to properly initialize these fields in each siTD. Once the budget for a split-transaction isochronous endpoint is established, *S-mask*, *T-Count*, and *TP* initialization values for all the siTD associated with the endpoint are constant. They remain constant until the budget for the endpoint is recalculated by software and the periodic schedule adjusted.

For IN-endpoints, the transaction translator simply annotates the response data packets with enough information to allow the host controller to identify the last data. As with split transaction Interrupt, it is the host controller's responsibility to detect when it has missed an opportunity to execute a complete-split. The following field in the siTD is used to track and detect errors in the execution of a split transaction for an IN isochronous endpoint.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets a bit for each complete-split executed. The bit position is determined by the micro-frame (FRINDEX[2:0]) number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a complete-split transaction. If the previous complete-splits have not been executed, then it means one (or more) have been skipped and data has potentially been lost. System software is required to initialize this field to zero before setting an siTD's *Active* bit to a one.

If a transaction translator returns with the final data before all of the complete-splits have been executed, the state of the transfer is advanced so that the remaining complete-splits are not executed. Refer to Section Periodic Isochronous - Do Complete Split for a description on how the state of the transfer is advanced. It is important to note that an IN siTD is retired based solely on the responses from the Transaction Translator to the complete-split transactions. This means, for example, that it is possible for a transaction translator to respond to a complete-split with an MDATA PID. The number of bytes in the MDATA's data

payload could cause the siTD field *Total Bytes to Transfer* to decrement to zero. This response can occur, before all of the scheduled complete-splits have been executed. In other interface, data structures (e.g. high-speed data streams through queue heads), the transition of *Total Bytes to Transfer* to zero signals the end of the transfer and results in setting of the *Active* bit to zero. However, in this case, the result has not been delivered by the Transaction Translator and the host must continue with the next complete-split transaction to extract the residual transaction state. This scenario occurs because of the pipeline rules for a Transaction Translator (see Chapter 11 of the Universal Serial Bus Revision 2.0). In summary the periodic pipeline rules require that on a micro-frame boundary, the Transaction Translator will hold the final two bytes received (if it has not seen an End Of Packet (EOP)) in the full-speed bus pipe stage and give the remaining bytes to the high-speed pipeline stage. At the micro-frame boundary, the Transaction Translator could have received the entire packet (including both CRC bytes) but not received the packet EOP. In the next micro-frame, the Transaction Translator will respond with an MDATA and send all of the data bytes (with the two CRC bytes being held in the full-speed pipeline stage). This could cause the siTD to decrement it's *Total Bytes to Transfer* field to zero, indicating it has received all expected data. The host must still execute one more (scheduled) complete-split transaction in order to extract the results of the full-speed transaction from the Transaction Translator (for example, the Transaction Translator may have detected a CRC failure, and this result must be forwarded to the host).

If the host experiences hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous OUT, then the protocol to the transaction translator will not be consistent and the transaction translator will detect and react to the problem. Likewise, for host hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous IN, the *C-prog-mask* is used by the host controller to detect errors. However, if the host experiences a hold-off that causes it to skip all of an siTD, or an siTD expires during a host hold off (e.g. a hold-off occurs and the siTD is no longer reachable by the host controller in order for it to report the hold-off event), then system software must detect that the siTDs have not been processed by the host controller (e.g. state not advanced) and report the appropriate error to the client driver.

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame*.Split Transaction Execution State Machine for Isochronous

If the *Active* bit in the *Status* byte is a zero, the host controller will ignore the siTD and continue traversing the periodic schedule. Otherwise the host controller will process the siTD as specified below. A split transaction state machine is used to manage the split-transaction protocol sequence. The host controller uses the fields defined in Section Tracking Split Transaction Progress for Isochronous Transfers , plus the variable *cMicroFrameBit* defined in Section Split Transaction Execution State Machine for Interrupt to track the progress of an isochronous split transaction. Figure 30-72 illustrates the state machine for managing an siTD through an isochronous split transaction. Bold, dotted circles denote the state of the *Active* bit in the *Status* field of a siTD. The Bold, dotted arcs denote the transitions between these states. Solid circles denote the states of the split transaction state machine and the solid arcs denote the transitions between these states. Dotted arcs and boxes reference actions that take place either as a result of a transition or from being in a state.

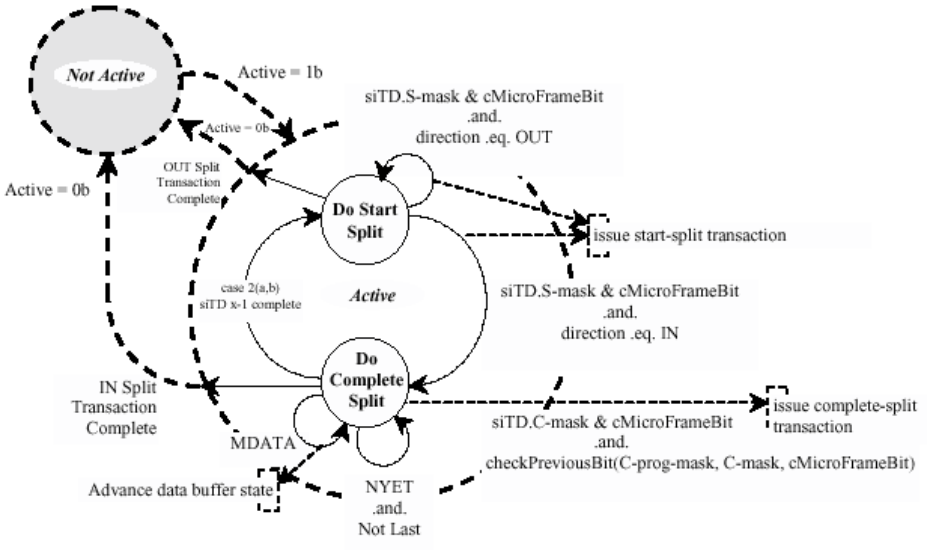


Figure 30-72. Split Transaction State Machine for Isochronous

Periodic Isochronous - Do Start Split

Isochronous split transaction OUTs use only this state. An siTD for a split-transaction isochronous IN is either initialized to this state, or the siTD transitions to this state from **Do Complete Split** when a case 2a (IN) or 2b scheduling boundary isochronous split-transaction completes.

Each time the host controller reaches an active siTD in this state, it checks the *siTD.S-mask* against *cMicroFrameBit*. If there is a one in the appropriate position, the siTD will execute a start-split transaction. By definition, the host controller cannot *reach* an siTD at the wrong time. If the *I/O* field indicates an IN, then the start-split transaction includes only the extended token plus the full-speed token. Software must initialize the *siTD.Total Bytes To Transfer* field to the number of bytes expected. This is usually the maximum packet size for the full-speed endpoint. The host controller exits this state when the start-split transaction is complete.

The remainder of this section is specific to an isochronous OUT endpoint (i.e. the *I/O* field indicates an OUT). When the host controller executes a start-split transaction for an isochronous OUT it includes a data payload in the start-split transaction. The memory buffer address for the data payload is constructed by concatenating *siTD.Current Offset* with the page pointer indicated by the page selector field (*siTD.P*). A zero in this field selects Page 0 and a 1 selects Page 1. During the start-split for an OUT, if the data transfer crosses a page boundary during the transaction, the host controller must detect the page cross, update the *siTD.P*-bit from a zero to a one, and begin using the *siTD.Page 1* with *siTD.Current Offset* as the memory address pointer. The field *siTD.TP* is used to annotate each start-split transaction with the indication of which part of the split-transaction data the current payload represents (ALL, BEGIN, MID, END). In all cases the host controller simply uses the value in *siTD.TP* to mark the start-split with the correct transaction position code.

T-Count is always initialized to the number of start-splits for the current frame. *TP* is always initialized to the first required transaction position identifier. The scheduling boundary case (see Figure 30-71) is used to determine the initial value of *TP*. The initial cases are summarized in Table 30-71.

Table 30-71. Initial Conditions for OUT siTD's TP and T-count Fields

Case	T-count	TP	Description
1, 2a	=1	ALL	When the OUT data payload is less than (or equal to) 188 bytes, only one start-split is required to move the data. The one start-split must be marked with an ALL.
1, 2a	!=1	BEGIN	When the OUT data payload is greater than 188 bytes more than one start-split must be used to move the data. The initial start-split must be marked with a BEGIN.

After each start-split transaction is complete, the host controller updates *T-Count* and *TP* appropriately so that the next start-split is correctly annotated. Table 30-72 illustrates all of the *TP* and *T-count* transitions, which must be accomplished by the host controller.

Table 30-72. Transaction Position (TP)/Transaction Count (T-Count) Transition Table

TP	T-count next	TP next	Description
ALL	0	N/A	Transition from ALL, to done.
BEGIN	1	END	Transition from BEGIN to END. Occurs when *T-count* starts at 2.
BEGIN	!=1	MID	Transition from BEGIN to MID. Occurs when *T-count* starts at greater than 2.
MID	!=1	MID	*TP* stays at MID while *T-count* is not equal to 1 (e.g. greater than 1). This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 3.
MID	1	END	Transition from MID to END. This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 2.

The start-split transactions do not receive a handshake from the transaction translator, so the host controller always advances the transfer state in the siTD after the bus transaction is complete. To advance the transfer state the following operations take place:

- The *siTD.Total Bytes To Transfer* and the *siTD.Current Offset* fields are adjusted to reflect the number of bytes transferred.
- The *siTD.P* (page selector) bit is updated appropriately.
- The *siTD.TP* and *siTD.T-count* fields are updated appropriately as defined in Table 30-72.

These fields are then written back to the memory based siTD. The *S-mask* is fixed for the life of the current budget. As mentioned above, *TP* and *T-count* are set specifically in each siTD to reflect the data to be sent from this siTD. Therefore, regardless of the value of *S-mask*, the actual number of start-split transactions depends on *T-count* (or equivalently, *Total Bytes to Transfer*). The host controller must set the *Active* bit to a zero when it detects that all of the schedule data has been sent to the bus. The preferred method is to detect when *T-Count* decrements to zero as a result of a start-split bus transaction. Equivalently, the host

controller can detect when *Total Bytes to Transfer* decrements to zero. Either implementation must ensure that if the initial condition is *Total Bytes to Transfer* equal to zero and *T-count* is equal to a one, then the host controller will issue a single start-split, with a zero-length data payload. Software must ensure that *TP*, *T-count* and *Total Bytes to Transfer* are set to deliver the appropriate number of bus transactions from each siTD. An inconsistent combination will yield undefined behavior.

If the host experiences hold-offs that cause the host controller to skip start-split transactions for an OUT transfer, the state of the transfer will not progress appropriately. The transaction translator will observe protocol violations in the arrival of the start-splits for the OUT endpoint (i.e. the transaction position annotation will be incorrect as received by the transaction translator).

Example scenarios are described in Section Split Transaction for Isochronous - Processing Examples .

A host controller implementation can optionally track the progress of an OUT split transaction by setting appropriate bits in the *siTD.C-prog-mask* as it executes each scheduled start-split. The *checkPreviousBit*() algorithm defined in Section Periodic Isochronous - Do Complete Split can be used prior to executing each start-split to determine whether start-splits were skipped. The host controller can use this mechanism to detect missed micro-frames. It can then set the siTD's *Active* bit to zero and stop execution of this siTD. This saves on both memory and high-speed bus bandwidth.

Periodic Isochronous - Do Complete Split

This state is only used by a split-transaction isochronous IN endpoint. This state is entered unconditionally from the **Do Start State** after a start-split transaction is executed for an IN endpoint. Each time the host controller visits an siTD in this state, it conducts a number of tests to determine whether it should execute a complete-split transaction. The individual tests are listed below. The sequence they are applied depends on which micro-frame the host controller is currently executing which means that the tests might not be applied until after the siTD referenced from the back pointer has been fetched.

- **Test A**. *cMicroFrameBit* is bit-wise anded with *siTD.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame. This test is always applied to a newly fetched siTD that is in this state.
- **Test B**. The *siTD.C-prog-mask* bit vector is checked to determine whether the previous complete splits have been executed. An example algorithm is below (this is slightly different than the algorithm used in Section Periodic Interrupt - Do Complete Split). The sequence in which this test is applied depends on the current value of FRINDEX[2:0]. If FRINDEX[2:0] is 0 or 1, it is not applied until the back pointer has been used. Otherwise it is applied immediately.

Algorithm Boolean CheckPreviousBit(`siTD.C-prog-mask, siTD.C-mask, cMicroFrameBi`**t)**

```
Begin

        Boolean rvalue = TRUE;

        previousBit = cMicroFrameBit rotate-right(1)

        -- Bit-wise anding previousBit with C-mask indicates whether there was an intent

        -- to send a complete split in the previous micro-frame. So, if the

        -- 'previous bit' is set in C-mask, check C-prog-mask to make sure it

        -- happened.
```

```
        if previousBit bitAND siTD.C-mask then
                if not (previousBit bitAND siTD.C-prog-mask) then
                        rvalue = FALSE
                End if
        End if
        Return rvalue
End Algorithm
```

If Test A is true and FRINDEX[2:0] is zero or one, then this is a case 2a or 2b scheduling boundary (see Figure 30-70). See Section Periodic Isochronous - Do Complete Split for details in handling this condition.

If Test A and Test B evaluate to true, then the host controller will execute a complete-split transaction using the transfer state of the current siTD. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. The transfer state is advanced based on the completion status of the complete-split transaction. To advance the transfer state of an IN siTD, the host controller must:

- Decrement the number of bytes received from *siTD.Total Bytes To Transfer*,
- Adjust *siTD.Current Offset* by the number of bytes received,
- Adjust *siTD.P* (page selector) field if the transfer caused the host controller to use the next page pointer, and
- Set any appropriate bits in the *siTD.Status* field, depending on the results of the transaction.

Note that if the host controller encounters a condition where *siTD.Total Bytes To Transfer* is zero, and it receives more data, the host controller must not write the additional data to memory. The *siTD.Status.Active* bit must be set to zero and the *siTD.Status.Babble Detected* bit must be set to a one. The fields *siTD.Total Bytes To Transfer*, *siTD.Current Offset*, and *siTD.P* (page selector) are not required to be updated as a result of this transaction attempt.

The host controller must accept (assuming good data packet CRC and sufficient room in the buffer as indicated by the value of *siTD.Total Bytes To Transfer*) MDATA and DATA0/1 data payloads up to and including 192 bytes. A host controller implementation may optionally set *siTD.Status Active* to a zero and *siTD.Status.Babble Detected* to a one when it receives and MDATA or DATA0/1 with a data payload of more than 192 bytes. The following responses have the noted effects:

- ERR. The full-speed transaction completed with a time-out or bad CRC and this is a reflection of that error to the host. The host controller sets the *ERR* bit in the *siTD.Status* field and sets the *Active* bit to a zero.
- Transaction Error (XactErr). The complete-split transaction encounters a Timeout, CRC16 failure, etc. The *siTD.Status* field *XactErr* field is set to a one and the complete-split transaction must be retried immediately. The host controller must use an internal error counter to count the number of retries as a counter field is not provided in the siTD data structure. The host controller will not retry

more than two times. If the host controller exhausts the retries or the end of the micro-frame occurs, the *Active* bit is set to zero.

- DATAx (0 or 1). This response signals that the final data for the split transaction has arrived. The transfer state of the siTD is advanced and the *Active* bit is set to a zero. If the *Bytes To Transfer* field has not decremented to zero (including the reception of the data payload in the DATAx response), then less data than was expected, or allowed for was actually received. This *short packet* event does not set the USBINT status bit in the USBSTS register to a one. The host controller will not detect this condition.
- NYET (and Last). On each NYET response, the host controller also checks to determine whether this is the last complete-split for this split transaction. Last was defined in Section Periodic Interrupt - Do Complete Split . If it is the last complete-split (with a NYET response), then the transfer state of the siTD is not advanced (never received any data) and the *Active* bit is set to a zero. No bits are set in the *Status* field because this is essentially a skipped transaction. The transaction translator must have responded to all the scheduled clompete-splits with NYETs, meaning that the start-split issued by the host controller was not received. This result should be interpreted by system software as if the transaction was completely skipped. The test for whether this is the last complete split can be performed by XORing C-mask with C-prog-mask. A zero result indicates that all complete-splits have been executed.
- MDATA (and Last). See above description for testing for **Last**. This can only occur when there is an error condition. Either there has been a babble condition on the full-speed link, which delayed the completion of the full-speed transaction, or software set up the *S-mask* and/or *C-mask*s incorrectly. The host controller must set *XactErr* bit to a one and the *Active* bit is set to a zero.
- NYET (and not Last). See above description for testing for **Last**. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask*) and stay in this state.
- MDATA (and not Last). The transaction translator responds with an MDATA when it has partial data for the split transaction. For example, the full-speed transaction data payload spans from micro-frame X to X+1 and during micro-frame X, the transaction translator will respond with an MDATA and the data accumulated up to the end of micro-frame X. The host controller advances the transfer state to reflect the number of bytes received.

If Test A succeeds, but Test B fails, it means that one or more of the complete-splits have been skipped. The host controller sets the *Missed Micro-Frame* status bit and sets the *Active* bit to a zero.

Complete-Split for Scheduling Boundary Cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 30-70) require that the host controller use the transaction state context of the previous siTD to finish the split transaction. Table 30-73 enumerates the transaction state fields.

Table 30-73. Summary siTD Split Transaction State

Buffer State	Status	Execution Progress
Total Bytes To Transfer P (page select) Current Offset TP (transaction position) T-count (transaction count)	All bits in the status field	C-prog-mask

Note: *TP* and *T-count* are used only for Host to Device (OUT) endpoints.

If software has budgeted the schedule of this data stream with a frame wrap case, then it must initialize the *siTD.Back Pointer* field to reference a valid siTD and will have the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer*

field set to a zero. Otherwise, software must set the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer* field to a one. The host controller's rules for interpreting when to use the *siTD.Back Pointer* field are listed below. These rules apply only when the siTD's *Active* bit is a one and the *SplitXState* is **Do Complete Split**.

- When cMicroFrameBit is a 1h and the siTDX.Back Pointer.T-bit is a zero, or
- If cMicroFrameBit is a 2h and siTDX.S-mask[0] is a zero

When either of these conditions apply, then the host controller must use the transaction state from $siTD_{X-1}$.

In order to access $siTD_{X-1}$, the host controller reads on-chip the siTD referenced from $siTD_X$*.Back Pointer*.

The host controller must save the entire state from $siTD_X$ while processing $siTD_{X-1}$. This is to accommodate for case 2b processing. The host controller must not recursively walk the list of *siTD.Back Pointers*.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is **Do Complete Split**), then both Test A and Test B are applied as described above. If these criteria to execute a complete-split are met, the host controller executes the complete split and evaluates the results as described above. The transaction state (see Table 30-73) of $siTD_{X-1}$ is appropriately advanced based on the results and written back to memory. If the resultant state of $siTD_{X-1}$'s *Active* bit is a one, then the host controller returns to the context of $siTD_X$, and follows its next pointer to the next schedule item. No updates to $siTD_X$ are necessary.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is **Do Start Split**), then the host controller must set *Active* bit to a zero and *Missed Micro-Frame* status bit to a one and the resultant status written back to memory.

If $siTD_{X-1}$'s *Active* bit is a zero, (because it was zero when the host controller first visited $siTD_{X-1}$ via $siTD_X$'s back pointer, it transitioned to zero as a result of a detected error, or the results of $siTD_{X-1}$'s complete-split transaction transitioned it to zero), then the host controller returns to the context of $siTD_X$ and transitions its *SplitXState* to **Do Start Split**. The host controller then determines whether the case 2b start split boundary condition exists (i.e. if *cMicroframeBit* is a 1b and $siTD_X$*.S-mask*[0] is a 1b). If this criterion is met the host controller immediately executes a start-split transaction and appropriately

advances the transaction state of $siTD_X$, then follows $siTD_X$.*Next Pointer* to the next schedule item. If the criterion is not met, the host controller simply follows $siTD_X$.*Next Pointer* to the next schedule item. Note that in the case of a 2b boundary case, the split-transaction of $siTD_{X-1}$ will have its *Active* bit set to zero when the host controller returns to the context of $siTD_X$. Also, note that software should not initialize an siTD with *C-mask* bits 0 and 1 set to a one and an *S-mask* with bit zero set to a one. This scheduling combination is not supported and the behavior of the host controller is undefined.

Split Transaction for Isochronous - Processing Examples

There is an important difference between how the hardware/software manages the isochronous split transaction state machine and how it manages the asynchronous and interrupt split transaction state machines. The asynchronous and interrupt split transaction state machines are encapsulated within a single queue head. The progress of the data stream depends on the progress of each split transaction. In some respects, the split-transaction state machine is sequenced via the **Execute Transaction** queue head traversal state machine (see Figure 30-63).

Isochronous is a pure time-oriented transaction/data stream. The interface data structures are optimized to efficiently describe transactions that need to occur at specific times. The isochronous split-transaction state machine must be managed across these time-oriented data structures. This means that system software must correctly describe the scheduling of split-transactions across more than one data structure.

Then the host controller must make the appropriate state transitions at the appropriate times, in the correct data structures.

For example, Table 30-74 illustrates a couple of frames worth of scheduling required to schedule a case 2a full-speed isochronous data stream.

Table 30-74. Example Case 2a - Software Scheduling siTDs for an IN Endpoint

siTDx		Micro-Frames								Initial SplitXState
#	Masks	0	1	2	3	4	5	6	7	
X	S-Mask					1				Do Start Split
	C-Mask	1	1					1	1	
X+1	S-Mask					1				Do Complete Split
	C-Mask	1	1					1	1	
X+2	S-Mask					1				Do Complete Split
	C-Mask	1	1					1	1	
X+3	S-Mask	Repeats previous pattern								Do Complete Split
	C-Mask									

This example shows the first three siTDs for the transaction stream. Since this is the case-2a frame-wrap case, *S-mask*s of all siTDs for this endpoint have a value of 10h (a one bit in micro-frame 4) and *C-mask* value of C3h (one-bits in micro-frames 0,1, 6 and 7). Additionally, sofware ensures that the *Back Pointer* field of each siTD references the appropriate siTD data structure (and the *Back Pointer T-bits* are set to zero).

The initial *SplitXState* of the first siTD is **Do Start Split**. The host controller will visit the first siTD eight times during frame X. The C-mask bits in micro-frames 0 and 1 are ignored because the state is **Do Start Split**. During micro-frame 4, the host controller determines that it can run a start-split (and does) and changes *SplitXState* to **Do Complete Split**. During micro-frames 6 and 7, the host controller executes complete-splits. Notice the siTD for frame X+1 has it's *SplitXState* initialized to **Do Complete Split**. As the host controller continues to traverse the schedule during *H-Frame* X+1, it will visit the second siTD eight times. During micro-frames 0 and 1 it will detect that it must execute complete-splits.

During *H-Frame* X+1, micro-frame 0, the host controller detects that $siTD_{X+1}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+1}$ and fetches $siTD_X$. It executes the complete split transaction using the transaction state of $siTD_X$. If the $siTD_X$ split transaction is complete, siTD's *Active* bit is set to zero and results written back to $siTD_X$. The host controller retains the fact that $siTD_X$ is retired and transitions the *SplitXState* in the $siTD_{X+1}$ to **Do Start Split.** At this point, the host controller is prepared to execute the start-split for $siTD_{X+1}$ when it reaches micro-frame 4. If the split-transaction completes early (transaction-complete is defined in Section Periodic Isochronous - Do Complete Split), i.e. before all the scheduled complete-splits have been executed, the host controller will transition $siTD_X$.*SplitXState* to **Do Start Split** early and naturally skip the remaining scheduled complete-split transactions. For this example, $siTD_{X+1}$ does not receive a DATA0 response until *H-Frame* X+2, micro-frame 1.

During *H-Frame X+2,* micro-frame 0, the host controller detects that $siTD_{X+2}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. As described above, it executes another split transaction, receives an MDATA response, updates the transfer state, but does not modify the *Active* bit. The host controller returns to the context of $siTD_{X+2}$, and traverses it's next pointer without any state change updates to $siTD_{X+2}$. S

During *H-Frame X+2,* micro-frame 1, the host controller detects $siTD_{X+2}$'s *S-mask[0]* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. It executes another complete-split transaction, receives a DATA0 response, updates the transfer state and sets the *Active* bit to a zero. It returns to the state of $siTD_{X+2}$ and changes its *SplitXState* to **Do Start Split**. At this point, the host controller is prepared to execute start-splits for $siTD_{X+2}$ when it reaches micro-frame 4. <TBD… describe how software detects that there was missing micro-frames (don't think we care about missing out micro-frames. There is enough residual state to identify than not all transactions were executed.).

30.8.3.13 Host Controller Pause

When the host controller's *HCHalted* bit in the USBSTS register is a zero, the host controller is sending SOF (Start OF Frame) packets down all enabled ports. When the schedules are enabled, the EHCI host controller will access the schedules in main memory each micro-frame. This constant pinging of main memory is known to create CPU power management problems for mobile systems. Specifically, mobile systems aggressively manage the state of the CPU, based on recent history usage. In the more aggressive power saving modes, the CPU can disable its caches. Current PC architectures assume that bus-master accesses to main memory must be cache-coherent. So, when bus masters are busy touching memory, the CPU power management software can detect this activity over time and inhibit the transition of the CPU into its lowest power savings mode. USB controllers are bus-masters and the frequency at which they access their memory-based schedules keeps the CPU power management software from placing the CPU into its lowest power savings state.

USB Host controllers don't access main memory when they are suspended. However, there are a variety of reasons why placing the USB controllers into suspend won't work, but they are beyond the scope of this document. The base requirement is that the USB controller needs to be kept out of main memory, while at the same time, the USB bus is kept from going into suspend.

EHCI controllers provide a large-grained mechanism that can be manipulated by system software to change the memory access pattern of the host controller. System software can manipulate the schedule enable bits in the USBCMD register to turn on/off the scheduling traversal. A software heuristic can be applied to implement an on/off duty cycle that allows the USB to make reasonable progress and allow the CPU power management to get the CPU into its lowest power state. This method is not intended to be applied at all times to throttle USB, but should only be applied in very specific configurations and usage loads. For example, when only a keyboard or mouse is attached to the USB, the heuristic could detect times when the USB is attempting to move data only very infrequently and can adjust the duty cycle to allow the CPU to reach it's low power state for longer periods of time. Similarly, it could detect increases in the USB load and adjust the duty cycle appropriately, even to the point where the schedules are never disabled. The assumption here is that the USB is moving data and the CPU will be required to process the data streams.

It is suggested that in order to provide a complete solution for the system, the companion host controllers should also provide a similar method to allow system software to inhibit the companion host controller from accessing it's shared memory based data structures (schedule lists or otherwise).

30.8.3.14 Port Test Modes

EHCI host controllers must implement the port test modes **Test J_State**, **Test K_State**, **Test_Packet**, **Test Force_Enable**, and **Test SE0_NAK** as described in the USB Specification Revision 2.0. The system is only allowed to test ports that are owned by the EHCI controller (e.g. *CF-bit* is a one and *PortOwner* bit is a zero). System software is allowed to have at most one port in test mode at a time. Placing more than one port in test mode will yield undefined results. The required, per port test sequence is (assuming the *CF-bit* in the CONFIGFLAG register is a one):

- Disable the periodic and asynchronous schedules by setting the *Asynchronous Schedule Enable* and *Periodic Schedule Enable* bits in the USBCMD register to a zero.
- Place all enabled root ports into the suspended state by setting the *Suspend* bit in each appropriate PORTSC register to a one.
- Set the *Run/Stop* bit in the USBCMD register to a zero and wait for the *HCHalted* bit in the USBSTS register, to transition to a one. Note that an EHCI host controller implementation may optionally allow port testing with the *Run/Stop* bit set to a one. However, all host controllers must support port testing with *Run/Stop* set to a zero and *HCHalted* set to a one.
- Set the *Port Test Control* field in the port under test PORTSC register to the value corresponding to the desired test mode. If the selected test is **Test_Force_Enable**, then the *Run/Stop* bit in the USBCMD register must then be transitioned back to one, in order to enable transmission of SOFs out of the port under test.
- When the test is complete, system software must ensure the host controller is halted (*HCHalted* bit is a one) then it terminates and exits test mode by setting *HCReset* to a one.

30.8.3.15 Interrupts

The EHCI Host Controller hardware provides interrupt capability based on a number of sources. There are several general groups of interrupt sources:

- Interrupts as a result of executing transactions from the schedule (success and error conditions),
- Host controller events (Port change events, etc.), and
- Host Controller error events

All transaction-based sources are maskable through the Host Controller's Interrupt Enable register (USBINTR, see Section USBINTR). Additionally, individual transfer descriptors can be marked to generate an interrupt on completion. This section describes each interrupt source and the processing that occurs in response to the interrupt.

During normal operation, interrupts may be immediate or deferred until the next interrupt threshold occurs. The interrupt threshold is a tunable parameter via the *Interrupt Threshold Control* field in the USBCMD register. The value of this register controls when the host controller will generate an interrupt on behalf of normal transaction execution. When a transaction completes during an interrupt interval period, the interrupt signaling the completion of the transfer will not occur until the interrupt threshold occurs. For example, the default value is eight micro-frames. This means that the host controller will not generate interrupts any more frequently than once every eight micro-frames.

Section Host System Error details effects of a host system error.

If an interrupt has been scheduled to be generated for the current interrupt threshold interval, the interrupt is not signaled until after the status for the last complete transaction in the interval has been written back to host memory. This may sometimes result in the interrupt not being signaled until the next interrupt threshold.

Initial interrupt processing is the same, regardless of the reason for the interrupt. When an interrupt is signaled by the hardware, CPU control is transferred to host controller's USB interrupt handler. The precise mechanism to accomplish the transfer is OS specific. For this discussion it is just assumed that control is received. When the interrupt handler receives control, its first action is to reads the USBSTS (USB Status Register). It then acknowledges the interrupt by clearing all of the interrupt status bits by writing ones to these bit positions. The handler then determines whether the interrupt is due to schedule processing or some other event. After acknowledging the interrupt, the handler (via an OS-specific mechanism), schedules a deferred procedure call (DPC) which will execute later. The DPC routine processes the results of the schedule execution. The precise mechanisms used are beyond the scope of this document.

Note: the host controller is not required to de-assert a currently active interrupt condition when software sets the interrupt enables (in the USBINR register, see Section USBINTR) to a zero. The only reliable method software should use for acknowledging an interrupt is by transitioning the appropriate status bits in the USBSTS register (Section USBSTS) from a one to a zero.

30.8.3.15.1 Transfer/Transaction Based Interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

Transaction Error

A transaction error is any error that caused the host controller to think that the transfer did not complete successfully. Table 30-75 lists the events/responses that the host can observe as a result of a transaction. The effects of the error counter and interrupt status are summarized in the following paragraphs. Most of these errors set the *XactErr* status bit in the appropriate interface data structure.

There is a small set of protocol errors that relate only when executing a queue head and fit under the umbrella of a WRONG PID error that are significant to explicitly identify. When these errors occur, the *XactErr* status bit in the queue head is set and the *CErr* field is decremented. When the *PIDCode* indicates a SETUP, the following responses are protocol errors and result in *XactErr* bit being set to a one and the *CErr* field being decremented.

- *EPS* field indicates a high-speed device and it returns a Nak handshake to a SETUP.
- *EPS* field indicates a high-speed device and it returns a Nyet handshake to a SETUP.
- *EPS* field indicates a low- or full-speed device and the complete-split receives a Nak handshake.

Table 30-75. Summary of Transaction Errors

Event / Result	Queue Head/qTD/iTD/siTD Side-effects		USB Status Register (USBSTS)
	Cerr	Status Field	USBERRINT
CRC	-1	XactErr set to a one.	1[1]
Timeout	-1	XactErr set to a one.	1[1]
Bad PID[2]	-1	XactErr set to a one.	1[1]
Babble	N/A	Section Serial Bus Babble	1
Buffer Error	N/A	Section Data Buffer Error	

1 If occurs in a queue head, then *USBERRINT* is asserted only when *CErr* counts down from a one to a zero. In addition the queue is halted, see Section Section , “Halting a Queue Head .”

2 The host controller received a response from the device, but it could not recognize the PID as a valid PID.

Serial Bus Babble

When a device transmits more data on the USB than the host controller is expecting for this transaction, it is defined to be babbling. In general, this is called a *Packet Babble*. When a device sends more data than the *Maximum Length* number of bytes, the host controller sets the *Babble Detected* bit to a one and halts the endpoint if it is using a queue head (see Section Section , “Halting a Queue Head ”). *Maximum Length* is defined as the minimum of *Total Bytes to Transfer* and *Maximum Packet Size*. The *CErr* field is not decremented for a packet babble condition (only applies to queue heads). A babble condition also exists if IN transaction is in progress at High-speed EOF2 point. This is called a frame babble. A frame babble condition is recorded into the appropriate schedule data structure. In addition, the host controller must disable the port to which the frame babble is detected.

The *USBERRINT* bit in the USBSTS register is set to a one and if the *USB Error Interrupt Enable* bit in the USBINTR register is a one, then a hardware interrupt is signaled to the system at the next interrupt

threshold. The host controller must never start an OUT transaction that will babble across a micro-frame EOF.

NOTE

When a host controller detects a data PID mismatch, it must either: disable the packet babble checking for the duration of the bus transaction or do packet babble checking based solely on *Maximum Packet Size*. The USB core specification defines the requirements on a data receiver when it receives a data PID mismatch (e.g. expects a DATA0 and gets a DATA1 or visa-versa). In summary, it must ignore the received data and respond with an ACK handshake, in order to advance the transmitter's data sequence.

The EHCI interface allows System software to provide buffers for a Control, Bulk or Interrupt IN endpoint that are not an even multiple of the maximum packet size specified by the device. Whenever a device misses an ACK for an IN endpoint, the host and device are out of synchronization with respect to the progress of the data transfer. The host controller may have advanced the transfer to a buffer that is less than maximum packet size. The device will re-send its maximum packet size data packet, with the original data PID, in response to the next IN token. In order to properly manage the bus protocol, the host controller must disable the packet babble check when it observes the data PID mismatch.

Data Buffer Error

This event indicates that an overrun of incoming data or a underrun of outgoing data has occurred for this transaction. This would generally be caused by the host controller not being able to access required data buffers in memory within necessary latency requirements. These conditions are not considered transaction errors, and do not effect the error count in the queue head. When these errors do occur, the host controller records the fact the error occurred by setting the *Data Buffer Error* bit in the queue head, iTD or siTD.

If the data buffer error occurs on a non-isochronous IN, the host controller will not issue a handshake to the endpoint. This will force the endpoint to resend the same data (and data toggle) in response to the next IN to the endpoint.

If the data buffer error occurs on an OUT, the host controller must corrupt the end of the packet so that it cannot be interpreted by the device as a good data packet. Simply truncating the packet is not considered acceptable. An acceptable implementation option is to 1's complement the CRC bytes and send them. There are other options suggested in the Transaction Translator section of the USB Specification Revision 2.0.

USB Interrupt (Interrupt on Completion (IOC))

Transfer Descriptors (iTDs, siTDs, and queue heads (qTDs)) contain a bit that can be set to cause an interrupt on their completion. The completion of the transfer associated with that schedule item causes the USB Interrupt (USBINT) bit in the USBSTS register to be set to a one. In addition, if a short packet is encountered on an IN transaction associated with a queue head, then this event also causes USBINT to be set to a one. If the USB Interrupt Enable bit in the USBINTR register is set to a one, a hardware interrupt

is signaled to the system at the next interrupt threshold. If the completion is because of errors, the *USBERRINT* bit in the USBSTS register is also set to a one.

Short Packet

Reception of a data packet that is less than the endpoint's Max Packet size during Control, Bulk or Interrupt transfers signals the completion of the transfer. Whenever a short packet completion occurs during a queue head execution, the *USBINT* bit in the USBSTS register is set to a one. If the *USB Interrupt Enable* bit is set in the USBINTR register, a hardware interrupt is signaled to the system at the next interrupt threshold.

30.8.3.15.2 Host Controller Event Interrupts

These interrupt sources are independent of the interrupt threshold (with the one exception being the Interrupt on Async Advance, see Section Interrupt on Async Advance).

Port Change Events

Port registers contain status and status change bits. When the status change bits are set to a one, the host controller sets the *Port Change Detect* bit in the USBSTS register to a one. If the *Port Change Interrupt Enable* bit in the USBINTR register is a one, then the host controller will issue a hardware interrupt. The port status change bits include:

- Connect Status Change
- Port Enable/Disable Change
- Over-current Change
- Force Port Resume

Frame List Rollover

This event indicates that the host controller has wrapped the frame list. The current programmed size of the frame list effects how often this interrupt occurs. If the frame list size is 1024, then the interrupt will occur every 1024 milliseconds, if it is 512, then it will occur every 512 milliseconds, etc. When a frame list rollover is detected, the host controller sets the *Frame List Rollover* bit in the USBSTS register to a one. If the *Frame List Rollover Enable* bit in the USBINTR register is set to a one, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

Interrupt on Async Advance

This event is used for deterministic removal of queue heads from the asynchronous schedule. Whenever the host controller advances the on-chip context of the asynchronous schedule, it evaluates the value of the *Interrupt on Async Advance Doorbell* bit in the USBCMD register. If it is a one, it sets the *Interrupt on Async Advance* bit in the USBSTS register to a one. If the *Interrupt on Async Advance Enable* bit in the USBINTR register is a one, the host controller issues a hardware interrupt at the next interrupt threshold. A detailed explanation of this feature is described in Section Removing Queue Heads from Asynchronous Schedule .

Host System Error

The host controller is a bus master and any interaction between the host controller and the system may experience errors. The type of host error may be catastrophic to the host controller (such as a Master Abort) making it impossible for the host controller to continue in a coherent fashion. In the presence of non-catastrophic host errors, such as parity errors, the host controller could potentially continue operation. The recommended behavior for these types of errors is to escalate it to a catastrophic error and halt the host controller. Host-based error must result in the following actions:

- The *Run/Stop* bit in the USBCMD register is set to a zero.
- The following bits in the USBSTS register are set:
  - *Host System Error* bit is to a one.
  - *HCHalted* bit is set to a one.
- If the *Host System Error Enable* bit in the USBINTR register is a one, then the host controller will issue a hardware interrupt. This interrupt is not delayed to the next interrupt threshold. Table 30-76 summarizes the required actions taken on the various host errors.

Table 30-76. Summary Behavior of EHCI Host Controller on Host System Errors

Cycle Type	Master Abort	Target Abort	Data Phase Parity
Frame list pointer fetch (read)	Fatal	Fatal	Fatal [o]
siTD fetch (read)	Fatal	Fatal	Fatal [o]
siTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
iTD fetch (read)	Fatal	Fatal	Fatal [o]
iTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
qTD fetch (read)	Fatal	Fatal	Fatal [o]
qHD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
Data write	Fatal [o]	Fatal [o]	Fatal [o]
Data read	Fatal	Fatal	Fatal [o]

[o] Potentially, a host controller implementation could continue operation without a halt. However, the recommended behavior is to halt the host controller.

NOTE

After a *Host System Error*, Software must reset the host controller via *HCReset* in the USBCMD register before re-initializing and restarting the host controller.

30.8.4 EHCI Deviation

For the purposes a dual-role Host/Device controller with support for On-The-Go applications, it is necessary to deviate from the EHCI specification Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com. Device operation & On-The-Go operation is not specified in the EHCI and thus the implementation supported in this core is proprietary. The host mode operation of the core is near EHCI compatible with few minor differences documented in this section.

The particulars of the deviations occur in the areas summarized here:

- Embedded Transaction Translator—Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Device operation - In host mode the device operational registers are generally disabled and thus device mode is mostly transparent when in host mode. However, there are a couple exceptions documented in the following sections.
- Embedded design interface - This core does not a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.
- On-The-Go Operation - This design includes an On-The-Go controller for Port #1.

30.8.4.1 Embedded Transaction Translator Function

The ARC USB-HS OTG High-Speed USB On-The-Go OTG controller supports directly connected full and low speed devices without requiring a companion controller by including the capabilities of a USB 2.0 high speed hub transaction trasnslator. Although there is no separate Tranaction Translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and Protocol engine blocks. The embedded transaction translator fuction is an extension to EHCI interface, but makes use of the standard data structures and operational models that exist in the EHCI specification to support full and low speed devices.

30.8.4.1.1 Capability Registers

The following additions have been added to the capability registers to support the embedded Transaction Translator Function:

- N_TT added to HCSPARAMS—Host Control Structural Parameters
- N_PTT added toHCSPARAMS—Host Control Structural Parameters

30.8.4.1.2 Operational Registers

The following additions have been added to the operational registers to support the embedded TT:

- is a new register.
- Addition of two-bit Port Speed (PSPD) to the PORTSCx register.

30.8.4.1.3 Discovery

In a standard EHCI controller design, the EHCI host controller driver detects a Full speed (FS) or Low speed (LS) device by noting if the port enable bit is set after the port reset operation. The port enable will

only be set in a standard EHCI controller implementation after the port reset operation and when the host and device negotiate a High-Speed connection (i.e. Chirp completes successfully).

Since this controller has an embedded Transaction Translator, the port enable will always be set after the port reset operation regardless of the result of the host device chirp result and the resulting port speed will be indicated by the PSPD field in PORTSCx.

Therefore, the standard EHCI host controller driver requires an alteration to handle directly connected Full and Low speed devices or hubs.

The change is a fundamental one in that is summarized in Table 30-77.

Table 30-77. Summary of EHCI

Standard EHCI	EHCI with Embedded Transaction Translator
After port enable bit is set following a connection and reset sequence, the device/hub is assumed to be HS.	After port enable bit is set following a connection and reset sequence, the device/hub speed is noted from PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub thus, all port-level control is performed through the Hub Class to the nearest Hub.	FS and LS device can be either downstream from a HS hub or directly attached. When the FS/LS device is downstream from a HS hub, then port-level control is done using the Hub Class through the nearest Hub. When a FS/LS device is directly attached, then port-level control is accomplished using PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub with HubAddr=X. [where HubAddr > 0 and HubAddr is the address of the Hub where the bus transitions from HS to FS/LS (ie. Split target hub)]	FS and LS device can be either downstream from a HS hub with HubAddr = X [HubAddr > 0] or directly attached [where HubAddr = 0 and HubAddr is the address of the Root Hub where the bus transitions from HS to FS/LS (ie. Split target hub is the root hub)]

30.8.4.1.4 Data Structures

The same data structures used for FS/LS transactions though a HS hub are also used for transactions through the Root Hub with sm embedded Transaction Translator. Here it is demonstrated how the Hub Address and Endpoint Speed fields should be set for directly attached FS/LS devices and hubs:

1. QH (for direct attach FS/LS)—Async. (Bulk/Control Endpoints) Periodic (Interrupt)
   - — Hub Address = 0
   - — Transactions to direct attached device/hub.
     - – QH.EPS = Port Speed
   - — Transactions to a device downstream from direct attached FS hub.
     - – QH.EPS = Downstream Device Speed

Note: When QH.EPS = 01 (LS) and PORTSCx.PSPD = 00 (FS), a LS-pre-pid will be sent before the transmitting LS traffic.

- Maximum Packet Size must be less than or equal 64 or undefined behaviour may result.

2. siTD (for direct attach FS)—Periodic (ISO Endpoint)
   — All FS ISO transactions:
   – Hub Address = 0
   – siTD.EPS = 00 (full speed)
   - Maximum Packet Size must less than or equal to 1023 or undefined behaviour may result.

30.8.4.1.5 Operational Model

The operational models are well defined for the behavior of the Transaction Translator (see USB 2.0 specification Universal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org) and for the EHCI controller moving packets between system memory and a USB-HS hub. Since the embedded Transaction Translator exists within the host controller there is no physical bus between EHCI host controller driver and the USB FS/LS bus. These sections will briefly discuss the operational model for how the EHCI and Transaction Translator operational models are combined without the physical bus between. The following sections assume the reader is familiar with both the EHCI and USB 2.0 Transaction Translator operational models.

Micro- Frame Pipeline

The EHCI operational model uses the concept of H-frames and B-frames to describe the pipeline between the Host (H) and the Bus (B). The embedded Transaction Translator shall use the same pipeline algorithms specified in the USB 2.0 specification for a Hub-based Transaction Translator.

All periodic transfers always begin at B-frame 0 (after SOF) and continue until the stored periodic transfers are complete. As an example of the micro-frame pipeline implemented in the embedded Transaction Translator, all periodic transfers that are tagged in EHCI to execute in H-frame 0 will be ready to execute on the bus in B-frame 0.

It is important to note that when programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded Transaction Translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream Hub-based Transaction Translators.

Once periodic transfers are exhausted, any stored asynchronous transfer will be moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to H-frame and B-frame boundaries with the exception that an asynchronous transfer can not babble through the SOF (start of B-frame 0.)

Split State Machines

The start and complete split operational model differs from EHCI slightly because there is no bus medium between the EHCI controller and the embedded Transaction Translator. Where a start or complete-split operation would occur by requesting the split to the HS hub, the start/complete split operation is simple an

internal operation to the embedded Transaction Translator. Table 30-78 summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 30-78. Summary of the conditons of handshakes

Condition	Emulate TT Response
Start-Split: All asynchronous buffers full.	NAK
Start-Split: All periodic buffers full.	ERR
Start-Split: Success for start of Async. Transaction.	ACK
Start-Split: Start Periodic Transaction.	No Handshake (Ok)
Complete-Split: Failed to find transaction in queue.	Bus Time Out
Complete-Split: Transaction in Queue is Busy.	NYET
Complete-Split: Transaction in Queue is Complete.	[Actual Handshake from LS/FS device]

Note: The un-shaded cells represent Start-Splits and the shaded cells represent Complete-Splits.

Asynchronous Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

USB 2.0—11.17.3

- Sequencing is provided & a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.

USB 2.0—11.17.4

- Transaction tracking for 2 data pipes.

USB 2.0—11.17.5

- Clear_TT_Buffer capability provided though the use of the register.

Periodic Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

USB 2.0—11.18.6.[1-2]

- Abort of pending start-splits
  - — EOF (and not started in micro-frames 6)
  - — Idle for more than 4 micro-frames

- Abort of pending complete-splits
  — EOF
  — Idle for more than 4 micro-frames

USB 2.0—11.18.[7-8]

- Transaction tracking for up to 16 data pipes.
- Some applications may not require transaction tracking up to a maximum of 16 periodic data pipes. The option to limit the tracking to only 4 periodic data pipes exists in the by changing the configuration constant VUSB_HS_TT_PERIODIC_CONTEXTS to 4. The result is a significant gate count savings to the core given the limitations implied.

CAUTION	**Note:** Limiting the number of tracking pipes in the EMBedded –TT to four (4) will impose the restriction that no more than 4 periodic transactions (INTERRUPT/ISOCHRONOUS) can be scheduled through the embedded-tt per frame. the number 16 was chosen in the USB specification because it is sufficient to ensure that the high-speed to full-speed periodic pipeline can remain full. keeping the pipeline full puts no constraint on the number of periodic transactions that can be scheduled in a frame and the only limit becomes the flight time of the packets on the bus.

- Complete-split transaction searching.

Note: There is no data schedule mechanism for these transactions other than the micro-frame pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 ms) or else undefined behavior may result.

Multiple Transaction Translators

The maximum number of embedded Transaction Translators that is currently supported is one as indicated by the N_TT field in the HCSPARAMS—Host Control Structural Parameters register.

30.8.4.2 Device Operation

The co-existence of a device operational controller within the host controller has little effect on EHCI compatibility for host operation except as noted in this section.

30.8.4.3 USBMODE Register

Given that the dual-role controller is initialized in neither host nor device mode, the USBMODE register must be programmed for host operation before the EHCI host controller driver can begin EHCI host operations.

30.8.4.3.1 Non-Zero Fields the Register File.

Some of the reserved fields and reserved addresses in the capability registers and operational register have use in device mode, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields) with the operation registers should always be written to zero. This is an EHCI requirement of the device controller driver that must be adhered to.
- Read operations by the host controller must properly mask EHCI reserved fields (some of which are device fields) because fields that are used exclusive for device are undefined in host mode .

30.8.4.3.2 SOF Interrupt

This SOF Interrupt used for device mode is shared as a free running 125us interrupt for host mode. EHCI does not specify this interrupt but it has been added for convenience and as a potential software time base. See USBSTS and USBINTR registers.

30.8.4.4 Embedded Design Interface

This is an Embedded USB Host Controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

30.8.4.4.1 Frame Adjust Register

Given that the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like is provided by the Frame Adjust register in the PCI configuration registers. Starts of micro-frames are timed precisely to 125 us using the transceiver clock as a reference clock. i.e. 60 Mhz transceiver clock for 8-bit physical interfaces & full-speed serial interfaces or 30 Mhz transceiver clock for 16-bit physical interfaces.

30.8.4.5 Miscellaneous Variations from EHCI

30.8.4.5.1 Programmable Physical Interface Behaviour

This design supports multiple Physical interfaces which can operate in differing modes when the core is configured with software programmable Physical Interface Modes. Software programmability allows the selection of the Physical interface part during the board design phase instead of during the chip design phase. The control bits for selecting the Physical Interface operating mode have been added to the PORTSCx register providing a capability that is not defined by EHCI.

30.8.4.5.2 Discovery

Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the PORTSCx register for a duration of 10ms. Due to the complexity required to support the attachment of devices that are not high speed there are counter already present in the design that can count the 10ms reset pulse to alleviate the

requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a '1' to the reset the device.
- Software shall write a '0' to the reset the device after 10 ms.
  - This step, which is necessary in a standard EHCI design, may be omitted with this implementation. Should the EHCI host controller driver attempt to write a '0' to the reset bit while a reset is in progress the write will simple be ignored and the reset will continue until completion.
- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device in now operational and at this point the port speed has been determined.

Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices. Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator has been added to PORTSC to provide the current operating speed of the port to the host controller driver.
- A 1-bit High Speed indicator has been added to PORTSC to signify that the port is in High-Speed vs. Full/Low Speed—*This information is redundant with the 2-bit Port Speed indicator above.*

30.8.4.5.3 Port Test Mode

Port Test Control mode behaves fully as described in EHCI since the release of revision 3.2.1. In earlier product revisions, the test packet mode was not EHCI compatible. An alternate host controller driver procedure is no longer necessary or supported.

30.8.5 Device Data Structures

This section defines the interface data structures used to communicate control, status, and data between Device Controller Driver (DCD) Software and the Device Controller. The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of device Queue Heads and Transfer Descriptors.

Note: Software must ensure that no interface data structure reachable by the Device Controller spans a 4K-page boundary.

The data structures defined in the chapter are (from the device controller's perspective) a mix of read-only and read/ writeable fields. The device controller must preserve the read-only fields on all data structure writes.

The ARC USB-HS OTG High-Speed USB On-The-Go core includes DCD Software called the USB 2.0 Device API. The Device API provides an easy to use Application Program Interface for developing device (peripheral) applications using the ARC USB-HS OTG High-Speed USB On-The-Go core. The Device API incorporates and abstracts for the application developer all of the elements of the program interface.

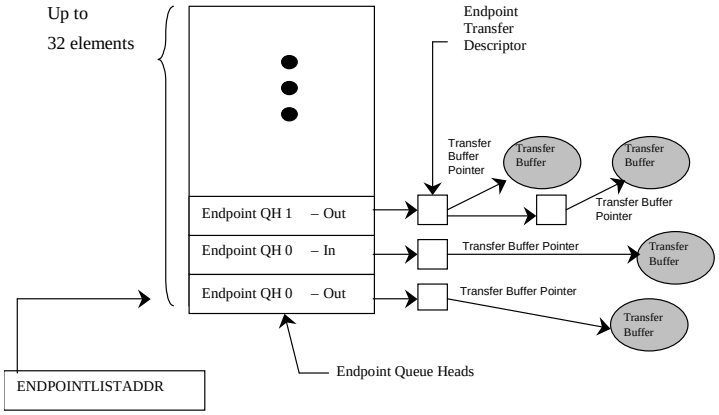


Figure 30-73. End Point Queue Head Organization

Device queue heads are arranged in an array in a continuous area of memory pointed to by the *ENDPOINTLISTADDR* pointer. The even –numbered device queue heads in the list support receive endpoints (OUT/SETUP) and the odd-numbered queue heads in the list are used for transmit endpoints (IN/INTERRUPT). The device controller will index into this array based upon the endpoint number received from the USB bus. All information necessary to respond to transactions for all primed transfers is contained in this list so the Device Controller can readily respond to incoming requests without having to traverse a linked list.

NOTE	The Endpoint Queue Head List must be aligned to a 2k boundary.

30.8.5.1 Endpoint Queue Head (dQH)

The device Endpoint Queue Head (dQH) is where all transfers are managed. The dQH is a 48-byte data structure, but must be aligned on 64-byte boundaries. During priming of an endpoint, the dTD (device transfer descriptor) is copied into the overlay area of the dQH, which starts at the nextTD pointer DWord and continues through the end of the buffer pointers DWords. After a transfer is complete, the dTD status DWord is updated in the dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.

31 30 29	28 27	26 25 24 23 22 21 20 19 18 17 16	15	14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Mult	zl	0	Maximum Packet Length	io	0
Current dTD Pointer				0

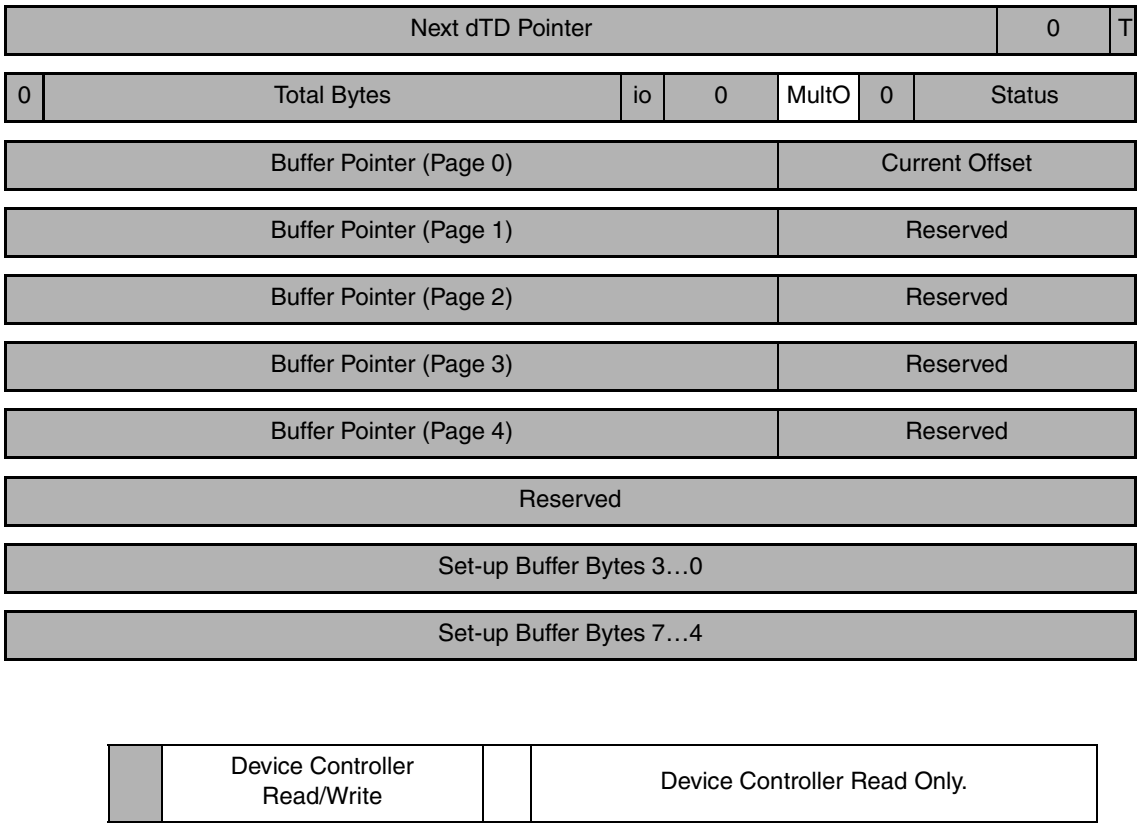


Figure 30-74. Endpoint Queue Head (dQH)

30.8.5.1.1 Endpoint Capabilities/Characteristics

This DWord specifies static information about the endpoint, in other words, this information does not change over the lifetime of the endpoint. Device Controller software should not attempt to modify this information while the corresponding endpoint is enabled.

Table 30-79. Endpoint Capabilities/Characteristics

Bit	Description
31:30	Mult. This field is used to indicate the number of packets executed per transaction description as given by the following: 00—Execute N Transactions as demonstrated by the USB variable length packet protocol where N is computed using the Maximum Packet Length (dQH) and the Total Bytes field (dTD) 01 Execute 1 Transaction. 10 Execute 2 Transactions. 11 Execute 3 Transactions. Note: Non-ISO endpoints must set Mult=”00”. Note: ISO endpoints must set Mult = 01, 10, or 11, as needed.
29	Zero Length Termination Select. This bit is used to indicate when a zero length packet is used to terminate transfers where to total transfer length is a multiple . This bit is not relevant for Isochronous 0—Enable zero length packet to terminate transfers equal to a multiple of the Maximum Packet Length. (default). 1—Disable the zero length packet on transfers that are equal in length to a multiple Maximum Packet Length.
28:27	Reserved. These bit reserved for future use and should be set to zero.
26:16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15	Interrupt On Setup (IOS). This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.
14:0	Reserved. Bits reserved for future use and should be set to zero.

30.8.5.1.2 Transfer Overlay

The seven DWords in the overlay area represent a transaction working space for the device controller. The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it will not read the associated endpoint.

After an endpoint is readied, the dTD will be copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue.

See dTD for a description of the overlay fields.

30.8.5.1.3 Current dTD Pointer

The current dTD pointer is used by the device controller to locate the transfer in progress. This word is for Device Controller (hardware) use only and should not be modified by DCD software.

Table 30-80. Next dTD Pointer

Bit	Description
31:5	Current dTD. This field is a pointer to the dTD that is represented in the transfer overlay area. This field will be modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.
4:0	Reserved. Bit reserved for future use and should be set to zero.

30.8.5.1.4 Set-up Buffer

The set-up buffer is dedicated storage for the 8-byte data that follows a set-up PID.

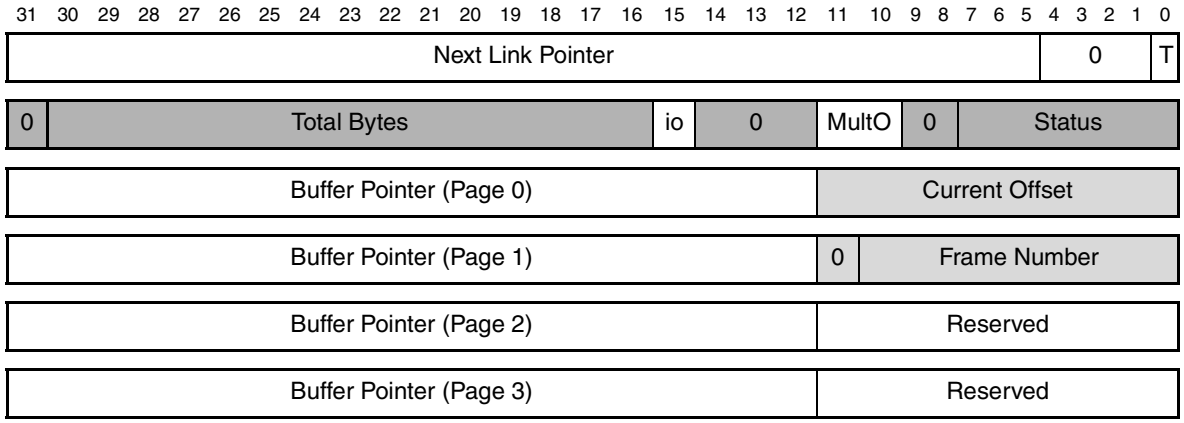
Note: Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets.

Table 30-81. Multiple Mode Control (HCCPARAMS)

DWord	Bits	Description
1	31:0	Setup Buffer 0. This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.
2	31:0	Setup Buffer 1. This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.

30.8.5.2 Endpoint Transfer Descriptor (dTD)

The dTD describes to the device controller the location and quantity of data to be sent/received for given transfer. The DCD should not attempt to modify any field in an active dTD except the Next Like Pointer, which should only be modified as described in section Managing Transfers with Transfer Descriptors.



Buffer Pointer (Page 4)	Reserved

	Device Controller Read/Write		Device Controller Read Only.

Figure 30-75. Endpoint Transfer Descriptor (dTD)

Table 30-82. Next dTD Pointer

Bit	Description
31:5	Next Transfer Element Pointer. This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively.
4:1	Reserved. Bits reserved for future use and should be set to zero.
0	Terminate (T). 1=pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Device Controller that there are no more valid entries in the queue.

Table 30-83. dTD Token

Bit	Description
31	Reserved. Bit reserved for future use and should be set to zero.
30:16	Total Bytes. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and only on the successful completion of the transaction. The maximum value software may store in the field is 5*4K(5000H). This is the maximum number of bytes 5 page pointers can access. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16K(4000H). If the value of the field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the device controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for IN transfers that Total Bytes To Transfer be an even multiple of *Maximum Packet Length*. If software builds such a transfer descriptor for an IN transfer, the last transaction will always be less that *Maximum Packet Length*.
15	Interrupt On Complete (IOC). This bit is used to indicate if USBINT is to be set in response to device controller being finished with this dTD.
14:12	Reserved. Bits reserved for future use and should be set to zero.

Bit	Description
11:10	Multiplier Override (MultO). This field can be used for transmit ISO's (ie. ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit-ISO. Example: if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 0 [default] Three packets are sent: {Data2(8); Data1(7); Data0(0)} if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 2 Two packets are sent: {Data1(8); Data0(7)} For maximal efficiency, software should compute MultO = greatest integer of (Total Bytes / Max. Packet Size) except for the case when Total Bytes = 0; then MultO should be 1. Note: Non-ISO and Non-TX endpoints must set MultO="00".
9:8	Reserved. Bits reserved for future use and should be set to zero.
7:0	Status. This field is used by the Device Controller to communicate individual command execution states back to the Device Controller software. This field contains the status of the last transaction performed on this qTD. The bit encodings are: Bit Status Field Description 7 Active. 6 Halted. 5 Data Buffer Error. 3 Transaction Error. 4,2,0Reserved.

Table 30-84. dTD Buffer Page Pointer List

Bit	Description
31:12	Buffer Pointer. Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
0;11:0	Current Offset. Offset into the 4kb buffer where the packet is to begin.
1;10:0	Frame Number. Written by the device controller to indicate the frame number in which a packet finishes. This is typically be used to correlate relative completion times of packets on an ISO endpoint.

30.8.6 Device Operational Model

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list **transfer descriptors**, pointed to by a **queue head**, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

The ARC USB-HS OTG High-Speed USB On-The-Go is shipped with a DCD called the ARC USB-HS OTG High-Speed USB On-The-Go Device API. The ARC USB-HS OTG High-Speed USB On-The-Go

Device API provides an easy to use application interface for developing USB device (peripheral) applications. The ARC USB-HS OTG High-Speed USB On-The-Go Device API incorporates and abstracts for the application developer all of the information contained in the device operational model. For more information on the ARC USB-HS OTG High-Speed USB On-The-Go Device API, refer to the "Software Design document for the Precise USB 2.0 Device API".

30.8.6.1 Device Controller Initialization

After hardware reset, the device is disabled until the Run/Stop bit is set to a '1'. In the disabled state, the pull-up on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs. Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A Queue head must be prepared so that the device controller can store the incoming setup packet.

In order to initialize a device, the software should perform the following steps:

1. Set Controller Mode in the USBMODE register to device mode.
2. Allocate and Initialize device queue heads in system memory.

> **Note:** Transitioning from host mode to device mode requires a device controller reset before modifying USBMODE.

   — Minimum: Initialize device queue heads 0 Tx & 0 Rx.
   — For information on device queue heads, refer to section Device Data Structures.

> **Note:** All device queue heads must be initialized for control endpoints must be initialized before the endpoint is enabled. Non-Control device queue heads before the endpoint can be used.

3. Configure ENDPOINTLISTADDR Pointer.
   — For additional information on ENDPOINTLISTADDR, refer to the register table.
4. Enable the microprocessor interrupt associated with the ARC USB-HS OTG High-Speed USB On-The-Go core.
   — Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend.
   — For a list of available interrupts refer to the USBINTR and the USBSTS register tables.
5. Set Run/Stop bit to Run Mode.
   — After the Run bit is set, a device reset will occur. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the following Port State and Control section below.

> **Note:** Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register.

It is also not necessary to initially prime Endpoint 0 because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework (Chapter 9) command set.

30.8.6.2 Port State and Control

From a chip or system reset, the device controller enters the *powered* state. A transition from the *powered* state to the *attach* state occurs when the Run/Stop bit is set to a '1'. After receiving a reset on the bus, the port will enter the *defaultFS or defaultHS* state in accordance with the reset protocol described in Appendix C.2 of the USB Specification Rev. 2.0. The following state diagram depicts the state of a USB 2.0 device.

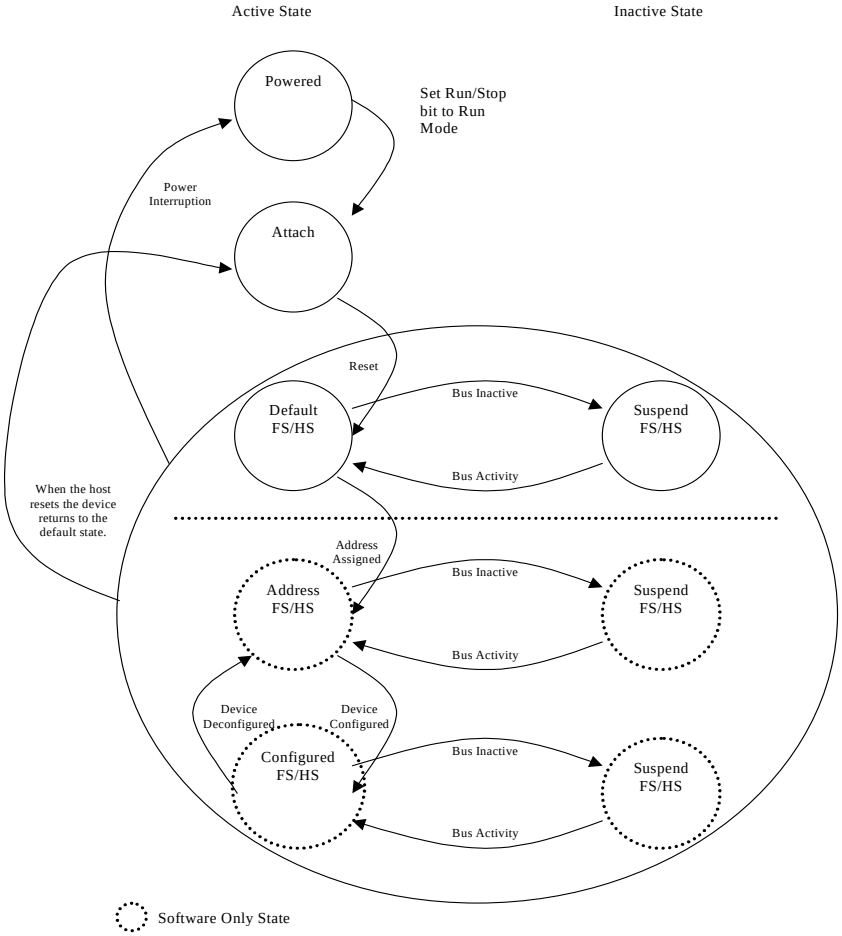


Table 30-85. Device State Diagram

States *powered, attach, defaultFS/HS, suspendFS/HS* are implemented in the device controller and are communicated to the DCD using the following status bits:

Table 30-86. Device Controller State Information Bits

Bit	Register
DCSuspend	<XREF>USBSTS
USB Reset Received	<XREF>USBSTS
Port Change Detect	<XREF>USBSTS
High-Speed Port	<XREF>PORTSC

It is the responsibility of the DCD to maintain a state variable to differentiate between the *DefaultFS/HS* state and the *Address/Configured* states. Change of state from *Default* to *Address* and the *Configured* states is part of the enumeration process described in the device framework section of the USB 2.0 Specification.

As a result of entering the *Address* state, the device address register (DEVICEADDR) must be programmed by the DCD.

Entry into the *Configured* indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the ENDPTCTRLx registers and initializing the associated queue heads.

30.8.6.2.1 Bus Reset

A bus reset is used by the host to initialize downstream devices. When a bus reset is detected, the device controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB Reset Interrupt Enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions will be cancelled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received:

Clear all setup token semaphores by reading the ENDPTSETUPSTAT register and writing the same value back to the ENDPTSETUPSTAT register.

Clear all the endpoint complete status bits by reading the ENDPTCOMPLETE register and writing the same value back to the ENDPTCOMPLETE register.

Cancel all primed status by waiting until all bits in the ENDPTPRIME are 0 and then writing 0xFFFFFFFF to ENDPTFLUSH.

Read the reset bit in the PORTSCx register and make sure that it is still active. A USB reset will occur for a minimum of 3 ms and the DCD must reach this point in the reset cleanup before end of the reset occurs, otherwise a hardware reset of the device controller is recommended (rare.)

- A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD reset. Note: a hardware reset will cause the device to detach from the bus by clearing the Run/Stop bit. Thus, the DCD must completely re-initialize the device controller after a hardware reset.

Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated.

At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port Change Detect is indicated.

After a Port Change Detect, the device has reached the default state and the DCD can read the PORTSCx to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration according to the USB Chapter 9 - Device Framework.

Note: The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device.

In some applications, it may not be possible to enable one or more pipes while in FS mode. *Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.*

30.8.6.2.2 Suspend/Resume

Suspend

Suspend Description

In order to conserve power, USB devices automatically enter the suspended state when the device has observed no bus traffic for a specified period. When suspended, the USB device maintains any internal status, including its address and configuration. Attached devices must be prepared to suspend at any time they are powered, regardless of if they have been assigned a non-default address, are configured, or neither. Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

A USB device exits suspend mode when there is bus activity. A USB device may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If the USB device is capable of remote wakeup signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup signaling must be disabled.

Suspend Operational Model

The device controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the suspend state, the DCD is notified by an interrupt (assuming *DC Suspend Interrupt* is enabled). When the *DCSuspend* bit in the PORTSCx is set to a ‘1’, the device controller is suspended.

DCD response when the device controller is suspended is application specific and may involve switching to low power operation.

Information on the bus power limits in suspend state can be found in USB 2.0 specification.

Note: Review system level clocking issues defined in section (Ref: Signals-Clocking) for the clocking requirements of a suspended device controller.

Resume

If the device controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port. In addition, the device can signal the system to resume operation by forcing resume signaling to the upstream port. Resume signaling is sent upstream by writing a '1' to the Resume bit in the in the PORTSCx while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one more devices) back to the active condition.

Note: Before resume signaling can be used, the host must enable it by using the Set Feature command defined in device framework (chapter 9) of the USB 2.0 Specification.

Port Test Modes

Contact ARC International for port test mode capabilities.

30.8.6.2.3 Managing Endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device. The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a *control* type data channel used for device discovery and enumeration. Other types of endpoints support by USB include *bulk, interrupt*, and *isochronous*. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification.

The ARC USB-HS OTG High-Speed USB On-The-Go device controller hardware supports up to the USB 2.0 maximum of 32 endpoint specified numbers. Each additional endpoint beyond the required endpoint position adds additional hardware logic. The maximum number of endpoint numbers available to the DCD is configured at hardware synthesis timer. After synthesis, the DCD can enable, disable and configure endpoint type up to the maximum selected during synthesis.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1-OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0 is, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a *queue head* allocated in memory. If the maximum of 16 endpoint numbers, one for each endpoint direction are being used by the device controller, then 32 *queue heads* are required. The operation of an endpoint and use of *queue heads* are described later in this document.

30.8.6.2.4 Endpoint Initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the ENDPTCTRLx register. Each 32-bit ENDPTCTRLx is split into an upper and lower half. The lower half of ENDPTCTRLx is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the ENDPTCTRLx register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization.

Table 30-87. Device Controller Endpoint Initialization

Field	Value
Data Toggle Reset	'1'
Data Toggle Inhibit	'0'
Endpoint Type	00 Control 01 Isochronous 10 Bulk 11 Interrupt
Endpoint Stall	'0'

30.8.6.2.5 Stalling

There are two occasions where the device controller may need to return to the host a STALL

The first occasion is the **functional stall**, which is a condition set by the DCD as described in the USB 2.0 device framework. A functional stall is only used on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the ENDPTCTRLx register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD.

A **protocol stall**, unlike a function stall, is used on control endpoints is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the ENDPTCTRLx register can ensure that both stall bits are set at the same instant.

Note: Any write to the ENDPTCTRLx register during operational mode must preserve the endpoint type field (i.e. perform a read-modify-write).

Table 30-88. Device Controller Stall Response Matrix

USB Packet	Endpoint Stall Bit.	Effect on STALL bit.	USB Response
SETUP packet received by a non-control endpoint.	N/A	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	‘1’	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	‘0’	None.	ACK/ NAK/ NYET
SETUP packet received by a control endpoint.	N/A	Cleared	ACK
IN/OUT/PING packet received by a control endpoint	‘1’	None	STALL
IN/OUT/PING packet received by a control endpoint.	‘0’	None.	ACK/ NAK/ NYET

30.8.6.2.6 Data Toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe. For more information on data toggle, refer to the USB 2.0 specification.

Data Toggle Reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a ‘1’ to the data toggle reset bit in the ENDPTCTRLx register. This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

Data Toggle Inhibit

Note: This feature is for test purposes only and should never be used during normal device controller operation.

Setting the *data toggle Inhibit bit* active (‘1’) causes the device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the device controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the host controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

30.8.6.3 Operational Model For Packet Transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification. At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were architected so that the device controller could access main memory or interrupt a host protocol processor in order to respond to the USB 1.1 transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High-speed turnaround time requirements by simply increasing clock rate.

A USB host will send requests to the device controller in an order that can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 3 (transmit direction) is configured as a bulk pipe, then we can expect the host will send IN requests to that endpoint. This device controller is architected in such a way that it can prepare packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as "**priming**" the endpoint. This term will be used throughout the following documentation to describe the device controller operation so the DCD can be architected properly use priming. Further, note that the term "**flushing**" is used to describe the action of clearing a packet that was queued for execution.

Priming Transmit Endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH). After the dTD is fetched, it will be stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time.

After a priming request is complete, an endpoint state of primed is indicated in the ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB.

Since only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus. More information about FIFO sizing is presented in section .

Priming Receive Endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD. At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host.

Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

30.8.6.3.1 Interrupt/Bulk Endpoint Operational Model

The behaviors of the device controller for interrupt and bulk endpoints are identical. All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD will be retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and table on the following page describe how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

N = INT(Number Of Bytes/Max. Packet Length) + 1

With Zero Length Termination (ZLT) = 1

N = MAXINT(Number Of Bytes/Max. Packet Length)

Table 30-89. Variable Length Transfer Protocol Example (ZLT = 0)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	3	256	256	0
512	512	2	512	0	

Table 30-90. Variable Length Transfer Protocol Example (ZLT = 1)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	2	256	256	
512	512	1	512		

Note: The MULT field in the dQH must be set to “00” for bulk, interrupt, and control endpoints.

TX-dTD is complete when:

- All packets described dTD were successfully transmitted. *** Total bytes in dTD will equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. *** Total bytes in dTD will equal zero when this occurs.
- A short packet (number of bytes < maximum packet length) was received. *** This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.
- A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). *** This is an error condition. The device controller will discard the remaining packet, and set the Buffer Error bit in the dTD. In addition, the endpoint will be flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the device controller will flush the endpoint/direction and cease operations for that endpoint/direction.

On the unsuccessful completion of a packet (see long packet above), the dQH will be left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

Note: All packet level errors such as a missing handshake or CRC error will be retried automatically by the device controller.

There is no required interaction with the DCD for handling such errors.

Interrupt/Bulk Endpoint Bus Response Matrix

Table 30-91. Interrupt/Bulk Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	Ignore	Ignore	Ignore	N/A	N/A
In	STALL	NAK	Transmit	BS Error	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK
Ping	STALL	NAK	ACK	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NYET/ACK—NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR—System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

30.8.6.3.2 Control Endpoint Operation Model

Setup Phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller will always accept the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet.

In hardware versions 2.3 and later, the setup lockout mechanism can be disabled and a new tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted be an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

Setup Packet Handling (Pre-2.3 hardware)

- After receiving an interrupt and inspecting USBMODE to determine that a setup packet was received on a particular pipe:
  - 1. Duplicate contents of dQH.SsetupBuffer into local software byte array.
  - 2. Write '1' to clear corresponding ENDPTSETUPSTAT bit and thereby disabling Setup Lockout. (i.e. the Setup Lockout activates as soon as a setup arrives. By writing to the ENDPTSETUPSTAT, the device controller will accept new setup packets.)
  - 3. Process setup packet using local software byte array copy and execute status/handshake phases.
    - Note: After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

Note: To limit the exposure of setup packets to the setup lockout mechanism (if used), the DCD should designate the priority of responding to setup packets above responding to other packet completions.

Setup Packet Handling (2.3 hardware and later)

- Disable Setup Lockout by writing ‘1’ to Setup Lockout Mode (SLOM) in USBMODE. (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

Note: leaving the Setup Lockout Mode As ‘0’ will result in pre-2.3 hardware behavior.

- After receiving an interrupt and inspecting ENDPTSETUPSTAT to determine that a setup packet was received on a particular pipe:
  - 1. Write '1' to clear corresponding bit ENDPTSETUPSTAT.

- 2. Write ‘1’ to Setup Tripwire (SUTW) in USBCMD register.
- 3. Duplicate contents of dQH.SetupBuffer into local software byte array.
- 4. Read Setup TripWire (SUTW) in USBCMD register. (if set - continue; if cleared - goto 2)
- 5. Write '0' to clear Setup Tripwire (SUTW) in USBCMD register.
- 6. Process setup packet using local software byte array copy and execute status/handshake phases.

Note: Note: After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

Data Phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the ENDPTPRIME register is zero and the associated bit in the ENDPTSTAT register is a one. If a prime fails, ie. The ENDPTPRIME bit goes to zero and the ENDPTSTAT bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status (ENDPTSTAT) to enforce data coherency with the setup packet.

Note: The MULT field in the dQH must be set to “00” for bulk, interrupt, and control endpoints.

Note: Error handling of data phase packets is the same as bulk packets described previously.

Status Phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase. The DCD must also perform the same checks of the ENDPTSETUPSTAT as described above in the data phase.

Note: The MULT field in the dQH must be set to “00” for bulk, interrupt, and control endpoints.

Note: Error handling of data phase packets is the same as bulk packets described previously.

Control Endpoint Bus Response Matrix

Shown in the following table is the device controller response to packets on a control endpoint according to the device controller state.

Table 30-92. Control Endpoint Bus Response Matrix

Token Type	Endpoint State					Setup Lockout
	Stall	Not Primed	Primed	Underflow	Overflow	
Setup	ACK	ACK	ACK	N/A	SYSERR	
In	STALL	NAK	Transmit	BS Error	N/A	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK	N/A
Ping	STALL	NAK	ACK	N/A	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NYET/ACK—NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR—System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

30.8.6.3.3 Isochronous Endpoint Operational Model

Isochronous endpoints are used for real-time scheduled delivery of data and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes. Real time delivery by the device controller will is accomplished by the following:

- Exactly MULT Packets per (micro)Frame are transmitted/received. Note: MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.
- NAK responses are not used. Instead, zero length packets and sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro)frame. If the ISO-dTD is still active after that frame, then the ISO-dTD will be held ready until executed or canceled by the DCD.

An EHCI compatible host controller uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro)frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro)frame. Once an ISO transaction is started in a (micro)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the *Transaction Error* bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired
  - — MULT counter reaches zero.
  - — Fulfillment Error [*Transaction Error* bit is set]
    - – # Packets Occurred > 0 AND # Packets Occurred < MULT

Note: For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field in hardware versions 2.3 and later. If the Multiplier Override is zero, the MULT Counter is initialized to the Multiplier in the QH.

- RX Packet Retired:
  - — MULT counter reaches zero.
  - — Non-MDATA Data PID is received**
    - – ** Exit criteria only valid in hardware version 2.3 or later. Previous to hardware version 2.3, any PID sequence that did not match the MULT field exactly would be flagged as a transaction error due to PID mismatch or fulfillment error.
  - — Overflow Error:
    - – Packet received is > maximum packet length. [*Buffer Error* bit is set]

- Packet received exceeds total bytes allocated in dTD. [*Buffer Error* bit is set]

— Fulfillment Error [*Transaction Error* bit is set]

- # Packets Occurred > 0 AND # Packets Occurred < MULT

— CRC Error [*Transaction Error* bit is set

Note: For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (micro)frame to (micro)frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at least two (micro)frames.

Isochronous Pipe Synchronization

When it is necessary to synchronize an isochroous data pipe to the host, the (micro)frame number (FRINDEX register) can be used as a marker. To cause a packet transfer to occur at a specific (micro)frame number [N], the DCD should interrupt on SOF during frame N-1. When the FRINDEX=N–1, the DCD must write the prime bit. The device controller will prime the isochronous endpoint in (micro)frame N–1 so that the device controller will execute delivery during (micro)frame N.

CAUTION	Priming an endpoint towards the end of (micro)frame N-1 will not guarantee delivery in (micro)frame N. The delivery may actually occur in (micro)frame N+1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

Isochronous Endpoint Bus Response Matrix

Table 30-93. Isochronous Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	N/A	N/A
In	NULL Packet	NULL Packet	Transmit	BS Error	N/A
Out	Ignore	Ignore	Receive	N/A	Drop Packet
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NULL Packet = Zero Length Packet

30.8.6.4 Managing Queue Heads

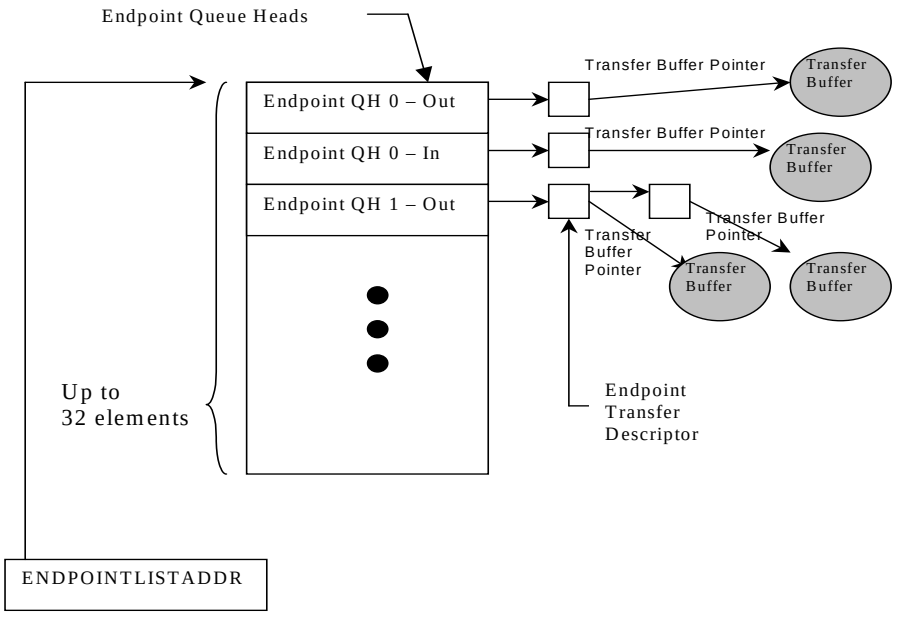


Figure 30-76. End Point Queue Head Diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTD). An area of memory pointed to by ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in Figure 30-76. The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors since pointers will no longer exist within the queue head once the dTD is retired (see section Software Link Pointers).

In addition to the current and next pointers and the dTD overlay examined in section Operational Model For Packet Transfers, the dQH also contains the following parameters for the associated endpoint: Multipler, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

30.8.6.4.1 Queue Head Initialization

One pair of device queue heads must be initialized for each active endpoint. To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required bandwidth an in conjuction with the USB Chapter 9 protocol. *Note: In FS mode, the multiplier field can only be 1 for ISO endpoints.*
- Write the next dTD Terminate bit field to "1".

- Write the Active bit in the status field to “0”.
- Write the Halt bit in the status field to “0”.

Note: The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTD’s.

30.8.6.4.2 Operational Model For Setup Transfers

As discussed in section Control Endpoint Operation Model, setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8-byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH - RX to software buffer.
2. Acknowledge setup backup by writing a “1” to the corresponding bit in ENDPTSETUPSTAT.

Note: The acknowledge must occur before continuing to process the setup packet.

Note: After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH–RX. Only the local software copy should be examined.

3. Check for pending data or status dTD’s from previous control transfers and flush if any exist as discussed in section Flushing/De-priming an Endpoint.
4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

Note: It is possible for the device controller to receive setup packets before previous control transfers complete. Existing control packets in progress must be flushed and the new control packet completed.

30.8.6.5 Managing Transfers with Transfer Descriptors

30.8.6.5.1 Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers to the for the linked list of dTDs for each respective queue head. This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list.

Note: To conserve memory, the reserved fields at the end of the dQH can be used to store the Head & Tail pointers but it still remains the responsibility of the DCD to maintain the pointers.

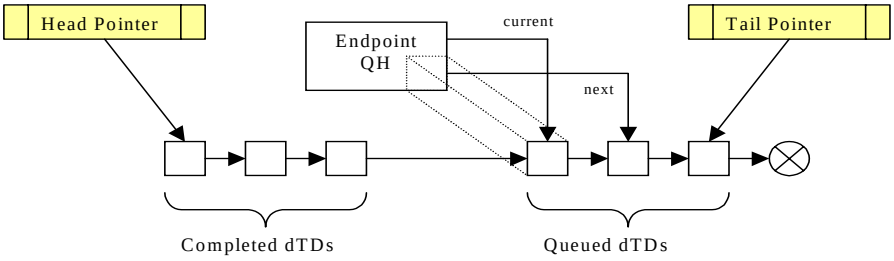


Figure 30-77. Software Link Pointers

30.8.6.5.2 Building a Transfer Descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer. Use the following procedure for building dTDs.

Allocate 8-DWord dTD block of memory aligned to 8-DWord boundaries. Example: bit address 4:0 would be equal to “00000”

Write the following fields:

1. Initialize first 7 DWords to 0.
2. Set the terminate bit to “1”.
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.
5. Initialize the status field with the active bit set to “1” and all remaining status bits set to “0”.
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

30.8.6.5.3 Executing A Transfer Descriptor

To safely add a dTD, the DCD must be follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty:

— Check DCD driver to see if pipe is empty (internal representation of linked-list should indicate if any packets are outstanding)

- Case 1: Link list is empty
  - — 1. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.
  - — 2. Clear active & halt bit in dQH (in case set from a previous error).

— 3. Prime endpoint by writing ‘1’ to correct bit position in ENDPTPRIME.

- Case 2: Link list is not empty
  - — 1. Add dTD to end of linked list.
  - — 2. Read correct prime bit in ENDPTPRIME—if ‘1’ DONE.
  - — 3. Set ATDTW bit in USBCMD register to ‘1’.
  - — 4. Read correct tatus bit in ENDPTSTAT. (store in tmp. variable for later)
  - — 5. Read ATDTW bit in USBCMD register.
  - — If ‘0’ goto 3.
  - — If ‘1’ continue to 6.
  - — 6. Write ATDTW bit in USBCMD register to ‘0’.
  - — 7. If status bit read in (3) is ‘1’ DONE.
  - — 8. If status bit read in (3) is ‘0’ then Goto Case 1: Step 1.

30.8.6.5.4 Transfer Completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

CAUTION	Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0
- Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device Error Matrix.

In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is by decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

30.8.6.5.5 Flushing/De-priming an Endpoint

It is necessary for the DCD to flush to de-prime one more endpoints on a USB device reset or during a broken control transfer. There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a '1' to the corresponding bit(s) in ENDPTFLUSH.
2. Wait until all bits in ENDPTFLUSH are '0'.
   — Software note: this operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.
3. Read ENDPTSTAT to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now '0'. If the corresponding bits are '1' after step #2 has finished, then the flush failed as described in the following:
   — Explanation: In very rare cases, a packet is in progress to the particular endpoint when commanded flush using ENDPTFLUSH. A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush be repeating steps 1-3 until each endpoint is successfully flushed.

30.8.6.5.6 Device Error Matrix

The following table summarizes packet errors that are not automatically handled by the Device Controller.

Table 30-94. Device Error Matrix

Error	Direction	Packet Type	Data Buffer Error Bit	Transaction Error Bit
Overflow **	RX	Any	1	0
ISO Packet Error	RX	ISO	0	1
ISO Fulfillment Error	Both	ISO	0	1

Notice that the device controller handles all errors on Bulk/Control/Interrupt Endpoints except for a data buffer overflow. However, for ISO endpoints, errors packets are not retried and errors are tagged as indicated.

Error Descriptions:

Error	Description
Overflow	Number of bytes received exceeded max. packet size or total buffer length.
	** This error will also set the Halt bit in the dQH and if there are dTDs remaining in the linked list for the endpoint, then those will not be executed.

Error	Description
ISO Packet Error	CRC Error on received ISO packet. Contents not guaranteed to be correct.
ISO Fulfillment Error	Hst failed to complete the number of packets defined in the dQH mult field within the given (micro)frame. For scheduled data delivery the DCD may need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (micro)frame. During the “dead” (micro)frame, the Device Controller reports error on the pipe and primes for the following frame.

30.8.6.6 Servicing Interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

30.8.6.6.1 High-Frequency Interrupts

High frequency interrupts in particular should be handed in the order below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

Table 30-95. High Frequency Interrupt Events

Execution Order	Interrupt	Action
1a	USB Interrupt ** - **ENDPTSETUPSTATUS**	Copy contents of setup buffer and acknowledge setup packet (as indicated in section Managing Queue Heads). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.
1b	USB Interrupt ** - **ENDPTCOMPLETE**	Handle completion of dTD as indicated in section Managing Queue Heads.
2	SOF Interrupt	Action as deemed necessary by application. This interrupt may not have a use in all applications.

** It is likely that multiple interrupts to stack up on any call to the Interrupt Service Routine AND during the Interrupt Service Routine.

30.8.6.6.2 Low-Frequency Interrupts

The low frequency events include the following interrupts. These interrupt can be handled in any order since they don’t occur often in comparison to the high-frequency interrupts.

Table 30-96. Low Frequency Interrupt Events

Interrupt	Action
Port Change	Change software state information.
Sleep Enable (Suspend)	Change software state information. Low power handling as necessary.
Reset Received	Change software state information. Abort pending transfers.

30.8.6.6.3 Error Interrupts

Error interrupts will be least frequent and should be placed last in the interrupt service routine.

Table 30-97. Error Interrupt Events

Interrupt	Action
USB Error Interrupt.	This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD will more aptly handle packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

Book II, Part 5: Timer Peripherals

Introduction

The i.MX27 processor provides a variety of timers for timing and scheduling of software applications, as well as peripherals and audio waveform generation. The timers covered in this part are as follows:

General Purpose Timer (GPT)

The General Purpose Timer (GPT) has a 32 bit up-counter. The timer counter value can be captured in a register using an event on an external pin. The capture trigger can be programmed to be a rising or/and falling edge. The GPT can also generate an event on ipp_do_cmpout pins and an interrupt when the timer reaches a programmed value. It has a 12-bit prescaler providing a programmable clock frequency derived from multiple clock sources. The GPT has one 32 bit up-counter with clock source selection, including external clock, two input capture channels with programmable trigger edge, and three output compare channels with programmable output mode. The GPT can perform a forced compare and can configured to be programmed to be active in low power and debug modes Interrupt generation can be programmed for capture, compare, rollover events and the timers offers both restart or free-run modes of operation.

Pulse-Width Modulation Module (PWM)

While the Pulse-Width Modulation Module (PWM) is a timer/counter, its primary use is for sound and melody generation. The PWM of the i.MX27 uses a 16-bit counter which is optimized to generate sound from stored sample audio images and it can also generate tones. It uses the 16-bit resolution and a 4 × 16 data FIFO to generate sound. The PWM follows IP Bus protocol for interfacing with the ARM9 processor core. It does not have any interface signals with any other module inside the chip except for clock and reset inputs from the Clock and Reset Controller module and interrupt signals to the processor interrupt handler. There is a single output signal going to a pin in the i.MX27.

The PWM module of the i.MX27 offers the following features:

- Pulse-width modulation (PWM) module has the following features:
- 4 x 16 FIFO to minimize interrupt overhead

- 16-bit resolution
- Sound and melody generation
- Secondary Display contrast control

Real Time Clock (RTC)

The Real Time Clock (RTC) module provides a current stamp of seconds, minutes, hours and days. Alarm and Timer functions are also available for programming. RTC support dates from the year 1980 to 2050. The RTC module includes the following features:

- Full clock—days, hours, minutes, seconds
- Minute countdown timer with interrupt
- Programmable daily alarm with interrupt
- Sampling timer with interrupt
- Once-per-day, once-per-hour, once-per-minute, and once-per-second interrupts
- Operation at 32.768 kHz or 32 kHz, or 38.4 kHz (determined by reference clock crystal)

Watchdog Timer (WDOG)

The Watchdog (WDOG) timer module protects the i.MX27 against system failures by providing a method of escaping from unexpected events or programming errors.

Once the WDOG module is activated, it must be serviced by software on a periodic basis. If servicing does not take place, the timer times out. Upon a time-out, the WDOG Timer module either asserts the $\overline{\text{wdog}}$ signal or a system reset signal $\overline{\text{wdog\_rst}}$ depending on software configuration. The WDOG Timer module also generates a system reset via a software write to the Watchdog Control Register (WCR) a detection of a clock monitor event, an external reset, an external JTAG reset signal, or if a power-on-reset has occurred. The $\overline{\text{wdog}}$ signal is asserted via a software write to the WCR, a detection of a clock monitor event, or upon a watchdog time-out.

In case of a Time-out Even, the following actions can be programmed:

- Interrupt to the ARM9
- Internal Reset (can follow the interrupt after a predefined time-out)
- External pin toggle for external devices reset (issued together with the Internal Reset)

The WDOG module can continue/suspend the timer operation in the low power modes (WAIT, DOZE and STOP). For ARM (DOZE and STOP), it emulates these modes.

Chapter 31
General Purpose Timer (GPT)

31.1 Introduction

The i.MX27 device contains six identical 32-bit general-purpose timers (GPT) with programmable prescalers and compare and capture registers. Each timer's counter value can be captured using an external event and can be configured to trigger a capture event on either the leading or trailing edges of an input pulse. Each GPT can also generate an interrupt when the timer reaches a programmed value. Each GPT has an11-bit prescaler providing a programmable clock frequency derived from multiple clock sources. Figure 31-1 illustrates the general-purpose timer block diagram for one of the timers.

The General-Purpose Timers have the following features:

- Programmable sources for the clock input, including external clock
- Input capture capability with programmable trigger edge
- Output compare with programmable mode
- Free-run and restart modes
- Software reset function

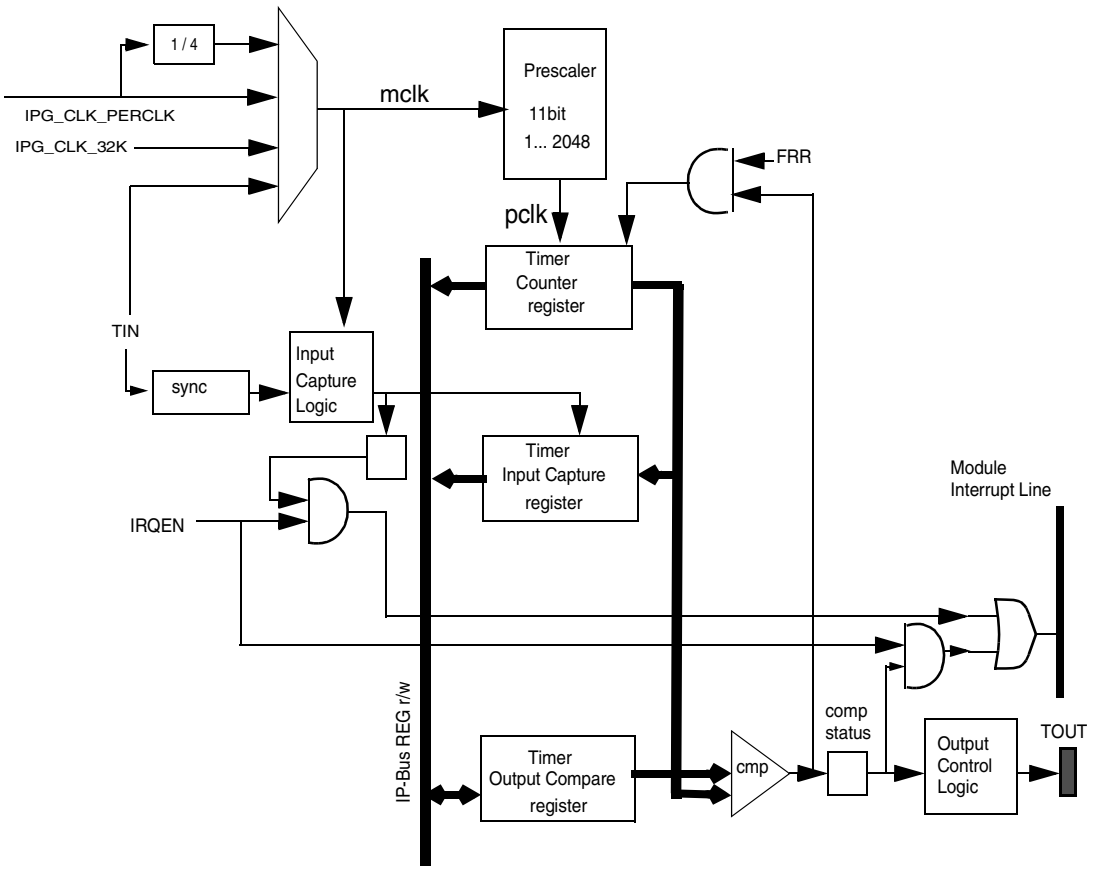


Figure 31-1. General-Purpose Timer Block Diagram

31.2 Operation

After a hardware reset the Counter, Control, Prescaler, Status and Capture registers of the GPT are reset. The Compare value is set to 0xFFFFFFFF. The output pin (TOUT) is also reset. The GPT is enabled when the TEN bit in the GPT Control Register (TCTL) is set and the counter starts running. It is recommended that all the registers be set to appropriate values first before enabling the GPT. When the TEN bit is cleared the counter value freezes or clears depending on CC bit in GPT Control Register (TCTL). All other register values are retained.

There is a Software Reset bit (SWR) in the TCTL register. When this bit is set the GPT generates a reset signal for 3 IPG_CLK periods and this bit clears after 5 IPG_CLK periods. The software reset results in all the registers in the GPT being reset except for the TEN bit which is not cleared by software reset if set. The software reset can be asserted with the TEN bit off, however, the IPG_CLK clock (module clock) signals to GPT must be on for the software reset to be functional.

31.2.1 Clocks

The clock that feeds the prescaler can be selected from the following:

- ipg_clk_perclk (divided by 1 or by 4). The frequency of ipg_clk_perclk is constant and is independent of changes to the IPG_CLK. The module internally synchronizes PERCLK to the IPG_CLK. This clock is synchronized inside the module to IPG_CLK. Hence the frequency of this clock has to be at least 1/4 that of IPG_CLK if the prescaler is programmed to divide by 1.
- GPT I/O pin (IPP_GPT_TIN). External clock from outside the chip.
- 32 kHz clock (IPG_CLK_32K). This is the 32 kHZ low reference clock which is provided by CRM. This clock is supposed to be on in low power mode when the ipg_clk is turned off. Thus the GPT can be run on this clock in low power mode. The CRM is expected to provide this clock after synchronizing it to ahb_clk in normal functional mode and switch to the unsynchronized version in the low power mode.

The clock input source is determined by the CLKSOURCE field of the GPT control register. The CLKSOURCE value should only be changed when the GPT is disabled. The external clock coming from the IPP_GPT_TIN pin is not synchronized with IPG_CLK.

The GPT prescaler register (TPRER) selects the divide ratio of the input clock that drives the main counter (TCN). The prescaler can divide the input clock by a value between 1 and 2048.

31.2.2 Operation During Low-Power Mode

In low-power mode when the IPG_CLK (module clock used for register accesses) and IPG_CLK_PERCLK clocks are switched off, the GPT can also work only if the ipg_clk_32k is available. The counter continues to run, when it reaches the value contained in the compare register, the compare interrupt will occur. The capture interrupt can also be produced in low-power mode if the capture function is enable. When the capture interrupt or compare interrupt occurs, the status register can be written on the positive edge of mclk.

31.2.3 Capture Event

Each GPT have a 32-bit capture register that takes a snapshot of the counter when a defined transition of TIN is detected by the capture edge detector. The type of transition that triggers this capture is selected by the CAP field of the GPT Control Register (TCTL). Because the capture event is triggered by mclk, so each transition must be valid for at least two mclk periods to ensure a capture event is triggered.

When a capture event occurs, the corresponding status bit is set in the GPT status register (TSTAT) and an interrupt is posted if the capture function is enabled and if the CAPTEN bit of the GPT control register is set. If another capture event occurs the new count value will still be captured in the capture register even if the interrupt is not serviced and capture status bit is set.

31.2.4 Compare Event

Each GPT have a 32-bit compare register. When the value in this register matches with the value of the counter register a compare event occurs. On a compare event the appropriate GPT output pin (TOUT) is

toggled or an active low pulse (for one count period) is generated on it according to the setting of Output Mode (OM) bit in the GPT control register. When in toggle mode the toggling takes place at the end of the count period in which the match has occurred.

The corresponding status bit is set in the status register and an interrupt is posted if the COMPEN bit of the GPT control register is set. The GPT output pin continues to produce an output on a compare event even if the interrupt is not serviced and compare status bit is set.

31.2.5 Modes of Operation

The GPT can be configured for free-run or restart modes by programming the Free-run/Restart bit (FRR) of the GPT control register.

- Restart mode: In restart mode, when a compare event occurs, the counter resets to 0x00000000. Subsequently it resumes counting up.
- Freerun mode: In free-run mode, when a compare event occurs, it has no effect on the counter value. The counter continues counting until 0xFFFFFFFF is reached and then it is reset to 0x00000000 and resumes counting.

31.3 Programming Model

The General-Purpose Timer modules each have six user-accessible 32-bit registers. They are shown with offsets from their respective base addresses in the detailed register descriptions.

Table 31-1 summarizes the general-purpose timer registers and their addresses. Figure 31-2 provides the detailed register summary.

Table 31-1. GPT Register Summary

Description	Name	Address
GPT Control Register 1	TCTL1	0x10003000
GPT Control Register 2	TCTL2	0x10004000
GPT Control Register 3	TCTL3	0x10005000
GPT Control Register 4	TCTL4	0x10019000
GPT Control Register 5	TCTL5	0x1001a000
GPT Control Register 6	TCTL6	0x1001f000
GPT Prescaler Register 1	TPRER1	0x10003004
GPT Prescaler Register 2	TPRER2	0x10004004
GPT Prescaler Register 3	TPRER3	0x10005004
GPT Prescaler Register 4	TPRER4	0x10019004
GPT Prescaler Register 5	TPRER5	0x1001a004
GPT Prescaler Register 6	TPRER6	0x1001f004
GPT Compare Register 1	TCMP1	0x10003008

Table 31-1. GPT Register Summary (continued)

Description	Name	Address
GPT Compare Register 2	TCMP2	0x10004008
GPT Compare Register 3	TCMP3	0x10005008
GPT Compare Register 4	TCMP4	0x10019008
GPT Compare Register 5	TCMP5	0x1001a008
GPT Compare Register 6	TCMP6	0x1001f008
GPT Capture Register 1	TCR1	0x1000300C
GPT Capture Register 2	TCR2	0x1000400C
GPT Capture Register 3	TCR3	0x1000500C
GPT Capture Register 4	TCR4	0x1001900C
GPT Capture Register 5	TCR5	0x1001a00C
GPT Capture Register 6	TCR6	0x1001f00C
GPT Counter Register 1	TCN1	0x10003010
GPT Counter Register 2	TCN2	0x10004010
GPT Counter Register 3	TCN3	0x10005010
GPT Counter Register 4	TCN4	0x10019010
GPT Counter Register 5	TCN5	0x1001a010
GPT Counter Register 6	TCN6	0x1001f010
GPT Status Register 1	TSTAT1	0x10003014
GPT Status Register 2	TSTAT2	0x10004014
GPT Status Register 3	TSTAT3	0x10005014
GPT Status Register 4	TSTAT4	0x10019014
GPT Status Register 5	TSTAT5	0x1001a014
GPT Status Register 6	TSTAT6	0x1001f014

Table 31-2. General Purpose Timer Register Summary

Name		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
0x1000_3000 (TCTL1)– 0x1000_F000 (TCTL6)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	CC	OM	FRR	CAP		CAP T EN	COM P EN	CLK SOURCE			TEN
	W	SWR															

Table 31-2. General Purpose Timer Register Summary (continued)

Name		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
0x1000_3004 (TPRER1)–0x1000_F004 (TPRER6)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	PRESCALER										
	W																
0x1000_3008 (TCMP1)–0x1000_F008 (TCMP6)	R	COMPARE VALUE															
	W																
	R	COMPARE VALUE															
	W																
0x1000_300C (TCR1)–0x1000_F00C (TCR6)	R	CAPTURE VALUE															
	W																
	R	CAPTURE VALUE															
	W																
0x1000_3010 (TCN1)–0x1000_F010 (TCN6)	R	COUNTER VALUE															
	W																
	R	COUNTER VALUE															
	W																
0x1000_3014 (TSTAT1)–0x1000_F014 (TSTAT6)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CAPT	COMP
	W															W1C	W1C

31.3.1 GPT Control Registers

The GPT control (TCTL) register controls the overall operation of the timer. Figure 31-2 shows field assignments for this register and Table 31-3 provides the field descriptions.

0x1000_3000 (TCTL1)
0x1000_4000 (TCTL2)
0x1000_5000 (TCTL3)
0x1000_9000 (TCTL4)
0x1000_A000 (TCTL5)
0x1000_F000 (TCTL6)

Access: User read-write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SWR	0	0	0	0	CC	OM	FRR	CAP		CAPT EN	COM P EN	CLK SOURCE			TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-2. GPT Control Register

Table 31-3. GPT Control Registers Field Descriptions

Name	Description
31–16	Reserved.
SWR 15	SOFTWARE RESET. GPT is reset when this bit is set to 1. This bit is set when the module is in reset state and is cleared when the reset procedure is over. The reset signal is asserted 2 clock cycles (IPG_CLK) after this bit is set and the reset remains asserted for 3 clock cycles. The whole reset procedure is complete 5 clock cycles after this bit is asserted This software reset does not reset the TEN bit. 0 GPT is not reset. 1 GPT is reset.
14–11	Reserved. These bits are reserved and should not be used.
10 CC	Counter Clear. This bit determines whether the counter is to be cleared when TEN=0 (timer disabled). 0 Counter is halted at the current count when TEN = 0. 1 Counter will be reset when TEN=0.
9 OM	Output Mode. This bit controls the output mode of the timer after compare event occurs. 0 Active-low pulse for one clock period. 1 Toggle output.
8 FRR	Free-Run/Restart. This bit controls how the timer operates after a compare event occurs. In free-run mode, the timer continues counting till 0xffffffff. In restart mode, the counter resets to 0x00000000 and resumes counting. 0 Restart mode 1 Free-run mode

Table 31-3. GPT Control Registers Field Descriptions (continued)

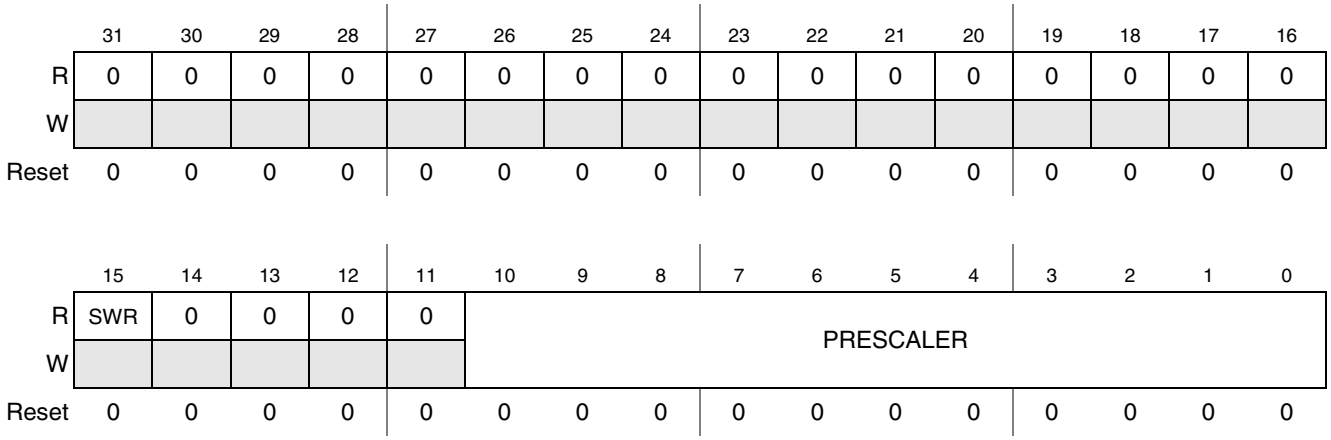
Name	Description
7–6 CAP	Capture Edge. This field controls the operation of the capture function. The value in the counter is loaded into the Capture register on the detection of an event on the TIN pin. The event which will trigger this capture is determined by this field. 00 Capture function disabled 01 Capture on rising edge and generate interrupt 10 Capture on falling edge and generate interrupt 11 Capture on rising or falling edge and generate interrupt
5 CAPT EN	Capture Interrupt Enable. This bit enables the capture interrupt. 0 Capture interrupt is disabled. 1 Capture interrupt is enabled.
4 COMP EN	Compare Interrupt Enable. This bit enables the compares interrupt. 0 Compare interrupt disabled. 1 Compare interrupt enabled.
3–1 CLKSOURCE	Clock Source. This field controls the source of the clock to the prescaler. The stop-count freezes the timer at its current value. **Note:** This field value should only be changed when the GPT is disabled. 000 Stop count (clock disabled). 001 PERCLK1 to prescaler. 010 PERCLK1 divided by 4 to prescaler. 011 TIN to prescaler. 1xx 32 kHz clock to prescaler.
0 TEN	Timer Enable. TEN bit enables the general-purpose timer. The bit can be cleared either by writing 0 or by a hardware reset. The bit cannot be cleared by asserting the software reset. 0 Timer is disabled 1 Timer is enabled.

31.3.2 GPT Prescaler Register

The GPT prescaler register (TPRER) controls the divide value of the prescaler. Figure 31-3 shows field assignments for this register and Table 31-4 provides the field descriptions.

0x1000_3004 (TPRER1)
0x1000_4004 (TPRER2)
0x1000_5004 (TPRER3)
0x1000_9004 (TPRER4)
0x1000_A004 (TPRER5)
0x1000_F004 (TPRER6)

Access: User read-write



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

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

Figure 31-3. GPT Control Register

Table 31-4. GPT Prescaler Register Description

Name	Description
31–11	Reserved.
10–0 PRESCALER	Counter Clock Prescaler. This field determines the division value of the prescaler between 1 and 2048. 0x00 divides by 1 and 0x7FF divides by 2048.Note: When ipg_clk_perclk freq.= IPG_CLK FREQ. the minimum value to be programmed in PRESCALER field should be > 2 to get proper functioning of the module. 0x00 = Divide by 1 ... 0x7ff = Divide by 2048

31.3.3 GPT Compare Register

The GPT compare (TCMP) register contains the value that is compared with the free-running counter. A compare event is generated when the counter matches the value in this register. This register reset to 0xFFFFFFFF. Figure 31-4 shows field assignments for this register and Table 31-5 provides the field descriptions.

0x1000_3008 (TCMP1)
0x1000_4008 (TCMP2)
0x1000_5008 (TCMP3)
0x1000_9008 (TCMP4)
0x1000_A008 (TCMP5)
0x1000_F008 (TCMP6)

Access: User read-write

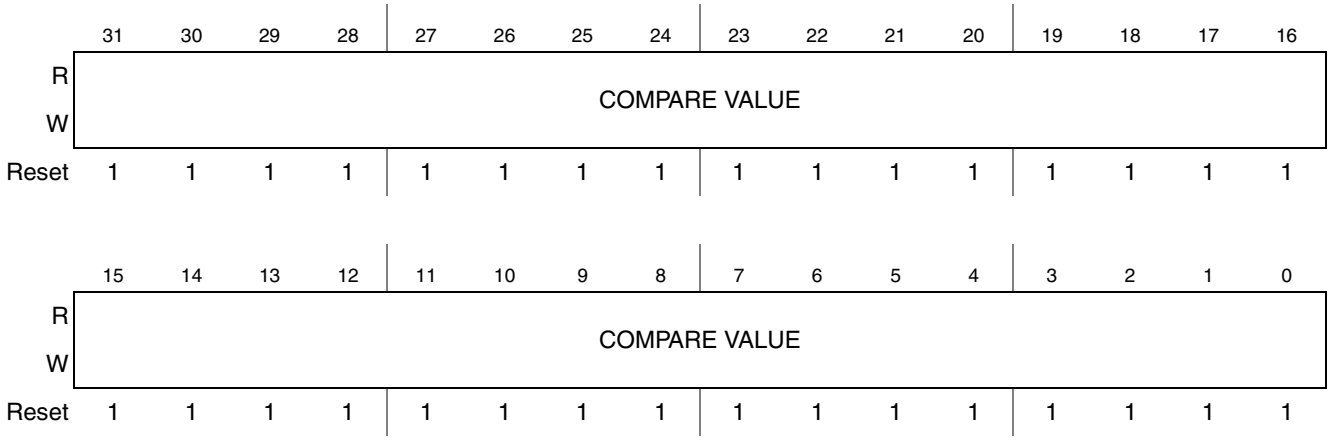


Figure 31-4. GPT Compare Register

Table 31-5. GPT Compare Registers Descriptions

Name	Description
31–0 COMPARE	Compare Value. A compare event occurs when the counter value matches the value in this field.

31.3.4 GPT Capture Register

This register is read only, and resets to 0x00000000. The GPT capture register (TCR) stores the counter value when a capture event occurs. Figure 31-5 shows field assignments for this register and Table 31-6 provides the field descriptions.

0x1000_300C (TCR1)
0x1000_400C (TCR2)
0x1000_500C (TCR3)
0x1000_900C (TCR4)
0x1000_A00C (TCR5)
0x1000_F00C (TCR6)

Access: User read-write

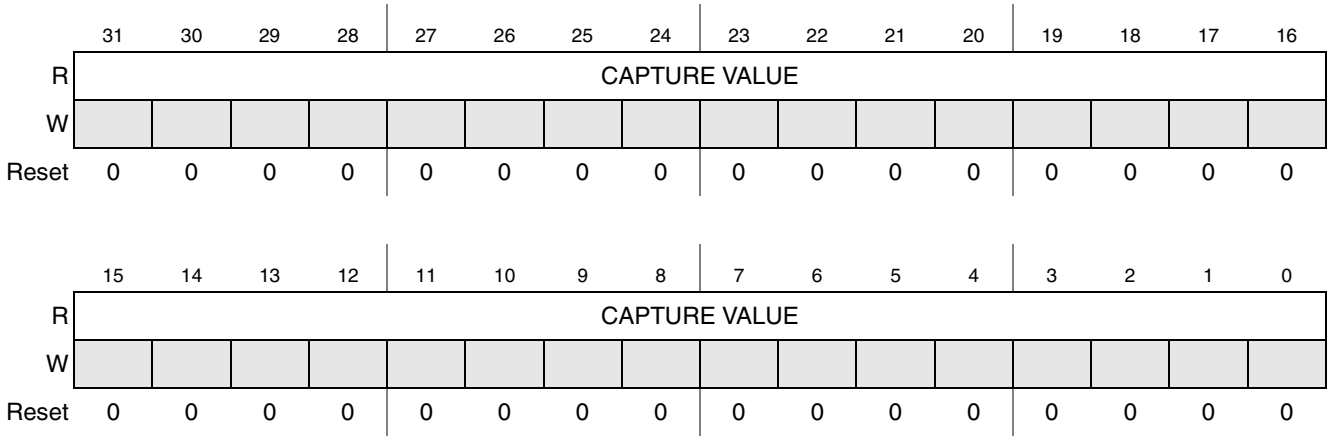


Figure 31-5. GPT Capture Register

Table 31-6. GPT Capture Registers Description

Name	Description
31–0 CAPTURE	Capture Value. This field stores the counter value at the time of a capture event.

31.3.5 GPT Counter Register

The read-only GPT counter (TCN) register can be read at anytime without disturbing the current count. Figure 31-6 shows field assignments for this register and Table 31-7 provides the field descriptions.

0x1000_3010 (TCN1)
0x1000_4010 (TCN2)
0x1000_5010 (TCN3)
0x1000_9010 (TCN4)
0x1000_A010 (TCN5)
0x1000_F010 (TCN6)

Access: User read-write

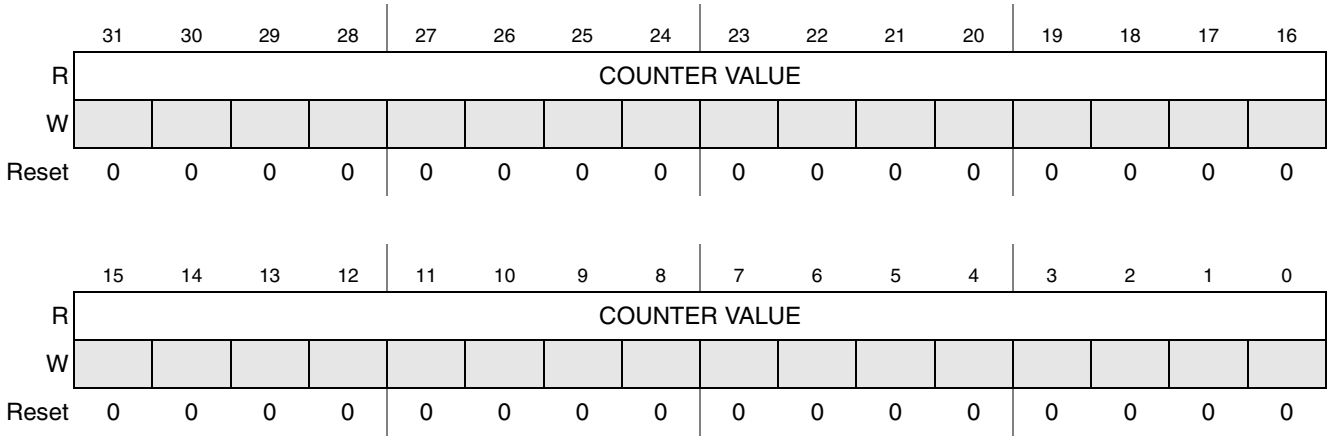


Figure 31-6. GPT Counter Register

Table 31-7. GPT Counter Register Field Descriptions

Name	Description
31–0 COUNT	Count Value. This field contains the current count value. Whenever there is an update of compare register the counter is reset to zero and the count starts afresh.

31.3.6 GPT Status Register

The GPT status (TSTAT) register (write 1 to clear) indicates the GPT's status. When a capture event occurs, the CAPT bit is set. When a compare event occurs, the COMP bit is set. These bits can only be cleared by writing 1 to clear the interrupt status. Figure 31-6 shows field assignments for this register and Table 31-7 provides the field descriptions.

0x1000_3014 (TSTAT1)
0x1000_4014 (TSTAT2)
0x1000_5014 (TSTAT3)
0x1000_9014 (TSTAT4)
0x1000_A014 (TSTAT5)
0x1000_F014 (TSTAT6)

Access: User read-write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CAPT	COMP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	w1c	w1c

Figure 31-7. GPT Status Register

Table 31-8. GPT Status Register Field Descriptions

Name	Description
31–2	Reserved.
1 CAPT	Capture Event. This bit indicates that a capture event has occurred. 0 No capture event 1 A capture event has occurred
0 COMP	Compare Event. This bit indicates that a compare event has occurred. 0 No compare event 1 A compare event has occurred

Chapter 32
Pulse-Width Modulator (PWM)

The Pulse-Width Modulator (PWM) has a 16-bit counter and is optimized to generate sound from stored sample audio images and it can also generate tones. It uses 16-bit resolution and a 4 x 16 data FIFO to generate sound.

32.1 Overview

This section presents an overview of the PWM. Figure 32-1 illustrates the PWM block diagram.

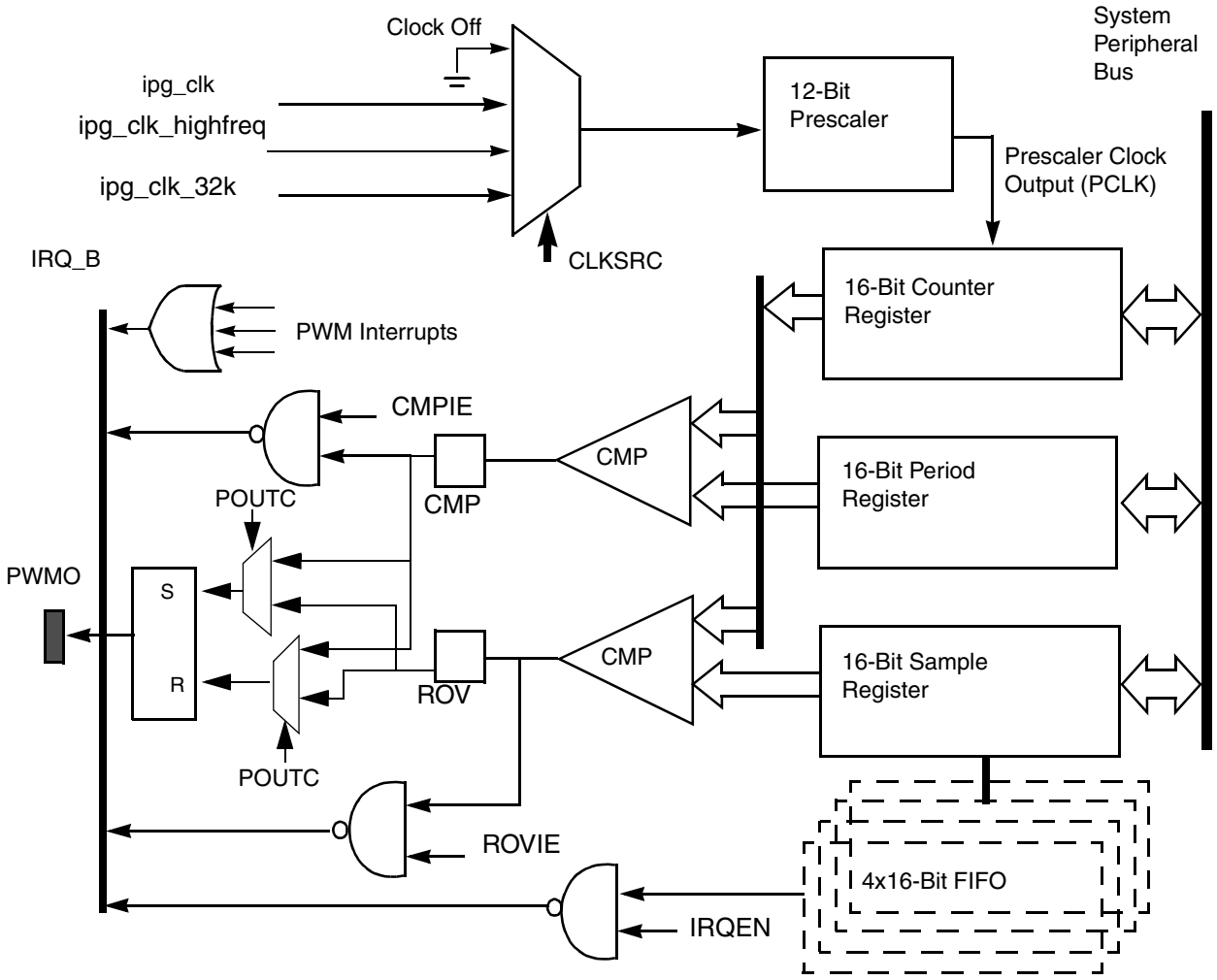


Figure 32-1. Pulse-Width Modulator Block Diagram

The following features characterize the PWM:

- 16-bit up-counter with clock source selection
- 4 x 16 FIFO to minimize interrupt overhead
- 12-bit prescaler for division of clock
- Sound and melody generation
- Active high or active low configurable output
- Can be programmed to be active in low power and debug modes
- Interrupts at compare and rollover

32.2 Signal Description

The PWM follows IP bus protocol for interfacing with the processor core. It does not have any interface signals with any other module inside the chip except for clock and reset inputs from the Clock Control module (CCM) and interrupt signals to the processor interrupt handler. There is a single output signal going outside the chip boundary.

Figure 32-2 shows the PWM signals at the module boundary.

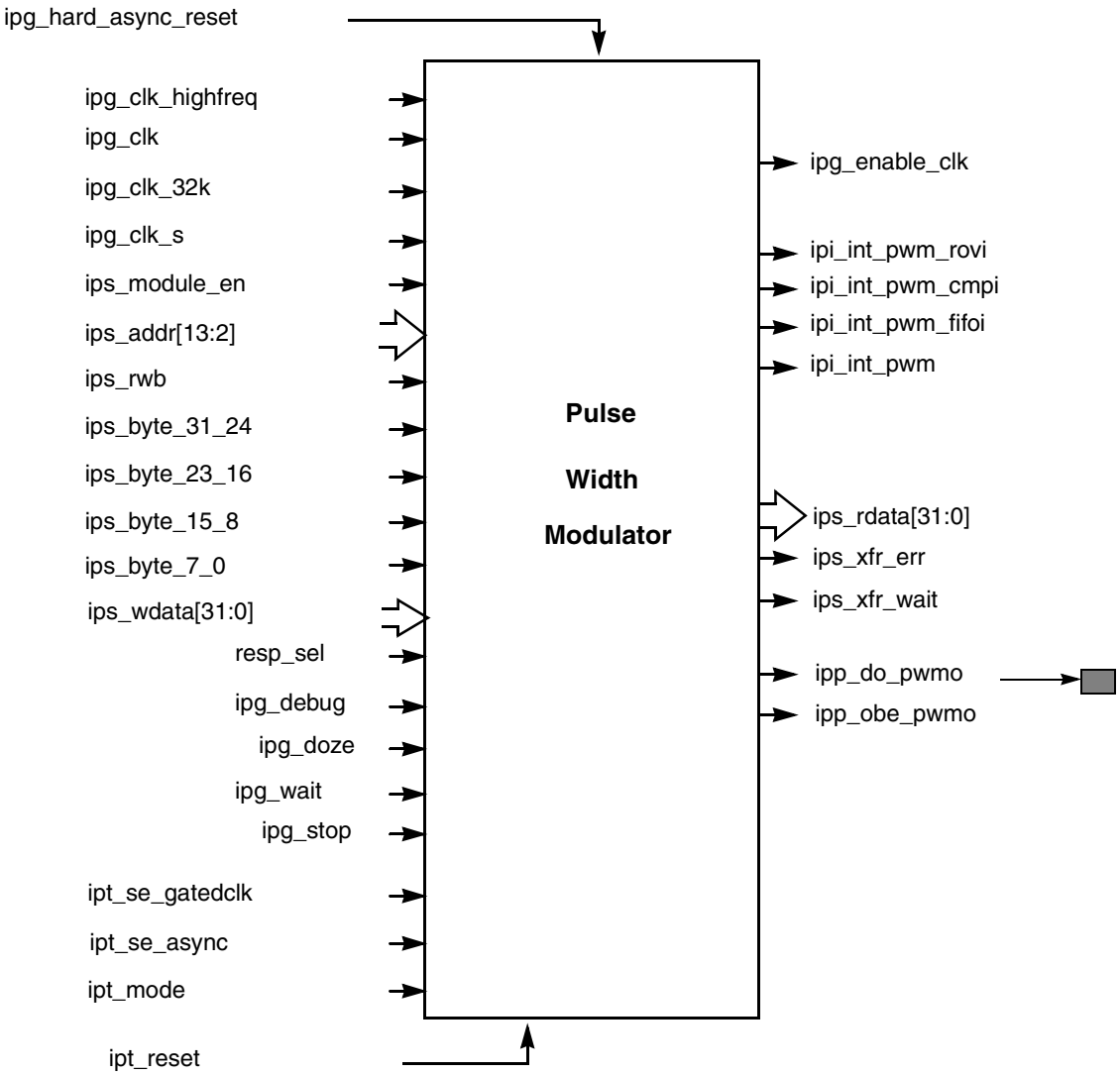


Figure 32-2. PWM Module Signals

32.2.1 External Signals

PWM has a single output signal to the i.MX27 chip boundary named PWMO.

Table 32-1 describes the external signals for the i.MX27 device.

Table 32-1. i.MX27 External Signals

Name	Direction	Function	Reset State	Pull-Up
PWMO	Output	This is the functional output of the PWM. The modulated signal of the module is observed at this pin	0	—

32.2.1.1 PWMO Signal

The PWMO is the modulated output signal of the PWM. This signal can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the PWM. The smallest period can be two ipg_clk clock periods with duty cycle of 50%.

32.3 Memory Map and Register Definition

The PWM module includes 6 user-accessible 32-bit registers. Section 32.3.2, "Register Descriptions" provides the detailed descriptions for all of the PWM registers.

Table 32-2. PWM Memory Map

Address	Register	Access	Reset Value	Section/Page
0x1000_6000 (PWMCR)	PWM Control Register (PWMCR)	R/W	0x0000_0000	32.3.2.1/32-6
0x1000_6004 (PWMSR)	PWM Status Register (PWMSR)	R/W	0x0000_0008	32.3.2.2/32-8
0x1000_6008 (PWMIR)	PWM Interrupt Register (PWMIR)	R/W	0x0000_0000	32.3.2.3/32-9
0x1000_600C (PWMSAR)	PWM Sample Register (PWMSAR)	R/W	0x0000_0000	32.3.2.4/32-10
0x1000_6010 (PWMPR)	PWM Period Register (PWMPR)	R/W	0x0000_FFFE	32.3.2.5/32-11
0x1000_6014 (PWMCNR)	PWM Counter Register (PWMCNR)	R	0x0000_0000	32.3.2.6/32-11

32.3.1 Register Summary

Figure 32-3 shows the key to the register fields, and Table 32-3 shows the register figure conventions.

Figure 32-3. Key to Register Fields

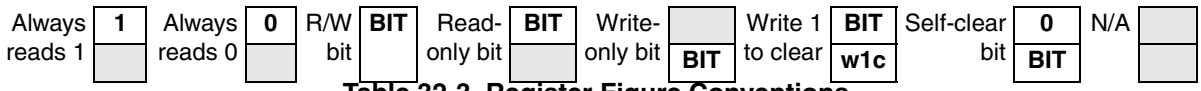


Table 32-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	

Table 32-3. Register Figure Conventions (continued)

Convention	Description
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 32-4 shows the PWM register summary.

Table 32-4. PWM Register Summary

Name		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
0x1000_6000 (PWMCR)	R	0	0	0	0	FWM		STOPEN	DOZEN	WAITEN	DBGEN	BCTR	HCTR	POUTC		CLKSRC	
	W																
	R	PRESCALER												SWR	REPEAT		EN
	W																
0x1000_6004 (PWMSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	FWE	CMP	ROV	FE	FIFOAV		
	W										w1c	w1c	w1c	w1c			
0x1000_6008 (PWMIR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CIE	RIE	FIE
	W																
0x1000_600C (PWMSAR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SAMPLE[15:0]															
	W																
0x1000_6010 (PWMPR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	PERIOD[15:0]															
	W																

Table 32-4. PWM Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x1000_6014 (PWMCNR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	COUNT[15:0]															
	W																

32.3.2 Register Descriptions

This section contains the detailed register descriptions for the PWM registers.

32.3.2.1 PWM Control Register (PWMCR)

The PWM control register (PWMCR) is used to configure the operating settings of the PWM. It contains the prescaler for the clock division. Figure 32-4 shows the register; Table 32-5 provides its field descriptions.

0x1000_6000 (PWMCR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	FWM		STOP EN	DOZ EN	WAIT EN	DBG EN	BCT R	HCT R	POUTC		CLKSRC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRESCALER												SWR	REPEAT		EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-4. PWM Control Register (PWMCR)

Table 32-5. PWMCR Field Descriptions

Field	Description
31–28 Reserved	Reserved. These reserved bits are always read as zero.
27–26	FIFO Water Mark. These bits are used to set the data level at which the FIFO empty flag will be set and the corresponding interrupt generated 00 FIFO empty flag is set when there are more than or equal to 1 empty slots in FIFO. 01 FIFO empty flag is set when there are more than or equal to 2 empty slots in FIFO. 10 FIFO empty flag is set when there are more than or equal to 3 empty slots in FIFO. 11 FIFO empty flag is set when there are more than or equal to 4 empty slots in FIFO.

Table 32-5. PWMCR Field Descriptions (continued)

Field	Description
25 STOPEN	Sleep Mode Enable. This bit keeps the PWM functional while in Sleep Mode. When this bit is cleared, the input clock is gated off in Sleep Mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in Sleep Mode 1 Active in Sleep Mode
24 DOZEN	Doze Mode Enable. This bit keeps the PWM functional in doze mode. When this bit is cleared, the input clock is gated off in doze mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in doze mode 1 Active in doze mode
23 WAITEN	Wait Mode Enable. This bit keeps the PWM functional in wait mode. When this bit is cleared, the input clock is gated off in wait mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in wait mode 1 Active in wait mode
22 DBGEN	Debug Mode Enable. This bit keeps the PWM functional in debug mode. When this bit is cleared, the input clock is gated off in debug mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in debug mode 1 Active in debug mode
21 BCTR	Byte Data Swap Control. This bit determines the byte ordering of the 16-bit data when it goes into the FIFO from the sample register. 0 Byte ordering remains the same. 1 Byte ordering is reversed.
20 HCTR	Halfword Data Swap Control. This bit determines which halfword data from the 32-bit IP-Bus interface is written into the lower 16 bits of the sample register. 0 Halfword swapping does not take place. 1 Halfwords from write data bus are swapped.
19–18 POUTC	PWM Output Configuration. This bit field determines the mode of PWM output on the output pin. 00 Output pin is set at rollover and cleared at comparison 01 Output pin is cleared at rollover and set at comparison 10 PWM output is disconnected. 11 PWM output is disconnected.
17–16 CLKSRC	Select Clock Source. These bits determine which clock input will be selected for running the counter. After reset the system functional clock is selected. The input clock can also be turned off if these bits are set to 00. This field value should only be changed when the PWM is disabled 00 Clock is off 01 ipg_clk 10 ipg_clk_highfreq 11 ipg_clk_32k
15–4 PRESCALER	Counter Clock Prescaler Value. This bit field determines the value by which the clock will be divided before it goes to the counter. 0x000 Divide by 1 0x001 Divide by 2 ... 0xFFF Divide by 4096

Table 32-5. PWMCR Field Descriptions (continued)

Field	Description
3 SWR	Software Reset. PWM is reset when this bit is set to 1. It is a self clearing bit. A write 1 to this bit is a single wait state write cycle. When the module is in reset state this bit is set and is cleared when the reset procedure is over. Setting this bit resets all the registers to their reset values except for the STOPEN, DOZEN, WAITEN, and DBGEN bits in this control register. 0 PWM is out of reset. 1 PWM is undergoing reset.
2–1 REPEAT	Sample Repeat. This bit field determines the number of times each sample from the FIFO is to be used. 00 Use each sample once 01 Use each sample twice 10 Use each sample four times 11 Use each sample eight times
0 EN	PWM Enable. This bit enables the PWM. If this bit is not enabled, the clock prescaler and the counter is reset. When the PWM is enabled, it begins a new period, the output pin is set to start a new period while the prescaler and counter are released and counting begins. 0 PWM is disabled. 1 PWM is enabled.

32.3.2.2 PWM Status Register (PWMSR)

The PWM status register (PWMSR) contains seven bits which display the state of the FIFO and the occurrence of rollover and compare events. The FIFOAV bit is read-only but the other four bits can be cleared by writing 1 to them. FE, ROV, and CMP bits are associated to FIFO-Empty, Roll-over, and Compare interrupts, respectively. Figure 32-5 shows the register; Table 32-6 provides its field descriptions.

0x1000_6004 (PWMSR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	FWE	CMP	ROV	FE	FIFOAV		
W										w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 32-5. PWM Status Register (PWMSR)

Table 32-6. PWMSR Field Descriptions

Field	Description
31–7 Reserved	Reserved. These reserved bits are always read as zero.
6 FWE	FIFO Write Error Status. This bit shows that an attempt has been made to write FIFO when it is full. 0 FIFO write error has not occurred. 1 FIFO write error has occurred.
5 CMP	Compare Status. This bit shows that a compare event has occurred. 0 Compare event has not occurred. 1 Compare event has occurred.
4 ROV	Roll-over Status. This bit shows that a roll-over event has occurred. 0 Roll-over event has not occurred. 1 Roll-over event has occurred.
3 FE	FIFO Empty Status Bit. This bit indicates the FIFO data level in comparison to the water level set by FWM field in the control register. 0 Data level is above water mark. 1 The data level falls below the mark set by the FWM field.
2–0 FIFOAV	FIFO Available. These read-only bits indicate the data level remaining in the FIFO. An attempted write to these bits will not affect their value and no transfer error is generated. 000 No data available 001 1 word of data in FIFO 010 2 words of data in FIFO 011 3 words of data in FIFO 100 4 words of data in FIFO 101 Unused 110 Unused 111 Unused

32.3.2.3 PWM Interrupt Register (PWMIR)

The PWM Interrupt register (PWMIR) contains three bits that control the generation of the compare, rollover and FIFO empty interrupts. Figure 32-6 shows the register; Table 32-7 provides its field descriptions.

0x1000_6008 (PWMIR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CIE	RIE	FIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-6. PWM Interrupt Register (PWMIR)

Table 32-7. PWMIR Field Descriptions

Field	Description
31–3 Reserved	Reserved. These reserved bits are always read as zero.
2 CIE	Compare Interrupt Enable. This bit controls the generation of the Compare interrupt. 0 Compare Interrupt is not enabled. 1 Compare Interrupt is enabled.
1 RIE	Roll-over Interrupt Enable. This bit controls the generation of the Rollover interrupt. 0 Roll-over interrupt is not enabled. 1 Roll-over Interrupt is enabled.
0 FIE	FIFO Empty Interrupt Enable. This bit controls the generation of the FIFO Empty interrupt. 0 FIFO Empty is interrupt disabled. 1 FIFO Empty is interrupt enabled.

32.3.2.4 PWM Sample Register (PWMSAR)

The PWM sample register (PWMSAR) is the input to the FIFO. 16-bit words are loaded into the FIFO. The FIFO can be written and read when the PWM is disabled. The PWM runs at the last set duty-cycle setting if all the values of the FIFO has been utilized, until the FIFO is reloaded or the PWM is disabled. When a new value is written, the duty cycle changes after the current period is over.

A value of zero in the sample register will result in the ipp_pwm_pwmo output signal being always low/high (POUTC =00 it will be low and POUTC = 01 it will be high), and hence no output waveform will be produced. If the value in this register is higher than the PERIOD + 1, the output will never be reset/set depending on POUTC value. Figure 32-7 shows the register; Table 32-8 provides its field descriptions.

0x1000_600C (PWMSAR) Access: User Read/Write

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

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

Figure 32-7. PWM Sample Register (PWMSAR)

Table 32-8. PWMSAR Field Descriptions

Field	Description
31–16 Reserved	These are reserved bits and writing a value will not affect the functionality of PWM and are always read as zero.
15–0 SAMPLE	Sample Value. This is the input to the 4x16 FIFO. The value in this register denotes the value of the sample being currently used.

32.3.2.5 PWM Period Register (PWMPR)

The PWM period register (PWMPR) determines the period of the PWM output signal. After the counter value matches PERIOD + 1, the counter is reset to start another period.

PWMO (Hz) = PCLK(Hz)/(period +2)

A value of zero in the PWMPR results in a period of two clock cycles for the output signal. Writing 0xFFFF to this register achieves the same result as writing 0xFFFE.

A change in the period value due to a write in PWMPR results in the counter being reset to zero and the start of a new count period. Figure 32-8 shows the register; Table 32-9 provides its field descriptions.

0x1000_6010 (PWMPR) Access: User Read/Write

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

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

Figure 32-8. PWM Period Register (PWMPR)

Table 32-9. PWMPR Field Descriptions

Field	Description
31–16 Reserved	These are reserved bits and writing a value will not affect the functionality of PWM and are always read as zero.
15–0 PERIOD	Period Value. These bits determine the Period of the count cycle. The counter counts up to [Period Value] +1 and is then reset to 0x0000.

32.3.2.6 PWM Counter Register (PWMCNR)

The read-only pulse-width modulator counter register (PWMCNR) contains the current count value and can be read at any time without disturbing the counter. Figure 32-9 shows the register; Table 32-10 provides its field descriptions.

0x1000_6014 (PWMCNR) Access: User read-only

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

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

Figure 32-9. PWM Counter Register (PWMCNR)

Table 32-10. PWMCNR Field Descriptions

Field	Description
31–16 Reserved	These are reserved bits and writing a value will not affect the functionality of PWM and are always read as zero.
15–0 COUNT	Counter Value. These bits are the counter register value and denotes the current count state the counter register is in.

32.4 Functional Description

The following sections detail the PWM operation and function.

32.4.1 Operation

The output of the PWM is a toggling signal whose frequency and duty cycle can be modulated by programming the appropriate registers. It has a 16-bit up counter which counts from 0x0000 until the counter value equals the [Value in Period register] + 1. After this match occurs the Counter is reset to 0x0000.

At the beginning of a count period cycle, the PWMO pin is set to one (default) and the counter begins counting up from 0x0000. The sample value in the sample FIFO is compared on each count of prescaler clock. When the sample and count values match, the PWMO signal is cleared to zero (default). The counter continues counting until the period match occurs and subsequently another period cycle begins.

When the PWM is enabled the counter starts running and generates an output with the reset values in the period and sample registers. It is recommended that the programming of these registers be done before PWM is enabled.

A hardware reset results in all the PWM count and sample registers begin cleared and the FIFO being flushed. The control register shows that FIFO is empty and it can be written into, and the PWM is disabled. A software reset has the same results, however the state of the STOPEN, DOZEN, WAITEN, and DBGEN

bits in the control register are not affected. Software reset can be asserted even when the PWM is in disabled state.

32.4.1.1 Clocks

The clock that feeds the prescaler can be selected from the following:

- High Frequency Clock (ipg_clk_highfreq) pat_ref or ckih

  This is a high frequency clock provided by the Clock Control Module (CCM). This clock is supposed to be on in the low power mode when the ipg_clk is turned off. Thus the PWM can be run on this clock in the low power mode. The CRM is expected to provide this clock after synchronizing it to ahb_clk in the normal functional mode and switch to the unsynchronized version in the low power mode.
- Low Reference Clock (ipg_clk_32k) ckil

  This is the 32 KHz low reference clock which is provided by the CCM. This clock is supposed to be on in the low power mode when ipg_clk is turned off. Thus PWM can be run on this clock in the low power mode. The CRM is expected to provide this clock after synchronizing it to ahb_clk in the normal functional mode and switch to the unsynchronized version in the low power mode.
- Global Functional Clock (ipg_clk)

  This clock is supposed to be on in normal operations. In low power modes it can be switched off.

The clock input source is determined by the CLKSRC field of the PWM control register. The CLKSRC value should only be changed when the PWM is disabled.

The PWM input clock can be divided from 1 to 4096 by using a prescaler by appropriately setting the PRESCALER field in the control register. A change in the value of the PRESCALER field is immediately reflected on its output clock frequency.

32.4.1.2 FIFO

Digital sample values can be loaded into the pulse-width modulator as 16-bit words. The Endianness can be changed using the BCTR and HCTR bits of the control register. A 4-word (16-bit) FIFO minimizes interrupt overhead. A maskable interrupt is generated when the number of data words fall below the water level set by the FWM field in the control register.

A write in the sample register results in the value being stored into the FIFO if it is not full. A write when the FIFO is full sets FWE (FIFO write error) bit in the status register and the FIFO contents remain unchanged. The FIFO can be written into when the PWM is disabled. The FIFOAV field shows how many data words are currently contained in the FIFO and if it can be written into.

A read on the sample register yields the current FIFO value being used or will be used by the PWM for generation on the output signal. Therefore a write and a subsequent read on the sample register may result in different values being obtained.

32.4.1.3 Rollover and Compare Event

The counter is reset to 0x0000 after its value equals the PERIOD + 1 and resumes counting thereafter. This event is referred to as a rollover. When PERIOD = 0x0000, the counter is reset after count reaches 0x0001.

Therefore PERIOD = 0xFFFF or 0xFFFE results in the counter value being reset after count till 0xFFFF. During a rollover event the output is either set (default), reset or has no effect according to the programming of the POUTC field in the control register. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

When the counter value reaches the sample value the output of the PWM is reset (default), set or has no effect according to the programming of the POUTC field of control register. This event is referred to as a compare event. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

If the rollover event sets the PWM output signal the compare event will reset it and vice versa for a particular programming configuration of POUTC field.

32.4.1.4 Low Power Mode Behavior

In low power modes if the clock from the selected clock source is available, the PWM counter continues to run and an output is produced depending on whether the control bit for that mode is set. In the absence of the clock itself or if the corresponding low power bit in the control register is 0, the counter is reset and resumes counting when it exits the low power mode.

32.4.1.5 Debug Mode Behavior

In debug mode, PWM has the option of continuing to run or be halted. If the DBGEN bit is not set in the PWMCR, the PWM is halted. If the DBGEN bit is set, then the PWM will continue to run in the debug mode.

32.5 PWM Clocking

Figure 32-10 shows the relationship of clocks used by the PWM.

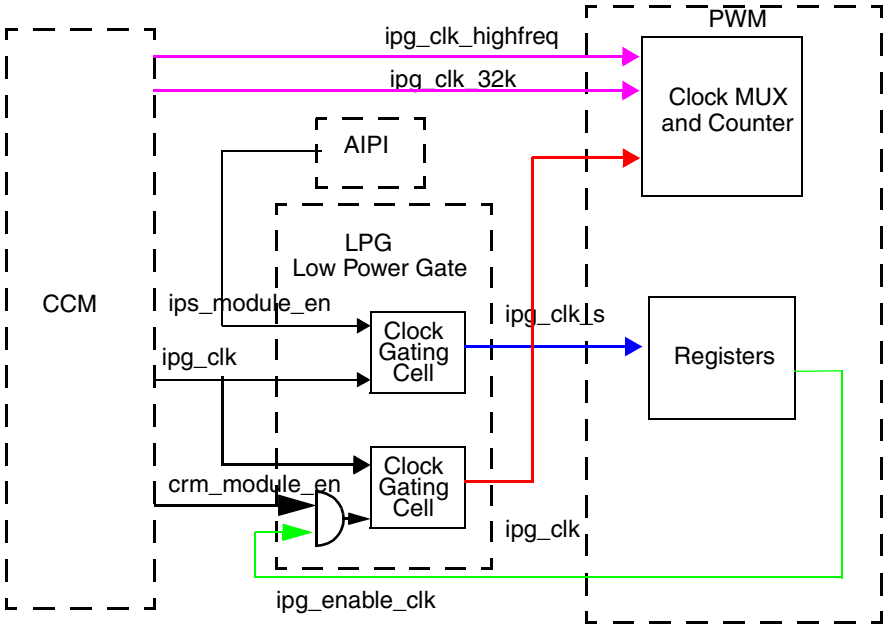


Figure 32-10. PWM Clocking

32.5.1 PWM Clock Inputs

Figure 32-11 shows the clock that feeds the prescaler (sys_clk).

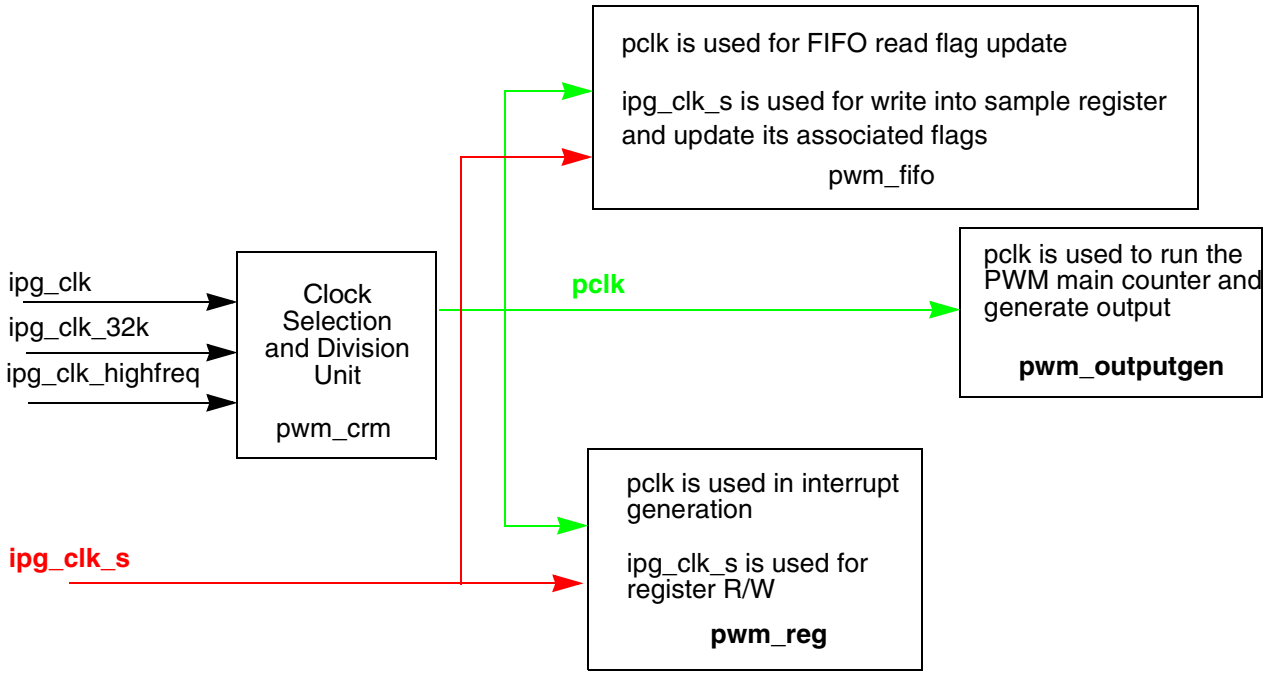


Figure 32-11. Clock Distribution Inside PWM

The clock that feeds the prescaler shown in Figure 32-11 can be selected from the following clock inputs:

- High frequency Clock (ipg_clk_highfreq) pat_ref or ckih
- Low Reference Clock (ipg_clk_32k) ckil
- Global Functional Clock (ipg_clk)

The selected clock sys_clk is prescaled with a 12-bit prescaler value. The sys_clk is gated with an enable signal that is generated from a prescaler counter and PWM enable to get the pclk. The main PWM counter runs on pclk.

pclk is used to generate the output signal of PWM and also the interrupts.

ipg_clk_s is the clock used for register read/write.

The only hand instantiated clock gating inside the module is done inside pwm_crm submodule for division of sys_clk to generate the pclk.

Figure 32-12 shows an overview of the clock selection and division unit.

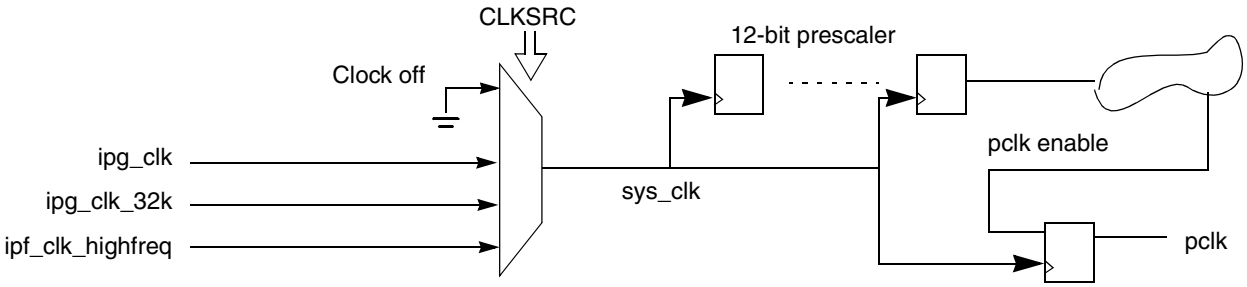


Figure 32-12. Clock Selection and Division Unit

32.5.2 ipg_enable_clk Generation

Whenever the module is enabled and ipg_clk is selected, this signal is asserted. Figure 32-13 shows the ipg_enable_clk generation logic.

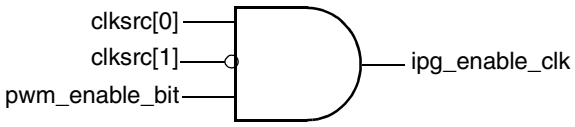


Figure 32-13. ipg_enable_clk Generation Logic

Chapter 33
Real Time Clock (RTC)

33.1 Introduction

See Figure 33-1 for a block diagram of the functional organization of the Real Time Clock (RTC) block. The block consists of the following blocks:

- Prescaler
- Time-of-day (TOD) clock counter
- Alarm
- Sampling timer
- Minute stopwatch
- Associated control and bus interface hardware

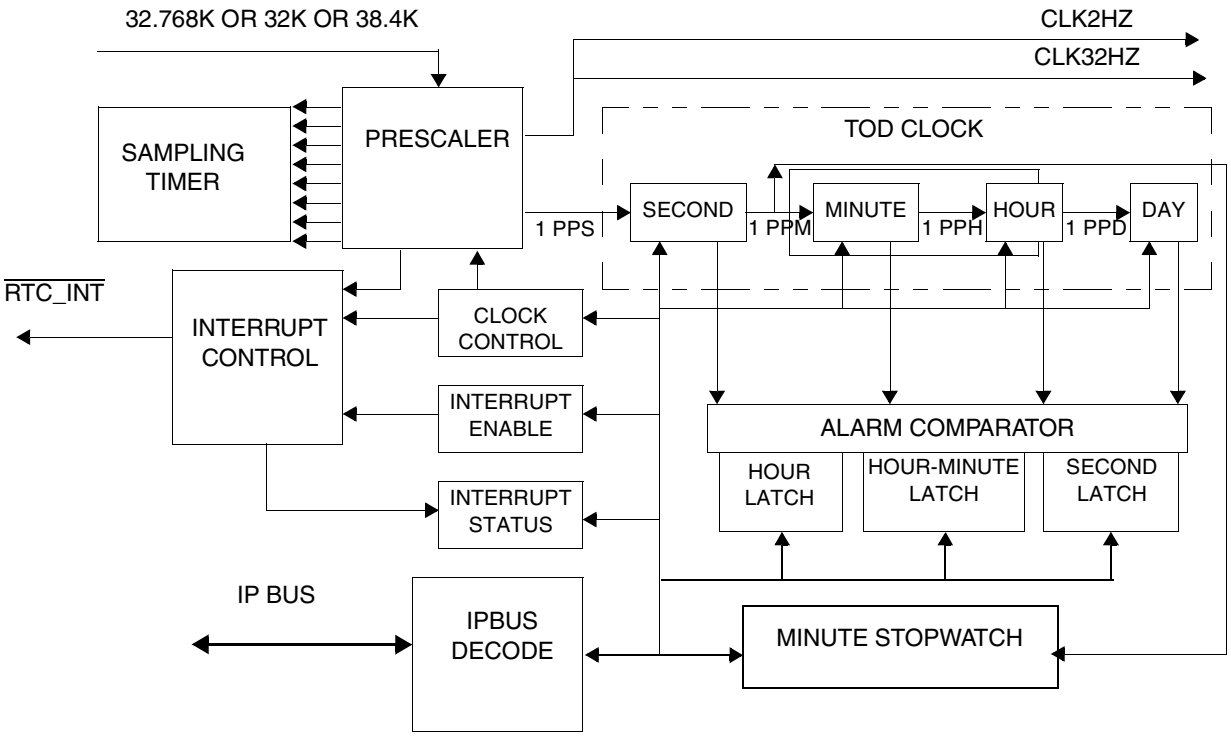


Figure 33-1. Real Time Clock Block Diagram

33.2 Overview

This section discusses how to operate and program the real-time clock (RTC) module that maintains the system clock, provides stopwatch, alarm, and interrupt functions, and supports the features described in Section 33.2.1, "Features."

33.2.1 Features

The RTC module includes the following features:

- Full clock: days, hours, minutes, seconds
- Minute countdown timer with interrupt
- Programmable daily alarm with interrupt
- Sampling timer with interrupt
- Once-per-day, once-per-hour, once-per-minute, and once-per-second interrupts
- Operation at 32.768 kHz or 32 kHz, or 38.4 kHz (determined by reference clock crystal)

33.2.2 Modes of Operation

The prescaler converts the incoming crystal reference clock to a 1 Hz signal that is used to increment the seconds, minutes, hours, and days TOD counters. The alarm functions, when enabled, generate RTC interrupts when the TOD settings reach programmed values. The sampling timer generates fixed-frequency interrupts, and the minute stopwatch allows for efficient interrupts on minute boundaries.

- Prescaler and Counter

  The prescaler divides the reference clock down to 1 Hz. The reference frequencies of 32.768 kHz, 38.4 kHz and 32 kHz are supported.

  The counter portion of the RTC module consists of four groups of counters that are physically located in three registers:

  — The 6-bit seconds counter is located in the SECONDS register

  — The 6-bit minutes counter and the 5-bit hours counter are located in the HOURMIN register

  — The 16-bit day counter is located in the DAYR register

- Alarm

  There are three alarm registers that mirror the three counter registers. An alarm is set by accessing the real-time clock alarm registers (ALRM_HM, ALRM_SEC, and DAYALARM) and loading the exact time that the alarm should generate an interrupt. When the TOD clock value and the alarm value coincide, an interrupt occurs.

- Sampling Timer

  The sampling timer is designed to support application software. The sampling timer generates a periodic interrupt with the frequency specified by the SAMx bits of the RTCIENR register. This timer can be used for digitizer sampling, keyboard debouncing, or communication polling. The sampling timer operates only if the real-time clock is enabled. See Table 33-15 for the list of the interrupt frequencies of the sampling timer for the possible reference clocks.

- Minute Stopwatch

  The minute stopwatch performs a countdown with a one minute resolution. It can be used to generate an interrupt on a minute boundary.

33.3 External Signal Description

The RTC has no external signals.

33.3.1 Overview

There are no signals connected off the chip.

See Table 33-1 for a complete list of RTC block level signal names.

33.4 Memory Map and Register Definition

The RTC module has ten 32-bit registers. Section 33.4.3, "Register Descriptions" provides the detailed descriptions for all of the RTC registers.

33.4.1 Memory Map

Table 33-2 shows the RTC memory map.

Table 33-2. RTC Register Memory Map

Address	Register	Access	Reset Values	Section/Page
0x1000_7000 (HOURMIN)	RTC Hours and Minutes Counter Register (HOURMIN)	R/W	0x0000_– – – –	33.4.3.1/33-6
0x1000_7004 (SECONDS)	RTC Seconds Counter Register (SECONDS)	R/W	0x0000_00 – –	33.4.3.2/33-7
0x1000_7008 (ALRM_HM)	RTC Hours and Minutes Alarm Register (ALRM_HM)	R/W	0x0000_0000	33.4.3.3/33-7
0x1000_700C (ALRM_SEC)	RTC Seconds Alarm Register (ALRM_SEC)	R/W	0x0000_0000	33.4.3.4/33-8
0x1000_7010 (RTCCTL)	RTC Control Register (RCCTL)	R/W	0x0000_0000	33.4.3.5/33-9
0x1000_7014 (RTCISR)	RTC Interrupt Status Register (RTCISR)	R/W	0x0000_0000	33.4.3.6/33-10
0x1000_7018 (RTCIENR)	RTC Interrupt Enable Register (RTCIENR)	R/W	0x0000_0000	33.4.3.7/33-13
0x1000_701C (STPWCH)	Stopwatch Minutes Register (STPWCH)	R/W	0x0000_0000	33.4.3.8/33-14
0x1000_7020 (DAYR)	RTC Days Counter Register (DAYR)	R/W	0x0000_– – – –	33.4.3.9/33-15
0x1000_7024 (DAYALARM)	RTC Days Alarm Register (DAYALARM)	R/W	0x0000_0000	33.4.3.10/33-16

33.4.2 Register Summary

Figure 33-2 shows the key to the register fields and Table 33-3 shows the register figure conventions.

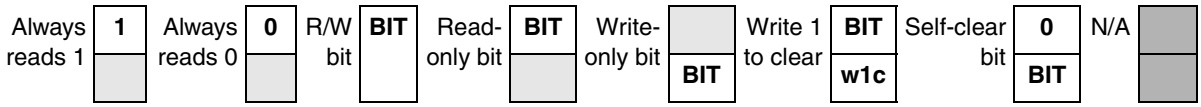


Figure 33-2. Key to Register Fields

Table 33-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 33-4 shows the RTC register summary.

Table 33-4. RTC Register Summary

Name		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
0x1000_7000 (HOURMIN)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	HOUR					0	0	MINUTES					
	W																
0x1000_7004 (SECONDS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	SECONDS					
	W																

Table 33-4. RTC Register Summary (continued)

Name		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
0x1000_7008 (ALRM_HM)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	HOUR					0	0	MINUTES					
	W																
0x1000_700C (ALRM_SEC)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	SECONDS					
	W																
0x1000_7010 (RTCCTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0		EN	XTL		0	0	0	GEN	SWR
	W																
0x1000_7014 (RTCISR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0	2HZ	0	HR	1HZ	DAY	ALM	MIN	SW
	W																
0x1000_7018 (RTCIENR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0	2HZ	0	HR	1HZ	DAY	ALM	MIN	SW
	W																
0x1000_701C (STPWCH)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	CNT					
	W																
0x1000_7020 (DAYR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DAYS															
	W																
0x1000_7024 (DAYALARM)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DAYSAL															
	W																

33.4.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bits and field functions follow the register diagrams in their bit order.

33.4.3.1 RTC Hours and Minutes Counter Register (HOURMIN)

The real-time clock hours and minutes counter register (HOURMIN) is used to program the hours and minutes for the TOD clock. It can be read or written at any time. After a write, the time changes to the new value. This register cannot be reset since the real-time clock is always enabled at reset.

See Figure 33-3 for an illustration of valid bits in the Hours and Minutes Counter Register and Table 33-5 for descriptions of the bit fields.

0x1000_7000 (HOURMIN) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	HOUR					0	0	MINUTES					
W																
Reset	0	0	0	—	—	—	—	—	0	0	—	—	—	—	—	—

Figure 33-3. RTC Hours and Minutes Counter Register

Table 33-5. RTC Hours and Minutes Counter Register Field Descriptions

Field	Description
31–13	Reserved
12–8 HOUR	Hour setting indicates the current hour that can be set to any value between 0 and 23. 00000 current hour is 0. 00001 current hour is 1. .... 10111 current hour is 23: Indicates the current hour that can be set to any value between 0 and 23.
7–5	Reserved
5–0 MINUTES	Minutes setting indicates the current minutes that can be set to any value between 0 and 59. 000000 current minute is 0. 000001 current minute is 1. .... 111011 current minute is 59.

The HOURS and MINUTES are not affected by any of the hardware or software reset, their RESET values are “Not unknown” but we just do not know the exact values to put into the table.

33.4.3.2 RTC Seconds Counter Register (SECONDS)

The real-time clock seconds register (SECONDS) is used to program the seconds for the TOD clock. It can be read or written at any time. After a write, the time changes to the new value. This register cannot be reset since the real-time clock is always enabled at reset.

See Figure 33-4 for an illustration of valid bits in the RTC Seconds Counter Register and Table 33-6 for descriptions of the bit fields.

0x1000_7004 (SECONDS) Access: User read/write

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

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

Figure 33-4. RTC Seconds Counter Register

Table 33-6. RTC Seconds Counter Register Field Descriptions

Field	Description
31–6	Reserved
5–0 SECONDS	Seconds setting indicates the current seconds that can be set to any value between 0 and 59. 000000 current second is 0. 000001 current second is 1. .... 111011 current second is 59.

The SECONDS are not affected by any of the hardware or software reset, their RESET values are “Not unknown,” but we just do not know the exact values to put into the table.

33.4.3.3 RTC Hours and Minutes Alarm Register (ALRM_HM)

The real-time clock hours and minutes alarm (ALRM_HM) register is used to configure the hours and minutes setting for the alarm. The alarm settings can be read or written at any time.

See Figure 33-5 for an illustration of valid bits in the RTC Hours and Minutes Alarm Register and Table 33-7 for descriptions of the bit fields.

0x1000_7008 (ALRM_HM) Access: User read/write

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

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

Figure 33-5. RTC Hours and Minutes Alarm Register

Table 33-7. RTC Hours and Minutes Alarm Register Field Descriptions

Field	Description
31–13	Reserved
12–8 HOURS	Hour setting of the alarm hours that can be set to any value between 0 and 23. 00000 current hour is 0. 00001 current hour is 1. .... 10111 current hour is 23.
7–6	Reserved
5–0 MINUTES	Minutes setting of the alarm minutes that can be set to any value between 0 and 59. 000000 current minute is 0. 000001 current minute is 1. .... 111011 current minute is 59.

33.4.3.4 RTC Seconds Alarm Register (ALRM_SEC)

The real-time clock seconds alarm (ALRM_SEC) register is used to configure the seconds setting for the alarm. The alarm settings can be read or written at any time.

See Figure 33-6 for an illustration of valid bits in the RTC Seconds Alarm Register and Table 33-8 for descriptions of the bit fields.

0x1000_700C (ALRM_SEC) Access: User read/write

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

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

Figure 33-6. RTC Seconds Alarm Register

Table 33-8. RTC Seconds Alarm Register Field Descriptions

Field	Description
31–6	Reserved
5–0 SECONDS	Seconds setting of the alarm seconds, can be set to any value between 0 and 59. 000000 current second is 0. 000001 current second is 1. .... 111011 current second is 59.

33.4.3.5 RTC Control Register (RTCCTL)

The real-time clock control (RTCCTL) register is used to enable the real-time clock module and specify the reference frequency information for the prescaler.

See Figure 33-7 for an illustration of valid bits in the RTC Control Register and Table 33-9 for descriptions of the bit fields.

0x1000_7010 (RTCCTL) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	EN	XTL		0	0	0	GEN	SWR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-7. RTC Control Register

Table 33-9. RTC Control Register Field Descriptions

Field	Description
31–8	Reserved
7 EN	Enables/Disables the real-time clock. The software reset bit (SWR) has no effect on this bit. 0 Disables the real-time clock 1 Enables the real-time clock
6–5 XTL	Crystal Selection. Selects the proper input crystal frequency. It is important to set these bits correctly or the real-time clock will be inaccurate. 00 Input crystal frequency is 32.768 kHz. 01 Input crystal frequency is 32 kHz. 10 Input crystal frequency is 38.4 kHz. 11 Input crystal frequency is 32.768 kHz.
4–2	Reserved
1 GEN	GEN — IPG_CLK gating enable. Decides whether to enable or disable the ipg_clk gating. Upon reset the ipg_clk gating is enabled. 0 Enables ipg_clk gating 1 Disables ipg_clk gating
0 SWR	Software reset. Resets the module to its default state. However, a software reset will have no effect on the RTC enable (EN) bit. 0 No effect 1 Reset the module to its default state

33.4.3.6 RTC Interrupt Status Register (RTCISR)

The real-time clock interrupt status register (RTCISR) indicates the status of the various real-time clock interrupts. When an event of the types included in this register occurs, then the bit will be set in this register regardless of its corresponding interrupt enable bit.These bits are cleared by writing a value of 1, which also clears the interrupt. Interrupts may occur while the system clock is idle or in sleep mode. Every interrupt status bit is independent of each other.

See Figure 33-8 for an illustration of valid bits in the RTC Interrupt Status Register and Table 33-10 for descriptions of the bit fields.

0x1000_7014 (RTCISR) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0	2HZ	0	HR	1HZ	DAY	ALM	MIN	SW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-8. RTC Interrupt Status Register

Table 33-10. RTC Interrupt Status Register Field Descriptions

Field	Description
31–16	Reserved
15 SAM7	Sampling Timer Interrupt Flag at SAM7 Frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 512, 500, or 600 Hz depending on different input clock. 0 No SAM7 interrupt has occurred. 1 A SAM7 interrupt has occurred.
14 SAM6	Sampling Timer Interrupt Flag at SAM6 Frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 256, 250, or 300 Hz depending on different input clock. 0 No SAM6 interrupt has occurred. 1 A SAM6 interrupt has occurred.
13 SAM5	Sampling Timer Interrupt Flag at SAM5 Frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 128, 125, or 150 Hz depending on different input clock. 0 No SAM5 interrupt has occurred. 1 A SAM5 interrupt has occurred.
12 SAM4	Sampling Timer Interrupt Flag at SAM4 Frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 64, 62.5, or 75 Hz depending on different input clock. 0 No SAM4 interrupt has occurred. 1 A SAM4 interrupt has occurred.
11 SAM3	Sampling Timer Interrupt Flag at SAM3 Frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 32, 31.25, or 37.5 Hz depending on different input clock. 0 No SAM3 interrupt has occurred. 1 A SAM3 interrupt has occurred.

Table 33-10. RTC Interrupt Status Register Field Descriptions (continued)

Field	Description
10 SAM2	Sampling Timer Interrupt Flag at SAM2 Frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 16, 15.625, or 18.75 Hz depending on different input clock. 0 No SAM2 interrupt has occurred. 1 A SAM2 interrupt has occurred.
9 SAM1	Sampling Timer Interrupt Flag at SAM1 Frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 8, 7.8125, or 9.375 Hz depending on different input clock. 0 No SAM1 interrupt has occurred. 1 A SAM1 interrupt has occurred.
8 SAM0	Sampling Timer Interrupt Flag at SAM0 Frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 4, 3.90625, or 4.6875 Hz depending on different input clock. 0 No SAM0 interrupt has occurred. 1 A SAM0 interrupt has occurred.
7 2 Hz	2 Hz Flag. Indicates that an interrupt has occurred. If enabled, this bit is set at every 2 Hz frequency. 0 No 2 Hz interrupt has occurred. 1 A 2 Hz interrupt has occurred.
6	Reserved
5 HR	Hour Flag. Indicates that the hour counter has increment. If enabled, this bit is set on every increment of the hour counter in the time-of-day clock. 0 No 1-hour interrupt has occurred. 1 A 1-hour interrupt has occurred.
4 1 Hz	1 Hz Flag. Indicates that the second counter has increment. If enabled, this bit is set on every increment of the second counter of the time-of-day clock. 0 No 1 Hz interrupt has occurred. 1 A 1 Hz interrupt has occurred.
3 DAY	Day Flag. indicates that the day counter has increment. If enabled, this bit is set on every increment of the day counter of the time-of-day clock. 0 No 24-hour rollover interrupt has occurred. 1 A 24-hour rollover interrupt has occurred.
2 ALM	Alarm Flag. Indicates that the real-time clock matches the value in the alarm registers. The alarm will reoccur every 65536 days. For a single alarm, clear the interrupt enable for this bit in the interrupt service routine. 0 No alarm interrupt has occurred. 1 An alarm interrupt has occurred.
1 MIN	Minute Flag. Indicates that the minute counter has increment. If enabled, this bit is set on every increment of the minute counter in the time-of-day clock. 0 No 1-minute interrupt has occurred. 1 A 1-minute interrupt has occurred.
0 SW	Stopwatch Flag. Indicates that the stopwatch countdown timed out. 0 The stopwatch did not time out. 1 The stopwatch timed out.

33.4.3.7 RTC Interrupt Enable Register (RTCIENR)

The real-time clock interrupt enable register (RTCIENR) is used to enable/disable the various real-time clock interrupts. Masking an interrupt bit has no effect on its corresponding status bit. Every interrupt enable bit is independent of each other.

See Figure 33-9 for an illustration of valid bits in the RTC Interrupt Enable Register and Table 33-11 for descriptions of the bit fields.

0x1000_7018 (RTCIENR) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0	2HZ	0	HR	1HZ	DAY	ALM	MIN	SW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-9. RTC Interrupt Enable Register

Table 33-11. RTC Interrupt Enable Register Field Descriptions

Field	Description
31–16	Reserved
15 SAM7	Sampling Timer Interrupt Flag at SAM7 Interrupt. Enables/Disables the real-time sampling timer interrupt 7. 0 The SAM7 interrupt is disabled. 1 The SAM7 interrupt is enabled.
14 SAM6	Sampling Timer interrupt Flag at SAM 6 Interrupt. Enables/Disables the real-time sampling timer interrupt 6. 0 The SAM6 interrupt is disabled. 1 The SAM6 interrupt is enabled.
13 SAM5	Sampling Timer Interrupt Flag at SAM5 Interrupt. Enables/Disables the real-time sampling timer interrupt 5. 0 The SAM5 interrupt is disabled. 1 The SAM5 interrupt is enabled.
12 SAM4	Sampling Timer Interrupt Flag at SAM4 Interrupt. Enables/Disables the real-time sampling timer interrupt 4. 0 The SAM4 interrupt is disabled. 1 The SAM4 interrupt is enabled.
11 SAM3	Sampling Timer Interrupt Flag at SAM3 Interrupt. Enables/Disables the real-time sampling timer interrupt 3. 0 The SAM3 interrupt is disabled. 1 The SAM3 interrupt is enabled.

Table 33-11. RTC Interrupt Enable Register Field Descriptions (continued)

Field	Description
10 SAM2	Sampling Timer Interrupt Flag at SAM2 Interrupt. Enables/Disables the real-time sampling timer interrupt 2. 0 The SAM2 interrupt is disabled. 1 The SAM2 interrupt is enabled.
9 SAM1	Sampling Timer Interrupt Flag at SAM1 Interrupt. Enables/Disables the real-time sampling timer interrupt 1. 0 The SAM1 interrupt is disabled. 1 The SAM1 interrupt is enabled.
8 SAM0	Sampling Timer Interrupt Flag at SAM0 Interrupt. Enables/Disables the real-time sampling timer interrupt 0. 0 The SAM0 interrupt is disabled. 1 The SAM0 interrupt is enabled.
7 2 Hz	2 Hz Interrupt Enable. Enables/Disables an interrupt at a 2 Hz rate. 0 The 2-Hz interrupt is disabled. 1 The 2-Hz interrupt is enabled.
6	Reserved
5 HR	Hour Interrupt Enable. Enables/Disables an interrupt whenever the hour counter of the real-time clock increments. 0 The 1-hour interrupt id disabled. 1 The 1-hour interrupt is enabled.
4 1HZ	1 Hz Interrupt Enable. Enables/Disables an interrupt whenever the second counter of the real-time clock increments. 0 The 1-Hz interrupt is disabled. 1 The 1-Hz interrupt is enabled.
3 DAY	Day Interrupt Enable. Enables/Disables an interrupt whenever the hours counter rolls over from 23 to 0. (midnight rollover) 0 The 24-hour interrupt is disabled. 1 The 24-hour interrupt is enabled.
2 ALM	Alarm Interrupt Enable. Enables/Disables the alarm interrupt. 0 The alarm interrupt is disabled. 1 The alarm interrupt is enabled.
1 MIN	Minute Interrupt Enable. Enables/Disables an interrupt whenever the minute counter of the real-time clock increments. 0 The 1-minute interrupt is disabled. 1 The 1-minute interrupt is enabled.
0 SW	Stopwatch Interrupt Enable. Enables/Disables the stopwatch interrupt. **Note:** The stopwatch counts down and remains at decimal -1 until it is reprogrammed. If this bit is enabled with -1 (decimal) in the STPWCH register, an interrupt will be posted on the next minute tick. 0 Stopwatch interrupt is disabled. 1 Stopwatch interrupt is enabled.

33.4.3.8 RTC Stopwatch Minutes Register (STPWCH)

The stopwatch minutes (STPWCH) register contains the current stopwatch countdown value. When the minute counter of the TOD clock increments, the value in this register decrements.

See Figure 33-10 for an illustration of valid bits in the Stopwatch Minutes Counter Register and Table 33-12 for descriptions of the bit fields.

0x1000_701C (STPWCH) Access: User read/write

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

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

Figure 33-10. RTC Stopwatch Minutes Register

Table 33-12. RTC Stopwatch Minutes Register Field Descriptions

Field	Description
31–6	Reserved
5-0 CNT	Stopwatch Count. Contains the stopwatch countdown value. **Note:** The stopwatch counter is decremented by the minute (MIN) tick output from the real-time clock, so the average tolerance of the count is 0.5 minutes. For better accuracy, enable the stopwatch by polling the MIN bit of the RTCISR register or by polling the minute interrupt service routine. 000000 stopwatch countdown value is 0. 000001 stopwatch countdown value is 1. .... 111111 stopwatch countdown value is 63.

33.4.3.9 RTC Days Counter Register (DAYR)

The real-time clock days counter register (DAYR) is used to program the day for the TOD clock. When the HOUR field of the HOURMIN register rolls over from 23 to 00, the day counter increments. It can be read or written at any time. After a write, the time changes to the new value. This register cannot be reset since the real-time clock is always enabled at reset.

See Figure 33-11 for an illustration of valid bits in the Counter Register and Table 33-13 for descriptions of the bit fields.

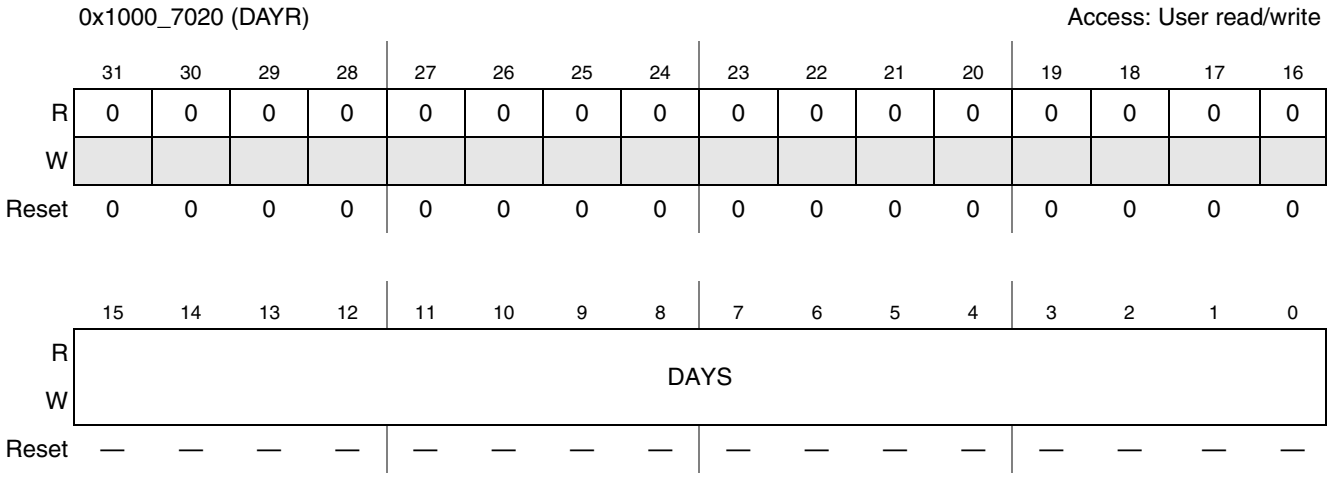


Figure 33-11. RTC Days Counter Register

Table 33-13. RTC Days Counter Register Field Descriptions

Field	Description
31–16	Reserved
15–0 DAYS	Day Setting. Indicates the current day count, can be set to any values between 0 and 65535. 0x0000 current day count is 0. 0x0001 current day count is 1. .... 0xFFFF current day count is 65535.

The DAYS are not affected by any of the hardware or software reset, their reset values are “Not unknown,” but we just do not know the exact values to put into the table.

33.4.3.10 RTC Day Alarm Register (DAYALARM)

The real-time clock day alarm (DAYALARM) register is used to configure the day for the alarm. The alarm settings can be read or written at any time.

See Figure 33-12 for an illustration of valid bits in the RTC Day Alarm Register and Table 33-14 for descriptions of the bit fields.

0x1000_7024 (DAYALARM) Access: User read/write

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

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

Figure 33-12. RTC Day Alarm Register

Table 33-14. RTC Day Alarm Register Field Descriptions

Field	Description
31–16	Reserved
15–0 DAYSAL	Day Setting of the Alarm. Indicates the current day setting of the alarm. It can be set to any value between 0 and 65535. 0x0000 current day setting of alarm is 0. 0x0001 current day setting of alarm is 1. .... 0xFFFF current day setting of alarm is 65535.

33.5 Functional Description

The prescaler converts the incoming crystal reference clock to a 1 Hz signal which is used to increment the seconds, minutes, hours, and days TOD counters. The alarm functions, when enabled, generate RTC interrupts when the TOD settings reach programmed values. The sampling timer generates fixed-frequency interrupts, and the minute stopwatch allows for efficient interrupts on minute boundaries.

33.5.1 Prescaler and Counter

The prescaler divides the reference clock down to 1 Hz. The reference frequencies of 32.768 kHz, 38.4 kHz, and 32 kHz are supported. The prescaler stages are tapped to support the sampling timer.

The counter portion of the RTC module consists of four groups of counters that are physically located in three registers:

- The 6-bit seconds counter is located in the SECONDS register.
- The 6-bit minutes counter and the 5-bit hours counter are located in the HOURMIN register.
- The 16-bit day counter is located in the DAYR register.

These counters cover a 24-hour clock over 65536 days. All three registers can be read or written at any time.

Interrupts signal when each of the four counters increments, and can be used to indicate when a counter rolls over. For example, each tick of the seconds counter causes the 1 Hz interrupt flag to be set. When the seconds counter rolls from 59 to 00, the minute counter increments and the MIN interrupt flag is set. The same is true for the minute counter with the HR signal, and the hour counter with the DAY signal.

33.5.2 Alarm

There are three alarm registers that mirror the three counter registers:

- ALRM_HM
- ALRM_SEC
- DAYALARM

An alarm is set by accessing the three real-time clock alarm registers and loading the exact time that the alarm should generate an interrupt. When the TOD clock value and the alarm value coincide, if the ALM bit in the real-time clock interrupt enable register (RTCIENR) is set, an interrupt occurs.

NOTE

If the alarm is not disabled, it will reoccur every 65536 days. If a single alarm is desired, the alarm function must be disabled through the RTC Interrupt Enable Register (RTCIENR).

33.5.3 Sampling Timer

The sampling timer is designed to support application software. The sampling timer generates a periodic interrupt with the frequency specified by the SAMx bits of the RTCIENR register. This timer can be used for digitizer sampling, keyboard debouncing, or communication polling. The sampling timer operates only if the real-time clock is enabled. See Table 33-15 for the list of the interrupt frequencies of the sampling timer for the possible reference clocks.

Multiple SAMx bits may be set in the RTC Interrupt Enable Register (RTCIENR). The corresponding bits in the RTC Interrupt Status Register (RTCISR) will be set at the noted frequencies.

Table 33-15. Sampling Timer Frequencies

Sampling Frequency	32.768 kHz Reference Clock	32 kHz Reference Clock	38.4 kHz Reference Clock
SAM7	512 Hz	500 Hz	600 Hz
SAM6	256 Hz	250 Hz	300 Hz
SAM5	128 Hz	125 Hz	150 Hz
SAM4	64 Hz	62.5 Hz	75 Hz
SAM3	32Hz	31.25 Hz	37.5 Hz
SAM2	16 Hz	15.625 Hz	18.75 Hz
SAM1	8 Hz	7.8125 Hz	9.375 Hz
SAM0	4 Hz	3.90625 Hz	4.6875 Hz

33.5.4 Minute Stopwatch

The minute stopwatch performs a countdown with a one minute resolution. It can be used to generate an interrupt on a minute boundary. For example, to turn off the LCD controller after five minutes of inactivity, program a value of 0x04 into the Stopwatch Count (CNT) field of the Stopwatch Minutes (STPWCH) register. At each minute, the value in the stopwatch is decremented. When the stopwatch value reaches –1, the interrupt occurs. The value of the register does not change until it is reprogrammed.

NOTE

The actual delay includes the seconds from setting the stopwatch to the next minute tick.

33.6 Initialization/Application Information

33.6.1 Flowchart of RTC Operation

See Figure 33-13 for the illustration of the flowchart of a typical RTC operation. Refer to Example 33-1 for the code example of ARM instruction for configuring RTC.

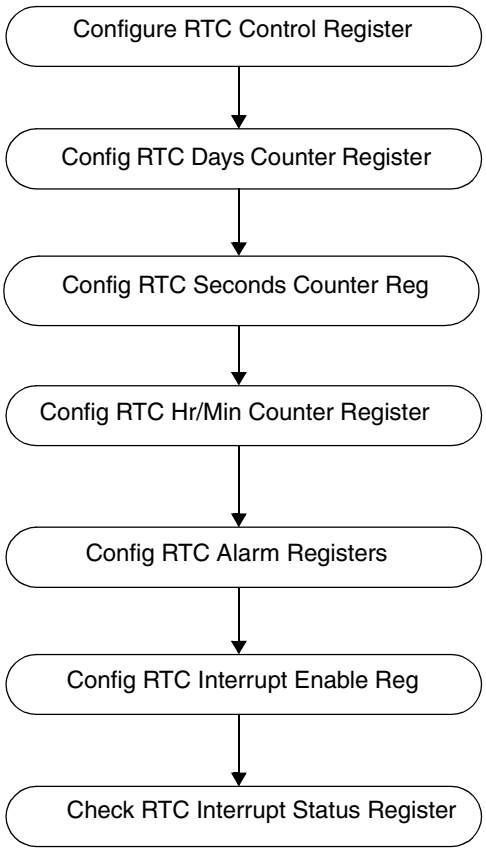


Figure 33-13. Flowchart of RTC Operation

33.6.2 Code Example of ARM Instruction

Example 33-1. Code Example of ARM Instruction

```
        LDR     r1,=RTC_BASE_ADDR

        LDR     r2,[r1,#0x10]
        ORR     r2,r2,#0x21
        STR     r2,[r1,#0x10]           ; Software reset and 32k crystal


        LDR     r3,=0x0000
        STR     r3,[r1,#0x20]           ;DAY
        LDR     r3,=0x00038
        STR     r3,[r1,#0x04]           ;SECOND
        LDR     r3,=0x173B
        STR     r3,[r1]                 ;HR, MIN

        LDR     r3,=0x0001
        STR     r3,[r1,#0x24]           ;Alarm Day
        LDR     r3,=0x0000
        STR     r3,[r1,#0x08]           ;Alarm hour, minute
        LDR     r3,=0x01
        STR     r3,[r1,#0x0C]           ;Alarm seconds

        LDR     r2,[r1,#0x18]           ;set ALARM interrupt
        ORR     r2,r2,#0x4
        STR     r2,[r1,#0x18]

ALARM_STATUS_3
        LDR     r2,[r1,#0x18]           ;check ALARM STATUS FLAG
        TST     r2,#0x04
        BNE     ALARM_STATUS_3
```

Chapter 34
Watchdog Timer (WDOG)

The Watchdog (WDOG) timer module protects against system failures by providing a method of escaping from unexpected events or programming errors. Figure 34-1 shows the WDOG block diagram.

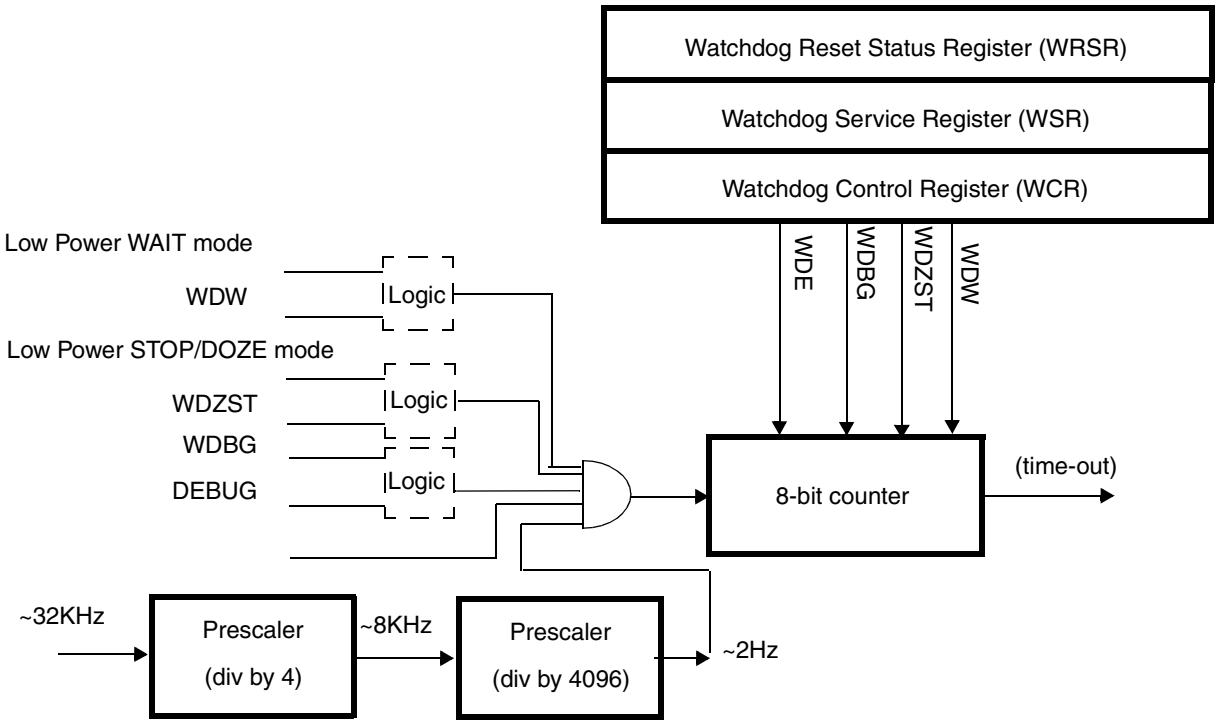


Figure 34-1. WDOG Block Diagram

34.1 Overview

Once the WDOG module is activated, it must be serviced by the software on a periodic basis. If servicing does not take place, the timer times out. Upon a time-out, the WDOG Timer module either asserts the $\overline{\text{WDOG}}$ signal or a system reset signal, wdog_rst, depending on the software configuration. The WDOG Timer module also generates a system reset via a software write to the Watchdog Control Register (WCR). The $\overline{\text{WDOG}}$ signal is asserted via a software write to the WCR, a detection of a clock monitor event, or upon a watchdog time-out. A state machine of the counter operation is shown in Figure 34-6, which demonstrates the time-out operation.

The WDOG module cannot be deactivated again after activation.

The input clocks to the WDOG module (ipg_clk_32k, ipg_clk, and ipg_clk_s) must be synchronized with each other.

NOTE

ipg_clk_32k is the synchronized version of the input ~32k clock (CKIL) on the IP global functional clock. Because the synchronized version is not available (the IP global functional clock is off) in low-power mode, it will receive the raw CKIL (ipg_clk_32k) clock from the CRM. These synchronizers will be bypassed while going into low power modes in CRM.

The WDOG module can continue or suspend the timer operation in the low power modes (WAIT and STOP).

34.1.1 Features

The WDOG features are as follows:

- Time-out periods from 0.5 seconds up to 128 seconds
- Time resolution of 0.5 seconds
- Configurable counters that can be programed to run or stop during low-power modes
- Configurable counters that can be programed to run or stop during DEBUG mode

34.2 External Signal Description

The WDOG module port signals going to pins are listed in Table 34-2.

Table 34-2. Signal Properties

Name	Port	Function	Reset State	Pull-Up
$\overline{\text{IPP\_WDOG}}$	—	Asserted by software timeout event	1	—
IPP_WDOG_OE	—	$\overline{\text{WDOG}}$ output enable at pin	0	—

34.2.1 Detailed External Signal Descriptions

34.2.1.1 $\overline{\text{IPP\_WDOG}}$, IPP_WDOG_OE

The $\overline{\text{IPP\_WDOG}}$ signal can be provided externally to the IC. It is asserted by a software request (setting the WCR bits) or a clock monitor event. The IPP_WDOG_OE signal is the output enable for the pin.

34.2.2 Internal Port Signals

The WDOG internal port signals are listed in Table 34-3.

Table 34-3. WDOG Module Port List

Port Name	Direction	Description
ipg_enable_clk	Output (O)	IP global clock gating signal. It is available to the Clock Controller to gate off the clock to the WDOG module for power saving.
ipg_clk	Input (I)	IP Global functional clock. All functionality inside the WDOG module is synchronized to this clock.
ipg_clk_s	I	IP slave bus clock. This clock is synchronized to ipg_clk and is only used for register read/write operations.
ipg_clk_32k	I	Low frequency (32.768 kHz) clock that continues to run in low-power mode. It is assumed that the Clock Controller will provide this clock signal synchronized to ipg_clk in the normal mode, and switch to a non-synchronized signal in low-power mode when the ipg_clk is off.
$\overline{\text{ipg\_hard\_async\_reset}}$	I	IP global hardware reset (active low)
$\overline{\text{ipg\_por}}$	I	Power-on-reset signal (active low)
ipg_debug	I	IP global signal indicating that WDOG should enter debug mode operation
ipg_stop	I	IP global signal indicating that WDOG should enter low-power Sleep Mode operation
ipg_wait	I	IP global signal indicating that WDOG should enter low-power wait mode operation
ips_rdata[15:0]	O	IP slave bus read data line
ips_xfr_wait	O	IP slave bus transfer wait indicator
ips_xfr_err	O	IP slave bus transfer error indicator
ips_module_en	I	IP slave bus module enable signal. This signal indicates when a module bus transaction is occurring.
ips_rwb	I	IP slave bus read/write signal. Shows whether a bus transaction is read or write.
ips_addr[13:1]	I	IP slave bus address signal. Denotes the register being accessed during a bus transaction.
ips_wdata[15:0]	I	IP slave bus write date line
ips_byte_15_8	I	IP slave bus byte enable signal for bits 15 to 8
ips_byte_7_0	I	IP slave bus byte enable signal for bits 7 to 0
ipt_reset	I	IP scan reset signal used to work in place of generated resets in scan mode (active low)
ipt_se_async	I	IP scan signal for bypassing generated resets and making latches transparent in scan mode
ipt_scan_mode	I	IP scan mode signal
ipt_se_gatedclk	I	IP scan mode clock gating signal
$\overline{\text{wdog\_rst}}$	O	Watchdog Timer reset (active low)
resp_sel	I	Indicates if the error response needs to be generated on access on unimplemented registers or not. Refer to Section 34.5.4, “Generation of Transfer Error on the IP Bus” for details.

34.3 Memory Map and Register Definitions

The WDOG module has three, user-accessible, 16-bit registers used to configure, operate, and monitor the state of the Watchdog Timer. Section 34.4, "Register Descriptions" provides the detailed descriptions for all of the WDOG registers.

34.3.1 Watchdog Timer Memory Map

Table 34-4 shows the WDOG memory map.

Table 34-4. WDOG Memory Map

Address	Register	Access	Reset Value	Section/Page
0x1000_2000 (WCR)	Watchdog Control Register	R/W	0x0030	34.4.1/34-5
0x1000_2002 (WSR)	Watchdog Status Register	R/W	0x0010	34.4.2/34-6
0x1000_2004 (WRSR)	Watchdog Reset Status Register	R	0x00– –	34.5.6/34-10

34.3.2 Register Summary

Figure 34-2 shows the key to the register fields, and Table 34-5 shows the register figure conventions.

Figure 34-2. Key to Register Fields

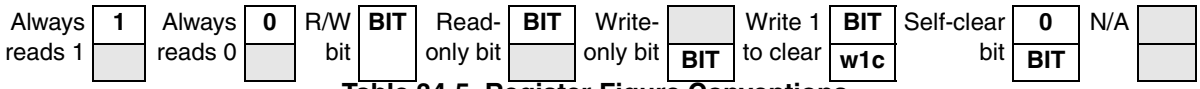


Table 34-5. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.

Table 34-5. Register Figure Conventions (continued)

Convention	Description
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 34-6 shows the WDOG register summary.

Table 34-6. WDOG Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000_2000 (WCR)	R	WT								0	WOE	WDA	SRS	WRE	WDE	WDBG	WDZST
	W																
0x1000_2002 (WSR)	R	WSR															
	W																
0x1000_2004 (WRSR)	R	0	0	0	0	0	0	0	0	0	0	JRST	PWR	EXT	CMON	TOUT	SFTW
	W																

34.4 Register Descriptions

34.4.1 Watchdog Control Register (WCR)

The Watchdog Control Register (WCR) is a 16-bit read/write register. It controls the WDOG operation. All bits except for bits[5:4] are cleared during reset. Bits[5:4] are set to 1 during reset. Figure 34-3 shows the register; Table 34-7 provides its field descriptions.

0x1000_2000 (WCR) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WT								0	0	WDA	SRS	WRE	WDE	WDBG	WDZST
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Figure 34-3. Watchdog Control Register

Table 34-7. WCR Register Descriptions

Field	Description
15–8 WT	Watchdog Timeout Field. This 8-bit field contains the time-out value that is loaded into the Watchdog counter after the service routine has been performed. After reset, WT[7:0] must have a value written to it before enabling the Watchdog.
7–6	Reserved
5 WDA	$\overline{\text{WDOG}}$ Assertion. Controls the software assertion of the $\overline{\text{WDOG}}$ signal. 0 Assert $\overline{\text{WDOG}}$ output. 1 No effect on system.

Table 34-7. WCR Register Descriptions (continued)

Field	Description
4 SRS	Software Reset Signal. Controls the software assertion of the WDOG-generated reset signal. This bit automatically resets to “1” after it has been asserted to “0”. **Note:** This bit does not generate the software reset to the module. The ipg_clk must be on to write to this bit. 0 Assert system reset signal 1 No effect on the system
3 WRE	$\overline{\text{wdog}}$/$\overline{\text{wdog\_rst}}$ Enable. Determines if the Watchdog generates a reset signal or a $\overline{\text{WDOG}}$ signal upon a Watchdog timeout. This is a write once-only bit. 0 Generate a reset signal 1 Generate a $\overline{\text{WDOG}}$ signal
2 WDE	Watchdog Enable. Enables or disables the WDOG module. Software can only write “1” in this bit. It is not possible to reset this bit by a software write, once the bit is set. **Note:** This bit can be set/reset as per the IP writes in debug mode (exception). 0 Disable the Watchdog 1 Enable the Watchdog
1 WDBG	Watchdog DEBUG Enable. Determines the operation of the WDOG module during DEBUG mode. This bit is write once-only. 0 Continue WDOG timer operation 1 Suspend the watchdog timer
0 WDZST	Watchdog Low Power. Determines the operation of the WDOG module during low-power modes. This bit is write once-only. **Note:** The WDOG module can continue/suspend the timer operation in the low-power modes (WAIT and STOP). 0 Continue timer operation 1 Suspend the watchdog timer

34.4.2 Watchdog Service Register (WSR)

When enabled, the WDOG requires that a service sequence be written to the Watchdog Service Register (WSR). Figure 34-4 shows the register; Table 34-8 provides its field descriptions.

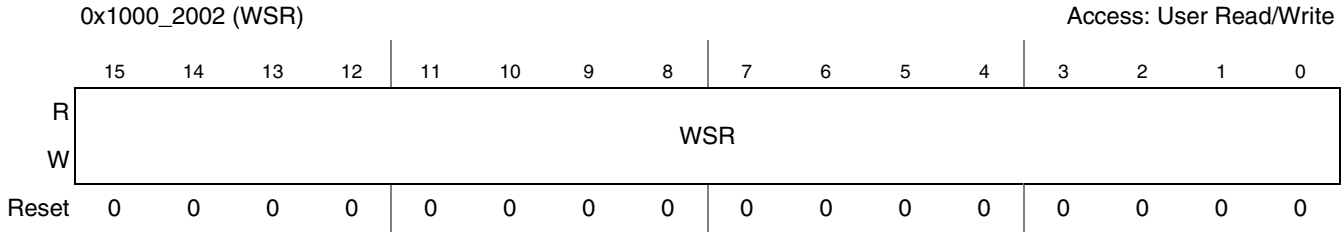


Figure 34-4. Watchdog Service Register (WSR)

Table 34-8. Watchdog Service Register Description

Field	Description
15–0 WSR	Watchdog Service Register. This 15-bit field contains the Watchdog service sequence. Both writes must occur in the order listed prior to the time-out, but any number of instructions can be executed between the two writes. The service sequence must be performed as follows: Write 0x 5555 to the Watchdog Service Register (WSR) Write 0x AAAA to the Watchdog Service Register (WSR)

34.4.2.1 Watchdog Reset Status Register (WRSR)

The WRSR is a read-only register that records the source of the output reset assertion. It is not cleared by a hard reset. It records the source of the output reset assertion. Therefore, only one bit in the WRSR will always be asserted high. The register will always indicate the source of the last reset.

A reset can be generated by the following sources, as listed in priority from highest to lowest:

- Power-on reset
- External reset
- Watchdog Time-out
- Software reset

Figure 34-5 shows the register; Table 34-8 provides its field descriptions.

NOTE

Do not write to this register. Attempting to write to the WRSR register generates a bus error.

0x1000_2004 (WRSR) Access: User Read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	PWR	EXT	0	TOUT	SFTW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 34-5. Watchdog Reset Status Register (WRSR)

Table 34-9. Watchdog Reset Status Register Field Descriptions

Field	Description
15–5	Reserved—These bits are read as 0
4 PWR	Power-On Reset. Indicates whether the reset was a result of a power-on reset. 0 Reset is not a result of a power-on reset 1 Reset is a result of a power-on reset
3 EXT	External Reset. Indicates whether the reset was a result of an external reset. 0 Reset is not a result of an external reset 1 Reset is a result of an external reset
2	Reserved—This bit reads 0

Table 34-9. Watchdog Reset Status Register Field Descriptions (continued)

Field	Description
1 TOUT	Time-out. Indicates whether the reset was a result of WDOG time-out. 0 Reset is not the result of WDOG time-out 1 Reset is the result of a WDOG time-out
0 SFTW	Software Reset. Indicates whether the reset was a result of a software reset. 0 Reset is not a result of a software reset 1 Reset is a result of a software reset

34.5 Functional Description

This section describes the timing information for the WDOG module.

34.5.1 Timing Specifications

The WDOG provides time-out periods from 0.5 seconds up to 128 seconds with a time resolution of 0.5 seconds. It uses the ipg_clk_32k clock (32.768 kHz frequency clock) as an input to prescalers. The prescalers divide the clock by a fixed value of 16384 (divide by 4 and divide by 4096) to achieve a resolution of 0.5 seconds at a frequency of 2 Hz. The output of the prescaler circuitry is connected to the input of an 8-bit counter to obtain a range of 0.5 to 128 seconds. The user can determine the time-out period by writing a time-out value to the WDOG Time-out field (WT[7:0]) in the WDOG Control Register (WCR).

34.5.2 Watchdog During Reset

A system reset resets all registers (except the WRSR) to their initial default values, and places the counter in the idle state until the WDOG is enabled. The Watchdog Reset Status Register (WRSR) contains the source of the reset event and is not reset by a system reset.

34.5.3 Watchdog After Reset

The following subsections define the WDOG Timer state after reset.

34.5.3.1 Initial Load

The Watchdog Control Register (WCR) field WT[7:0] must have a time-out value written to it before the Watchdog can be enabled. The WDOG is enabled by setting the Watchdog Enable (WDE) bit in the WCR. The time-out value is loaded into the counter after the service sequence is written to the Watchdog Service Register (WSR), or after the WDOG has been enabled. The service sequence is described in Section 34.5.3.3, “Reloading the Counter.” (The counter state machine is shown in Figure 34-6.)

34.5.3.2 Timer Countdown

The counter is activated and begins to count down from its initial programmed value after the WDOG is enabled. If any system errors occur that prevent the software from servicing the Watchdog Service Register (WSR), the timer will time-out when the counter reaches zero. If the WSR is serviced prior to the counter

reaching zero, the WDOG reloads its counter to the time-out value indicated by bits WT[7:0] of the WCR, and it re-starts the countdown. A system reset will reset the counter and place it in the idle state at any time during the countdown. (The counter state machine is shown in Figure 34-6.)

34.5.3.3 Reloading the Counter

The proper service sequence to write a time-out value to the counter begins by writing 0x 5555 followed by 0x AAAA to the WSR. To reload the counter, the writes must take place within the time-out value indicated by bits WT[7:0] of the WCR. Any number of instructions can be executed between the two writes. This service sequence is also used to activate the counter during the initial load. See Section 34.5.3.1, "Initial Load."

If the WSR is not loaded with 0x 5555 prior to writing 0x AAAA to the WSR, the counter is not reloaded. If any value other than 0x AAAA is written to the WSR after 0x 5555, the counter is not reloaded.

34.5.3.4 Time-Out

If the counter reaches zero, the WDOG outputs either a system reset or the $\overline{\text{WDOG}}$ signal, depending on the state of the WRE bit in the WCR. A "0" written to the WRE bit configures the WDOG to generate a system reset. A "1" causes the WDOG to generate the $\overline{\text{WDOG}}$ signal. (The counter state machine is shown in Figure 34-6.)

34.5.4 Generation of Transfer Error on the IP Bus

The WDOG module asserts a transfer error signal (IPS_XFR_ERROR) on the IP bus in the following cases:

- Receiving an IP access to an address that is not implemented
- A write access to the WRSR register that is a read-only register

An error on an unimplemented address is generated only if the input pin resp_sel is low. Otherwise, an error is only asserted on a write access to the WRSR register (a read-only register).

34.5.5 Low-Power and DEBUG Modes

The WDOG module can either continue or suspend the timer operation during low-power modes (WAIT and STOP) and DEBUG mode.

34.5.5.1 Low-Power Mode (WAIT, STOP)

While in low-power mode, the WDOG Timer can be configured for continual operation or its operation can be suspended. If the WDOG low-power Enable (WDZST) bit in the WCR is set to "0", the WDOG continues to operate using the ipg_clk_32k clock (32.768 kHz frequency clock) as its source. If the low-power enable (WDZST) bit is set to "1", then the WDOG operation is suspended during low-power mode. Upon exiting low-power mode, the WDOG returns to the operational mode it was in prior to entering low-power mode.

34.5.5.2 DEBUG Mode

The WDOG Timer can be configured for continual operation, or the operation can be suspended during debug mode. If the WDOG debug enable (WDBG) bit is set to “1” in the Watchdog Control Register (WCR), the WDOG module operation is suspended in debug mode. At this point, the counter is stopped, but register read and write accesses continue to function normally. Also, while in DEBUG mode, the WDE bit can be enabled-disabled directly.

NOTE

If the WDE bit is cleared while in DEBUG mode, it will remain cleared upon exiting DEBUG mode. If the WDE bit is not cleared while in DEBUG mode, the WDOG count will continue from its value before DEBUG mode was entered.

34.5.6 Watchdog Reset Control

The WDOG generated reset signal $\overline{\text{wdog\_rst}}$ can be asserted by a software write to the Software Reset Signal (SRS) bit of the WCR. It can also be generated by the following event:

- WDOG time-out

The $\overline{\text{wdog\_rst}}$ is generated for 0.5 seconds for a time-out, but is deasserted early if a system reset is detected. In case of a software reset, the $\overline{\text{wdog\_rst}}$ is asserted after three clocks of resetting the SRS bit and remains asserted for three IPG clocks (IP global functional clock). If a system reset is asserted in between, it deasserts before three IPG clocks have elapsed.

.

The watchdog-generated reset signal $\overline{\text{wdog\_rst}}$ is an output to the (CCM) for system reset generation.

NOTE

The CCM of the IC generates the system reset signal on assertion of $\overline{\text{wdog\_rst}}$.

34.5.7 $\overline{\text{WDOG}}$ Operation

$\overline{\text{WDOG}}$ can be asserted through software writes to the $\overline{\text{WDOG}}$ Assertion (WDA) bit of the WCR. It can also be generated as a result of a WDOG time-out.

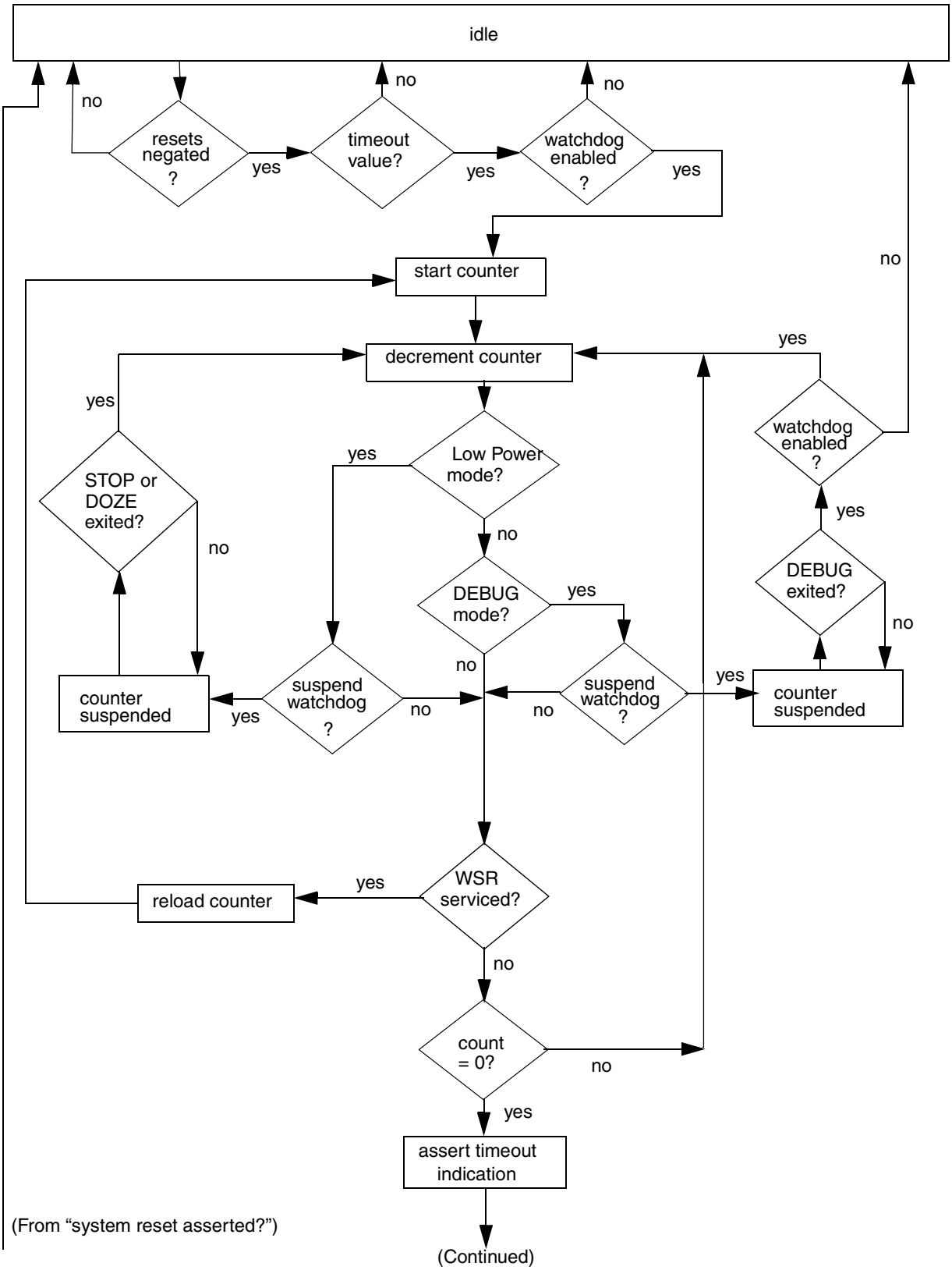
If asserted by a software write to the WDA bit, it remains asserted as long as the WDA bit is “0”. The counter timeout asserts it for 0.5 seconds. Both a system reset and a power-on reset can deassert it in between.

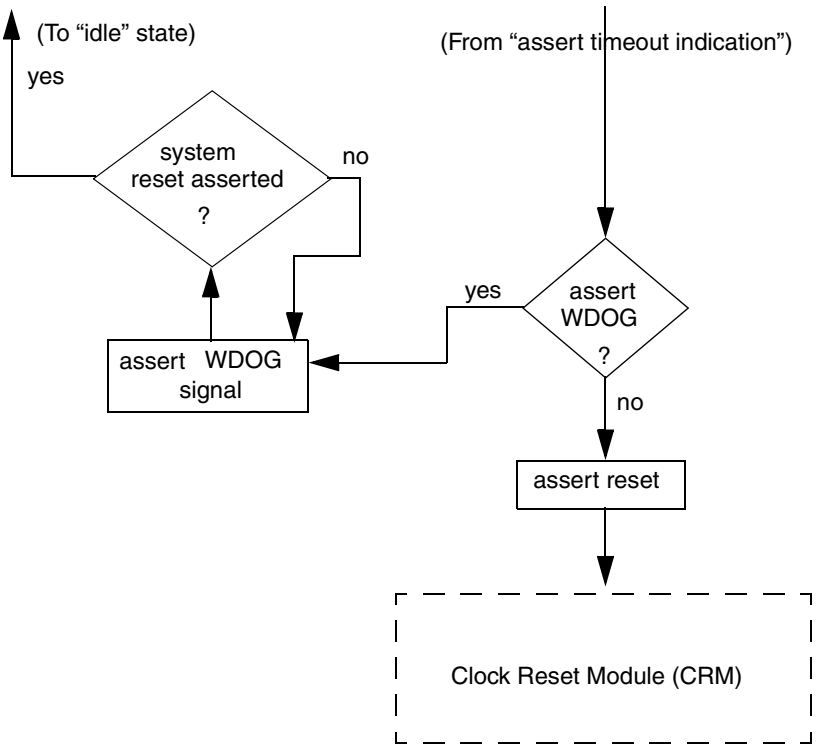
$\overline{\text{WDOG}}$ is applied to the $\overline{\text{WDOG}}$ pin. After reset, the WOE bit in the WCR register controls the direction of this pin.

34.6 Initialization/Application Information

34.6.1 State Machine

The watchdog state machine is shown in Figure 34-6.


idle
no
resets negated ?
yes
no
timeout value?
yes
no
watchdog enabled ?
yes
start counter
no
decrement counter
yes
yes
STOP or DOZE exited?
no
yes
Low Power mode?
no
watchdog enabled ?
yes
DEBUG mode?
yes
DEBUG exited?
no
no
counter suspended
yes
suspend watchdog ?
no
no
suspend watchdog ?
yes
counter suspended
WSR serviced?
yes
reload counter
no
count = 0?
no
yes
assert timeout indication
(From “system reset asserted?”)
(Continued)



NOTE: A system reset will force the state machine to “idle” at any time during countdown.

Figure 34-6. Counter State Machine

Book II, Part 6: System Control Peripherals

Introduction

The transfer of data between modules are controlled by the following system control peripherals:

Chapter 35, “AHB-Lite IP Interface (AIPI) Module,” on page 35-1

Chapter 36, “Multi-Layer AHB Crossbar Switch (MAX),” on page 36-1

Chapter 37, “Direct Memory Access Controller (DMAC),” on page 37-1

AHB-Lite to IPS (AIPI)

The AHB-Lite to IPS (AIPI) interfaces—AIPI1 and AIPI2—facilitate proper communication between the ARM-11 platform and devices that use AHB-Lite with peripherals that are IPS-compliant. Each AIPI in this device handles a separate MCU peripheral bus: AIPI1 interfaces to the MCU Peripheral Bus 1, which supports 16 on-platform peripherals; AIPI2 interfaces to the MCU Peripheral Bus 2, which also supports 32 on-platform peripherals). The AIPI provides all of the necessary bus structure and handshaking signals to allow high-speed communication between these peripheral devices and the internal bus structure of the ARM-11.

Multi-Layer AHB Crossbar Switch (MAX)

The purpose of the MAX is to concurrently support up to five simultaneous connections between master ports and slave ports. The MAX supports a 32-bit address bus width, and a 32-bit data bus width at all master and slave ports.

Direct Memory Access Controller (DMAC)

The Direct Memory Access Controller (DMAC) of i.MX27 device provides 16 DMA channels supporting linear memory, 2D memory and FIFO transfers to provide support for a wide variety of DMA operations.

Chapter 35
AHB-Lite IP Interface (AIPI) Module

This chapter provides an overview of the AHB-Lite to IP bus interface (AIPI) module. The AIPI acts as an interface between the advanced ARM High-performance Bus “Lite” (AHB-Lite) and lower bandwidth peripherals conforming to the *Freescale IP Bus Specification*. There are two AIPI modules in i.MX27—AIPI1 and AIPI2.

The following list summarizes the key features of the AIPI:

- All peripheral read transactions require a minimum of 2 system clocks (R-AHB side) and all write transactions require a minimum of 3 system clocks (R-AHB side).
- The AIPI supports 8-bit, 16-bit and 32-bit IP bus peripherals. (Byte, half word and word reads and write are supported to each.)
- The AIPI supports multi-cycle accesses (16-bit operations to 8-bit peripherals and 32-bit operations to 16-bit and 8-bit peripherals).
- The AIPI supports 31 external IP bus peripherals each with a 4 Kbyte memory map (a slot).

Table 35-1. AHB-Lite To IP Bus V2.0 Interface Operation (Little Endian)

Transfer Size	haddr		IP Bus Size	ips_addr		Active Bus Section (R-AHB to IP Bus)			
	[1]	[0]		[1]	[0]	R-AHB[31:24]	R-AHB[23:16]	R-AHB[15:8]	R-AHB[7:0]
Byte	0	0	8 bit	0	0	—	—	—	ips_data[7:0]
	0	1		0	1	—	—	ips_data[7:0]	—
	1	0		1	0	—	ips_data[7:0]	—	—
	1	1		1	1	ips_data[7:0]	—	—	—
	0	0	16 bit	0	X	—	—	—	ips_data[7:0]
	0	1				—	—	ips_data[15:8]	—
	1	0		1	X	—	ips_data[7:0]	—	—
	1	1				ips_data[15:8]	—	—	—
	0	0	32 bit	X	X	—	—	—	ips_data[7:0]
	0	1				—	—	ips_data[15:8]	—
	1	0		X	X	—	ips_data[23:16]	—	—
	1	1				ips_data[31:24]	—	—	—

Table 35-1. AHB-Lite To IP Bus V2.0 Interface Operation (Little Endian) (continued)

Transfer Size	haddr		IP Bus Size	ips_addr		Active Bus Section (R-AHB to IP Bus)			
	[1]	[0]		[1]	[0]	R-AHB[31:24]	R-AHB[23:16]	R-AHB[15:8]	R-AHB[7:0]
Half Word	0	NA	8 bit	0	0	—	—	—	ips_data[7:0]
					1	—	—	ips_data[7:0]	—
	1			1	0	—	ips_data[7:0]	—	—
					1	ips_data[7:0]	—	—	—
	0		16 bit	0	X	—	—	ips_data[15:8]	ips_data[7:0]
	1			1	X	ips_data[15:8]	ips_data[7:0]	—	—
	0		32 bit	X	X	—	—	ips_data[15:8]	ips_data[7:0]
	1			X	X	ips_data[31:24]	ips_data[23:16]	—	—
Word	NA	NA	8 bit	0	0	—	—	—	ips_data[7:0]
					1	—	—	ips_data[7:0]	—
				1	0	—	ips_data[7:0]	—	—
					1	ips_data[7:0]	—	—	—
			16 bit	0	X	—	—	ips_data[15:8]	ips_data[7:0]
				1	X	ips_data[15:8]	ips_data[7:0]	—	—
			32 bit	X	X	ips_data[31:24]	ips_data[23:16]	ips_data[15:8]	ips_data[7:0]

35.1 Programming Model

There are three registers that reside inside the AIPI. These registers correspond to the first slot (4 Kbyte memory region) of each of the 2 AIPIs in i.MX21 at 0x1000_0000 and 0x1002_0000, respectively. All three registers are 32-bit registers and can only be accessed in supervisor mode. Additionally, these registers can only be read from or written to by a 32-bit access.

Two system clocks are required for read accesses and three system clocks are required for write accesses to the AIPI registers.

CAUTION

Writing to reserved register locations within the 4 Kbyte memory map of the AIPI register space (other than the three AIPI registers) will result in unknown behavior and an abort exception.

Access to Reserved or Unoccupied locations in the AIPI space will result in an abort exception.

35.1.1 Peripheral Size Registers[1:0]

These registers are used to tell the AIPI what size of IP bus peripheral is in each IP bus peripheral location. Peripheral locations that are not occupied should have their corresponding bits in the peripheral size registers (PSRs) programmed to 1 in each register.

The least significant bit in the PSRs is a read-only bit as it governs the AIPI registers themselves. They are set and cleared appropriately to indicate the registers are 32-bit.

PSR0 — Peripheral Size Register 0 — Addr 0x1000_0000 / 0x1002_0000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PSR1 — Peripheral Size Register 1 — Addr 0x1000_0004 / 0x1002_0004

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The PSRs work together to indicate the size of the IP bus peripheral occupying the corresponding location, or to indicate there is no IP bus peripheral occupying the corresponding location. Table 35-2 shows how to program the PSR registers based on the size or availability of an IP bus peripheral.

Table 35-2. PSR 1–0 Data Bus Size Encoding

PSR 1–0	IP Bus Peripheral SIZE [x]
00	8-bit
01	16-bit
10	32-bit
11	Unoccupied

35.1.2 Peripheral Access Register

The peripheral access register (PAR) tells the AIPI whether the IP bus peripheral corresponding to the bit location in this register may be accessed in user mode. If the peripheral may be accessed in supervisor mode only and a user mode access is attempted, an abort is generated and no IP bus activity occurs.

The least significant bit in the PAR is a read-only bit as it governs the AIPI registers themselves. It is set to indicate supervisor access only.

PAR — Peripheral Access Register[1] — Addr 0x1000_0008, 0x1002_0008

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

[1] A “1” indicates the corresponding peripheral is a supervisor access only peripheral. A “0” indicates the decision is left up to the peripheral (the AIPI allows user accesses).

35.2 AIPI1 and AIPI2 Peripheral Widths and PSR Setting

Table 35-5 shows the data bus widths of the peripherals on the AIPI1 and the AIPI2 interfaces. System software should make use of information in the column, “Data Bus Width” to configure the PSR registers, accordingly.

Table 35-3 and Table 35-4 show PSR settings for AIPI1 and AIPI2.

Table 35-5 shows the peripheral sizes for the occupied, reserved and unoccupied locations. All reserved locations must be programmed as 32-bit locations.

Table 35-3. AIPI1 PSR Setting

PSR	Setting
PSR[1]	0xFFFB_FCFB
PSR[0]	0x0004_0304

Table 35-4. AIPI2 PSR Setting

PSR	Setting
PSR[1]	0xFFFF_FFFF
PSR[0]	0x3FFC_0000

Table 35-5. i.MX21 AIPI Peripheral Access Sizes and IP Access Types

Location	Peripheral	PSR[1]	PSR[0]	Data Bus Width	16-bit		8-bit	
					Read	Write	Read	Write
AIPI1								
0	AIPI1 Control	1	0	32-bit	—	—	—	—
1	DMA	1	0	32-bit	Y	Y	Y	Y
2	WDOG	0	1	16-bit	—		Y	Y
3	GPT1	1	0	32-bit	N	N	N	N
4	GPT2	1	0	32-bit	N	N	N	N
5	GPT3	1	0	32-bit	N	N	N	N
6	PWM	1	0	32-bit	N	N	N	N
7	RTC	1	0	32-bit	Y	Y	Y	Y
8	KPP	0	1	16-bit	—		N	N
9	1-Wire	0	1	16-bit	—		N	N
10	UART1	1	0	32-bit	N	Y	N	Y
11	UART2	1	0	32-bit	N	Y	N	Y
12	UART3	1	0	32-bit	N	Y	N	Y
13	UART4	1	0	32-bit	N	Y	N	Y
14	CSPI1	1	0	32-bit	N	N	N	N
15	CSPI2	1	0	32-bit	N	N	N	N
16	SSI1	1	0	32-bit	N	N	N	N
17	SSI2	1	0	32-bit	N	N	N	N
18	I2C	0	1	16-bit	—		Y	Y
19	SDHC1	1	0	32-bit	N	N	N	N
20	SDHC2	1	0	32-bit	N	N	N	N
21	GPIO	1	0	32-bit	N	N	N	N
22	AUDMUX	1	0	32-bit	N	N	N	N
23	CSPI3	1	0	32-bit	N	N	N	N
23-31	Reserved	1	0	32-bit	N	N	N	N
AIPI2								
0	AIPI2	1	0	32-bit	—			
1	LCDC	1	0	32-bit	N	N	N	N
2	SLCDC	1	0	32-bit	N	N	N	N
3	Reserved	1	0	32-bit	N	N	N	N

Table 35-5. i.MX21 AIPI Peripheral Access Sizes and IP Access Types (continued)

Location	Peripheral	PSR[1]	PSR[0]	Data Bus Width	16-bit		8-bit	
					Read	Write	Read	Write
4	USB OTG	1	0	32-bit	N	N	Y	Y
5	USB OTG	1	0	32-bit	N	N	N	N
6	EMMA	1	0	32-bit	N	N	N	N
7	CRM	1	0	32-bit	Y	Y	Y	Y
8	FIRI	1	0	32-bit	Y	Y	Y	Y
9	Reserved	—	—	—	—	—	—	—
10-17	Reserved	1	0	32-bit	N	N	N	N
18-29	Unoccupied	1	1	—	N	N	N	N

35.3 Interface Timing

This section describes AIPI interface timing characteristics.

35.3.1 Read Cycles

Two clock read accesses are possible with the AIPI when the requested access size is equal to or smaller than the size of the targeted IP bus peripheral. If the requested access size is larger than that of the targeted IP bus peripheral (for example, a 32-bit access to a 16 bit peripheral) then a minimum of three clocks are required to complete the access.

35.3.2 Write Cycles

Three clock write accesses are possible with the AIPI when the requested access size is equal to or smaller than the size of the targeted IP bus peripheral. If the requested access size is larger than that of the targeted IP bus peripheral (for example, a 32-bit access to a 16 bit peripheral) then a minimum of four clocks are required to complete the access.

35.3.3 Aborted Cycles

The AIPI follows a standard procedure when a cycle is aborted and the abort is initiated by the AIPI itself or the targeted IP bus peripheral. The AIPI either fails to initiate or immediately terminates any IP bus activity that is ongoing.

There are several conditions that can cause the AIPI to abort the current operation and report an error. The first is the case in which the targeted IP bus peripheral asserts its internal error signal. In this case the AIPI immediately terminates access to the targeted IP bus peripheral. Whether the current IP bus access is a multi-cycle access or a single cycle access has no bearing on the behavior of the AIPI. The AIPI responds identically in both cases.

The second case that can cause an error response to the AHB-Lite is when a user-mode access is attempted to an IP bus peripheral whose corresponding PAR bit indicates it is a supervisor-only peripheral. In this case the AIPI does not initiate any IP bus activity but instead responds immediately by following the abort procedure described above.

The third case that can cause an error response to the AHB-Lite is when an access is attempted to a location at which the PSRs indicate there is no IP bus peripheral. In this case the AIPI does not initiate any IP bus activity but instead responds immediately by following the abort procedure described above.

Chapter 36
Multi-Layer AHB Crossbar Switch (MAX)

This chapter provides an overview of Multi-Layer AHB Crossbar Switch (MAX). The purpose of MAX is to concurrently support up to three simultaneous connections between 6 master ports and 3 slave ports.

36.1 Features

The MAX module has the ability to gain control of all the slave ports and prevent any masters from making accesses to the slave ports. This feature is useful when the user wishes to turn off the clocks to the system and needs to ensure that no bus activity will be interrupted. MAX can put each slave port into a low power park mode so that slave port will not dissipate any power transitioning address, control or data signals when not being actively accessed by a master port.

Each slave port can also support multiple master priority schemes. Each slave port has a hardware input which selects the master priority scheme so the user can dynamically change master priority levels on a slave port by slave port basis.

The MAX allows concurrent transactions to occur from any master port to any slave port. It is possible for three master ports and all slave ports to be in use at the same time as a result of independent master requests. If a slave port is simultaneously requested by more than one master port, arbitration logic will select the higher priority master and grant it ownership of the slave port. All other masters requesting that slave port will stalled until the higher priority master completes its transactions.

36.2 Overview

The MAX module routes bus transactions initiated on the master ports to the appropriate slave ports. There is no provision included to route transactions initiated on the slave ports to other slave ports or to master ports. Simply put, the slave ports do not support the bus request/bus grant protocol, the MAX assumes it is the sole master of each slave port.

Since the MAX does not support the bus request/bus grant protocol, if multiple masters are to be connected to a single master port an external arbiter will need to be used.,Each master and slave port is fully AHB-Lite + AMBA V6 extensions compliant. The ports are not fully AHB compliant because the MAX does not support SPLITs or RETRYs.

NOTE

i.MX27 ARM9 platform implements a 6 master by 3 slave configuration.

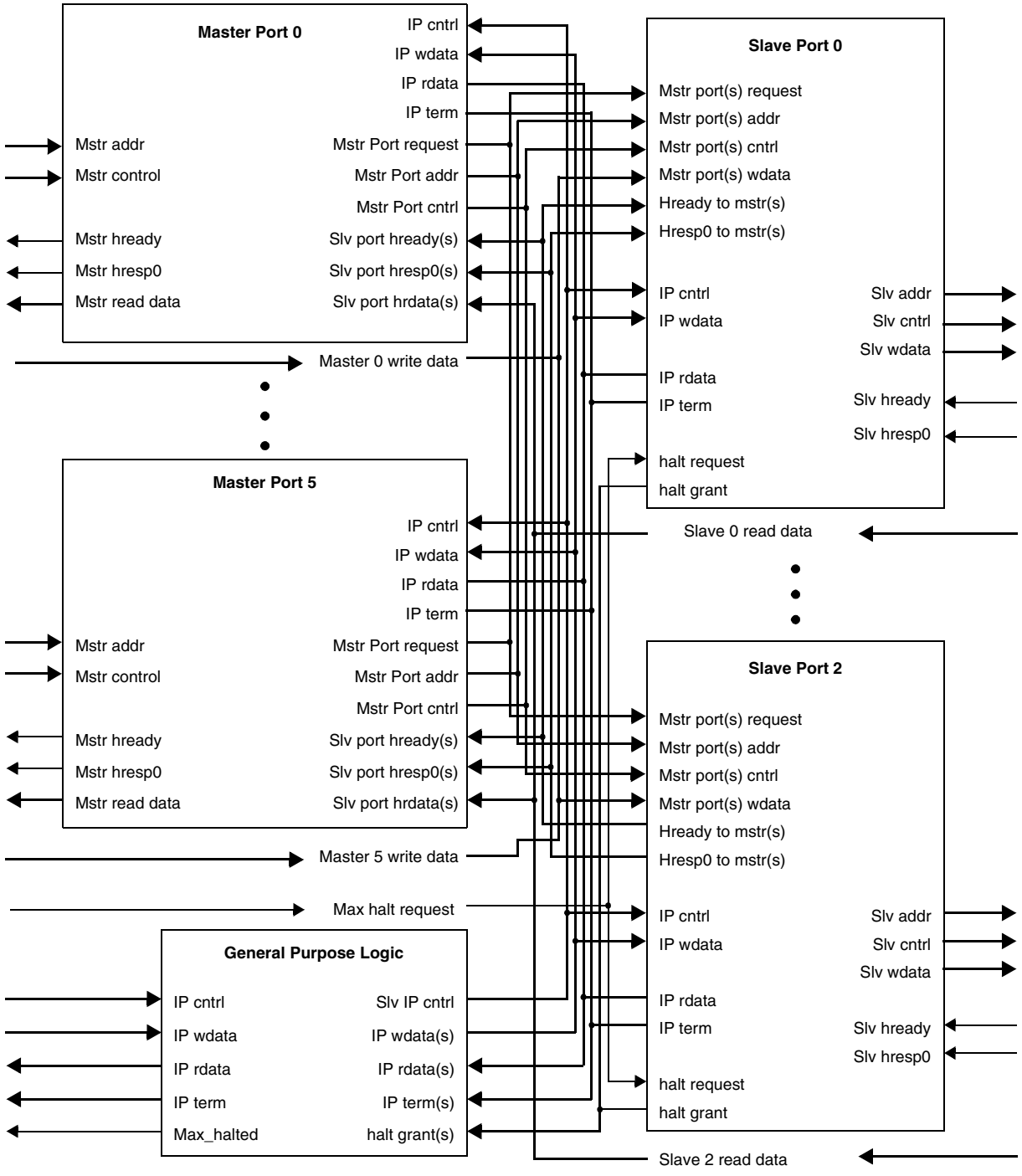


Figure 36-1. MAX Block Diagram

36.3 General Operation

When a master makes an access to the MAX, access will be immediately taken by the MAX. If the targeted slave port of the access is available then the access will be immediately presented on the slave port. It is possible to make single clock (zero wait state) accesses through the MAX. If the targeted slave port of the

access is busy or parked on a different master port the requesting master will simply see wait states inserted (**hready** held negated) until the targeted slave port can service the master's request. Latency in servicing the request will depend on each master's priority level and the responding peripheral's access time. Since MAX appears to be just another slave to the master device, master device will have no knowledge of whether or not it actually owns the slave port it is targeting. When the master does not have control of the slave port it is targeting, it will simply be wait stated.

A master will be given control of the targeted slave port only after a previous access to a different slave port has completed, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when a master has an outstanding request to one slave port that has a long response time, has a pending access to a different slave port, and a lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

Once the master has control of the slave port it is targeting the master will remain in control of that slave port until it gives up the slave port by running an IDLE cycle or by leaving that slave port for its next access. The master could also lose control of the slave port if another higher priority master makes a request to the slave port; however, if the master is running a locked or fixed length burst transfer it will retain control of the slave port until that transfer is completed. Based on the AULB bit in the MGPCR (Master General Purpose Control Register) the master will either retain control of the slave port when doing undefined length incrementing burst transfers or will lose the bus to a higher priority master.

The MAX terminates all master IDLE transfers (as opposed to allowing the termination to come from one of the slave busses). Additionally, when no master is requesting access to a slave port, MAX will drive IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port. When MAX is controlling the slave bus (that is, during low power park or halt mode), **hmaster** field will indicate 4'b0000.

When a slave bus is being IDLEd by the MAX, it can park the slave port on the master port indicated by the PARK bits in the SGPCR or ASGPCR. This can be done in an attempt to save the initial clock of arbitration delay that would otherwise be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into low power park mode in attempt to save power.

36.4 Memory Map and Register Definition

This section provides the register programming information for the MAX registers.

36.4.1 Register Summary

There are four registers that reside in each slave port of the MAX and one register that resides in each master port of the MAX. These registers are IP bus compliant registers. Read and write transfers both require two IP bus clock cycles. The registers can only be read from and written to in privileged mode. Additionally, these registers can only be read from or written to by 32-bit accesses.

The registers are fully decoded and an error response is returned if an unimplemented location is accessed within the MAX.

The slave registers also feature a bit, which when written with a 1, will make it readable, and future write attempts will have no effect on the registers, thus resulting in an error response. Memory map for the MAX program-visible registers is shown in Table 36-1.

36.4.2 Memory Map

Table 36-1 shows the MAX memory map.

Table 36-1. MAX Memory Map

Address	Register	Access	Reset Value	Section/Page
0x1003_F000 (MPR0) 0x1003_F100 (MPR1) 0x1003_F200 (MPR2)	Master Priority Register for Slave Port 0 Master Priority Register for Slave Port 1 Master Priority Register for Slave Port 2	R/W	0x0054_3210	36.4.5/36-6
0x1003_F004 (AMPR0) 0x1003_F104 (AMPR1) 0x1003_F204 (AMPR2)	Alternate Master Priority Register for Slave Port 0 Alternate Master Priority Register for Slave Port 1 Alternate Master Priority Register for Slave Port 2	R/W	0x0000_0000	36.4.6/36-7
0x1003_F010 (SGPCR0) 0x1003_F110 (SGPCR1) 0x1003_F210 (SGPCR2)	General Purpose Control Register for Slave Port 0 General Purpose Control Register for Slave Port 1 General Purpose Control Register for Slave Port 2	R/W	0x0000_0000	36.4.7/36-9
0x1003_F014 (ASGPCR0) 0x1003_F114 (ASGPCR1) 0x1003_F214 (ASGPCR2)	Alternate SGPCR for Slave Port 0 Alternate SGPCR for Slave Port 1 Alternate SGPCR for Slave Port 2	R/W	0x0000_0000	36.4.8/36-11
0x1003_F800 (MGPCR0) – 0x1003_FD00 (MGPCR5)	General Purpose Control Register for Master Port 0 – General Purpose Control Register for Master Port 5	R/W	0x0054_3210	36.4.5/36-6

36.4.3 Register Summary

Figure 36-2 shows the key to the register fields, and Table 36-2 shows the register figure conventions.

Figure 36-2. Key to Register Fields

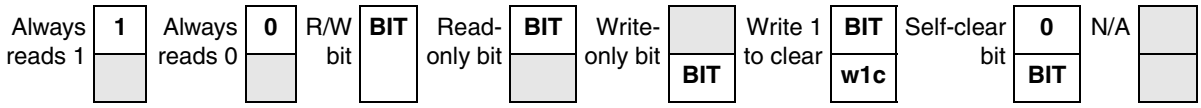


Table 36-2. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.

Table 36-2. Register Figure Conventions (continued)

Convention	Description
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 36-3. MAX Detailed Register Summary

Name		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
0x1003_F000 (MPR0) 0x1003_F100 (MPR1) 0x1003_F200 (MPR2)	R	0	0	0	0	0	0	0	0	0	MSTR_5			0	MSTR_4		
	W																
	R	0	MSTR_3			0	MSTR_2			0	MSTR_1			0	MSTR_0		
	W																
0x1003_F004 (AMPR0) 0x1003_F104 (AMPR1) 0x1003_F204 (AMPR2)	R	0	0	0	0	0	0	0	0	0	MSTR_5			0	MSTR_4		
	W																
	R	0	MSTR_3			0	MSTR_2			0	MSTR_1			0	MSTR_0		
	W																
0x1003_F010 (SGPCR0) 0x1003_F110 (SGPCR1) 0x1003_F210 (SGPCR2)	R	RO	HLP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	ARB		0	0	PCTL		0	PARK		
	W																
0x1003_F014 (ASGPCR0) 0x1003_F114 (ASGPCR1) 0x1003_F214 (ASGPCR2)	R	0	HLP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	ARB		0	0	PCTL		0	PARK		
	W																
0x1003_F800 (MGPCR0) – 0x1003_FD00 (MGPCR5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	AULB		
	W																

36.4.4 MAX Register Descriptions

This section contains the detailed register descriptions for the MAX registers.

36.4.5 Master Priority Registers (MPR0–MPR2)

The Master Priority Register (MPR) sets the priority of each master port on a per slave port basis and resides in each slave port.

The Master Priority Register can only be accessed in supervisor mode with 32-bit accesses. Once the RO (Read Only) bit has been set in the Slave General Purpose Control Register the Master Priority Register can only be read from, attempts to write to it will have no effect on the MPR and result in an error response.

Additionally, no two available master ports may be programmed with the same priority level. Attempts to program two or more available masters with the same priority level will result in an error response and the MPR will not be updated.

0x1003_F000 (MPR0)
0x1003_F100 (MPR1)
0x1003_F200 (MPR2)

Access: Supervisor Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	MSTR_5			0	MSTR_4		
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MSTR_3			0	MSTR_2			0	MSTR_1			0	MSTR_0		
W																
Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

Table 36-4. Master Priority Register (MPR0–MPR2)

Table 36-5. Master Priority Register Field Descriptions

Field	Description
31–23	Reserved. They are read as zero and should be written with zero for upward compatibility.
22–20 MSTR_5	Master 5 Priority. These bits set the arbitration priority for master port 5 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
19	Reserved. They are read as zero and should be written with zero for upward compatibility.
18–16 MSTR_4	Master 4 Priority. These bits set the arbitration priority for master port 4 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
15	Reserved. They are read as zero and should be written with zero for upward compatibility.

Table 36-5. Master Priority Register Field Descriptions (continued)

Field	Description
14–12 MSTR_3	Master 3 Priority. These bits set the arbitration priority for master port 3 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
11	Reserved. They are read as zero and should be written with zero for upward compatibility.
10–8 MSTR_2	Master 2 Priority. These bits set the arbitration priority for master port 2 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
7	Reserved. They are read as zero and should be written with zero for upward compatibility.
6–4 MSTR_1	Master 1 Priority. These bits set the arbitration priority for master port 1 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
3	Reserved. They are read as zero and should be written with zero for upward compatibility.
2–0 MSTR_0	Master 0 Priority. These bits set the arbitration priority for master port 0 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.

36.4.6 Alternate Master Priority Register for Slave Port 0–2 (AMPR0–2)

The Alternate Master Priority register AMPR sets alternate priorities of each master port on a per slave port basis. AMPR has identical function like MPR. AMPR purpose is to allow the user to set up an alternate set of priorities in the event they want to do some sort of context switching. A hardware input to the MAX controls (on a slave port by slave port basis) whether or not the slave port uses MPR or AMPR. Refer Table 36-5 for AMPR bit descriptions as they are identical to MPR. AMPR can only be accessed in privileged mode with 32-bit accesses. Once the RO (Read Only) bit is set in the General Purpose Control Register, AMPR can only be read from, attempts to write to it will result in an error response. Additionally, no two available master ports may be programmed with the same priority level. Attempts to program two or more available masters with the same priority level will result in an error response and the AMPR will not be updated.

0x1003_F004 (AMPR0)
0x1003_F104 (AMPR1)
0x1003_F204 (AMPR2)

Access: Supervisor Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	MSTR_5			0	MSTR_4		
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MSTR_3			0	MSTR_2			0	MSTR_1			0	MSTR_0		
W																
Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

Table 36-6. Alternate Priority Register (MPR0–MPR2)

Table 36-7. Alternate Priority Register Field Descriptions

Field	Description
31–23	Reserved. They are read as zero and should be written with zero for upward compatibility.
22–20 MSTR_5	Master 5 Priority. These bits set the arbitration priority for master port 5 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
19	Reserved. They are read as zero and should be written with zero for upward compatibility.
18–16 MSTR_4	Master 4 Priority. These bits set the arbitration priority for master port 4 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
15	Reserved. They are read as zero and should be written with zero for upward compatibility.
14–12 MSTR_3	Master 3 Priority. These bits set the arbitration priority for master port 3 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
11	Reserved. They are read as zero and should be written with zero for upward compatibility.
10–8 MSTR_2	Master 2 Priority. These bits set the arbitration priority for master port 2 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
7	Reserved. They are read as zero and should be written with zero for upward compatibility.
6–4 MSTR_1	Master 1 Priority. These bits set the arbitration priority for master port 1 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.

Table 36-7. Alternate Priority Register Field Descriptions (continued)

Field	Description
3	Reserved. They are read as zero and should be written with zero for upward compatibility.
2–0 MSTR_0	Master 0 Priority. These bits set the arbitration priority for master port 0 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.

36.4.7 General Purpose Control Register for Slave Port 0–2 (SGPCR0–2)

The Slave General Purpose Control Register (SGPCR) controls several features of each slave port.

The Read Only (RO) bit will prevent any registers associated with this slave port from being written to once set. This bit may be written with 0 as many times as the user desires, but once it is written to a 1 only a reset condition will allow it to be written again.

The Halt Low Priority (HLP) bit will set the priority of the max_halt_request input to the lowest possible priority for initial arbitration of the slave ports. By default it is the highest priority.

NOTE

Setting this bit will not effect the max_halt_request from attaining highest priority once it has control of the slave ports.

The PCTL bits determine how the slave port will park when no master is actively making a request. The available options are to park on the master defined by the PARK bits, park on the last master to use the slave port, or go into a low power park mode which will force all the outputs of the slave port to inactive states when no master is requesting an access. The low power park feature can result in an overall power savings if a the slave port is not saturated; however, it will force an extra clock of latency whenever any master tries to access it when it is not in use because it will not be parked on any master.

The PARK bits determine which master the slave will park on when no master is making an active request and the max_halt_request input is negated.

CAUTION

Only select master ports that are actually present in the i.MX27. If you program the PARK bits to a master not present in the i.MX27 undefined behavior will result.

See Figure 36-3 for an illustration of valid bits in the SGPCR, and Table 36-8 for its field descriptions.

0x1003_F010 (SGPCR0)
0x1003_F110 (SGPCR1)
0x1003_F210 (SGPCR2)

Access: Supervisor Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RO[1]	HLP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	ARB		0	0	PCTL		0	PARK		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-3. Slave General Purpose Control Register (SGPCR0-2)

[1] Once this bit is written to a 1, then only hardware reset will return it to a 0.

Table 36-8. Slave General Purpose Control Register Field Descriptions

Field	Description
31 RO	Read Only. This bit is used to force all of a slave port's registers to be read only. Once written to 1 it can only be cleared by hardware reset. This bit is initialized by hardware reset. 0 All of this slave port's registers can be written. 1 All of this slave port's registers are read only and cannot be written (attempted writes have no effect and result in an error response).
30 HLP	Halt Low Priority**.** This bit is used to set the initial arbitration priority of the max_halt_request input. This bit is initialized by hardware reset. 0 The max_halt_request input has the highest priority for arbitration on this slave port 1 The max_halt_request input has the lowest initial priority for arbitration on this slave port.
29–10	Reserved**.** They read as zero and should be written with zero for upward compatibility.
9–8 ARB	Arbitration Mode. These bits are used to select the arbitration policy for the slave port. These bits are initialized by hardware reset. 00 Fixed Priority. 01 Round Robin (rotating) Priority 10 Reserved 11 Reserved
7–6	Reserved. They read as zero and should be written with zero for upward compatibility.
5–4 PCTL	Parking Control.These bits determine the parking control used by this slave port. These bits are initialized by hardware reset. 00 When no master is making a request the arbiter will park the slave port on the master port defined by the PARK bit field. 01 When no master is making a request the arbiter will park the slave port on the last master to be in control of the slave port. 10 When no master is making a request the arbiter will park the slave port on no master and will drive all outputs to a constant safe state. 11 Reserved

Table 36-8. Slave General Purpose Control Register Field Descriptions (continued)

Field	Description
3	Reserved. They read as zero and should be written with zero for upward compatibility.
2–0 PARK	Park. These bits are used to determine which master port this slave port parks on when no masters are actively making requests and the PCTL bits are set to 00. These bits are initialized by hardware reset. 000 Park on Master Port 0 001 Park on Master Port 1 010 Park on Master Port 2 011 Park on Master Port 3 100 Park on Master Port 4 101 Park on Master Port 5 110 Reserved 111 Reserved

NOTE: Reserved bits are for future expansion. It is read as zero and should be written with zero for upward compatibility

36.4.8 Alternate SGPCR for Slave Port 0–2 (ASGPCR0–2)

The ASGPCR has identical function as the SGPCR with the notable exception that it lacks the RO (Read Only) bit contained in the SGPCR. ASGPCR purpose is same as AMPR, to allow the user to set up an alternate set of general control fields in the event they want to do some sort of context switching. A hardware input to the MAX controls (on a slave port by slave port basis) whether or not the slave port uses the SGPCR or ASGPCR. Refer Table 36-8 for descriptions of bit fields in ASGPCR as they are identical except for the RO bit. ASGPCR can only be accessed in privileged mode with 32-bit accesses. Once the RO (Read Only) bit has been set in the SGPCR the ASGPCR can only be read from, attempts to write to it will have no effect on the ASGPCR and result in an error response.

NOTE

ASGPCR does not contain a RO (Read Only) bit. RO bit in SGPCR has control over the ASGPCR's ability to be written.

0x1003_F014 (ASGPCR0)
0x1003_F114 (ASGPCR1)
0x1003_F214 (ASGPCR2)

Access: Supervisor Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	ARB		0	0	PCTL		0	PARK		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-4. Alternate SGPCR for Slave Port0–2 (ASGPCR0-2)

Table 36-9. Alternate Slave General Purpose Control Register Field Descriptions

Field	Description
31	Reserved**.** They read as zero and should be written with zero for upward compatibility.
30 HLP	Halt Low Priority**.** This bit is used to set the initial arbitration priority of the max_halt_request input. This bit is initialized by hardware reset. 0 The max_halt_request input has the highest priority for arbitration on this slave port 1 The max_halt_request input has the lowest initial priority for arbitration on this slave port.
29–10	Reserved**.** They read as zero and should be written with zero for upward compatibility.
9–8 ARB	Arbitration Mode. These bits are used to select the arbitration policy for the slave port. These bits are initialized by hardware reset. 00 Fixed Priority. 01 Round Robin (rotating) Priority 10 Reserved 11 Reserved
7–6	Reserved. They read as zero and should be written with zero for upward compatibility.
5–4 PCTL	Parking Control.These bits determine the parking control used by this slave port. These bits are initialized by hardware reset. 00 When no master is making a request the arbiter will park the slave port on the master port defined by the PARK bit field. 01 When no master is making a request the arbiter will park the slave port on the last master to be in control of the slave port. 10 When no master is making a request the arbiter will park the slave port on no master and will drive all outputs to a constant safe state. 11 Reserved

Table 36-9. Alternate Slave General Purpose Control Register Field Descriptions (continued)

Field	Description
3	Reserved. They read as zero and should be written with zero for upward compatibility.
2–0 PARK	Park. These bits are used to determine which master port this slave port parks on when no masters are actively making requests and the PCTL bits are set to 00. These bits are initialized by hardware reset. 000 Park on Master Port 0 001 Park on Master Port 1 010 Park on Master Port 2 011 Park on Master Port 3 100 Park on Master Port 4 101 Park on Master Port 5 110 Reserved 111 Reserved

36.4.8.1 General Purpose Control Register for Master Port 0–5 (MGPCR0–5)

The Master General Purpose Control Register (MGPCR) presently controls only whether or not the master's undefined length burst accesses will be allowed to complete uninterrupted or whether they can be broken by requests from higher priority masters.

The AULB (Arbitrate on Undefined Length Bursts) bit field determines whether (and when) or not the MAX will arbitrate away the slave port the master owns when the master is performing undefined length burst accesses.

See Figure 36-5 for an illustration of valid bits in the MGPCR, and Table 36-10 for its field descriptions.

0x1003_F800 (MGPCR0)
0x1003_F900 (MGPCR1)
0x1003_FA00 (MGPCR2)
0x1003_FB00 (MGPCR3)
0x1003_FC00 (MGPCR4)
0x1003_FD00 (MGPCR5)

Access: Supervisor Read/Write

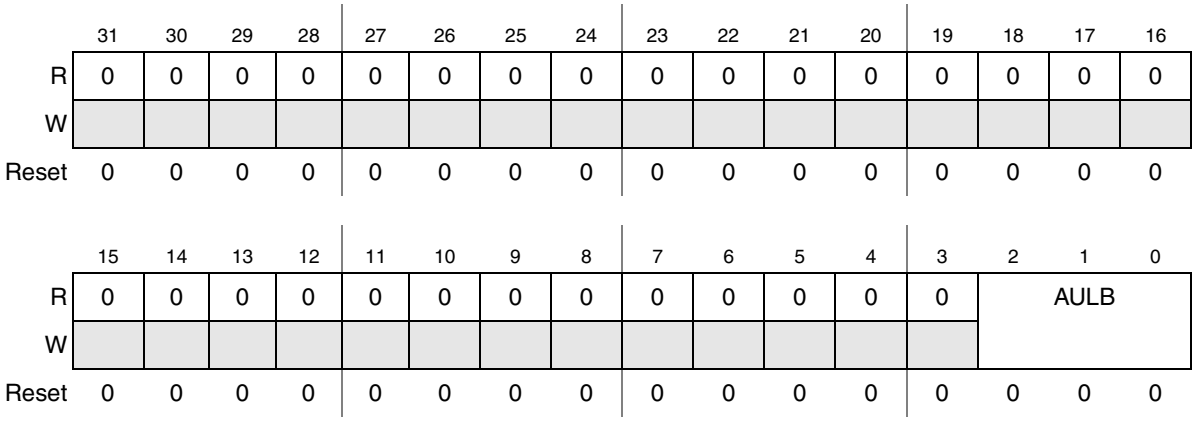


Figure 36-5. Master General Purpose Control Registers (MGPCR0-5)

Table 36-10. Master General Purpose Control Register Field Descriptions

Field	Description
31–3	Reserved. They read as zero and should be written with zero for upward compatibility.
2–0 AULB	Arbitrate on Undefined Length Bursts. These bits are used to select the arbitration policy during undefined length bursts by this master. These bits are initialized by hardware reset. 000 No arbitration will be allowed during an undefined length burst. 001 Arbitration will be allowed at any time during an undefined length burst. 010 Arbitration will be allowed after four beats of an undefined length burst. 011 Arbitration will be allowed after eight beats of an undefined length burst. 100 Arbitration will be allowed after 16 beats of an undefined length burst. 101 Reserved 110 Reserved 111 Reserved

36.5 Function

This section describes the functionality of the MAX in greater detail.

36.5.1 Arbitration

MAX supports two arbitration schemes; a simple fixed-priority comparison algorithm, and a simple round-robin fairness algorithm. Arbitration schemes are independently programmable for each slave port.

36.5.1.1 Arbitration During Undefined Length Bursts

Arbitration points during an undefined length burst are defined by the current master's MGPCR AULB field setting. When a defined length is imposed on the burst via the AULB bits the undefined length burst will be treated as a single or series of single back to back fixed length burst accesses.

Example: A master runs an undefined length burst and AULB bits in MGPCR indicate arbitration will occur after fourth beat of the burst. Master runs two sequential beats and then starts what will be an 12 beat undefined length burst access to a new address within the same slave port region as the previous access. MAX will not allow an arbitration point until the fourth overall access (second beat of second burst). At that point all remaining accesses will be open for arbitration until the master loses control of the slave port.

Assume master loses control of the slave port after fifth beat of the second burst. Once the master regains control of the slave port, no arbitration point will be available until master has run four more beats of its burst. After fourth beat of the (now continued) burst (9th beat of second burst from master's perspective) is taken, all beats of the burst will once again be open for arbitration until the master loses control of the slave port. Assume the master again loses control of the slave port on the fifth beat of the third (now continued) burst (10th beat of second burst from master's perspective). Once the master regains control of the slave port it will be allowed to complete its final two beats of its burst without facing arbitration.

NOTE

Fixed length burst accesses will not be affected by AULB bits. All fixed length burst accesses will lock out arbitration until the last beat of the fixed length burst.

36.5.1.2 Fixed Priority Operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the MPR and AMPR. If two masters both request access to a slave port the master with the highest priority in the selected priority register will gain control over the slave port. Any time a master makes a request to a slave port the slave port checks to see if the new requesting master's priority level is higher than that of the master that currently has control over the slave port (unless the slave port is in a parked state). The slave port does an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the current master that has control of the slave port, the new requesting master will be granted control over the slave port at the next clock edge. The exception to this rule is if the master that currently has control over the slave port is running a fixed length burst transfer or a locked transfer. In this case the new requesting master will have to wait until the end of the burst transfer or locked transfer before it will be granted control of the slave port. If the master is running an undefined length burst transfer, the new requesting master must wait until an arbitration point for the undefined length burst transfer before it will be granted control of the slave port. Arbitration points for an undefined length burst are defined in MGPCR for each master.

If the new requesting master's priority level is lower than that of the current master that has control of the slave port, the new requesting master will be forced to wait until the current master has control of the slave port either runs an IDLE cycle or runs a non IDLE cycle to a location other than the current slave port.

36.5.1.3 Round-Robin Priority Operation

When operating in round-robin mode, each master is assigned a relative priority based on the master number. This relative priority is compared to the ID of the last master to perform a transfer on the slave bus. The highest priority requesting master will become owner of the slave bus as the next transfer boundary (accounting for locked and fixed-length burst transfers). Priority is based on how far ahead the ID of the requesting master is to the ID of the last master (ID is defined by master port number, not hmaster field). Once granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line will be granted access to the slave port at the next assertion of sX_hready, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the MAX is implemented with master ports 0, 1, 2, 3, 4 and 5. If the last master of the slave port was master 1, and master 0, 4 and 5 make simultaneous requests, (master ports 2 and 3 make no requests), they will be serviced in the order 4, 5 and then 0.

Parking may still be used in a round-robin mode, but will not affect the round-robin pointer unless the parked master actually performs a transfer. Hand-off will occur to the next master in line after one cycle of arbitration. If the slave port is put into low power park mode the round-robin pointer will be reset to point at master port 0, giving it the highest priority.

36.5.2 Priority Assignment

Each master port needs to be assigned a unique 3 bit priority level. If an attempt is made to program multiple master ports with the same priority level within a register (MPR or AMPR) the MAX will respond with an error and the registers will not be updated.

36.5.2.1 Context Switching

The MAX has a hardware input per slave port (sX_ampr_sel) which is used to select which registers the master priority levels and general purpose control bits will be taken from. When sX_ampr_sel is 0, MPR and SGPCR will be selected and when sX_ampr_sel is 1, AMPR and ASGPCR will be selected. This hardware input is useful for context switching so the user does not have to rewrite the MPR or SGPCR if a particular slave port would temporarily benefit from modifying the master priority levels or functions affected by the bits in the SGPCR.

36.5.3 Master Port Functionality

36.5.3.1 General

Each master port consists of two decoders, a capture unit, a register slice, a mux and a small state machine. The first decoder is used to decode haddr and control signals coming directly from the master, telling the state machine where the master's next access will be and if it is in fact a legal access. The second decoder gets its input from the capture unit, so it may be looking directly at the signals coming from the master or it may be looking at captured signals coming from the master, depending entirely on the targeted slave port state. The second decoder is then used to generate the access requests that go to the slave ports.

Capture unit is used to capture the address and control information coming from the master in the event that the targeted slave port cannot immediately service the master. Capture unit is controlled by outputs from the state machine which tell it to either pass through the original master signals or the captured signals

Register slice contains the registers associated with the specific master port. The registers have a quasi-IP bus interface at this level for reads and writes and the outputs feed directly into the state machine. The mux is used simply to select which slave's read data is sent back to the master. The mux is controlled by the state machine.

The state machine controls all aspects of the master port. It knows which slave port the master wants to make a request to and controls when that request is made. It also has knowledge of each slave port, knowing whether or not the slave port is ready to accept an access from the master port. This will determine whether or not the master may immediately have its request taken by the slave port or whether the master port will have to capture the master's request and queue it at the slave port boundary. A block diagram of the master port can be seen in Figure 36-6.

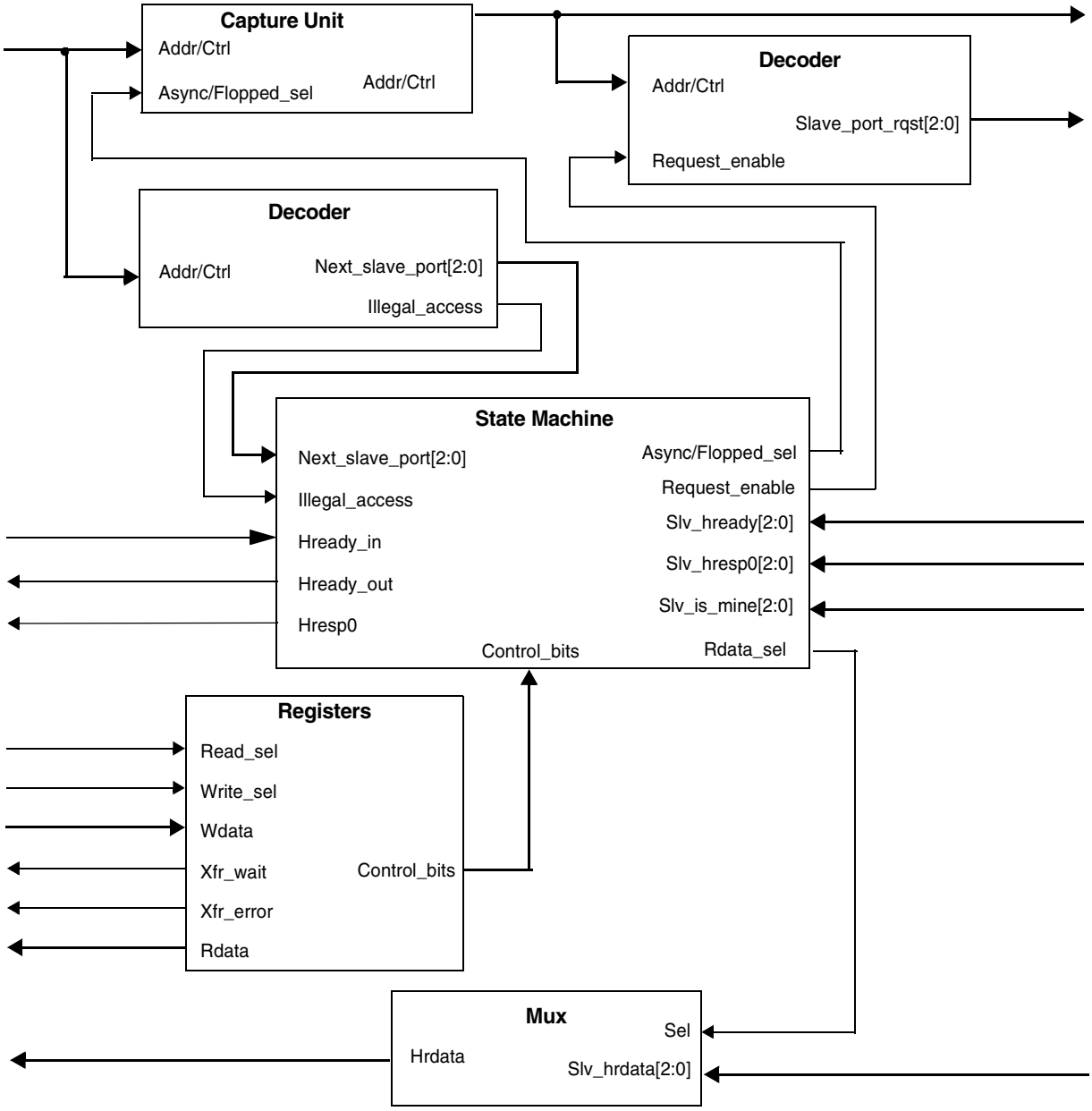


Figure 36-6. MAX Master Port Block Diagram

36.5.3.2 Decoders

Decoders are very simple as they ensure an access request is allowed to be made and that the slave port targeted is actually present in the design. The decoders feeding the state machine are always enabled. The decoders that select the slave are enabled only when the master port controlling state machine wants to make a request to a slave port. This is necessary so that if a master port is making an access to a slave port and is being wait stated, and its next access is to a different slave port, the request to the second slave port can be held off until the access to the first slave port is terminated. The decoders also output a “hole

decode" or illegal access signal which tells the state machine that the master is trying to access a slave port that does not exist.

36.5.3.3 Capture Unit

The capture unit simply captures the state of the master's address and control signals if the MAX cannot immediately pass the master's request through to the proper slave port. The capture unit consists of a set of flops and a mux which selects either the asynchronous path from address and control or the flopped (captured) address and control information.

36.5.3.4 Registers

Registers in the master port are only those registers associated with this particular master port. The read and write interface for the registers is a quasi-IP bus interface. It is not a full IP bus interface at this level because not all the IP bus signals are routed this deep in the design. There is a register control block at the same level of the master port and slave port instantiations in the MAX. This control block ensures that all accesses are 32-bit privileged accesses before passing them on to the master ports. The register outputs are connected directly to the state machine.

36.5.3.5 State Machine

36.5.3.5.1 States

The master side state machine's main function is to monitor the master port activities. There are six states: busy, idle, stalled, steady state, first cycle error response and second cycle error response.

The busy state is used when the master runs a BUSY cycle to the master port. The master port maintains its request to the slave port if it currently owns the slave port; however, if it loses control of the slave port it will no longer maintain its request. If the master port loses control of the slave port it will not be allowed to make another request to the slave port until it runs a NSEQ or SEQ cycle.

The idle state is used when the master runs a valid IDLE cycle to the master port. The master port makes no requests to the slave ports (disables the slave port decoder) and terminates the IDLE cycle.

The stalled state is used when the master makes a request to a slave port that is not immediately ready to receive the request. In this case the state machine will direct the capture unit to send out the captured address and control signals and will enable the slave port decoder to indicate a pending request to the appropriate slave port.

The steady state is used when the master port and slave port are in fully asynchronous mode, making the MAX completely transparent in the access. The state machine selects the appropriate slave's hresp0, hready and hrdata to pass back to the master.

The first cycle error response and second cycle error response states are self explanatory. The MAX will respond with an error response to the master if the master tries to access an unimplemented memory location through the MAX (that is, a slave port that does not exist).

36.5.3.5.2 Slave Swapping

The design of the master side state machine is fairly straight forward. The one real decision to be made is how to handle the master moving from one slave port access to another slave port access. The approach that was taken was to minimize or eliminate, when possible, any “bubbles” that would get inserted into the access due to switching slave ports. The state machine will not allow the master to request access to another slave port until the current access being made is terminated. This prevents a single master from owning two slave ports at the same time (the slave port it is currently accessing and the slave port it wishes to access next).

The state machine also maintains watch on the slave port the master is accessing as well as the slave port the master wishes to switch to. If the new slave port is parked on the master then the master will be able to make the switch without incurring any delays. The termination of the current access will also act as the launch of the new access on the new slave port. If the new slave port is not parked on the master then the master will incur a minimum one clock delay before it can launch its access on the new slave port.

This is the same for switching from the busy or idle state to actively accessing a slave port. If the slave port is parked on the master the state machine will go to the steady state and the access will begin immediately. If the slave port is not parked on the master (serving another master, parked on another master or in low power park mode) then the state machine will transition to the stalled state and at least a one clock penalty will be paid.

36.5.4 Slave Port Functionality

36.5.4.1 General

Each slave port consists of a register slice, a bank of muxes and a state machine. The register slice contains the registers associated with the specific slave port. The registers have a quasi-IP bus interface at this level for reads and writes and the outputs feed directly into the state machine. A block diagram of a slave port can be seen in Figure 36-7.

36.5.4.2 Muxes

Figure 36-7 shows only one block for all the muxes. In reality that block instantiates many 6 to 1 muxes, one for each master-to-slave signal in fact. All the muxes are designed in an AND - OR fashion, so that if no master is selected the output of the muxes will be zero. (This is an important feature for low power park mode.) The muxes also have an override signal which is used by the slave port to asynchronously force IDLE cycles onto the slave bus. When the state machine forces an IDLE cycle it zeros out htrans and hmastlock, making sure the slave bus sees a valid IDLE cycle being run by the MAX.

The mux controlling htrans also contains an additional control signal from the state machine so that a NSEQ transaction can be forced. This is done at any time when the slave port switches masters to ensure that no IDLE-SEQ, BUSY-SEQ or NSEQ-SEQ transactions are seen on the slave port when they shouldn’t be. If the state machine indicates to run both IDLE and NSEQ cycle, the IDLE directive will have priority.

NOTE

IDLE-SEQ is in fact an illegal access, but a possible scenario given the multi-master environment in the MAX unless corrected by the MAX.

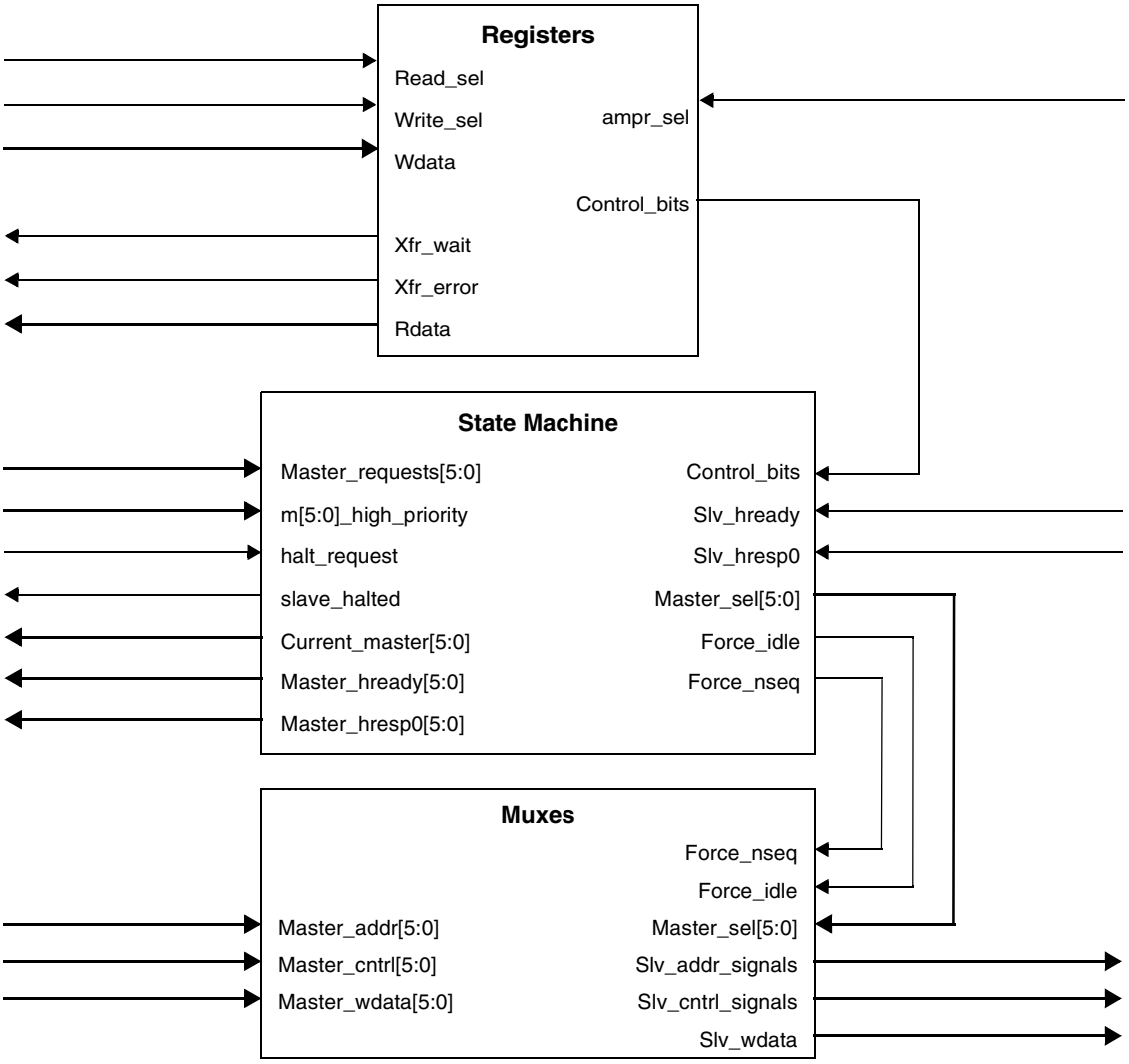


Figure 36-7. MAX Slave Port Block Diagram

36.5.4.3 Registers

There is a register control block at the same level of the master port and slave port instantiations in the MAX. This control block ensures that all accesses are 32-bit privileged accesses before passing them on to the master and slave ports. The registers in the slave port are only those registers associated with this particular slave port. The read and write interface for the registers is a quasi-IP bus interface. It is not a full IP bus interface at this level because not all the IP bus signals are routed this deep in the design. The register outputs are connected directly to the slave state machine with the sX_ampr_sel input determining which priority register values, halt priority value, arbitration algorithm and parking control bits are passed to the state machine. The registers can be read from an unlimited number of times. The registers can only

be written to as long as the RO bit is written to 0 in the SGPCR, once it is written to a 1 only a hardware reset will allow the registers to be written again.

36.5.4.4 State Machine

36.5.4.4.1 States

At the heart of the slave port is the state machine. The state machine is simplicity itself, requiring only four states—steady state, transition state, transition hold state and hold stat**e**. Either the slave port is owned by the same master it was in the last clock cycle (either by active use or by parking), it is transitioning to a new master (either for active use or parking), it is transitioning to a new master during wait states or it is being held on the same master pending a transition to a new master.

36.5.4.4.2 Arbitration

The real work of the state machine is to determine which master port will be in control of the slave port in the next clock cycle. Each master is programmed with a fixed 3 bit priority level. The MAX uses these bits in determining priority levels when programmed for fixed priority mode of operation. Arbitration always occurs on a clock edge, but only occurs on edges when a change in mastership will not violate AHB-Lite protocols. Valid arbitrations points include any clock cycle in which **sX_hready** is asserted (provide the master is not performing a burst or locked cycle) and any wait state in which the master owning the bus indicates a transfer type of IDLE (provided the master is not performing a locked cycle). Since arbitration can occur on every clock cycle the slave port masks off all master requests if the current master is performing a locked transfer or a protected burst transfer, guaranteeing that no matter how low its priority level it will be allowed to finish its locked or protected portion of a burst sequence.

36.5.4.4.3 Master Hand-Off

The only times slave port will switch masters when programmed for fixed priority mode of operation is when a higher priority master makes a request or when the current master is the highest priority and it gives up the slave port by either running and IDLE cycle to the slave port or running a valid access to a location other than the slave port.

If the current master loses control of the slave port because a higher priority master takes it away the slave port will not incur any wasted cycles. The current master will get its current cycle terminated by the slave port at the same time the new master’s address and control information will be recognized by the slave port. This will look like a seamless transition on the slave port.

If the current master is being wait stated when the higher priority master makes its request, then the current master will be allowed to make one more transaction on the slave bus before giving it up to the new master. Figure 36-8 illustrates the effect of a higher priority master taking control of the bus when the slave port is programmed for a fixed priority mode of operation.

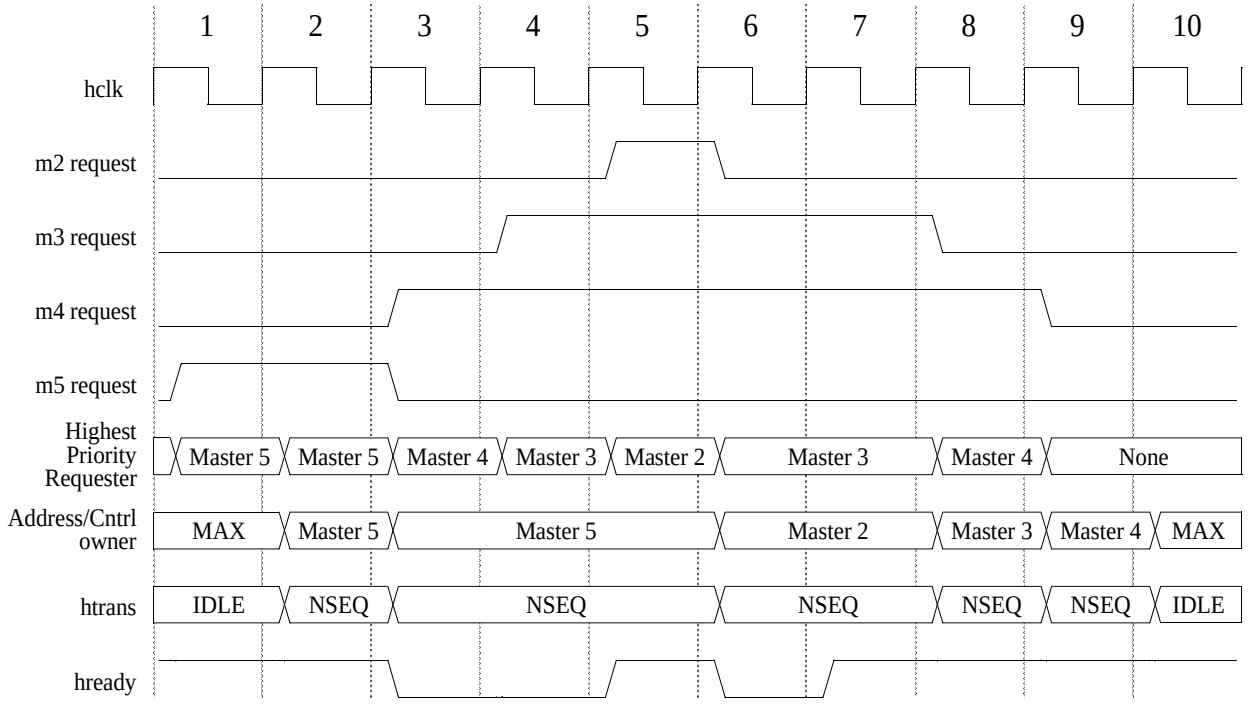


Figure 36-8. Low to High Priority Mastership Change

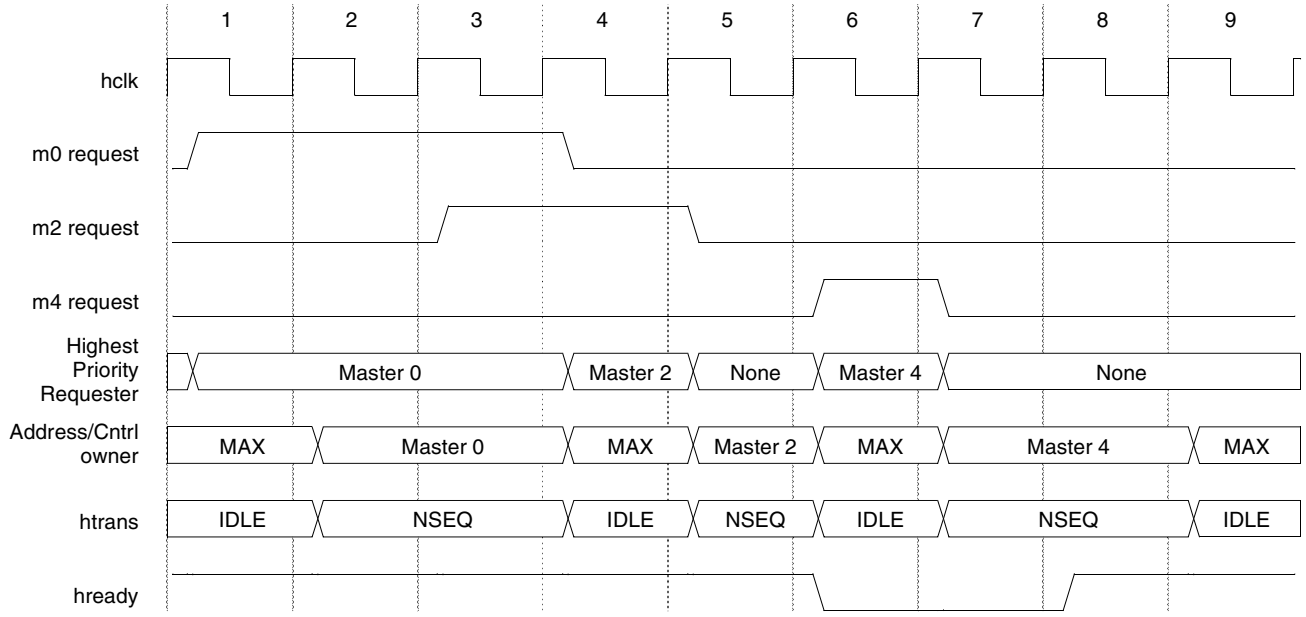


Figure 36-9. High to Low Priority Mastership Change

If current master is the highest priority master and it gives up the slave port by running an IDLE cycle or a valid cycle to another location other than the slave port, the next highest priority master will gain control of the slave port. If the current access incurs any wait states then the transition will be seamless and no bandwidth will be lost; however, if the current transaction is terminated without wait states, then one IDLE cycle will be forced onto the slave bus by the MAX before the new master will be able to take control of

the slave port. If no other master is requesting the bus then IDLE cycles will be run by the MAX but no bandwidth will be lost since no master is making a request. Figure 36-9 illustrates the effect of a higher priority master giving up control of the bus. When the slave port is programmed for round-robin mode of arbitration, then it will switch masters at any time there is more than one master actively making a request to it. This will happen because any master other than the one which presently owns the bus will be considered to have higher priority. Figure 36-10 shows an example of round-robin mode of operation.

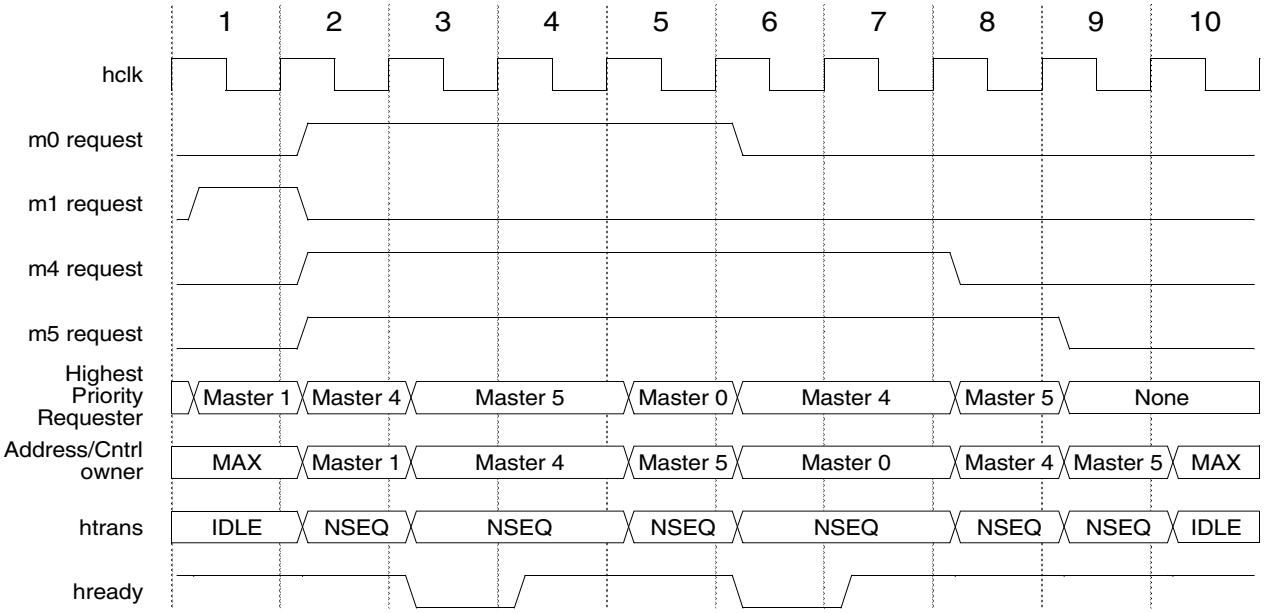


Figure 36-10. Round-Robin Mastership Change

36.5.4.4.4 Parking

If no master is currently making a request to the slave port then the slave port will be parked. It will park in one of four places, dictated by the PCTL and PARK bits in the GPCR or AGPCR (depending on the state of the sX_ampr_sel) and the locked state of the last master to access it.

If the last master to access the slave port ran a locked cycle and continues to run locked cycles even after leaving the slave port, the slave port will park on that master irrespective of GPCR bit settings and without regard to pending requests from other masters. This is done so that a master can run a locked transfer to the slave port, leave it, and return to it and be guaranteed that no other master accessed it. If locking is not an issue for parking the GPCR bits will dictate the parking method.

If PCTL bits are set for “low power park” mode then the slave port will enter low power park mode. It will not recognize any master as being in control of it and it will not select any master’s signals to pass through to the slave bus. In this case all slave bus activity will effectively halt because all slave bus signals being driven from the MAX will be 0. This of course can save quite a bit of power if the slave port will not be in use for some time. The down side is that when a master does make a request to the slave port it will be delayed by one clock since it will have to arbitrate to acquire ownership of the slave port.

If PCTL bits are set to “park on last” mode then the slave port will park on the last master to access it, passing all that masters signals through to the slave bus. MAX will asynchronously force htrans[1:0], hmaster[3:0], hburst[2:0] and hmastlock to 0 for all access that the master does not run to the slave port.

When that master access the slave port again it will not pay any arbitration penalty; however, if any other master wishes to access the slave port a one clock arbitration penalty will be imposed. Figure 36-11 illustrates parking on the last master. Note that in cycle 6 simultaneous requests are made by master 2 and master 4. Although master 2 has higher priority, the slave bus is parked on master 4 so master 4's access will be taken first. The slave port parks on master 2 once it has given control to master 2. This same situation can occur when parking on a specific master as well.

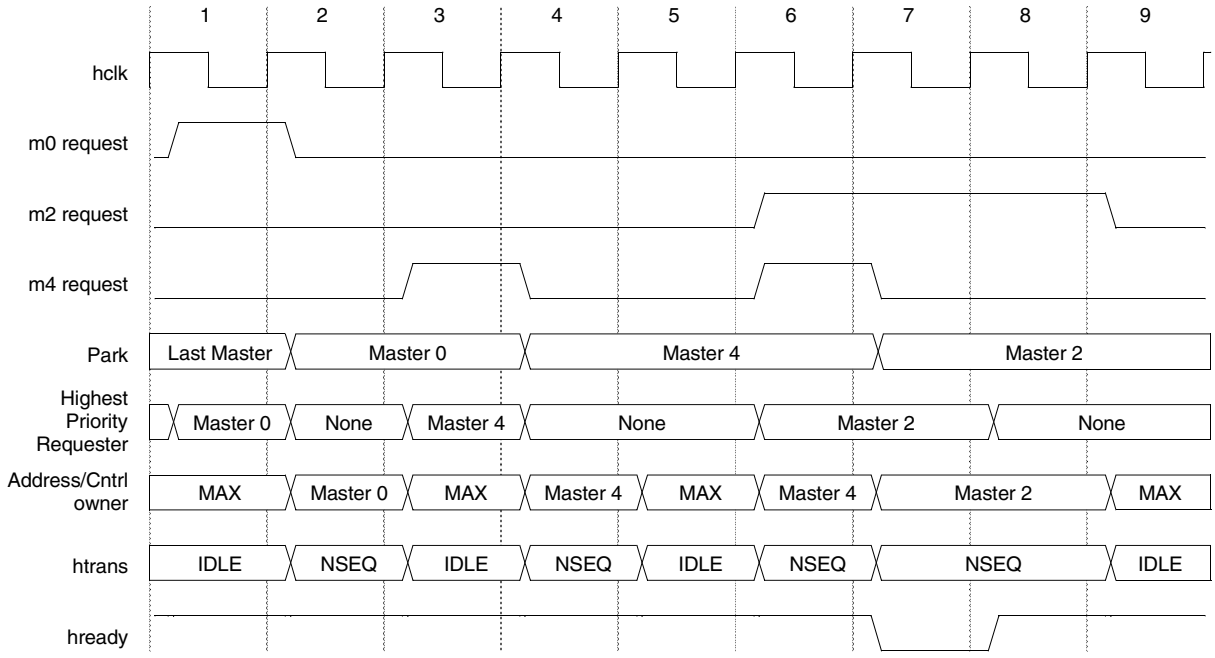


Figure 36-11. Parking on Last Master

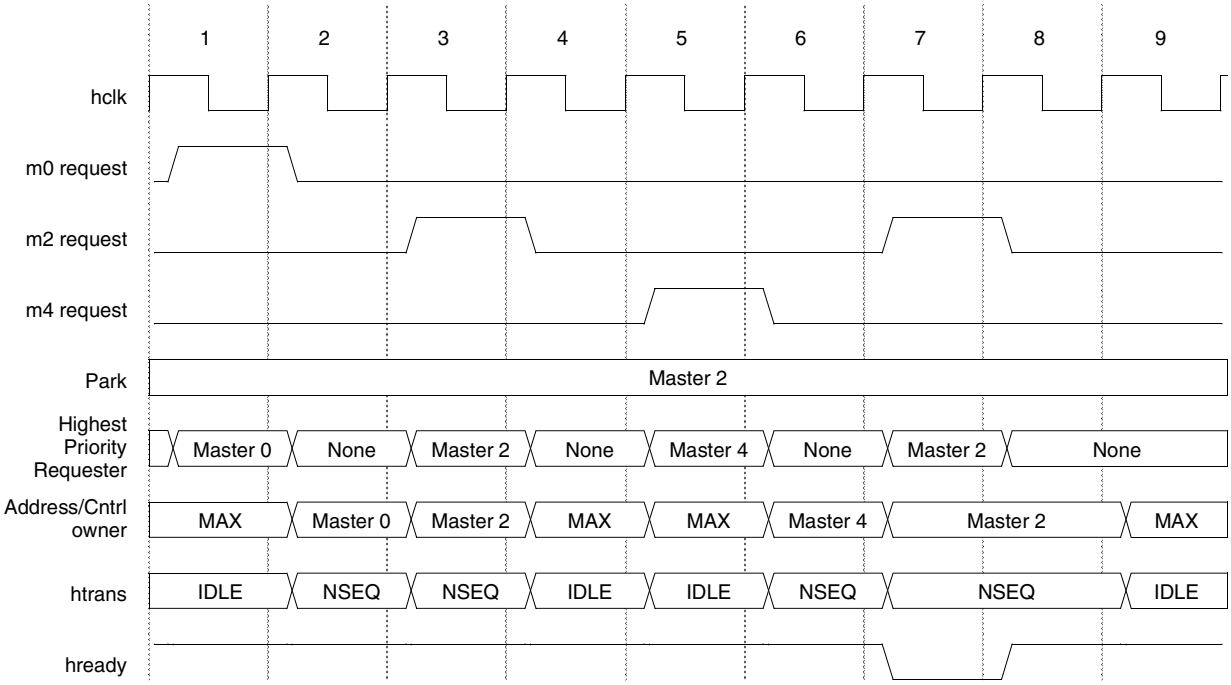


Figure 36-12. Parking on a Specific Master

If PCTL bits are set to “use PARK/APARK” mode, then the slave port will park on the master designated by the PARK bits. The behavior here is the same as for the “park on last” mode with the exception that a specific master will be parked on instead of the last master to access the slave port. If the master designated by the PARK bits tries to access the slave port it will not pay an arbitration penalty while any other master will pay a one clock penalty. Figure 36-12 illustrates parking on a specific master.

36.5.4.4.5 Halt Mode

If max_halt_request input is asserted, slave port will eventually halt all slave bus activity and go into halt mode, which is almost identical to low power park mode. HLP bit in GPCR controls the priority level of max_halt_request in arbitration algorithm. If HLP bit is cleared, then max_halt_request will have the highest priority of any master and will gain control of the slave port at the next arbitration point (most likely the next bus cycle, unless current master is running a locked or fixed length burst transfer). If HLP bit is set, then the slave port will wait until no masters are actively making requests before moving to halt mode. Regardless of HLP bit state, once slave port has gone into halt mode as a result of max_halt_request being asserted, it will remain in halt mode until max_halt_request is negated, regardless of the priority level of any masters that may make requests. In halt mode no master is selected to own the slave port so all the outputs of the slave port are set to 0.

36.6 Initialization/Application Information

No initialization is required by or for MAX. Hardware reset ensures all register bits used by MAX are properly initialized.

36.7 Interface

MAX main goal is to increase overall system performance by allowing multiple masters to communicate in parallel with multiple slaves. In order to maximize data throughput, it is essential to keep arbitration delays to a minimum. This section examines data throughput from the point of view of masters and slaves, detailing when MAX will stall the masters or insert bubbles on the slave side. Master accesses will receive one of four responses from MAX. They will either be terminated, taken, stalled or responded to an error.

36.7.1 Master Ports

Master accesses will receive one of four responses from the MAX. They will either be terminated, taken, stalled or responded to with an error.

Terminated Accesses: A master access will be terminated if the transfer type is IDLE. MAX will terminate the access and it will not be allowed to pass through the MAX.

Taken Accesses: A master access will be taken if the transfer type is non IDLE and the slave port to which the access decodes is either currently servicing the master or is parked on the master. In this case MAX will be completely transparent and the master's access will be immediately seen on the slave bus and no arbitration delays will be incurred.

Stalled Accesses: A master access will be stalled if transfer type is non IDLE and access decodes to a slave port that is busy serving another master, parked on another master or is in low power park mode. MAX will indicate to the master that address phase of the access has been taken but will then queue the access to appropriate slave port to enter into arbitration for access to that slave port. If the slave port is currently parked on another master or is in low power park mode and no other master is requesting access to it, then only one clock of arbitration will be incurred. If the slave port is currently serving another master of a lower priority and the master has a higher priority than all other requesting masters then the master will gain control over the slave port as soon as the data phase of the current access is completed (burst and locked transfers excluded). If the slave port is currently servicing another master of a higher priority then the master will gain control of the slave port once the other master releases control of the slave port if no other higher priority master is also waiting for the slave port.

Error Response Terminated Accesses:

A master access will be responded to with an error if the transfer type is non IDLE and the access decodes to a location not occupied by a slave port. This is the only time the MAX will respond with an error response. All other error responses received by the master are result of error responses on the slave ports being passed through MAX.

36.7.2 Slave Ports

The goal of MAX with respect to the slave ports is to keep them 100% saturated when masters are actively making requests. In order to do this, MAX must not insert any bubbles onto the slave bus unless absolutely

necessary. There is only one instance when MAX will force a bubble onto the slave bus when a master is actively making a request. This occurs when a higher priority master has control of the slave port and is running single clock (zero wait state) access while a lower priority master is stalled waiting for control of the slave port. When higher priority master either leaves the slave port or runs an IDLE cycle to the slave port, MAX will take control of the slave bus and run a single IDLE cycle before giving the slave port to the lower priority master that was waiting for control of the slave port.

The only other times, MAX will have control of the slave port is when MAX is halting or when no masters are making access requests to the slave port and the MAX is forced to either park the slave port on a specific master or put the slave port into low power park mode. In most instances when MAX has control of slave port, it will indicate IDLE for the transfer type, negate all control signals and indicate ownership of the slave bus via the **hmaster** encoding of 4'b0000. One exception to this rule is when a master running locked cycles has left the slave port but continues to run locked cycles. In this case, MAX will control the slave port and will indicate IDLE for the transfer type but it will not affect any other signals.

NOTE

When a master runs a locked cycle through the MAX, master will be guaranteed ownership of all slave ports it accesses while running locked cycles for one cycle beyond when the master finishes running locked cycles.

Chapter 37
Direct Memory Access Controller (DMAC)

The Direct Memory Access Controller (DMAC) of i.MX27 device provides 16 DMA channels supporting linear memory, 2D memory and FIFO transfers to provide support for a wide variety of DMA operations. Figure 37-1 shows a simplified block diagram of the DMAC.

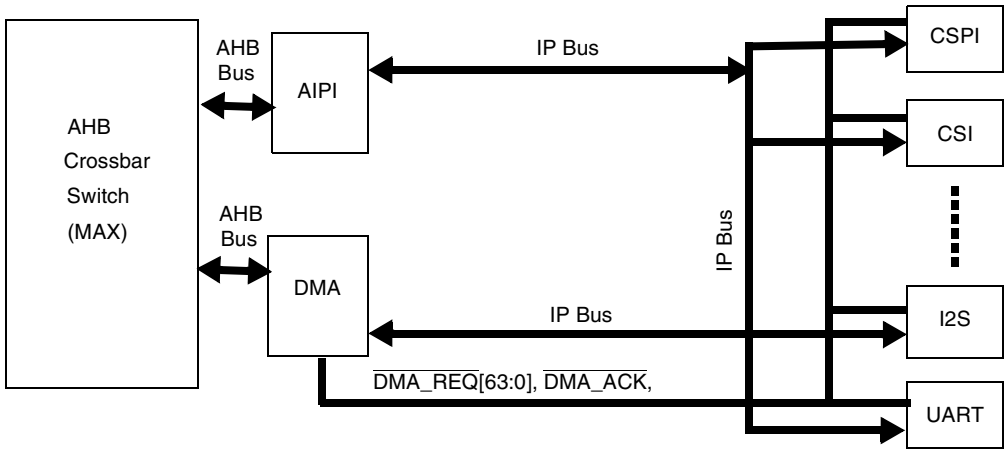


Figure 37-1. Block Diagram of DMAC

37.1 Features

- Sixteen channels support linear memory, 2D Memory, FIFO for both source and destination.
- DMA chaining for variable length buffer exchanges and high allowable interrupt latency requirement.
- Increment, decrement, and no-change support for source and destination addresses.
- Each channel is configurable to respond to any of the 64 DMA request signals.
- Supports 8, 16, or 32-bit FIFO and memory port size data transfers.
- DMA burst length configurable up to a maximum of 16 words, 32 half-words, or 64 bytes for each channel.
- Bus utilization control for the channel that is not trigger by a DMA request.
- Burst time-out errors terminates the DMA cycle when the burst cannot be completed within a programmed time count.
- Buffer overflow error terminates the DMA cycle when the internal buffer receives more than 64 bytes of data.
- Transfer error terminates the DMA cycle when a transfer error is detected during a DMA burst.
- DMA request time-out errors are generated for channels that are triggered by DMA requests to interrupt the CPU when a DMA burst does not start on that channel after a programmed time count.

- Interrupts are provided to the interrupt controller (and subsequently to the core) on bulk data transfer complete or transfer error.
- Each peripheral that supports DMA transfer generates a $\overline{\text{DMA\_REQ}}$ signal to the DMA controller, assuming that each FIFO has a unique system address and generates a dedicated $\overline{\text{dma\_req}}$ signal to the DMA controller. For example, a USB device with 8 end-points has 8 DMA request signals to the DMA if they all support DMA transfer.
- DMA controller provides an acknowledge signal to the peripheral after DMA burst is complete. This signal is sometimes used by the peripheral to clear the status bits.
- Repeat data transfer function supports automatic USB–host–USB device bulk/iso data stream transfer.
- Dedicated external DMA request signal.

DMA has a FIFO for storing data read from AHB Bus. This FIFO is 32-bits wide and 16 deep to store up to 64 bytes. This FIFO is common for all channels and is used by the active channel.

37.2 DMA Request and Acknowledge

Initiation of a DMA cycle can be done through software control (setting CEN = 1 and REN =0 in channel control register) or by DMA request assertion (setting CEN = 1 and REN =1 in channel control register). A DMA cycle consists of a number of DMA bursts depending on burst length and count register settings. Table 37-1 contains the DMA request map.

37.2.1 DMA Request

A typical DMA request is an active low signal asserted by the peripheral. The sampling of this signal is done when REN and CEN bits in Channel Control register are set and there is no other ongoing DMA transfer on the AHB bus. There is no configurable priority associated with any request. However, the 16 channels have a fixed priority; channel 15 has the highest priority and channel 0 has the lowest priority. The priority of any request depends on the channel number to which the request is mapped (through Request Source Select register settings). DMAC does not store DMA request inputs, it processes on the highest priority channel request out of the asserted channel requests (when no other transfer is taking place). A peripheral must keep the request asserted until it is serviced by DMAC. There are 64 input DMA request signals available. 1 DMA request will initiate 1 DMA burst. Once a DMA burst has started, DMA request can be de-asserted by the peripheral. The peripheral should de-assert the DMA request based on data read from or written into it. If the request is not de-asserted till the end of the DMA burst, it can initiate another DMA burst.

37.2.2 External DMA Request and Grant

After assertion of External DMA Request, the DMA burst will start when the corresponding DMA channel becomes the highest priority channel. External DMA Request should be kept asserted until it is serviced by the DMAC. One External request will initiate at least one DMA burst. The output External Grant signal from the DMAC is an active-low signal. This signal will be asserted during the time when a DMA burst is ongoing for an External DMA Request, when the following conditions are true:

— The DMA channel for which the DMA burst is ongoing has requested source as external DMA Request (as per RSSR settings).
— REN and CEN bit of this channel are set
— External DMA Request is asserted

Once the grant is asserted, External DMA Request will not be sampled until completion of the DMA burst. The priority of external request will become low for the next consecutive burst, if another DMA request signal is asserted. The waveforms are shown for the worst case, that is, smallest burst (1 byte read/write). Minimum and maximum timings for External request and External grant signal are present in the data sheet. Figure 37-2 shows the minimum time for which the External Grant signal remains asserted if External DMA request is de-asserted immediately after sensing grant signal active.

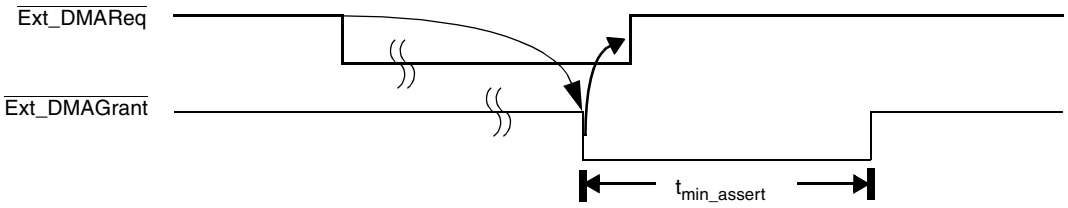
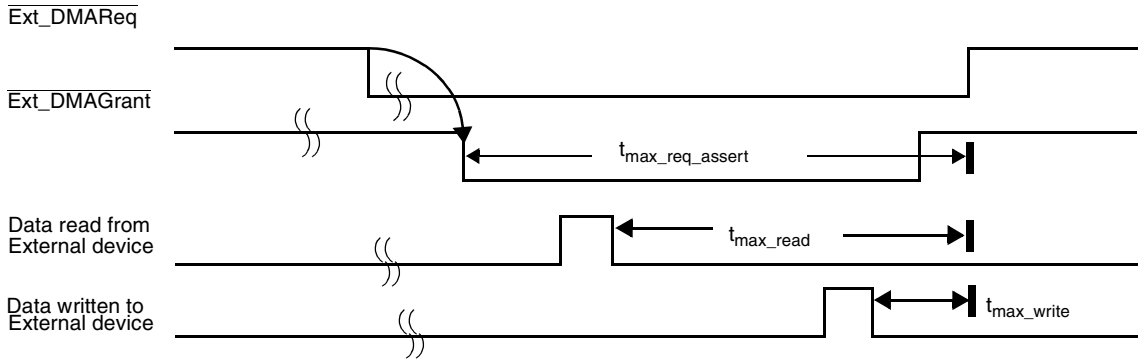


Figure 37-2. Assertion of DMA External Grant Signal

Figure 37-3 shows the safe max time for which External DMA request can be kept asserted, after sensing grant signal active as if a new burst is not initiated.



NOTE: Assuming in worst case the data is read/written from/to External device as per the above waveform.

Figure 37-3. Safe Maximum Timings for External Request De-Assertion

37.3 DMA Request Mapping

Table 37-1 shows requests connection from various modules in the i.MX27 to DMA Request input of DMAC

Table 37-1. DMA Request Mapping

DMA Request Number	Module Assigned	Channel Name
DMA_REQ[63:38]	Reserved	Reserved
DMA_REQ[37]	EMI	NFC

Table 37-1. DMA Request Mapping (continued)

DMA Request Number	Module Assigned	Channel Name
DMA_REQ[36]	SDHC3	SDHC3
DMA_REQ[35]	UART6	UART6_RX_FIFO
DMA_REQ[34]	UART6	UART6_TX_FIFO
DMA_REQ[33]	UART5	UART5_RX_FIFO
DMA_REQ[32]	UART5	UART5_TX_FIFO
DMA_REQ[31]	CSI	CSI_RX_FIFO
DMA_REQ[30]	CSI	CSI_STAT_FIFO
DMA_REQ[29]	ATA	ATA_RCV_FIFO
DMA_REQ[28]	ATA	ATA_TX_FIFO
DMA_REQ[27]	UART1	UART1_TX_FIFO
DMA_REQ[26]	UART1	UART1_RX_FIFO
DMA_REQ[25]	UART2	UART2_TX_FIFO
DMA_REQ[24]	UART2	UART2_RX_FIFO
DMA_REQ[23]	UART3	UART3_TX_FIFO
DMA_REQ[22]	UART3	UART3_RX_FIFO
DMA_REQ[21]	UART4	UART4_TX_FIFO
DMA_REQ[20]	UART4	UART4_RX_FIFO
DMA_REQ[19]	CSPI1	CSPI1_TX_FIFO
DMA_REQ[18]	CSPI1	CSPI1_RX_FIFO
DMA_REQ[17]	CSPI2	CSPI2_TX_FIFO
DMA_REQ[16]	CSPI2	CSPI2_RX_FIFO
DMA_REQ[15]	SSI1	SSI1_TX1_FIFO
DMA_REQ[14]	SSI1	SSI1_RX1_FIFO
DMA_REQ[13]	SSI1	SSI1_TX0_FIFO
DMA_REQ[12]	SSI1	SSI1_RX0_FIFO
DMA_REQ[11]	SSI2	SSI2_TX1_FIFO
DMA_REQ[10]	SSI2	SSI2_RX1_FIFO
DMA_REQ[9]	SSI2	SSI2_TX0_FIFO
DMA_REQ[8]	SSI2	SSI2_RX0_FIFO
DMA_REQ[7]	SDHC1	SDHC1
DMA_REQ[6]	SDHC2	SDHC2
DMA_REQ[5]	Reserved	Reserved
DMA_REQ[4]	MSHC	MSHC

Table 37-1. DMA Request Mapping (continued)

DMA Request Number	Module Assigned	Channel Name
DMA_REQ[3]	External DMA request	—
DMA_REQ[2]	CSPI3	CSPI3_TX_FIFO
DMA_REQ[1]	CSPI3	CSPI3_RX_FIFO
DMA_REQ[0]	Reserved	—

37.4 Memory Map and Register Definition

The DMAC module contains 158 32-bit registers. The registers are divided into four groups according to the register functions as follows:

- General registers for all functional blocks (see Section 37.4.3, “General Registers”).
- 2D memory registers to control display width and x and y of the window (see Section 37.4.4, “2D Memory Registers (A and B)”).
- Channel registers to control and configure channels 0–15 (see Section 37.4.5, “Channel Registers”).
- Test Registers.

The base address of DMA Controller for i.MX27 is 0x10001000. Table 37-4 summarizes the registers and offset addresses. Section 37.4, “Memory Map and Register Definition” provides detailed descriptions of the DMAC registers.

37.4.1 DMAC Memory Map

Table 37-2 shows the DMAC memory map.

Table 37-2. DMAC Memory Map

Address	Use	Access	Reset Value	Section/Page
0x1000_1000 (DCR)	DMA Control Register	R/W	0x0000_0000	37.4.3.1/37-10
0x1000_1004 (DISR)	DMA Interrupt Status Register	R/W	0x0000_0000	37.4.3.2/37-11
0x1000_1008 (DIMR)	DMA Interrupt Mask Register	R/W	0x0000_0000	37.4.3.3/37-12
0x1000_100C (DBTOSR)	DMA Burst Time-Out Status Register	R/W	0x0000_0000	37.4.3.4/37-13
0x1000_1010 (DRTOSR)	DMA Request Time-Out Status Register	R/W	0x0000_0000	37.4.3.5/37-13
0x1000_1014 (DSESR)	DMA Transfer Error Status Register	R/W	0x0000_0000	37.4.3.6/37-14
0x1000_1018 (DBOSR)	DMA Buffer Overflow Status Register	R/W	0x0000_0000	37.4.3.7/37-15
0x1000_101C (DBTOCR)	DMA Burst Time-Out Control Register	R/W	0x0000_0000	37.4.3.8/37-15
0x1000_1040 (WSRA) 0x1000_104C (WSRB)	W-Size Register A W-Size Register B	R/W	0x0000_0000	37.4.4.1/37-17
0x1000_1044 (XSRA) 0x1000_1050 (XSRB)	X-Size Register A X-Size Register B	R/W	0x0000_0000	37.4.4.2/37-18

Table 37-2. DMAC Memory Map (continued)

Address	Use	Access	Reset Value	Section/Page
0x1000_1048 (YSRA) 0x1000_1054 (YSRB)	Y-Size Register A Y-Size Register B	R/W	0x0000_0000	37.4.4.3/37-19
0x1000_1080 (SAR0) – 0x1000_1440 (SAR15)	Channel 0 Source Address Register – Channel 15 Source Address Register	R/W	0x0000_0000	37.4.5.1/37-20
0x1000_1084 (DAR0) – 0x1000_1444 (DAR15)	Channel 0 Destination Address Register – Channel 15Destination Address Register	R/W	0x0000_0000	37.4.5.2/37-20
0x1000_1088 (CNTR0) – 0x1000_1448 (CNTR15)	Channel 0 Count Register – Channel 15 Count Register	R/W	0x0000_0000	37.4.5.3/37-21
0x1000_108C (CCR0) – 0x1000_144C (CCR15)	Channel 0 Control Register – Channel 15 Control Register	R/W	0x0000_0000	37.4.5.4/37-22
0x1000_1090 (RSSR0) – 0x1000_1450 (RSSR15)	Channel 0 Request Source Select Register – Channel 15 Request Source Select Register	R/W	0x0000_0000	37.4.5.5/37-25
0x1000_1094 (BLR0) – 0x1000_1454 (BLR15)	Channel 0 Burst Length Register – Channel 15Burst Length Register	R/W	0x0000_0000	37.4.5.6/37-26
0x1000_1098 (RTOR0) – 0x1000_1458 (RTOR15)[1]	Channel 0 Request Time-Out Register/ – Channel 15 Request Time-Out Register/	R/W	0x0000_0000 0x0000_0000	37.4.5.7/37-26
0x1000_1098 (BUCR0) – 0x1000_1458 (BUCR15)[2]	Channel 0 Bus Utilization Control Register – Channel 15 Bus Utilization Control Register	R/W	0x0000_0000 0x0000_0000	37.4.5.8/37-27
0x1000_109C (CCNR0) – 0x1000_145C (CCNR15)	Channel 0 Channel Counter Register – Channel 15Channel Counter Register	R/W	0x0000_0000	37.4.5.9/37-28

[1] The RTOR registers use the same memory addresses as the BUCR registers.

[2] he BUCR registers use the same memory addresses as the RTOR registers.

37.4.2 Register Summary

Figure 37-4 shows the key to the register fields, and Table 37-3 shows the register figure conventions.

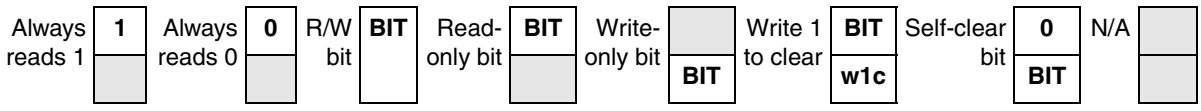


Figure 37-4. Key to Register Fields

Table 37-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 37-4 shows the DMAC register summary.

Table 37-4. DMAC Register Summary

Name		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
0x1000_1000 (DCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	DAM	0	DEN
	W															DRST	
0x1000_1004 (DISR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
	W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C

Table 37-4. DMAC Register Summary (continued)

Name		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
0x1000_1008 (DIMR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
	W																
0x1000_100C (DBTOSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
	W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
0x1000_1010 (DRTOSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
	W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
0x1000_1014 (DSESR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
	W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
0x1000_1018 (DBOSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
	W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
0x1000_101C (DBTOCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EN	CNT [14: 0]														
	W																

Table 37-4. DMAC Register Summary (continued)

Name		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
0x1000_1040 (WSRA) 0x1000_104C (WSRB)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	WS [15: 0]															
	W																
0x1000_1044 (XSRA) 0x1000_1050 (XSRB)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	XS [15: 0]															
	W																
0x1000_1048 (YSRA) 0x1000_1054 (YSRB)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	YS [15: 0]															
	W																
0x1000_1080 (SAR0) – 0x1000_1440 (SAR15)	R	SA [31: 16]															
	W																
	R	SA [15: 0]															
	W																
0x1000_1084 (DAR0) – 0x1000_1444 (DAR15)	R	DA [31: 16]															
	W																
	R	DA [15: 0]															
	W																
0x1000_1088 (CNTR0) – 0x1000_1448 (CNTR15)	R	0	0	0	0	0	0	0	0	CNT [23: 16]							
	W																
	R	CNT [15: 0]															
	W																
0x1000_108C (CCR0) – 0x1000_144C (CCR15)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	ACR PT	DMOD		SMOD		MDI R	MS EL	DSIZ		SSIZ		REN	RPT	0	CEN
	W															FRC	

Table 37-4. DMAC Register Summary (continued)

Name		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
0x1000_1090 (RSSR0) – 0x1000_1450 (RSSR15)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0 RSS [5: 0]					
	W																
0x1000_1094 (BLR0) – 0x1000_1454 (BLR15))	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	BL [5: 0]					
	W																
0x1000_1098 (RTOR0) – 0x1000_1458 (RTOR15))	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EN	CLK	PSC	CNT [12: 0]												
	W																
0x1000_1098 (BUCR0) – 0x1000_1458 (BUCR15))	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	BU_CNT [15: 0]															
	W																
0x1000_109C (CCNR0) – 0x1000_145C (CCNR15)	R	0	0	0	0	0	0	0	0	CCNR[23:16]							
	W																
	R	CCNR [15: 0]															
	W																

37.4.3 General Registers

37.4.3.1 DMA Control Register (DCR)

The DMA Control Register (DCR) controls the input of the system clock, enabling, disabling, and resetting of the DMA module.

0x1000_1000 (DCR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DAM	0	DEN
W															DRST	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-5. DMA Control Register (DCR)

Table 37-5. DMA Control Register Field Descriptions

Field	Description
31–3	Reserved. These bits are reserved and should read 0.
2 DAM	DMA Access Mode. Specifies user or privileged access to be performed by DMA. 0 Privileged access 1 User access
1 DRST	DMA Soft Reset. Generates a 3-cycle reset pulse that resets the entire DMA module, bringing the module to its reset condition. DRST always reads 0. 0 No effect 1 Generates a 3-cycle reset pulse
0 DEN	DMA Enable. Enables/Disables the system clock to the DMA module. However the bit is not used for clock gating in i.MX27 as the clock is controlled from CRM in the i.MX27 processor. 0 DMA disable 1 DMA enable

37.4.3.2 DMA Interrupt Status Register (DISR)

The DISR contains interrupt status of each channel in the DMAC. The status bit is set whenever the corresponding DMA channel data transfer is complete. When any bit in DISR is set and the corresponding bit in the interrupt mask register is cleared, dma_int is asserted to the interrupt controller (AITC). When an interrupt occurs, an interrupt service routine must check DISR to determine the interrupting channel. Clear each bit by writing a value 1 to it.

0x1000_1004 (DISR) Access: User Read-Only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-6. DMA Interrupt Status Register

Table 37-6. DMA Interrupt Status Register Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15–0 CH15–CH0	Channel 15 to 0 Interrupt Status. Indicates interrupt status for each DMA channel. 0 No interrupt 1 Interrupt is pending

37.4.3.3 DMA Interrupt Mask Register (DIMR)

DIMR masks both normal interrupts and error interrupts generated by the corresponding channel. There is one control bit for each channel. When an interrupt is masked, the interrupt controller does not generate an interrupt request to the AITC, however the status of the interrupt can be observed from the interrupt status register, burst time-out status register, request time-out status register, or the transfer error status register. At reset, all the interrupts are masked and all the bits in this register are set to 1.

0x1000_1008 (DIMR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-7. DMA Interrupt Status Register (DIMR)

Table 37-7. DMA Interrupt Mask Register Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15–0 CH15–CH0	Channel 15 to 0 Interrupt Mask. Controls the interrupts for each DMA channel. 0 Enables interrupts 1 Disables interrupts

37.4.3.4 DMA Burst Time-Out Status Register (DBTOSR)

A burst time-out is set when a DMA burst cannot be completed within the number of clock cycles specified in the DMA Burst Time-Out Control Register (DBTOCR) of the channel. When any bit is set in this register and the corresponding bit in the interrupt mask register is cleared, a DMA Error interrupt is asserted to the interrupt controller (AITC). DBTOSR indicates the channel, if any, that is currently being serviced and whether a burst time-out was detected. Each bit is cleared by writing 1 to it.

0x1000_100C (DBTOSR) Access: User Read-Only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-8. DMA Burst Time-Out Status Register (DBTOSR)

Table 37-8. DMA Burst Time-Out Status Register Description

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15–0 CH15–CH0	Channel 15 to 0. Indicates the burst time-out status of each DMA channel. 0 No burst time-out 1 Burst time-out

37.4.3.5 DMA Request Time-Out Status Register (DRTOSR)

A DMA request time-out is set when there is no DMA burst started on the channel (when REN =1, either due to no DMA Request or DMA channel not acquiring the bus) within the pre-assigned number of clock cycles specified in the Request Time-Out control register (RTOR) for the channel. When any bit is set in this register and the corresponding bit in the interrupt mask register is cleared, a DMA Error Interrupt is asserted to the interrupt controller (AITC). DRTOSR indicates the enabled channel, if any, that detected a DMA request time-out. Clear each bit by writing 1 to it.

0x1000_1010 (DRTOSR) Access: User Read-Only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-9. DMA Request Time-Out Status Register (DRTOSR)

Table 37-9. DMA Request Time-Out Status Register Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15–0 CH15–CH0	Channel 15 to 0. Indicates the request time-out status of each DMA channel. 0 No DMA request time-out 1 DMA request time-out

37.4.3.6 DMA Transfer Error Status Register (DSESR)

A DMA transfer error is set when DMA data transfer results in an error. When any bit is set in this register and the corresponding bit in the interrupt mask register is cleared, a DMA Error Interrupt is asserted to the interrupt controller (AITC). DSESR indicates the channel, if any, the detected transfer error during a DMA burst. Clear each bit by writing 1 to it.

0x1000_1014 (DSESR) Access: User Read-Only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-10. DMA Transfer Error Status Register (DSESR)

Table 37-10. DMA Transfer Error Status Register Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15–0 CH15–CH0	Channel 15 to 0. Indicates the DMA transfer error status of each DMA channel. 0 No transfer error 1 Transfer error

37.4.3.7 DMA Buffer Overflow Status Register (DBOSR)

The DBOSR indicates whether DMA Controller's internal FIFO buffer overflowed during a data transfer. Before a channel can be enabled for DMA, the corresponding bit in this register must be cleared. When any bit in this register is set and the corresponding bit in the interrupt mask register is cleared, a DMA Error Interrupt is asserted to the interrupt controller (AITC).

0x1000_1018 (DBOSR) Access: User Read-Only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-11. DMA Buffer Overflow Status Register (DBOSR)

Table 37-11. DMA Buffer Overflow Status Register Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15–0 CH15–CH0	Channel 15 to 0. Indicates the buffer overflow error status of each DMA channel. 0 No buffer overflow occurred 1 Buffer overflow occurred

37.4.3.8 DMA Burst Time-Out Control Register (DBTOCR)

This register sets the DMA burst time-out (common for all DMA channels), so that DMAC can release the AHB and IP buses in the event of an error. An internal counter starts counting when a DMA burst starts, and resets to zero when the burst is completed. When the counter reaches the count value set in the register, it asserts an interrupt and sets the corresponding error bit in the DMA Burst Time-Out Status Register (DBTOSR). The system clock is used as the input clock to the counter.

0x1000_101C (DBTOCR) Access: User Read/Write

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

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

Figure 37-12. DMA Burst Time-Out Control Register (DBTOCR)

Table 37-12. DMA Burst Time-Out Control Register Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15 EN	Enable. Enables/Disables the burst time-out. 0 Disables burst time-out 1 Enables burst time-out
14–0 CNT	Count. This count is the number of system clock cycles to be used for the time-out value

37.4.4 2D Memory Registers (A and B)

There are two sets of 2D memory registers that allow every channel to select any register set to define the respective 2D memory size. Each data transfer performed by DMA is strictly as per Source and destination sizes specified in Channel Control Register (this is valid for Linear memory, 2D memory, and FIFO mode). In the case of a transfer to/from 2D Memory, the Channel Count register value is ignored and number of bytes transferred is equal to 2D Memory size. 2D Memory Size is computed as follows.

Size (in number of bytes) = No of bytes per row (value in X register) * No. of rows (Value in Y Register)

At a time any number of channels can be programmed for 2D Memory (even all 16 Channels). 2D Memory can be selected for a channel as source or destination or even both. In the last condition, only selected set of 2D Registers (selected as per MSEL bit setting in the Channel Control Register) is used for both source and destination. The advantage of having 2 sets of registers, which are usable by all DMA channels, is that this allows the developer to have two different window size settings for 2D memory. Each channel can be programmed to use any one of the 2 settings. In Figure 37-13, shaded portion shows the data transfer zone in the 2D memory for X= 4, Y = 4, and W (display size) = 6 with memory increment option and starting address 0x001. In Figure 37-14, shaded portion shows the data transfer zone in 2D memory for X= 4, Y = 5, and W = 6 with memory decrement option and starting address as 0x11C.

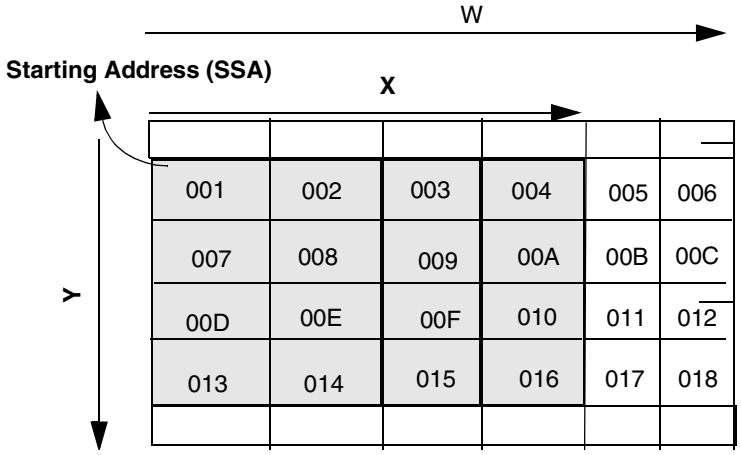


Figure 37-13. 2D Memory Increment Diagram

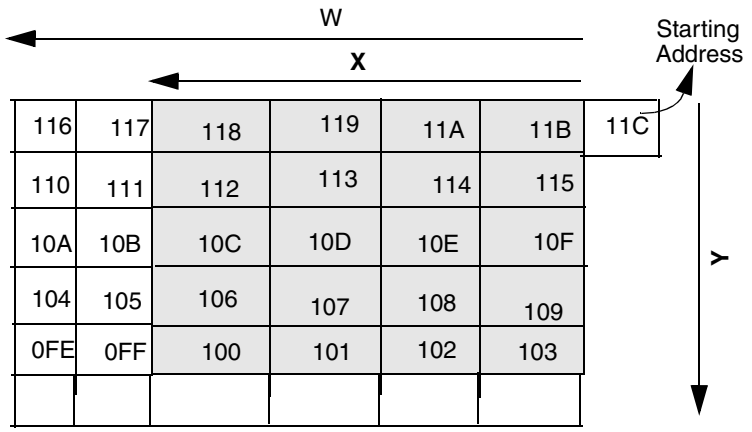


Figure 37-14. 2D Memory Decrement Diagram

37.4.4.1 W-Size Registers (WSRA, WSRB)

The W-Size registers (WSRA, WSRB) define the number of bytes that make up the display width. This allows the DMAC to calculate the next starting address of another row by adding the source/destination address to the W-Size register content.

0x1000_1040 (WSRA)
0x1000_104C (WSRB)

Access: User Read/Write

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

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

Figure 37-15. W-Size Registers (WSRA, WSRB)

Table 37-13. W-Size Registers Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15–0 WS	W-Size. Contains the number of bytes that makes up the display width. W and X must follow the relation: W ≥ X W and Access Size must follow the relation: W ≥ access size. Wsize needs to be a multiple of Source or Destination Access size whichever is a 2D memory.

37.4.4.2 X-Size Registers (XSRA, XSRB)

X-Size registers (XSRA, XSRB) contain the number of bytes per row of the window. The value of this register is used by the DMA controller to determine when to jump to the next row.

0x1000_1044 (XSRA)
0x1000_1050 (XSRB)

Access: User Read/Write

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

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

Figure 37-16. X-Size Registers (XSRA, XSRB)

Table 37-14. X-Size Registers Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15–0 XS	X-Size. Contains the number of bytes per row that defines the X-Size of 2D memory. The value in the X Register should follow the following 2 rules: • X ≥ Burst Length (BL) • X/ BL = Whole number.

37.4.4.3 Y-Size Registers (YSRA, YSRB)

The Y-Size registers (YSRA, YSRB) contain the number of rows in the 2D memory window. This setting is used by the DMA controller to calculate the total size of transfer.

0x1000_1048 (YSRA)
0x1000_1054 (YSRB)

Access: User Read/Write

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

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

Figure 37-17. Y Size Registers (YSRA, YSRB)

Table 37-15. Y-Size Registers Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15–0 YS	Y-Size. Contains the number of rows that makes up the 2D memory window.

37.4.5 Channel Registers

Channels 0 to 15 supports linear memory, 2D memory, FIFO transfer. The DMA request ($\overline{\text{dma\_req}}$ [63:0]) signals do not have any configurable priority. The only priority available is the priority that is defined for each channel: channel 15 has the highest priority and channel 0 has the lowest priority. The channel priority is used only when more than one request occurs at the same time. Otherwise, channels are serviced on a first come, first serve basis. Each channel generates a normal interrupt to the interrupt handler when the data count reaches the selected value. Each channel generates an error interrupt to the interrupt handler when any of the following conditions exist:

- A DMA request time-out is true
- A DMA burst time-out is true during a burst cycle
- The internal buffer overflows during a burst cycle
- A transfer error acknowledge is asserted during a burst cycle

37.4.5.1 Channel Source Address Registers (SAR0–SAR15)

Each of the SAR contains the DMA cycle source address. The implementation must ensure that SAR's value is stored internally before use, to allow software to modify the register value for DMA chaining (see Section 37.5, "DMA Chaining). The value should be stored when CEN bit is set or at the end of the transfer when RPT bit is found set (before initiating the new transfer). If the memory direction bit (MDIR) in the channel control register (CCR) is clear (indicating a memory address increment), then SAR contains memory block start address. If MDIR is set (indicating a memory address decrement), then SAR contains the memory block end address.

0x1000_1080 (SAR0) Access: User Read/Write
–
0x1000_1440 (SAR15)

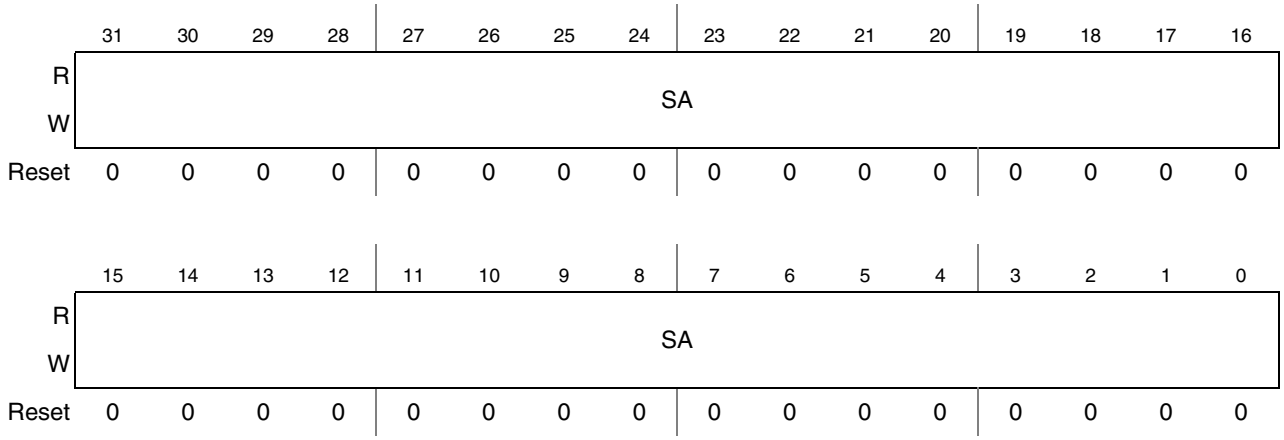


Figure 37-18. Channel Source Address Register (SAR0–SAR15)

Table 37-16. Channel Source Address Register Field Description

Field	Description
31–0 SA	Source Address. Contains source address from where data is read during a DMA transfer. DMA will not perform misaligned accesses. That is to say, for 32-bit transfers, the lower two bits of this address are ignored. 8-bit accesses begins from the address in this register. Software must take care if the system does not support non-word aligned accesses in this case.

37.4.5.2 Destination Address Registers (DAR0–DAR15)

Each of the DAR contains the DMA cycle destination address. The implementation needs to ensure that DAR's value is stored internally before use, to allow software to modify the register value for DMA Chaining (see Section 37.5, "DMA Chaining"). The value should be stored when CEN bit is set or at the end of the transfer when RPT bit is found set (before initiating the new transfer). If the memory direction bit (MDIR) in the channel control register (CCR) is clear (indicating a memory address increment), then

DAR contains memory block start address. If MDIR is set (indicating a memory address decrement), then DAR contains the memory block end address.

0x1000_1084 (DAR0) Access: User Read/Write
–
0x1000_1444 (DAR15)

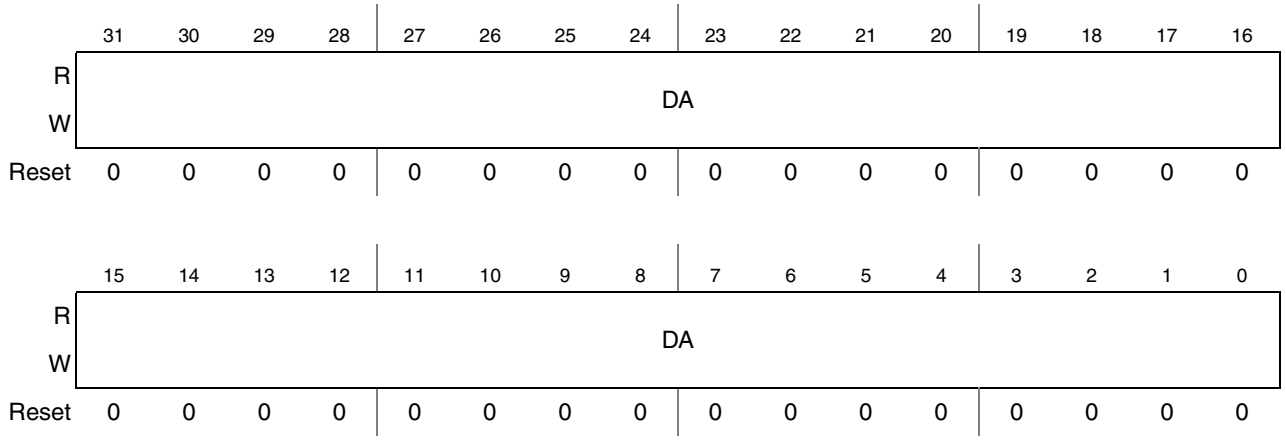


Figure 37-19. Channel Destination Address Registers (DAR0–DAR15)

Table 37-17. Channel Destination Address Registers Field Descriptions

Field	Description
31–0 DA	Destination Address. Contains destination address to which data is written to during a DMA transfer. DMAC will not perform misaligned accesses. That is, for a 32-bit transfers, the lower two bits of this address are ignored. 8-bit accesses begin from the address in this register. Software must take care if the system does not support non-word aligned accesses in this case.

37.4.5.3 Channel Count Registers (CNTR0–CNTR15)

The DEN bit in DCR should be set to enable writes to this register. The implementation needs to ensure that CNTR's value is stored internally before use, to allow software to modify the register value for DMA Chaining (see Section 37.5, "DMA Chaining"). The value should be stored in an Internal Count Register when CEN bit is set or at the end of the transfer when RPT bit is found set (before initiating new transfer).

Each of the CNTR contain the number of bytes of data to be transferred. There is an internal counter that counts up (number of bytes: 4 for word, 2 for half-word and 1 for byte) for every DMA transfer. The internal counter is compared with the Internal Count Register after every transfer. When the counter value matches with the register value, the channel is disabled until the CEN bit is cleared and set again, or the RPT bit in the corresponding channel control register is set to 1. The internal counter is reset to 0 when the channel is enabled again.

The length of the last DMA burst can be shorter than the regular burst length specified in the burst length register. However, when data is transferred out from an I/O FIFO and the last burst is less than BL, the I/O device must generate a DMA request for the last transfer. When data is transferred to an I/O FIFO and the last burst is less than BL, only the remaining number of data is transferred.

0x1000_1088 (CNTR0) Access: User Read/Write
–
0x1000_1448 (CNTR15)

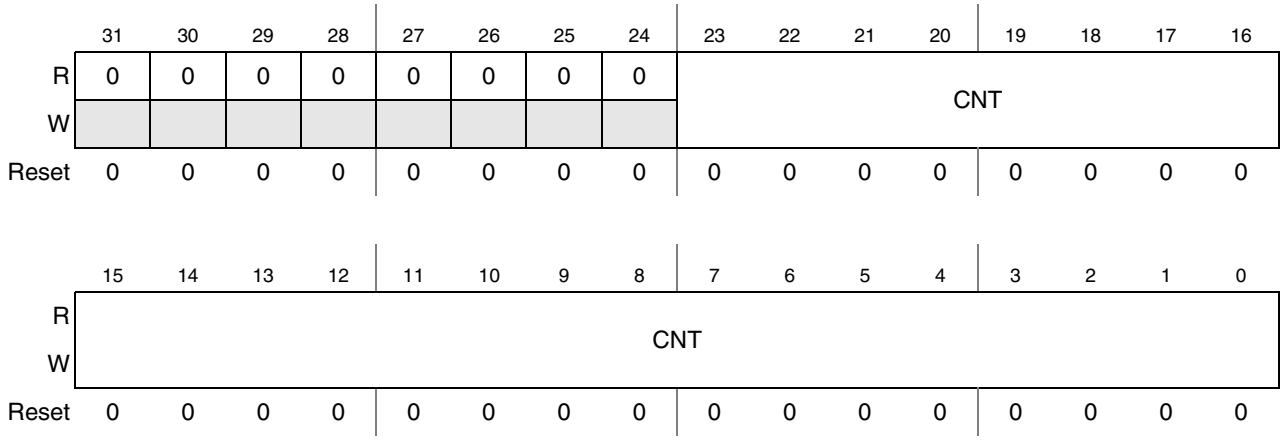


Figure 37-20. Channel Count Registers (CNTR0–CNTR15)

Table 37-18. Channel Count Registers Field Descriptions

Field	Description
31–24	Reserved. These bits are reserved and should read 0.
23–0 CNT	Count. Contains the number of bytes of data to be transferred during a DMA cycle.

37.4.5.4 Channel Control Registers (CCR0–CCR15)

Each of the CCR controls and displays the status of that DMA channel operation. DMAC has the capability to perform burst transfers of byte and half word data types while SDRAM controller and EIM support is restricted to burst transfers of word (32-bit) data types. Therefore, while using the DMA in conjunction with SDRAM controller and EIM, ensure that all burst transfers to/from SDRAM controller and EIM are of word data types. This is configured in the DMA Channel Control Register. While choosing SDRAM memory as the source or destination address, set SDRAMC and EIM as a 32-bit port.

0x1000_108C (CCR0) Access: User Read/Write
–
0x1000_144C (CCR15)

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	ACR PT	DMOD		SMOD		MDIR	MSEL	DSIZ		SSIZ		REN	RPT	FRC	CEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-21. Channel Control Registers (CCR0–CCR15)

Table 37-19. Channel Control Registers Field Descriptions

Field	Description
31–15	Reserved. These bits are reserved and should read 0.
14 ACRPT	Auto Clear RPT. This bit is to be sampled at the end of the transfer along with the RPT bit. When this bit and RPT are set, a new transfer is initiated and RPT is reset before issuing any interrupts. 0 Do not modify RPT 1 Reset RPT at end of current transfer.
13–12 DMOD	Destination Mode. Selects the destination transfer mode. 00 Linear memory 01 2D memory 10 FIFO 11 Reserved
11–10 SMOD	Source Mode. Selects the source transfer mode. 00 Linear memory 01 2D memory 10 FIFO 11 Reserved
9 MDIR	Memory Direction. Selects the memory address direction. Note: When address increment is chosen, data transfer starts from the values in SAR and DAR. When address decrement is chosen, data transfer will be done till the addresses mentioned in SAR and DAR, that is, no data read or write will be done at the address mentioned in SAR and DAR. 0 Memory address increment 1 Memory address decrement
8 MSEL	Memory Select. Selects 2D memory register set when source and/or destination is programmed to 2D memory mode. 0 2D memory register set A selected 1 2D memory register set B selected

Table 37-19. Channel Control Registers Field Descriptions

Field	Description
7–6 DSIZ	Destination Size. Selects the destination size of a data transfer. If number of bytes to be written is less than the DSIZ setting, then only that many bytes will be valid in the DMA write cycle to AHB. However all DMA write cycles to the destination will be of DSIZ size. DMA always writes data as per DSIZ in all modes, that is, Linear memory, 2D memory and FIFO mode. 00 32-bit destination port 01 8-bit destination port 10 16-bit destination port 11 Reserved
5–4 SSIZ	Source Size. Selects the source size of data transfer. 1. SSIZ1:SSIZ0 always reads/writes 00 when source mode is programmed as end-of-burst enable FIFO, because end of burst operation only works for 32-bit FIFO. 21. If the number of bytes to be read is less than the SSIZ setting, then only that many bytes will be used by the DMA. However, all DMA read cycles to the source will be of “SSIZ” size. 32. DMA always reads data as per SSIZ in all modes, that is, EOBE mode, Linear memory, 2D memory and FIFO mode. 00 32-bit source port 01 8-bit source port 10 16-bit source port 11 Reserved
3 REN	Request Enable. Enables/Disables the DMA request signal. When REN bit is set, DMA burst is initiated by the dma_req signal from the I/O FIFO. When REN is cleared, DMA transfer is initiated by CEN. 0 Disables the DMA request signal (when the peripheral asserts a DMA request, no DMA transfer is triggered); DMA transfer is initiated by CEN only 1 Enables the DMA request signal (when the peripheral asserts a DMA request, a DMA transfer is triggered)
2 RPT	Repeat. This is a status/control bit. Software has a priority and can write to this bit at any time. This bit Enables/Disables the data transfer repeat function. When enabled and when the counter reaches the value set in Internal Count Register: a. Source address, Destination address and Count Register values are stored (reloaded) for the next DMA Burst. b. If ACRPT bit is set, RPT bit is cleared. c. Next DMA cycle is enabled. After this an interrupt is asserted, if the corresponding channel bit in the Interrupt Mask Register is cleared. Data transfer is carried out continuously until the channel is disabled or it completes the last DMA burst after RPT is cleared. The status information in this bit is that it gets cleared when ACRPT is set as described above. **Note:** Implementation must ensure that RPT bit is sampled only at the time the counter reaches the value in the Internal Count Register before asserting the interrupt. RPT bit should be allowed to be modified at all other times by the software (for example in the interrupt subroutine). 0 Disables repeat function 1 Enables repeat function

Table 37-19. Channel Control Registers Field Descriptions

Field	Description
1 FRC	Force a DMA Cycle. Forces a DMA burst to occur when DMA cycle is software enabled or DMA Request is Enabled. When FRC bit is set, it will remain set till a DMA burst for this channel starts as per channel priority, and will get cleared after the DMA burst for the channel starts. When set, software will read this bit as'1' till it gets cleared automatically or cleared by software. 0 No effect 1 Force DMA cycle
0 CEN	DMA Channel Enable. Enables/Disables the DMA channel. **Note:** To re-program a particular channel after completion of a DMA cycle refer Section 37.8, "Application Note." **Note:** Disabling CEN during an ongoing burst on the AHB will stop the burst in between the transfer 0 Disables the DMA channel 1 Enables the DMA channel

37.4.5.5 Channel Request Source Select Registers (RSSR0–RSSR15)

Each of the 64-bit RSSR selects one of the 64 DMA request signals (DMA_REQ [63:0]) to initiate a DMA transfer for the corresponding channel.

0x1000_1090 (RSSR0) Access: User Read/Write
–
0x1000_1450 (RSSR15)

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

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

Figure 37-22. Channel Request Source Select Registers (RSSR0–RSSR15)

Table 37-20. Channel Request Source Select Registers Field Descriptions

Field	Description
31–6	Reserved. These bits are reserved and should read 0.
5–0 RSSR	Request Source Select. Selects 1 of the 64 $\overline{\text{dma\_req}}$ signals that initiates DMA transfer cycle for the channel. 000000 select $\overline{\text{dma\_req}}$[0] 000001 select $\overline{\text{dma\_req}}$ [1] ... 011111 select $\overline{\text{dma\_req}}$ [31] .... 111111 select $\overline{\text{dma\_req}}$ [63]

37.4.5.6 Channel Burst Length Registers (BLR0–BLR15)

The BLR controls burst length of a DMA cycle. For a FIFO channel setting, burst length is normally assigned according to FIFO size of the selected I/O device, or by FIFO level at which its dma_req signal is asserted. For example, when UART RxD FIFO is 12x8, and it asserts dma_req when it receives more than 8 bytes of data, BL is 8. When memory port size also is 8-bit, DMA burst is 8-byte read followed by 8-byte write. When memory port access size is smaller than I/O port access size, burst length of the byte write is doubled. For example, if I/O port is 32-bit and memory port is 16-bit, then burst length is set to 32. In this configuration, DMA performs 8 word burst reads and 16 half-word burst writes for I/O to memory transfer. When burst length is not programmed as a multiple of access sizes of source and destination, refer Section 37.9, "DMA Burst Termination" for behavior of DMA.

0x1000_1094 (BLR0) Access: User Read/Write
–
0x1000_1454 (BLR15)

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

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

Figure 37-23. Channel Burst Length Registers (BLR0–BLR15)

Table 37-21. Channel Burst Length Registers Field Descriptions

Field	Description
31–6	Reserved. These bits are reserved and should read 0.
5–0 BL	Burst Length. Contains the number of data bytes that are transferred in a DMA burst. 000000 64 bytes read follow 64 bytes write 000001 1byte read follow 1 byte write 000010 2 bytes read follow 2 bytes write .... 111111 63 bytes read follow 63 bytes write

37.4.5.7 Channel Request Time-Out Registers (RTOR0–RTOR15)

RTOR set the time-out for DMA Request from the channel's selected request source, which detects any discontinuity of data transfer. The request time-out takes effect only when the corresponding request enable (REN) bit in the Channel Control Register (CCR) is set. An Internal Request Time-out Counter starts counting when DMA channel is enabled and burst on that channel ends. Internal Request Time-out Counter is reset to zero when another burst for that channel starts. When counter reaches the count value set in this register, it asserts an interrupt (if it is not masked) and set its error bit in the DMA request

time-out status register (RTOSR). The input clock of the counter is selectable either from the system clock (HCLK) or input crystal (CLK32K).

Internal Request Time-out Counter will not generate an error status (or count) for the first burst of a DMA cycle. It can be programmed to count (and can generate an error status as described above) for all other bursts in the DMA cycle.

NOTE

This register shares the same address as the bus utilization control register.

0x1000_1098 (RTOR0) Access: User Read/Write
–
0x1000_1458 (RTOR15)

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EN	CLK	PSC	CNT												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-24. Channel Request Time-Out Registers (RTOR0–RTOR15)

Table 37-22. Channel Request Time-Out Registers Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15 EN	Enable. Enables/Disables the DMA request time-out. 0 Disables DMA request time-out 1 Enables DMA request time-out
14 CLK	Clock Source. Selects the counter of input clock source. 0 HCLK 1 32.768 kHz
13 PSC	Prescaler Count. Sets the prescaler of input clock. 0 Divide by 1 1 Divide by 256
12–0 CNT	Request Time-Out Count. Contains time-out count down value for internal counter in number of clocks. This value remains unchanged through out the DMA cycle.

37.4.5.8 Channel Bus Utilization Control Registers (BUCR0–BUCR15)

BUCR controls the bus utilization of an enabled channel when REN bit in Channel Control Register (CCR) is cleared. The channel does not request a DMA transfer until the internal bus_untilization_counter reaches

the count value set in this register except for the very first burst. This counter is cleared when the channel burst is started. When this count value is set to zero, DMA carries on burst transfers one after another until it reaches the value set in Channel Count Register (CNTR). In this case, user must be careful not to violate the maximum bus request latency of other devices.

NOTE

The BUCR0–BUCR15 registers share the same address as the Channel Request Time-Out (RTOR0–RTOR15) Registers.

0x1000_1098 (BUCR0) Access: User Read/Write
–
0x1000_1458 (BUCR15)

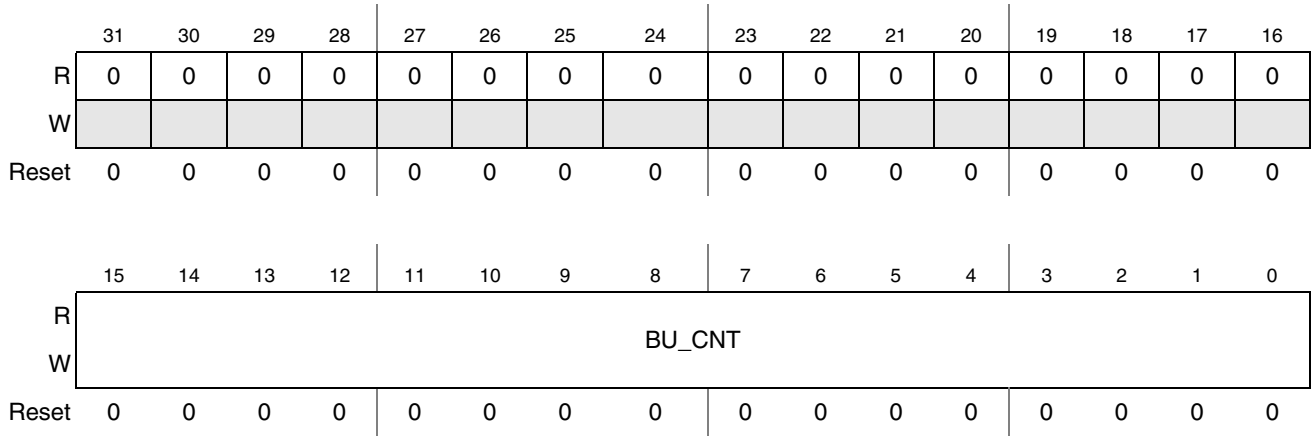


Figure 37-25. Channel Bus Utilization Control Register (BUCR0–BUCR15)

Table 37-23. Channel Bus Utilization Control Register Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should read 0.
15–0 BU_CNT	Bus Utilization Clock Count. Sets the number of system clocks that must occur before the channel starts the next burst.

37.4.5.9 Channel Counter Registers (CCNR0–CCNR15)

The CCNR indicates the number of bytes transferred for the channel. It is reset to zero after channel is enabled and keeps incrementing for each transfer during the DMA burst. This counter will retain its value after the channel is disabled, till it is enabled again. If RPT bit is found set at the end of last burst of the DMA cycle, this counter retains its value and will be reset to zero only at the start of another DMA burst, that is, the first burst of new DMA cycle. If a DMA channel is disabled before completion of DMA cycle, this counter will retain the value of the number of data transferred in that DMA cycle. When the peripheral responds with a error response during a DMA data transfer, CCNR value will not be increment for that AHB cycle as no data was transferred in that cycle.

0x1000_109C (CCNR0) Access: User Read/Write
–
0x1000_145C (CCNR15)

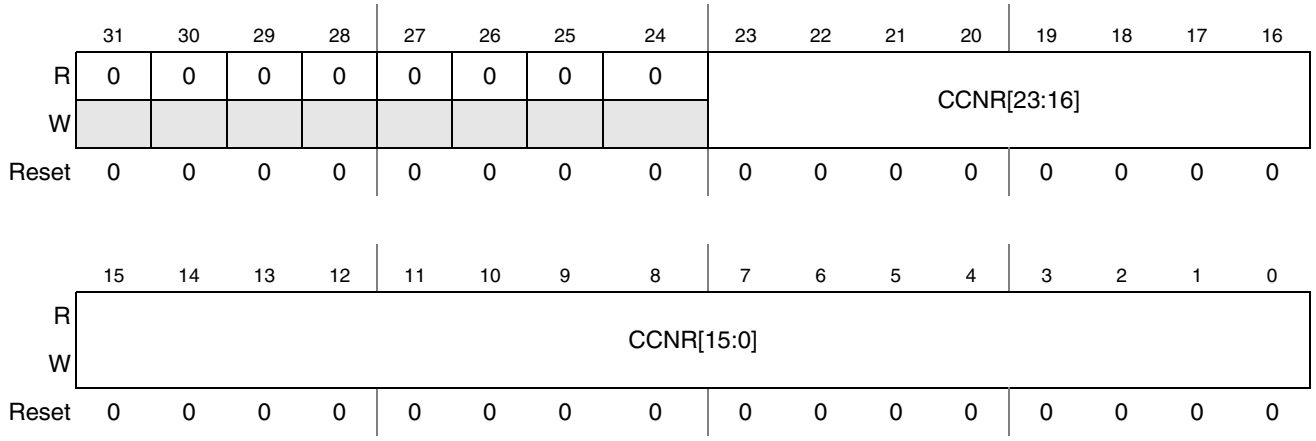


Figure 37-26. Channel Counter Register (CCNR0–CCNR15)

Table 37-24. Channel Counter Register Field Descriptions

Field	Description
31–24	Reserved. These bits are reserved and should read 0.
23–0 CCNR	Channel Counter. Indicates the number of bytes transferred for the channel.

37.5 DMA Chaining

DMA chaining refers to using the same DMA Channel to automatically transfer a second data buffer (possibly of a different length) between another two sets of Source and Destination addresses, with an increase in the allowable value of interrupt service time. This is possible because the ISR execution (that is, the setup for next transfer) can occur in parallel to the next buffer transfer from the DMA (when RPT bit is set). To achieve DMA Chaining: the SAR, DAR and CNTR for each Channel are double buffered internally. With this, the Host can update these three register values during an ongoing DMA Transfer for the same channel in preparation for the next DMA transfer. With the use of RPT and ACRPT bits, the second transfer can occur for different source, destination addresses and different amount of data.

As an example, consider a Data Transfer of 14 Kbytes from memory to a FIFO using 4 Kbyte buffers.

- Driver writes 4 Kbyte of data into buffer 1, sets source register to buffer 1 and count to 4 Kbyte, sets ACRPT, then enables the transfer. DMA hardware will immediately latch the registers and start the transfer.
- Driver immediately writes 4 Kbyte of data into buffer 2, sets the same source register to buffer 2, count to 4 Kbyte, and sets the RPT bit.
- Transfer of buffer 1 completes, DMA hardware samples the RPT bit, finds it set, latches the register (now set for buffer 2), clears the RPT bit because ACRPT is set, and starts the next transfer.
- It then generates the 1st interrupt.

- Driver ISR writes 4 Kbyte of new data to buffer 1, sets the source register to buffer 1 and count to 4 Kbyte, and sets the RPT bit again.
- Transfer of buffer 2 completes, DMA hardware samples RPT bit, finds it set, latches the registers (now set for buffer 1), clears the RPT bit because ACRPT is still set, and starts the next transfer.
- It then generates the 2nd interrupt.
- Driver ISR writes 2 Kbyte of new data to buffer 2, sets the source register to buffer 2 and count to 2 Kbyte, and sets the RPT bit again.
- Transfer of buffer 1 completes, DMA hardware samples the RPT bit, finds it set, latches the register (now set for buffer 2), clears the RPT bit because ACRPT is still set, and starts the next transfer.
- It then generates the 3rd interrupt.
- Driver ISR has no more data to send so does nothing.
- Transfer of buffer 2 completes, DMA hardware samples the RPT bit, finds it clear so it stops the transfer.
- It then generates the 4th interrupt.
- Driver ISR disables the DMA and the transfer is complete.

37.6 Special Cases of Burst Length and Access Size Settings

DMA burst length should normally be programmed as a multiple of source and destination access sizes. The following sections discuss the behavior that occurs when burst length is not a multiple of access size.

37.6.1 Memory Increment

The following are the possible adverse effects:

1. Unknown data can be written at some locations, however, there is no data loss.
2. Number of bytes transferred can be more than the count value set.

These effects are explained in the examples below:

Example 1: Source is Linear memory with access size of 1 byte. Destination is linear memory with access size of 2 bytes. Burst Length is programmed as 3 bytes with memory increment. Source Address Register: 0x0000_1000. Destination Address Reg: 0x0000_2000. For the first burst, DMA would read 3 bytes from addresses: 1000, 1001, and 1002. During the write cycle of first burst, DMA would write 2 bytes each at addresses 2000 and 2002. One extra memory location (0x2003) is written with unknown data from the DMA internal FIFO (8’h00 after hardware reset).

Example 2: Source is Linear Memory with access size of 2 bytes. Destination is linear memory with access size of 2 bytes. Burst Length is programmed as 3 bytes with memory increment. Source Address Register: 0x0000_1000. Destination Address Reg: 0x0000_2000. For the first burst, DMA would read 2 bytes each from addresses 1000 and 1002. During the write cycle of first burst, DMA would write 2 bytes each at addresses 2000 and 2002. One extra data byte is transferred per burst. When programmed with a count of say 9 bytes, DMA would perform data transfer of 12 bytes.

37.6.2 Memory Decrement

Possible Adverse Effects:

1. Unknown data can be written at some locations. In certain cases there can be data loss.

Example: Source is linear memory with access size of 1 byte. Destination is linear memory with access size of 2 bytes. Burst Length is programmed as 3 bytes with memory decrement. Source Address Register: 0x0000_1000. Destination Address Reg: 0x0000_2000. For the first burst, DMA would read 3 bytes from addresses: 0FFD, 0FFE, and 0FFF. During write cycle of the first burst, DMA would write 2 bytes each at addresses 1FFC and 1FFE. An extra byte is written at the address 1FFF with unknown data from the DMA internal FIFO (8'h00 after hardware reset).

For the second burst, DMA would read 3 bytes from the addresses: 0FFA, 0FFB, and 0FFC. During write cycle of the second burst, DMA would write 2 bytes each at addresses 1FFA and 1FFC. In this case data written at 1FFC and 1FFD in the first burst have been overwritten. Data written at 1FFD will be unknown data from the DMA internal FIFO (8'h00 after hardware reset).

NOTE

In the case of 2D Memory writing extra bytes would mean writing beyond the limits of X-Size programmed in a row. Similarly for linear memory, this can lead data overflowing the allocated buffer for DMA.

37.7 Special Cases When CCNR and CNTR Values Differ

There are two combinations of events that can cause the values of CCNR and CNTR to differ. This situations are discussed in detail in the following sections.

37.7.1 CNTR Not A Multiple of Destination Access Size

If CNTR register value is not a multiple of destination access size, then CCNR value will not match the value programmed in CNTR after completion of the DMA cycle for the channel. Table 37-25 illustrates the values of CCNR with different combinations of source and destination access sizes when CNTR = 5 bytes. This table holds good when BL = 3 bytes or BL = 4 bytes.

Table 37-25. CNNR Value Combinations

Source Size (Bytes)	Destination Size (Bytes)	No. of Bytes Read by DMA		No. of Bytes Written by DMA		CCNR (Bytes)
		Memory Increment	Memory Decrement	Memory Increment	Memory Decrement	
2	2	6	6	6	6	6
4	4	8	8	8	8	8
2	4	6	6	8	8	8
4	2	8	8	6	6	6
1	2	5	5	6	6	6
1	4	5	5	8	8	8

37.7.2 BL is Not a Multiple of Destination Access Size, CNTR Is

If BL register value is not a multiple of destination access size but CNTR is, then the value of CCNR will not match the value programmed in CNTR after completion of DMA cycle for the channel. Table 37-26 illustrates the values of CCNR with different combinations of source and destination access sizes when BL = 3 bytes and CNTR = 4 bytes.

Table 37-26. CCNR Value Combinations

Source Size (Bytes)	Destination Size (Bytes)	No. of Bytes Read by DMA		No. of Bytes Written by DMA		CCNR (Bytes)
		Memory Increment	Memory Decrement	Memory Increment	Memory Decrement	
2	2	6	6	6	6	6
4	4	8	8	8	8	8
2	4	6	6	8	8	8
4	2	8	8	6	6	6
1	2	4	4	6	6	6
1	4	4	4	8	8	8

NOTE

In case of memory decrement there might be some cases where DMA overwrites data written by itself so the number of bytes seen by the user can be different than those mentioned in the tables above.

37.8 Application Note

Following is the sequence to re-program a channel for data transfer:

1. Clear the status register bit corresponding to that channel (DISR, DBTOSR, DSESR, DRTOSR, DBOSR) after the DMA cycle is completed.
2. Change SMOD (source mode) to 2’b00 and clear CEN.
3. Re-program all registers corresponding to that particular channel, except CCR.
4. Program CCR and set CEN bit to 1.

NOTE

This sequence applies to all 16 channels in all modes: Linear memory, 2D memory, and FIFO.

37.9 DMA Burst Termination

DMA Controller needs to terminate its burst in case of:

- Transfer error response from slave
- Burst time-out error.
- Buffer overflow error.

- Channel disable (by software using CEN bit).

CAUTION

DMA burst termination may not occur immediately.

Burst termination occurs immediately in DMAC only on occurrence of Transfer Error response from the Slave. In other cases, Burst time-out, Buffer overflow and Channel disable (by software using CEN bit) takes about 2 more AHB transfers to terminate the burst after these are sensed. The burst termination is not done immediately to avoid AHB protocol violation.

In case the burst hangs and hready is not asserted for a large number of cycles, then this must be handled by the watchdog timer in the ABCD Module in i.MX27 device or by the system software.

37.10 Glossary of Terms Used

DMA burst	This refers to the burst cycles on the AHB Bus performed by the DMA.
DMA cycle	DMA cycle can consists of a number of DMA bursts depending on the Channel Burst Length and Channel Count register settings. for example, if BL = 4 and CNTR = 8, then DMA cycle will consist of 2 DMA bursts.

Book II, Part 7: Multimedia Peripherals

Introduction

CMOS Sensor Interface (CSI)

This section presents the CMOS Sensor Interface (CSI) on the architecture, operation principles, and programming model. The CSI is a logic interface which enables the i.MX27 to directly connect to external CMOS sensors and CCIR656 video source.

Digital Audio MUX (AUDMUX)

The Digital Audio MUX (AUDMUX) provides a programmable interconnect fabric for voice, audio, and synchronous data routing between host serial interfaces (for example, SSI or SAP) and peripheral serial interfaces (for example, audio and voice CODECs). The AUDMUX allows the audio system connectivity to be modified through programming (as opposed to altering the PCB schematics of the system). The Digital Audio MUX is configured by software.

With the AUDMUX, resources do not need to be hard-wired and can be effectively shared in different configurations. The AUDMUX interconnections allow multiple, simultaneous audio/voice/data flows between the ports in point-to-point or point-to-multipoint configurations.

The AUDMUX includes two types of interfaces. Internal ports connect to the processor serial interfaces and external ports connect to off-chip audio devices and serial interfaces of other processors. A desired connectivity is achieved by configuring the appropriate internal and external ports.

enhanced Multimedia Accelerator light (eMMA_lt)

The *enhanced* Multimedia Accelerator light (eMMA_lt) consists of the video Pre-Processor (PrP) and Post-Processor (PP), similar functionalities with original eMMA which also includes Mpeg4 Encoder

(EN) and Decoder (DE). These blocks work together to provide video acceleration and off-load the CPU from computation intensive tasks. The PrP and PP can be used for generic video pre and post processing such as scaling, resizing, and color space conversions.

Synchronous Serial Interface (SSI)

The SSI is a full-duplex, serial port that allows the chip to communicate with a variety of serial devices. These serial devices can be standard CODECs, Digital Signal Processors (DSPs), microprocessors, peripherals, and popular industry audio CODECs that implement the inter-IC sound bus standard (I2S) and Intel AC97 standard.

SSI is typically used to transfer samples in a periodic manner. The SSI consists of independent transmitter and receiver sections with independent clock generation and frame synchronization.

The SSI contains independent (asynchronous) or shared (synchronous) transmit and receive sections with separate or shared internal/external clocks and frame syncs, operating in Master or Slave mode. The SSI can work in normal mode operation using frame sync and in Network mode operation allowing multiple devices to share the port with as many as thirty-two time slots. The SSI provides 2 sets of Transmit and Receive FIFOs. Each of the four FIFOs is 8x24 bits. The two sets of Tx/Rx FIFOs can be used in Network mode to provide 2 independent channels for transmission and reception. It also has programmable data interface modes such like I2S, LSB, MSB aligned and programmable word lengths. Other program options include frame sync and clock generation and programmable I2S modes (Master, Slave or Normal). Oversampling clock, ccm_ssi_clk available as output from SRCK in I2S Master mode.

In addition to AC97 support the SSI has completely separate clock and frame sync selections for the receive and transmit sections. In AC97 standard, the clock is taken from an external source and frame sync is generated internally. the SSI also has a programmable internal clock divider and Time Slot Mask.

Liquid Crystal Display Controller (LCDC)

The Liquid Crystal Display Controller (LCDC) provides display data for external gray-scale or color LCD panels. The LCDC is capable of supporting black-and-white, gray-scale, passive-matrix color (passive color or CSTN), and active-matrix color (active color or TFT) LCD panels.

Smart Liquid Crystal Display Controller (SLDC)

The Smart Liquid Crystal Display Controller module transfers data from the display memory buffer to the external display device. Direct Memory Access (DMA) transfers the data transparently with minimal software intervention. Bus utilization of the DMA is controllable and deterministic.

Chapter 38
Digital Audio MUX (AUDMUX)

The Digital Audio MUX (AUDMUX) provides a programmable interconnect fabric for voice, audio, and synchronous data routing between the i.MX27 device SSI modules and an external SSI, or between audio and voice codecs. With the AUDMUX, resources do not need to be hard-wired and can be effectively shared in different configurations. The AUDMUX interconnections allow multiple simultaneous separate audio/voice/data flows between the ports in a point-to-point or point-to-multipoint configuration.

38.1 Features

The Digital Audio MUX offers the following features:

- Three host interfaces (two internal and one external)
- Three peripheral interfaces (All external)
- Full 6-wire SSI interfaces for asynchronous receive and transmit
- Configurable 4-wire, synchronous or 6-wire asynchronous Rx and Tx external host and peripheral interfaces
- Independent Frame sync and clock direction selection for host or peripheral. Clock direction selection to function as master of the flow
- Each host interface can be connected to any other host or peripheral interface in a point-to-point or point-to-multipoint (network mode)
- Transmit and Receive Data switching to support external network mode

38.2 Overview

Figure 38-1 shows the block diagram of the AUDMUX. On the left of the illustration are the internal interfaces and on the right, the external interfaces. Port 1 and Port 2 are internally connected to SSI-1 and SSI-2, respectively, and Port 3 has special muxes allowing connection to an external SSI, such as a synchronous audio port (SAP) commonly found on a baseband modem. Ports 4–6 are identical and can be connected to any 4-wire or 6-wire SSI, voice, I^2S or AC97 codec. Ports 1–3 are also known as host ports, and Ports 4–6 as peripheral ports.

Port 1–Port 6 have configurable 4-wire or 6-wire interfaces. When configured as a 6-wire interface, the additional RFS and RCLK signals of the interface enable the SSIs to be used in asynchronous mode with separate receive and transmit clocks. In this mode, a device at one port can be connected to two ports (internal or external) configured as input only (simplex) and output only (simplex). Ports 1–3 have muxing arrangements to support internal network mode. Ports 3–6 have a Tx/Rx switch to support external network mode. The Tx/Rx switch enables the Da and Db lines to be swapped so that more than one master connected to any of Ports 1–3 can communicate to more than one slave externally attached at the external ports.

Bit Clock selection direction enables each port to be configured as a master or slave in the flow.

Possible scenarios are as follows:

1. SSI1 (Internal host port) drives voice codec and BT (on External Peripheral Port 6) and the Bottom Connector (on External Peripheral Port 5) simultaneously using network mode. SSI1 is the master.
2. SAP (External audio port from Baseband) drives voice codec and BT (on Port 6) and the Bottom Connector (on Port 5) simultaneously using network mode. SAP is the master.

NOTE

The first scenario supports External Network mode when SSI1 provides the corresponding Output Enables for TxData, and the Slave devices receiving the data must be configured to only receive in their corresponding time slot.

In the second case, any slave devices attached locally to the SAP in network mode must be disabled to access slave devices on the other ports (for example, SSI, voice codec, and/or BT). Frame Sync and Bit Clock selections enable each port to be configured as a master or slave in the flow.

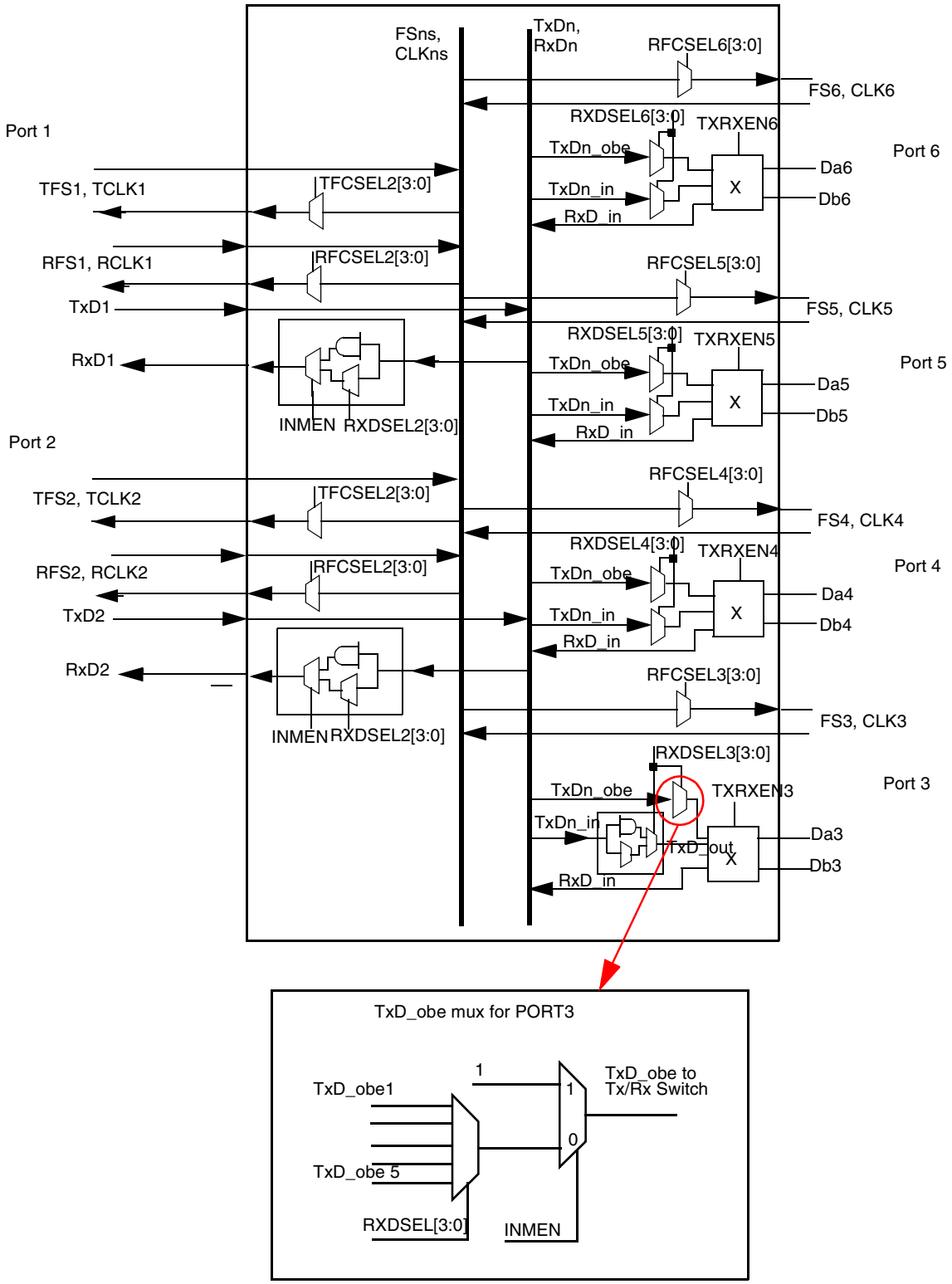


Figure 38-1. AUDMUX Block Diagram

38.3 Internal Network Mode

Figure 38-2 shows the internal network mode selection logic. Network mode is where a master SSI is connected to more than one slave SSI device and communication occurs on a time-slotted frame. Though network mode allows communication between master-slave and slave-slave, the internal network mode supports only master-slave network mode.

In internal network mode (INMEN=1), the output of the AND gate is routed to the output of the port and to the RxD signal of the SSI. The INMMASK bit vector selects the transmit signals of the ports that are to be connected in network mode. The transmit signals (TxD_in from SSI and RxD_in from external ports) are ANDed together to form the output. In network mode, only one device can be transmitting in its predesignated timeslot and all other transmit signals will remain high (tri-stated and pulled-up), hence non-active signals in the selection will be high and do not influence the output of the AND gate.

In normal mode (INMEN=0), the SSI is connected in point to point (as a master or slave) and the RXDSEL[2:0] settings select the transmit signal from the other ports. Internal network mode can be used with external network as long as slave-only devices are attached in external network mode at a port or in. Internal network mode can also be used with external network mode if all slave devices connected to a master in external network mode are disabled.

Figure 38-2 shows the connections for Port 1.

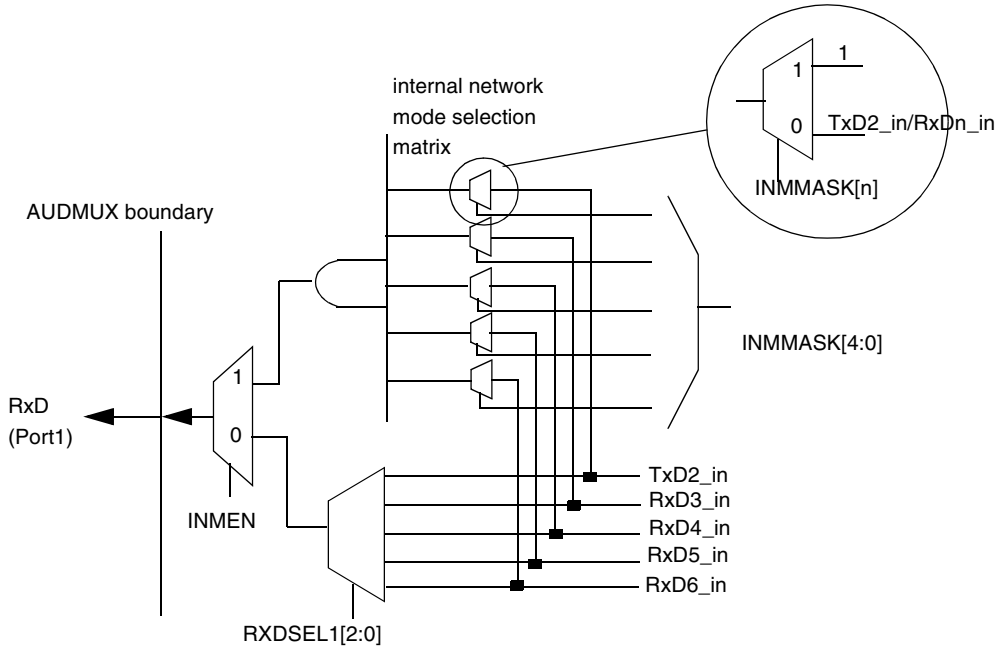


Figure 38-2. Internal Network Mode

38.4 Tx/Rx Switch and External Network Mode

External network mode is the traditional network mode connection. It is called external network mode to differentiate from the internal network mode. In external network mode, devices are connected to the external ports in a star or multidrop configuration.

In network mode, there can be only one master (frame sync and clock source) and the other devices are configured in normal slave mode or network slave mode. Unlike internal network mode, in external network mode both master-slave and slave-slave communication can take place. Codec devices transmit on a single timeslot and SSIs in network master (for example SAP) or slave mode (SSI-2) can process more than one timeslot of data.

Figure 38-3 shows the Tx/Rx switch. TxD_obe is the output buffer enable signal and TxD_out is the data transmit signal from the internal SSI. The RxD_in signal is the receive data signal going towards the RXDSEL and TXDSEL muxes of the SSI ports and external ports.

In normal mode and network slave mode, TXRXEN is disabled (TXRXEN=0) and TxD_out is routed to Da (Da_out) and Db (Db_in) is routed to RxD_in. In normal mode, the output buffer enable, Da_obe is always enabled (asserted) and TxD_out is routed to Da_out. In network mode, the TxD_obe signal is enabled during the SSI's timeslot(s) and the Da output is tri-stated in other timeslots.

In network mode (SSIx as master), the Tx/Rx switch is enabled (TXRXEN=1) and TxD_out is routed to Db_out and Da_in is routed to RxD_in. The TxD_obe signal is enabled during the SSI's timeslot(s) and the Db output is tristated in other timeslots.

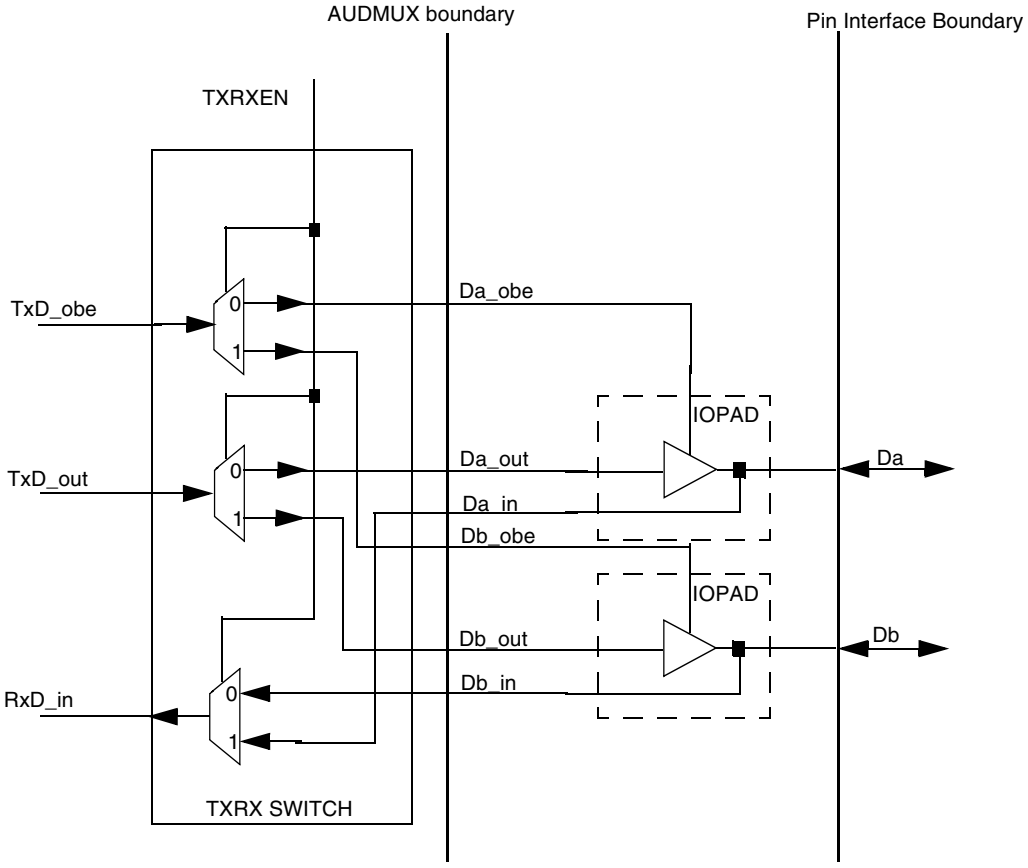


Figure 38-3. Tx/Rx Switch

38.5 Frame Sync and Clocks

The routing of frame syncs and interface clocks are shown in Figure 38-4.

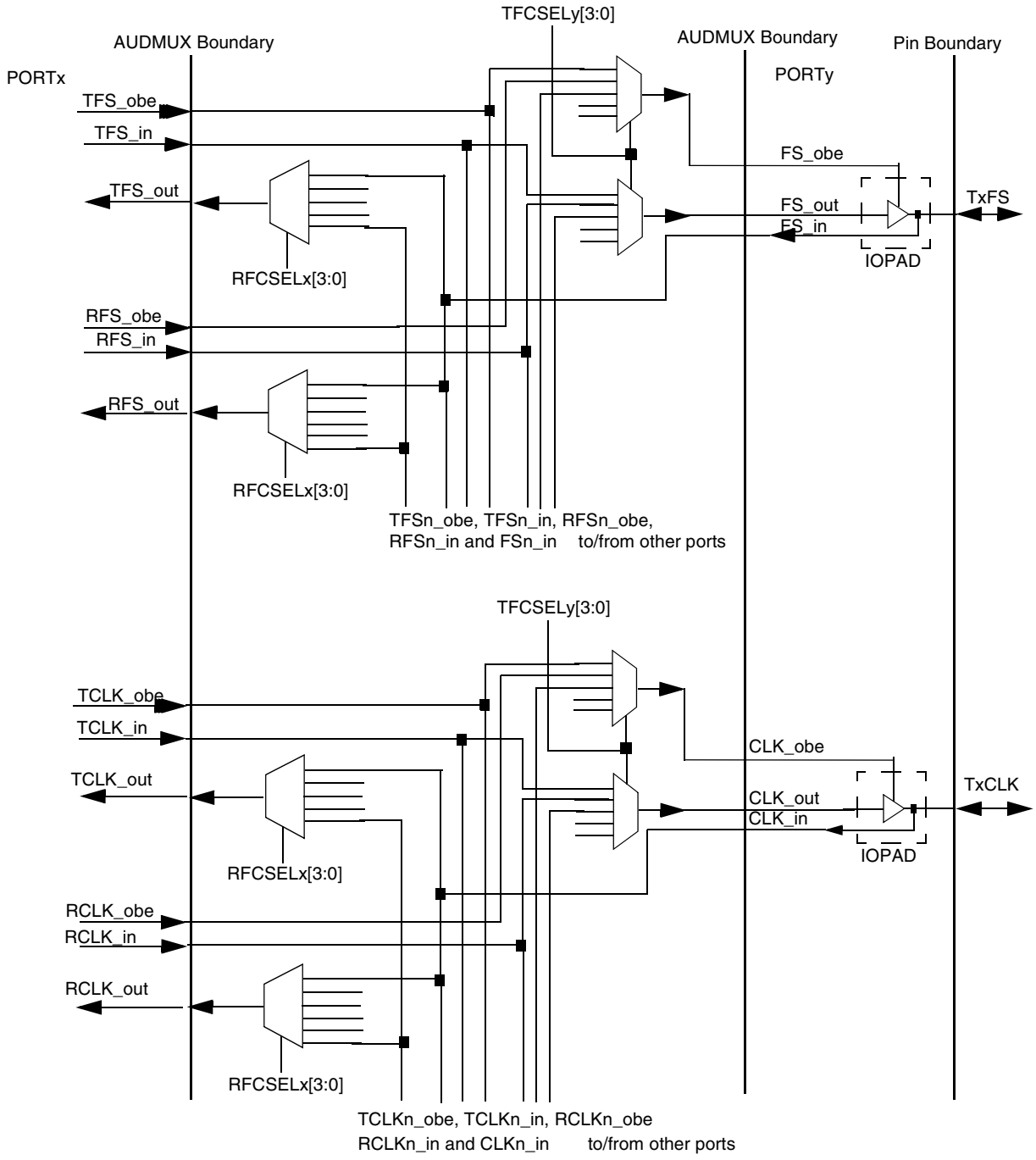


Figure 38-4. Frame Sync and Clock Routing when Peripheral Port is 4-Wire

38.6 Synchronous Mode (4-Wire Interface)

In Synchronous mode the port will have a 4-wire interface—that is, RXD,TXD,TxCLK,TxFS.The Receive clock and the receive frame sync will be the same as Transmit clock (TxCLK) and Transmit frame sync (TxFS), respectively.

As shown in Figure 38-4, PORTx signals can be routed to PORTy, showing a 6-wire to 4-wire port connectivity.

TFS_in, RFS_in, TCLK_in, and RCLK_in are input Frame Sync and Bit Clocks from the SSI with their corresponding output buffer enable signals (_obe). TFS_out, RFS_out, TCLK_out, and RCLK_out are the Frame Sync and Bit Clocks that are transmitted to the SSI from the other ports.

TFS_out and TCLK_out are selected by the TFCSEL MUX settings and RFS_out and RCLK_out are selected by the RFCSEL MUX settings. Similarly, in the external direction, the TFCSEL selects the FS_obe and FS_out signals. In this mode RFCSEL is not used.

38.7 Asynchronous Mode (6-Wire Interface)

In Asynchronous mode the port will have a 6-wire interface—that is, RXD, TXD, TxCLK, TxFS, RxCLK, RxFS. There will be additional Receive clock (RxCLK) and the frame sync (RxFS) pins as compared to the Synchronous or 4-wire interface.

Refer to Figure 38-5 and Figure 38-6, PORTx signals can be routed to PORTy, depicting a 6-wire to 6-wire port connectivity.

TFS_in, RFS_in, TCLK_in and RCLK_in are input Frame Sync and Bit Clocks from the SSI (PORTx) with their corresponding output buffer enable signals (_obe). TFS_out, RFS_out, TCLK_out, and RCLK_out are the Frame Sync and Bit Clocks that are transmitted to the SSI from the other ports.

TFS_out and TCLK_out are selected by the TFCSEL MUX settings and RFS_out and RCLK_out are selected by the RFCSEL MUX settings. Similarly, in the external direction, the TFCSEL selects the TxFS_obe, TxFS_out and TxCLK_obe, TxClk_out signals. The RFCSEL selects the RxFS_obe and RxFS_out and RxCLK_obe, RxCLK_out.

NOTE

Noticed that because FS_in and CLK_in from external interfaces are also routed to the TFCSEL muxes of the external ports, these signals do not have corresponding buffer enable signals. Consequently, their corresponding inputs to the TFCSEL MUX of the external ports must be tied high.

38.8 SSI to Peripheral Connection

The Figure 38-7 shows the data path interconnections between an internal SSI port and a peripheral port. TxD_obe is the buffer enable signal from the SSI, TxD_in, the input transmit data from the SSI and RxD_out, the receive data output from the AUDMXUX to the SSI.

TXDSEL[2:0] of the peripheral port, selects the buffer enable signal (TxD_obe) and transmit data output (TxD_out) signal from the TxD_obe and TxD_in and RxD_in signals. TXDSEL[2:0] is a common signal to both selection muxes.

NOTE

Because RxD_in signals from external interfaces do not have their buffer enable signals, their corresponding buffer enable signals into the selection MUX should be tied to high. This will ensure that selection of RxD_in as TxD_out will also drive the TxD_obe output high.

Transmit Data from the SSI goes into the TXDSEL data MUX and comes out as TxD_out and is routed to Da_out when TXRXEN is disabled and to Db_out when TXRXEN is enabled. Similarly, Db_in is routed to RxD_in when TXRXEN is disabled and Da_in is routed to RxD_in when TXRXEN is enabled. If The routing of frame syncs are shown in Figure 38-5 and the routing of interface clocks are shown in Figure 38-6.

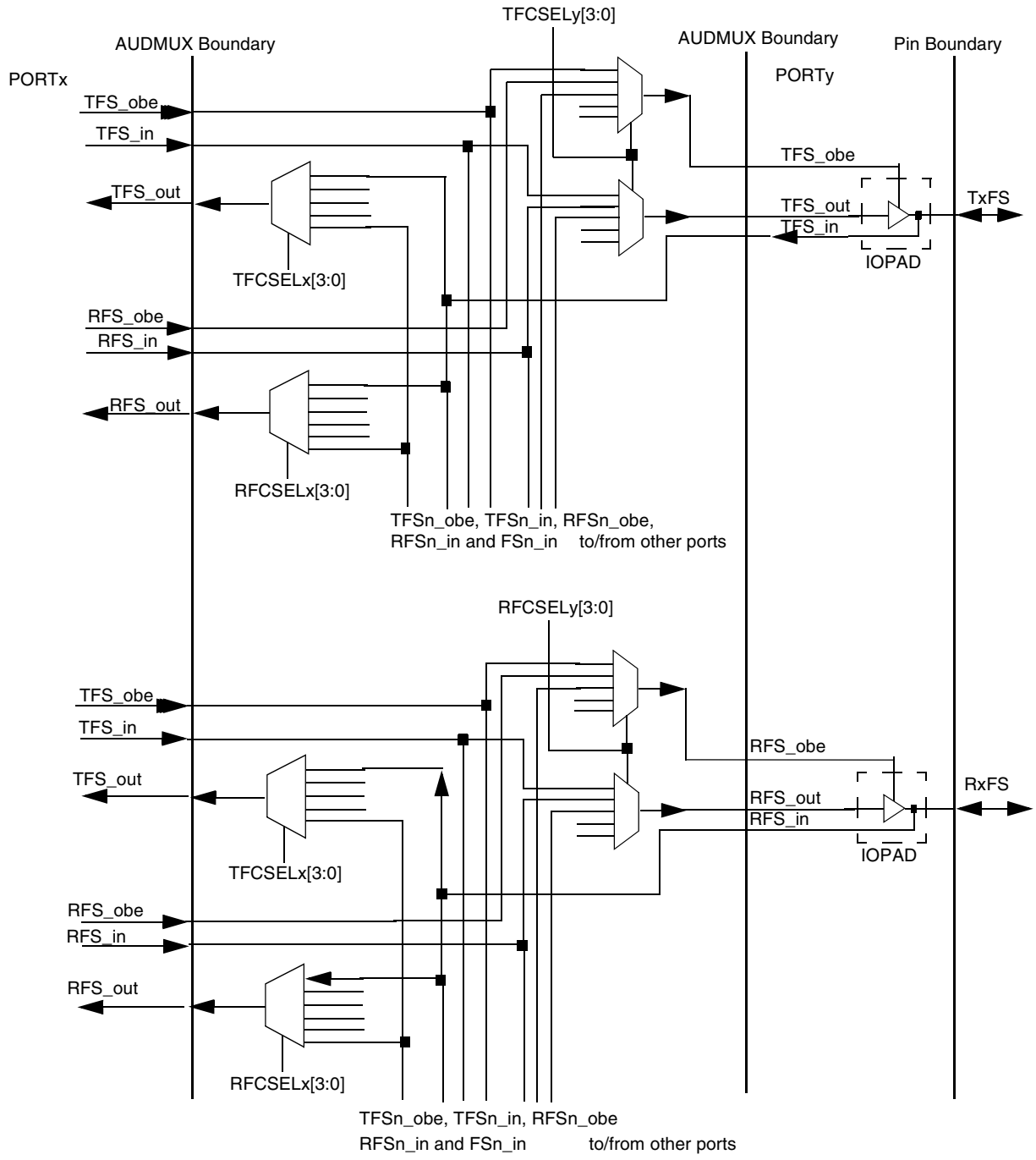


Figure 38-5. Frame Sync Routing when Peripheral Port is 6-Wired

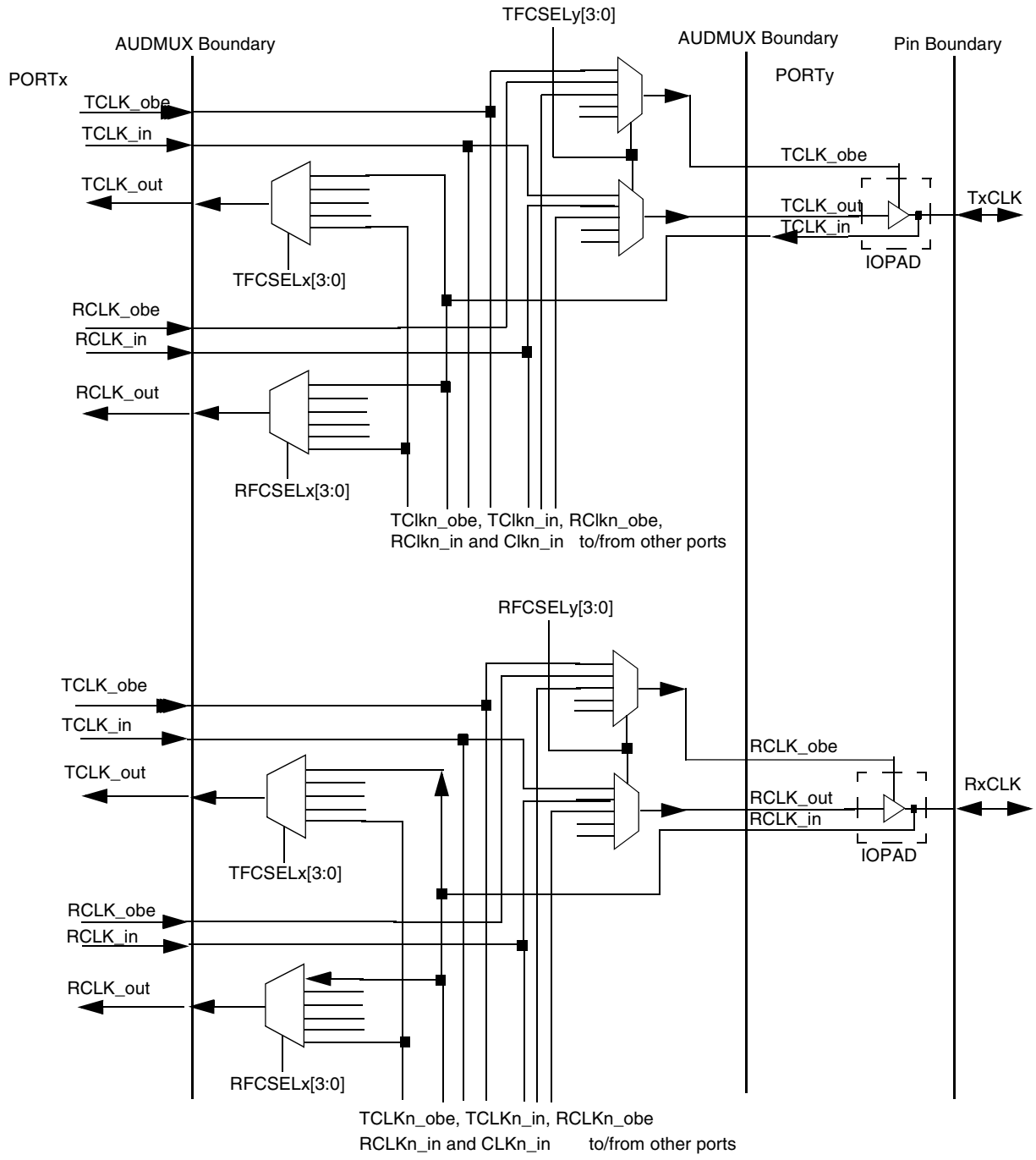


Figure 38-6. Clock Routing when Peripheral Port is 6-Wired

If internal network mode is disabled, then RXDSEL selects the RxD_in which is then output from the AUDMUX to the SSI. When internal network mode is selected, the RxD_in is ANDed with other TxD_in and RxD_in signals from other ports before output as RxD_out to the SSI.

If there are more than one device attached to the external port at Da and Db interfaces and one of the devices is a network master, then two conditions have to be noted:

a) When the external master is enabled in network mode, then the SSI must be configured as slave (normal or network mode). No Tx/Rx switching is required.

b) When the external master is disabled and the SSI and other slave devices require to communicate, then the SSI must be configured as network mode master and the Tx/Rx switch must be enabled (TXRXEN=1). This ensures that the transmit and receive paths are connected appropriately.

To communicate with more than one port, internal network mode must be enabled at the SSI port. In internal network mode, it is possible to communicate with any device attached to the other ports. Internal network mode must be enabled at the port that is the SSI network mode master.

38.9 SSI to SAP

The Figure 38-8 shows the detailed interconnection of a SSI port to a SAP port. The SSI and SAP port can act as masters or slaves and be configured in normal or network mode. The SAP port can communicate with more than one external device attached to its own interface in external network mode and additionally with one device attached to another AUDMUX port.

If the SAP must communicate with more than one port, then the internal network mode must be enabled.

38.10 Peripheral Port to Peripheral Port

Peripherals attached to the AUDMUX can communicate with each other in 2 ways:

a) One peripheral acts is configured as master and which sources the clock and/or the frame sync and the other peripheral is configured as slave.

b) Both peripherals are configured as slaves but with data routing established from external port to external port with one additional port configured as master (SSI or SAP), which sources the frame sync and clock to the two ports.

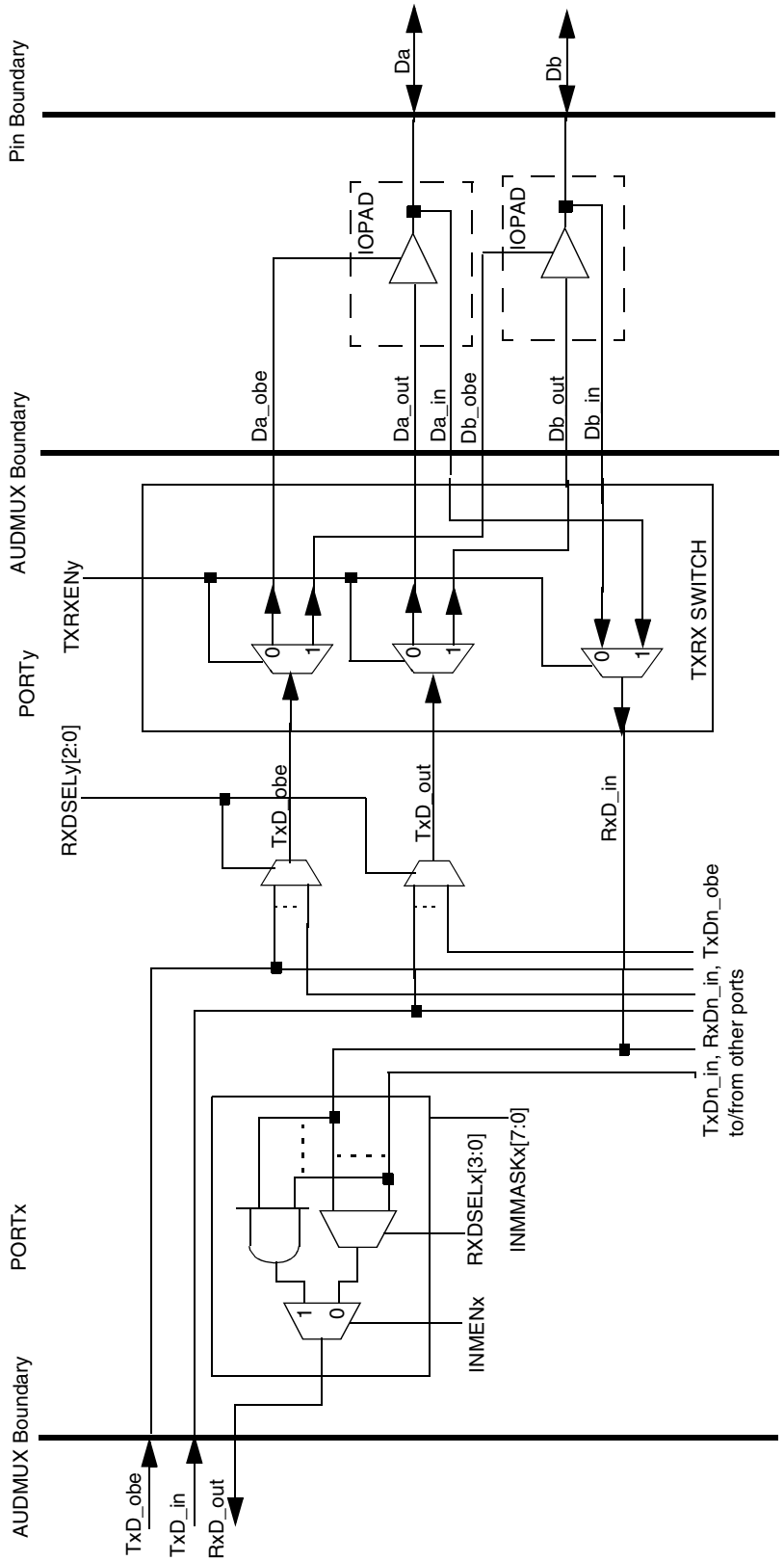


Figure 38-7. SSI to Peripheral Port Interconnection

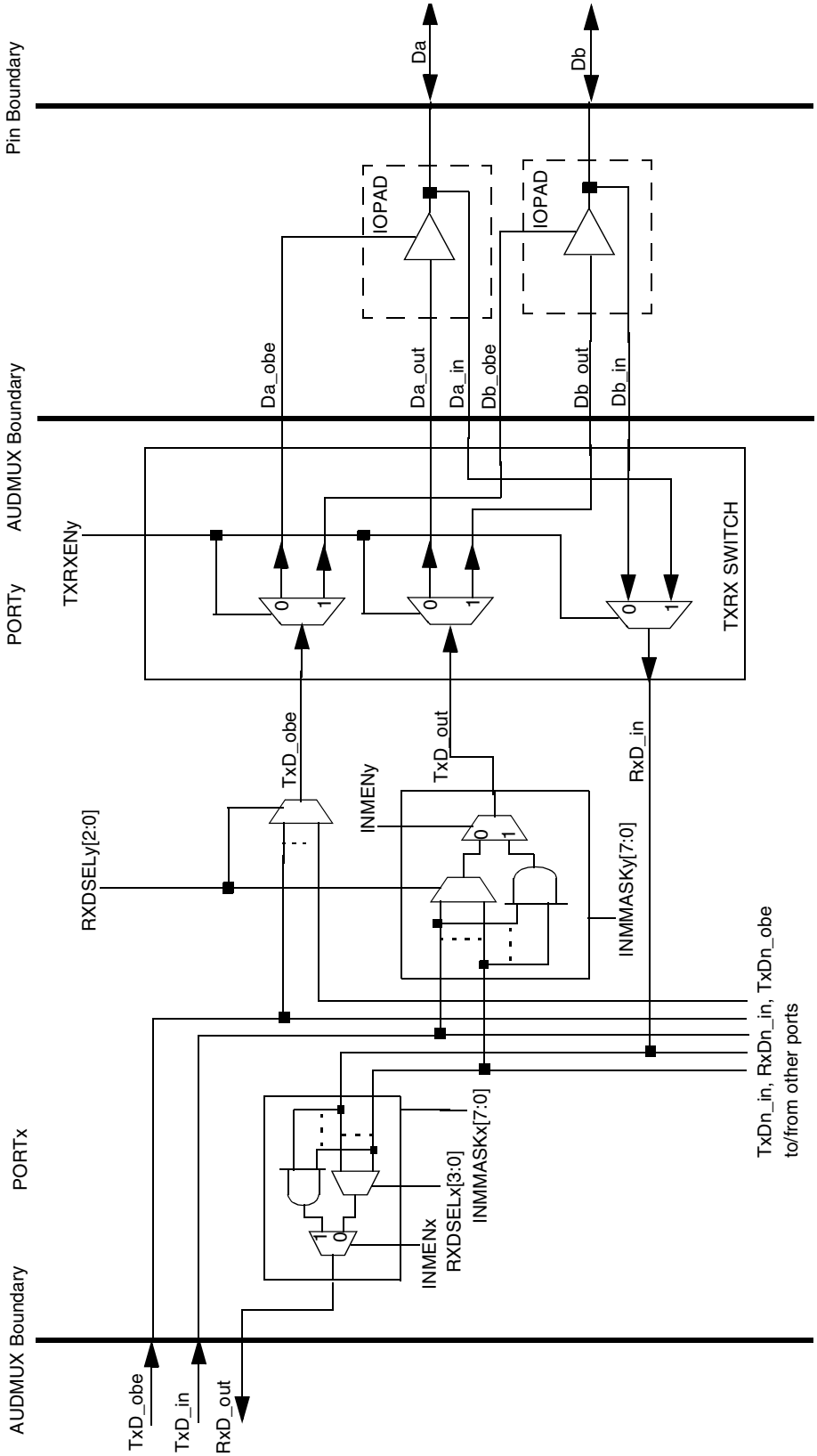


Figure 38-8. SSI to SAP Interconnection

38.11 Memory Map and Register Definition

There is one configuration register per host port and per peripheral port. The AUDMUX has a total of 6 registers. Table 38-3 shows the control register summary and address mapping for AUDMUX. The base address is 0x1001 6000.

38.11.1 AUDMUX Memory Map

Table 38-1 shows the AUDMUX memory map.

Table 38-1. AUDMUX Memory Map

Address	Use	Access	Reset Value	Section/Page
0x1001_6000 (HPCR1)	Host Port Configuration Register 1	R/W	0x0000_0000	38.11.3/38-16
0x1001_6004 (HPCR2)	Host Port Configuration Register 2	R/W	0x0000_0000	38.11.3/38-16
0x1001_6008 (HPCR3)	Host Port Configuration Register 3	R/W	0x0000_0000	38.11.3/38-16
0x1001_6010 (PPCR1)	Peripheral Port Configuration Register 1	R/W	0x0000_1000	38.11.4/38-18
0x1001_6014 (PPCR2)	Peripheral Port Configuration Register 2	R/W	0x0000_1000	38.11.4/38-18
0x1001_601C (PPCR3)	Peripheral Port Configuration Register 3	R/W	0x0000_1000	38.11.4/38-18

38.11.2 Register Summary

Figure 38-9 shows the key to the register fields, and Table 38-2 shows the register figure conventions.

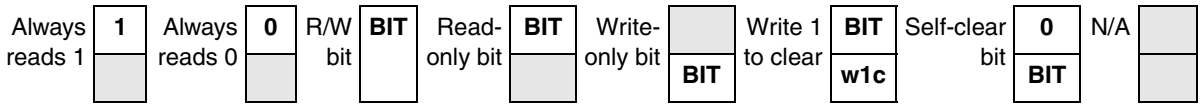


Figure 38-9. Key to Register Fields

Table 38-2. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.

Table 38-2. Register Figure Conventions (continued)

Convention	Description
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 38-3 shows the AUDMUX register summary.

Table 38-3. AUDMUX Register Summary

Name		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
0x1001_6000 (HPCR1)	R	TFS DIR	TC LK DIR	TFCSEL[3:0]				RFS DIR	RCLKDIR	RFCSEL[3:0]				0	0	0	0
	W																
	R	RXDSEL[2:0]			SYN	0	0	0	INMEN	INMMASK[7:0]							
	W																
0x1001_6004 (HPCR2)	R	TFS DIR	TC LK DIR	TFCSEL[3:0]				RFS DIR	RCLKDIR	RFCSEL[3:0]				0	0	0	0
	W																
	R	RXDSEL[2:0]			SYN	0	0	0	INMEN	INMMASK[7:0]							
	W																
0x1001_6008 (HPCR3)	R	TFS DIR	TC LK DIR	TFCSEL[3:0]				RFS DIR	RCLKDIR	RFCSEL[3:0]				0	0	0	0
	W																
	R	RXDSEL[2:0]			SYN	0	TXR XEN	0	INMEN	INMMASK[7:0]							
	W																
0x1001_6010 (PPCR1)	R	TFS DIR	TC LK DIR	TFCSEL[3:0]				RFS DIR	RCLKDIR	RFCSEL[3:0]				0	0	0	0
	W																
	R	RXDSEL[2:0]			SYN	0	TXR XEN	0	0	0	0	0	0	0	0	0	0
	W																
0x1001_6014 (PPCR2)	R	TFS DIR	TC LK DIR	TFCSEL[3:0]				RFS DIR	RCLKDIR	RFCSEL[3:0]				0	0	0	0
	W																
	R	RXDSEL[2:0]			SYN	0	TXR XEN	0	0	0	0	0	0	0	0	0	0
	W																

Table 38-3. AUDMUX Register Summary (continued)

Name		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
0x1001_601C (PPCR3)	R	TFS DIR	TC LK DIR	TFCSEL[3:0]				RFS DIR	RCLKDI R	RFCSEL[3:0]				0	0	0	0
	W																
	R	RXDSEL[2:0]			SY N	0	TXR XEN	0	0	0	0	0	0	0	0	0	0
	W																

38.11.3 Host Port Configuration Register (HPCR1–2)

There is one Host Port Configuration Register (HPCR) for each host port.

0x1001_6000 (HPCR1)
0x1001_6004 (HPCR2)
0x1001_6008 (HPCR3)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TFS DIR	TCLK DIR	TFCSEL				RFS DIR	RCLK DIR	RFCSEL				0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDSEL			SYN	0	TXRX EN	0	INME N	INMMASK							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-10. Host Port Configuration Register (HPCR1–2)

Table 38-4. Host Port Configuration Register Field Descriptions

Field	Description
31 TFSDIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as output or input.When set as input, the TFCSEL settings are ignored.When set as output, the TFCSEL settings determine the source port of the Frame Sync. 0 TxFS is input pin. 1 TxFS is output.
30 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as output or input.When set as input, the TFCSEL settings are ignored.When set as output, the TFCSEL settings determine the source port of the Clock. 0 TxClk is input pin. 1 TxClk is output.

Table 38-4. Host Port Configuration Register Field Descriptions (continued)

Field	Description
29–26 TFCSEL	Transmit Frame Sync and Clock Select. Selects the source port from which TxFS and TxClk are sourced. 0xxx Selects TxFS and TxClk from port 1xxx Selects RxFS and RxClk from port 000 101 Port 1–Port 6 110 Reserved 111 Reserved
25 RSFDIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as output or input.When set as input, the RFCSEL settings are ignored.When set as output, the RFCSEL settings determine the source port of the Frame Sync. 0 RxFS is input pin. 1 RxFS is output.
24 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as output or input.When set as input, the RFCSEL settings are ignored.When set as output, the RFCSEL settings determine the source port of the Clock. 0 RxClk is input pin. 1 RxClk is output.
23–20 RFCSEL	Receive Frame Sync and Clock Select. Selects the source port from which RxFS and RxClk are sourced. RxFS and RxClk can be sourced from TxFS and TxClk, respectively, from other ports. 0xxx Selects TxFS and TxClk from port 1xxx Selects RxFS and RxClk from port 000–101 Port 1–Port 6 110 Reserved 111 Reserved
19–16	Reserved. These bits are reserved and should read 0.
15–13 RXDSEL	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if INMEN is enabled. xxx Port number for RxD, ignored if equal to self port number 000–101 Port 1–Port 6 110 Reserved 111 Reserved
12 SYN	Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals. That is, the port is a 4-wire interface. When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections. That is, the port is a 6-wire interface. 0 Asynchronous mode 1 Synchronous mode (default)
11	Reserved. This bit is reserved and should read 0.
10 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals from (Da-TxD, Db-RxD) to (Da-RxD, Db-TxD) **Note:** Present only in Port 3 0 No switch 1 Switch
9	Reserved. This bit is reserved and should be read as 0.

Table 38-4. Host Port Configuration Register Field Descriptions (continued)

Field	Description
8 INMEN	Internal Network Mode Enable. RxD from ports in internal network mode are ANDed together. RXDSEL is ignored. INMMASK determines which RxD signals are ANDed together. When internal network mode is enabled at Port 3, then RXDSEL3[3:0] for TxDn_obe selection is ignored and TxD_obe is always driven high. That is, asserted for all timeslots. This places a restriction on slave devices connected in external network mode such that these slave devices have all to be disabled. See Figure 38-16. 0 Disable 1 Enable internal network mode
7–0 INMMASK	Internal Network Mode Mask. Bit mask that selects which of the RxD signals from ports are to be ANDed together for internal network mode. Bit 7 represents RxD from Port 8 and Bit 0 represents RxD from Port 1. **Note:** Bit in self port position should be set as 1. 0 Include RxDn for ANDing. 1 Excludes the RxDn from ANDing.

38.11.4 Peripheral Port Configuration Registers (PPCR1–2)

There is one Peripheral Port Configuration Register (PPCR) for each peripheral port.

0x1001_6010 (PPCR1)
0x1001_6014 (PPCR2)
0x1001_601C (PPCR3)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TFS DIR	TCLK DIR	TFCSEL				RFS DIR	RCLK DIR	RFCSEL				0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDSEL			SYN		TXRXEN	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-11. Peripheral Port Configuration Registers (PPCR1–2)

Table 38-5. Peripheral Port Configuration Register Field Descriptions

Field	Description
31 TFSDIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as output or input.When set as input, the TFCSEL settings are ignored.When set as output, the TFCSEL settings determine the source port of the Frame Sync. 0 TxFS is input pin. 1 TxFS is output.
30 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as output or input.When set as input, the TFCSEL settings are ignored.When set as output, the TFCSEL settings determine the source port of the Clock. 0 TxClk is input pin. 1 TxClk is output.
29–26 TFCSEL	Transmit Frame Sync and Clock Select. Selects the source port from which FS_obe, FS_out, CLK_obe, and CLK_out are sourced. 0xxx Selects TxFS and TxClk from port 1xxx Selects RxFS and RxClk from port xxx Selection ignored if self-port number 110 Reserved 111 Reserved
25 RSFDIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as output or input.When set as input, the RFCSEL settings are ignored.When set as output, the RFCSEL settings determine the source port of the Frame Sync. 0 RxFS is input pin. 1 RxFS is output.
24 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the Waxlike pin of the interface as output or input.When set as input, the RFCSEL settings are ignored.When set as output, the RFCSEL settings determine the source port of the Clock. 0 RxClk is input pin. 1 RxClk is output.
23–20 RFCSEL	Receive Frame Sync and Clock Select. Selects the source port from which RxFS and RxClk are sourced. RxFS and RxClk can be sourced from TxFS and TxClk, respectively, from other ports. 0xxx Selects TxFS and TxClk from port 1xxx Selects RxFS and RxClk from port 000–101 Port 1–Port 6 110 Reserved 111 Reserved
19–16	Reserved
15–13 RXDSEL	Receive Data Select. selects the source port for the RxD data (TxD_in or RxD_in). RXDSEL is ignored if INMEN is enabled. xxx Port number for TxD_in or RxD_in, ignored if equal to self port number 110 Reserved 111 Reserved
12 SYN	Synchronous/Asynchronous Select. SYN controls whether the receive and transmit functions of the port occur synchronously or asynchronously with respect to each other. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals. That is, the port is a 4-wire interface.When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections; that is, the port is a 6-wire interface. 0 Asynchronous mode 1 Synchronous mode (default)

Table 38-5. Peripheral Port Configuration Register Field Descriptions (continued)

Field	Description
11	Reserved. These bits are reserved and should read 0.
10 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals from (Da-TxD, Db-RxD) to (Da-RxD, Db-TxD). 0 No switch 1 Switch
9–0	Reserved. These bits are reserved and should read 0.

38.12 Peripheral Connectivity Through AUDMUX Configuration

This section describes some of the peripheral connectivity scenarios through AUDMUX configuration and some limitations.

38.12.1 Generic Configuration

Figure 38-12 shows a general configuration of the AUDMUX. It does not show what paths are enabled or possible.

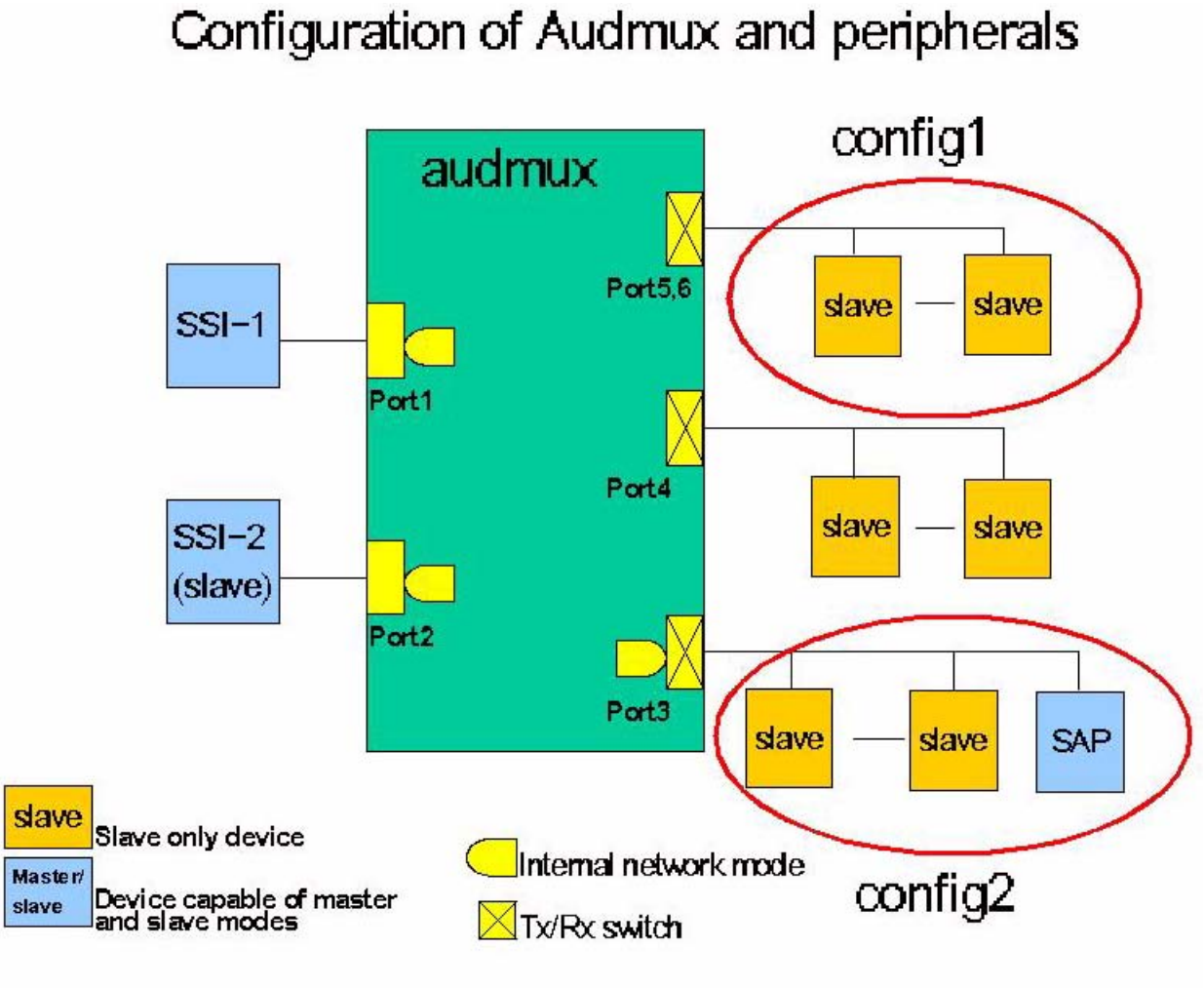


Figure 38-12. SAP as Master to SSI2 as Slave Interconnection

Only Ports 1, 2, 3 are capable of internal network mode. Only Ports 3, 4, 5, 6 are capable of Tx/Rx switch.

Where internal network mode is enabled, for example at Port 1, then Port 1 is referred to as the egress port. Therefore, Ports 1, 2, and 3 become egress ports when internal network mode is enabled at that port.

— *Config1:* only slave devices are connected in network mode.

— *Config2:* one master/slave capable device with slave only devices.

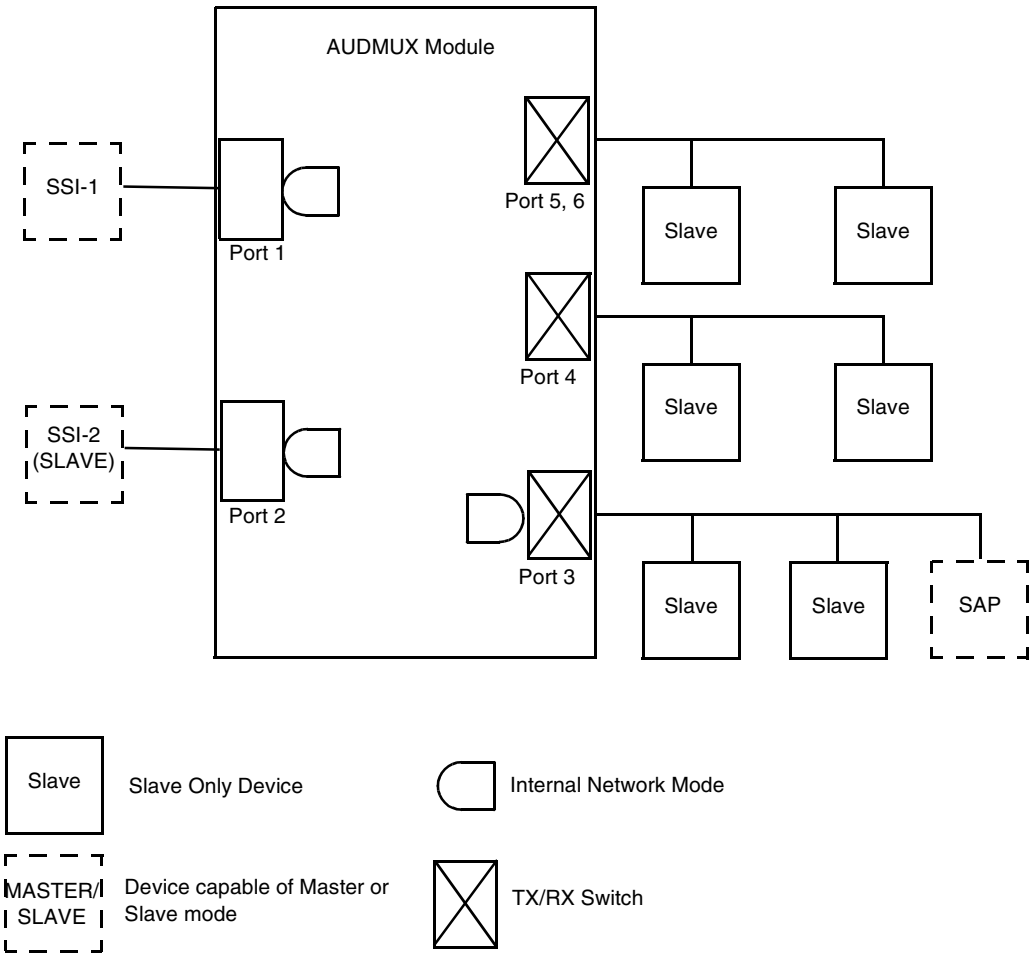


Figure 38-13. Configuration Overview

38.12.2 AUDMUX Configuration with SSI1 and SAP as Master

Figure 38-14 shows two possible audio paths that can be configured simultaneously.

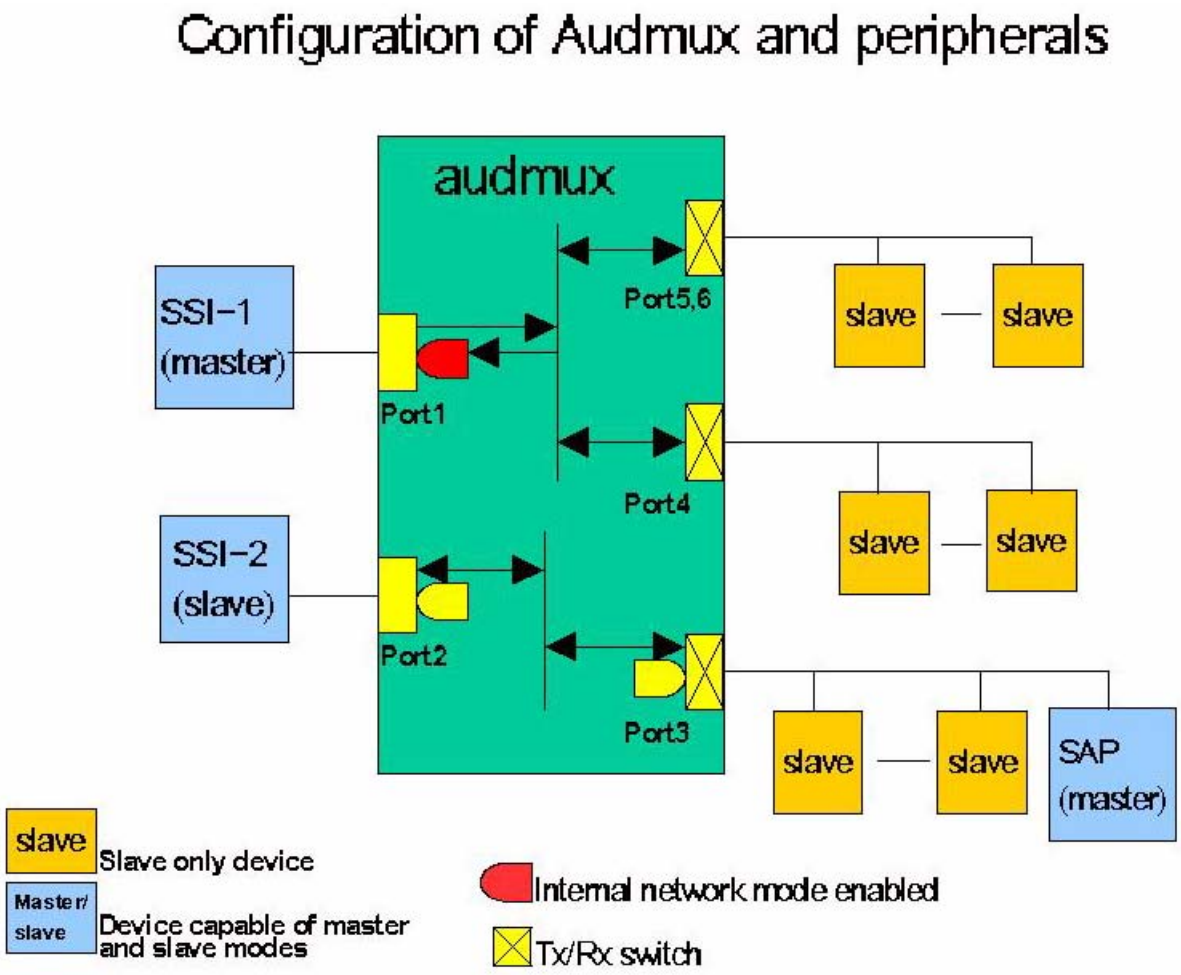


Figure 38-14. SSI1 as Master in Internal Network Mode

Audio Path 1:

- The above configuration shows, SS1-1 configured as master communicating to slaves on Ports 4, 5, and 6.
- Port 1 internal network mode is enabled hence it is the egress port.

Audio Path 2:

- The SAP (Audio port from the Baseband) is connected to slave ports and SSI-2. Tx/Rx switch is not enabled, neither is internal network mode.
- SSI-2 (Internal to the i.MX27 device) is connected as slave.

38.12.3 Tx-Rx Switch Enabled

Figure 38-15 AUDMUX configuration with SSI-1 as the master of the flow.

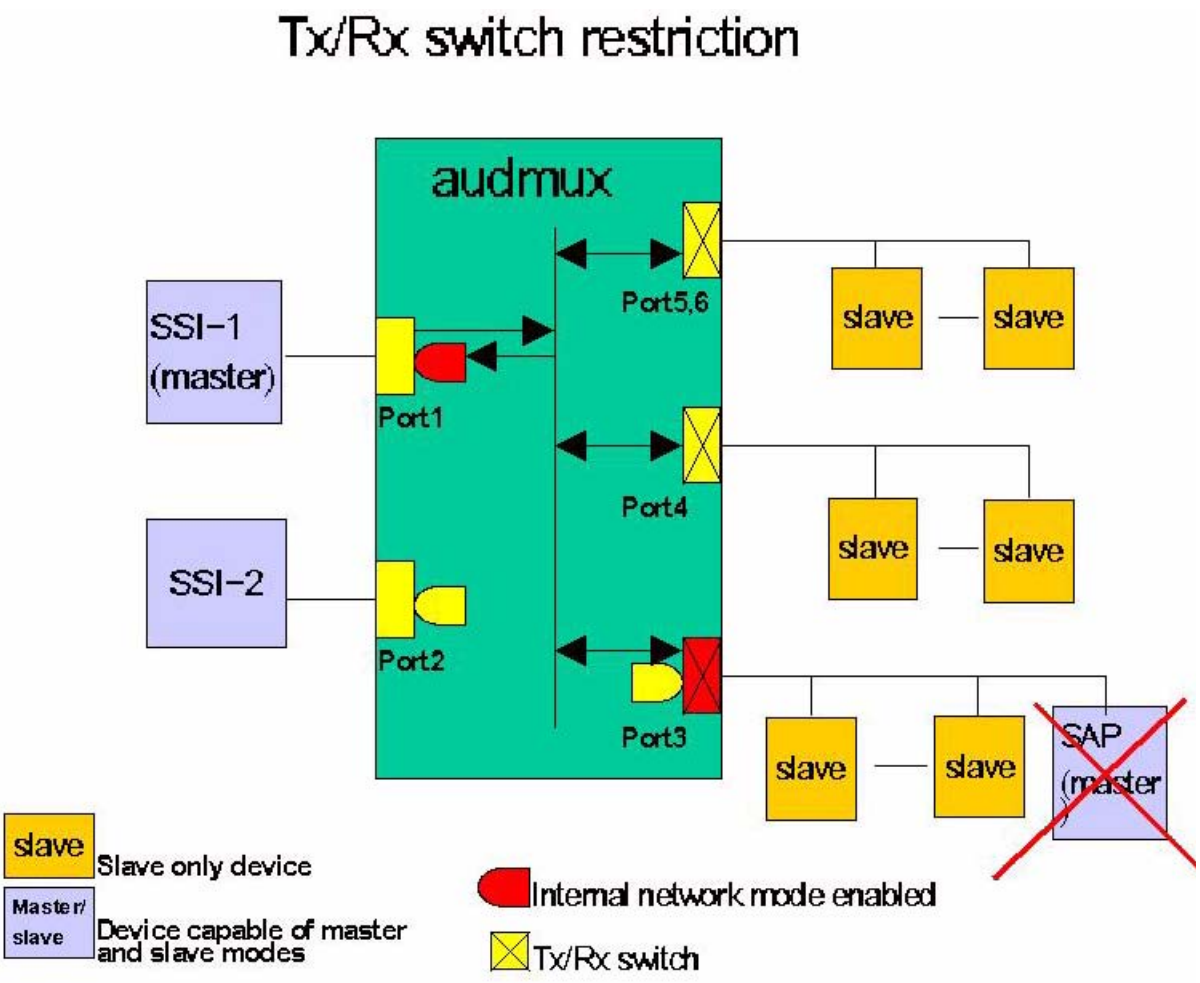


Figure 38-15. Tx-Rx Switch Restriction

Flow:

- SSI-1 is the master that is connected to all peripheral ports.
- Internal network mode is enabled at Port 1 to receive data from Ports 3, 4, 5, 6.
- Tx/Rx Switch is enabled only at Port 3 to maintain signal directions to be consistent for slaves on Port 3.
- The SAP (Audio port from the Baseband) master must be disabled because its Tx and Rx signals will not be consistent.
- Though disabling the SAP master, apparently, makes Config2 into Config1, signal directions are not the same as config1, hence the requirement for the Tx/Rx switch.

38.12.4 Internal/External Network Mode

Figure 38-16 shows the limitation posed by Internal network mode.

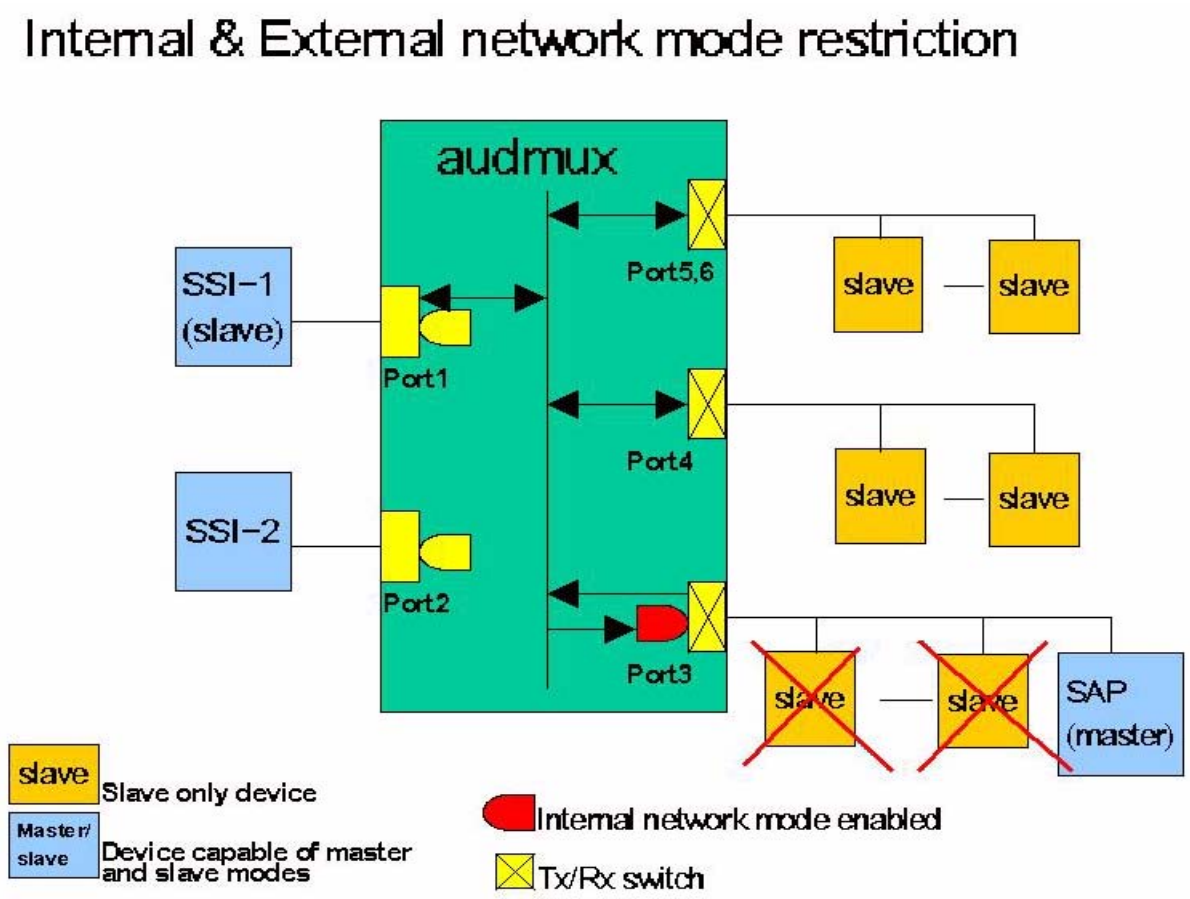


Figure 38-16. Internal and External Mode Restriction

- In this flow, the SAP (Audio port from the baseband) is now the master and internal network mode is enabled at Port 3.
- Port 3 is now an egress port for receive data coming from the other ports.
- The locally attached slave devices have to be disabled because the internal network mode will ALWAYS drive the output at the egress port regardless of timeslots that will cause a multiple driver conflict with slave transmission at Port 3, otherwise.
- Disabling of slave devices in config2 effectively removes external network mode at Port 3. This is what is meant by internal network mode cannot be used with external network mode.

Chapter 39
CMOS Sensor Interface (CSI)

This chapter presents the CMOS Sensor Interface (CSI) architecture, operation principles, and programming model. The CSI enables the i.MX21 to connect directly to external CMOS image sensors. CMOS image sensors are separated into two classes, dumb and smart. Dumb sensors are those that support only traditional sensor timing (Vertical SYNC and Horizontal SYNC) and output only Bayer and statistics data, while smart sensors support CCIR656 video decoder formats and perform additional processing of the image (for example, image compression, image pre-filtering, and various data output formats).

The capabilities of the CSI include:

- Configurable interface logic to support most commonly available CMOS sensors.
- Support for CCIR656 video interface as well as traditional sensor interface.
- 8-bit data port for YCC, YUV, Bayer, or RGB data input.
- Full control of 8-bit and 16-bit data to 32-bit FIFO packing.
- 32 × 32 FIFO to store received image pixel data that can be read through programmed IO or DMA.
- Direct interface to eMMA Preprocessing block (PrP).
- Single interrupt source to interrupt controller from maskable sensor interrupt sources: Start of Frame, End of Frame, Change of Field, FIFO full.
- Configurable master clock frequency output to sensor.
- Statistic data generation for Auto Exposure (AE) and Auto White Balance (AWB) control of the camera (for Bayer data only).

39.1 CSI Architecture

Figure 39-5 shows the block diagram of the CMOS Sensor Interface. It consists of 2 control registers (Control Register 1 and 3) to set up the interface timing and interrupt generation, a control register (Control Register 2) for statistic data generation, a status register, interface logic, data packing logic, CCIR timing decoder, interrupt controller, master clock generator, statistical data generator, 32 × 32 image data receive FIFO (RxFIFO), and a 16 × 32 statistic data FIFO (StatFIFO).

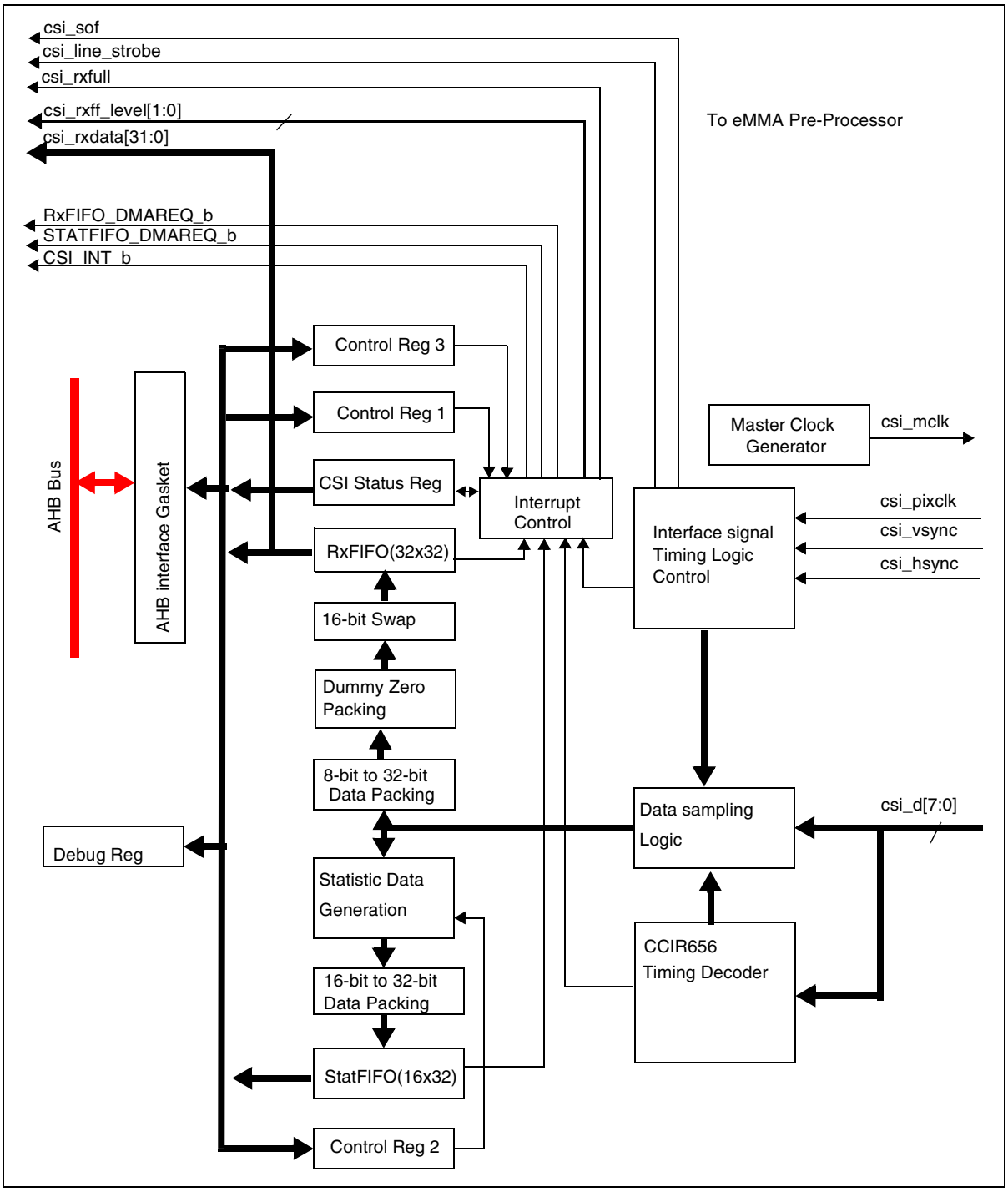


Figure 39-1. CSI Block Diagram

39.2 CSI Interface Signal Description

Table 39-1 provides a listing of the input and output signals between the CSI module of the i.MX27 device and an external CMOS sensor.

Table 39-1. Signals Between CSI and Sensor

CSI Signals	Direction	Description
CSI_VSYNC	Input	Vertical Sync (Start Of Frame)
CSI_HSYNC	Input	Horizontal Sync (Blank Signal)
CSI_D[7:0]	Input	8-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)
CSI_MCLK	Output	Sensor Master Clock
CSI_PIXCLK	Input	Pixel Clock

39.2.1 Signals from CSI to eMMA Pre-Processor Block (PrP)

There is a dedicated bus from CSI RxFIFO to the eMMA Pre-Processor Block (PrP) for fast data transfer.

The bus can be enabled or disabled. When it is enabled, the RxFIFO is detached from the AHB bus and connected to the PrP. Any CPU read or DMA access to the RxFIFO register is ignored. All CSI Interrupts are also masked to prevent software access to the FIFO and status registers.

Users select the RxFIFO full level according to the data format and line width, each of the burst from CSI RxFIFO to PrP must use a size that equal to the RxFIFO full level. To ensure complete transfer of the whole frame, the size of the image (in words) must be integer multiples of the RxFIFO full level. A simple calculation is shown in Table 39-2.

Table 39-2. Integer Multiples of RxFIFO Full Levels

Data Format	Byte Per Pixel	Pixel Per Word	Options for RxFIFO Full Level (Words)	Requirement on Line Width (Pixels)
YUV422	2	2	4 / 8 / 16	Multiple of 8 / 16 / 32
YCC422	2	2		Multiple of 8 / 16 / 32
RGB565	2	2		Multiple of 8 / 16 / 32
RGB888	4	1		Multiples of 4 / 8 / 16
Bayer	1	4		Multiple of 16 / 32 / 64

NOTE

FIFO full level of 24 words is not supported in the CSI-PrP interface. If a 24-word full setting is used the internal logic will regard it as 8 words.

39.3 Principles of Operation

This section describes the modes of operation of the sensor interface.

The CSI is designed to support generic sensor interface timing as well as CCIR656 video interface timing. Traditional CMOS sensors typically use SOF, HSYNC (BLANK), and PIXCLK signals to output Bayer or YUV data. Smart CMOS sensors, that come with on-chip imaging processing, usually support video

mode transfer. They use an embedded timing codec to replace the SOF and BLANK signal. The timing codec is defined by the CCIR656 standard.

39.3.1 Gated Clock Mode

VSYNC, HSYNC, and PIXCLK signals are used in gated clock mode.

A frame starts with a rising edge on VSYNC, then HSYNC goes to HIGH and holds for the entire line. The Pixel clock is valid as long as HSYNC is HIGH. Data is latched at the rising edge of the valid pixel clocks. HSYNC goes to LOW at the end of line. Pixel clocks then become invalid and CSI stops receiving data from the stream. For the next line the HSYNC timing repeats. For the next frame the VSYNC timing repeats.

39.3.2 Non-Gated Clock Mode

In non-gated clock mode, only the VSYNC and PIXCLK signals are used; the HSYNC signal is ignored.

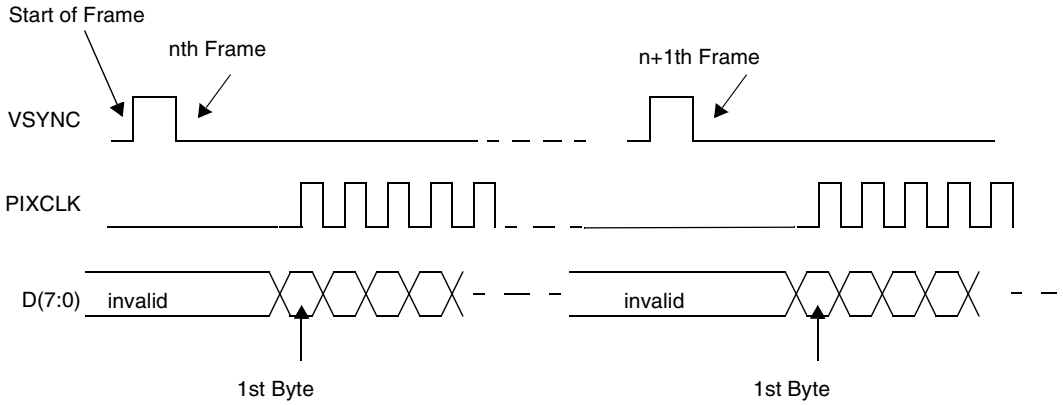


Figure 39-2. Non-Gated Clock Mode Timing Diagram

The overall timing of non-gated mode is the same as the gated-clock mode, except for the HSYNC signal. HSYNC signal is ignored by the CSI. All incoming pixel clocks are valid and cause data to be latched into RxFIFO. The PIXCLK signal is inactive (states low) until valid data is ready to be transmitted over the bus.

Figure 39-2 shows the timing using a typical sensor, other sensors may have the slightly different timing from that shown. The CSI should be programed to support rising/falling-edge triggered VSYNC; active-high/low HSYNC; and rising/falling-edge triggered PIXCLK.

39.3.3 CCIR656 Interlace Mode

In CCIR656 mode, only the PIXCLK and DATA[7:0] signals are used. The start of frame and blank signals are replaced by a timing codec which is embedded in the data stream. Each active line starts with a SAV code and ends with a EAV code. In some cases, digital blanking is inserted in between EAV and SAV code. The CSI decodes and filters out the timing-coding from the data stream, thus recovering VSYNC and HSYNC signals for internal use, such as statistical block control and CSI-to-PrP interconnection. Data is forwarded to the data receive and packing block in a *sequential* manner without re-ordering—that is, field 1 followed by field 2. The fields must be re-ordered in software to get back the original image.

Change Of Field interrupt (COF) is triggered upon every field change. The interrupt service routine reads the status register to check for the current field.

According to the CCIR656 specification, the image must be in 625/50 PAL or 525/60 NTSC format. In addition, the image is interlaced into odd and even fields, with vertical and horizontal blank data being filled into certain lines. Data must be in YCC422 format, each pixel contains 2 bytes, either Y + Cr or Y + Cb. These requirements are set for TV systems. The CSI module supports PAL and NTSC format only. Figure 39-3 shows the frame structure in PAL system, showing vertical blanking and horizontal blanking.

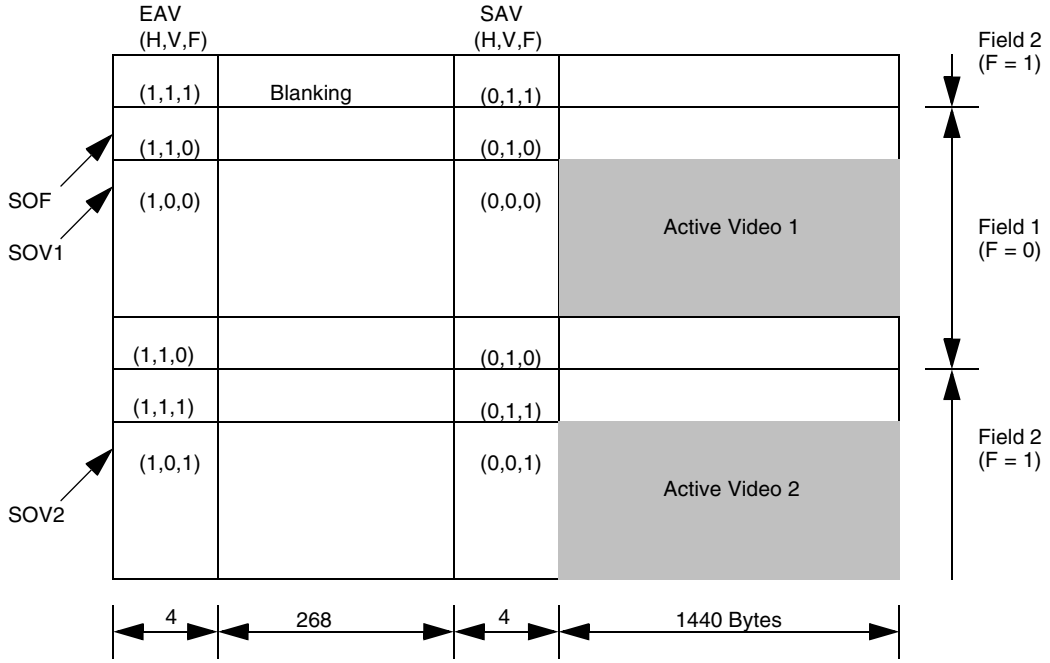


Figure 39-3. CCIR656 Interlace Mode (PAL)

Figure 39-4 shows the general timing for a single line, showing SAV and EAV.

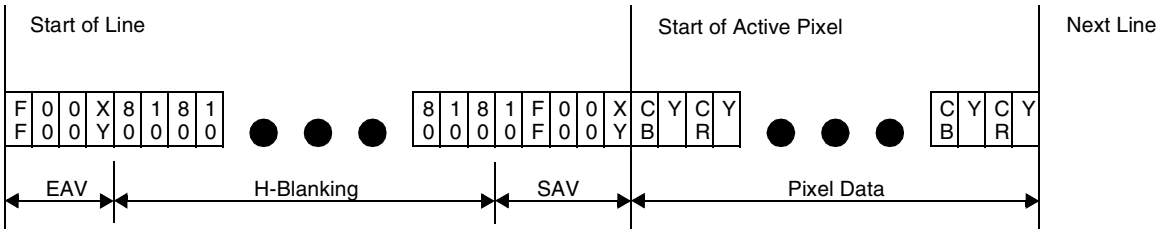


Figure 39-4. CCIR656 General Line Timing

The coding tables recommended by the CCIR656 specification are shown in Table 39-3, Table 39-4 and Table 39-5. It is used in the CCIR656 mode to decode the video stream. An interrupt is generated for SOF, which is decoded from the embedded timing codec.

Table 39-3. Coding for SAV and EAV

Data Bit Number	1st Byte 0xFF	2nd Byte 0x00	3rd Byte 0x00	4th Byte 0xXY
7 (MSB)	1	0	0	1
6	1	0	0	F
5	1	0	0	V
4	1	0	0	H
3	1	0	0	P3
2	1	0	0	P2
1	1	0	0	P1
0	1	0	0	P0

Table 39-4. Coding for Protection Bits

F	V	H	P3	P2	P1	P0
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1

Table 39-5. Representations by F-Bit

F-Bit	Representations
0	ODD FIELD (FIELD 1)
1	EVEN FIELD (FIELD 2)

39.3.4 CCIR656 Progressive Mode

For a CMOS camera system of VGA or CIF resolution, strict adherence to the interlace requirements stated in the CIR standard is not required. The image is considered to have only 1 active field which is scanned in a progressive manner. This active field is regarded as field 1 and the F-bit in the timing codec is ignored by the decoder. Most sensors support CCIR timing in this mode (progressive) by default.

Figure 39-5 shows the typical flow of progressive mode.

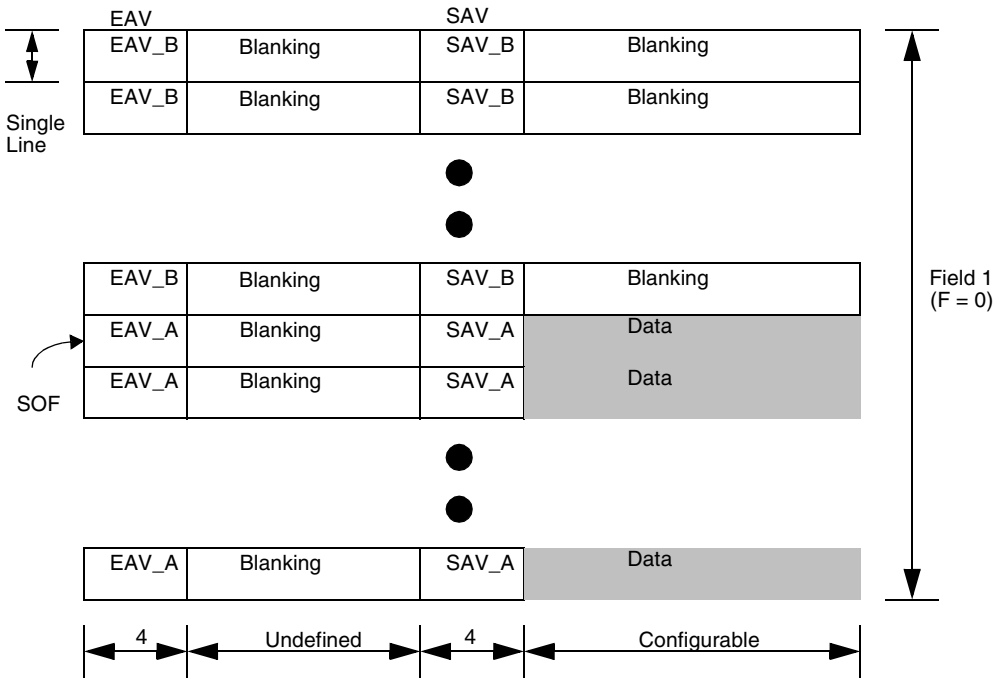


Figure 39-5. CCIR656 Progressive Mode (General Case)

An interrupt is generated for SOF but not for COF. In the general case, when SOF information is retrieved from the embedded coding, it is known as internal VSYNC mode. In other cases, when the VSYNC signal is provided by the sensor, it is known as external VSYNC mode. The CSI can be operated in internal or external VSYNC mode.

39.3.5 Error Correction for CCIR656 Coding

According to the algorithm for CCIR coding, protection bits in the SAV and EAV are encoded in the way that allows a 1-bit error to be corrected, or a 2-bit error to be detected by the decoder. This feature is supported by the CCIR decoder in CSI, for interlace mode only.

For the 1-bit error case, users can select the error to be corrected automatically, or simply shown as a status flag instead. For the 2-bit error case, because the decoder is unable to make a correction, the error would be shown as a status flag only.

An interrupt can be generated upon the detection of an error. This signal can be enabled or disabled without affecting the operation of the status bit.

39.4 Interrupt Generation

This section describes CSI events that generate interrupts.

39.4.1 Start Of Frame Interrupt (SOF_INT)

The source of an SOF interrupt is dependent on the mode of operation.

In traditional mode, VSYNC signal is taken from sensor and SOF_INT is generated at the rising or falling edge (programmable) of VSYNC.

In CCIR interlace mode, the SOF interrupt information is retrieved from the embedded coding and SOF_INT is generated.

In CCIR progressive mode, there are two sources of an SOF interrupt:

- In *internal* VSYNC mode, SOF is retrieved from the embedded coding.
- In *external* VSYNC mode, VSYNC is taken from the sensor and SOF is generated at the rising edge of VSYNC.

39.4.2 End Of Frame Interrupt (EOF_INT)

An EOF interrupt is generated when the frame ends and the complete frame data in RXFIFO is read. The EOF interrupt does not work in CSI PreProcessing mode.

The EOF event triggering works with the RX count register (CSIRXCNT). Software sets the RX count register to the frame size (in bytes). The CSI RX logic then counts the number of pixel data being received and compares it with the RX count. If the preset value is reached, then an EOF interrupt is generated and the data in the RXFIFO are read. If an SOF event is detected before this happens, then the EOF interrupt is not generated.

39.4.3 Change Of Field Interrupt (COF_INT)

The Change of Field interrupt is only valid in CCIR Interlace mode. The COF interrupt is generated when the field toggles, either from field 1 to field 2, or field 2 to field 1.

Software should first check on COF_INT bit in the CSI Status Register (CSISTAT), before checking that F1_INT or F2_INT is turned on.

In PAL systems, the field changes at the beginning of the frame and coincides with SOF. For the first field, a COF interrupt is not generated, only an SOF is generated. The COF interrupt is generated for the second field.

39.4.4 CCIR Error Interrupt (ECC_INT)

The CCIR Error Interrupt is only valid for CCIR Interlace mode. An ECC interrupt is generated when an error is found on the SAV or EAV codes in the incoming stream. When this happens, the ECC_INT status bit is set.

39.4.5 Data Packing Style

Owing to different port sizes at different stages of the image capture path, the endianess of data is important. To enable flexible packing of image data, the CSI module provides data swapping through the PACK_DIR and the SWAP16_EN bits in CSI Control Register 1 (CSICR1) which enables data swapping before it is presented to the FIFOs.

Data is packed from 8-bit to 32-bit according to the setting of PACK_DIR bit, and then put into the RX FIFO according to the setting of the SWAP16_EN bit.

39.4.6 RX FIFO Path

Bayer data is a type of raw data from the image sensor. This byte-wide data must be converted to the RGB space or YUV space by software. The data path for Bayer data is from the CSI to memory. If the system is in little endian, then the PACK_DIR bit should be set to 0. Doing so results in the data being packed to 32-bit as P3P2P1P0, where P0 is the pixel coming in time slot 0 (first data), while P3 is the pixel coming in time slot 3 (last data). When the data is addressed as bytes by software, P0 goes out first, and ends up with P3.

39.4.6.1 RGB565 Data

RGB565 data is processed data from the image sensor, which can be put directly into the display buffer. The data is 16 bits wide. The data path is from CSI to memory, memory to LCDC. On the sensor side, data must be output as P0 first, followed by P1, and so on. Within each pixel, either MSB or LSB will come out first. This is controlled by the endian style of the sensor. Data is 16 bits wide with the MSB labeled RG, and the LSB labeled GB. So for P0, it is represented as RG0, GB0, and so on for P1.

CSI receives data in one of the following sequence:

- RG0, GB0, RG1, GB1, while RG0 comes out at time slot 0 (first data), and GB1 comes out at time slot 3 (last data), or
- GB0, RG0, GB1, RG1.

Using the first sequence as an example, and assuming the system is running in little endian the data is presented as:

- 8-bit data from sensor: RG0, GB0, RG1, GB1, …
- 32-bit data before CSI RX FIFO (PACK_DIR bit = 1): RG0GB0RG1GB1
- 32-bit data in CSI RX FIFO (SWAP16_EN bit enabled): RG1GB1RG0GB0
- 32-bit transfer to system memory: RG1GB1RG0GB0
- 16-bit read by LCDC: RG0GB0, RG1GB1

39.4.6.2 RGB888 Data

This is another kind of processed data from image sensor, which can be used for further image processing directly. Each of the data consist of 8-bit Red, 8-bit Green, and 8-bit Blue data. An example of a possible timing scheme is shown in Figure 39-6.

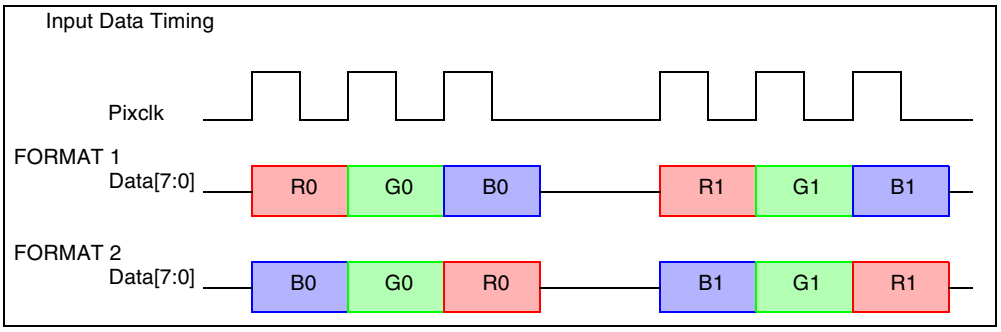


Figure 39-6. Sample Timing Diagram for RGB888 Data

The 3-byte per pixel structure is not an optimum choice for a CSI to PRP path. To improve the data transfer from CSI to PRP, an optional dummy byte packing scheme is provided. For every group of 3 bytes data, a dummy zero is packed to form a 32-bit word as shown in Figure 39-7. The dummy zero is always packed at the LSB position. This byte will be ignored by the PRP

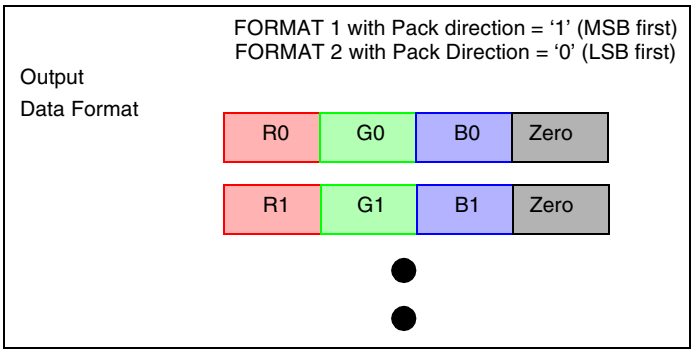


Figure 39-7. Optional Dummy Byte Packing Scheme

39.4.7 STAT FIFO Path

Statistics only works for Bayer data. It generates 16-bit statistical output from the 8-bit Bayer input. The outputs are Sum of Green (G), Sum of Red (R), Sum of Blue (B), and Auto Focus (F). Each output is 16-bits wide.

The settings of PACK_DIR and SWAP16_EN bits in the CSICR1 register have no effect on the input path. The PACK_DIR only controls how the 16-bit stat output is packed into the 32-bit STAT FIFO.

When the PACK_DIR bit = 1, the stat data is packed as:

First 32-bit: RG

Second 32-bit: BF

…

When the PACK_DIR bit = 0, the stat data is packed as:

First 32-bit GR

Second 32-bit: FB

…

39.5 Memory Map and Register Definition

The CSI module contains six 32-bit registers which are summarized in Table 39-6. The base address of the CSI module for i.MX27 is 0x1000 8000. Table 39-8 summarizes the registers and offset addresses. Section 39.5, "Memory Map and Register Definition" provides detailed descriptions of the DMAC register.

39.5.1 CSI Memory Map

Table 39-6 shows the CSI memory map.

Table 39-6. CSI Memory Map

Address	Use	Access	Reset Value	Section/Page
0x8000_00000 (CSICR1)	CSI Control Register 1	R/W	0x4000_0800	39.5.3/39-14
0x8000_00004 (CSICR2)	CSI Control Register 2	R/W	0x0000_0000	39.5.4/39-17
0x8000_0001C (CSICR3)	CSI Control Register 3	R/W	0x0000_0000	39.5.5/39-19
0x8000_00008 (CSISR)	CSI Status Register	R/W	0x0000_4000	39.5.6/39-20
0x8000_0000C (CSISTATFIFO)	CSI Statistic FIFO Register	R	0x0000_0000	39.5.7/39-22
0x8000_00010 (CSIRFIFO)	CSI RX FIFO Register	R	0x0000_0000	39.5.8/39-22
0x8000_00014 (CSIRXCNT)	CSI RX Count Register	R/W	0x0000_0000	39.5.9/39-23

39.5.2 Register Summary

Figure 39-8 shows the key to the register fields, and Table 39-7 shows the register figure conventions.

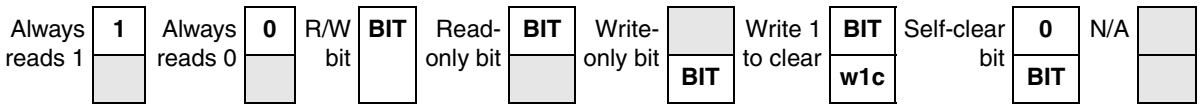


Figure 39-8. Key to Register Fields

Table 39-7. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.

Table 39-7. Register Figure Conventions (continued)

Convention	Description
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 39-8 shows the CSI register summary.

Table 39-8. CSI Register Summary

Name		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
0x8000_00000 (CSICR1)	R	SWAP16_EN	EXT_VSYNC	EOF_INT_EN	PrP_IF_EN	CCIR_MODE	COF_INT_EN	SF_OR_INTEN	RF_OR_INTEN	STATFF_LEVEL		STATFF_INTEN	RxFF_LEVEL		RxFF_INTEN	SOF_POL	SOF_INTEN
	W																
	R	MCLKDIV				HSYNC_POL	CCIR_EN	MCLKEN	FCC	PACK_DIR	CLR_STATFIFO	CLR_RxFIFO	GCLK_MODE	INV_DATA	INV_PCLK	REDGE	0
	W																
0x8000_00004 (CSICR2)	R	0	0	0	0	0	DRM	AFS		SCE	0	0	BTS		LVRM		
	W																
	R	VSC								HSC							
	W																
0x8000_0001C (CSICR3)	R	FRMCNT															
	W																
	R	FRMCNT_RST	0	0	0	0	0	0	0	0	0	0	0	CSI_SUP	ZERO_PACK_EN	ECC_INT_EN	ECC_AUTO_EN
	W																

Table 39-8. CSI Register Summary (continued)

Name		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
0x8000_00008 (CSISR)	R	0	0	0	0	0	0	SFF_OR_INT	RFF_OR_INT	0	0	STATFF_INT	0	0	RxFF_INT	EOF_INT	SOF_INT
	W																
	R	F2_INT	F1_INT	COF_INT	0	0	0	0	0	0	0	0	0	0	0	ECC_INT	DRDY
	W																
0x8000_0000C (CSISTATFIFO)	R	STAT															
	W																
	R	STAT															
	W																
0x8000_00010 (CSIRFIFO)	R	IMAGE															
	W																
	R	IMAGE															
	W																
0x8000_00014 (CSIRXCNT)	R	0	0	0	0	0	0	0	0	0	0	RXCNT					
	W																
	R	RXCNT															
	W																

39.5.3 CSI Control Register 1 (CSICR1)

This register controls the sensor interface timing, CSI-to-PrP bus interface and interrupt generation. The CSI module is enabled through the Peripheral Clock Control Register 0 (PCCR0). The interrupt enable bits in this register control the interrupt signals and the status bits. That means status bits will only function when the corresponding interrupt bits are enabled.

0x8000_00000 (CSICR1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SWAP16_EN	EXT_VSYNC	EOF_INT_EN	PrP_IF_EN	CCIR_MODE	COF_INT_EN	SF_OR_INTEN	RF_OR_INTEN	STATFF_LEVEL		STATFF_INTEN	RxFF_LEVEL		RxFF_INTEN	SOF_POL	SOF_INTEN
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MCLKDIV				HSYNC_POL	CCIR_EN	MCLKEN	FCC	PACK_DIR	CLR_STATFIFO	CLR_RxFIFO	GCLK_MODE	INV_DATA	INV_PCLK	REDGE	0
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 39-9. CSPI Control Register 1 (CSICR1)

Table 39-9. CSI Control Register 1 Field Descriptions

Field	Description
31 SWAP16_EN	SWAP 16-Bit Enable. This bit enables the swapping of 16-bit data. Data is packed from 8-bit to 32-bit first (according to the setting of PACK_DIR and then swapped as 16-bit words before putting into the RX FIFO. The action of the bit only affects the RX FIFO and has no affect on the STAT FIFO. **Note:** Example of swapping enabled: Data input to FIFO = 0x11223344 Data in RX FIFO = 0x 33441122 **Note:** Example of swapping disabled: Data input to FIFO = 0x11223344 Data in RX FIFO = 0x11223344 0 Disable swapping 1 Enable swapping
30 EXT_VSYNC	External VSYNC Enable. This bit controls the operational VSYNC mode. **Note:** This only works when the CIS is in CCIR progressive mode. 0 Internal VSYNC mode 1 External VSYNC mode
29 EOF_INT_EN	End-of-Frame Interrupt Enable. This bit enables and disables the EOF interrupt. 0 EOF interrupt is disabled. 1 EOF interrupt is generated when RX count value is reached.

Table 39-9. CSI Control Register 1 Field Descriptions (continued)

Field	Description
28 PrP_IF_EN	CSI—PrP Interface Enable. This bit controls the CSI to Prp bus. When enabled the RxFIFO is detached from the AHB bus and connected to PrP. All CPU reads or DMA accesses to the RxFIFO register are ignored. All CSI interrupts are also masked. 0 CSI to PrP bus is disabled 1 CSI to PrP bus is enabled
27 CCIR_MODE	CCIR Mode Select. This bit controls the CCIR mode of operation. This bit only works in CCIR interface mode. 0 Progressive mode is selected 1 Interlace mode is selected
26 COF_INT_E	Change Of Image Field (COF) Interrupt Enable. This bit enables the COF interrupt. This bit works only in CCIR interlace mode which is when CCIR_EN = 1 and CCIR_MODE = 1. 0 COF interrupt is disabled 1 COF interrupt is enabled
25 SF_OR_INTEN	STAT FIFO Overrun Interrupt Enable. This bit enables the STATFIFO overrun interrupt. 0 STATFIFO overrun interrupt is disabled 1 STATFIFO overrun interrupt is enabled
24 RF_OR_INTEN	RxFIFO Overrun Interrupt Enable. This bit enables the RX FIFO overrun interrupt. 0 RxFIFO overrun interrupt is disabled 1 RxFIFO overrun interrupt is enabled
23–22 STATFF_LEVEL	STATFIFO Full Level. When the number of data in STATFIFO reach this level, STATFIFO full interrupt is generated, or STATFIFO DMA request is sent. 00 4 Words 01 8 Words 10 12 Words 11 16 Words
21 STATFF_INTEN	STATFIFO Full Interrupt Enable. This bit enables the STAT FIFO interrupt. 0 STATFIFO full interrupt disable 1 STATFIFO full interrupt enable
20–19 RXFF_LEVEL	RxFIFO Full Level. When the number of data in RxFIFO reach this level, a RxFIFO full interrupt is generated, or an RXFIFO DMA request is sent, or CSI-PrP burst cycle is issued. 00 4 Words 01 8 Words 10 16 Words 11 24 Words **Note:** In the case when PrP I/F is enabled, 24-words option is not supported, internal logic will regard it as 8-word. This is not reflected in the register value.
18 RXFF_INTEN	RxFIFO Full Interrupt Enable. This bit enables the RxFIFO full interrupt. 0 RxFIFO full interrupt disable 1 RxFIFO full interrupt enable
17 SOF_POL	SOF Interrupt Polarity. This bit controls the condition that generates an SOF interrupt. 0 SOF interrupt is generated on SOF falling edge 1 SOF interrupt is generated on SOF rising edge
16 SOF_INTEN	Start Of Frame (SOF) Interrupt Enable. This bit enables the SOF interrupt. 0 SOF interrupt disable 1 SOF interrupt enable

Table 39-9. CSI Control Register 1 Field Descriptions (continued)

Field	Description
15–12 MCLKDIV	Sensor Master Clock (MCLK) Divider. This field contains the divisor MCLK. The MCLK is derived from the PERCLK4. 0000 Divided by 2 0001 Divided by 4 0010 Divided by 6 ... 1111 Divided by 32
11 HSYNC_POL	HSYNC Polarity Select. This bit controls the polarity of HSYNC. **Note:** This bit only works in gated-clock—that is, GCLK_MODE = 1 and CCIR_EN = 0. 0 HSYNC is active low 1 HSYNC is active high
10 CCIR_EN	CCIR656 Interface Enable. This bit selects the type of interface used. When the CCIR656 timing decoder is enabled, it replaces the function of timing interface logic. 0 Traditional interface is selected. Timing interface logic is used to latch data. 1 CCIR656 interface is selected.
9 MCLKEN	Sensor Master Clock (MCLK) Enable. This bit enables or disables the MCLK input to the sensor. 0 MCLK disable 1 MCLK enable
8 FCC	FIFO Clear Control. This bit determines how the RXFIFO and STATFIFO are cleared. When Synchronous FIFO clear is selected the RXFIFO and STATFIFO are cleared, and STAT block is reset, on every SOF. FIFOs and STAT block restarts immediately after reset. For information on the operation when Asynchronous FIFO clear is selected, refer to the descriptions for the CLR_RXFIFO and CLR_STATFIFO bits. 0 Asynchronous FIFO clear is selected. 1 Synchronous FIFO clear is selected.
7 PACK_DIR	Data Packing Direction. This bit Controls how 8-bit image data is packed into 32-bit RX FIFO, and how 16-bit statistical data is packed into 32-bit STAT FIFO. 0 Pack from LSB first. For image data, 0x11, 0x22, 0x33, 0x44, it will appear as 0x44332211 in RX FIFO. For stat data, 0xAAAA, 0xBBBB, it will appear as 0xBBBBAAAA in STAT FIFO. 1 Pack from MSB first. For image data, 0x11, 0x22, 0x33, 0x44, it will appear as 0x11223344 in RX FIFO. For stat data, 0xAAAA, 0xBBBB, it will appear as 0xAAAABBBB in STAT FIFO.
6 CLR_STATFIFO	Asynchronous STATFIFO Clear. This bit clears the STATFIFO and Reset STAT block. **Note:** This bit works only in async FIFO clear mode—that is, FCC = 0. Otherwise this bit is ignored. Writing 1 will clear STATFIFO and reset STAT block immediately, STATFIFO and STAT block then wait and restart after the arrival of next SOF. The bit is restored to 0 automatically after finish. Normally reads 0.fffffffff
5 CLR_RXFIFO	Asynchronous RXFIFO Clear. This bit clears the RXFIFO. This bit works only in async FIFO clear mode—that is, FCC = 0. Otherwise this bit is ignored. Writing 1 clears the RXFIFO immediately, RXFIFO restarts immediately after that. The bit is restore to 0 automatically after finish. Normally reads 0.
4 GCLK_MODE	Gated Clock Mode Enable. Controls if CSI is working in gated or non-gated mode. **Note:** This bit works only in traditional mode—that is, CCIR_MODE = 0. Otherwise this bit is ignored. 0 Non-gated clock mode. All incoming pixel clocks are valid. HSYNC is ignored. 1 Gated clock mode. Pixel clock signal is valid only when HSYNC is high.

Table 39-9. CSI Control Register 1 Field Descriptions (continued)

Field	Description
3 INV_DATA	Invert Data Input. This bit enables or disables internal inverters on the data lines. 0 CSI_D[7:0] data lines are directly applied to internal circuitry 1 CSI_D[7:0] data lines are inverted before applied to internal circuitry
2 INV_PCLK	Invert Pixel Clock Input. This bit determines if the Pixel Clock (CSI_PIXCLK) is inverted before it is applied to the CSI module. 0 CSI_PIXCLK is directly applied to internal circuitry 1 CSI_PIXCLK is inverted before applied to internal circuitry
1 REDGE	Valid Pixel Clock Edge Select. Selects which edge of the CSI_PIXCLK is used to latch the pixel data. 0 Pixel data is latched at the falling edge of CSI_PIXCLK 1 Pixel data is latched at the rising edge of CSI_PIXCLK
0	Reserved. This bit is reserved and should read 0.

39.5.4 CSI Control Register 2 (CSICR2)

This register provides the statistic block with data about which live view resolution is being used, and the starting sensor pixel of the Bayer pattern. It also contains the horizontal and vertical count used to determine the number of pixels to skip between the 64×64 blocks of statistics when generating statistics on live view image that are greater than 512×384.

0x8000_00004 (CSICR2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	DRM	AFS		SCE	0	0	BTS		LVRM		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 39-10. CSI Control Register 2 (CSICR2)

Table 39-10. CSI Control Register 2 Description

Field	Description
31–27	Reserved. These bits are reserved and should read 0.
26 DRM	Double Resolution Mode. Controls size of statistics grid. 0 Stats grid of 8×6 1 Stats grid of 8×12

Table 39-10. CSI Control Register 2 Description (continued)

Field	Description
25–24 AFS	Auto Focus Spread. Selects which green pixels are used for auto-focus. 00 Abs Diff on consecutive green pixels 01 Abs Diff on every third green pixels 1x Abs Diff on every four green pixels
23 SCE	Skip Count Enable. Enables or disables the skip count feature. 0 Skip count disable 1 Skip count enable
22–21	Reserved. These bits are reserved and should read 0.
20–19 BTS	Bayer Tile Start. Controls the Bayer pattern starting point. 00 GR 01 RG 10 BG 11 GB
18–16 LVRM	Live View Resolution Mode. Selects the grid size used for live view resolution. 0 512 × 384 1 448 × 336 2 384 × 288 3 384 × 256 4 320 × 240 5 288 × 216 6 400 × 300
15–8 VSC	Vertical Skip Count. Contains the number of rows to skip. SCE must be 1, otherwise VSC is ignored. 0–255 Number of rows to skip minus 1
7–0 HSC	Horizontal Skip Count. Contains the number of pixels to skip. SCE must be 1, otherwise HSC is ignored. 0–255 Number of pixels to skip minus 1

39.5.5 CSI Control Register 3 (CSICR3)

This read/write register acts as an extension of the functionality of the CSI Control register 1 adding additional control and features.

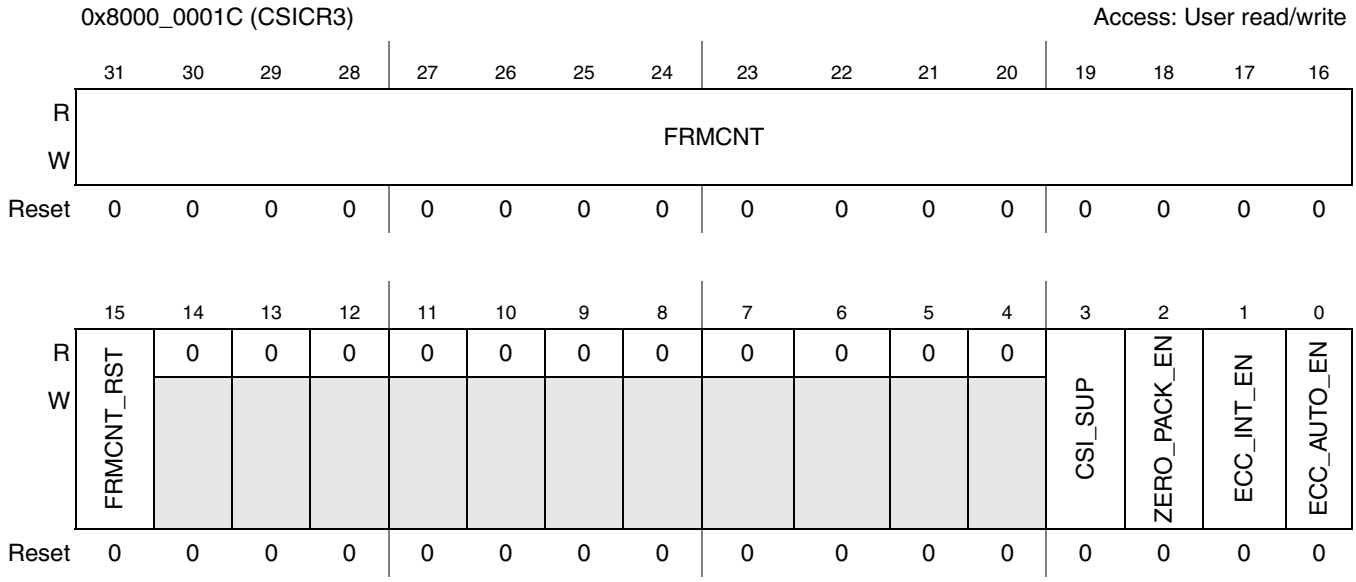


Figure 39-11. CSI Control Register 3 (CSICR3)

Table 39-11. CSI Control Register 3 Field Descriptions

Field	Description
31–16 FRMCNT	Frame Counter. This is a 16-bit Frame Counter (Wrap around automatically after reaching the maximum)
15 FRMCNT_RST	Frame Count Reset. Resets the Frame Counter. 0 Do not reset 1 Reset frame counter immediately
14–4	Reserved. These bits are reserved and should read 0.
3 CSI_SVR	Supervisor Mode Access Control. This bit enables and disables ARM9 supervisor mode access. 0 Module can be accessed in any ARM9 mode 1 Module can only be accessed in ARM9 supervisor mode. Accessing the module in non-supervisor mode will cause a Data Abort exception.
2 ZERO_PACK_EN	Dummy Zero Packing Enable. This bit causes a dummy zero to be packed with every 3 incoming bytes, forming a 32-bit word. The dummy zero is always packed to the LSB position. 0 Zero packing disabled 1 Zero packing enabled

Table 39-11. CSI Control Register 3 Field Descriptions

Field	Description
1 ECC_INT_EN	Error Detection Interrupt Enable. This bit enables and disables the error detection interrupt. This feature only works in CCIR interlace mode. 0 No interrupt is generated when error is detected. Only the status bit ECC_INT is set. 1 Interrupt is generated when error is detected.
0 ECC_AUTO_EN	Automatic Error Correction Enable. This bit enables and disables the automatic error correction. If an error occurs and error correction is disabled only the ECC_INT status bit is set. This feature only works in CCIR interlace mode. 0 Auto Error correction is disabled. 1 Auto Error correction is enabled.

39.5.6 CSI Status Register (CSISR)

This read/write register shows sensor interface status, and which kind of interrupt is being generated. The corresponding interrupt bits must be set for the status bit to function. Status bits should function normally even if the corresponding interrupt enable bits are not enabled.

0x8000_00008 (CSISR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	SFF_OR_INT	RFF_OR_INT	0	0	STATFF_INT	0	0	RxFF_INT	EOF_INT	SOF_INT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	F2_INT	F1_INT	COF_INT	0	0	0	0	0	0	0	0	0	0	0	ECC_INT	DRDY
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-12. CSI Status Register (CSISR)

Table 39-12. CSI Status Register Field Descriptions

Field	Description
31–26	Reserved. These bits are reserved and should read 0.
25 SF_OR_INT	STATFIFO Overrun Interrupt Status. Indicates the overflow status of the STATFIFO register. 0 STATFIFO has not overflowed. 1 STATFIFO has overflowed. (Cleared by writing 1)

Table 39-12. CSI Status Register Field Descriptions (continued)

Field	Description
24 RF_OR_INT	RxFIFO Overrun Interrupt Status. Indicates the overflow status of the RxFIFO register. 0 RXFIFO has not overflowed. 1 RXFIFO has overflowed. (Cleared by writing 1)
23–22	Reserved. These bits are reserved and should read 0.
21 STATFF_INT	STATFIFO Full Interrupt Status. Indicates whether the STATFIFO condition is either full or not full /overflowed. 0 STATFIFO is not full, or has overflowed. 1 STATFIFO is full. (this bit is cleared automatically by reading the STATFIFO)
20–19	Reserved. These bits are reserved and should read 0.
18 RxFF_INT	RXFIFO Full Interrupt Status. Indicates whether the RxFIFO condition is either full or not full /overflowed. 0 RxFIFO is not full, or has overflowed. 1 RxFIFO is full. (this bit is cleared automatically by reading the RxFIFO)
17 EOF_INT	End of Frame (EOF) Interrupt Status. Indicates when EOF is detected. 0 EOF is not detected. 1 EOF is detected. (Cleared by writing 1)
16 SOF_INT	Start of Frame Interrupt Status. Indicates when SOF is detected. 0 SOF is not detected. 1 SOF is detected. (Cleared by writing 1)
15 F2_INT	CCIR Field 2 Interrupt Status. Indicates the presence of field 2 of video in CCIR mode. **Note:** Only works in CCIR Interlace mode. 0 Field 2 of video is not detected 1 Field 2 of video is about to start (Cleared automatically when current field does not match)
14 F1_INT	CCIR Field 1 Interrupt Status. Indicates the presence of field 1 of video in CCIR mode. **Note:** Only works in CCIR Interlace mode. 0 Field 1 of video is not detected. 1 Field 1 of video is about to start. (Cleared automatically when current field does not match)
13 COF_INT	Change Of Field Interrupt Status. Indicates that a change of the video field has been detected. Only works in CCIR Interlace mode. Software should read this bit first and then dispatch the new field from F1_INT and F2_INT. 1 Change of video field is detected. 0 Video field has no change. (Cleared by writing 1)
12– 2	Reserved. These bits are reserved and should read 0.

Table 39-12. CSI Status Register Field Descriptions (continued)

Field	Description
1 ECC_INT	CCIR Error Interrupt. This bit indicates an error has occurred. This only works in CCIR Interlace mode. 0 No error detected (Cleared by writing 1) 1 Error is detected in CCIR coding
0 DRDY	RXFIFO Data Ready. Indicates the presence of data that is ready for transfer in the RxFIFO. 0 No data (word) is ready 1 At least 1 data (word) is ready in RXFIFO. (Cleared automatically by reading FIFO)

39.5.7 CSI STATFIFO Register (CSISTATFIFO)

The StatFIFO is a read-only register containing statistic data from the sensor. Writing to this register has no effect.

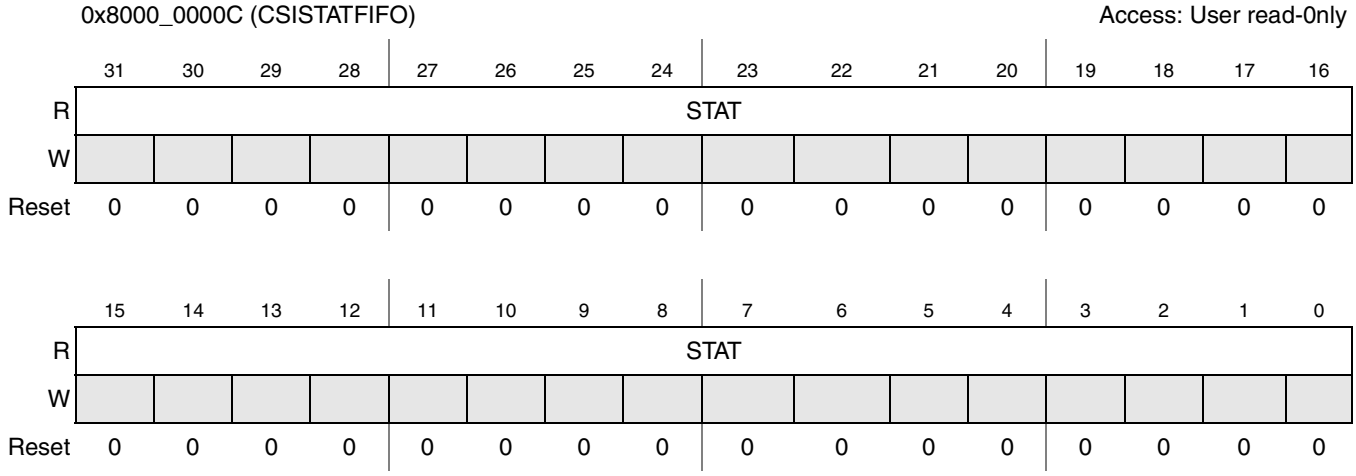


Figure 39-13. CSI STATFIFO Register (CSISTATFIFO)

39.5.8 CSI RxFIFO Register (CSIRFIFO)

This read-only register contains received image data. Writing to this register has no effect.

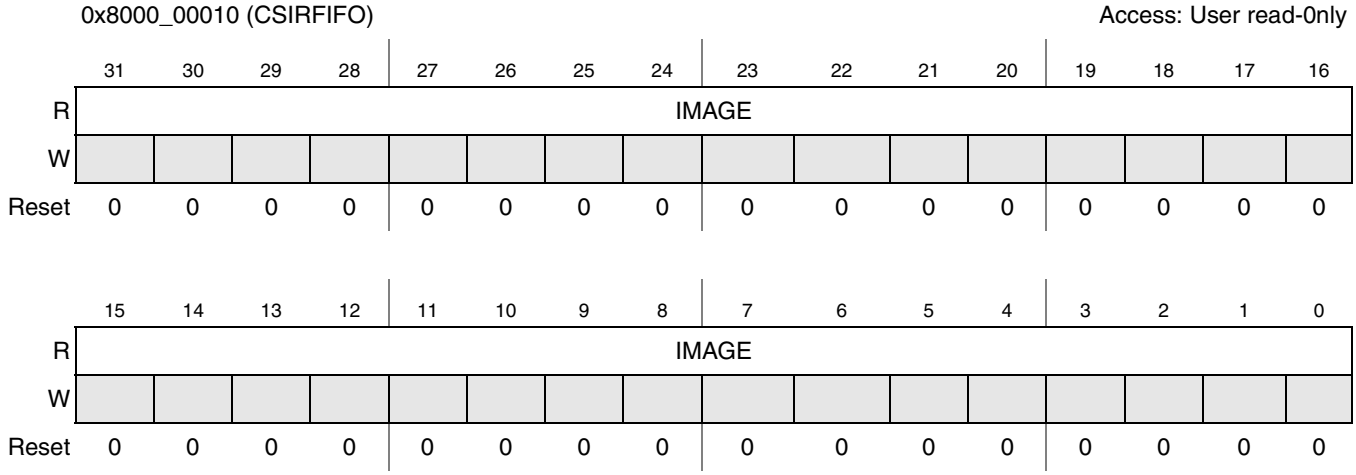


Figure 39-14. CSI RxFIFO Register (CSIRFIFO)

Because the incoming image data is 8-bit while the FIFO size is 32-bit, the incoming byte size data will be packed into 32-bit in the following sequence.

39.5.9 CSI RX Count Register (CSIRXCNT)

This register works for EOF interrupt generation. It should be set to the number of words to receive that would generate an EOF interrupt.

There is an internal counter that counts the number of words read from the RX FIFO. Whenever the RX FIFO is being read, by either the CPU or DMA, the counter value is updated and compared with this register. If the values match, then an EOF interrupt is triggered.

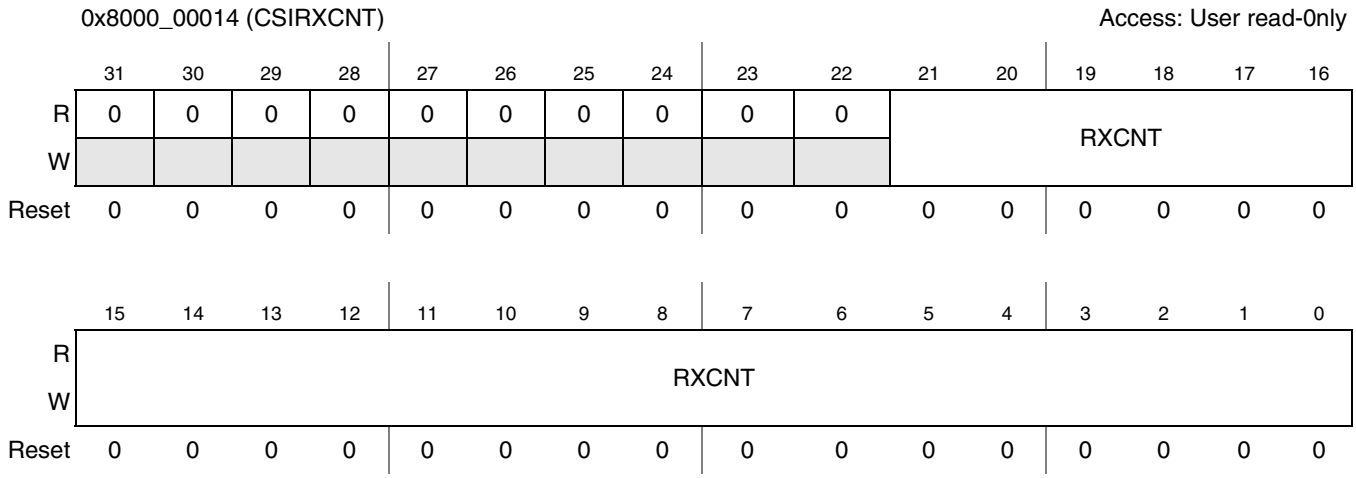


Figure 39-15. CSI RX Count Register (CSIRXCNT)

Table 39-13. CSI RX Count Register Field Descriptions

Field	Description
31–22	Reserved. These bits are reserved and should read 0.
21–0 RXCNT	RxFIFO Count. This 22-bit counter for RXFIFO is updated each time the RXFIFO is read by CPU or DMA.This counter should be set to the expected number of words to receive that would generate an EOF interrupt.

Chapter 40
Video Codec (Video_Codec)

The Video Codec module is the multimedia video processing module in the i.MX27 device. Figure 40-1 shows the top-level diagram for Video Codec.

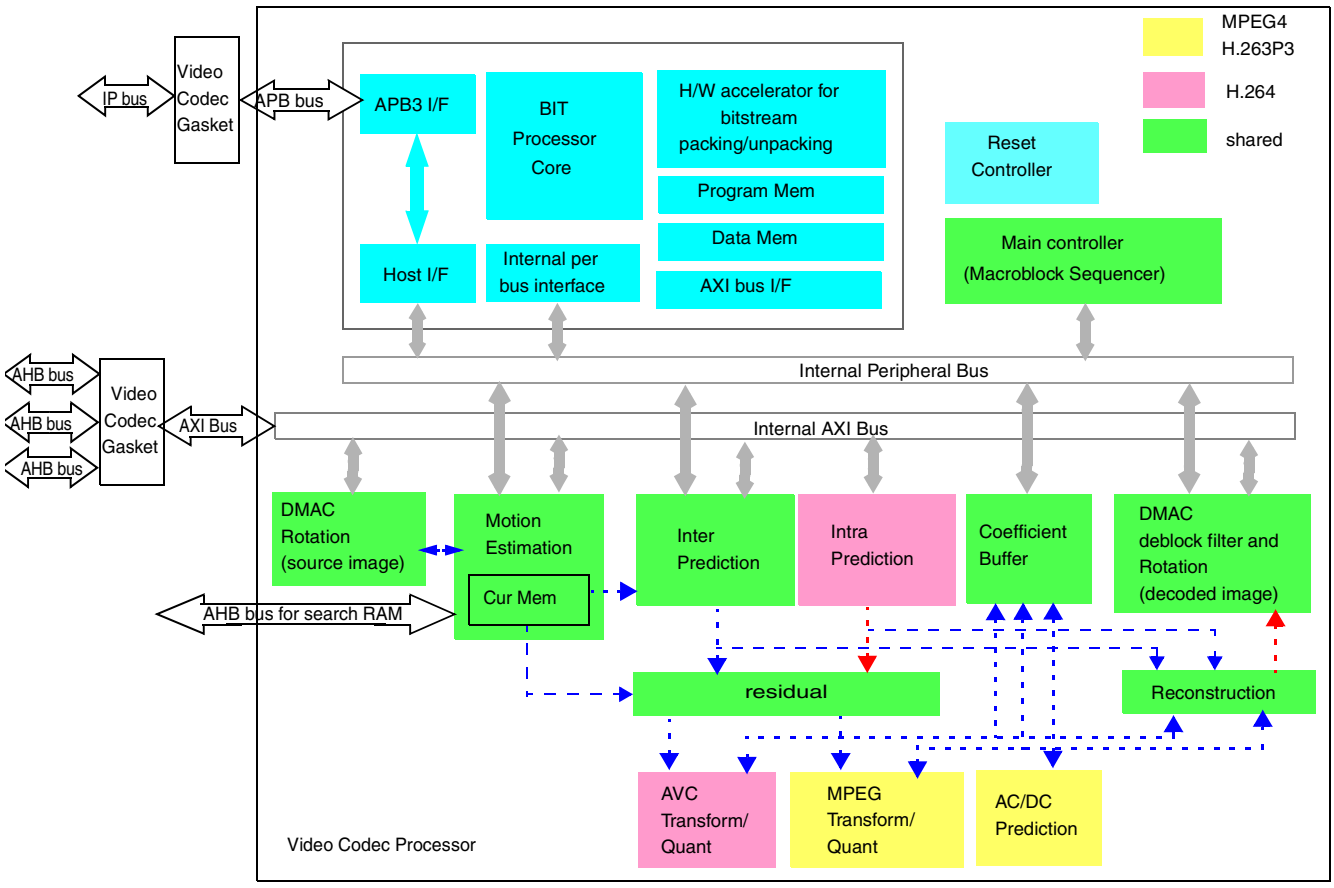


Figure 40-1. Video Codec Block Architecture Diagram

40.1 Features

Video Codec module support following multimedia video stream processing features:

- Multi-standard video codec
  - MPEG-4 part-II simple profile encoding/decoding
  - H.264/AVC baseline profile encoding/decoding
  - H.263 P3 encoding/decoding
  - Multi-party call: max processing 4 image/bitstream encoding and/or decoding simultaneously.

— Multi-format: encodes MPEG4 bitstream, and decodes H.264 bitstream simultaneously.

- Coding tools
  - — high-performance motion estimation.
    - – Single reference frame for both MPEG4 and H.264 encoding.
    - – Support 16 reference frame for H264 decoding
    - – Quarter-pel and half-pel accuracy motion estimation
    - – [+/–16, +/–16] Search range
    - – Unrestricted motion vector
  - — MPEG-4 AC/DC prediction and H.264 Intra prediction.
  - — All variable block sizes are supported (in case of encoding, 8X4, 4X8, and 4X4 block sizes are not supported).
  - — H.263 Annex I, J, K (RS = 0 and ASO =0), and T are supported. In case of encoding, Annex I and K (RS=1 or ASO=1) are not supported.
  - — CIR (Cyclic Intra Refresh)/AIR (Adaptive Intra Refresh)
  - — Error resilience tools.
    - – MPEG-4 re-synchronize marker and data-partitioning with RVLC (fixed number of bits/macroblocks between macroblocks)
    - – H.264/AVC FMO and ASO
    - – H.263 slice structured mode
  - — Bit-rate control (CBR and VBR)
- Pre/post rotation/mirroring
  - — 8 rotation/mirroring modes for image to be encoded
  - — 8 rotation/mirroring modes for image to be displayed
- Programmability
  - — Embeds 16-bit DSP processor that is dedicated to process bitstream and drive codec hardware
  - — General purpose registers and interrupt generation for communication between the system and the Video Codec module

40.2 Overview

The Video Codec module in i.MX27 device supports full duplex video codec processing and multi-party calls. It integrates multiple video processing standard together, including H264 BP, MPEG4 SP, and H263 P3 (including annex I, J, K, and T).

The Video Codec uses two bus interface protocols: the IP bus for register access control and the AHB bus for data throughput. The Video Codec module uses three memory components: embedded memory, system internal memory and system external memory. Embedded memories include dual-port register files, single-port register file, dual-port SRAM, and single-port SRAM. System internal memory is used by motion estimation to increase codec performance. System external memory is used to store input/output image pixel data and bit-stream data.

Video Codec module mainly includes two hardware components:

- Video Codec Processor: It is the heart of video codec processing. It supports multiple video processing standard, and integrates encoding and decoding function together with 32-bit AXI interface.
- Video Codec Interface: Composed of two parts, one responds to bus protocol transfers between 32-bit AXI bus interface and three 32-bit AHB-Lite master bus interfaces (two read channels and one write channel). The other responds to bus protocol transfers between 32-bit APB bus and IP bus.

Encoder and decoder processing share data-paths in the Video Codec module. There is an embedded BIT processor which is used to control the hardware sequence and as well as bitstream processing. The whole encoding or decoding is controlled by the firmware running on BIT processor in Video Codec. The host processor only need to access the Video Codec registers for initializing the Video Codec or setting frame parameters during the encoding/decoding frame gap.

The Video Codec module has 3 clock domains: IP bus clock, video codec core clock, and the AHB bus clock. The Video Codec Processing IP core clock is asynchronous to AHB clock and IP clock. All sequential logic use only rising edge of clocks.

40.3 Clock Domain and Reset

40.3.1 Clocks

There are three clock domains in Video Codec:

- AHB bus clock domain (hclk): Controls all AHB or AXI bus related functions. Maximum frequency is 133 MHz. The hclk is derived from PLL Clock Reset module.
- Codec core clock domain (cclk): This is the master clock for the video codec. It controls all video codec encoding/decoding functionality, at a maximum frequency of 133 MHz. The actual value of core clock is dependent upon application use case.
- IP bus clock domain (ipg_clk_s): Controls Video Codec registers read/write function, with a maximum frequency of 66.5 MHz. ipg_clk_s is a gated clock of IP clock (ipg_clk) with ips_module_en, it is turned off when there is no registers read/write for power saving.

Only positive edge clocks is used in Video Codec design. All clock domains are asynchronous to the others.

The Video Codec Processor uses all the three clock domains because it is needed for AXI signal generation, functional calculation, and APB bus configuration. The Video Codec interface clock belongs to AHB clock domain and IP bus clock domain, because they are related to AHB and AXI, IP and APB bus protocol transfers.

40.3.2 Reset

Corresponding to the three clock domains, there are three reset signals in Video Codec module, active low.

- AHB bus reset: used in AHB bus interface. Corresponding clock is hclk.

- Codec core reset: used in codec accelerating hardware. Corresponding clock is cclk.
- IP bus reset: used in IP bus interface. Corresponding clock is ipg_clk.

The number of cycles for each reset signal must be at least 8 cycles.

The Video Codec uses an internal reset controller for feature software reset from the BIT processor. If any of the Video Codec blocks, with the exception of BIT processor is needed to reset (software reset), the host processor can enable this software reset by setting the software reset register through Video Codec API. The BIT processor cannot be reset by this software reset scheme, because the reset signal of BIT processor is connected directly to an external reset signal.

If reset occurs when the Video Codec is processing a transaction through AHB bus, there is no guarantee that AHB bus will complete the transaction normally, since the Video Codec will be reset. If there are any corrupted data in memory, it can be discarded by software. Basically, if the host processor needs to issue a reset, it must check to ensure that there is no transaction on AHB bus between Video Codec and external AHB bus interface. In general, the AHB bus is free of Video Codec transactions after one frame of decoding/encoding is completed. The start of next frame processing requires software initiation.

NOTE

Both AHB read channels only allows read access, its corresponding AHB master bus has no hwdata signal, and hwrite is connected to ‘0’. Similarly, AHB write channel only allow write access, its corresponding AHB master bus has no hrdata signal, and hwrite is connected to ‘1’.

40.4 Memory Map and Register Definition

The Video Codec registers are all 32-bit wide and only support 32-bit aligned read/write operations. Video Codec registers are grouped into several regions corresponding to different codec processing stages. The registers are used for codec processing configuration and control. They can only be accessed through IP bus interface.

40.4.1 Memory Map

Video Codec module includes several internal register address map space, as shown in Table 40-1. The Video Codec module uses 0x10023000–0x10023FFF memory space as register mapping in i.MX27 system. The Video Codec module registers are divided into two categories.

- Address 0x1002_3000–0x1002_30FC (64 registers address space) are hardware registers. These registers have reset values and their functions are fixed (can not be configurable).
- Address 0x1002_3100–0x1002_31FC (64 registers address space) are software registers. They have no reset values and are configurable by internal BIT processor. So their definitions are not provided here. They can be used as general parameter registers between host and BIT processor.
  - The first 32 parameter registers (address 0x1002_3100–0x1002_317C) are used as static parameters. The meaning and functions of these registers are not changed regardless of the run commands used.

— The second 32 parameter registers (address 0x1002_3180–0x1002_31FC) are used as temporal parameters. The meaning and functions of these registers may be changed for each run commands.

The memory map for the hardware registers of Video Codec is shown in Table 40-1.

Table 40-1. Video Codec Hardware Register Memory Map

Address	Register	Access	Reset Value	Section/Page
0xBASE_3000 (CodeRun)	BIT Processor run start	W	0x0000_0000	40.4.3.1/40-7
0xBASE_3004 (CodeDown)	BIT Boot Code Download Data register	W	0x0000_0000	40.4.3.2/40-8
0xBASE_3008 (HostIntReq)	Host Interrupt Request to BIT	W	0x0000_0000	40.4.3.3/40-8
0xBASE_300C (BitIntClear)	BIT Interrupt Clear	W	0x0000_0000	40.4.3.4/40-9
0xBASE_3010 (BitIntSts)	BIT Interrupt Status	R	0x0000_0000	40.4.3.5/40-10
0xBASE_3014 (BitCodeReset)	BIT Code Reset	W	0x0000_0000	40.4.3.6/40-10
0xBASE_3018 (BitCurPc)	BIT Current PC	R	0x0000_0000	40.4.3.7/40-11
0xBASE_301C–0xBASE_30FC	Reserved	—	—	—

40.4.2 Register Summary

The conventions in Figure 40-2 and Figure 40-2 serve as a key for the register summary and individual register diagrams.

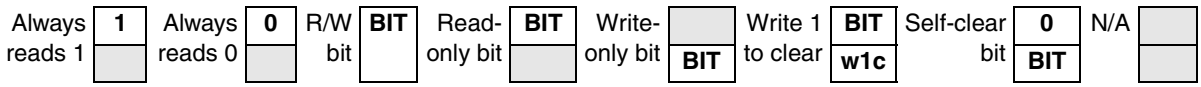


Figure 40-2. Key to Register Fields

Table 40-2 provides a key for register figures and tables and the register summary.

Table 40-2. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)

Table 40-2. Register Conventions (continued)

Convention	Description
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

The brief Video Codec hardware register summary is shown in Table 40-3.

Table 40-3. Video Codec Hardware Register Summary

Name		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
0xBASE_3000 (CodeRun)	R																
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	R																
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CodeRun
0xBASE_3004 (CodeDown)	R																
	W	0	0	0	CodeAddr												
	R																
	W	CodeData															
0xBASE_3008 (HostIntReq)	R																
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	R																
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IntReq
0xBASE_300C (BitIntClear)	R																
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	R																
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IntClear
0xBASE_3010 (BitIntSts)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IntSts
	W																

Table 40-3. Video Codec Hardware Register Summary (continued)

Name		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
0xBASE_3014 (BitCodeReset)	R																
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	R																
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CodeReset
0xBASE_3018 (BitCurPc)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	CurPc													
	W																

40.4.3 Register Descriptions

This section consists of Video Codec hardware register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

40.4.3.1 Video Codec Code Run Register (CodeRun)

See Figure 40-3 for an illustration of valid bits in Video Codec Code Run Register and Table 40-4 for descriptions of the bit fields in the register.

0xBASE_3000 (CodeRun) Access: User Write-Only

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

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

Figure 40-3. Video Codec Code Run Register

Table 40-4. Video Codec Code Run Register Field Descriptions

Field	Description
31–1	Reserved
0	CodeRun. BIT processor run start bit. 0 BIT Processor stop execution 1 BIT Processor start execution

40.4.3.2 Video Codec BIT Boot Code Download Data Register (CodeDown)

See Figure 40-4 for an illustration of valid bits in Video Codec BIT Boot Code Download Data Register and Table 40-5 for descriptions of the bit fields in the register.

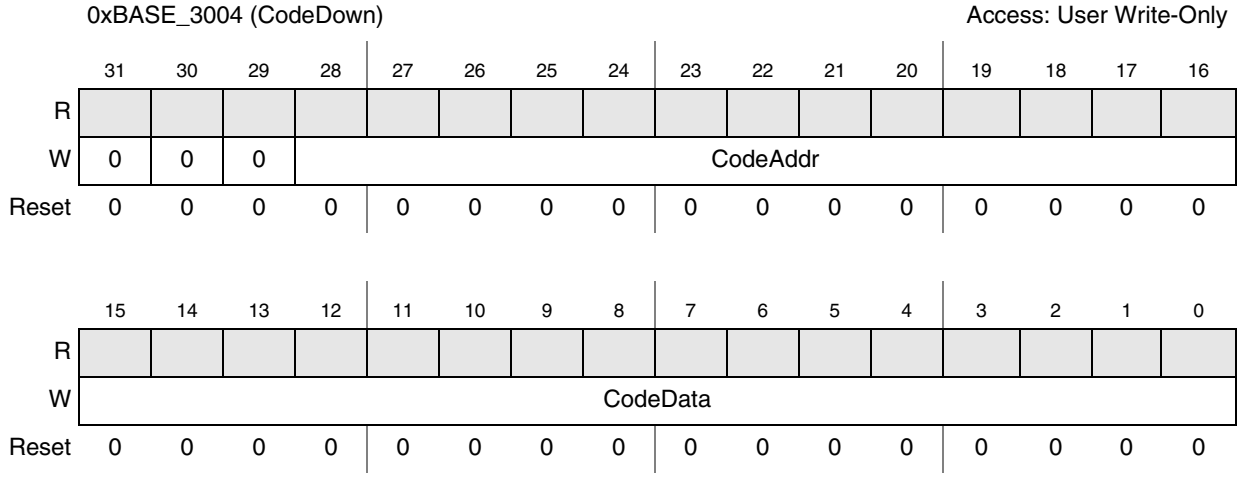


Figure 40-4. Video Codec BIT Boot Code Download Data Register

Table 40-5. Video Codec BIT Boot Code Download Data Register Field Descriptions

Field	Description
31–29	Reserved
28–16	CodeAddr[12:0]. Download address of Video Codec BIT boot code, which is Video Codec internal address of BIT processor.
15–0	CodeData[15:0]. Download data of Video Codec BIT boot code.

40.4.3.3 Video Codec Host Interrupt Request Register (HostIntReq)

See Figure 40-5 for an illustration of valid bits in Video Codec Host Interrupt Request Register and Table 40-6 for descriptions of the bit fields in the register.

0xBASE_3008 (HostIntReq) Access: User Write-Only

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

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

Figure 40-5. Video Codec Host Interrupt Request Register

Table 40-6. Video Codec Host Interrupt Request Register Field Descriptions

Field	Description
31–1	Reserved
0	IntReq. The host interrupt request bit. 0 No host interrupt is requested. 1 The host processor request interrupt to the BIT processor.

40.4.3.4 Video Codec BIT Interrupt Clear Register (BitIntClear)

See Figure 40-6 for an illustration of valid bits in Video Codec BIT Interrupt Clear Register and Table 40-7 for descriptions of the bit fields in the register.

0xBASE_300C (BitIntClear) Access: User Write-Only

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

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

Figure 40-6. Video Codec BIT Interrupt Clear Register

Table 40-7. Video Codec BIT Interrupt Clear Register Field Descriptions

Field	Description
31–1	Reserved
0	IntClear. BIT interrupt clear bit. 0 No operation is issued. 1 Clear the BIT interrupt to the host.

40.4.3.5 Video Codec BIT Interrupt Status Register (BitIntSts)

See Figure 40-7 for an illustration of valid bits in Video Codec BIT Interrupt Status Register and Table 40-8 for descriptions of the bit fields in the register.

0xBASE_3010 (BitIntSts) Access: User Read-Only

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

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

Figure 40-7. Video Codec BIT Interrupt Status Register

Table 40-8. Video Codec BIT Interrupt Status Register Field Descriptions

Field	Description
31–1	Reserved
0	IntSts. BIT interrupt status bit. 0 No BIT interrupt is asserted. 1 The BIT interrupt is asserted to the host. It is cleared when the host processor write “1” to BitIntClear register.

40.4.3.6 Video Codec BIT Code Reset Register (BitCodeReset)

See Figure 40-8 for an illustration of valid bits in Video Codec BIT Code Reset Register and Table 40-9 for descriptions of the bit fields in the register.

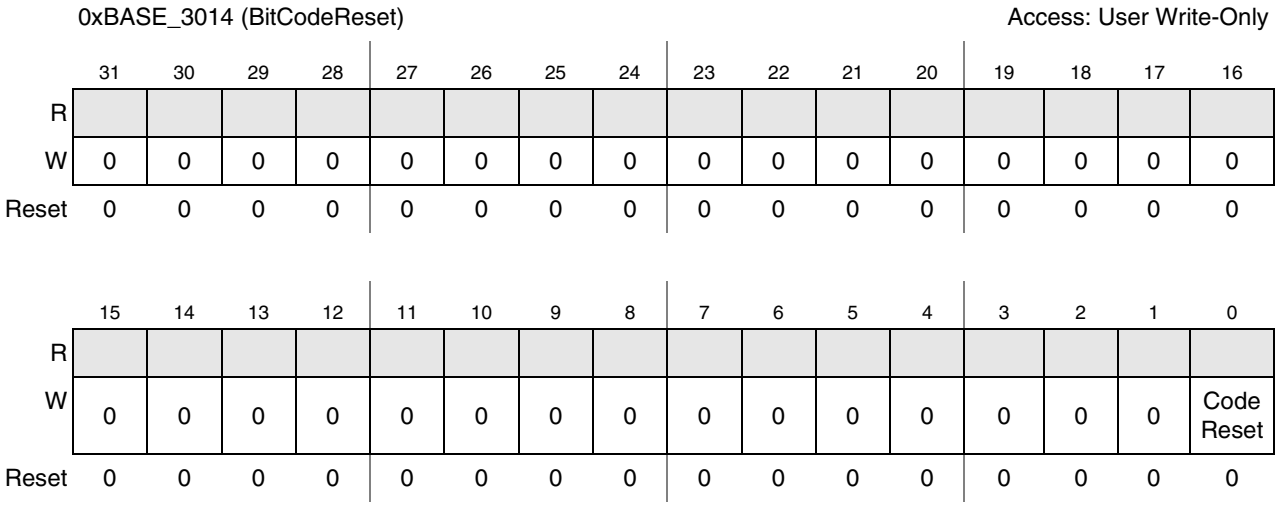


Figure 40-8. Video Codec BIT Code Reset Register

Table 40-9. Video Codec BIT Code Reset Register Field Descriptions

Field	Description
31–1	Reserved
0	CodeReset. BIT code reset bit. 0 No operation is issued. 1 The program counter of BIT processor is set to “0”, BIT processor restart at initial routine.

40.4.3.7 Video Codec BIT Current PC Register (BitCurPc)

See Figure 40-9 for an illustration of valid bits in Video Codec BIT Current PC Register and Table 40-10 for descriptions of the bit fields in the register.

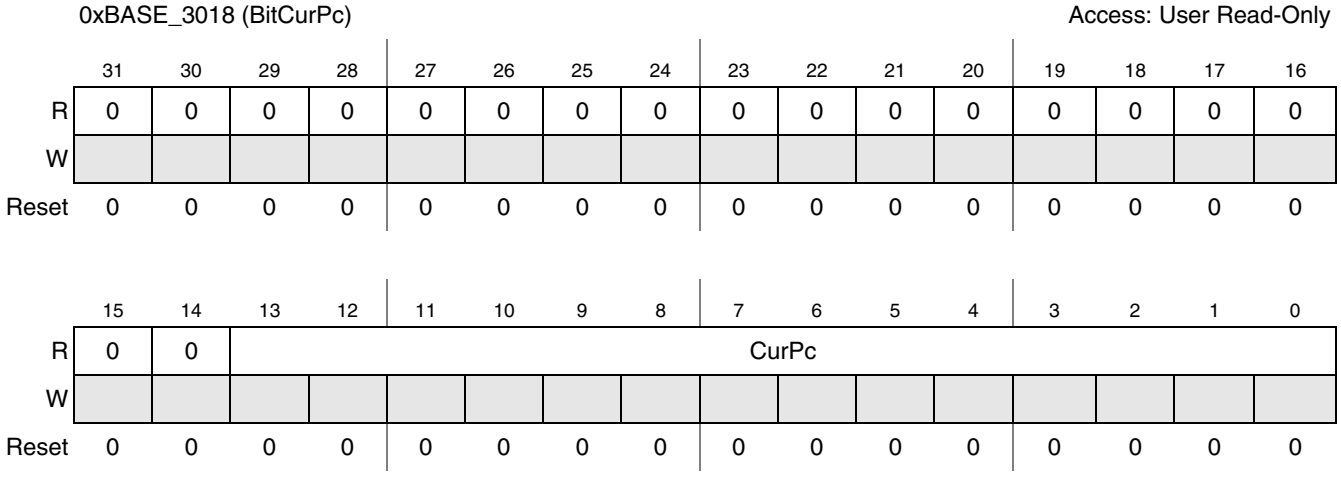


Figure 40-9. Video Codec BIT Current PC Register

Table 40-10. Video Codec BIT Current PC Register Field Descriptions

Field	Description
31–14	Reserved
13–0	CurPc[13:0]. BIT current PC value. Returns the current program counter of BIT processor by reading this register.

40.5 Functional Description

The Video Codec module is a high-performance multi-standard video processing unit in the system which supports H.263P3, MPEG-4 SP, and H.264 BP.

40.5.1 Video Codec Architecture

The Video Codec module mainly includes two hardware components: Video Codec Processing IP and Vide Codec Gasket. Video Codec Processing IP is optimized to reduce logic gate count with many sharing parts of sub-modules for multi-standard. It is responsible for bitstream parsing and frame data coding. It mainly consists an embedded 16bit BIT processor, codec hardware accelerator,and bus arbiter/interface. BIT processor is in charge of parsing/coding the bitstream/image, controlling video codec process. Hardware accelerators are designed to speed up bitstream/image processing. Video Codec Gasket converts AMBA APB3 bus to IP Sky Blue bus and AXI bus to AHB bus. Refer to Figure 40-1 for the block diagram of Video Codec. Below sections describe the main function of Video Codec Processing IP components.

40.5.1.1 Embedded BIT processor

The embedded BIT processor is 16-bit programmable processor which is highly optimized to handle bitstream data. It is used for parsing or forming bitstream. It includes some hardware accelerators to speed up the bitstream processing. In addition to handling bitstream, the BIT processor controls video codec hardware and communicates with host processor through IP Sky Blue bus and AXI bus interface.

Before running codec, BIT firmware common routine should be downloaded into embedded program memory through system level control. For codec programing data, BIT processor read it from a specified region of system external memory through AHB bus interface. The region is specified by application settings.

40.5.1.2 Codec Hardware Accelerator

All video codec processing, except handling coefficients for VLC and VLD, are implemented in hardware accelerator. Codec hardware accelerator is designed to reduce logic gate count by sharing parts of sub-modules for multi-standard video encoding and decoding.

Codec hardware accelerator supports rotation/mirroring function. In case of rotating/mirroring source image during encoding, rotated image is encoded directly without writing it to any memory space. So no additional bandwidth is needed for the processing. However, the rotation/mirroring process in decoding process requires additional bandwidth because decoding has to re-use the un-rotated image for decoding the next image. So the rotated image is written to other memory space. In this scheme, the display I/F has

not to change memory space for displaying the decoded image because subsequent rotated image is written to the same space.

The rotator modules support 8-types mode of 90 x n degree (n=0, 1, 2, 3) rotating and mirroring. Table 40-11 shows the supported rotating/mirroring lists and some possible combinations of rotating and mirroring. Figure 40-10 gives architecture diagram of pre/post rotation/mirroring module.

Table 40-11. Rotation and Mirroring Mode

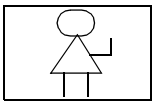
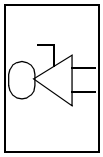
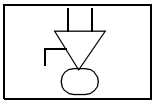
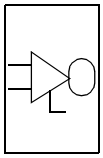
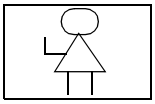
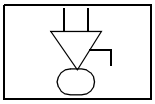
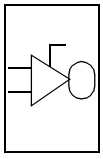
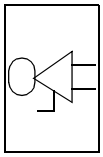
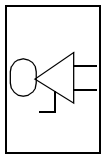
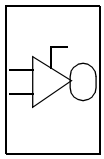
Image	Description
	Original Image (No rotating/mirroring) Example Image Size: 720x480
	Rotate Left 90 (Rotate Right 270) Example Image Size: 480x720
	Rotate Left 180 (Rotate Right180) Example Image Size: 720x480
	Rotate Left 270 (Rotate Right 90) Example Image Size: 480x720
	Horizontal mirroring Example Image Size: 720x480
	Vertical mirroring Example Image Size: 720x480
	Horizontal mirroring and rotate right 90 Example Image Size: 480x720
	Horizontal mirroring and rotate left 90 Example Image Size: 480x720

Table 40-11. Rotation and Mirroring Mode

Image	Description
	Rotate right 90 and horizontal mirroring Example image size: 480x720
	Rotate left 90 and horizontal mirroring Example image size: 480x720

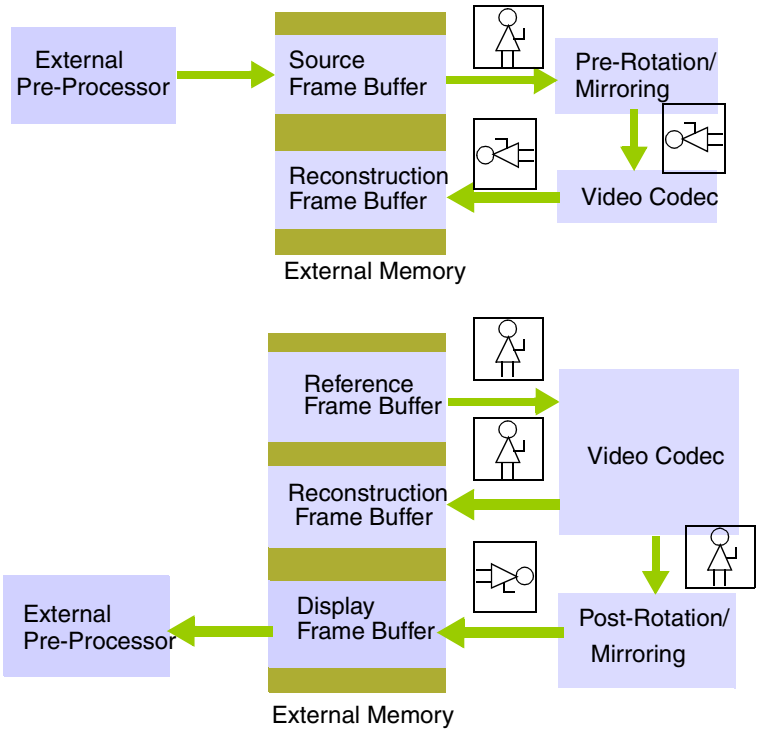


Figure 40-10. Rotation and Mirroring Data Flow

40.5.2 Interrupts

There is one interrupt signal output from Video Codec. Basically, this interrupt is used to indicate encoding/decoding processing state. It is generated when Video Codec interrupt is enabled and as well as the interrupt condition is met. The interrupt signal is active high and retains till the host processor clears it by writing “1” to the interrupt clear register. This signal is synchronized to the positive edge of hclk.

When getting frame completion interrupt, the software needs to set the parameters for the processing of the next frame and start the BIT processor again. The parameters mainly include source/destination frame

buffer base address. It can be different from previous frame buffer since the previous completed frame of data may be needed for other image block like display module or post-processing module in system.

Basically, operation responding to interrupt is dependent upon the application. For example, software can send the decoded frame to EMMA for post-processing, or transfer the encoded bitstream for storage, at the same time software could store the next bitstream to be processed to the external memory before starting a new encoding processing.

40.6 Initialization Information

Video Codec module embedded BIT processor is 16-bit programmable processor, which is highly optimized to handle bitstream/image data. In addition to processing bitstream/pixel data, BIT processor also controls the communicates between Video Codec module and system.

40.7 Application Information

Figure 40-11 shows roles of BIT processor and codec hardware accelerator, and how to interface with application software. At the frame level, a host processor communicates with Video Codec through provided APIs. To give Video Codec more flexibility and debugging capability, all processes related to the bitstream are assigned to the BIT processor.

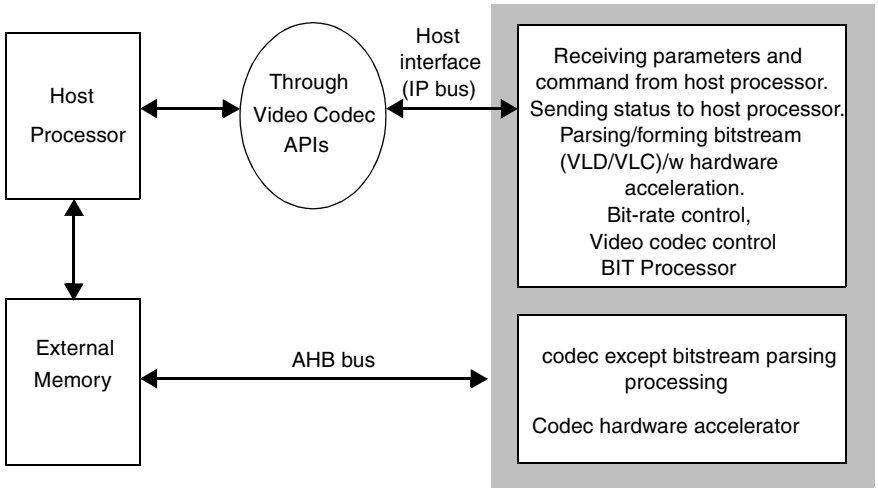


Figure 40-11. Video Codec Interface with Application Software Diagram

40.7.1 Video Codec Processing Control

This section describes how BIT processor controls video codec processing and communicates with the host.

40.7.1.1 Video Codec Processing Flow

The Video Codec can handle a maximum of 4 processes simultaneously. Each process can have a different format—MPEG-4, H.263P3, H.264, and different codec process—encoding or decoding. Figure 40-12 shows a simplified state diagram for running the codec process.

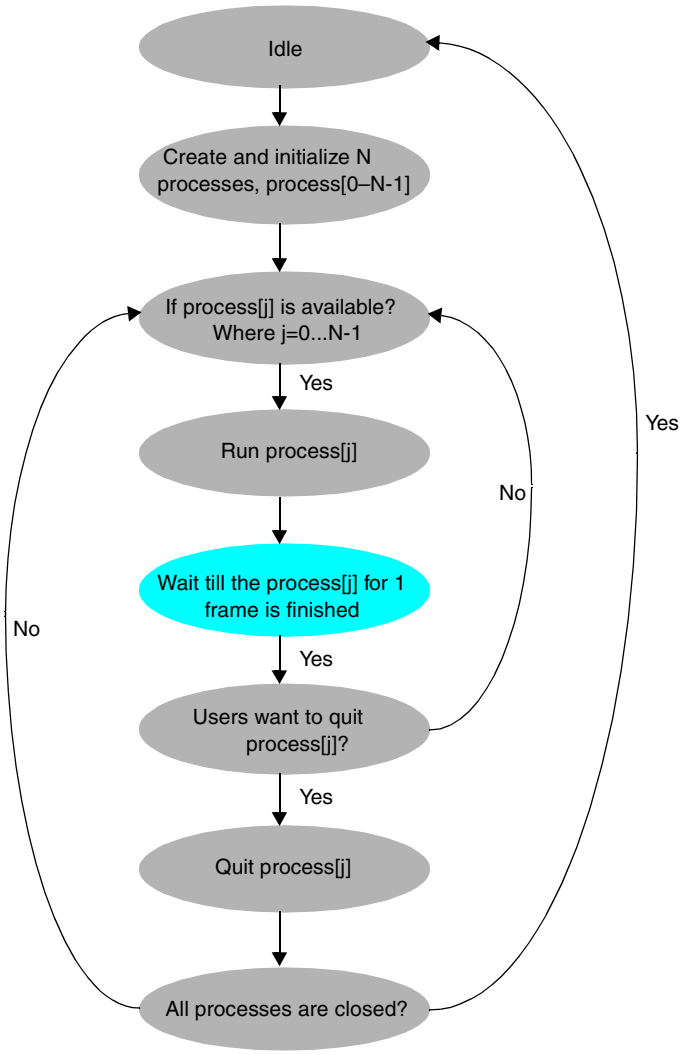


Figure 40-12. Codec Process State Diagram

Each codec process consists of three categories:

- Create processes: Software creates and configures processes.
- Running processes: At a proper time instance, software will begin a specific process. The proper time instance means when the codec is in idle state and image to be encoded or bitstream to be decoded is ready in the external memory.
- Quit Processes: Software can quit a specific process

If more than one process are ready to run, each process must be assigned to different process ID—RunIndex, which is range from 0 to 3. Basically, the ID is assigned based on the order of creation. For example, when 1 MPEG-4 Decoding + 1 H.264 Decoding + 1 H.263 Decoding + 1 H.264 Encoding are running simultaneously, MPEG-4 Decoding is assigned to process index “0”, H.264 Decoding is assigned to process index “1”, H.263 Decoding is assigned to index “2”, and H.264 Encoding is assigned to process index “3”.

There is no priority rules for executing processes, after creating all processes at the initialization step, host enables BIT processor to execute process specified with the RunIndex. All processes are executed in time-division like mechanism, after one process finishes encoding or decoding a frame, another process then can be executed.

In conjunction with the process ID, RunCodStd needs to be set, to define which coding standard is used with the created process and whether the created process will encode an image or decode a bitstream. Table 40-12 shows the dedicated RunCodStd value for each coding standard. All this can be done through Video Codec API.

Table 40-12. RunCodStd Register Value for Coding Standard

Coding Standard	RunCodStd
MPEG-4/H.263 P3 Decoding	0
MPEG-4/H.263 P3 Encoding	1
H.264 Decoding	2
H.264 Encoding	3

40.7.1.2 Video Codec Processing Finish Detection

The Video Codec module raises interrupt signal or busy state when frame processing command is finished. So there are three ways of detecting whether decoding/encoding of a frame is finished:

- Polling Video Codec interrupt status register. Interrupt status register indicates interrupt is generated.
- Polling Video Codec busy status register. During decoding/encoding process, as soon as the busy status becomes 0, decoding/encoding processing is finished.
- Capture interrupt signal from system level, respond to interrupt request within the system interrupt service routine.

NOTE

Interrupt status can be cleared by writing 1 to interrupt clear register. Busy state is self cleared after read out.

40.7.1.3 Video Codec Processing Flow Example

Figure 40-13 shows a process flow example for decoding an H.264 bitstream and encoding images as H.264 format simultaneously. At first both decoding and encoding process are created and initialized, then each process is executed with PICTURE_RUN command alternately. More details are described as below.

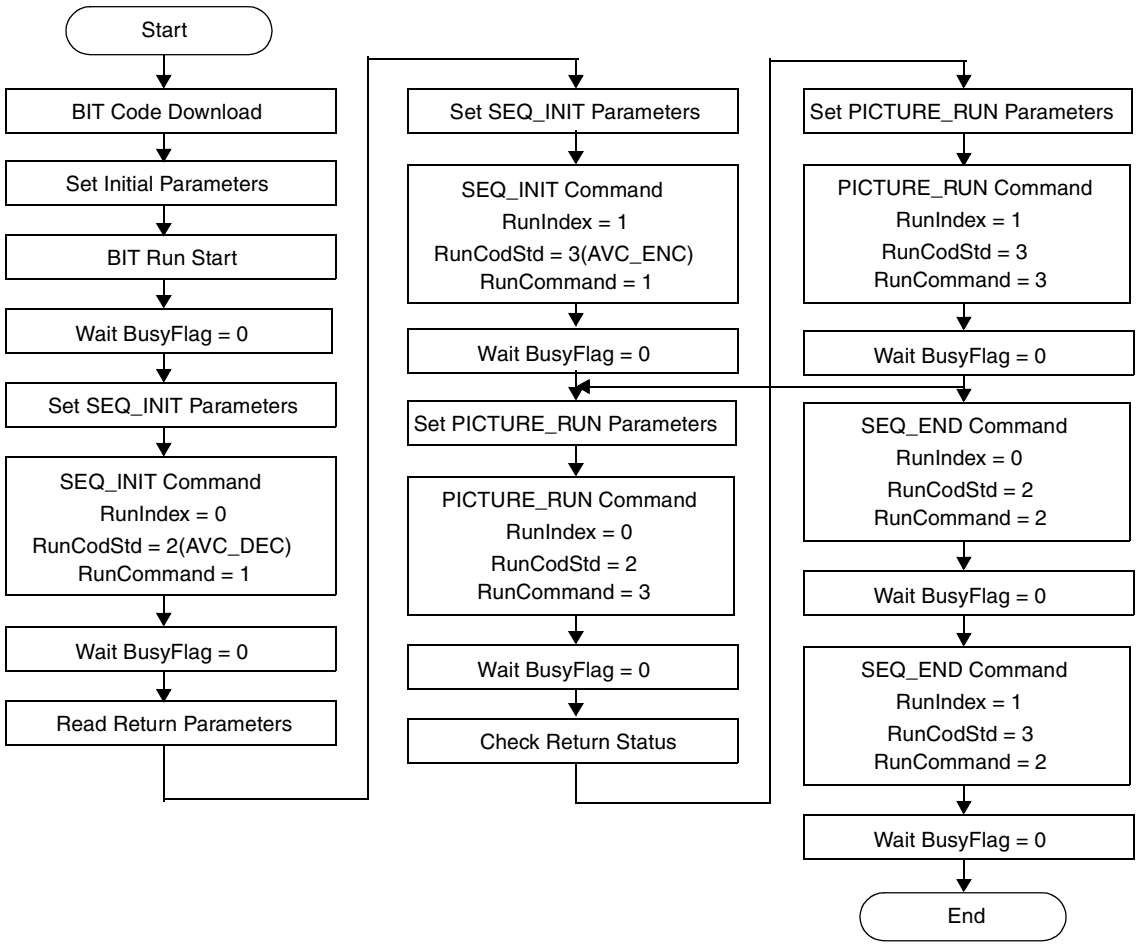


Figure 40-13. H.264 Codec Process Flow Example

*RunCommand = 1 (SEQ_INIT); RunCommand = 2 (SEQ_END);
RunCommand = 3 (PICTURE_RUN)

1. Initialize the Video Codec
   - BIT Code Download: Load BIT Processor firmware to memory.
   - Set Initial Parameters: General configuration for BIT processor, setting working buffer base address, BIT Code memory address, bitstream buffer control and so on.
   - BIT Run Start: Run BIT processor to initialize Video Codec.
2. Create and initialize an H.264 decoding process
   - Set SEQ_INIT parameters: Configure base address and size of bitstream buffer, base address of frame buffers and so on.
   - Run SEQ_INIT command: Initiate an H.264 decoding process.
   - Wait BusyFlag=0: Wait BIT processor completes SEQ_INIT command execution.

— Read Return Parameters: Read the features of decoded bitstream, such as the picture resolution and number of reference frames through the Video Codec API. In this way, the host can prepare the required frame buffers.

3. Create and initialize an H.264 encoding process
   — The flow is similar to the H.264 decoding process except the stage of Read Return Parameters, the encoding frame buffer size can be configured according to the feature of video clips.
4. Run the H.264 decoding process
   — Set PICTURE_RUN Parameters: Configure the frame destination address.
   — Run PICTURE_RUN command: Start the H.264 decoding process.
   — Wait BusyFlag=0: Wait the BIT processor completes PICTURE_RUN command execution. It also means one frame process is finished. The decoded frame can be sent to the EMMA for post-processing. The actual operation is dependant on the application.
5. Run the H.264 encoding process
   — The flow is similar to the H.264 decoding process. The encoding process should configure frame source address in addition to destination address.
6. Execute step 4 and step 5 alternately.
   — Before running decoding process, the host should load new bitstream to the bitstream buffer if it is empty, and update the frame destination address according to the application. As for encoding, the next frame should be ready in external memory before the process is run.
7. Stop the codec process
   — Run SEQ_END command to each process to terminate it.

Basically, the process flow for encoding and decoding is similar, though it may have minor change for different firmware version.

40.7.1.4 Frame Buffer

This section describes the memory map of the frame buffer and size requirement.

Frame buffer is specified with the base address and stride line. A complete image consists of Y, U and V component. Therefore, an image frame has 3 component buffers for each component. Stride line is the width of the luminance component buffer in pixel unit and must be multiple of 8. Stride line for chrominance component buffer is a half of the luminance component. Video Codec module supports 11-bit stride line configuration which can be larger or equal to the width of image frame.

The relationship between image size and stride line is shown in Figure 40-14. YUV components are stored line-by-line. U component in one image frame is stored adjoining Y component of the same frame. V component follows U component. Next frame data follows previous frame YUV component data. The relationship between image size and stride line also decides the memory map distribution of codec image data in external memory space. Between every YUV component line data, there is some optional unused memory space which size is (Stride_Line - Image_Width) Byte. It is greater than or equal to ZERO byte, and can not be larger than (2^11 - Image_Width) (because Stride_Line is 11bit width).

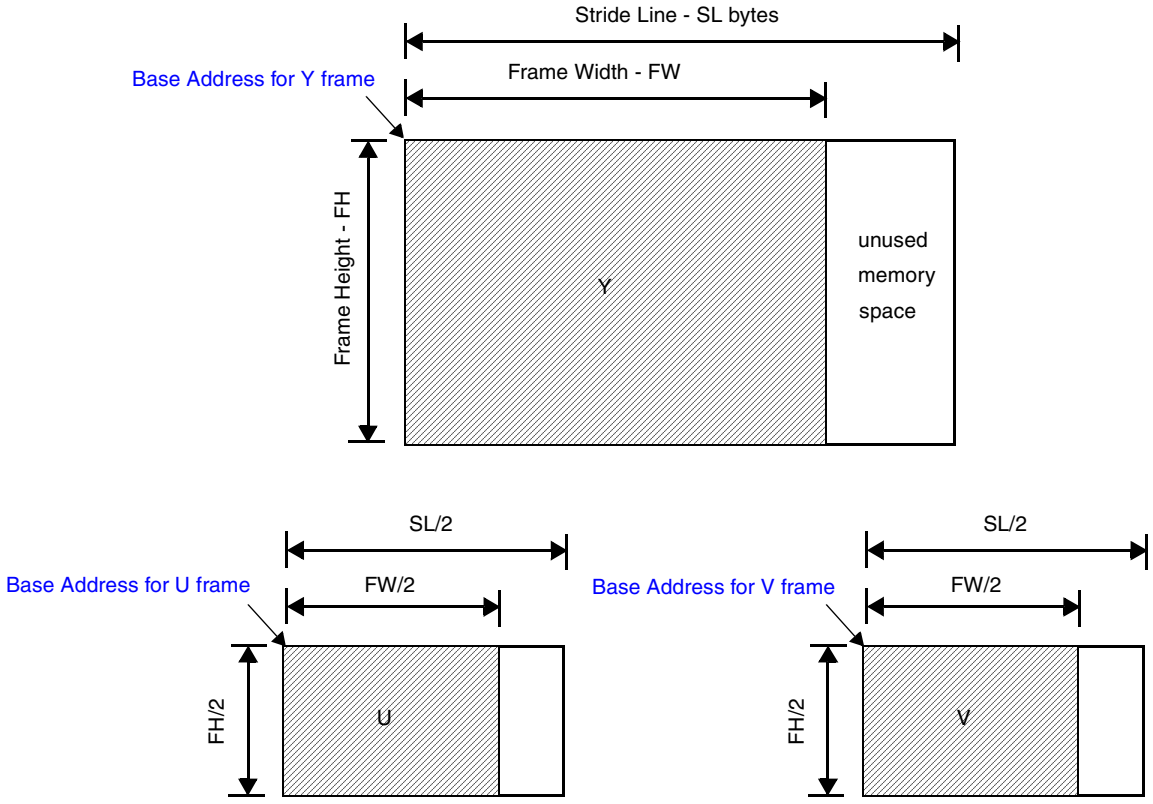


Figure 40-14. Frame Buffer Configuration

Figure 40-15 shows the memory map of frame buffer. For V frame buffer, memory map is the same as U frame buffer except the base address.

Video Codec supports both little and big endian system. It means Y(0,0) in Figure 40-15 could be located in the bit[31:24]. User can specify the endianness through Video Codec API.

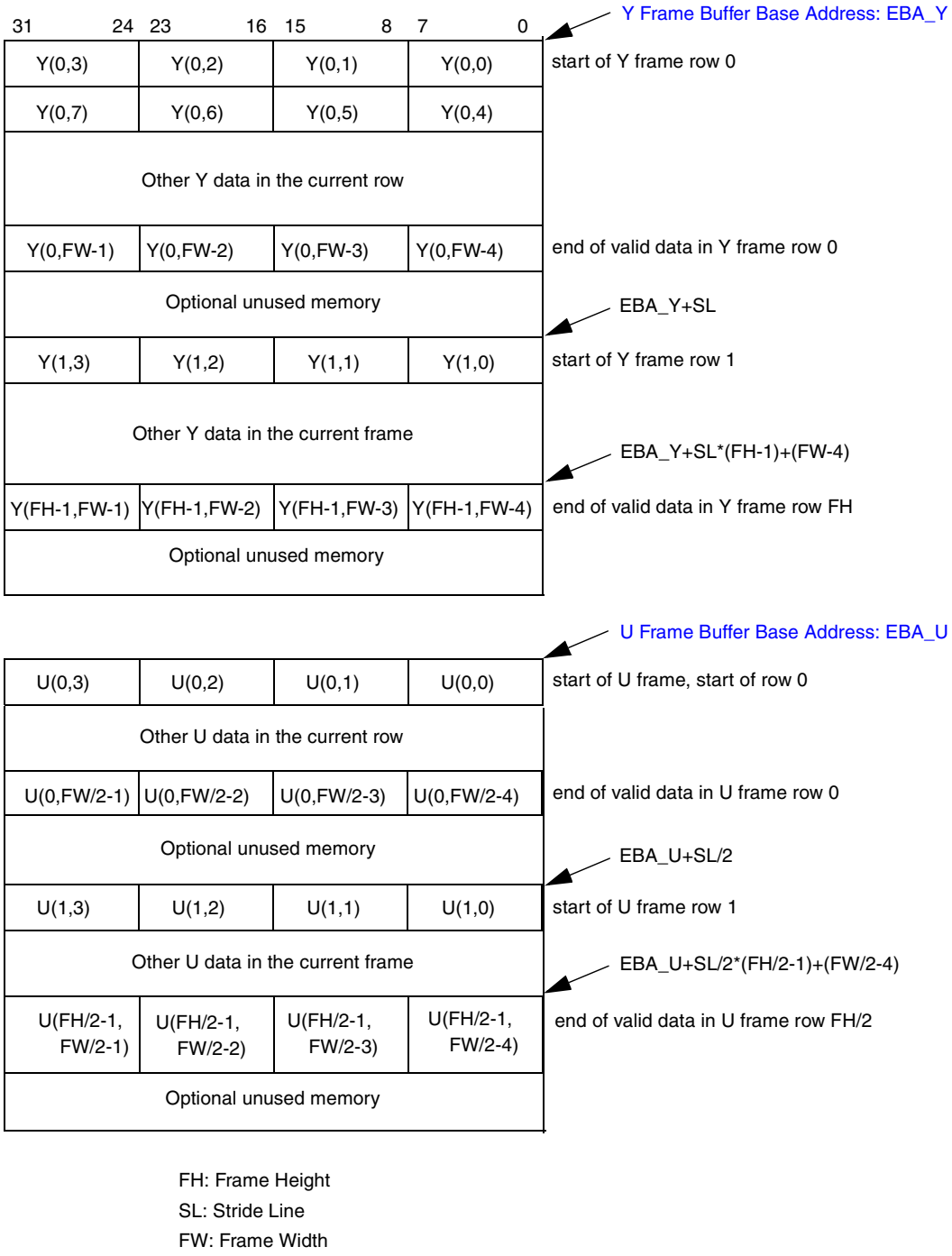


Figure 40-15. Frame Buffer Address Map in Little Endian

Table 40-13 also shows the memory requirement in case of QCIF/CIF/VGA resolution image. In the case of H.264, the required size for the reference frame is dependent on the level being supported. Video Codec supports up to H.264 level 3.0, which the maximum decoded picture buffer size is defined as 3037.5

Kbytes in the standard. To support H.264 CIF at level 3.0, 2524 Kbyte is needed if 16 reference frames are used.

Table 40-13. Frame Buffer Requirement

		MPEG-4 Encoder	MEPG-4 Decoder	H.264 Encoder	H.264 Decoder
QCIF	Reference Frames	1	1	1	16
	Current Frames	1	0	1	0
	Reconstruction Frames	1	1	2	1
	Display Frame	0	1	0	1
	Total Frames	3	3	4	18
	Picture Size[1]	37 Kbyte	37 Kbyte	37 Kbyte	37 Kbyte
	Total Frame Size	111 Kbyte	111 Kbyte	148 Kbyte	666 Kbyte
CIF	Reference Frames	1	1	1	2376 Kbyte
	Current Frames	1	0	1	0
	Reconstruction Frames	1	1	2	1
	Display Frame	0	1	0	1
	Total Frames	3	3	4	2
	Picture Size	148 Kbyte	148 Kbyte	148 Kbyte	148 Kbyte
	Total Frame Size	444 Kbytes	444 Kbytes	592 Kbytes	2672 Kbytes
VGA	Reference Frames	1	1	1	3037.5 Kbytes
	Current Frames	1	0	1	0
	Reconstruction Frames	1	1	2	1
	Display Frame	0	1	0	1
	Total Frames	3	3	4	2
	Picture Size	450 Kbytes	450 Kbytes	450 Kbytes	450 Kbytes
	Total Frame Size	1350 Kbytes	1350 Kbytes	1800 Kbytes	3937.5 Kbytes

[1] The Picture Size is the minimum size of one frame buffer with assumption that the picture is YUV 4:2:0 format and the stride line is equal to frame width.

40.7.1.5 BIT Processor Program Memory

At the initialization stage of Video Codec, host processor must download common routine to BIT processor. After initialization, BIT processor loads a program corresponding to the activated standard.

There are total 65 Kbyte size for current version of BIT firmware to support three standards (MPEG-4, H.263 P3 and H.264), including 1 Kbyte common routine.

40.7.1.6 Working Buffer

Besides buffers for frames and firmware, additional working buffer for intermediate data from BIT processor and codec processing is needed. The buffers are such as the reconstructed pixel row buffer for MPEG-4 AC/DC prediction or H.264 intra prediction, context saving buffer for running multiple processes, bitstream re-ordering buffer for MPEG-4 data partition or H.264 FMO/ASO and so on.

The required working buffer size varies according to codec size, standard and capability. For example, AC/DC prediction buffer size is determined by picture width and the maximum bitstream re-ordering buffer for data partition is determined by the maximum bitstream size of one picture. Working buffer size may change for different firmware version. The current version of firmware requires 256 Kbytes for working buffer when decodes/encodes 720 x 576 (D1 size) up to 10 Mbps. Its size can be set through Video Codec API. The detailed working buffer is organized as Table 40-14.

Table 40-14. Working Buffer Organization

Working Buffer	Description	Size
static Buffer	Used commonly in whole processes/codecs	48 Kbytes
temp_pic for MPEG4 decoding	AC/DC prediction buffer and bitstream reordering buffer for data partition.	16 Kbytes
temp_pic for MPEG4 encoding	AC/DC prediction buffer and bitstream reordering buffer for data partition.	16 Kbytes
temp_pic for AVC decoding	Intra-prediction buffer, FMO grout status buffer, and slice information buffer	208 Kbytes
temp_pic for AVC encoding	Intra-prediction buffer	72 Kbytes

40.7.1.7 Bitstream Buffer

Host processor has to assign buffers for bitstream on a per instance basis. If Video Codec handles N-bitstreams simultaneously in an application, the host should assign N bitstream buffers, and specify the base address and size. The External bitstream buffer is “ring buffer” type. Start address of ring buffer and buffer size must be written by host to BIT processor. The current read or write address of ring buffer is automatically wrapped-around by firmware.

In decoding case, host writes bitstream to be decoded then BIT processor reads the bitstream. In this case, the bitstream overwriting or underflow may occur and if it occurs, decoding will fail. To prevent overwriting or underflow, current bitstream read/write pointer must be exchanged between the host and BIT processor.

BIT processor writes current read pointer of ring buffer to internal register and host must write current write pointer of ring buffer to internal register. BIT processor checks the bit buffer empty (underflow) status by comparing current read pointer and write pointer. If no more bitstream data is available to be decoded (buffer empty status), BIT processor stops bitstream decoding to prevent mis-reading the

bitstream and waits until host writes more bitstream data and updates write pointer. Host must check the current read pointer and write pointer before writing more bitstream data to ring buffer to prevent overwriting bitstream data. Bitstream data is read from the external bitstream buffer by 512 bytes. The read pointer or write pointer is increased by 512 bytes.

40.7.1.8 Buffer Requirement Summary

Table 40-15 shows a summary of buffer requirement for each decoding instance, where bitstream buffer size is not considered because the size is not limited. The total size may change for different firmware version. Except BIT processor program memory, other kinds of buffer has to be assigned on a per instance basis. The overall buffer size for a multi-party call application is nearly the sum of each instanced decoding.

Table 40-15. Summary of Buffer Requirement

		MPEG-4 Encoder	MEPG-4 Decoder	H.264 Encoder	H.264 Decoder
QCIF	Frames Size	111 Kbytes	111 Kbytes	148 Kbytes	666 Kbytes
	Program Size	65 Kbytes			
	Working Buffer	64 Kbytes	64 Kbytes	120 Kbytes	256 Kbytes
	Total	240 Kbytes	240 Kbytes	333 Kbytes	987 Kbytes
CIF	Frames Size	444 Kbytes	444 Kbytes	592 Kbytes	2672 Kbytes
	Program Size	65 Kbytes			
	Working Buffer	64 Kbytes	64 Kbytes	120 Kbytes	256 Kbytes
	Total	573 Kbytes	573 Kbytes	777 Kbytes	2993 Kbytes
VGA	Frames Size	1350 Kbytes	1350 Kbytes	1800 Kbytes	3937.5 Kbytes
	Program Size	65 Kbytes			
	Working Buffer	64 Kbytes	64 Kbytes	120 Kbytes	256 Kbytes
	Total	1479 Kbytes	1479 Kbytes	1985 Kbytes	4258.5 Kbytes

Except the BIT processor program memory, other kinds of buffer has to be assigned on a per instance basis. Therefore the overall buffer size for a multi-party call application is nearly the sum of each instanced codec.

40.7.2 Application Using Cases

Video Codec module allows 4 channels to encode/decode simultaneously. That extends i.MX27 video processing application fields, and make it possible for multi-channel bitstream/image processing. For example, it can encode captured image in one channel, and decode other 3 channels bitstream at the same time, as shown in Figure 40-16. In Figure 40-16, gray rectangle represents the bitstream data. All codec channels image/bitstream can be encoded/decoded by different codec standard, like one image is encoded

to H264/AVC bitstream, and one channel H264/AVC bitstream, one channel MPEG4 bitstream, one channel H263 bitstream are decoded separately.

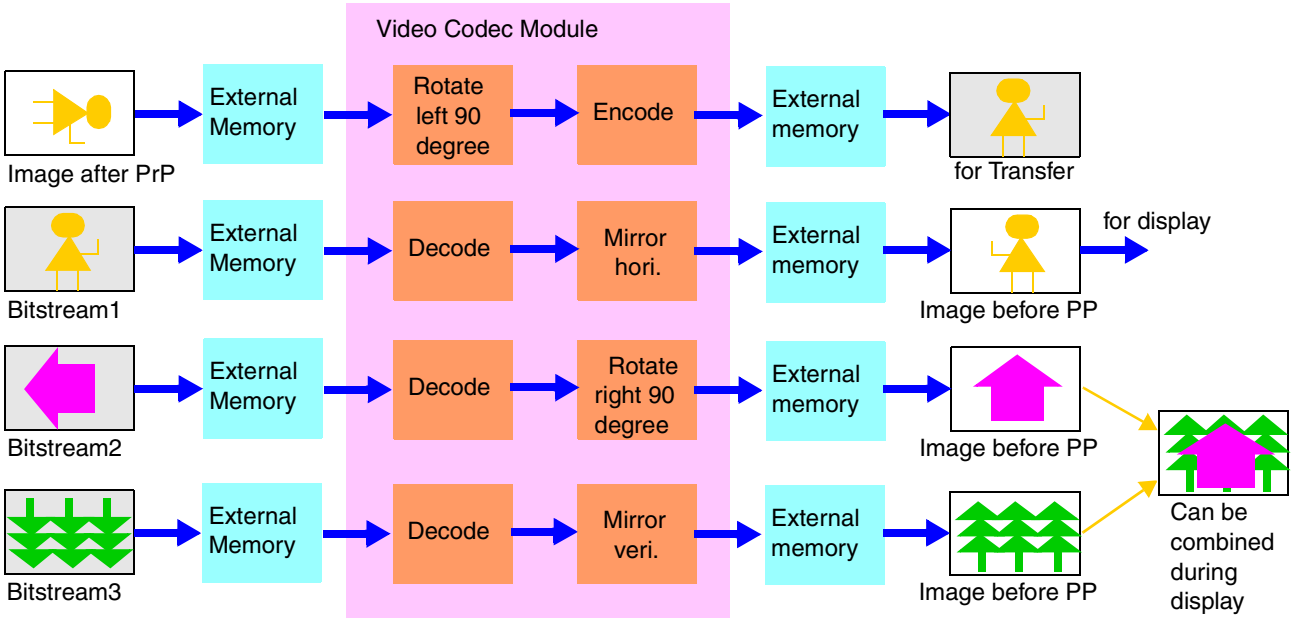


Figure 40-16. One of Application Using Case: One Channel Encoding and Three Channels Decoding

Chapter 41
enhanced Multimedia Accelerator Light (eMMA_lt)

41.1 Introduction

The *enhanced* MultiMedia Accelerator Light (eMMA_lt) consists of the video Pre-Processor (PrP) and Post-Processor (PP), which provide video acceleration and off-load the CPU from computation intensive tasks. The PrP and PP can be used for generic video pre- and post-processing, such as scaling, resizing, and color space conversions.

41.1.1 Features

- Pre-Processor:
  - — Data input:
    - – System memory
    - – Private DMA between CMOS Sensor Interface module and Pre-Processor
  - — Data input formats:
    - – Arbitrarily formatted RGB pixels (16 or 32 bits)
    - – YUV 4:2:2 (Pixel interleaved)
    - – YUV 4:2:0 (IYUV, YV12)
  - — Input image size: 32×32 to 2044×2044
  - — Image scaling:
    - – Programmable independent CH-1 and CH-2 resizer. Can program to be in cascade or parallel.
    - – Each resizer supports downscaling ratios from 1:1 to 8:1 in fractional steps.
  - — Channel-1 output data format
    - – Channel 1
    - – RGB 16 and 32 bpp
    - – YUV 4:2:2 (YUYV, YVYU, UYVY, VYUY)
  - — Channel-2 output data format
    - – YUV 4:2:2 (YUYV)
    - – YUV 4:4:4
    - – YUV 4:2:0 (IYUV, YV12)
    - – RGB data and YUV data format can be generated concurrently
  - — 32/64-bit AHB bus
- Post-Processor

- Input data:
  - From system memory
- Input format:
  - YUV 4:2:0 (IYUV, YV12)
- Image Size: 32 × 32 to 2044 × 2044
- Output format:
  - YUV 4:2:2 (YUYV)
  - RGB16 and RGB32 bpp
- Image Resize
  - Upscaling ratios ranging from 1:1 to 1:4 in fractional steps
  - Downscaling ratios ranging from 1:1 to 2:1 in fractional steps and a fixed 4:1
  - Ratios provide scaling between QCIF, CIF, QVGA (320 × 240, 240 × 320)

41.2 eMMA_lt Architecture

Figure 41-1 shows the block diagram of eMMA_lt.

The eMMA_lt consists of the Pre-Processor and Post-Processor modules. Each module has individual control and configuration registers which are accessed via the IP interface and are capable of bus mastering the AMBA bus to independently access system memory without any CPU intervention. This allows each module to be used independently of each other and enables the Pre-Processor and Post-Processor modules to provide acceleration features for other software codec implementations and image processing software. A 32 bit to 64 bit AHB gasket is used to convert AHB bus from 32-bit to 64-bit protocol. A bypass function is implemented to bypass this 64 bit gasket if it is not needed.

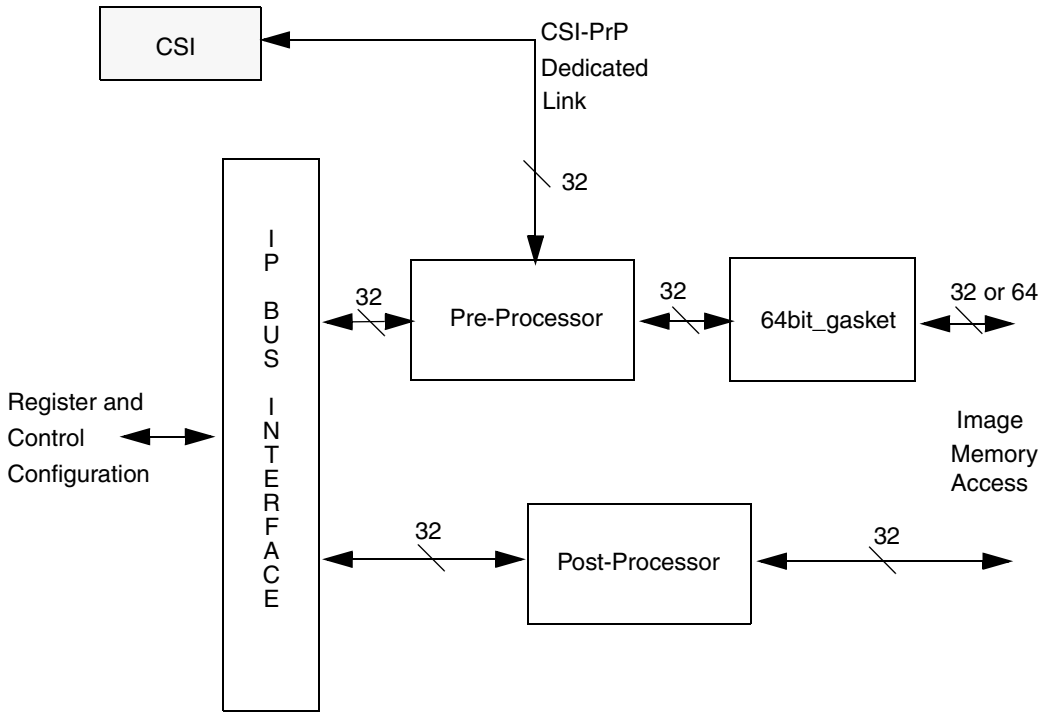


Figure 41-1. eMMA_lt Block Diagram

41.2.1 Pre-Processor (PrP)

The Pre-Processor (PrP) module accepts input from main memory via a 64-bit AHB port directly from EMI or from the dedicated link that connects it to camera input via the CMOS Sensor Interface (CSI) module. The PrP can be operated in two modes; single or continuous frame (loop) mode. In single frame mode, a single frame is processed either from memory or from the dedicated CSI-PrP link. In this mode, the PrP must be re-enabled each time a frame is to be processed. This mode is suitable for still image capture, processing and display and for very low frame rate operation. In continuous frame or loop mode, the PrP processes input frames from the dedicated CSI-PrP link continuously until it is disabled or an error occurs.

The PrP has two output channels (Channel-1 and Channel-2) and both channels store processed frames to main memory. The output from Channel-1 is dedicated for display purposes and the output from Channel-2 for input to a hardware encoder (MPEG-4 Encoder module) or a software encoder or image compressor. The PrP resizes input frames from memory of from the CSI and performs color space conversion.

41.2.2 Post-Processor (PP)

The Post-Processor module takes decoded frames from memory and performs additional processing to de-block, de-ring, resize, and color space convert the decoded frames for display. The decoded input can be either from the Decoder module or a software decoder module.

41.2.3 64-Bit Gasket

In order to connect PrP to a 64-bit AHB port of EMI, a 32-bit to 64-bit AHB protocol gasket is used to manage AHB signals between 32-bit master (PrP) and 64-bit AHB bus. In the event where 64 bit transfer is not required, a bypass function is implemented in the gasket to allow the PrP to ignore this 64 bit gasket and work as the original 32 bit AHB protocol. Figure 1-2 shows the basic structure of the eMMA-lt AHB 64-bit gasket with a bypass function. When the ahb64_sel signal is low the gasket is avoided.

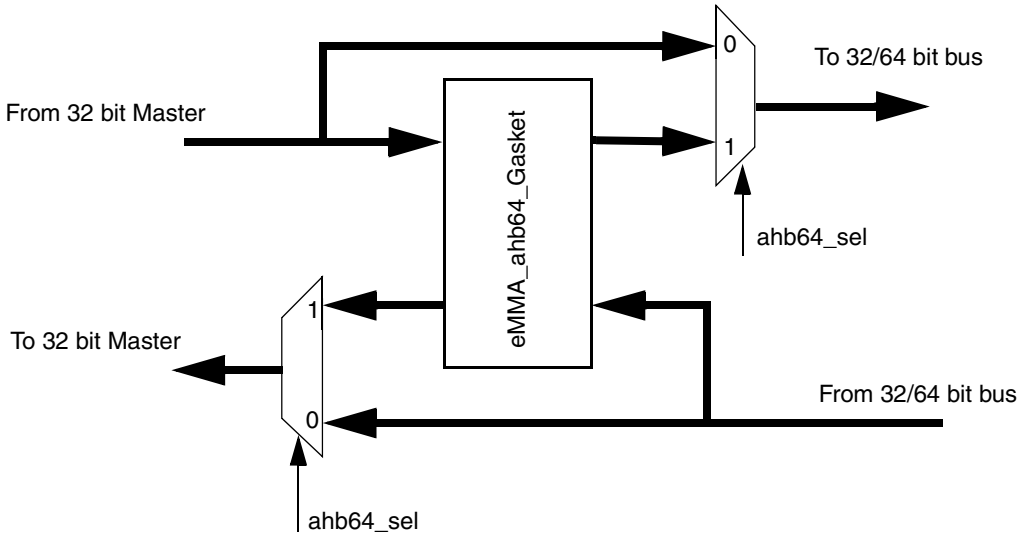


Figure 41-2. eMMA AHB 64-Bit Gasket with Bypass

41.3 Post-Processor (PP)

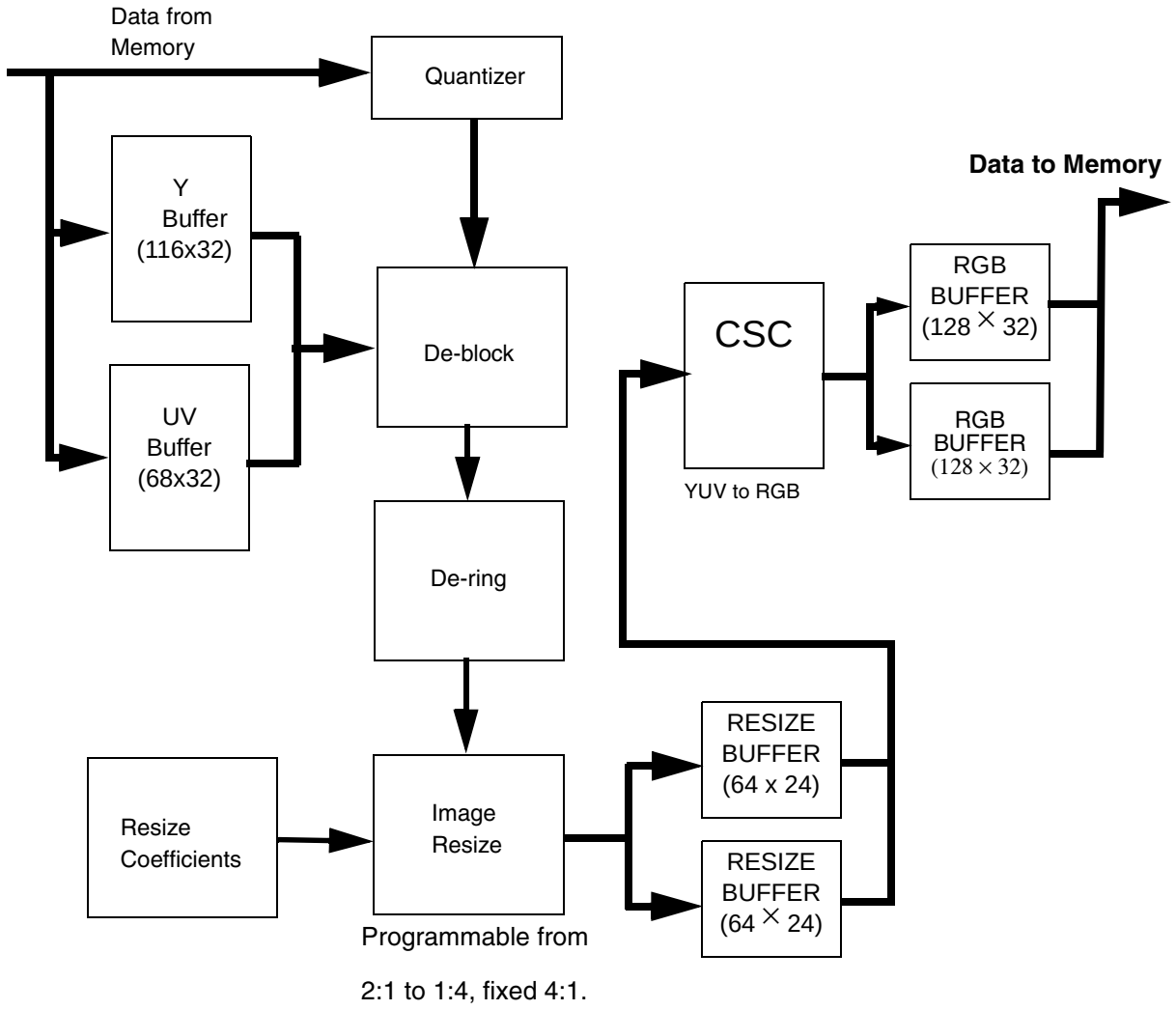


Figure 41-3. Post-Processor (PP) Block Diagram

The Post-Processor (PP) performs postprocessing functions after video decoding.

The key modules in the PP are:

De-block—Removes blocking artifacts while preserving natural edges in the images. Deblock processing is bypassed if not selected.

De-ring—Removes ringing artifacts from decoded images caused by the truncation of high spatial frequencies. Subjective tests have shown that performing both de-ringing and de-blocking improves slightly the visual quality than performing de-blocking alone. De-ring processing is bypassed if not selected.

Image Resize—Scales input image to a different size—for example, from QCIF (176 × 144) to QVGA (320 × 240) to match the size of the display or to scale from one aspect ratio to another ratio—for example, from 1:1 to 4:3. It supports programmable resize ratios from 2:1 to 1:4, and a fixed 4:1. Horizontal resizing and vertical resizing are independent and can be set to different resizing ratios.

Bilinear interpolation algorithm is implemented and used for both upscaling and downscaling—that is, two adjacent pixels are loaded and multiplied by respective weighting coefficients to produce an output pixel. The weighting coefficients for a particular resize ratio are calculated by software and preloaded into the resize coefficient table of the PP from which the resize block reads the coefficients to use.

For example, the output samples for 3:5 bilinear interpolation can be calculated as follows:

out[0] = in[0]

out[1] = 2/5 * in[0] + 3/5 * in[1]

out[2] = 4/5 * in[1] + 1/5 * in[2]

out[3] = 1/5 * in[1] + 4/5 * in[2]

out[4] = 3/5 * in[2] + 2/5 * in[3]

The output samples for the 5:3 bilinear interpolation can be calculated as follows:

out[0] = 2/3 * in[0] + 1/3 * in[1]

out[1] = 0/3 * in[1] + 3/3 * in[2]

out[2] = 1/3 * in[3] + 2/3 * in[4]

A programmable resize engine is implemented in hardware, which reads instructions from the resize coefficient table (Register eMMA PP RESIZE COEF TABLE).

An output pixel will be generated with the value (w1 * in1 + w2 * in2)/32 and the resize engine will then read in *n* new input pixels, where in1 and in2 are two adjacent pixels. If *n* is zero, then no new pixels are read and the in1 and in2 pixel values are reused.

Each instruction in the table is in the form of (w1, n,o) where each coefficient (w1) is represented with 5 bits and n with 2 bits and ‘o’ in 1-bit. w2 is calculated as 32-w1.

NOTE

Coefficient value of 31 (5’b11111) is treated as 32 (6’b100000), consequently, coefficient values of 1 and 31 are not possible.

The Table 41-1 and Table 41-2 show some example resize coefficients.

Table 41-1. Resize Coefficients for 3:5

w1	w2	n	Right Coefficient	Left Coefficient	in1	in2	Out
1	0	0	5’b11111 (32)	5’b00000 (0)	in[0]	–	in[0]
2/5	3/5	1	5’b01101 (13)	5’b10011 (19)	in[0]	in[1]	13/32 * in[0] + 19/32 * in[1]
4/5	1/5	1	5’b11010 (26)	5’b00110 (6)	in[1]	in[2]	26/32 * in[1] + 6/32 * in[2]

Table 41-1. Resize Coefficients for 3:5 (continued)

w1	w2	n	Right Coefficient	Left Coefficient	in1	in2	Out
1/5	4/5	0	5'b00110 (6)	5'b11010 (26)	in[1]	in[2]	6/32 * in[1] + 26/32 * in[2]
3/5	2/5	1	5'b10011 (19)	5'b01101 (13)	in[2]	in[3]	19/32 * in[2] + 13/32 * in[3]

Table 41-2. Resize Coefficients for 5:3

w1	w2	n	Left Coefficient	Right Coefficient	in1	in2	Out
2/3	1/3	1	5'b10101 (21)	5'b01011 (11)	in[0]	in[1]	21/32 * in[0] + 11/32 * in[1]
0	1	2	5'b00000 (0)	5'b11111 (32)	in[1]	in[2]	0 * in[0] + 1 * in[1]
1/3	2/3	2	5'b01011 (11)	5'b10101 (21)	in[3]	in[4]	11/32 * in[0] + 21/32 * in[1]

Table 41-3. Resize Coefficients for 4:1

w1	w2	n	Left Coefficient	Right Coefficient	in1	in2	Out
1/2	1/2	1	5'b10000 (16)	5'b10000 (16)	in[0]	in[1]	1/2 * in[0]+ 1/2 * in[1]
0	0	1	5'b00000 (0)	5'b00000 (0)	in[1]	in[2]	—
0	0	1	5'b00000 (0)	5'b00000 (0)	in[2]	in[3]	—
0	0	1	5'b00000 (0)	5'b00000 (0)	in[3]	in[4]	—

41.3.1 Color Space Conversion (CSC)

The color space conversion block converts input images from YUV to RGB color space needed for display. The CSC block is fully programmable.

Equation used for YCbCr to RGB calculation:

R = C0*(Y - X0) + C1*(Cr-128)

G = C0*(Y - X0) - C2*(Cb-128) - C3*(Cr-128)

B = C0*(Y - X0) + C4*(Cb-128)

Equation used for YUV to RGB calculation:

R = C0*(Y - X0) + C1*(U-128)

G = C0*(Y - X0) - C2*(U-128) - C3*(V-128)

B = C0*(Y - X0) + C4*(U-128)

X0, C0, C1, C2, C3 and C4 are coefficients that can be programmed through registers PP_CSC_COEF_0123 and PP_CSC_COEF_4X.

C0[7:0] Range from 0 to 1.9921875 in steps of (1/128).

C1[7:0] Range from 0 to 1.9921875 in steps of (1/128).

C2[7:0] Range from 0 to 1.9921875 in steps of (1/128).

C3[7:0] Range from 0 to 1.9921875 in steps of (1/128).

C4[8:0] Range from 0 to 3.9921875 in steps of (1/128).

X0 1 - 16, 0 - 0.

All the 10 formats defined by MPEG-4, including YUV to RGB, YCbCr to RGB and ITU-R BT 709 to RGB are supported through this programmable CSC block. The MPEG-4 standard allows for a number of conversion scenarios. The particular type of color space used by an MPEG-4 encoder is signaled in the bit stream syntax and is determined by two fields, matrix_coefficients and video_range. 5 color space conversion equations (matrix coefficients) and 2 video ranges for each equation (matrix) are defined in MPEG-4. This gives a total of 10 (5x2) color space conversion possibilities.

The 5 matrix coefficients can be categorized into two sets. The matrices represented by matrix_coefficients field values of 4, 5, and 6 are similar and can be grouped together into one set (Set A). The two remaining matrices, represented by matrix_coefficients field values of 1 and 7, are similar and hence can be grouped together into a second set (Set B). Set A and Set B represent CSC matrices in accordance with Recommendation ITU-R BT.470 and Recommendation ITU-R BT.709, respectively. For each CSC matrix, there are two possible video ranges, indicated by the video_range field which is set to either 1 or 0. Therefore there are now two video ranges x 2 sets = 4 CSC scenarios and each of these are summarized in Table 41-4.

Table 41-4. YUV to RGB CSC Equations

Eqn	Matrix Co-Efficient	Video Range	Input to CSC and Notation	Matrix	Register Values
A1	4,5 or 6 (Set A)	1	YUV Y ranges from 0-255 U ranges from 0-255 V ranges from 0-255	R = Y + 1.4026 * (V-128) G = Y – 0.3444 * (U-128) – 0.7144 * (V-128) B = Y + 1.7730 * (U-128)	C0 = 8’b1000 0000 C1 = 8’b1011 0011 C2 = 8’b0010 1100 C3 = 8’b0101 1011 C4 = 9’b0 1110 0010 X0 = 1’b0
A0	4,5 or 6 (Set A)	0	YCrCb Y ranges from 16-235 Cr ranges from 16-240 Cb ranges from 16-240	R = 1.164*(Y – 16) + 1.596*(Cr-128) G = 1.164*(Y – 16) – 0.813*(Cr-128) – 0.391*(Cb-128) B = 1.164*(Y – 16) + 2.018*(Cb-128)	C0 = 8’b1001 0100 C1 = 8’b1100 1100 C2 = 8’b0011 0010 C3 = 8’b0110 1000 C4 = 9’b1 0000 0010 X0 = 1’b1

Table 41-4. YUV to RGB CSC Equations (continued)

Eqn	Matrix Co-Efficient	Video Range	Input to CSC and Notation	Matrix	Register Values
B1	1 or 7 (Set B)	1	Y'U'V' Y' ranges from 0-255 U' ranges from 0-255 V' ranges from 0-255	R = Y' + 1.5749 * (V'-128) G = Y' – 0.1875 * (U'-128) – 0.4682 * (V'-128) B = Y' + 1.8554 * (U'-128)	C0 = 8'b1000 0000 C1 = 8'b1100 1001 C2 = 8'b0001 1000 C3 = 8'b0011 1100 C4 = 9'b0 1110 1101 X0 = 1'b0
B0	1 or 7 (Set B)	0	Y'Cr'Cb' Y' ranges from 16-235 Cr' ranges from 16-240 Cb' ranges from 16-240	R = 1.164(Y'-16) + 1.793 * (Cr'-128) G = 1.164(Y'-16) – 0.533 * (Cr'-128) – 0.213 * (Cb'-128) B = 1.164(Y'-16) + 2.112 * (Cb'-128)	C0 = 8'b1001 0100 C1 = 8'b1110 0110 C2 = 8'b0001 1011 C3 = 8'b0100 0100 C4 = 9'b1 0000 1110 X0 = 1'b1

The correct matrix has to be selected based on the video_range and matrix_coefficient for color space conversion.

41.3.2 Input Interface

The PP reads IYUV or YV12 data from external memory. Quantization Parameter (QP) data is required if Deblock and/or De-ring operations are selected. Figure 41-4 shows an example layout for QCIF frames to be processed by the PP.

The input Y, U, V, and QP data are expected in 4 sections of memory. The first data in every row starts with a new word. When the row size is not a multiple of 4 bytes, the last few pixels in a row will not occupy a full word and the extra bytes, if any, in the last word are ignored.

The PP expects one QP byte per macroblock (MB). Only the lower 5 bits of a QP byte are used and the 3 most significant bits must be set to 0. The first QP in every row starts with a new word and four QP bytes from 4 adjacent MBs in a row are packed into one word. When the number of MBs in a row is not a multiple of 4, then the extra QP bytes in the last word of a QP row are ignored.

Data is stored in the memory in the order of natural scan lines. For Y, U, and V data, it is permitted that the distance between the start of two neighboring lines (line stride) is greater than the number of pixels in a line—that is, there could be fixed number of unused words between the end and the beginning of every two neighboring lines. However, unused words are not permitted for QP data.

Each row in Figure 41-4 represents a word in memory. Y(j,i) denotes Y data of pixel row j and column i. Layout of U and V data is similar to that of Y data. The figure shows there can be unused space in memory between rows. This parameter is controlled by the Input Line Stride parameter and applies only to Y, U, and V data in the frame. The start of Y, U, and V data can be anywhere in addressable memory.

In a QCIF image, there are 11 × 9 = 99 MBs, representing all the QPs for a QCIF image. However, due to the packing constraints specified above, the 11 QPs of a row are packed into 3 words with one unused byte in the last word. This is repeated for each row and there can be no optional space between QP rows.

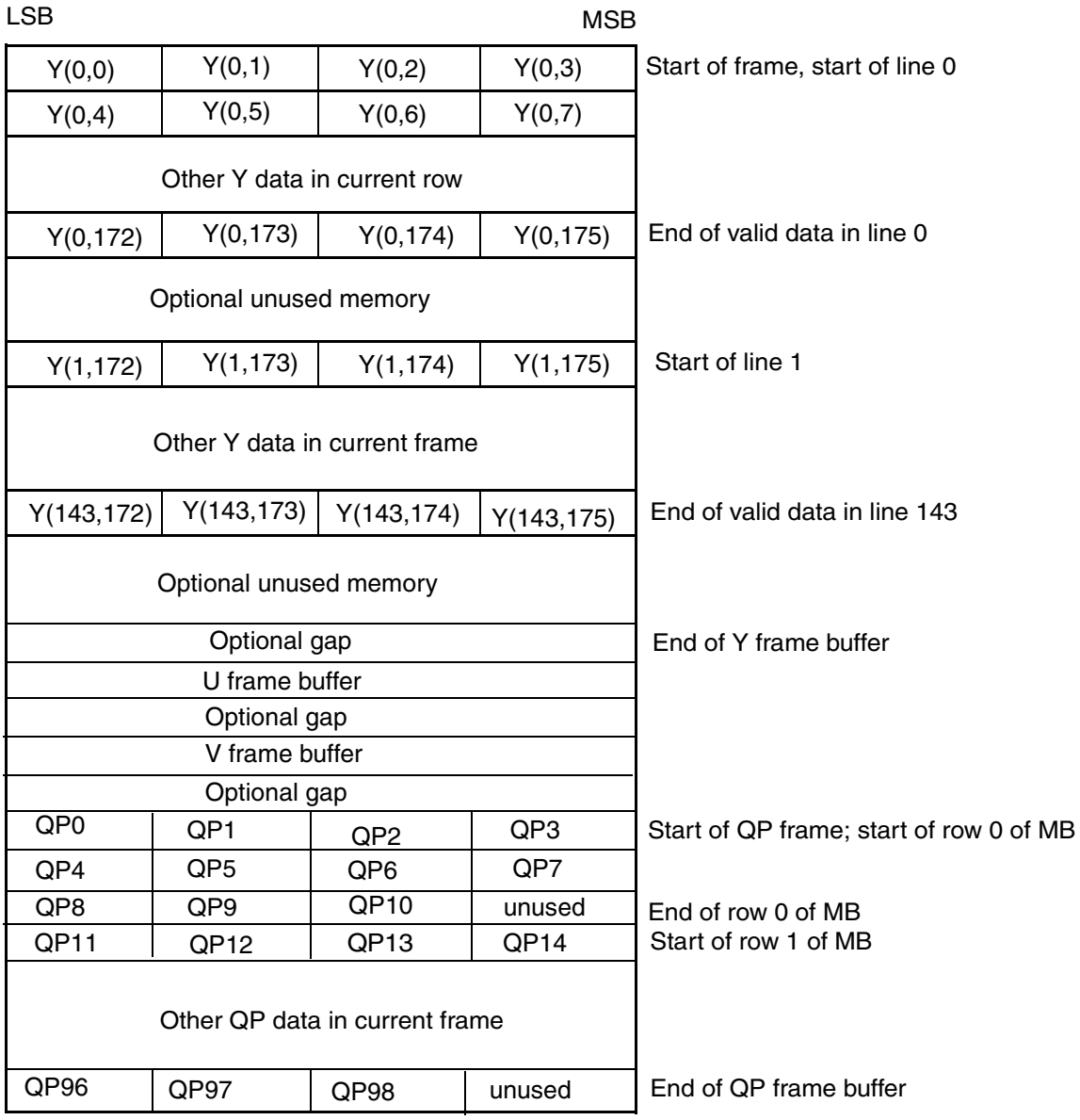


Figure 41-4. PP Input Data Layout (QCIF)

41.3.3 Output Interface

The output of the PP is selectable between RGB and YUV 4:2:2 (YUYV).

RGB data is internally represented with 24 bits resolution (8 bits per color component) and color bits are truncated according to the programmed color widths. This truncation is done at the last stage of color space

conversion by discarding the least significant bits. For 8 bpp output only 1:1 resize ratio is supported. Table 41-5 shows some examples for 16 bpp and unpacked 24 bpp settings.

Table 41-5. RGB Color Width and Offsets

Pixel Format	bpp	RGB Width	RGB Offset
Packed 16-bit RGB565	16	RedWidth = 5 GreenWidth = 6 BlueWidth = 5	RedOffset = 11 GreenOffset = 5 BlueOffset = 0
Unpacked 32-bit RGB888	32	RedWidth = 8 GreenWidth = 8 BlueWidth = 8	RedOffset = 16 GreenOffset = 8 BlueOffset = 0

41.3.4 Data Flow

The Post-Processing block reads YUV 4:2:0 data from external memory and writes processed RGB or YUYV 4:2:2 data into external memory. A typical use case of the PP in i.MX27 device is as follows:

1. Software or hardware decoder decodes one frame
2. PP is programmed with frame buffer address and other ancillary information
3. Software enables PP
4. For every MB read from memory, PP performs Deblock, Dering, Resize, and CSC and writes to output buffer for display
5. When all MBs of the current frame are processed, PP signals frame completion to the core
6. When PP completes frame processing, it sets the frame completion status and interrupts the CPU.

41.3.5 Relationship of Register Fields Related to the Input Frame

Figure 41-5 shows how the Input Line Stride affects the area of frame that is processed by the PP. There are two rectangular areas in the diagram. The inner rectangle shows the frame area to be processed and the outer rectangle indicates the actual memory allocated for the total frame. PP_Y_SOURCE, PP_CR_SOURCE, and PP_CB_SOURCE are pointers to the start addresses of frame data. The Input Line Stride, Width, and Height parameters are automatically divided by 2 when processing the Cr and Cb frame components.

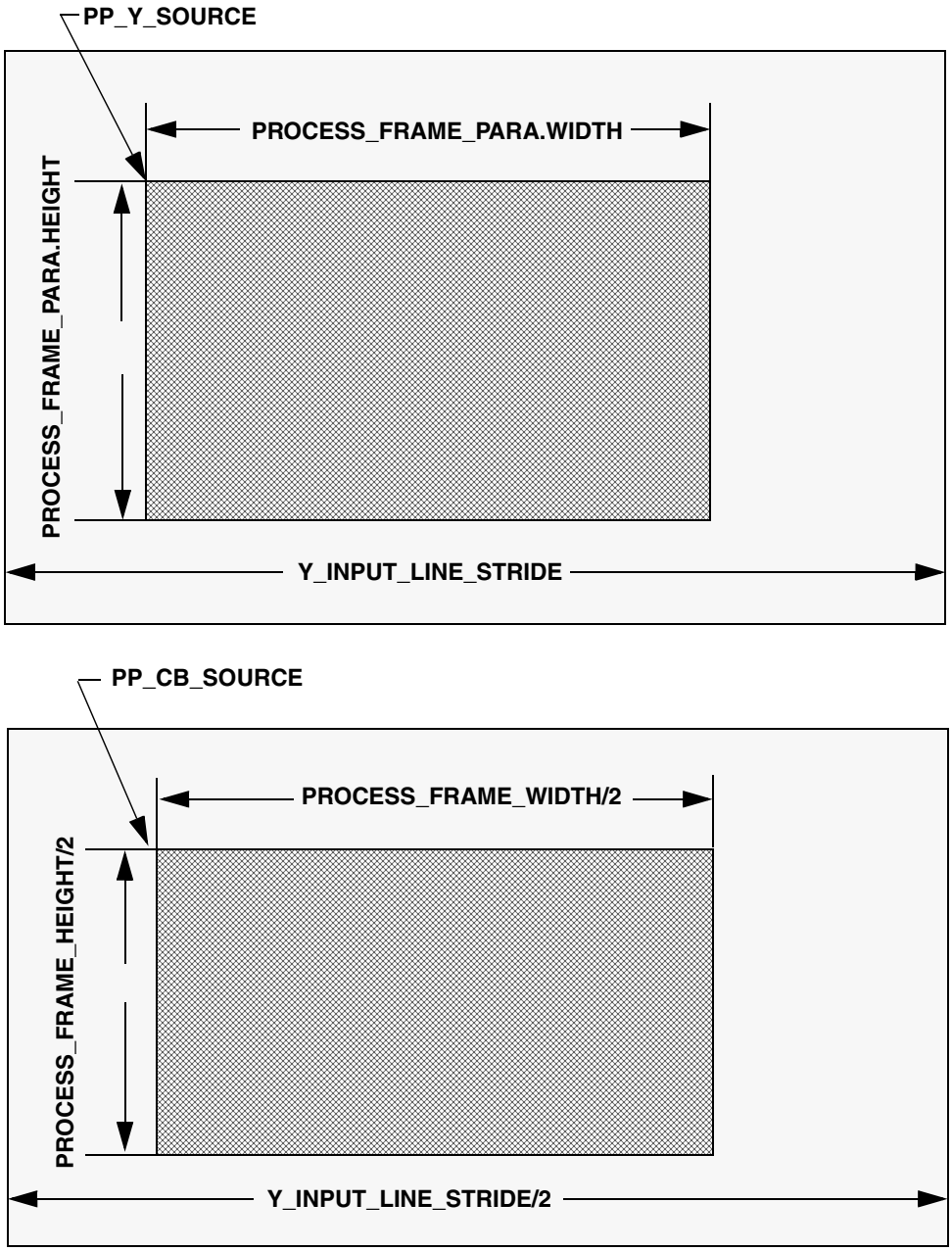


Figure 41-5. Input Line Stride

41.3.6 Relationship of Register Fields Related to Output Frame

Figure 41-6 shows the effect of the Output Line Stride parameter on the output frame. The Output Line Stride can be used to select a smaller area of the processed and resized frame. The figure shows two rectangular areas. The outer rectangle shows the memory allocated for display. The inner rectangle shows the size of the final output image. The output image size is defined by the Output Line Stride, IMAGE_WIDTH, and IMAGE_HEIGHT parameters.

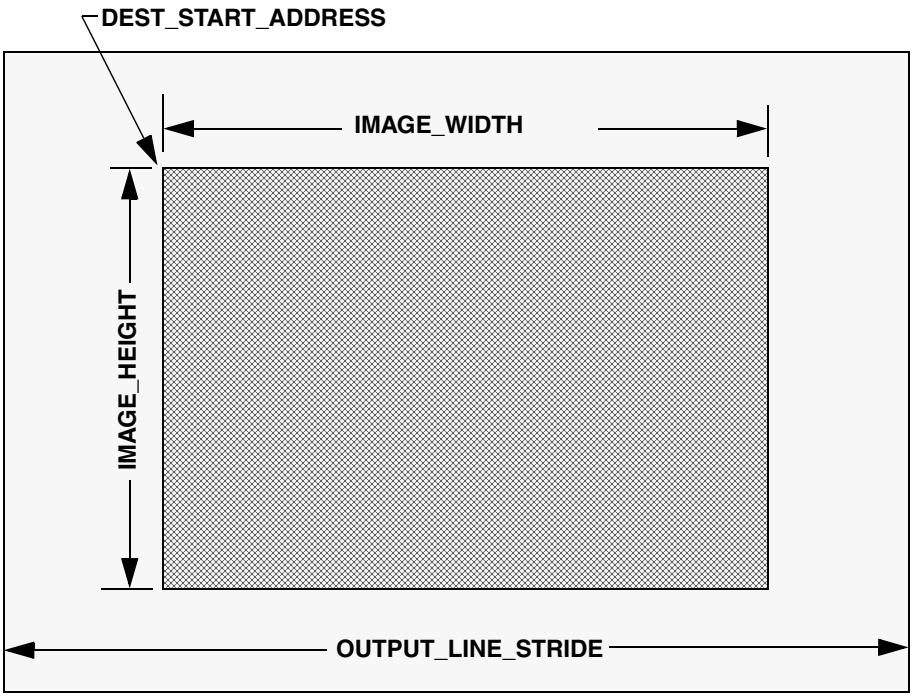


Figure 41-6. Output Line Stride

NOTE

Output Line Stride is specified in bytes while IMAGE_WIDTH and IMAGE_HEIGHT are specified in pixels.

41.4 Post Processor (PP) Programming Model

Only 32-bit accesses (read/write) is supported. All reserved bits should always be written with 0 and all registers are R/W unless specified. Table 41-6 shows the memory map.

Table 41-6. PP Register Memory Map

Address	Description	Access	Reset Value	Section/Page
0x1002_6000 (PP_CNTL)	PP Control Register	R/W	0x0000_0876	41.4.1/41-14
0x1002_6004 (PP_INTRCNTL)	PP Interrupt Control	R/W	0x0000_0000	41.4.2/41-15
0x1002_6008 (PP_INTRSTATUS)	PP Interrupt Status	R/W	0x0000_0–0–	41.4.3/41-16
0x1002_600C (PP_SOURCE_Y_PTR)	PP Source Y Frame data Pointer	R/W	0x0000_0000	41.4.4/41-17
0x1002_6010 (PP_SOURCE_CB_PTR)	PP Source CB Frame data Pointer	R/W	0x0000_0000	41.4.5/41-18
0x1002_6014 (PP_SOURCE_CR_PTR)	PP Source CR Frame data Pointer	R/W	0x0000_0000	41.4.6/41-19
0x1002_6018 (PP_DEST_RGB_PTR)	PP Destination RGB Frame start address	R/W	0x0000_0000	41.4.7/41-19
0x1002_601C (PP_QUANTIZER_PTR)	PP Quantizer start address	R/W	0x0000_0000	41.4.8/41-20
0x1002_6020 (PP_PROCESS_PARA)	PP Process frame parameter, width and height	R/W	0x0000_0000	41.4.9/41-21

Table 41-6. PP Register Memory Map (continued)

Address	Description	Access	Reset Value	Section/Page
0x1002_6024 (PP_FRAME_WIDTH)	PP Source Frame width	R/W	0x0000_0000	41.4.10/41-21
0x1002_6028 (PP_DISPLAY_WIDTH)	PP Destination Display width	R/W	0x0000_0000	41.4.11/41-22
0x1002_602C (PP_IMAGE_SIZE)	PP Destination Image Size	R/W	0x0000_0000	41.4.12/41-23
0x1002_6030 (PP_DEST_FRAME_FORMAT_CNTL)	PP Destination Frame Format Control	R/W	0x0000_0000	41.4.13/41-24
0x1002_6034 (PP_RESIZE_INDEX)	PP Resize Table Index	R/W	0x0000_0000	41.4.14/41-25
0x1002_6038 (PP_CSC_COEF_123)	PP CSC coefficients	R/W	0x0000_0000	41.4.15/41-26
0x1002_603C (PP_CSC_COEF_4)	PP CSC coefficients	R/W	0x0000_0000	41.4.16/41-27
0x1002_6000–0x1002_607C (PP_RESIZE_COEF_TBL)	PP Resize Coefficient Table	W	0x0000_0000	41.4.17/41-28

41.4.1 PP Control Register

Figure 41-7 shows the register; Table 41-7 provides its field descriptions.

0x1002_6000 (PP_CNTL) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	BSDI	CSC_OUT		MB_MODE	SWRST	0	CSC TABLE SEL		CSCEN	0	DERINGEN	DEBLOCKEN	PP_EN
W																
Reset	0	0	0	0	1	0	0	0	0	1	1	1	0	1	1	0

Figure 41-7. PP Control Register

Table 41-7. PP Control Register Field Descriptions

Name	Description
31–13	Reserved—These bits are reserved and should read 0.
12 BSDI	Byte Swap Input Data. The input data word from memory is byte swapped (32-bit little endian to big endian or vice versa) before use. 0 Swap disabled 1 Swap enabled

Table 41-7. PP Control Register Field Descriptions (continued)

Name	Description
11–10 CSC_OUT	CSC Output. Sets RGB output resolution. 00 32-bit (unpacked RGB888) 01 Reserved 10 16-bit 11 32-bit (unpacked RGB888)
9	Reserved—This bit should always read 0
8 SWRST	Software Reset. Resets entire module, all registers return to their reset default values. 0 No reset 1 Reset
7 Reserved	Reserved. This bit is reserved and should read 0.
6–5 CSC_TABLE_SEL	CSC Table Select. Selects one of the 4 CSC matrices. Refer to Table 41-4 for more information. 00 A1 01 A0 10 B1 11 B0
4 CSCEN	CSC Enable. Enables CSC to output RGB data else YUV 4:2:2 data when disabled. YUV 4:2:2 output is in YUYV interleaved format. 0 YUV 4:2:2 1 RGB
3	Reserved. This bit is reserved and should read 0.
2 DERINGEN	De-ring Enable. Enable or disable Dering operation. 0 No Dering 1 Enable Dering
1 DEBLOCKEN	De-block Enable. Enable or disable Deblock operation. 0 No Deblock 1 Enable Deblock
0 PP_EN	PP Enable. Start frame processing. Bit has no effect if the LOCK_BIT was not previously read successfully with IDLE status. Once enabled, this bit cannot be reset unless one of the following has occurred; Frame is completely processed or SWRST is set or Data abort error. 0 Not enabled 1 Enabled (self-clearing)

41.4.2 PP Interrupt Control Register

Figure 41-8 shows the register; Table 41-8 provides its field descriptions.

0x1002_6004 (PP_INTRCNTL) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0		0	
W														ERR INTR _EN		FRA ME COM P INTR EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-8. PP Interrupt Control Register

Table 41-8. PP Interrupt Control Register Description

Name	Description
31–3	Reserved. This bit is reserved and should read 0.
2 ERR_INTR_EN	Error Interrupt Enable. If set, enables interrupt on error. 0 interrupt disabled 1 interrupt enabled
1	Reserved. This bit is reserved and should read 0
0 FRAME_COMP_INTR_EN	Frame Complete Interrupt Enable. If set and in Frame mode (MB_MODE=0), enables interrupt on completion of frame processing. 0 Interrupt disabled 1 Interrupt enabled

41.4.3 PP Interrupt Status Register

Figure 41-9 shows the register; Table 41-9 provides its field descriptions.

0x1002_6008 (PP_INTRSTATUS) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0		0	FRAME COMP INTR EN
W														ERR INTR _EN		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	w1c	0	w1c

Figure 41-9. PP Interrupt Status Register

Table 41-9. PP Interrupt Status Register Field Descriptions

Name	Description
31–3	Reserved. These bits are reserved and should be set to 0.
2 ERR_INTR	Error Interrupt Status. If set an error has occurred. The PP has to be reset (SWRST = 1) before further operations can be initiated.
1	Reserved.
0 FRAME_COMP_INTR	Frame Complete Interrupt Status. If set, a frame has been processed.

41.4.4 PP Source Y Address Register

Figure 41-10 shows the register; Table 41-10 provides its field descriptions.

0x1002_600C (PP_SOURCE_Y_PTR) Access: User Read/Write

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

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

Figure 41-10. PP Source Y Address Register

Table 41-10. PP Source Y Address Register Field Descriptions

Name	Description
31–0 PP_Y_SOURCE	PP_SOURCE_Y_PTR. 32-bit frame start address of Y data (Luminance). Bits 1–0 are always set to 0 (word aligned)

41.4.5 PP Source Cb Address Register

Figure 41-11 shows the register; Table 41-11 provides its field descriptions.

0x1002_6010 (PP_SOURCE_CB_PTR) Access: User Read/Write

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

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

Figure 41-11. PP Source Cb Address Register

Table 41-11. PP Source Cb Address Register Field Descriptions

Name	Description
31–0 PP_CB_SOURCE	PP_SOURCE_CB_PTR. 32-bit frame start address of Cb data (U or Chrominance). Bit 1–0 are always set to 0 (word aligned)

41.4.6 PP Source Cr Address Register

Figure 41-12 shows the register; Table 41-12 provides its field descriptions.

0x1002_6014 (PP_SOURCE_CR_PTR) Access: User Read/Write

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

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

Figure 41-12. PP Source Cr Address Register

Table 41-12. PP Source Cr Address Register Field Descriptions

Name	Description
31–0 PP_CR_SOURCE	PP_SOURCE_CR_PTR. 32-bit frame start address of Cr data (V or Chrominance). Bits 1–0 are always set to 0 (word aligned)

41.4.7 PP Destination RGB Frame Start Address Register

Figure 41-13 shows the register; Table 41-13 provides its field descriptions.

0x1002_6018 (PP_DEST_RGB_PTR) Access: User Read/Write

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

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

Figure 41-13. PP Destination RGB Frame Start Address Register

Table 41-13. PP Destination RGB Frame Start Address Register Description

Name	Description
31–0 RGB_START_ADDR	RGB Start Address. Sets the destination frame start address. If CSCEN = 0, then these bits point to the start of YUV 4:2:2 (YUYV interleaved) data, else it points to RGB data. Bit 1–0 are always set to 0 (word aligned).

41.4.8 PP Quantizer Start Address Register

Figure 41-14 shows the register; Table 41-14 provides its field descriptions.

0x1002_601C (PP_QUANTIZER_PTR) Access: User Read/Write

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

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

Figure 41-14. PP Quantizer Start Address Register

Table 41-14. PP Quantizer Start Address Register Description

Name	Description
31–0 QUANTIZER_PTR	QUANTIZER_PTR. Sets the start address of the Quantization Parameter in memory. This register is ignored if Deblock and De-ring are both disabled. Bits 1–0 are always set to 0 (word aligned)

41.4.9 PP Process Frame Parameter Register

Figure 41-15 shows the register; Table 41-15 provides its field descriptions.

0x1002_6020 (PP_PROCESS_PARA) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	PROCESS_FRAME_WIDTH[9:0]									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PROCESS_FRAME_HEIGHT[9:0]									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-15. PP Process Parameter Register

Table 41-15. PP Process Parameter Register Description

Name	Description
31–26	Reserved. These bits are reserved and should read 0.
25–16 PROCESS_FRAME_WIDTH	Process Frame Width. Sets the input window width to be processed based on Y frame pixel count. PROCESS_FRAME_WIDTH/2 is used for Cb and Cr window width. PROCESS_FRAME_WIDTH must always be less than or equal to INPUT_LINE_STRIDE. This value should be a multiple of 8 pixels.
15–10	Reserved. These bits are reserved and should read 0.
9–0 PROCESS_FRAME_HEIGHT	Process Frame Height. Sets the input window height to be processed based on Y frame line count. PROCESS_FRAME_HEIGHT/2 is used for Cb and Cr window height. This value should be a multiple of 8 lines.

41.4.10 PP Source Frame Width Register

Figure 41-16 shows the register; Table 41-16 provides its field descriptions.

0x1002_6024 (PP_FRAME_WIDTH) Access: User Read/Write

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

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

Figure 41-16. PP Source Frame Width Register

Table 41-16. PP Source Frame Width Register Field Descriptions

Name	Description
31–24	Reserved. These bits are reserved and should read 0.
23–16 QUANTIZER_FRAME_WIDTH	Quantizer Frame Width. These bits set the number of bytes used to represent quantizers from all the MB in a row. QP data is packed into words with possibly unused bytes in the last word when the number of MB in a row in not a multiple of 4. In such cases, QUANTIZER_FRAME_WIDTH is rounded up to the nearest 4-byte multiple (word) that represents all the QP data for one row of MBs. For example, if there are 7 MBs in a row, then 2 words are required to represent the 7 QP data bytes with one unoccupied byte. The QUANTIZER_FRAME_WIDTH in this example is set as 8 (7 bytes rounded up to the nearest multiple of 4). This value should be a multiple of 4 bytes.
15–12	Reserved. These bits are reserved and should read 0.
11–0 Y_INPUT_LINE_STRIDE	Y Input Line Stride. These bits set the number of pixels between adjacent rows of pixels for Y input data. Y_INPUT_LINE_STRIDE/2 is used for Cb and Cr input data. This value must be a multiple of 8 pixels.

41.4.11 PP Destination Display Width Register

Figure 41-17 shows the register; Table 41-17 provides its field descriptions.

0x1002_6028 (PP_DISPLAY_WIDTH) Access: User Read/Write

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

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

Figure 41-17. PP Destination Display Width Register

Table 41-17. PP Destination Display Width Register Description

Name	Description
31–13	Reserved. These bits are reserved and should read 0.
12–0 OUTPUT_LINE_STRIDE	Output Line Stride. These bits set the distance in bytes between the start addresses of adjacent lines in the output frame. If the stride is equal to the OUT_IMAGE_WIDTH, then the stride should be calculated as: OUT_IMAGE_WIDTH * Bytes Per Pixel. This value should be a multiple of 4 bytes.

41.4.12 PP Destination Image Size Register

Figure 41-18 shows the register; Table 41-18 provides its field descriptions.

0x1002_602C (PP_IMAGE_SIZE) Access: User Read/Write

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

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

Figure 41-18. PP Destination Image Size Register

Table 41-18. PP Destination Image Size Register Description

Name	Description
31–28	Reserved. These bits are reserved and should read 0.
27–16 OUT_IMAGE_WIDTH	Out Image Width. These bits set the width of the output in pixels (not bytes). This value must always be a multiple of 2. OUT_IMAGE_WIDTH[0] is read-only and always 0.
15–12	Reserved. These bits are reserved and should read 0.
11–10 OUT_IMAGE_HEIGHT	Out Image Height. These bits set the number of lines in the output. This value must always be a multiple of 2. OUT_IMAGE_HEIGHT[0] is read-only and always 0.

41.4.13 PP Destination Frame Format Control Register

Figure 41-19 shows the register; Table 41-19 provides its field descriptions.

0x1002_6030 (PP_DEST_FRAME_FORMAT_CNTL) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RED_WIDTH				GREEN_WIDTH				BLUE_WIDTH			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-19. PP Destination Frame Format Control Register

NOTE

This register is used only when the output is in RGB format.

Table 41-19. PP Destination Frame Format Control Register Description

Name	Description
31	Reserved. These bits are reserved and should read 0.
30–26 RED_OFFSET	Red Offset. Specifies the bit offset of the Red color or Luminance component in the output pixel The offset is specified with respect to Bit 0.
25–21 GREEN_OFFSET	Green Offset. Specifies the bit offset of the Green color or Chrominance (U) component in the output pixel. The offset is specified with respect to Bit 0.
20–16 BLUE_OFFSET	Blue Offset. Specifies the bit offset of the Blue color or Chrominance (V) component in the output pixel. The offset is specified with respect to Bit 0.
15–12	Reserved. These bits are reserved and should read 0.

Table 41-19. PP Destination Frame Format Control Register Description (continued)

Name	Description
11–8 RED_WIDTH	Red Width. Specifies the number of bits in the Red color component in the output pixel. The width of the Luminance component is fixed at 8 bits, always. Allowed values are 0 to 8. Any value greater than 8 is fixed to 8 internally.
7–4 GREEN_WIDTH	Green Width. Specifies the number of bits in the Green color component in the output pixel. The width of the Chrominance (Cb or U) component is fixed at 8 bits, always. Allowed values are 0 to 8. Any value greater than 8 is fixed to 8 internally.
3–0 BLUE_WIDTH	Blue Width. Specifies the number of bits in the Blue color component in the output pixel.The width of the Chrominance (Cr or V) component is fixed at 8 bits, always. Allowed values are 0 to 8. Any value greater than 8 is fixed to 8 internally.

The following table shows some example configurations for the YUV 4:2:2 combinations.

Table 41-20. YUV 4:2:2 Configuration Settings

YUV Format	Offsets	Width	Settings
YUYV	RED_OFFSET=24 GREEN_OFFSET=16 BLUE_OFFSET=0	RED_WIDTH=8 GREEN_WIDTH=8 BLUE_WIDTH=8	0x6200_0888
YVYU	RED_OFFSET=24 GREEN_OFFSET=0 BLUE_OFFSET=16		0x6010_0888
UYVY	RED_OFFSET=16 GREEN_OFFSET=24 BLUE_OFFSET=8		0x4308_0888
VYUY	RED_OFFSET=16 GREEN_OFFSET=8 BLUE_OFFSET=24		0x4118_0888

41.4.14 PP Resize Table Index Register

This register sets the start and end indices of horizontal and vertical resize tables. The two resize tables share a memory of 40 locations. The minimum start index is 0 and the maximum end index is 39. If the horizontal and vertical resize ratios are the same, then one resize table can be used and the horizontal and vertical start and end indices can be set to the same value. However, if the resize ratios are different, the horizontal and vertical resize tables need to be programmed differently. Either the horizontal resize table or the vertical resize table can start from address 0.

Figure 41-20 shows the register; Table 41-21 provides its field descriptions.

0x1002_6034 (PP_RESIZE_INDEX) Access: User Read/Write

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

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

Figure 41-20. PP Resize Table Index Register

Table 41-21. PP Resize Table Index Register Field Descriptions

Name	Description Settings
31–30	Reserved. These bits are reserved and should read 0.
29–24 HORI_TBL_START_INDEX	Horizontal Table Start Index. Start index of horizontal resize table. Valid values: 0–39
23–22	Reserved. These bits are reserved and should read 0.
21–16 HORI_TBL_END_INDEX	Horizontal Table End Index. End index of horizontal resize table. Valid values: 0–39
15–14	Reserved. These bits are reserved and should read 0.
13–8 VERT_TBL_START_INDEX	Vertical Table Start Index. Start index of vertical resize table. Valid values: 0–39
7–6	Reserved. These bits are reserved and should read 0.
5–0 VERT_TBL_END_INDEX	Vertical Table End Index. End index of vertical resize table. Valid values: 0–39

41.4.15 PP CSC COEF 123 Register

Figure 41-21 shows the register; Table 41-22 provides its field descriptions.

0x1002_6038 (PP_CSC_COEF_123) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	C0								C1							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	C2								C3							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-21. PP CSC Coefficient_123 Register

Table 41-22. PP Lock Bit Register Field Descriptions

Name	Description
31–24 C0[7:0]	CSC Coefficient 0 Range from 0 to 1.9921875 in steps of (1/128)
23–16 C1[7:0]	CSC Coefficient 1 Range from 0 to 1.9921875 in steps of (1/128)
15–8 C2[7:0]	CSC Coefficient 2 Range from 0 to 1.9921875 in steps of (1/128)
7–0 C3[7:0]	CSC Coefficient 3 Range from 0 to 1.9921875 in steps of (1/128)

41.4.16 PP CSC COEF_4 Register

Figure 41-22 shows the register; Table 41-23 provides its field descriptions.

0x1002_603C (PP_CSC_COEF_4) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	X0	C4								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-22. PP CSC Coefficient_4 Register

Table 41-23. PP CSC COEF_4 Register Field Descriptions

Name	Description
31–10	Reserved. These bits are reserved and should read 0
9 X0	X0. Luminance Component Offset 1 = 16 0 = 0
8–0 C4	CSC Coefficient 4. Range from 0 to 3.9921875 in steps of 1/128

Table 41-24. CSC Coefficient Usage

YUV Format	YUV/RGB Conversions
YCbCr	R = C0*(Y – X0) + C1*(Cr-128) G = C0*(Y – X0) – C2*(Cb-128) – C3*(Cr-128) B = C0*(Y – X0) + C4*(Cb-128)
YUV	R = C0*(Y – X0) + C1*(V-128) G = C0*(Y – X0) – C2*(U-128) – C3*(V-128) B = C0*(Y – X0) + C4*(U-128)

41.4.17 PP Resize Coefficient Table

Figure 41-23 shows the register; Table 41-25 provides its field descriptions.

0x1002_6000–0x1002_607C (PP_RESIZE_COEF_TBL) Access: User Write-Only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									w[4:0]					n[1:0]		OP
W	0	0	0	0	0	0	0	0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-23. PP Resize Coefficient Table (Array of 32 Resize Coefficients)

NOTE

This is a write-only register.

Table 41-25. PP Resize Coefficient Table Register Field Descriptions

Name	Description
31–8	Reserved. These bits are reserved and should read 0.
7–4 w	Weighting Coefficient. These bits set the weighting coefficient applied to the older of the two pixels used in the resize equation. Valid values for weight are 0, 2 to 30, and 31. A value of 31 is treated as 32 and therefore 31 is an invalid co-efficient. The resizing algorithm uses w as the Weighting Coefficient 1, w1. w1 = w Weighting coefficient 2, w2, is calculated as: w2 = 32–w1
2–1 n	Number of pixels to read. These bits set the number of new pixels to read. 00 No pixels are read. 01 1 new pixel is read. 10 2 new pixels are read. 11 3 new pixels are read.
0 OP	Output Pixels. This bit controls if pixels are output 1 Pixels are output. 0 No pixels are output.

41.5 Pre-Processor

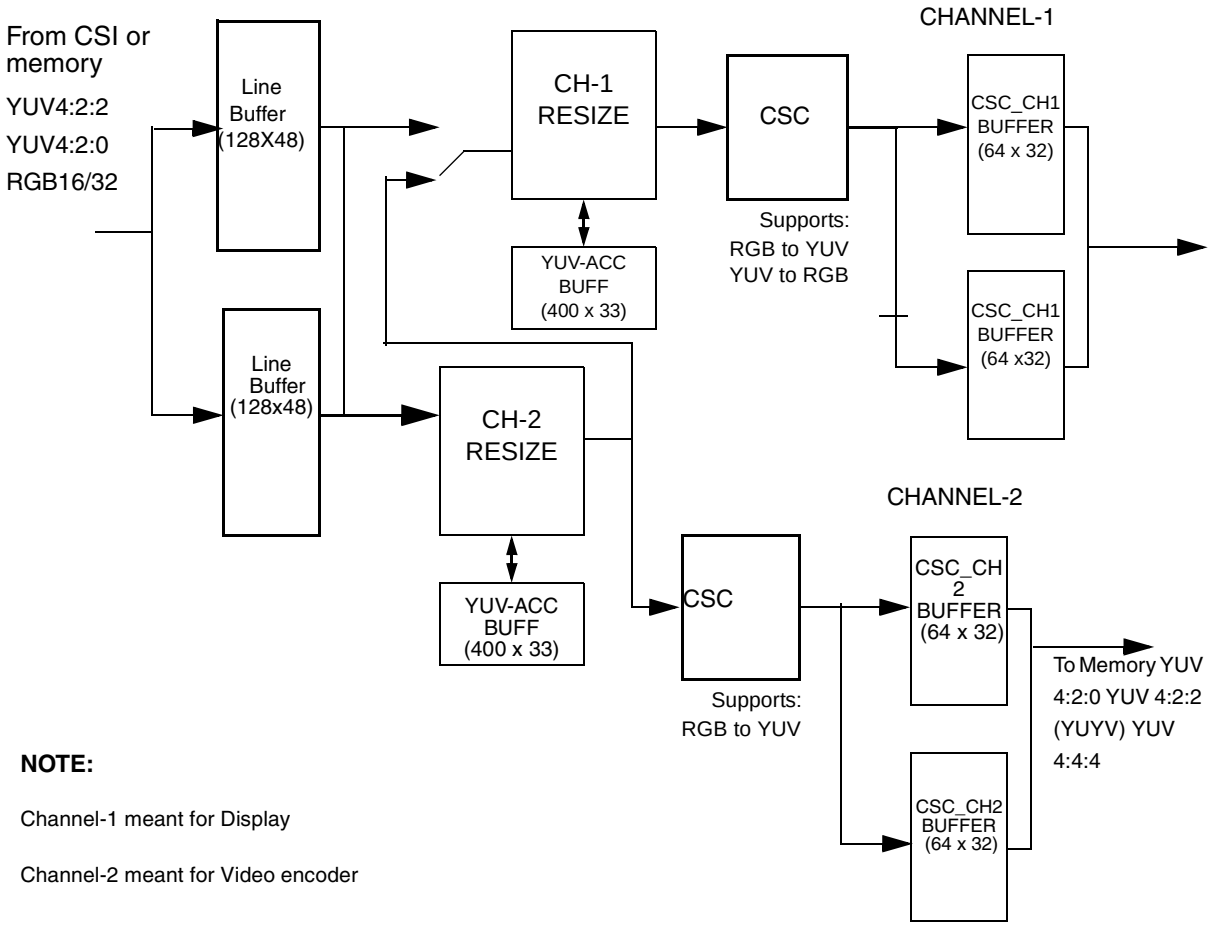


Table 41-26. Pre-Processing Block Diagram

The Pre-Processor receives input from main memory or from the camera sensor via the CMOS Sensor Interface (CSI) module and outputs two channels of data, one for video encoding and another for video display. Input data undergoes resize in a Channel-1 and Channel-2 resize block which provides programmable downscaling. The Channel-1 resize can be connected in cascade or parallel to Channel-2 resize. This is followed by programmable Color Space Conversion. The CSC block provides conversion from YUV to RGB and RGB to YUV for CH-1 and RGB to YUV for Channel-2.

After CSC, the data is channelled into memory. Channel-1 output is meant for display and both RGB and YUV 4:2:2 (interleaved) formats are supported. Data output on Channel-2 is meant for video encoding and various YUV formats are supported in this path. Namely, the YUV 4:2:0 (planar) format matches most MPEG-4 video encoder inputs and is required by the on-chip MPEG-4 encoder.

41.5.1 Features

41.5.2 Input Data Formats

Table 41-27 shows the input data formats for the Pre-Processor.

Table 41-27. Input Data Formats

Source	Format	Resolution
CSI	RGB	16 bpp
	RGB	32 bpp (unpacked RGB888)
	YUV 4:2:2	Pixel interleaved
	YUV 4:4:4	32 bpp—pixel interleaved
Memory	RGB	16 bpp
	RGB	32 bpp (unpacked RGB888)
	YUV 4:2:2	Pixel interleaved
	YUV 4:2:0	Band interleaved (IYUV and YV12)
	YUV 4:4:4	32 bpp—pixel interleaved

41.5.2.1 Input Size

The Pre-Processor can accept frames as small as 32 × 32 pixels to as large as 2044 × 2044 pixels. For YUV 4:2:0, the maximum frame size is limited to 2040 × 2040 pixels.

41.5.2.2 Resize Ratios

There are two independent and identical resize blocks in the Pre-Processor, called the Channel-1 Resize block and the Channel-2 Resize block. The Channel-1 resize block can either work in parallel to the Channel-2 resize block, that is, both blocks connect to the input, or in cascade to the output of the Channel-2 resize.

Each resize block supports two resize algorithms: bilinear and averaging. For each resize block, a user needs select one of the algorithms through register bit before starting resizing.

Table 41-28. Resize Ratios

Resize Block	Resize Ratio	Description
Channel-1 Resize	1:1	Data is copied from input to output. No resize is effected.
	Programmable from 1:1 to 8:1 or when cascaded, from 8:1 to 64:1	Downscaling
Channel-2 Resize	1:1	Data is copied from input to output. No resize is effected.
	Programmable from 1:1 to 8:1	Downscaling

41.5.2.3 Output Formats

Table 41-29 shows the output formats supported on Channel-1 and Channel-2.

Table 41-29. Output Formats

Channel	Output Format	Resolution	Description
Channel-1	RGB	8 bpp	4 pixels in an output word
		16 bpp	2 pixels in an output word
		32 bpp	Unpacked RGB888—1 pixel in an output word
	YUV 4:2:2	YUYV,YVYU,UYVY, VYUY	Pixel interleaved—2 pixels in an output word
Channel-2	YUV 4:2:2	YUYV	Pixel interleaved—2 pixels in an output word
	YUV 4:2:0	IYUV, YV12	Band interleaved
	YUV 4:4:4	YUV0	Pixel interleaved—1 pixel in an output word

41.5.2.4 Output Data

Output data is always written to memory pointed to by the destination pointers of Channel-1 and Channel-2.

41.5.2.5 Output Size

Minimum output size on Channel-1 and Channel-2 is 32×32 pixels and resize ratios must be calculated accordingly to prevent data truncation at the output.

41.5.3 Resize

The Channel-1 Resize and Channel-2 Resize modules are primarily used to resize captured sensor images and suitably format them to match the viewfinder display and video encoder input requirements. For example, an image or live video input from a 640 x 480 camera sensor can be resized to fit an LCD display of 240 x 320 or 320 x 320. The resizer can also prepare data for video encoding (Channel-2).

Each resize module implements two resize algorithms: bilinear and averaging, and one of the algorithms can be enabled at any single time.

41.5.3.1 Bilinear Resize in PrP

Bilinear interpolation algorithm is implemented and recommended for resize ratios between 1:1 and 2:1. Here two adjacent pixels are loaded and multiplied by respective weighting coefficients to produce an output pixel.

The weighting coefficients for a particular resize ratio are calculated by software and preloaded into the resize coefficient registers (eg. Register eMMA PrP_CH1_RZ_HORI_COEF1) of the PrP from which the resize block reads the coefficients to use.

For example, the output samples for the 5:3 bilinear interpolation can be calculated as follows:

out[0] = 5/8 * in[0] + 3/8 * in[1]

out[1] = 0/8 * in[1] + 8/8 * in[2]

out[2] = 3/8 * in[3] + 5/8 * in[4]

the entries should be (5,1),(0,1),(X,0),(3,1),(X,0).

A programmable resize engine has been implemented in hardware, which reads instructions from the PRP resize coefficient registers (Register eMMA PrP_CH1_RZ_HORI_COEF1, PrP_CH1_RZ_HORI_COEF2, PrP_CH1_RZ_VERT_COEF1, PrP_CH1_RZ_VERT_COEF2, PrP_CH2_RZ_HORI_COEF1, PrP_CH2_RZ_HORI_COEF2, PrP_CH2_RZ_VERT_COEF1, PrP_CH2_RZ_VERT_COEF2). An output pixel will be generated with the value (w1 * in1 + w2 * in2)/8 where in1 and in2 are two adjacent pixels. The resize engine will then read in one new input pixel, if the corresponding VOn bit in the corresponding registers (Register eMMA PrP_CH1_RZ_HORI_VALID, PrP_CH1_RZ_VERT_VALID, PrP_CH2_RZ_HORI_VALID, or PrP_CH2_RZ_VERT_VALID) is true “1”. Therefore, each resize instruction is in the form of (w1, n) where each coefficient (w1) is represented with 3 bits and n with 1-bit, and stored in 2 registers, one for w1 coefficient, and one for n valid. w2 is calculated as 8-w1.

- Allowed values of w1 are 0,1,2,3,4,5,6 and 7. w1 coefficient value of 7 (3’b111) is treated as 8 (4’b1000), consequently, w1 coefficient value of 7 is not possible.
- PrP resize weighting coefficients only have 3 bits, not 5 bits as in PP resize.

41.5.3.2 Averaging Resize in PrP

This is a special convolution filter where a weighted average of every N input pixels will produce an output pixel, when the resize ratio is N:1. Suppose in[0], in[1], ... in[N] are input pixels, w[i] are weighting coefficients, and out[0] is the corresponding output pixel, then

out[0] = w[0] * in[0] + w[1] * in[1] +... + w[N] * in[N].

The resize instruction for PRP averaging is also in the form of (w, n) where each coefficient (w) is represented with 3 bits and n with 1-bit. The averaging resizer is also implemented as a programmable resize engine. One pixel is loaded every cycle to the resize engine and a multiplication and accumulate operation is applied. A temporary register, T is used to store the interim result.

On reset or at the beginning of a line (for horizontal resize) or a row (for vertical resize), T is reset to 0. Then the process is as follows in every cycle:

1. Load a new input pixel value into the “in” register
2. T=T+ w*in; where 0 <= w <= 7.
3. If n bit is true “1”, then an output pixel is produced as:

out = T/8. After that T is reset to 0.

The sum of all w coefficients (w[0]+w[1]+...+w[n]) for an output pixel will be 8. The resize instructions (w, n) are stored in 2 registers, one for w coefficient (for example, PrP_CH1_RZ_HORI_COEF1), and one for n (for example, PrP_CH1_RZ_HORI_VALID).

- w coefficient value of 7 (3’b111) is treated as 8 (4’b1000), consequently, w coefficient value of 7 is not possible.

Following are some example resize tables:

3:1: (2, 0) (4, 0) (2, 1)

4:1: (1, 0) (3, 0) (3, 0) (1, 1)

7:1: (1, 0) (1, 0) (1, 0) (2, 0) (1, 0) (1, 0) (1, 1)

Non N:1 resize ratios - M:N - (greater than 2:1) can be supported in averaging resize algorithm by converting into N X[i]:1 resizing ratios, where the sum of X[i], i=1..N will be M. For example, 5:2 resize can be converted into 3:1 + 2:1 resize and the resize table could be 5:2 (2, 0) (4, 0) (2, 1) (4, 0) (4, 1)

41.5.3.3 Combined Bilinear and Averaging

Can choose bi-linear and averaging at same time for two different direction. That is horizontal can be averaging and vertical be using bi-linear.

41.5.3.4 Resize Output Image Size

For M:N resize ratio the output image width is

RZOUTWIDTH = [RZINWIDTH * (N/M)] where [] integer operation.

For example if input image width is 176 and resize ratio of 5:3 then

RZWIDTHOUT = [176*3/5] = 105 pixels.

It should be noted that the height calculation will be updated.

41.5.3.5 Channel-1 Output

Table 41-30 summarizes the Channel-1 output requirements and limits. All channel-1 output must be word aligned.

Table 41-30. Channel-1 Output Formats and Sizes

Output Resolution	CH1_WIDTH	Description	Example Output Format
8 bpp	RZWIDTHOUT and ~0x03	Rounded down to a multiple of 4	RGB 332
16 bpp	RZWIDTHOUT and ~0x01	Rounded down to a multiple of 2	RGB 565 or YUV 4:2:2
32 bpp	RZWIDTHOUT	Output is word aligned	Unpacked RGB888

The final output on Channel-1 is controlled by the CH1_OUT_IMAGE_WIDTH and CH1_OUT_IMAGE_HEIGHT settings and the maximum values for these cannot exceed those of CH1_WIDTH and RZHEIGHTOUT.

41.5.3.6 Channel-2 Output

Table 41-31 summarizes the Channel-2 output requirements and limits. The inputs are from CH2 resize.

Table 41-31. Channel-2 Output Formats and Sizes

Output Format	Y_Width	Y_Height	U, V Width, and Height
YUV 4:2:0	RZWIDTHOUT and ~0x07 (multiple of 8)	RZHEIGHTOUT and ~0x01	Band interleaved: U_WIDTH = Y_WIDTH/2 V_WIDTH = Y_WIDTH/2 U_HEIGHT = Y_HEIGHT/2 V_HEIGHT = Y_HEIGHT/2
YUV 4:2:2	RZWIDTHOUT and ~0x01	RZHEIGHTOUT	Pixel interleaved
YUV 4:4:4	RZWIDTHOUT	RZHEIGHTOUT	Pixel interleaved

41.5.4 Color Space Conversion (CSC)

If an image sensor produces RGB output then Color Space Conversion (CSC) is required to convert from RGB to a YUV color space format used for MPEG-4 encoding.

The MPEG-4 standard allows for a number of Color Space Conversion (CSC) scenarios. The particular type of CSC used by an MPEG-4 encoder is signaled in the bit stream syntax and is determined by two fields, these being *matrix_coefficients* and *video_range*. To allow total flexibility, programmable CSC matrices are implemented.

Calculation of each coefficient depend upon precision or steps it need to represent. Support a coefficient is specified in steps of (1/128) and user want to represent 0.345 in that then they need to calculate as INT(128*0.345) = 44.

41.5.5 RGB to YUV

The MPEG-4 encoder only operates in YUV (or also referred to as YCbCr) color space and therefore if an image sensor produces RGB output, there is a need to perform RGB to YUV color space conversion. The MPEG-4 standard allows a number of YUV formats and these can be summarized by four color space conversion equations.

The conversion from RGB to YUV 4:2:0 (and YUV 4:2:2) can be decomposed into two steps. The first step is the conversion from RGB to YUV 4:4:4 and the second step is the down sampling from YUV 4:4:4 to YUV 4:2:0 (or YUV 4:2:2). Down sampling from YUV 4:4:4 to YUV 4:2:0 is done by skipping alternate pixel and lines. Similarly down sampling from YUV 4:4:4: to YUV 4:2:2 is done by skipping alternate pixels.

The conversion matrices for color space conversion (CSC) and the downsampling procedure is discussed below.

The MPEG-4 standard allows four CSC matrices to be employed for RGB to YUV conversion. The actual matrix that is used needs to be indicated in the bit stream to allow the decoder to choose the appropriate inverse matrix for YUV to RGB conversion. See Table 41-32.

Table 41-32. YUV to RGB CSC Equations

Equation Name	Matrix Coefficient	Video Range	Input to CSC and Notation	Matrix
A1	4, 5 or 6 (Set A)	1	YUV Y ranges from 0–255 U ranges from 0–255 V ranges from 0–255	Y = 0.299*R + 0.587*G + 0.114*B U = -0.169*R – 0.331*G + 0.5*B + 128 V = 0.5*R - 0.419*G - 0.081*B + 128
A0	4, 5 or 6 (Set A)	0	YCrCb Y ranges from 16–235 Cr ranges from 16–240 Cb ranges from 16–240	Y = 0.2568*R + 0.5041*G + 0.0979*B + 16 Cb = -0.1484*R - 0.2907*G + 0.4392*B + 128 Cr = 0.4392*R - 0.3680*G - 0.0711*B + 128
B1	1 or 7 (Set B)	1	Y'U'V' Y' ranges from 0–255 U' ranges from 0–255 V' ranges from 0–255	Y = 0.2126*R + 0.7152*G + 0.0722*B U = -0.115*R - 0.386*G + 0.5*B + 128 V = 0.5*R - 0.454*G - 0.046*B + 128
B0	1 or 7 (Set B)	0	Y'Cr'Cb' Y' ranges from 16–235 Cr' ranges from 16–240 Cb' ranges from 16–240	Y = 0.1826*R + 0.6142*G + 0.0620*B + 16 Cb = -0.1010*R - 0.3390*G + 0.4392*B + 128 Cr = 0.4392*R - 0.3988*G - 0.0404*B + 128

41.5.5.1 YUV to RGB

For YUV to RGB conversion matrices, refer to Table 41-4.

41.5.5.2 Clipping of RGB and YUV Outputs

The output RGB or YUV values are clipped to the range of 0 to 255.

41.5.6 Frame Skip

Frame skipping is done to reduce the output frame rate. It can be done at three stages. Firstly at input, secondly at Channel-1 output and thirdly at Channel-2 output by using IN_SKIP,CH1_SKIP and CH2_SKIP bits of PRP_CNTL register, respectively. Frame Skip is valid only when input is from CSI (CSIEN = 1).

Input frames are skipped by using IN_SKIP then passed to Channel-1 and Channel-2. Each skip is controlled independently.

Example 1: Channel-1 and Channel-2 are configured in parallel

1. IN_SKIP = 3'b010
2. CH1_SKIP = 3'b001
3. CH2_SKIP = 3'b100
4. Let the input frame sequence be: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
5. After applying IN_SKIP, the reduced sequence is: 0, 2, 3, 5, 6, 8, 9, 11, 12. These frames are sent to Channel-1 and Channel-2 (since these are configured in parallel).
6. Output from Channel-1 after applying CH1_SKIP is: 0, 3, 6, 9, 12

7. Output from Channel-2 after applying CH2_SKIP is: 0, 2, 3, 6, 8, 9, 12

Example 2: Channel-1 cascaded with Channel-2

1. IN_SKIP = 3'b010
2. CH1_SKIP = 3'b001
3. CH2_SKIP = 3'b100
4. Let the input frame sequence be: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
5. After applying IN_SKIP, the reduced sequence is: 0, 2, 3, 5, 6, 8, 9, 11, 12. These frames are sent to Channel-2
6. Output from Channel-2 after applying CH2_SKIP: 0, 2, 3, 6, 8, 9, 12. These frames are sent to Channel-1
7. Output from Channel-1 after applying CH1_SKIP: 0, 3, 8, 12

41.5.7 LOOP Mode (LEN)

LOOP mode is effective when the CSI-PrP link is enabled and has no effect when input frames are processed from memory. LOOP mode is enabled by setting LEN bit in PRP_CNTL. When this bit is enabled, output data is written into output buffers in ping-pong fashion for each enabled channel. Two memory buffers are used for each channel output. For example, the current frame is written to Buffer 1 and the next frame to Buffer 2 followed by Buffer 1, again.

Table 41-33. Loop Mode Ping Pong Registers

Channel	Buffer 1 Registers	Buffer 2 Registers
Channel-1	PRP_DEST_RGB1_PTR	PRP_DEST_RGB2_PTR
Channel-2	PRP_DEST_Y_PTR[1] PRP_DEST_CB_PTR[2] PRP_DEST_CR_PTR	PRP_SOURCE_Y_PTR[3] PRP_SOURCE_CB_PTR PRP_SOURCE_CR_PTR

1 PRP_DEST_Y_PTR and PRP_SOURCE_Y_PTR are used in YUV 4:2:0, YUV 4:2:2 and YUV 4:4:4 modes.

2 PRP_DEST_CB_PTR, PRP_DEST_CR_PTR and PRP_DEST_CB_PTR, PRP_DEST_CR_PTR are only used when output is in YUV 4:2:0 mode.

3 These registers are re-used as output pointers in LOOP mode.

While in LOOP mode, if Channel-1 or Channel-2 is disabled, the PrP continues to output data for that channel until the frame is completed. When the channel is re-enabled again, output always starts at the next alternate buffer.

41.5.8 Channel-1 and Channel-2 Enable

CH1EN and CH2EN bits of PRP_CNTL are used to enable Channel-1 and Channel-2 processing. If a frame is to be processed by both channels then both bits should be enabled at the same time. If data input is from memory and the two channels are enabled separately one after the other, then the second channel will receive partial frame data. For example, if the input frame size is 320x240 and Channel-1 is enabled

first and Channel-2 is enabled when 120 lines have already been processed, then lines 121 to 240 will be written to Channel-2. Line 121 will be treated as line 1 for Channel-2.

If data input is from CSI then processing always begins with a new frame. For example, when Channel-1 is enabled and processing of the first frame is in progress when Channel-2 is enabled then Channel-2 will only begin to output frames from the second frame onwards and frame boundaries are maintained in this mode.

If LOOP is not enabled then an enabled channel is automatically disabled after a single frame has been processed. If a channel is disabled in the midst of a frame, it will continue processing and stop when the full frame has been processed.

41.5.9 Channel-2 Flow Control

Flow control is done to control Channel-2 frame processing. Flow control is valid when the CSI input is enabled. Flow control is enabled by setting CH2FEN bit to ‘1’ in PRP_CNTL register. When flow control is enabled, CH2B1EN and CH2B2EN bits indicate if the corresponding buffer is ready to accept new data.

When flow control is disabled CH2B1EN and CH2B2EN are not checked and Channel-2 will write to Buffer-1 and Buffer-2, alternately. Buffer-1 and Buffer-2 are ping pong buffers whose memory address is configured as the Channel-2 destination pointers.

If flow control is enabled, then at the start of frame processing the buffer enable bits (CH2B1EN or CH2B2EN) are checked to determine if the alternate buffer is enabled for writing. If the buffer is enabled then an input frame is processed and at the end of frame, the buffer enable bit is reset. If the buffer is not enabled, then an overflow condition is signaled. The C2FCFO bit is set in PRP_INTRSTATUS. If the C2FCIE bit is enabled in PRP_INTRCNTL, then an interrupt is also raised. When an overflow condition is encountered, input frame processing stops until the buffer is enabled and processing begins at the next start of frame.

41.5.10 Line Buffer Overflow

When PRP processing speed is slower than input data arrival then line buffer overflow can happen. When a line buffer overflow occurs, the LB_OVI bit in PRP_INTRSTATUS is set. If LB_OVIE bit in PRP_INTRCNTL is enabled then an interrupt is raised.

The FRAME_SKIP bit in the PRP_CNTL register determines if processing re-starts at the next start of frame or continues when an overflow occurs. When an overflow occurs and FRAME_SKIP is ‘1’ then processing of that frame stops and continues at the next start of frame. When FRAME_SKIP is ‘0’ and an overflow occurs, then processing continues but there will be some missing data in the output frame contributing to errors in the picture.

41.5.11 Relationship of Register Fields Related to the Input Frame

Figure 41-24 shows the relationship between PrP_Y_SOURCE, PICTURE_X_SIZE, PICTURE_Y_SIZE and SOURCE_LINE_STRIDE. There are two rectangles in the diagrams. The inner rectangle shows the processed frame, however a larger amount of memory may be allocated, as bounded by the outside rectangle. This windowing relationship applies when PrP takes input from main memory.

Stride information is not used when CSI-PrP link is enabled and instead cropping takes place.

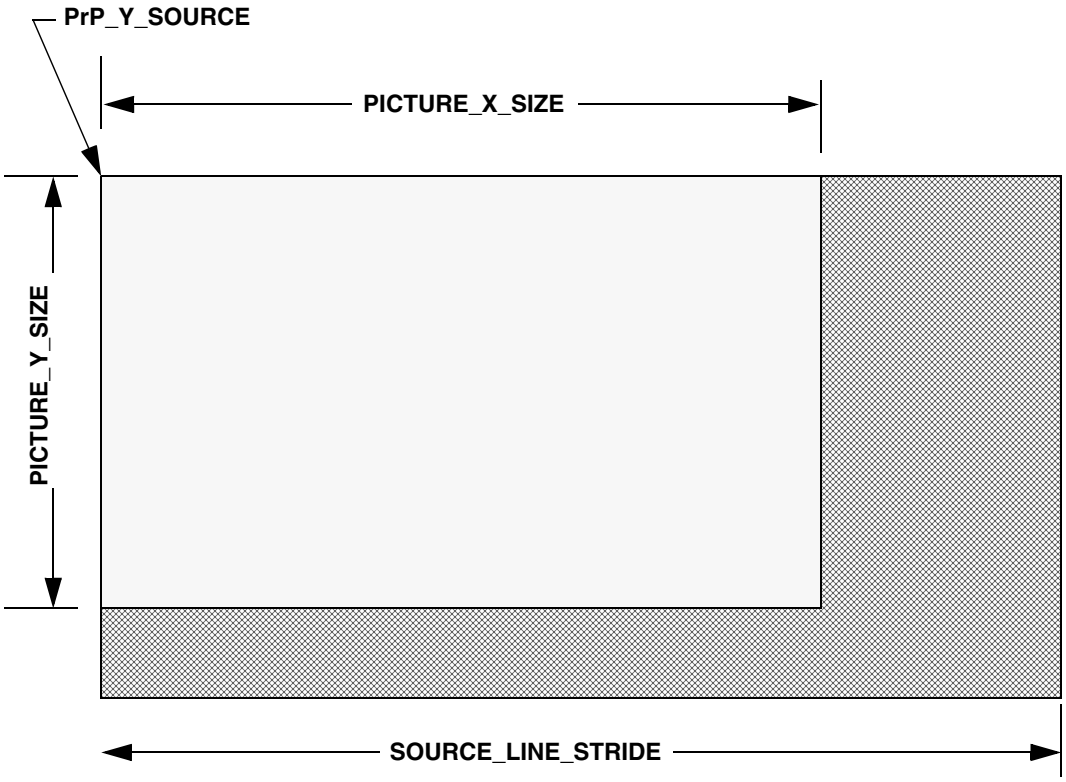


Figure 41-24. Memory Image Size and Source Line Stride

41.5.12 Relationship of Register Fields Related to Channel-1 Output Frame

Figure 41-25 shows the relationship between PrP_DEST_RGB1_PTR or PRP_DEST_RGB2_PTR, CH1_OUT_IMAGE_WIDTH, CH1_OUT_IMAGE_HEIGHT and CH1_OUT_LINE_STRIDE. There are two rectangles in the diagrams. The inner rectangle shows the Channel-1 written frame in memory, however a larger amount of memory may be allocated as bounded by the outer rectangle or the input image may be larger than the output image which is cropped.

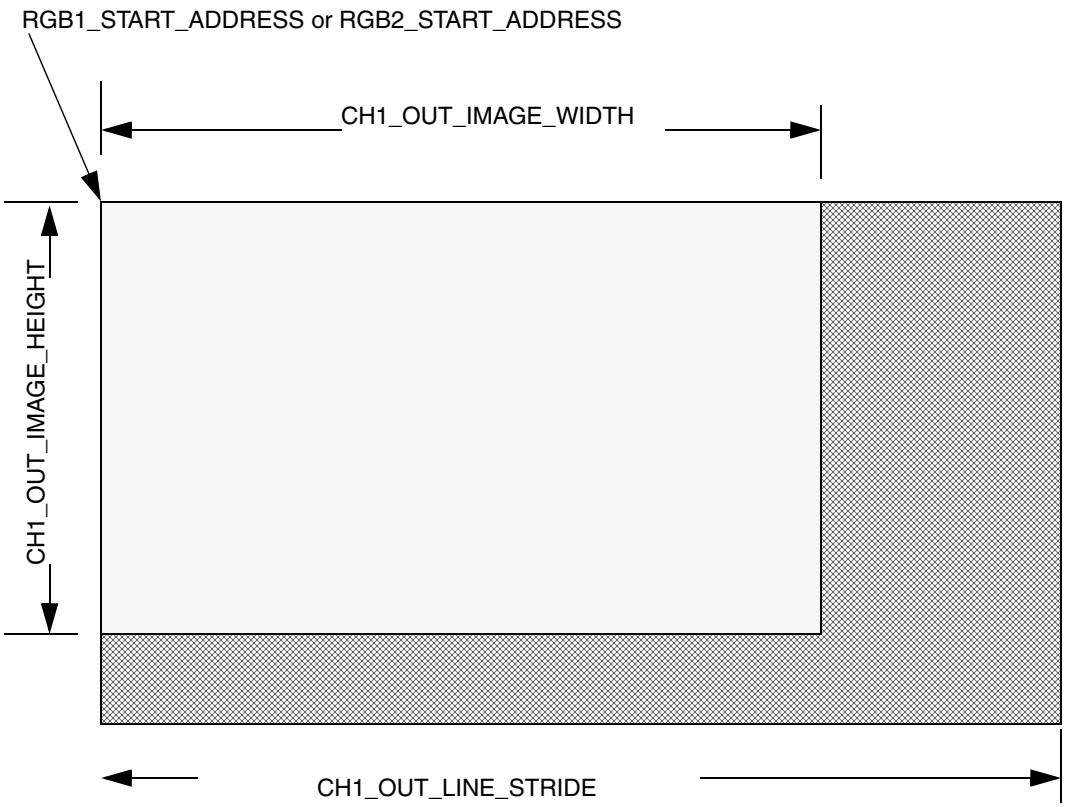


Figure 41-25. Memory Image and Output Stride

41.5.13 CSI Frame Cropping

The frame data from CSI can be cropped using CSI_LINE_SKIP and SOURCE_LINE_STRIDE of PRP_SRC_LINE_STRIDE register and PICTURE_X_SIZE and PICTURE_Y_SIZE of PRP_SRC_FRAME_SIZE register. When the input is 16-bit RGB or YUYV, SOURCE_LINE_STRIDE should be a multiple of two. SOURCE_LINE _STRIDE specifies the number of initial pixels to skip in a line and CSI_LINE_SKIP specifies the number of lines to skip.

The cropping parameter should be chosen such that this condition is always met.

SOURCE_LINE_STRIDE + PICTURE_X_SIZE <= CSI_FRAME_X_SIZE

CSI_LINE_SKIP + PICTURE_Y_SIZE <= CSI_FRAME_Y_SIZE

Cropping is enabled by setting WEN bit to ‘1’ in PRP_CNTL register.

Figure 41-26 shows the effect of the cropping parameters. The figure shows two rectangular areas. The outer rectangle shows the actual frame from CSI. The inner rectangle shows the size of the image user selects.

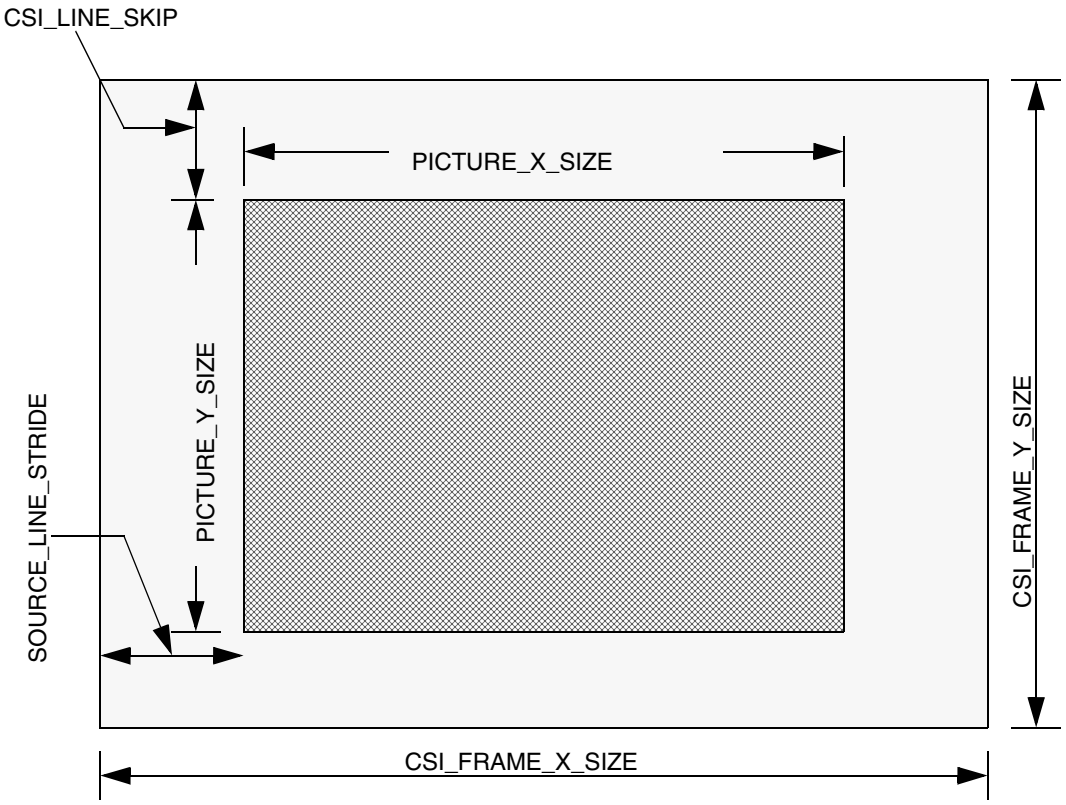


Figure 41-26. CSI Frame Cropping

NOTE

SOURCE_LINE_STRIDE and PICTURE_X_SKIP are specified in pixels and CSI_LINE_SKIP and PICTURE_Y_SIZE are specified in lines.

41.5.14 CSI-PrP Link

Figure 41-27 shows the timing diagram of the CSI-PrP dedicated link. The FIFO data from CSI is transferred to the Pre-Processor without any CPU intervention.

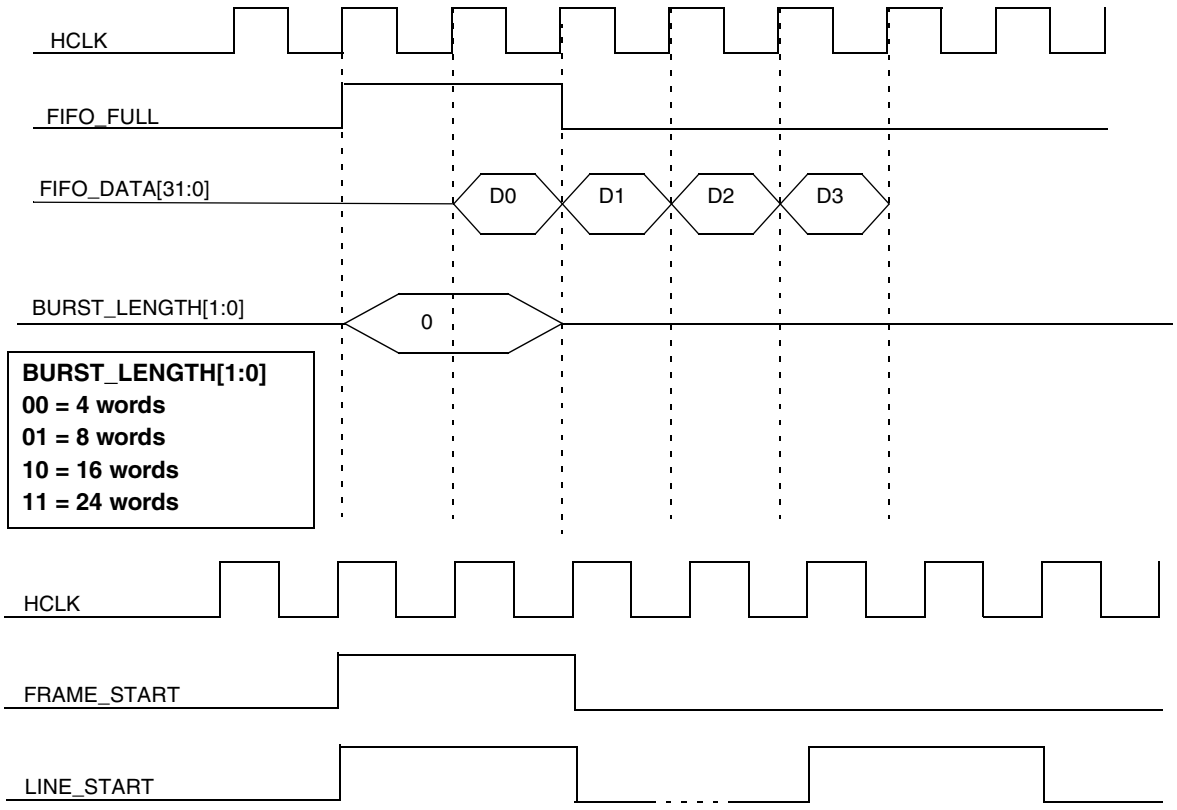


Figure 41-27. CSI-PrP Link

Table 41-34. CSI-PrP Link Internal Signals

Signal	Description
HCLK	AHB HCLK
FIFO_FULL	Asserted high when CSI FIFO has reached the high watermark
FIFO_DATA[31:0]	32-bit unidirectional data bus from CSI to PrP
FRAME_START	Asserted for 2 HCLKs when the CSI detects a Start-Of-Frame (VSYNC) condition
LINE_START	Asserted for 2 HCLKs when the CSI detects a Start-Of-Line (HSYNC) condition
BURST_LENGTH[1:0]	2-bit encoded to indicate to PrP how many data words there are in a FIFO_FULL burst. The FIFO high watermark must be chosen such that the PrP PICTURE_X_SIZE is a multiple of the burst length. YUV 4:2:2: Burst_count = PICTURE_X_SIZE / (BURST_LENGTH × 2) RGB (16 bpp): Burst_count = PICTURE_X_SIZE / (BURST_LENGTH × 2) RGB (32 bpp): Burst_count = PICTURE_X_SIZE / (BURST_LENGTH) Burst_count should be exact integer value.

41.6 Pre-Processor (PrP) Programming Model

All register reads and writes must be 32-bits access. Write 0 to reserved bits. All registers are read or write unless specified.

Table 41-35. PrP Register Memory Map

Address	Register Name	Access	Offset	Location
0x1002_6000 (PRP_CNTL)	PrP Control Register (PrP_CNTL)	R/W	0x0000_F232	41.6.1/41-44
0x1002_6004 (PRP_INTR_CNTL)	PrP Interrupt Control (PrP_INTRCNTL)	R/W	0x0000_0000	41.6.2/41-47
0x1002_6008 (PRP_INTRSTATUS)	PrP interrupt Status	R/W	0x0000_0000	41.6.3/41-49
0x1002_600C (PRP_SOURCE_Y_PTR)	PrP Source Y Frame Start Address	R/W	0x0000_0000	41.6.4/41-50
0x1002_6010 (PRP_SOURCE_CB_PTR)	PrP Source CB Frame Start Address	R/W	0x0000_0000	41.6.5/41-51
0x1002_6014 (PRP_SOURCE_CR_PTR)	PrP Source CR Frame Start Address	R/W	0x0000_0000	41.6.6/41-51
0x1002_6018 (PRP_DEST_RGB1_PTR)	PrP Destination RGB Frame-1 Start Address	R/W	0x0000_0000	41.6.7/41-52
0x1002_601C (PRP_DEST_RGB2_PTR)	PrP Destination RGB Frame-2 Start Address	R/W	0x0000_0000	41.6.8/41-53
0x1002_6020 (PRP_DEST_Y_PTR)	PrP Destination Y Frame Start Address	R/W	0x0000_0000	41.6.9/41-53
0x1002_6024 (PRP_DEST_CB_PTR)	PrP Destination CB Frame Start Address	R/W	0x0000_0000	41.6.10/41-54
0x1002_6028 (PRP_DEST_CR_PTR)	PrP Destination CR Frame Start Address	R/W	0x0000_0000	41.6.11/41-55
0x1002_602C (PRP_SRC_FRAME_SIZE)	PrP Source Frame Size	R/W	0x0140_00F0	41.6.12/41-55
0x1002_6030 (PRP_DEST_CH1_LINE_STRIDE)	PrP Channel-1 Line Stride	R/W	0x0000_0280	41.6.13/41-56
0x1002_6034 (PRP_SRC_PIXEL_FORMAT_CNTL)	PrP Source Pixel Format Control	R/W	0x2200_0888	41.6.14/41-57
0x1002_6038 (PRP_CH1_PIXEL_FORMAT_CNTL)	PrP CH1 Pixel Format Control	R/W	0x2CA0_0565	41.6.15/41-59
0x1002_603C (PRP_CH1_OUT_IMAGE_SIZE)	PrP CH1 Output Image Size	R/W	0x0540_04F0	41.6.16/41-61
0x1002_6040 (PRP_CH2_OUT_IMAGE_SIZE)	PrP CH2 Output Image Size	R/W	0x0140_04F0	41.6.17/41-61
0x1002_6044 (PRP_SRC_LINE_STRIDE)	PrP Source Line Stride	R/W	0x0000_0000	41.6.18/41-62
0x1002_6048 (PrP_CSC_COEF_012)	PrP CSC Coefficients C0, C1, C2	R/W	0x1005_A02C	41.6.19/41-63
0x1002_604C (PrP_CSC_COEF_345)	PrP CSC Coefficients C3 to C5	R/W	0x0B67_1800	41.6.20/41-64

Table 41-35. PrP Register Memory Map (continued)

Address	Register Name	Access	Offset	Location
0x1002_6050 (PrP_CSC_COEF_678)	PrP CSC Coefficients C6 to C8	R/W	0x0000_0000	41.6.21/41-65
0x1002_6054 (PrP_CH1_RZ_HORI_COEF1)	Prp Channel 1 Resize Horizontal Coefficients	R/W	0x0000_0007	41.6.22/41-67
0x1002_6058 (PrP_CH1_RZ_HORI_COEF2)	Prp Channel 1 Resize Horizontal Coefficients	R/W	0x0000_0000	41.6.23/41-68
0x1002_605C (PrP_CH1_RZ_HORI_VALID)	Prp Channel 1 Resize Horizontal Output Data Valid	R/W	0x0100_0001	41.6.24/41-69
0x1002_6060 (PrP_CH1_RZ_VERT_COEF1)	Prp Channel 1 Resize Vertical Coefficients	R/W	0x0000_0007	41.6.25/41-70
0x1002_6064 (PrP_CH1_RZ_VERT_COEF2)	Prp Channel 1 Resize Vertical Coefficients	R/W	0x0000_0000	41.6.26/41-71
0x1002_6068 (PrP_CH1_RZ_VERT_VALID)	Prp Channel 1 Resize Vertical Output Data Valid	R/W	0x0100_0001	41.6.27/41-72
0x1002_606C (PrP_CH2_RZ_HORI_COEF1)	Prp Channel 2 Resize Horizontal Coefficients	R/W	0x0000_0007	41.6.28/41-73
0x1002_6070 (PrP_CH2_RZ_HORI_COEF2)	Prp Channel 2 Resize Horizontal Coefficients	R/W	0x0000_0000	41.6.29/41-74
0x1002_6074 (PrP_CH2_RZ_HORI_VALID)	Prp Channel 2 Resize Horizontal Output Data Valid	R/W	0x0100_0001	41.6.30/41-75
0x1002_6078 (PrP_CH2_RZ_VERT_COEF1)	Prp Channel 2 Resize Vertical Coefficients	R/W	0x0000_0007	41.6.31/41-76
0x1002_607C (PrP_CH2_RZ_VERT_COEF2)	Prp Channel 2 Resize Vertical Coefficients	R/W	0x0000_0000	41.6.32/41-78
0x1002_6080 (PrP_CH2_RZ_VERT_VALID)	Prp Channel 2 Resize Vertical Output Data Valid	R/W	0x0100_0001	41.6.33/41-79

41.6.1 PrP Control Register

Figure 41-28 shows the register, and Table 41-36 provides its field descriptions.

0x1002_6000 (PRP_CNTL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CH2FEN	CH2B2EN	CH2B1EN	RZ_FIFO_LEVEL[1:0]		INPUT_FIFO_LEVEL[1:0]		CH2_TSKIP			CH1_TSKIP			IN_TSKIP		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH1BYP	WEN	CLKEN	SWRST	SKIPFRAME	CH2_L EN	CH1_LEN	CH2_OUT_MODE[1:0]		CH1_OUT_MODE[1:0]		DATAIN MODE [1:0]		CSIEN	CH2EN	CH1EN
Reset	1	1	1	1	0	0	1	0	0	0	1	1	0	0	1	0

Figure 41-28. PRP Control Register

Table 41-36. PRP Control Register Field Descriptions

Name	Description
31 CH2FEN	Channel 2 Flow Control Enable. When this bit is set then output is controlled by CH2B1EN and CH2B2EN, else output is written alternately to output buffers. 0 Disable Flow Control 1 Enable Flow Control
30 CH2B2EN	Channel 2 Buffer 2 Enable. This bit signals that Buffer 2 is ready for writing 0 Buffer is not ready for writing. 1 Buffer is ready for writing.
29 CH2B1EN	Channel 2 Buffer 1 Enable. This bit signals that Buffer 1 is ready for writing 0 Buffer is not ready for writing. 1 Buffer is ready for writing.
27–28 RZ_FIFO_LEVEL[1:0]	Resize Fifo Level. Selects the depth of resize buffers. The buffers are 24 bits wide. 00 64 words 01 48 words 10 32 words 11 16 words
25–26 INPUT_FIFO_LEVEL[1:0]	Input Fifo Level. Selects the depth of input line buffers. The buffers are 48 bits wide. When data is sourced from CSI (CSIEN = 1) this bit should always set to “00”. 00 128 words 01 96 words 10 64 words 11 32 words

Table 41-36. PRP Control Register Field Descriptions (continued)

Name	Description
22–24 CH2_TSKIP[2:0]	Channel 2 Skip Control. Selects frames to skip on Channel 2 output. This bit is valid when CSIEN=1. 000 no skip 001 skip 1 out of every 2 (1-0) 010 skip 1 out of every 3 (1-0-1) 011 skip 2 out of every 3 (1-0-0) 100 skip 1 out of every 4 (1-1-1-0) 101 skip 3 out of every 4 (1-0-0-0) 110 skip 2 out of every 5 (1-0-1-0-1) 111 skip 4 out of every 5 (1-0-0-0-0)
19–21 CH1_TSKIP[2:0]	Channel 1 Skip Control. Selects frames to skip for Channel 1 output. This bit is valid when CSIEN=1. 000 no skip 001 skip 1 out of every 2 (1-0) 010 skip 1 out of every 3 (1-0-1) 011 skip 2 out of every 3 (1-0-0) 100 skip 1 out of every 4 (1-1-1-0) 101 skip 3 out of every 4 (1-0-0-0) 110 skip 2 out of every 5 (1-0-1-0-1) 111 skip 4 out of every 5 (1-0-0-0-0)
16–18 IN_TSKIP[2:0]	Input Frame Skip. Selects frames to skip from CSI. This bit is valid when CSIEN=1. 000 no skip 001 skip 1 out of every 2 (1-0) 010 skip 1 out of every 3 (1-0-1) 011 skip 2 out of every 3 (1-0-0) 100 skip 1 out of every 4 (1-1-1-0) 101 skip 3 out of every 4 (1-0-0-0) 110 skip 2 out of every 5 (1-0-1-0-1) 111 skip 4 out of every 5 (1-0-0-0-0)
15 CH1BYP	Channel-1 Bypass. Cascade control of Channel-1 resize 0 Cascade Channel-1 resize block 1 Disable cascade of Channel-1 resize
14 WEN	Window Enable. Enables input windowing feature from main memory or cropping from CSI. 0 Disable 1 Enable
13 CLKEN	Clock Gating Enable. This bit controls clock gating to the PrP. When clock gating is disabled, logic in the PrP is always clocked. When clock gating is enabled, then clock is turned on when PrP module is enabled. 1 Clock gating off 0 Clock gating on (power saving feature)
12 SWRST	Software Reset. Writing ‘1’ to this bit resets entire PrP module. All registers return to their default values. This bit is self-clearing.
11 SKIP_FRAME	FRAME_SKIP. This bit selects whether to continue or stop when we get input FIFO overflow error. 0 Continue 1 Stop

Table 41-36. PRP Control Register Field Descriptions (continued)

Name	Description
10 CH2_LEN	Channel-2 Loop Enable. This bit is valid only when input data is from CSI (CSIEN = 1). When enabled, output data is written in ping-pong fashion 0 Disable 1 Enable
9 CH1_LEN	Channel-1 Loop Enable. This bit is valid only when input data is from CSI (CSIEN = 1). When enabled, output data is written in ping-pong fashion. 0 Disable 1 Enable
7–8 CH2_OUT_MODE[1:0]	Channel-2 Output Mode. These bits select Channel-2 output format. 00 or 11 - YUV 4:2:0 (IYUV, YV12) 01 YUV 4:2:2 (YUYV) 10 YUV 4:4:4 (YUV0)
5–6 CH1_OUT_MODE[1:0]	Channel-1 Output Mode. These bits select the Channel-1 output format 00 8 bpp RGB 01 16 bpp RGB 10 32 bpp RGB 11 YUV 422
3–4 DATA_IN_MODE[1:0]	Data Input Mode. These bits set the input data format. 00 YUV 4:2:0 (not supported for CSIEN=1) 01 YUV 4:2:2 10 16-bit RGB data 11 32-bit RGB data
2 CSIEN	CSI Enable. If set, enables the CSI-PrP link and input is sourced from the CSI. When cleared, input is sourced from main memory. 0 Input from main memory 1 Input from CSI
1 CH2EN	Channel-2 Enable. This bit enables or disables Channel-2 output. 0 Disable 1 Enable When enabled, self clearing on success or error. Does not self clear if in LOOP mode (CH2_LEN = 1)
0 CH1EN	Channel-1 Enable. This bit enables or disables Channel-1 output. 0 Disable 1 Enable When enabled, self clearing on success or error. Does not self clear if in LOOP mode (CH1_LEN = 1)

41.6.2 PrP Interrupt Control Register

Figure 41-29 shows the register, and Table 41-37 provides its field descriptions.

0x1002_6004 (PRP_INTR_CNTL) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	CH2 OVF IE	LB OVF IE	0	CH2F C IE	0	CH1F C IE	CH2 WER R IE	CH1 WER R IE	RD ERR IE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-29. PRP Interrupt Control Register

Table 41-37. PRP Interrupt Control Register Field Descriptions

Name	Description
31–9	Reserved. These bits are reserved and should read 0.
8 CH2OVFIE	Channel-2 Overflow Interrupt Enable. If set, an interrupt is generated when frame is missed due to flow control process. 0 Disable 1 Enable
7 LBOVFIE	Line Buffer Overflow Interrupt Enable. If set, an interrupt is generated when Line buffer overflow occurs. 0 Disable 1 Enable
6	Reserved. This bit is reserved and should read 0.
5 CH2FCIE	Channel-2 Frame Complete Interrupt Enable. If set, an interrupt is generated when a frame is completely written out through Channel-2. 0 Disable 1 Enable
4	Reserved. This bit is reserved and should read 0.
3 CH1FCIE	Channel-1 Frame Complete Interrupt Enable. If set, an interrupt is generated when a frame is completely written out through Channel-1. 0 Disable 1 Enable
2 CH2WERRIE	Channel-2 Write Error Interrupt Enable. If set, an interrupt is generated when an AHB write error is encountered during Channel-2 data output to memory 0 Disable 1 Enable

Table 41-37. PRP Interrupt Control Register Field Descriptions (continued)

Name	Description
1 CH1WERRIE	Channel-1 Write Error Interrupt Enable. If set, an interrupt is generated when an AHB write error is encountered during Channel-1 data output to memory. 0 Disable 1 Enable
0 RDERRIE	Read Error Interrupt Enable. If set, an interrupt is generated when an AHB read error is encountered during data input from memory. 0 Disable 1 Enable

41.6.3 PrP Interrupt Status Register

Figure 41-30 shows the register, and Table 41-38 provides its field descriptions.

0x1002_6008 (PRP_INTRSTATUS) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	CH2 OVF	LB OVF	CH1 B1CI	CH1 B2CI	CH2 B1CI	CH2 B2CI	CH2 WRE RR	CH1 WRE RR	RD ERR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-30. PrP Interrupt Status Register

Table 41-38. PrP Interrupt Status Register Field Descriptions

Name	Description
31–9	Reserved. These bits are reserved and should read 0.
8 CH2OVF	Channel-2 Buffer Overflow. When this bit is set, it indicates that a frame was missed because both CH2B1EN and CH2B2EN are disabled.
7 LBOVF	Line Buffer Overflow. When this bit is set, it indicates that a line buffer overflow has occurred.
6 CH1B1CI	Channel-1 Buffer-1 Complete Interrupt. If Channel-1 is enabled and when this bit is set, it indicates that a frame has been completely written into Buffer-1 of Channel-1. This bit is set when input is either from CSI or from memory.
5 CH1B2CI	Channel-1 Buffer-2 Complete Interrupt. If Channel-1 is enabled and when this bit is set, it indicates that a frame has been completely written into Buffer-2 of Channel-1. This bit is set only when CSIEN=1 and LEN=1.

Table 41-38. PrP Interrupt Status Register Field Descriptions (continued)

Name	Description
4 CH2B1CI	Channel-2 Buffer-1 Complete Interrupt. If Channel-2 is enabled and when this bit is set, it indicates that a frame has been completely written into Buffer-1 of Channel-2. This bit is set when input is either from CSI or from memory.
3 C2B2CI	Channel-2 Buffer-2 Complete Interrupt. If Channel-2 is enabled and when this bit is set, it indicates that a frame has been completely written into Buffer-2 of Channel-2. This bit is set only when CSIEN=1 and LEN=1.
2 CH2WRERR	Channel-2 Write Error. If set, then an AHB write error to memory was encountered during Channel-2 data output. PrP has to be reset (SWRST=1) and re-initialized.
1 CH1WRERR	Channel-1 Write Error. If set, then an AHB write error was encountered during Channel-1 data output. PrP has to be reset (SWRST=1) and re-initialized.
0 READERR	Read Error. If set, then an AHB read error was encountered during data input from memory. PrP has to be reset (SWRST=1) and re-initialized.

41.6.4 PrP Source Y Address Register

Figure 41-31 shows the register, and Table 41-39 provides its field descriptions.

0x1002_600C (PRP_SOURCE_Y_PTR) Access: User Read/Write

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

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

Figure 41-31. PrP Source Y Address Register

Table 41-39. PrP Source Y Address Register Field Descriptions

Name	Description
31–0 PRP_SOURCE_Y_PTR	PRP_SOURCE_Y_PTR. 32-bit pointer to memory. In RGB and YUV 4:2:2 modes, this register sets the frame start address. In YUV 4:2:0 mode (input or output), this register sets the Luminance band start address of the frame. When CSIEN=1 and LEN=1, this register is re-used as the Channel-2 output Buffer-2 destination address for the above formats.

41.6.5 PrP Source Cb Address Register

Figure 41-32 shows the register, and Table 41-40 provides its field descriptions.

0x1002_6010 (PRP_SOURCE_CB_PTR) Access: User Read/Write

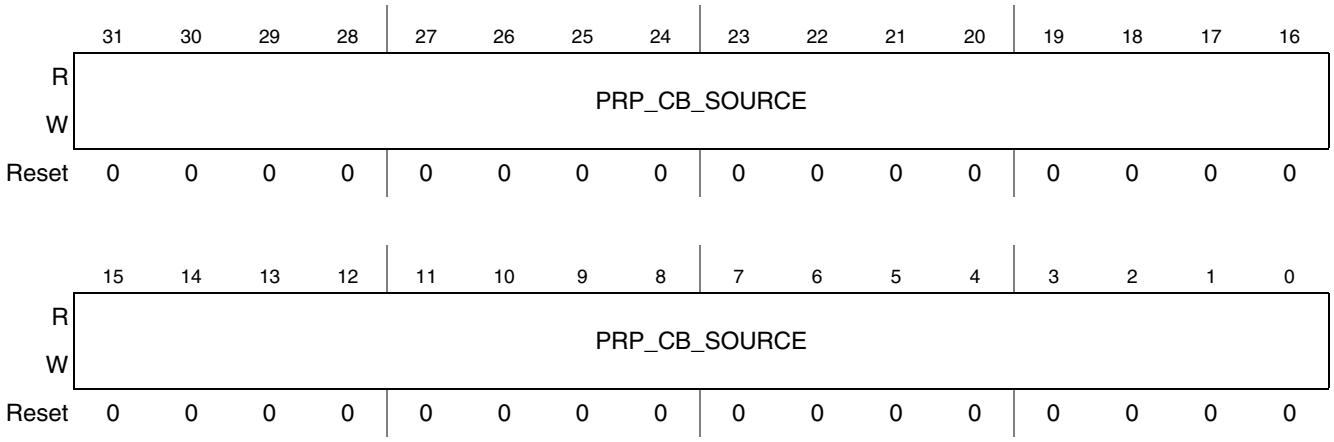


Figure 41-32. PrP Source Cb Address Register

Table 41-40. PrP Source Cb Address Register Field Descriptions

Name	Description
31–0 PRP_SOURCE_CB_PTR	PRP_SOURCE_CB_PTR. A 32-bit pointer to memory. This register sets the Chrominance (U or Cb) band start address when input is in YUV 4:2:0 band interleaved mode. It is not used for other input modes. When CSIEN=1 and LEN=1, this register is re-used as the Channel-2 U or Cb output Buffer-2 destination address for YUV 4:2:0.

41.6.6 PrP Source Cr Address Register

Figure 41-33 shows the register, and Table 41-41 provides its field descriptions.

0x1002_6014 (PRP_SOURCE_CR_PTR) Access: User Read/Write

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

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

Figure 41-33. PrP Source Cr Address Register

Table 41-41. PrP Source Cr Address Register Field Descriptions

Name	Description
31–0 PRP_SOURCE_CR_PTR	PRP_SOURCE_CR_PTR. 32-bit pointer to memory. This register sets the Chrominance (V or Cr) band start address when input is in YUV 4:2:0 band interleaved mode. It is not used for other input modes. When CSIEN=1 and LEN=1, this register is re-used as the Channel-2 Y or Cr output Buffer-2 destination address for YUV 4:2:0.

41.6.7 PrP Destination RGB1 Frame Start Address Register

Figure 41-34 shows the register, and Table 41-42 provides its field descriptions.

0x1002_6018 (PRP_DEST_RGB1_PTR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RGB1_START_ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RGB1_START_ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-34. PrP Destination RGB1 Start Address Register

Table 41-42. PrP Destination RGB1 Start Address Register Field Descriptions

Name	Description
31–0 PRP_DEST_RGB1_PTR	PRP_DEST_RGB1_PTR. 32-bit pointer to memory. This register sets the RGB or YUV 4:2:2 frame start address for Channel-1 output. When CSIEN=1 and LEN=1, this register becomes the output Buffer-1 address for Channel-1.

41.6.8 PrP Destination RGB2 Frame Start Address Register

Figure 41-35 shows the register, and Table 41-43 provides its field descriptions.

0x1002_601C (PRP_DEST_RGB2_PTR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RGB2_START_ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RGB2_START_ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-35. PrP Destination RGB2 Start Address Register

Table 41-43. PrP Destination RGB2 Start Address Register Field Descriptions

Name	Description
31–0 PRP_DEST_RGB2_PTR	PRP_DEST_RGB2_PTR. 32-bit pointer to memory. This register sets the output Buffer-2 RGB or YUV 4:2:2 frame start address for Channel-1 and is used only when CSIEN=1 and LEN=1.

41.6.9 PrP Destination Y Address Register

Figure 41-36 shows the register, and Table 41-44 provides its field descriptions.

0x1002_6020 (PRP_DEST_Y_PTR) Access: User Read/Write

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

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

Figure 41-36. PrP Destination Y Address Register

Table 41-44. PrP Destination Y Address Register Field Descriptions

Name	Description
31–0 PRP_Y_DEST	PRP_Y_DEST. 32-bit pointer to memory. This register sets the destination start address for Channel-2 output. This register is the Luminance band start address in YUV 4:2:0 mode and frame start address in YUV 4:2:2 and YUV 4:4:4 pixel interleaved modes. When CSIEN=1 and LEN=1, this register is the Buffer-1 output address.

41.6.10 PrP Destination Cb Address Register

Figure 41-37 shows the register, and Table 41-45 provides its field descriptions.

0x1002_6024 (PRP_DEST_CB_PTR) Access: User Read/Write

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

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

Figure 41-37. PrP Destination Cb Address Register

Table 41-45. PrP Destination Cb Address Register Field Descriptions

Name	Description
31–0 PRP_CB_DEST	PRP_CB_DEST. 32-bit pointer to memory. This register is used in YUV 4:2:0 band interleaved mode and sets the destination start address for Channel-2 U or Cb band data output. When CSIEN=1 and LEN=1, this register is the Channel-2 Buffer-1 output address for U or Cb band data in YUV 4:2:0 mode.

41.6.11 PrP Destination Cr Address Register

Figure 41-38 shows the register, and Table 41-46 provides its field descriptions.

0x1002_6028 (PRP_DEST_CR_PTR) Access: User Read/Write

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

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

Figure 41-38. PrP Destination Cr Address Register

Table 41-46. PrP Destination Cr Address Register Description

Name	Description
31–0 PRP_CR_DEST	PRP_CR_DEST. 32-bit pointer to memory. This register is used in YUV 4:2:0 band interleaved mode and sets the destination start address for Channel-2 V or Cr band data output. When CSIEN=1 and LEN=1, this register is the Channel-2 Buffer-1 output address for V or Cr band data in YUV 4:2:0 mode.

41.6.12 PrP Source Frame Size Register

Figure 41-39 shows the register, and Table 41-47 provides its field descriptions.

0x1002_602C (PRP_SRC_FRAME_SIZE) Access: User Read/Write

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

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

Figure 41-39. PrP Source Frame Size Register

Table 41-47. PrP Source Frame Size Register Field Descriptions

Name	Description
31–27	Reserved. These bits are reserved and should read 0.
26–16 PICTURE_X_SIZE	PICTURE_X_SIZE. These bits set the frame width to be processed in number of pixels. In YUV 4:2:0 mode, Cb and Cr widths are taken as PICTURE_X_SIZE/2 pixels. In YUV 4:2:0 mode, this value should be a multiple of 8-pixels. In other modes (RGB, YUV 4:2:2 and YUV 4:4:4) it should be a multiple of 4 pixels.
15–11	Reserved. These bits are reserved and should read 0.
10–0 PICTURE_Y_SIZE	PICTURE_Y_SIZE. These bits set the frame height in number of lines to be processed. In YUV 4:2:0 mode, Cb and Cr lines to be processed are taken as PICTURE_Y_SIZE/2. In YUV 4:2:0 mode, this value should be a multiple of 2.

41.6.13 PrP Destination Channel-1 Line Stride Register

Figure 41-40 shows the register, and Table 41-48 provides its field descriptions.

0x1002_6030 (PRP_DEST_CH1_LINE_STRIDE) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	CH1_OUT_LINE_STRIDE											
W																
Reset	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0

Figure 41-40. PrP Destination Channel-1 Line Stride Register

Table 41-48. PrP Destination Channel-1 Line Stride Register Field Descriptions

Name	Description
31–12	Reserved. These bits are reserved and should read 0.
11–0 CH1_OUT_LINE_STRIDE	CH1_OUT_LINE_STRIDE. These bits sets the distance in bytes between the start addresses of adjacent lines in Channel-1 output. The stride value should be a multiple of 4-bytes.

41.6.14 PrP Source Pixel Format Control Register

Figure 41-41 shows the register, and Table 41-49 provides its field descriptions.

0x1002_6034 (PRP_SRC_PIXEL_FORMAT_CNTL) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RED_WIDTH				GREEN_WIDTH				BLUE_WIDTH			
W																
Reset	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0

Figure 41-41. PrP Source Pixel Format Control Register

Table 41-49. PrP Source Pixel Format Control Register Field Descriptions

Name	Description
31	Reserved. This bit is reserved and should read 0.
30–26 RED_Y_OFFSET	RED_Y_OFFSET. These bits set the offset of the Red color or Y luminance component in the input pixel. The offset is calculated with respect to Bit 0.
25–21 GREEN_U_CB_OFFSET	GREEN_U_CB_OFFSET. These bits set the offset of the Green color, U or Cb Chrominance component in the input pixel. The offset is calculated with respect to Bit 0.
20–16 BLUE_V_CR_OFFSET	BLUE_V_CR_OFFSET. These bits set the offset of the Blue color, V or Cr Chrominance component in the input pixel. The offset is calculated with respect to Bit 0.
15–12	Reserved. These bits are reserved and should read 0.
11–8 RED_WIDTH	Red Width. These bits set the width in bits of the Red color component in the input pixel. Valid values are 0 to 8. Any value greater than 8 is set to 8.
7–4 GREEN_WIDTH	Green Width. These bits set the width in bits of the Green color component in the input pixel. Valid values are 0 to 8. Any value greater than 8 is set to 8.
3–0 BLUE_WIDTH	Blue Width. These bits set the width in bits of the Blue color component in the input pixel. Valid values are 0 to 8. Any value greater than 8 is set to 8.

Table 41-50 provides some examples of common input formats and the respective settings.

Table 41-50. Example Source Input Pixel Formats

Input Format	Pixel Arrangement	Offset	Width	PRP_SRC_PIXEL_FORMAT_CNTL
RGB 565	16 bpp	RED_Y_OFFSET=11 GREEN_U_CB_OFFSET=5 BLUE_V_CR_OFFSET=0	RED_WIDTH=5 GREEN_WIDTH=6 BLUE_WIDTH=5	0x2CA0_0565
RGB 888	Unpacked RGB888	RED_Y_OFFSET=16 GREEN_U_CB_OFFSET=8 BLUE_V_CR_OFFSET=0	RED_WIDTH=8 GREEN_WIDTH=8 BLUE_WIDTH=8	0x4100_0888
YUV 4:2:0	IYUV	Not applicable as input is band interleaved		Register is not used.
YUV 4:2:0	YV12			

Table 41-50. Example Source Input Pixel Formats (continued)

Input Format	Pixel Arrangement	Offset	Width	PRP_SRC_PIXEL_FORMAT_CNTL
YUV 4:2:2	YUYV	RED_Y_OFFSET=8 GREEN_U_CB_OFFSET=16 BLUE_V_CR_OFFSET=0	Width is not used, set to 8 by default.	0x2200_0888
YUV 4:2:2	YVYU	RED_Y_OFFSET=8 GREEN_U_CB_OFFSET=0 BLUE_V_CR_OFFSET=16		0x2010_0888
YUV 4:2:2	UYVY	RED_Y_OFFSET=0 GREEN_U_CB_OFFSET=24 BLUE_V_CR_OFFSET=8		0x0308_0888
YUV 4:2:2	VYUY	RED_Y_OFFSET=0 GREEN_U_CB_OFFSET=8 BLUE_V_CR_OFFSET=24		0x0118_0888
YUV 4:4:4	YUV0	RED_Y_OFFSET=24 GREEN_U_CB_OFFSET=16 BLUE_V_CR_OFFSET=8		0x6208_0888

41.6.15 PrP Channel-1 Pixel Format Control Register

Figure 41-42 shows the register, and Table 41-51 provides its field descriptions.

0x1002_6038 (PRP_CH1_PIXEL_FORMAT_CNTL) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RED_WIDTH				GREEN_WIDTH				BLUE_WIDTH			
W																
Reset	0	0	0	0	0	1	0	1	0	1	1	0	0	1	0	1

Figure 41-42. PrP CH1 Pixel Format Control Register

Table 41-51. PrP CH1 Pixel Format Control Register Field Descriptions

Name	Description
31	Reserved. This bit is reserved and should read 0.
30–26 RED_OFFSET	RED_OFFSET. These bits set the offset of the Red color component in the output pixel. The offset is calculated with respect to Bit 0.

Table 41-51. PrP CH1 Pixel Format Control Register Field Descriptions (continued)

Name	Description
25–21 GREEN_OFFSET	GREEN_OFFSET. These bits set the offset of the Green color component in the output pixel. The offset is calculated with respect to Bit 0.
20–16 BLUE_OFFSET	BLUE_OFFSET. These bits set the offset of the Blue color component in the output pixel. The offset is calculated with respect to Bit 0.
15–12	Reserved. These bits are reserved and should read 0.
11–8 RED_WIDTH	RED_WIDTH. These bits set the width in bits of the Red color component in the input pixel. Valid values are 0 to 8. Any value greater than 8 is set to 8.
7–4 GREEN_WIDTH	GREEN_WIDTH. These bits set the width in bits of the Green color component in the input pixel. Valid values are 0 to 8. Any value greater than 8 is set to 8.
3–0 BLUE_WIDTH	BLUE_WIDTH. These bits set the width in bits of the Blue color component in the input pixel. Valid values are 0 to 8. Any value greater than 8 is set to 8.

Table 41-52 shows some example output RGB formats and encodings.

Table 41-52. Example Channel-1 RGB Output Pixel Format

Input Format	Pixel Arrangement	Offset	Width	PRP_CH1_PIXEL_FORMAT_CNTL
RGB 332	8 bpp	RED_OFFSET=5 GREEN_OFFSET=2 BLUE_OFFSET=0	RED_WIDTH=3 GREEN_WIDTH=3 BLUE_WIDTH=2	0x1440_0332
RGB 565	16 bpp	RED_OFFSET=11 GREEN_OFFSET=5 BLUE_OFFSET=0	RED_WIDTH=5 GREEN_WIDTH=6 BLUE_WIDTH=5	0x2CA0_0565
RGB 888	Unpacked RGB888	RED_OFFSET=16 GREEN_OFFSET=8 BLUE_OFFSET=0	RED_WIDTH=8 GREEN_WIDTH=8 BLUE_WIDTH=8	0x4100_0888
YUV 4:2:2	YUYV	RED_Y_OFFSET=24 GREEN_U_CB_OFFSET=16 BLUE_V_CR_OFFSET=0	RED_WIDTH=8 GREEN_WIDTH=8 BLUE_WIDTH=8	0x6200_0888
YUV 4:2:2	YVYU	RED_Y_OFFSET=24 GREEN_U_CB_OFFSET=0 BLUE_V_CR_OFFSET=16		0x6010_0888
YUV 4:2:2	UYVY	RED_Y_OFFSET=16 GREEN_U_CB_OFFSET=24 BLUE_V_CR_OFFSET=8		0x4308_0888
YUV 4:2:2	VYUY	RED_Y_OFFSET=16 GREEN_U_CB_OFFSET=8 BLUE_V_CR_OFFSET=24		0x4118_0888

41.6.16 PrP Destination Channel-1 Output Image Size Register

Figure 41-43 shows the register, and Table 41-53 provides its field descriptions.

0x1002_603C (PRP_CH1_OUT_IMAGE_SIZE) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	CH1_OUT_IMAGE_WIDTH										
W																
Reset	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	CH1_OUT_IMAGE_HEIGHT										
W																
Reset	0	0	0	0	0	1	0	0	1	1	1	1	0	0	0	0

Figure 41-43. PrP Destination Channel-1 Output Image Size Register

Table 41-53. PrP Destination Channel-1 Output Image Size Register Field Descriptions

Name	Description
31–27	Reserved. These bits are reserved and should read 0.
26–16 CH1_OUT_IMAGE_WIDTH	CH1_OUT_IMAGE_WIDTH. These bits sets the output width in number of pixels to display. See Section 41.5.3, “Resize” for more details. 8 bpp—this value should be a multiple of 4. 16 bpp or YUV 4:2:2—this value should be a multiple of 2
15–11	Reserved. These bits are reserved and should read 0.
10–0 CH1_OUT_IMAGE_HEIGHT	CH1_OUT_IMAGE_HEIGHT. These bits set the output height in number of lines to display. See Section 41.5.3, “Resize” for more details.

41.6.17 PrP Destination Channel-2 Output Image Size Register

Figure 41-44 shows the register, and Table 41-54 provides its field descriptions.

0x1002_6040 (PRP_CH2_OUT_IMAGE_SIZE) Access: User Read/Write

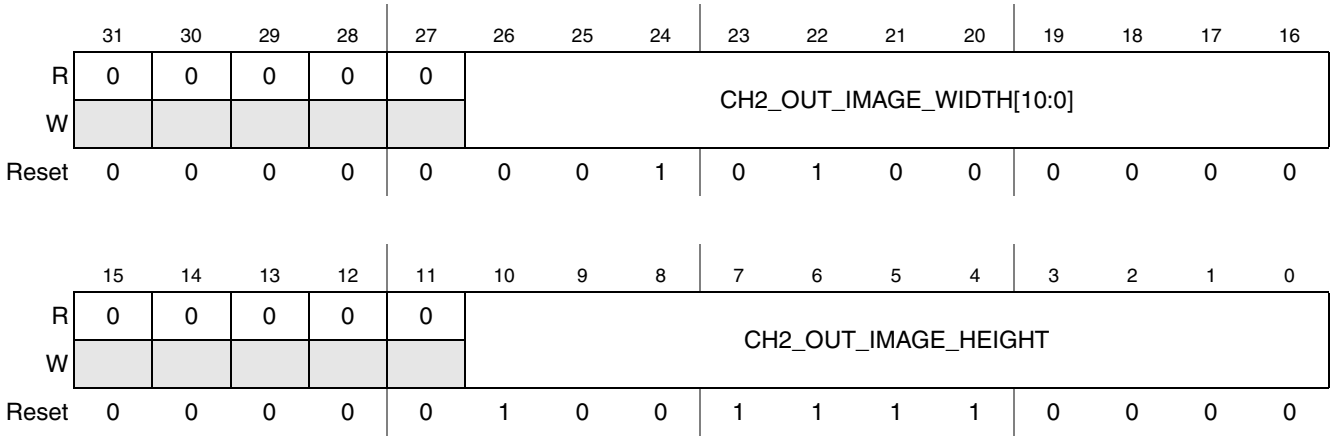


Figure 41-44. PrP Destination CH2 Output Image Size Register

Table 41-54. PrP Destination CH2 Output Image Size Register Field Description

Name	Description
31–27	Reserved. These bits are reserved and should read 0.
26–16 CH2_OUT_IMAGE_WIDTH[10:0]	Channel-2 Output Image Width. These bits sets the output width in number of pixels to display. YUV 422, YUV 444 - this value should be a multiple of 2. YUV 4:2:0 - this value should be a multiple of 8
16–11	Reserved. These bits are reserved and should read 0.
CH2_OUT_IMAGE_HEIGHT[1][10:0] 10–0	Channel 2 Output Image Height. These bits set the Channel-2 output image height in number of lines in display. In YUV 4:2:0 mode, number of Cr and Cb lines are CH2_OUT_IMAGE_HEIGHT/2. See Section 41.5.3, “Resize” for more details. In YUV 4:2:0 mode, this value should be in multiples of 2.

1

41.6.18 PrP Source Line Stride Register

Figure 41-45 shows the register, and Table 41-55 provides its field descriptions.

0x1002_6044 (PRP_SRC_LINE_STRIDE) Access: User Read/Write

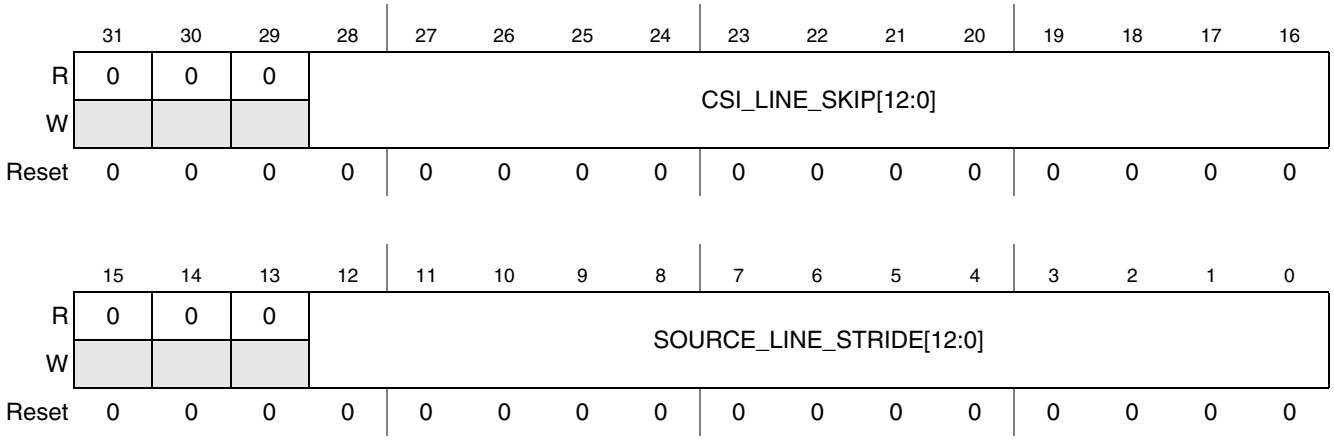


Figure 41-45. PrP Source Line Stride Register

Table 41-55. PrP Source Line Stride Register Field Descriptions

Name	Description
31–29	Reserved. These bits are reserved and should read 0
28–16 CSI_LINE_SKIP[12:0]	CSI_LINE_SKIP. These bits set the number lines to skip from start of a frame from CSI. Used only when WINEN=1 and CSIEN=1.
15–13	Reserved. These bits are reserved and should read 0
12–0 SOUCRE_LINE_STRIDE[12:0]	Source Line Stride. When CSIEN = 0, These bits set the line stride for the source frame in bytes. In YUV 4:2:0 mode, source line stride for ‘Cb’ and ‘Cr’ are SOURCE_LINE_STRIDE/2. When CSIEN = 1, then these bits sets number of pixels to skip from start of a line. It should be multiple of 2. YUV 4:2:0—Value should be a multiple of 8. YUV 4:2:2 or RGB—Value should be a multiple of 4.

41.6.19 PrP CSC Coefficient 012

Figure 41-46 shows the register, and Table 41-56 provides its field descriptions.

0x1002_6048 (PrP_CSC_COEF_012) Access: User Read/Write

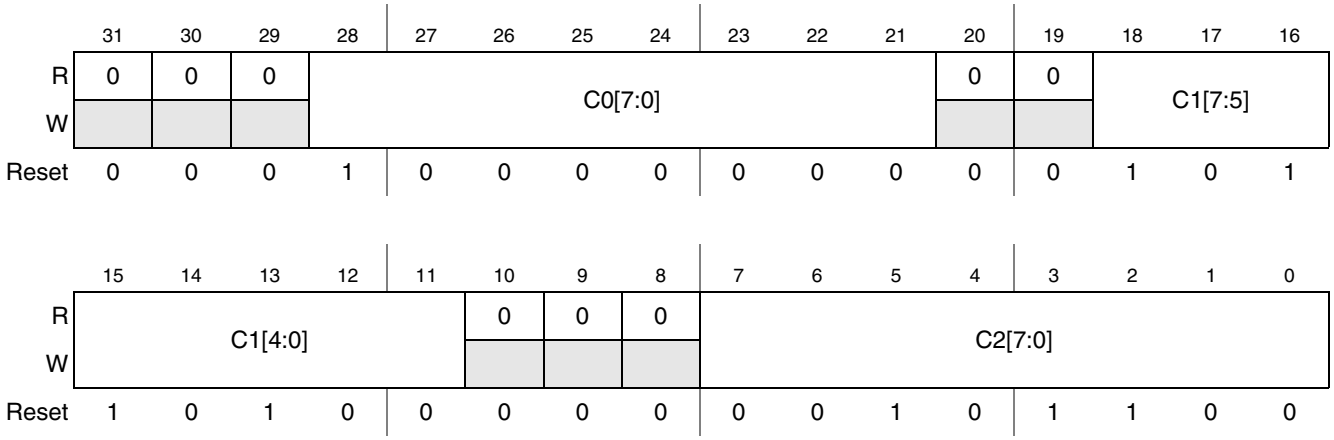


Figure 41-46. PrP CSC Coefficient 012

Table 41-56. YUV to RGB

Name	Description
31–29	Reserved. These bits are reserved and should read 0.
28–21 C0[7:0]	Coefficient 0. All 8 bits are significant Range from 0 to 1.9921875 in steps of (1/128)
20–19	Reserved. These bits are reserved and should read 0.
18–11 C1[7:0]	Coefficient 1. All 8 bits are significant. Range from 0 to 1.9921875 in steps of (1/128)
10–8	Reserved. These bits are reserved and should read 0.
7–0 C2[7:0]	Coefficient 2. All 8 bits are significant Range from 0 to 1.9921875 in steps of (1/128)

Table 41-57. For RGB to YUV

Name	Description
31–29	Reserved. These bits are reserved and should read 0
28–21 C0[7:0]	Coefficient 0. Only C0[6:0], 7 bits are significant Range from 0 to 0.49603750 in steps of (1/256)
20–19	Reserved. These bits are reserved and should read 0
18–11 C1[7:0]	Coefficient 1. Only C1[6:0], 7 bits are significant Range from 0 to 0.9921875 in steps of (1/128)
10–8	Reserved. These bits are reserved and should read 0
7–0 C2[7:0]	Coefficient 2. Only C2[6:0], 7 bits are significant Range from 0 to 0.248046875 in steps of (1/512)

41.6.20 PrP CSC Coefficient 345

Figure 41-47 shows the register, and Table 41-58 provides its field descriptions.

0x1002_604C (PrP_CSC_COEF_345) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	C3[7:0]								0	0	C4[8:5]		
W																
Reset	0	0	0	0	1	0	1	1	0	1	1	0	0	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	C4[4:0]					0	0	0	0	C5[6:0]						
W																
Reset	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Figure 41-47. PrP CSC Coefficient 345

Table 41-58. For YUV to RGB

Name	Description
31–29	Reserved. These bits are reserved and should read 0
28–21 C3[7:0]	Coefficient 3. All 8 bits are significant Range from 0 to 1.9921875 in steps of (1/128)
20	Reserved. This bit is reserved and should read 0
19–11 C4[8:0]	Coefficient 4. All 8 bits are significant Range from 0 to 3.9921875 in steps of (1/128)
10–7	Reserved. These bits are reserved and should read 0
6–0 C5[6:0]	This coefficient is unused in YUV to RGB conversion.

Table 41-59. For RGB to YUV

Name	Description
31–29	Reserved. These bits are reserved and should read 0
28–21 C3[7:0]	Coefficient 3. Only C3[6:0], 7 bits are significant. Range from 0.0 to 0.248046875 in steps of (1/512)
20	Reserved. This bit is reserved and should read 0
19–11 C4[8:0]	Coefficient 4. Only C4[6:0], 7 bits are significant. Range from 0 to 0.49603750 in steps of (1/256)
10–7	Reserved. These bits are reserved and should read 0
6–0 C5[6:0]	Coefficient 5. Only C5[6:0], 7 bits are significant. Range from 0 to 0.9921875in steps of (1/128)

41.6.21 PrP CSC Coefficient 678

Figure 41-48 shows the register, and Table 41-60 provides its field descriptions.

0x1002_6050 (PrP_CSC_COEF_678) Access: User Read/Write

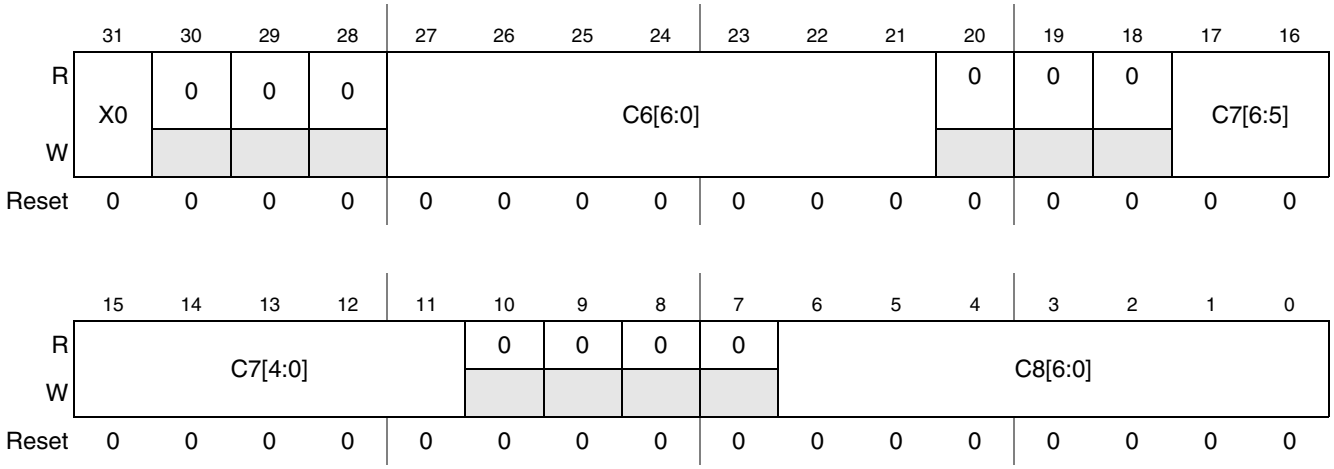


Figure 41-48. PrP CSC Coefficient 678

Table 41-60. For RGB to YUV

Name	Description
31 X0	X0. Luminance offset 1 X0 is equal to 16 in CSC. 0 X0 is zero in CSC.
30– 28	Reserved. These bits are reserved and should read 0.
27–21 C6[7:0]	Coefficient 6. All 7 bits are significant. Range from 0.0 to 0.9921875 in steps of (1/128)
20–18 Reserved	Reserved. These bits are reserved and should read 0.
17–11 C7[8:0]	Coefficient 7. All 7 bits are significant. Range from 0 to 0.49603750 in steps of (1/256)
10–7	Reserved. These bits are reserved and should read 0.
6–0 C8[6:0]	Coefficient 8. All 7 bits are significant. Range from 0 to 0.248046875 in steps of (1/512)

Table 41-61. CSC Equations

Conversion	CSC Equation
YCbCr to RGB	R = C0*(Y – X0) + C1*(Cr-128) G = C0*(Y – X0) – C2*(Cb-128) – C3*(Cr-128) B = C0*(Y – X0) + C4*(Cb-128)
YUV to RGB	R = C0*(Y – X0) + C1*(U-128) G = C0*(Y – X0) – C2*(U-128) – C3*(V-128) B = C0*(Y – X0) + C4*(U-128)
RGB to YUV	Y = C0 * R + C1 * G + C2 * B + X0 U = -C3 * R–C4 * G + C5 * B + 128 V = C6 * R–C7 * G –C8 * B + 128

41.6.22 PrP Channel 1 Horizontal Resize Coefficient-1

Figure 41-49 shows the register, and Table 41-62 provides its field descriptions.

0x1002_6054 (PrP_CH1_RZ_HORI_COEF1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	HC9			HC8				HC7				HC6			
W														HC5		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	HC4			HC3				HC2				HC1			
W														HC0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 41-49. PrP Channel-1 Horizontal Resize Coefficient 1

This register selects Channel-1 Horizontal resize coefficient values. The coefficient is 3-bits in width which represent value from 0 to 8. If HCx = 7 then it will treat as ‘8’.

- For bi-linear mode (w0*a+ w1*b)/8. Coefficient w0 is programmed.
- For m:n resize ratio number of coefficients will be “m”.

Table 41-62. PrP Channel 1 Horizontal Resize Coefficient-1 Register Field Descriptions

Name	Description
31	Reserved. This bit is reserved and should read 0.
30–28 HC9[2:0]	Horizontal Resize Coefficient 9. Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
27–25 HC8[2:0]	Horizontal Resize Coefficient 8. Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
24–22 HC7[2:0]	Horizontal Resize Coefficient 7. Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
21–19 HC6[2:0]	Horizontal Resize Coefficient 6. Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
18–16 HC5[2:0]	Horizontal Resize Coefficient 5. Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
15	Reserved. This bit is reserved and should read 0
14–12 HC4[2:0]	Horizontal Resize Coefficient 4. Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
11–9 HC3[2:0]	Horizontal Resize Coefficient 3. alid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
8–6 HC2[2:0]	Horizontal Resize Coefficient 2. Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.

Table 41-62. PrP Channel 1 Horizontal Resize Coefficient-1 Register Field Descriptions (continued)

Name	Description
5–3 HC1[2:0]	Horizontal Resize Coefficient 1. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
2–0 HC0[2:0]	Horizontal Resize Coefficient 0. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.

41.6.23 PrP Channel1 Horizontal Resize Coefficient-2

Figure 41-50 shows the register, and Table 41-63 provides its field descriptions.

0x1002_6058 (PrP_CH1_RZ_HORI_COEF2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	HC19			HC18				HC17				HC16			
W															HC15	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	HC14			HC13				HC12				HC11			
W															HC10	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-50. PrP Channel-1 Horizontal Resize Coefficient 2

This register selects Channel-1 Horizontal resize coefficient values. The coefficient is 3-bits in width which represent value from 0 to 8. If HCx = 7 then it will treat as '8'.

- For bi-linear mode (w0*a+ w1*b)/8. Coefficient w0 is programmed.
- For m:n resize ratio number of coefficients will be "m".

Table 41-63. PrP Channel1 Horizontal Resize Coefficient-2 Register Field Descriptions

Name	Description Settings
31	Reserved. This bit is reserved and should read 0.
30–28 HC10[2:0]	Horizontal Resize Coefficient 10. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
27–25 HC11[2:0]	Horizontal Resize Coefficient 11. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
24–22 HC12[2:0]	Horizontal Resize Coefficient 12. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
21–19 HC13[2:0]	Horizontal Resize Coefficient 13. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.

Table 41-63. PrP Channel1 Horizontal Resize Coefficient-2 Register Field Descriptions (continued)

Name	Description Settings
18–16 HC14[2:0]	Horizontal Resize Coefficient 14. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
15	Reserved. This bit is reserved and should read 0.
14–12 HC15[2:0]	Horizontal Resize Coefficient 15. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
11–9 HC16[2:0]	Horizontal Resize Coefficient 16. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
8–6 HC17[2:0]	Horizontal Resize Coefficient 17. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
5–3 HC18[2:0]	Horizontal Resize Coefficient 18. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
2–0 HC19[2:0]	Horizontal Resize Coefficient 19. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.

41.6.24 PrP Channel 1 Horizontal Resize Valid

Figure 41-51 shows the register, and Table 41-64 provides its field descriptions.

0x1002_605C (PrP_CH1_RZ_HORI_VALID) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	AVG_BIL	0	0	HORI_TBL_LEN					0	0	0	0	HOV[19:16]			
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HOV[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 41-51. PrP Channel-1 Resize Horizontal Valid

Table 41-64. PrP Channel 1 Horizontal Resize Valid Register Field Descriptions

Name	Description
31 AVG_BIL	Averaging or bilinear mode select 0 Averaging 1 Bi-linear
30–29	Reserved. These bits are reserved and should read 0.

Table 41-64. PrP Channel 1 Horizontal Resize Valid Register Field Descriptions (continued)

Name	Description
28–24 HORI_TBL_LEN	Horizontal Resize Table Length. Selects horizontal resize table length For M:N resize ratio for both bilinear and averaging mode table length should be set to “M”. Range from 1 to 20.
23–20	Reserved. These bits are reserved and should read 0.
19–0 HOV[19:0]	Horizontal Output Valid. Bit vector that selects when to output pixel. Skips output pixels when averaging and input pixels when bi-linear. 0 Pixel is not output. 1 Pixel is output.

41.6.25 PrP Channel1 Vertical Resize Coefficient-1

Figure 41-52 shows the register, and Table 41-65 provides its field descriptions.

0x1002_6060 (PrP_CH1_RZ_VERT_COEF1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	VC9			VC8				VC7				VC6			
W																VC5
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	VC4			VC3				VC2				VC1			
W																VC0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 41-52. PrP Channel 1 Vertical Resize Coefficient 1

This register selects Channel-1 vertical resize coefficient values. The coefficient is 3-bits in width which represent values from 0 to 8. If VCx = 7 then it will treat it as ‘8’.

- For bi-linear mode (w0*a+ w1*b)/8. Coefficient w0 is programmed.
- For m:n resize ratio number of coefficients will be “m”.

Table 41-65. PrP Channel1 Vertical Resize Coefficient-1 Register Field Descriptions

Name	Description
31	Reserved. This bit is reserved and should read 0.
30–28 VC9[2:0]	Vertical Resize Coefficient 9. Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
27–25 VC8[2:0]	Vertical Resize Coefficient 8. Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.

Table 41-65. PrP Channel1 Vertical Resize Coefficient-1 Register Field Descriptions (continued)

Name	Description
24–22 VC7[2:0]	Vertical Resize Coefficient 7. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
21–19 VC6[2:0]	Vertical Resize Coefficient 6. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
18–16 VC5[2:0]	Vertical Resize Coefficient 5. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
15	Reserved. This bit is reserved and should read 0.
14–12 VC4[2:0]	Vertical Resize Coefficient 4. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
11–9 VC3[2:0]	Vertical Resize Coefficient 3. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
8–6 VC2[2:0]	Vertical Resize Coefficient 2. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
5–3 VC1[2:0]	Vertical Resize Coefficient 1. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
2–0 VC0[2:0]	Vertical Resize Coefficient 0. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.

41.6.26 PrP Channel 1 Vertical Resize Coefficient 2

Figure 41-53 shows the register, and Table 41-66 provides its field descriptions.

0x1002_6064 (PrP_CH1_RZ_VERT_COEF2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	VC19			VC18				VC17				VC16			
W																VC15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	VC14			VC13				VC12				VC11			
W																VC10
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-53. PrP Channel-1 Vertical Resize Coefficient 2

This register selects Channel-1 Vertical resize coefficient values. The coefficient is 3-bits in width which represent values from 0 to 8. If VCx = 7 then it is treated as '8'.

- For bi-linear mode (w0*a+ w1*b)/8. Coefficient w0 is programmed.
- For m:n resize ratio number of coefficients will be "m".

Table 41-66. PrP Channel 1 Vertical Resize Coefficient 2 Register Field Descriptions

Name	Description
31	Reserved. This bit is reserved and should read 0.
30–28 VC10[2:0]	Vertical Resize Coefficient 10. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
27–25 VC11[2:0]	Vertical Resize Coefficient 11. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
24–22 VC12[2:0]	Vertical Resize Coefficient 12. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
21–19 VC13[2:0]	Vertical Resize Coefficient 13. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
18–16 VC14[2:0]	Vertical Resize Coefficient 14. Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
15	Reserved. This bit is reserved and should read 0.
14–12 VC15[2:0]	Vertical Resize Coefficient 15 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
11–9 VC16[2:0]	Vertical Resize Coefficient 16 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
8–6 VC17[2:0]	Vertical Resize Coefficient 17 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
5–3 VC18[2:0]	Vertical Resize Coefficient 18 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
2–0 VC19[2:0]	Vertical Resize Coefficient 19 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.

41.6.27 PrP Channel 1 Vertical Resize Valid

Figure 41-54 shows the register, and Table 41-67 provides its field descriptions.

0x1002_6068 (PrP_CH1_RZ_VERT_VALID) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	AVG_BIL	0	0	VERT_TBL_LEN					0	0	0	0	VOV[19:16]			
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VOV[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 41-54. PrP Channel1 Vertical Resize Valid

Table 41-67. PrP Channel 1 Vertical Resize Valid Register Field Descriptions

Name	Description
31 AVG_BIL	Averaging or bilinear mode select 0 Averaging 1 Bi-linear
30–29	Reserved. These bits are reserved and should read 0.
28–24 VERT_TBL_LEN	Vertical Resize Table Length. Selects vertical resize table length. For M:N resize ratio for both bilinear and averaging mode table length should be set to “M”. Range from 1 to 20.
23–20	Reserved. These bits are reserved and should read 0.
19–0 VOV[19:0]	Vertical Output Valid. Bit vector that selects when to output pixel. Skips output pixels when averaging and input pixels when bi-linear. 0 Pixel is not output. 1 Pixel is output.

41.6.28 PrP Channel-2 Horizontal Resize Coefficient-1

Figure 41-55 shows the register, and Table 41-68 provides its field descriptions.

0x1002_606C (PrP_CH2_RZ_HORI_COEF1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	HC9			HC8				HC7				HC6			
W															HC5	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	HC4			HC3				HC2				HC1			
W															HC0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 41-55. PrP Channel-2 Horizontal Resize Coefficient 1

This register selects Channel-2 Horizontal resize coefficient values. The coefficient is 3-bits in width which represent value from 0 to 8. If HCx = 7 then it will treat as ‘8’.

- For bi-linear mode (w0*a+ w1*b)/8. Coefficient w0 is programmed.
- For m:n resize ratio number of coefficients will be “m”.

Table 41-68. PrP Channel-2 Horizontal Resize Coefficient-1 Register Field Descriptions

Name	Description
31	Reserved. This bit is reserved and should read 0
30–28 HC9[2:0]	Horizontal Resize Coefficient 9 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
27–25 HC8[2:0]	Horizontal Resize Coefficient 8 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
24–22 HC7[2:0]	Horizontal Resize Coefficient 7 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
21–19 HC6[2:0]	Horizontal Resize Coefficient 6 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
18–16 HC5[2:0]	Horizontal Resize Coefficient 5 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
15	Reserved. This bit is reserved and should read 0
14–12 HC4[2:0]	Horizontal Resize Coefficient 4 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
11–9 HC3[2:0]	Horizontal Resize Coefficient 3 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
8–6 HC2[2:0]	Horizontal Resize Coefficient 2 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
5–3 HC1[2:0]	Horizontal Resize Coefficient 1 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
2–0 HC0[2:0]	Horizontal Resize Coefficient 0 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.

41.6.29 PrP Channel-2 Horizontal Resize Coefficient-2

Figure 41-56 shows the register, and Table 41-69 provides its field descriptions.

0x1002_6070 (PrP_CH2_RZ_HORI_COEF2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	HC19			HC18				HC17				HC16			
W															HC15	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	HC14			HC13				HC12				HC11			
W															HC10	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-56. PrP Channel-2 Horizontal Resize Coefficient 2

This register selects Channel-2 Horizontal resize coefficient values. The coefficient is 3-bits in width which represent value from 0 to 8. If HCx = 7 then it will be treated as ‘8’.

- For bi-linear mode (w0*a+ w1*b)/8. Coefficient w0 is programmed.
- For m:n resize ratio number of coefficients will be “m”.

Table 41-69. PrP Channel-2 Horizontal Resize Coefficient-2 Field Descriptions

Name	Description
31	Reserved. This bit is reserved and should read 0
30–28 HC10[2:0]	Horizontal Resize Coefficient 10 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
27–25 HC11[2:0]	Horizontal Resize Coefficient 11 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
24–22 HC12[2:0]	Horizontal Resize Coefficient 12 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
21–19 HC13[2:0]	Horizontal Resize Coefficient 13 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
18–16 HC14[2:0]	Horizontal Resize Coefficient 14 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
15	Reserved. This bit is reserved and should read 0
14–12 HC15[2:0]	Horizontal Resize Coefficient 15 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
11–9 HC16[2:0]	Horizontal Resize Coefficient 16 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
8–6 HC17[2:0]	Horizontal Resize Coefficient 17 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
5–3 HC18[2:0]	Horizontal Resize Coefficient 18 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
2–0 HC19[2:0]	Horizontal Resize Coefficient 19 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.

41.6.30 PrP Channel-2 Horizontal Resize Valid

Figure 41-57 shows the register, and Table 41-70 provides its field descriptions.

0x1002_6074 (PrP_CH2_RZ_HORI_VALID) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	AVG_BIL	0	0	HORI_TBL_LEN					0	0	0	0	HOV[19:16]			
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HOV[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 41-57. PrP Channel-2 Resize Horizontal Valid

Table 41-70. PrP Channel-2 Horizontal Resize Valid Register Field Descriptions

Name	Description
31 AVG_BIL	Averaging or bilinear mode select 0 Averaging 1 Bi-linear
30–29	Reserved. These bits are reserved and should read 0
28–24 HORI_TBL_LEN	Horizontal Resize Table Length. Selects horizontal resize table length For M:N resize ratio for both bilinear and averaging mode table length should be set to “M”. Range from 1 to 20.
23–20	Reserved. These bits are reserved and should read 0
19–0 HOV[19:0]	Horizontal Output Valid. Bit vector that selects when to output pixel. Skips output pixels when averaging and input pixels when bi-linear. 0 Pixel is not output 1 Pixel is output

41.6.31 PrP Channel2 Vertical Resize Coefficient-1

Figure 41-58 shows the register, and Table 41-71 provides its field descriptions.

0x1002_6078 (PrP_CH2_RZ_VERT_COEF1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	VC9			VC8			VC7			VC6		VC5			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	VC4			VC3			VC2			VC1		VC0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 41-58. PrP Channel2 Vertical Resize Coefficient 1

This register selects Channel-2 vertical resize coefficient values. The coefficient is 3-bits in width which represent values from 0 to 8. If VCx = 7 then it will treat it as ‘8’.

- For bi-linear mode (w0*a+ w1*b)/8. Coefficient w0 is programmed.
- For m:n resize ratio number of coefficients will be “m”.

Table 41-71. PrP Channel2 Vertical Resize Coefficient-1 Register Field Descriptions

Name	Description Settings
31	Reserved. This bit is reserved and should read 0
30–28 VC9[2:0]	Vertical Resize Coefficient 9 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
27–25 VC8[2:0]	Vertical Resize Coefficient 8 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
24–22 VC7[2:0]	Vertical Resize Coefficient 7 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
21–19 VC6[2:0]	Vertical Resize Coefficient 6 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
18–16 VC5[2:0]	Vertical Resize Coefficient 5 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
15	Reserved. This bit is reserved and should read 0
14–12 VC4[2:0]	Vertical Resize Coefficient 4 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
11–9 VC3[2:0]	Vertical Resize Coefficient 3 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.
8–6 VC2[2:0]	Vertical Resize Coefficient 2 Valid values are 0–6 and 8. To set a value of 8, use 3’b111 instead.

Table 41-71. PrP Channel2 Vertical Resize Coefficient-1 Register Field Descriptions (continued)

Name	Description Settings
5–3 VC1[2:0]	Vertical Resize Coefficient 1 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
2–0 VC0[2:0]	Vertical Resize Coefficient 0 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.

41.6.32 PrP Channel 2 Vertical Resize Coefficient 2

Figure 41-59 shows the register, and Table 41-72 provides its field descriptions.

0x1002_607C (PrP_CH2_RZ_VERT_COEF2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	VC19			VC18				VC17				VC16			
W															VC15	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	VC14			VC13				VC12				VC11			
W															VC10	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-59. PrP Channel-2 Vertical Resize Coefficient 2

This register selects Channel-2 Vertical resize coefficient values. The coefficient is 3-bits in width which represent values from 0 to 8. If VCx = 7 then it is treated as '8'.

- For bi-linear mode (w0*a+ w1*b)/8. Coefficient w0 is programmed.
- For m:n resize ratio number of coefficients will be "m".

Table 41-72. PrP Channel 2 Vertical Resize Coefficient 2 Register Field Descriptions

Name	Description
31	Reserved. This bit is reserved and should read 0
30–28 VC10[2:0]	Vertical Resize Coefficient 10 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
27–25 VC11[2:0]	Vertical Resize Coefficient 11 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
24–22 VC12[2:0]	Vertical Resize Coefficient 12 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
21–19 VC13[2:0]	Vertical Resize Coefficient 13 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.

Table 41-72. PrP Channel 2 Vertical Resize Coefficient 2 Register Field Descriptions (continued)

Name	Description
18–16 VC14[2:0]	Vertical Resize Coefficient 14 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
15	Reserved. This bit is reserved and should read 0
14–12 VC15[2:0]	Vertical Resize Coefficient 15 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
11–9 VC16[2:0]	Vertical Resize Coefficient 16 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
8–6 VC17[2:0]	Vertical Resize Coefficient 17 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
5–3 VC18[2:0]	Vertical Resize Coefficient 18 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.
2–0 VC19[2:0]	Vertical Resize Coefficient 19 Valid values are 0–6 and 8. To set a value of 8, use 3'b111 instead.

41.6.33 PrP Channel 2 Vertical Resize Valid

Figure 41-60 shows the register, and Table 41-73 provides its field descriptions.

0x1002_6080 (PrP_CH2_RZ_VERT_VALID) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	AVG_BIL	0	0	VERT_TBL_LEN					0	0	0	0	VOV[19:16]			
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VOV[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 41-60. PrP Channel2 Vertical Resize Valid

Table 41-73. PrP Channel 2 Vertical Resize Valid Register Field Descriptions

Name	Description
31 AVG_BIL	Averaging or bilinear mode select 0 Averaging 1 Bi-linear
30–29	Reserved. These bits are reserved and should read 0

Table 41-73. PrP Channel 2 Vertical Resize Valid Register Field Descriptions (continued)

Name	Description
28–24 VERT_TBL_LEN	Vertical Resize Table Length. Selects vertical resize table length For M:N resize ratio for both bilinear and averaging mode table length should be set to “M”. Range from 1 to 20.
23–20	Reserved. These bits are reserved and should read 0
19–0 VOV[19:0]	Vertical Output Valid. Bit vector that selects when to output pixel. Skips output pixels when averaging and input pixels when bi-linear. 0 Pixel is not output. 1 Pixel is output.

Chapter 42
Synchronous Serial Interface (SSI)

This chapter presents the Synchronous Serial Interface (SSI), and discusses the architecture, the programming model, the operating modes, and initialization of SSI.

Figure 42-1 shows a block diagram of the SSI. It consists of control registers to set up the port, status register, separate transmit and receive circuits with FIFO registers, and separate serial clock and frame sync generation for the transmit and receive sections. The second set of Tx and Rx FIFOs, replicates the logic used for the first set of FIFOs.

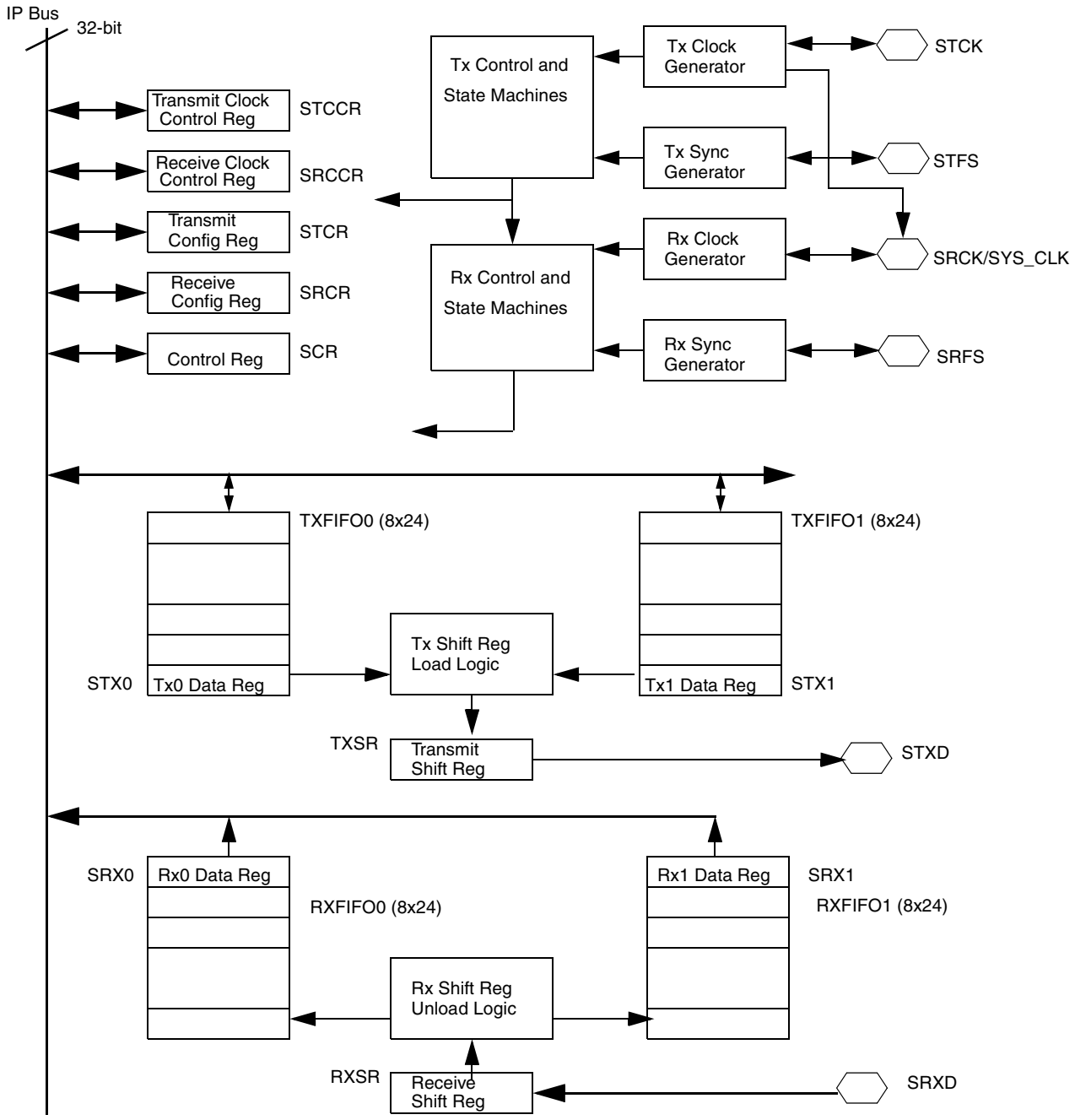


Figure 42-1. SSI Block Diagram

42.1 Overview

The SSI is a full-duplex, serial port that allows the chip to communicate with a variety of serial devices. These serial devices can be standard CODer-DECoder (CODEC), Digital Signal Processors (DSPs), microprocessors, peripherals, and popular industry audio CODECs that implement the inter-IC sound bus standard (I^2S) and Intel AC97 standard.

SSI is typically used to transfer samples in a periodic manner. The SSI consists of independent transmitter and receiver sections with independent clock generation and frame synchronization.

42.1.1 Features

The SSI includes the following features:

- Independent (asynchronous) or shared (synchronous) transmit and receive sections with separate or shared internal/external clocks and frame syncs, operating in Master or Slave mode.
- Normal mode operation using frame sync
- Network mode operation allowing multiple devices to share the port with as many as thirty-two time slots
- Gated Clock mode operation requiring no frame sync
- 2 sets of Transmit and Receive FIFOs. Each of the four FIFOs is 8x24 bits. The two sets of Tx/Rx FIFOs can be used in Network mode to provide 2 independent channels for transmission and reception
- Programmable data interface modes such like I^2S, LSB, MSB aligned
- Programmable word length (8, 10, 12, 16, 18, 20, 22 or 24 bits)
- Program options for frame sync and clock generation
- Programmable I^2S modes (Master, Slave or Normal). Oversampling clock, ccm_ssi_clk available as output from SRCK in I^2S Master mode
- AC97 support
- Completely separate clock and frame sync selections for the receive and transmit sections. In AC97 standard, the clock is taken from an external source and frame sync is generated internally.
- External ccm_ssi_clk input for use in I^2S Master mode. Programmable oversampling clock (SYS_CLK/ccm_ssi_clk) of the sampling frequency available as output in master mode at SRCK, when operated in sync mode.
- Programmable internal clock divider
- Time Slot Mask Registers for reduced CPU overhead (for Tx and Rx both)
- SSI power-down feature
- Programmable wait states for CPU accesses
- IP Interface for register accesses, compliant to SRS 3.0.2 standard

42.1.2 Modes of Operation

SSI has the following basic operating modes.

- Normal mode
  - — Asynchronous protocol
  - — Synchronous protocol
- Network mode
  - — Asynchronous protocol

  - Synchronous protocol
- Gated Clock mode
  - Synchronous protocol only

These modes can be programmed by several bits in the SSI control registers. Table 42-1 lists these operating modes and some of the typical applications in which they can be used.

Table 42-1. SSI Operating Modes

TX, RX Sections	Serial Clock	Mode	Typical Application
Asynchronous	Continuous	Normal	Multiple synchronous CODECs
Asynchronous	Continuous	Network	TDM CODEC or DSP networks
Synchronous	Continuous	Normal	Multiple synchronous CODECs
Synchronous	Continuous	Network	TDM CODEC or DSP network
Synchronous	Gated	Normal	SPI-type devices; DSP to MCU

The transmit and receive sections of the SSI can be synchronous or asynchronous. In Synchronous mode, the transmitter and the receiver use a common clock and frame synchronization signal. The RXBIT0 and RSHFD bits in SRCR still affect shifting-in of received data in synchronous mode. In Asynchronous mode, the transmitter and receiver each has its own clock and frame synchronization signals. Continuous or Gated Clock mode can be selected. In Continuous mode, the clock runs continuously. In Gated Clock mode, the clock is only functioning during transmission.

Normal or Network mode can also be selected. In Normal mode, the SSI functions with one data word of I/O per frame. In Network mode, any number from two to thirty-two data words of I/O per frame can be used. Network mode is typically used in star or ring time division multiplex networks with other processors or CODECs, allowing interface to time division multiplexed networks without additional logic. Use of the gated clock is not allowed in Network mode. These distinctions result in the basic operating modes that allow the SSI to communicate with a wide variety of devices.

The SSI supports both Normal and Network modes, and these can be selected independently of whether the transmitter and receiver are synchronous or asynchronous. Typically these protocols are used in a periodic manner, where data is transferred at regular intervals, such as at the sampling rate of an external CODEC. Both modes use the concept of a frame. The beginning of the frame is marked with a frame sync when programmed with continuous clock. The frame sync occurs at a periodic interval. The length of the frame is determined by the DC[4:0] bits in either the SRCCR or STCCR register, depending on whether data is being transmitted or received. The number of words transferred per frame depends on the mode of the SSI.

In Normal mode, one data word is transferred per frame. In Network mode, the frame is divided into anywhere between two and thirty-two time slots, where in each time slot one data word can optionally be transferred.

Apart from the above basic modes of operation, SSI supports the following modes which require some specific programming.

- I^2S mode
- AC97 mode
  - — AC97 Fixed mode
  - — AC97 Variable mode

In (non-I^2S) slave modes (external frame sync), the SSI programmed word length setting should be equal to the word length setting of the master. In I^2S slave mode, the SSI programmed word length setting can be lesser than or equal to the word length setting of the I^2S master (external CODEC).

In slave modes, the SSI programmed frame length setting (DC bits) can be less than or equal to the frame length setting of the master (external CODEC).

The following sections provide detailed descriptions of the above modes.

42.1.2.1 Normal Mode

Normal mode is the simplest mode of the SSI. It is used to transfer data in one time slot per frame. A time slot is a unit of data and the WL[3:0] bits define the number of bits in a time slot. In Continuous Clock mode, a frame sync occurs at the beginning of each frame. The length of the frame is determined by the following factors:

- The period of the Serial Bit Clock (DIV2, PSR, PM[7:0] bits for internal clock or the frequency of the external clock on the STCK port)
- The number of bits per time slot (WL[3:0] bits)
- The number of time slots per frame (DC[4:0] bits)

If Normal mode is configured with more than one time slot per frame, data is transferred only in the first time slot. No data is transferred in subsequent time slots. In Normal mode, DC[4:0] values corresponding to more than a single time slot in a frame, only result in lengthening the frame. Data transfer only takes place during the first time slot of the frame.

42.1.2.1.1 Normal Mode Transmit

The conditions for data transmission from the SSI in Normal mode are:

1. SSI enabled (SSIEN = 1)
2. Enable FIFO and configure Transmit and Receive Watermark if FIFO is used.
3. Write data to Transmit Data Register (STX)
4. Transmitter enabled (TE = 1)
5. Frame sync active (for continuous clock case)
6. Bit clock begins (for gated clock case)

When the above conditions occur in Normal mode, the next data word is transferred into the Transmit Shift Register (TXSR) from the Transmit Data Register 0 (STX0), or from the Transmit FIFO 0 Register, if transmit FIFO 0 is enabled. The new data word is transmitted immediately.

If transmit FIFO 0 is not enabled and the transmit data register empty (TDE0) bit is set, transmit interrupt 0 occurs if the transmit interrupt enable (TIE) and TDE0_EN bits are set.

The Transmit FIFO Empty 0 (TFE0) bit is set if the Transmit FIFO 0 reaches the selected threshold. If transmit FIFO 0 is enabled and the Transmit FIFO Empty (TFE0) bit is set, transmit interrupt 0 occurs if the transmit interrupt enable (TIE) and TFE0_EN bits are set. If transmit FIFO 0 is enabled and filled with data, 8 data words can be transferred before the core must write new data to the STX0 register.

The STXD port is disabled except during the data transmission period. For a continuous clock, the optional frame sync output and clock outputs are not disabled, even if both receiver and transmitter are disabled.

42.1.2.1.2 Normal Mode Receive

The conditions for data reception from the SSI are:

1. SSI enabled (SSIEN = 1)
2. Receiver enabled (RE = 1)
3. Frame sync active (for continuous clock case)
4. Bit clock begins (for gated clock case)

With the above conditions in Normal mode with a continuous clock, each time the frame sync signal is generated (or detected) a data word is clocked in. With the above conditions and a gated clock, each time the clock begins, a data word is clocked in.

If receive FIFO 0 is not enabled, the received data word is transferred from the Receive Shift Register (RXSR) to the Receive Data Register 0 (SRX0), the Receive Data Ready 0 (RDR0) flag is set. Receive Interrupt 0 occurs if RIE and RDR0_EN bits are set.

If receive FIFO 0 is enabled, the received data word is transferred to the Receive FIFO 0. The Receive FIFO Full 0 (RFF0) flag is set if the Receive Data Register (SRX0) is full and Receive FIFO 0 reaches the selected threshold. Receive Interrupt 0 occurs if Receive Interrupt Enable (RIE) and RFF0_EN bits are set.

The core program has to read the data from the Receive Data Register 0 (SRX0) before a new data word is transferred from the Receive Shift Register (RXSR), otherwise the Receive Overrun Error 0 (ROE0) bit is set. If receive FIFO 0 is enabled, the Receive Overrun Error 0 (ROE0) bit is set when the Receive FIFO 0 data level reaches the selected threshold and a new data word is ready to be transferred to the Receive FIFO 0.

Figure 42-2 shows transmitter and receiver timing for an 8-bit word in the first time slot in Normal mode, continuous clock with a late word length frame sync. The Tx Data register is loaded with the data to be transmitted. On arrival of the frame sync, this data is transferred to the Transmit Shift Register and transmitted on the STXD output. Simultaneously, the Receive Shift Register shifts in the received data available on the SRXD input and at the end of the time slot, this data is transferred to the Rx Data Register.

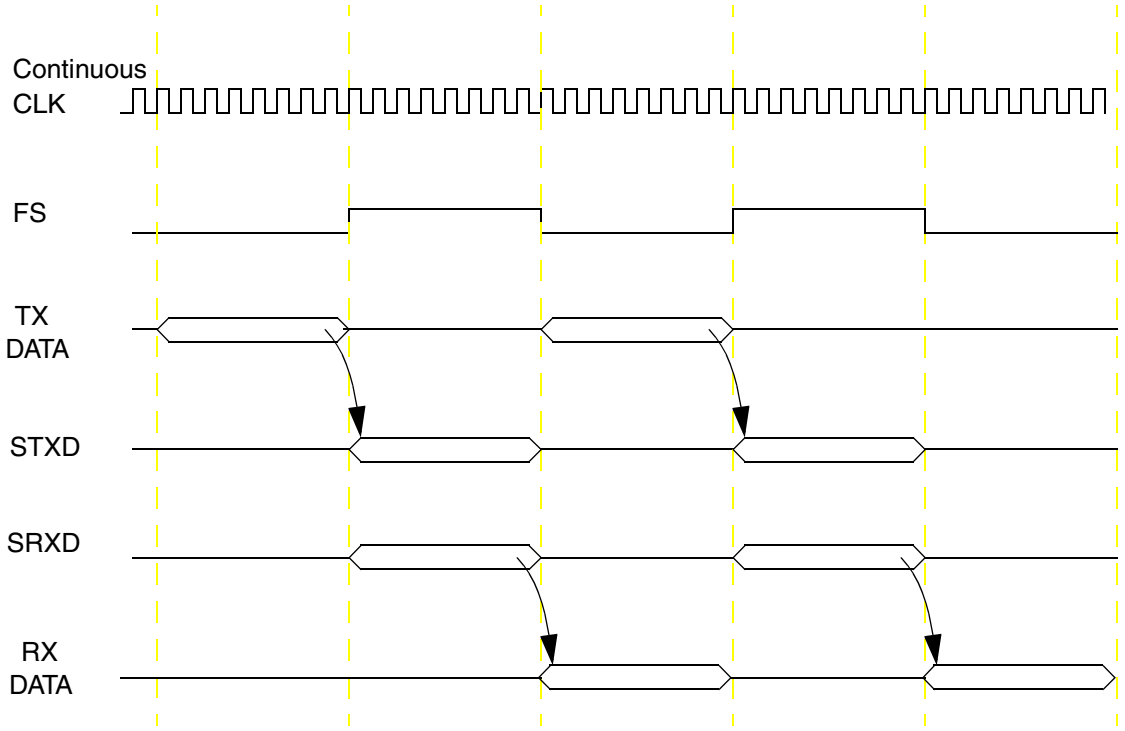


Figure 42-2. Normal Mode Timing—Continuous Clock

Figure 42-3 shows a similar case for internal (SSI generates clock) gated clock mode and Figure 42-4 shows a case for external (SSI receives clock) gated clock mode.

NOTE

A pull-down resistor is required in the gated clock case because the clock port is disabled between transmissions.

The Tx Data register is loaded with the data to be transmitted. On arrival of the clock, this data is transferred to the Transmit Shift Register and transmitted on the STXD output. Simultaneously, the Receive Shift Register shifts in the received data available on the SRXD input and at the end of the time slot, this data is transferred to the Rx Data Register. In case of Internal Gated clock mode, the Tx Data line and clock output port are put in the high-impedance state at the end of transmission of the last bit (at the completion of the complete clock cycle), whereas, in External Gated clock mode, the Tx Data line is tri-stated at the last inactive edge of the incoming bit clock (during the last bit in a data word).

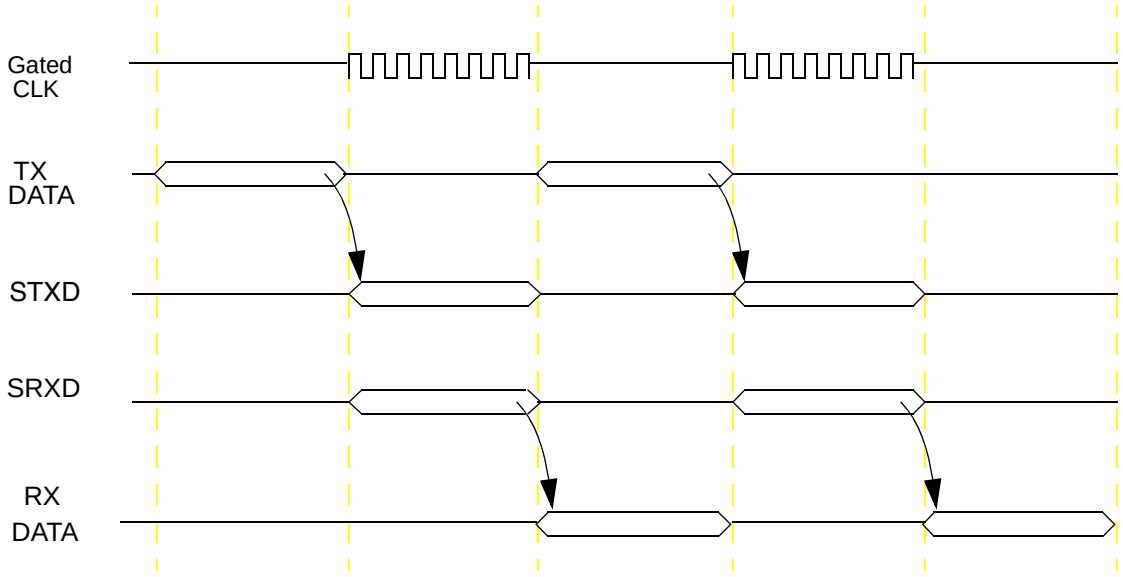


Figure 42-3. Normal Mode Timing—Internal Gated Clock

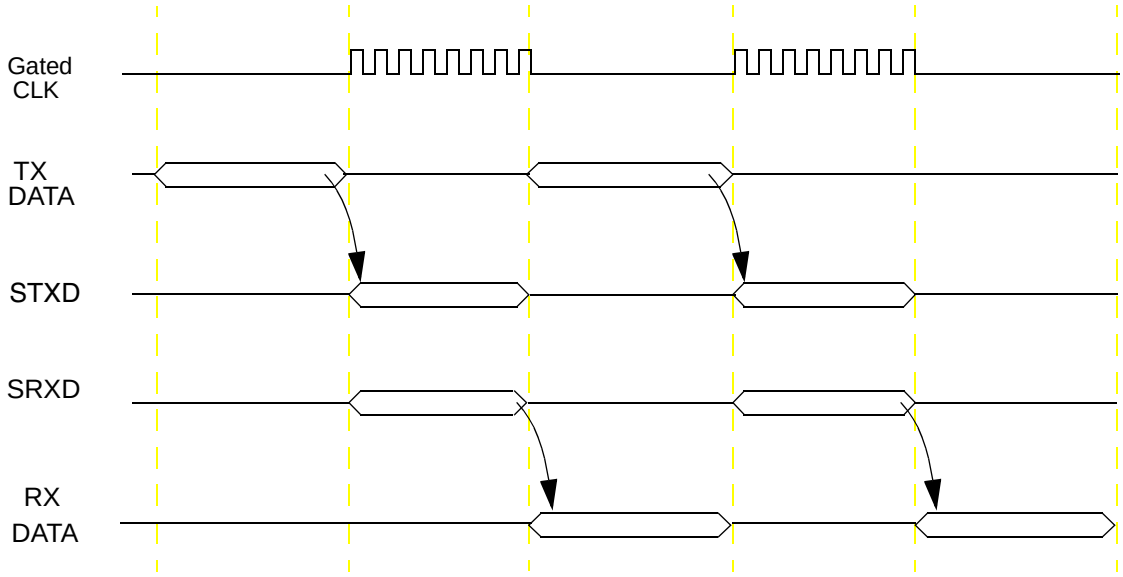


Figure 42-4. Normal Mode Timing—External Gated Clock

42.1.2.2 Network Mode

Network mode is used for creating a Time Division Multiplexed (TDM) network, such as a TDM CODEC network or a network of DSPs. In Continuous Clock mode, a frame sync occurs at the beginning of each frame. In this mode, the frame is divided into more than one time slot. During each time slot, one data word can be transferred. Each time slot is then assigned to an appropriate CODEC or DSP on the network. The DSP can be a master device that controls its own private network, or a slave device that is connected to an existing TDM network and occupies a few time slots.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots and transmission and/or reception of one data word can occur in each time slot (rather than in just the frame sync time slot as in Normal mode). The frame rate dividers, controlled by the DC[4:0] bits, select two to thirty-two time slots per frame. The length of the frame is determined by the following factors:

- The period of the serial bit clock (PSR, PM[7:0] bits for internal clock, or the frequency of the external clock on the STCK port)
- The number of bits per sample (WL[3:0] bits)
- The number of time slots per frame (DC[4:0] bits)

In Network mode, data can be transmitted in any time slot. The distinction of the Network mode is that each time slot is identified with respect to the frame sync (data word time). This time slot identification allows the option of transmitting data during the time slot by writing to the STX registers or ignoring the time slot as determined by STMSK register bits. The receiver is treated in the same manner and received data is only transferred to the receive data register/fifo if the corresponding time slot is enabled (through SRMSK).

By utilizing the STMSK and SRMSK registers, software only has to service the SSI during valid time slots. This eliminates any overhead associated with unused time slots. Refer to Section 42.3.3.20, “SSI Transmit Time Slot Mask Register (STMSK)” and Section 42.3.3.21, “SSI Receive Time Slot Mask Register (SRMSK)” for more information on STMSK and SRMSK.

In the Two-Channel mode of operation, the second set of Transmit and Receive FIFOs and Data Registers are used to create two separate channels. These channels are completely independent, with a their own set of Core interrupts and DMA requests, which are identical to the ones available for the default channel. In this mode, data is transmitted/received in enabled time slots alternately from/to FIFO 0 and FIFO 1, starting from FIFO 0. The first data word is taken from FIFO 0 and transmitted in the first enabled time slot and subsequently, data is loaded from FIFO 1 and FIFO 0 alternately and transmitted. Similarly, the first received data is sent to FIFO 0 and subsequent data is sent to FIFO 1 and FIFO 0 alternately. Time slots can be selected through the Transmit and Receive Time Slot Mask registers (STMSK and SRMSK). For using this mode of operation, the TCH_EN bit (SCR[8]) needs to be set.

42.1.2.2.1 Network Mode Transmit

The transmit portion of SSI is enabled when the SSIEN and the TE bits in the SCR are both set. However, for continuous clock, when the TE bit is set, the transmitter is enabled only after detection of a new frame sync (transmission starts from the next frame boundary).

Normal start-up sequence for transmission is to perform the following:

- Write the data to be transmitted to the STX register. This clears the TDE flag.
- Set the TE bit to enable the transmitter on the next word boundary (for continuous clock case).
- Enable transmit interrupts.

Alternately, the programmer may decide not to transmit in a time slot by writing to the STMSK. The TDE flag is not cleared, but the STXD port remains disabled during the time slot. When the frame sync is detected or generated (continuous clock), the first enabled data word is transferred from the STX register to the TXSR and is shifted out (transmitted). When the STX register is empty, the TDE bit is set, which causes a transmitter interrupt (in case FIFO is disabled) to be sent if the TIE bit is set. Software can poll

the TDE bit or use interrupts to reload the STX register with new data for the next time slot. Failing to reload the STX register before the TXSR is finished shifting (empty) causes a transmitter underrun, the TUE error bit to be set. In case FIFO is enabled, the TFE flag is set in accordance with the watermark setting and this flag causes the transmitter interrupt to occur.

The operation of clearing the TE bit disables the transmitter after completion of transmission of the current frame. Setting the TE bit enables transmission from the next frame. During that time the STXD port is disabled. The TE bit should be cleared after the TDE bit is set to ensure that all pending data is transmitted.

To summarize, the Network mode transmitter generates interrupts every enabled time slot and requires the core program to respond to each enabled time slot. These responses from the core are one of the following:

- Write data in data register to enable transmission in the next time slot.
- Configure the time slot register to disable transmission in the next time slot (unless time slot is already masked by STMSK register bit).
- Do nothing—transmit underrun occurs at the beginning of the next time slot and the previous data is re-transmitted.

In the Two-Channel mode of operation, both the channels (Data Registers, FIFOs, Interrupts and DMA requests) operate in the same manner, as described above. The only difference in case of the second channel is that the Interrupts related to this channel are generated only in case this mode of operation is selected (TDE1 is low by default).

42.1.2.2.2 Network Mode Receive

The receiver portion of the SSI is enabled when both the SSIEN and the RE bits in the SCR are set. However, the receive enable only takes place during that time slot if RE is enabled before the second to last bit of the word. If the RE bit is cleared, the receiver is disabled at the end of the current frame. SSI is capable of finding the start of the next frame automatically. When the word is completely received, it is transferred to the SRX register, which sets the RDR bit (Receive Data Ready). Setting the RDR bit causes a receive interrupt to occur if the receiver interrupt is enabled (the RIE bit is set). The second data word (second time slot in the frame), begins shifting in immediately after the transfer of the first data word to the SRX register. The DSP program has to read the data from the Receive Data Register (which clears RDR) before the second data word is completely received (ready to transfer to RX data register) or a receive overrun error occurs (the ROE bit is set).

An interrupt can occur after the reception of each enabled data word or the programmer can poll the RDR flag. The core program response can be one of the following:

- Read RX and use the data.
- Read RX and ignore the data.
- Do nothing—the receiver overrun exception occurs at the end of current time slot.

NOTE

For a continuous clock, the optional frame sync output and clock output signals are not affected, even if the transmitter or receiver is disabled. TE and RE do not disable the bit clock or the frame sync generation. To disable the bit clock and the frame sync generation, the SSIEN bit in the SCR can be cleared, or the port control logic external to the SSI (for example, in the IOMUX) can be reconfigured.

In the Two-Channel mode of operation, both the channels (Data Registers, FIFOs, Interrupts and DMA requests) operate in the same manner, as described above. The only difference in case of the second channel is that the Interrupts related to this channel are generated only in case this mode of operation is selected.

The transmitter and receiver timing for an 8-bit word with continuous clock, FIFO disabled, three words per frame sync in Network mode is shown in Figure 42-5.

NOTE

The transmitter repeats the value 0x5E because of an underrun condition

For the transmit section, the STMSK value is updated in the last time slot of frame 1, to mask the first two time slots (0x3). This value takes effect from the next time slot and consequently, the next frame transmits data in the third time slot only.

For the receive section, data received on the SRXD pin gets transferred to the Rx Data register at the end of each time slot. If FIFO is disabled, the RDR flag gets set and causes a receiver interrupt if RE, RIE and RDR_EN bits are set. If FIFO is enabled, then the RFF flag is used for interrupt generation (this flag is set in accordance with the watermark settings). Here all time slots are enabled. The receive data ready flag is set after reception of the first data (0x55). Since the flag is not cleared (Rx Data Register not read by core), the Receive Overrun Error (ROE) flag is set on reception of the next data (0x5E). ROE flag is cleared on reading the SSI Status Register followed by reading the Rx Data register.

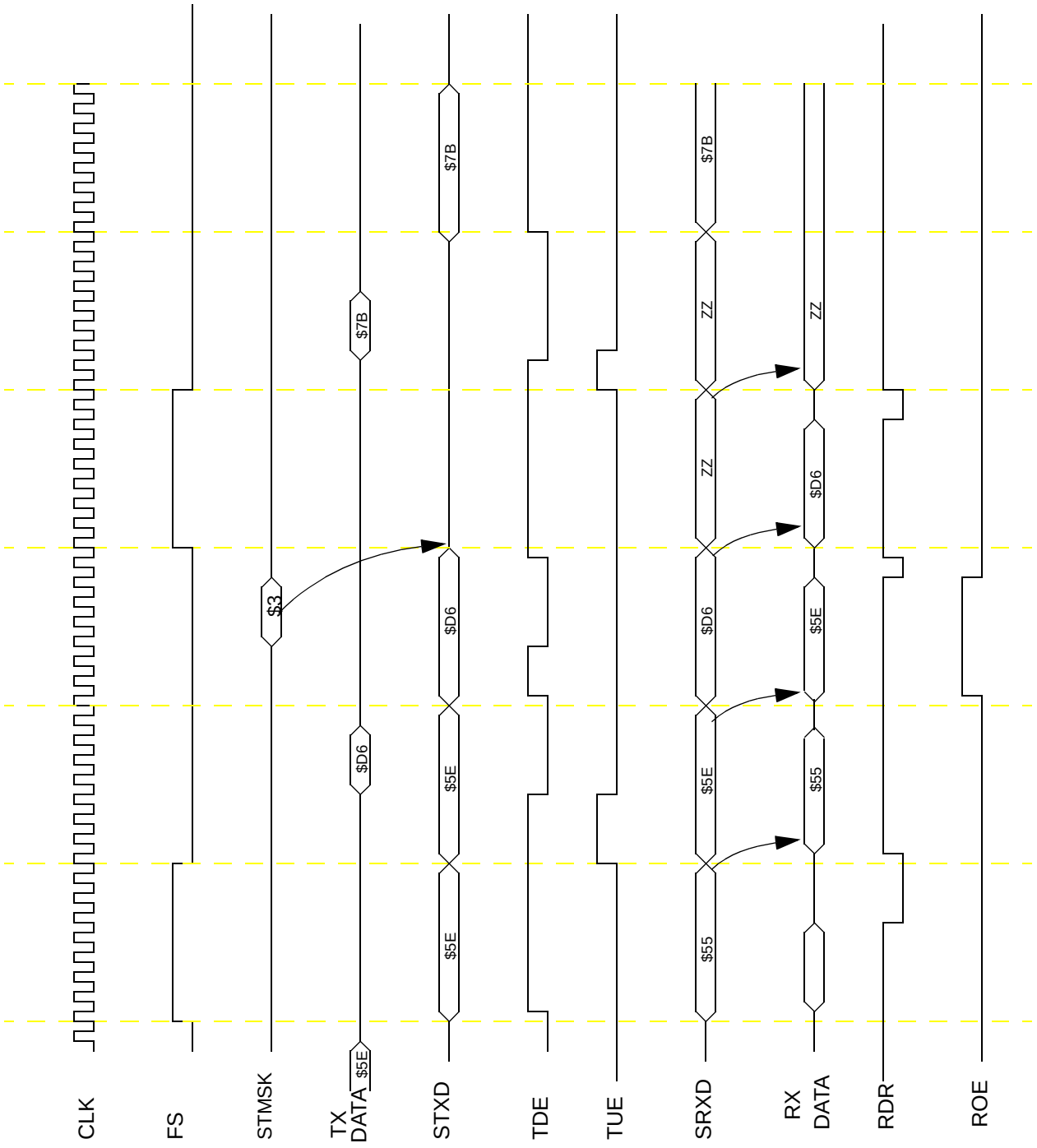


Figure 42-5. Network Mode Timing—Continuous Clock

42.1.2.3 Gated Clock Mode

Gated Clock mode is often used to hook up to SPI-type interfaces on Microcontroller Units (MCUs) or external peripheral chips. In Gated Clock mode, the presence of the clock indicates that valid data is on

the STXD or SRXD ports. For this reason, no frame sync is needed in this mode. Once transmission of data has completed, the clock is pulled to the inactive state. Gated clocks are allowed for both the transmit and receive sections with either internal or external clock and in Normal mode. Gated clocks are not allowed in Network mode. Refer to Table 42-5 for SSI configuration for gated-mode operation.

The clock runs when the TE bit and/or the RE bit are appropriately enabled. For the case of internally generated clock, all internal bit clocks, word clocks, and frame clocks continue to operate. When a valid time slot occurs (such as the first time slot in Normal mode), the internal bit clock is enabled onto the appropriate clock port. This allows data to be transferred out in periodic intervals in Gated Clock mode. With an external clock, the SSI waits for a clock signal to be received. Once the clock begins, valid data is shifted in. Care should be taken to clear all DC bits (0x00000) when SSI is used in Gated mode.

For Gated clock operated in external clock mode, a proper clock signalling must be apply to the SSI STCK in order for it to function properly. If the SSI uses rising edge transition to clock data (TSCKP=0) and falling edge transition to latch data (RSCKP=0), the clock must be in an active low state when idle. If the SSI uses falling edge transition to clock data (TSCKP=1) and rising edge transition to latch data (RSCKP=1), the clock must be in a active high state when idle. Figure 42-6 through Figure 42-9 illustrate the different edge clocking/latching.

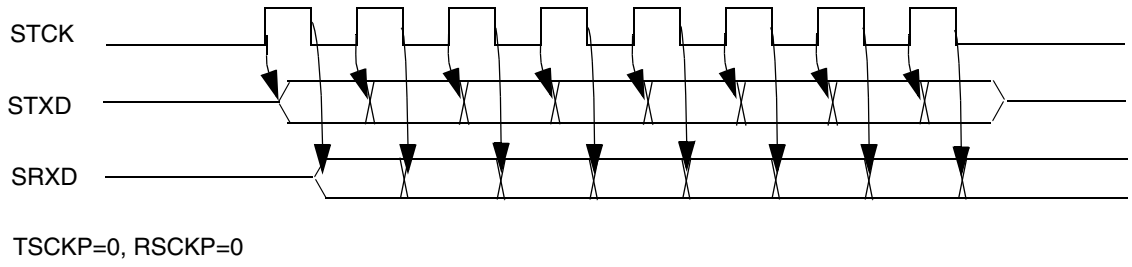


Figure 42-6. Internal Gated Mode Timing—Rising Edge Clocking/Falling Edge Latching

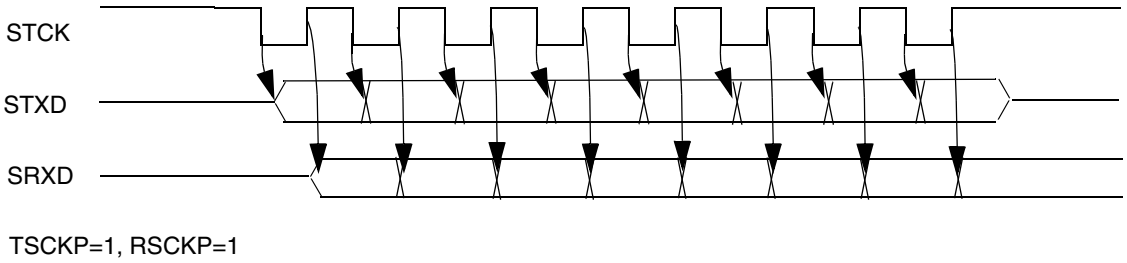


Figure 42-7. Internal Gated Mode Timing—Falling Edge Clocking/Rising Edge Latching

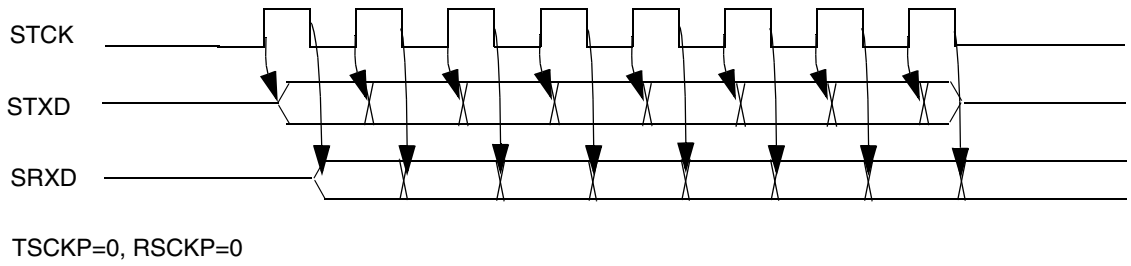


Figure 42-8. External Gated Mode Timing—Rising Edge Clocking/Falling Edge Latching

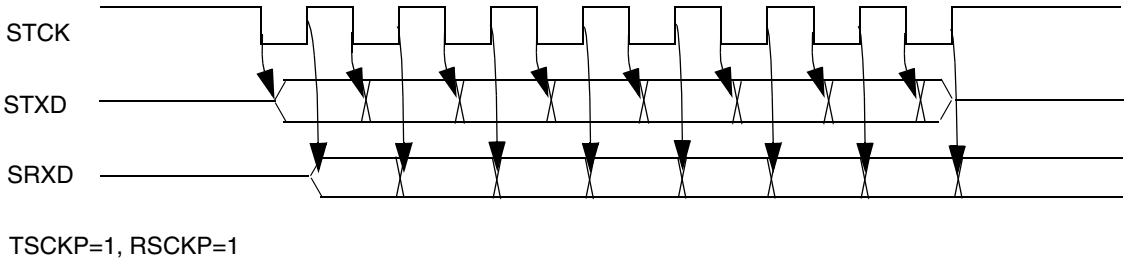


Figure 42-9. External Gated Mode Timing—Falling Edge Clocking/Rising Edge Latching

NOTE

- The bit clock ports must be kept free of timing glitches. If a single glitch occurs, all ensuing transfers will be out of synchronization.
- In case of External Gated Mode, even though the Tx Data line is put in the high-impedance state at the last non-active edge of the bit clock, the round trip delay should be sufficient to take care of hold time requirements at the external receiver.

42.1.2.4 I²S Mode

The SSI is compliant to I²S bus specification from Philips Semiconductors (February 1986, Revised June 5, 1996). Figure 42-10 depicts basic I²S protocol timing.

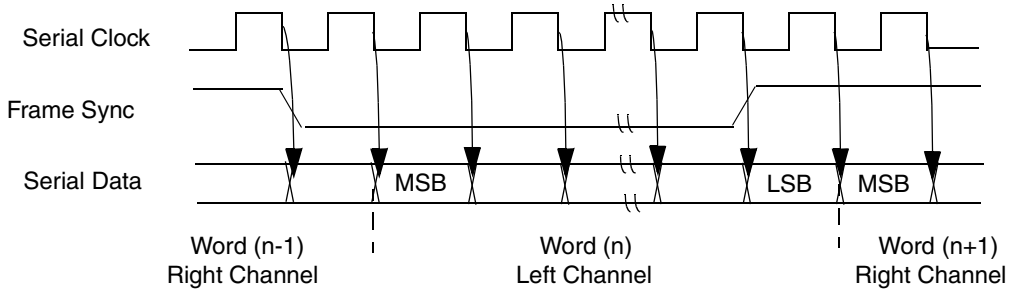


Figure 42-10. I²S Mode Timing—Serial Clock, Frame Sync, and Serial Data

I^2S mode can be selected in the following manner:

Table 42-2. I^2S Mode Selection

I^2S_MODE[1]	I^2S_MODE[0]	Remark
0	0	Normal mode
0	1	I^2S master mode
1	0	I^2S slave mode
1	1	Normal mode

In normal mode operation, all register bits are not forced to any particular state internally and user can program the SSI to work in any operating condition.

When I^2S modes are selected (I^2S master (01) or I^2S slave (10)), the following settings are recommended:

- Sync mode (SCR[4] =1)
- Tx shift direction: MSB transmitted first (STCR[4]=0)
- Rx shift direction: MSB received first (SRCR[4]=0)
- Tx data clocked at falling edge of the clock (STCR[3]=1)
- Rx data latched at rising edge of the clock (SRCR[3]=1)
- Tx frame sync active low (STCR[2]=1)
- Rx frame sync active low (SRCR[2]=1)
- Tx frame sync initiated one bit before data is transmitted (STCR[0]=1)
- Rx frame sync initiated one bit before data is received (SRCR[0]=1)

In I^2S master mode(SCR[6:5]=01), the following additional settings are recommended:

- TXDIR bit (STCR[5]) set to 1 to select internal generated bit clock
- TFDIR bit (STCR[6]) set to 1 to select internal generated frame sync

In I^2S master mode(SCR[6:5]=01), the following settings are done automatically by the hardware internally:

- Network mode is selected (SCR[3]=1)
- Tx frame sync length set to one-word-long-frame (STCR[1]=0)
- Rx frame sync length set to one-word-long-frame (SRCR[1]=0)
- Tx shifting w.r.t. bit 0 of TXSR (STCR[9]=1)
- Rx shifting w.r.t. bit 0 of RXSR (SRCR[9]=1)

The user needs to set the following control bits to configure the bit clock and frame sync:

- PM (STCCR[7:0])
- PSR (STCCR[17])
- DIV2(STCCR[18]
- WL (STCCR[16:13]
- DC (STCCR[12:8])

The word length is fixed to 32 in I^2S Master mode and the WL bits determine the number of bits that will contain valid data (out of the 32 transmitted/received bits in each channel). The fixing of word duration as 32 simplifies the relation between oversampling clock (ccm_ssi_clk) and Frame Sync (ccm_ssi_clk becomes an integer multiple of Frame Sync).

In I^2S slave mode(SCR[6:5]=10), the following additional settings are recommended:

- TXDIR bit(STCR[5]) set to 0 to select external generated bit clock
- TFDIR bit(STCR[6]) set to 0 to select external generated frame sync

In I^2S slave mode(SCR[6:5]=10), the following settings are done automatically by the hardware internally:

- Normal mode is selected (SCR[3]=0)
- Tx frame sync length set to one-bit-long-frame (STCR[1]=1)
- Rx frame sync length set to one-bit-long-frame (SRCR[1]=1)
- Tx shifting w.r.t. bit 0 of TXSR (STCR[9]=1)
- Rx shifting w.r.t. bit 0 of RXSR (SRCR[9]=1)

The user needs to set the following control bits to configure the data transmission:

- WL (STCCR[16:13])
- DC (STCCR[12:8])

The word length is variable in I^2S slave mode and the WL bits determine the number of bits that will contain valid data. The actual word length is determined by the external CODEC. The external I^2S Master still sends frame sync according to the I^2S protocol (early, word wide and active low), the SSI internally operates so that each frame sync transition is the start of a new frame (the WL bits determine the number of bits to be transmitted/received). After one data word has been transferred, the SSI waits for the next frame sync transition to start operation in the next time slot. Transmit (STMSK) and receive (SRMSK) mask bits should not be used in I^2S Slave mode of operation.

42.1.2.5 AC97 Mode

In AC97 mode of operation, the SSI transmits a 16-bit Tag Slot at the start of a frame and the rest of the slots (in that frame) are all 20-bits wide. The same sequence is followed while receiving data. Refer to the AC97 specification for details regarding transmit and receive sequences and data formats.

Note that the SSI only has one RxDATA pin so the SSI can only support one CODEC. Secondary CODECs are not supported.

When AC97 mode is enabled, the following settings are internally overridden by the hardware. The programmed register values are not changed by entering AC97 mode but they no longer apply to the module's operation. Writing to the programmed register fields will update their values; these updates can be seen by reading back the register fields. However, these settings will not take effect until AC97 mode is turned off.

The register bits within the bracket are the equivalent settings:

- Sync mode is entered (SCR[4] =1)
- Network mode is selected (SCR[3]=1)

- Tx shift direction is MSB transmitted first (STCR[4]=0)
- Rx shift direction is MSB received first (SRCR[4]=0)
- Tx data is clocked at rising edge of the clock (STCR[3]=0)
- Rx data is latched at falling edge of the clock (SRCR[3]=0)
- Tx frame sync is active high (STCR[2]=0)
- Rx frame sync is active high (SRCR[2]=0)
- Tx frame sync length is one-word-long-frame (STCR[1]=0)
- Rx frame sync length is one-word-long-frame (SRCR[1]=0)
- Tx frame sync initiated one bit before data is transmitted (STCR[0]=1)
- Rx frame sync initiated one bit before data is received (SRCR[0]=1)
- Tx shifting w.r.t. bit 0 of TXSR (STCR[9]=1)
- Rx shifting w.r.t. bit 0 of RXSR (SRCR[9]=1)
- Tx FIFO is enabled (STCR[7]=1)
- Rx FIFO is enabled (SRCR[7]=1)
- TFDIR bit (STCR[6]) is forced to 1 internally to select internal generated frame sync
- TXDIR bit(STCR[5]) is forced to 0 internally to select external generated bit clock

Any alteration of these bits individually will not affect the operational conditions of the SSI unless AC97 mode is deselected.

Hence the only control bits needed to set by user to configure the data transmission/reception are the WL (STCCR[16:13]) and DC (STCCR[12:8]) bits. In AC97 mode, the WL bits can only legally take the values corresponding to 16-bit (truncated data) or 20-bit time slots. In case WL bits are set to select 16-bit time slots, the SSI pads the transmit data (four least significant bits) with zeros and while receiving, stores only the most significant 16 bits in the Rx FIFO.

Follow below sequence for programming the SSI to work in AC97 mode:

- Program the WL bits to a value corresponding to either 16 or 20 bits. The WL bit setting is only for the data portion of the AC97 frame (Slots #3 through #12). The Tag slot (Slot #0) is always 16 bits wide and the Command Address and Command Data slots (Slots #1 and #2) are always 20 bits wide.
- Select the number of time slots through DC bits. For AC97 operation, DC bits should be set to a value of ‘0xC’, resulting in 13 time slots per frame.
- Write data to be transmitted, in Tx FIFO 0 (through Tx Data Register 0)
- Program the FV, TIF, RD, WR and FRDIV bits in SACNT register
- Update the contents of SACADD, SACDAT and SATAG (for Fixed mode only) registers
- Enable the AC97 mode of operation (AC97EN bit in SACNT register)

Once the SSI starts transmitting and receiving data (after being configured in AC97 mode), the programmer needs to service the interrupts, as and when they are raised (updates to command address/data or tag registers, reading of received data and writing more data for transmission). Further details regarding fixed and variable mode implementation are provided in the following sections.

While using AC97 in Two-Channel Mode (TCH_EN=1), it is recommended that the received tag is not stored in the Rx FIFO (TIF=0). In case the programmer needs to update the SATAG register and also issue a RD/WR command (in a single frame), it is recommended that the SATAG register be updated prior to issuing a RD/WR command.

42.1.2.5.1 AC97 Fixed Mode (SACNT[1]=0)

In fixed mode of operation, SSI transmits in accordance with the Frame Rate Divider bits which decides the number of frames for which the SSI should be idle, after operating for one frame.

In a valid frame, TAG Value (written by Core) will be transmitted in Slot #0, Command Address will be transmitted in Slot #1 in case of RD/WR Command, and Command Data will be transmitted in Slot #2 in case of a WR Command. The data from TX-FIFO is transmitted in Slot #3 to Slot #12 depending on the valid slots indicated by the TAG value.

While receiving, bit 15 of the TAG Value (Slot #0) is checked to see if the CODEC is ready. If this bit is set, the frame is received. The received TAG provides the information about Slots containing valid data. The the corresponding TAG bit is valid, the Command Address (Slot #1) and Command Data (Slot #2) values are stored in the corresponding registers. The received data (Slot #3 to Slot #12) is then stored in the Rx-FIFO (for valid slots).

42.1.2.5.2 AC97 Variable Mode (SACNT[1]=1)

In Variable Mode, the transmit slots which should contain data in the current frame are determined by SLOTREQ bits received in the previous frame. While receiving, if the CODEC is ready, the frame is received and the SLOTREQ bits (contained in Slot #1) are stored for scheduling transmission in the next frame.

The SACCST, SACCEN and SACCDIS registers help to determine which transmit slots are active. This information is used to ensure that SSI does not transmit data for powered-down/inactive channels.

42.1.2.6 External Frame and Clock Operation

When applying external frame sync and clock signals to SSI, there should be at least 4 bit clock cycles between the enabling of the transmit or receive section and the rising edge of the corresponding frame sync signal. The transition of STFS or SRFS should be synchronized with the rising edge of external clock signal, STCK or SRCK.

42.1.2.7 Data Alignment Formats Supported

The SSI supports three data formats in order to provide flexibility with handling data. These formats dictate how data is written to (and read from) the data registers. Therefore, data can appear in different places in STX0/1 and SRX0/1 based on the data format and the number of bits per word. Independent data formats are supported for both the transmitter and receiver (that is, the transmitter and receiver can use different data formats).

The supported data formats are:

- MSB alignment

- LSB alignment
  - — Zero-extended (receive data only)
  - — Sign-extended (receive data only)

With MSB alignment, the most significant byte is bits 31 through 24 of the data register if the word length is larger than or equal to 16 bits. If the word length is less than 16 bits and MSB alignment is chosen, the most significant byte is bits 15 through 8. With LSB alignment, the least significant byte is bits 7 through 0. Data alignment is controlled by the TXBIT0 bit in the STCR and the RXBIT0 bit in the SRCR. Table 42-3 shows the bit assignment for all the data formats supported by the SSI.

Table 42-3. Data Alignment

Format	Bit Number																													
	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
8-bit LSB Aligned																									7	6	5	4	3	2	1	0
8-bit MSB Aligned																	7	6	5	4	3	2	1	0								
10-bit LSB Aligned																							9	8	7	6	5	4	3	2	1	0
10-bit MSB Aligned																	9	8	7	6	5	4	3	2	1	0						
12-bit LSB Aligned																					11	10	9	8	7	6	5	4	3	2	1	0
12-bit MSB Aligned																	11	10	9	8	7	6	5	4	3	2	1	0				
16-bit LSB Aligned																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16-bit MSB Aligned	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
18-bit LSB Aligned															17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
18-bit MSB Aligned	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
20-bit LSB Aligned													19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
20-bit MSB Aligned	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
22-bit LSB Aligned										21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
22-bit MSB Aligned	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
24-bit LSB Aligned									23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24-bit MSB Aligned	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

In addition, receive data can either be zero-extended or sign-extended if LSB alignment is selected. With zero-extension, all bits above the most significant bit are ‘0’s. This format is useful when data is stored in a pure integer format. With sign-extension, all bits above the most significant bit are equal to the most significant bit. This format is useful when data is stored in a fixed-point integer format (which implies fractional values). Receive data extension is controlled by the RXEXT bit in the SRCR. Transmit data used with LSB alignment has no concept of sign/zero-extension. Unused bits above the most significant bit are simply ignored.

When configured in I^2S or AC97 mode, the SSI forces the selection of LSB alignment. However, RXEXT still permits a choice between zero-extension and sign-extension.

Refer to Section 42.3.3.10, “SSI Transmit Configuration Register (STCR)” and Section 42.3.3.11, “SSI Receive Configuration Register (SRCR)” for more detail on the relevant bits in the STCR and SRCR.

42.2 External Signal Description

42.2.1 Overview

The SSI has no external signals as its serial I/O signals are connected to Digital Audio MUX (AUDMUX) in the appropriate manner. The six SSI ports can be connected to various chip ports by programming the Digital Audio MUX (AUDMUX) in the appropriate manner. Refer to Chapter 38, “Digital Audio MUX (AUDMUX)” for details regarding this programming.

Table 42-4. Signal Properties

Name	Port	Function	Reset State	Pull up
SRCK	—	Serial Receive Clock	0	Passive
SRFS	—	Serial Receive Frame Sync	0	Passive
SRXD	—	Serial Receive Data	—	—
STCK	—	Serial Transmit Clock	0	Passive
STFS	—	Serial Transmit Frame Sync	0	Passive
STXD	—	Serial Transmit Data	0	Passive

42.2.2 Detailed Signal Descriptions

42.2.2.1 SRCK—Serial Receive Clock

The SRCK port can be used as either an input or an output. This clock signal is used by the receiver and is always continuous. During Gated Clock mode, the STCK port is used instead for clocking in data. In SSI master modes, this port can be used as an output port for the oversampling clock, SYS_CLK (ccm_ssi_clk). In I^2S master mode, this port can be used to output ccm_ssi_clk to external CODEC.

42.2.2.2 SRFS—Serial Receive Frame Sync

The SRFS port can be used as either an input or an output. The frame sync is used by the receiver to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur one bit before the transfer of data or right at the transfer of data. If SRFS is configured as input, the external device should drive SRFS during rising edge of STCK or SRCK.

42.2.2.3 SRXD—Serial Receive Data

The SRXD port is an input and is used to bring serial data into the Receive Data Shift Register.

42.2.2.4 STCK—Serial Transmit Clock

The STCK port can be used as either an input or an output. This clock signal is used by the transmitter and can be either continuous or gated. During Gated Clock mode, data on the STCK port is valid only during the transmission of data, otherwise it is pulled to the inactive state. In Synchronous mode, this port is used by both the transmit and receive sections.

42.2.2.5 STFS—Serial Transmit Frame Sync

The STFS port can be used as either an input or an output. The frame sync is used by the transmitter to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur one bit before the transfer of data or right at the transfer of data. In Synchronous mode, this port is used by both the transmit and receive sections. In Gated Clock mode, frame sync signals are not used. If STFS is configured as input, the external device should drive STFS during rising edge of STCK or SRCK.

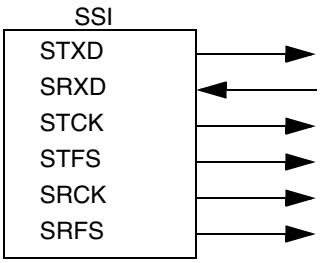
42.2.2.6 STXD—Serial Transmit Data

The STXD port is an output and transmits data from the Serial Transmit Shift Register. The STXD port is an output port when data is being transmitted and is disabled between data word transmissions and on the trailing edge of the bit clock after the last bit of a word is transmitted.

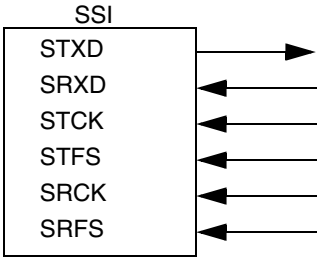
Figure 42-11 and Figure 42-12 show the main SSI configurations. These ports support all transmit and receive functions with continuous or gated clock as shown.

NOTE

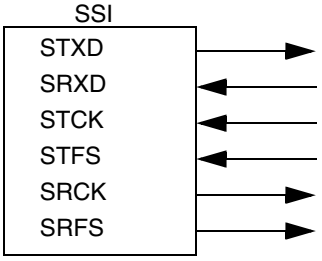
Gated clock implementations do not require the use of the frame sync ports (STFS and SRFS).



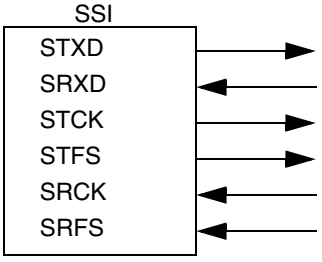
SSI Internal Continuous Clock for TX/RX (RXDIR=1,TXDIR=1,RFDIR=1,TFDIR=1,SYN=0)



SSI External Continuous Clock for TX/RX (RXDIR=0,TXDIR=0,RFDIR=0,TFDIR=0,SYN=0)

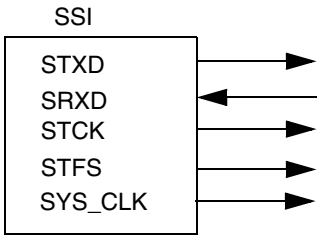


SSI Internal Continuous Clock for RX (RXDIR=1, TXDIR=0,RFDIR=1,TFDIR=0, SYN=0)
SSI External Continuous Clock for TX

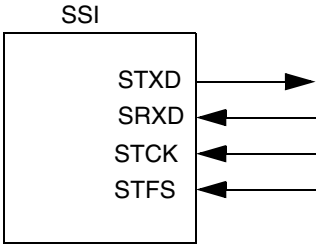


SSI Internal Continuous Clock for TX (RXDIR=0, TXDIR=1, RFDIR=0, TFDIR=1, SYN=0)
SSI EXternal Continuous Clock for RX

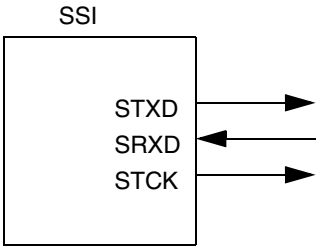
Figure 42-11. Asynchronous (SYN=0) SSI Configurations—Continuous Clock



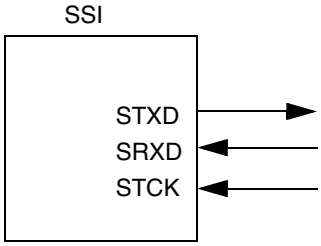
SSI Internal Continuous Clock (RXDIR=0, TXDIR=1, RFDIR=X, TFDIR=1, SYN=1, SYS_CLK_EN = 1)
SSI I2S Master Mode (I2S_Mode=01, SYS_CLK_EN)



SSI External Continuous Clock (RXDIR=0, TXDIR=0, RFDIR=X, TFDIR=0,SYN=1)
SSI I2S Slave Mode (I2S_Mode=10)



SSI Internal Gated Clock (RXDIR=1, TXDIR=1, SYN=1)



SSI External Gated Clock (RXDIR=1, TXDIR=0, SYN=1)

Figure 42-12. Synchronous SSI Configurations—Continuous and Gated Clock

An example of the port signals for an 8-bit data transfer is shown in Figure 42-13. Continuous and gated clock signals are shown, as well as the bit-length frame sync signal and the word-length frame sync signal.

NOTE

The shift direction can be defined as MSB first or LSB first, and that there are other options on the clock and frame sync.

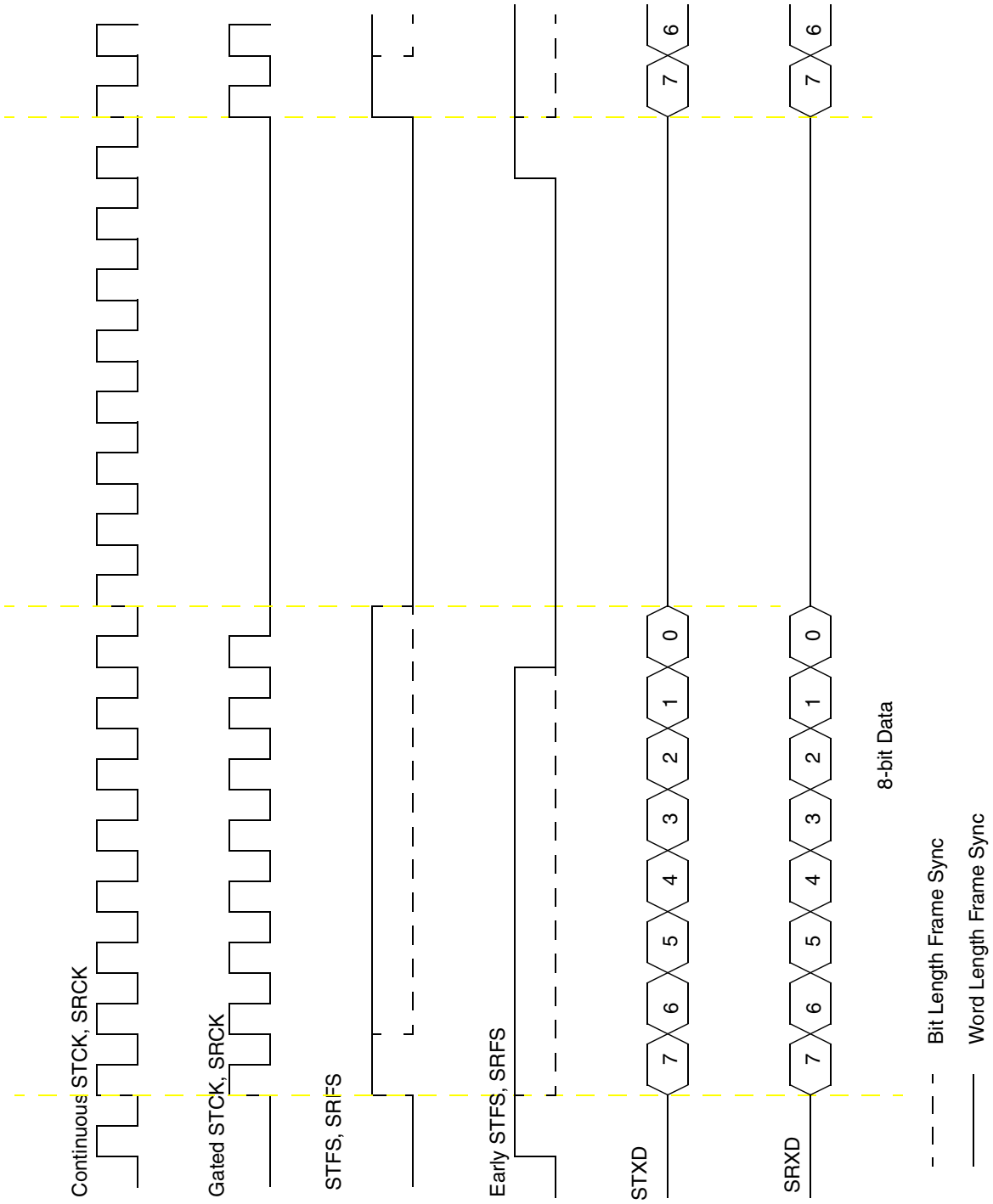


Figure 42-13. Serial Clock and Frame Sync Timing

Table 42-5. Clock Pin Configuration

SYN	RXDIR	TXDIR	RFDIR	TFDIR	SRCK	STCK	SRFS	STFS
Asynchronous Mode								
0	0	0	0	0	RCK in	TCK in	RFS in	TFS in
0	0	0	0	1	RCK in	TCK in	RFS in	TFS out
0	0	0	1	0	RCK in	TCK in	RFS out	TFS in
0	0	0	1	1	RCK in	TCK in	RFS out	TFS out
0	0	1	0	0	RCK in	TCK out	RFS in	TFS in
0	0	1	0	1	RCK in	TCK out	RFS in	TFS out
0	0	1	1	0	RCK in	TCK out	RFS out	TFS in
0	0	1	1	1	RCK in	TCK out	RFS out	TFS out
0	1	0	0	0	RCK out	TCK in	RFS in	TFS in
0	1	0	0	1	RCK out	TCK in	RFS in	TFS out
0	1	0	1	0	RCK out	TCK in	RFS out	TFS in
0	1	0	1	1	RCK out	TCK in	RFS out	TFS out
0	1	1	0	0	RCK out	TCK out	RFS in	TFS in
0	1	1	0	1	RCK out	TCK out	RFS in	TFS out
0	1	1	1	0	RCK out	TCK out	RFS out	TFS in
0	1	1	1	1	RCK out	TCK out	RFS out	TFS out
Synchronous Mode								
1	0	0	x	0	—	CK in	—	FS in
1	0	0	x	1	—	CK in	—	FS out
1	0	1	x	0	—	CK out	—	FS in
1	0	1	x	1	—	CK out	—	FS out
1	1	0	x	x	—	Gated in	—	—
1	1	1	x	x	—	Gated out	—	—

42.2.3 Internal I/O Signal Description

Table 42-6 provides a list of internal I/O signals.

Table 42-6. Internal I/O Signal Description

Name	I/O	Function
ipg_hard_async_reset_b	Input	Global Hardware Reset signal
ipg_clk	Input	IP Interface working clock
ipg_clk_s	Input	IP Interface register access clock

Table 42-6. Internal I/O Signal Description (continued)

Name	I/O	Function
ips_module_en	Input	IP Bus module enable
ips_addr[13:2]	Input	IP Bus address signal. Address of the register being accessed
ips_rwb	Input	IP Bus read/write signal
ips_byte_31_24	Input	IP Bus byte enable signal for the Bits [31:24]
ips_byte_23_16	Input	IP Bus byte enable signal for the Bits [23:16]
ips_byte_15_8	Input	IP Bus byte enable signal for the Bits [15:8]
ips_byte_7_0	Input	IP Bus byte enable signal for the Bits [7:0]
ips_wdata[31:0]	Input	IP Write Data Bus. All register data writes occur through this bus.
ipt_scan_mode	Input	DFT test mode signal
ipt_test_reset_b	Input	DFT test mode reset signal
ipt_se_async	Input	DFT test asynchronous select signal
ipt_se_gatedclk	Input	DFT test clock select signal
ccm_ssi_clk	Input	SSI input clock for bit clock generation
resp_sel	Input	Select behavior of ips_xfr_err signal (see Note below)
din_srxd	Input	SSI receive data input
din_stck	Input	SSI transmit clock input
din_stfs	Input	SSI transmit frame sync input
din_srck	Input	SSI receive clock input
din_srfs	Input	SSI receive frame sync input
ipg_enable_clk	Output	This signal can be used to gate off the functional clocks when the module is not enabled.
ipi_int_b	Output	SSI interrupt request (active low)
ipd_ssi_rx1_dmareq_b	Output	SSI receive 1 DMA request
ipd_ssi_tx1_dmareq_b	Output	SSI transmit 1 DMA request
ipd_ssi_rx0_dmareq_b	Output	SSI receive 0 DMA request
ipd_ssi_tx0_dmareq_b	Output	SSI transmit 0 DMA request
ips_rdata[31:0]	Output	IP Read Data Bus. All register data reads occur through this bus.
ips_xfr_err	Output	This signal indicates IP Bus access errors.
ips_xfr_wait	Output	IP Bus wait state signal
ssi_stxd	Output	SSI transmit data output
ssi_stck	Output	SSI transmit clock output
ssi_stfs	Output	SSI transmit frame sync output
ssi_srck	Output	SSI receive clock output

Table 42-6. Internal I/O Signal Description (continued)

Name	I/O	Function
ssi_srfs	Output	SSI receive frame sync output
ssi_ddr_stxd	Output	SSI transmit data (ssi_stxd) output enable
ssi_ddr_stck	Output	SSI transmit clock (ssi_stck) output enable
ssi_ddr_stfs	Output	SSI transmit frame sync (ssi_stfs) output enable
ssi_ddr_srck	Output	SSI receive clock (ssi_srck) output enable
ssi_ddr_srfs	Output	SSI receive frame sync (ssi_srfs) output enable

NOTE

In case the resp_sel signal is low (0), the ips_xfr_err signal is used to generate an ERROR response in case there is an access to the unused portion of SSI's 16 KB address space. Otherwise, an OKAY response is generated in case of such accesses.

42.3 Memory Map and Register Definition

Section 42.3.3, "Register Descriptions" provides the detailed descriptions for all of the SSI registers.

42.3.1 R/WSSI Memory Map

Table 42-7 shows the SSI memory map.

Table 42-7. SSI Memory Map

Address	Register	Access	Reset Value
0x1001_1000 (STX0) 0x1001_1004 (STX1)	SSI Transmit Data Register 0 (STX0) SSI Transmit Data Register 1 (STX1)	R/W R/W	0x0000_0000 0x0000_0000
0x1001_1008 (SRX0) 0x1001_100C (SRX1)	SSI Receive Data Register 0 (SRX0) SSI Receive Data Register 1 (SRX1)	R	0x0000_0000
0x1001_1010 (SCR)	SSI Control Register (SCR)	R/W	0x0000_0000
0x1001_1014 (SISR)	SSI Interrupt Status Register (SISR)	Read Only	0x0000_3003
0x1001_1018 (SIER)	SSI Interrupt Enable Register (SIER)	R/W	0x0000_3003
0x1001_101C (STCR)	SSI Transmit Configuration Register (STCR)	R/W	0x0000_0200
0x1001_1020 (SRCR)	SSI Receive Configuration Register (SRCR)	R/W	0x0000_0200
0x1001_1024 (STCCR)	SSI Transmit Clock Control Register (STCCR)	R/W	0x0004_0000
0x1001_1028 (SRCCR)	SSI Receive Clock Control Register (SRCCR)	R/W	0x0004_0000
0x1001_102C (SFCSR)	SSI FIFO Control/Status Register (SFCSR)	R/W	0x0081_0081
0x1001_1030 (STR)	SSI Test Register (STR)[1]	R/W	0x0000_1111

Table 42-7. SSI Memory Map (continued)

Address	Register	Access	Reset Value
0x1001_1034 (SOR)	SSI Option Register (SOR)[2]	R/W	0x0000_0000
0x1001_1038 (SACNT)	SSI AC97 Control Register (SACNT)	R/W	0x0000_0000
0x1001_103C (SACADD)	SSI AC97 Command Address Register (SACADD)	R/W	0x0000_0000
0x1001_1040 (SACDAT)	SSI AC97 Command Data Register (SACDAT)	R/W	0x0000_0000
0x1001_1044 (SATAG)	SSI AC97 Tag Register (SATAG)	R/W	0x0000_0000
0x1001_1048 (STMSK)	SSI Transmit Time Slot Mask Register (STMSK)	R/W	0x0000_0000
0x1001_104C (SRMSK)	SSI Receive Time Slot Mask Register (SRMSK)	R/W	0x0000_0000
0x1001_1050 (SACCST)	SSI AC97 Channel Status Register	R	0x0000_0000
0x1001_1054 (SACCEN)	SSI AC97 Channel Enable Register	W	0x0000_0000
0x1001_1058 (SACCEN)	SSI AC97 Channel Disable Register	W	0x0000_0000

[1]SSI Test Register is intended for debugging purposes only and is not visible to the end user.
[2]SSI Option Register intended for internal use only and is not visible to the end user.

42.3.2 Register Summary

Figure 42-14 shows the key to the register fields and Table 42-8 shows the register figure conventions.

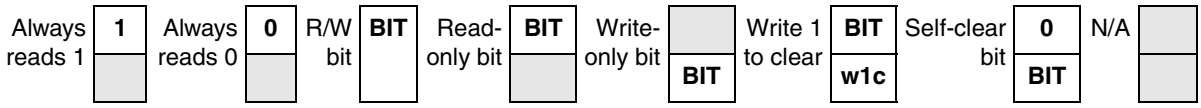


Figure 42-14. Key to Register Fields

Table 42-8. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit.	Writing a one has some effect on the module, but it always reads as zero.

Table 42-8. Register Figure Conventions

Convention	Description
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 42-9 shows the SSI register summary.

Table 42-9. Register Summary

Name		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
0x1001_1000 (STX0)	R	STX0[31:16]															
	W																
	R	STX0[15:0]															
	W																
0x1001_1004 (STX1)	R	STX1[31:16]															
	W																
	R	STX1[15:0]															
	W																
0x1001_1008 (SRX0)	R	SRX0[31:16]															
	W																
	R	SRX0[15:0]															
	W																
0x1001_100C (SRX1)	R	SRX1[31:16]															
	W																
	R	SRX1[15:0]															
	W																
0x1001_1010 (SCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	CLK_IST	TCH_EN	SYS_CLK_EN	I^2S MODE[1:0]		SYN	NET	RE	TE	SS-IEN
	W																

Table 42-9. Register Summary (continued)

Name		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
0x1001_1014 (SISR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CMDAU	CMDDU	RXT
	W																
	R	RDR1	RDR0	TDE1	TDE0	ROE1	ROE0	TUE1	TUE0	TFS	RFS	TLS	RLS	RFF1	RFF0	TFE1	TFE0
	W																
0x1001_1018 (SIER)	R	0	0	0	0	0	0	0	0	0	RDMAE	RIE	TDMAE	TIE	CMDAU_EN	CMDDU_EN	RXT_EN
	W																
	R	RDR1_EN	RDR0_EN	TDE1_EN	TDE0_EN	ROE1_EN	ROE0_EN	TUE1_EN	TUE0_EN	TFS_EN	RFS_EN	TLS_EN	RLS_EN	RFF1_EN	RFF0_EN	TFE1_EN	TFE0_EN
	W																
0x1001_101C (STCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	TXBIT0	TFEN1	TFEN0	TFDIR	TXDIR	TSHFD	TSCKP	TFSI	TFSL	TEFS
	W																
0x1001_1020 (SRCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	RXEXT	RXBIT0	RFEN1	RFEN0	RFDIR	RXDIR	RSHFD	RSCKP	RFSI	RFSL	REFS
	W																
0x1001_1024 (STCCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
	W																
	R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
	W																
0x1001_1028 (SRCCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
	W																
	R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
	W																
0x1001_102C (SFCSR)	R	RFCNT1[3:0]				TFCNT1[3:0]				RFWM1[3:0]				TFWM1[3:0]			
	W																
	R	RFCNT0[3:0]				TFCNT0[3:0]				RFWM0[3:0]				TFWM0[3:0]			
	W																

Table 42-9. Register Summary (continued)

Name		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
0x1001_1030 (STR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	TEST	RCK2TCK	RFS2TFS	RXSTATE[4:0]					TXD2RXD	TCK2RCK	TFS2RFS	TXSTATE[4:0]				
	W																
0x1001_1034 (SOR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	CLKOFF	RX_CLR	TX_CLR	INIT	WAIT[1:0]		SYNRST
	W																
0x1001_1038 (SACNT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0			
	W																
	R	0	0	0	0	0	FRDIV[5:0]						WR	RD	TIF	FV	AC97EN
	W																
0x1001_103C (SACADD)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	SACADD[18:16]		
	W																
	R	SACADD[15:0]															
	W																
0x1001_1040 (SACDAT)	R	0	0	0	0	0	0	0	0	0	0	0	0	SACDAT[19:16]			
	W																
	R	SACDAT[15:0]															
	W																
0x1001_1044 (SATAG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SATAG[15:0]															
	W																
0x1001_1048 (STMSK)	R	STMSK[31:0]															
	W																
	R																
	W																

Table 42-9. Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x1001_104C (SRMSK)	R	SRMSK[31:0]															
	W																
	R																
	W																
0x1001_1050 (SACCST)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	SACCST[9:0]									
	W																
0x1001_1054 (SACCEN)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W							SACCEN[9:0]									
0x1001_1058 (SACCEN)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W							SACCDIS[9:0]									

42.3.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

42.3.3.1 SSI Transmit Data Registers 0 and 1 (STX0/1)

See Figure 42-15 for an illustration of valid bits in SSI Transmit Data Register and Table 42-10 for descriptions of the bit fields in the register.

0x1001_1000 (STX0)
0x1001_1004 (STX1)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STX0/1[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STX0/1[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-15. SSI Transmit Data Registers

Table 42-10. SSI Transmit Data Register Field Descriptions

Field	Description
31–0 STX0 STX1	SSI Transmit Data. These bits store the data to be transmitted by the SSI. These are implemented as the first word of their respective Tx FIFOs. Data written to these registers is transferred to the Transmit Shift Register (TXSR), when shifting of the previous data is complete. If both FIFOs are in use, data is alternately transferred from STX0 and STX1, to TXSR. Multiple writes to the STX registers will not result in the previous data being over-written by the subsequent data. STX1 can only be used in Two-Channel mode of operation. Protection from over-writing is present irrespective of whether the transmitter is enabled or not. Example 1: If Tx FIFO0 is in use and user writes Data1...Data9 to STX0, Data9 will not over-write Data1. Data1...Data8 are stored in the FIFO while Data9 is discarded. Example 2: If Tx FIFO0 is not in use and user writes Data1, Data2 to STX0, then Data2 will not over-write Data1 and will be discarded.

NOTE

Enable SSI (SSIEN=1) before writing to SSI Transmit Data Registers.

42.3.3.2 SSI Transmit FIFO 0 and 1 Registers

The SSI Transmit FIFO registers are 8x24-bit registers. These registers are not directly accessible by the end user (except in SSI test mode). Transmit Shift Register (TXSR) receives its values from these FIFO registers. Transmitted data is first-in-first-out. When the Transmit Interrupt Enable (TIE) bit in the SIER and either of the Transmit FIFO Empty (TFE0 or 1) bits in the SISR are set, the core is interrupted whenever the data level in either of the SSI Transmit FIFOs falls below the selected threshold.

42.3.3.3 SSI Transmit Shift Register (TXSR)

The SSI Transmit Shift Register (TXSR) is a 24-bit shift register that contains the data being transmitted. This register is not directly accessible by the end user. When a continuous clock is used, data is shifted out to the Serial Transmit Data (STXD) port by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted out to the STXD port

by the selected (internal/external) gated clock. The Word Length control bits (WL[3:0]) in the STCCR (described in SSI Transmit and Receive Clock Control Registers) determine the number of bits to be shifted out of the TXSR before it is considered empty and can be written to again. This word length can be 8, 10, 12, 16, 18, 20, 22 or 24 bits. The data to be transmitted occupies the most significant portion of the shift register if TXBIT0 is '0', otherwise it occupies the least significant portion. The unused portion of the register is ignored. Data is always shifted out of this register with the Most Significant Bit (MSB) first when the SHFD bit of the STCR is cleared. If this bit is set, the Least Significant Bit (LSB) is shifted out first. Figure 42-16 through Figure 42-19 show the transmitter loading and shifting operation. The figures show the working for some WL values, the same can be extended for other values.

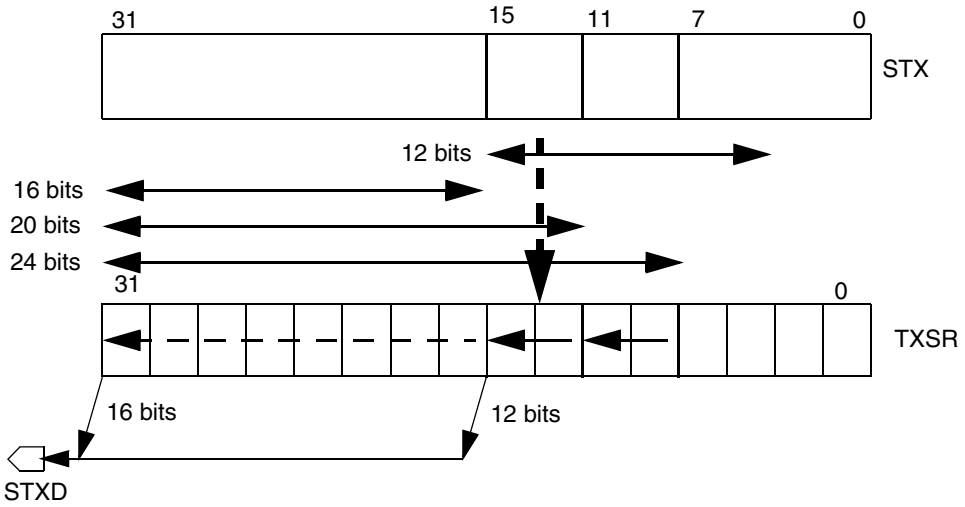


Figure 42-16. Transmit Data Path (TXBIT0=0, TSHFD=0) (MSB Alignment)

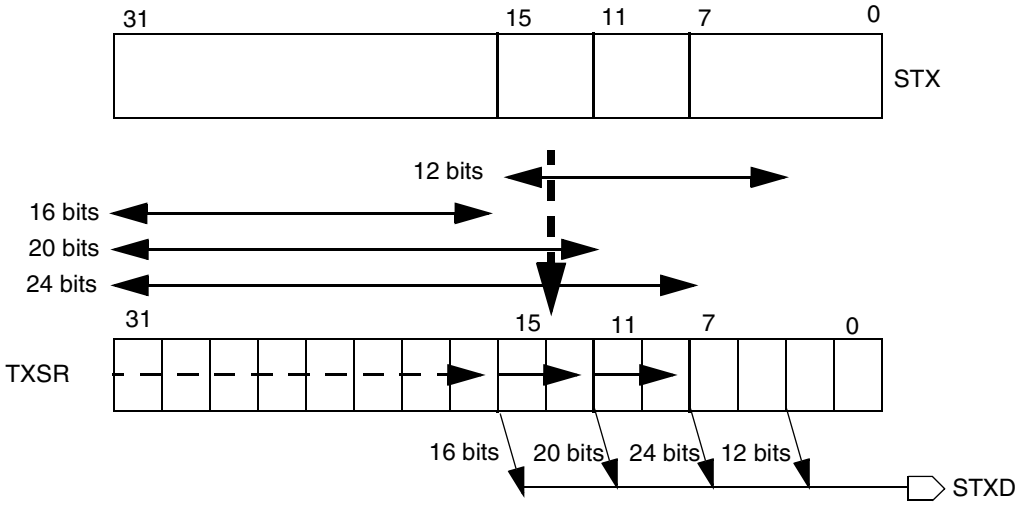


Figure 42-17. Transmit Data Path (TXBIT0=0, TSHFD=1) (MSB Alignment)

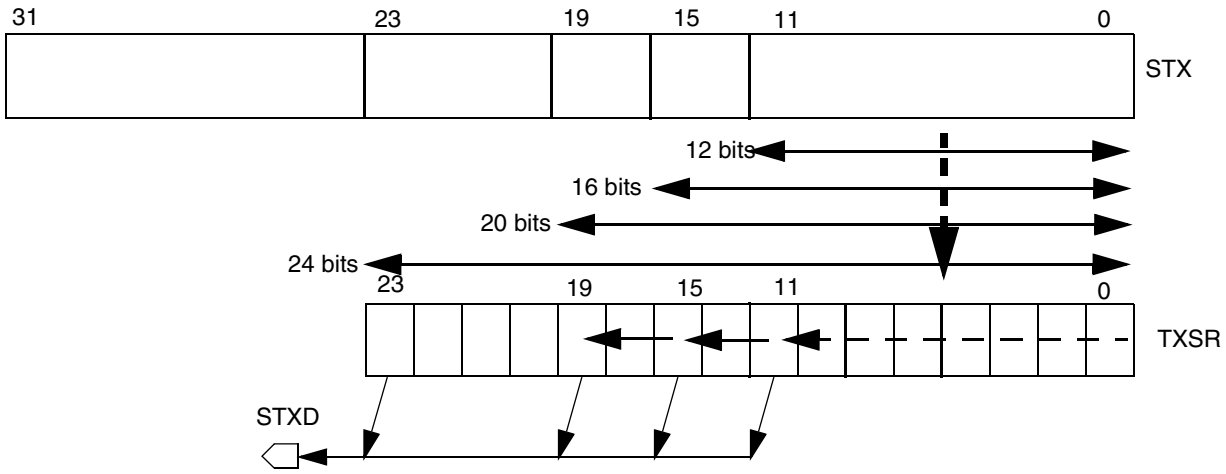


Figure 42-18. Transmit Data Path (TXBIT0=1, TSHFD=0) (LSB Alignment)

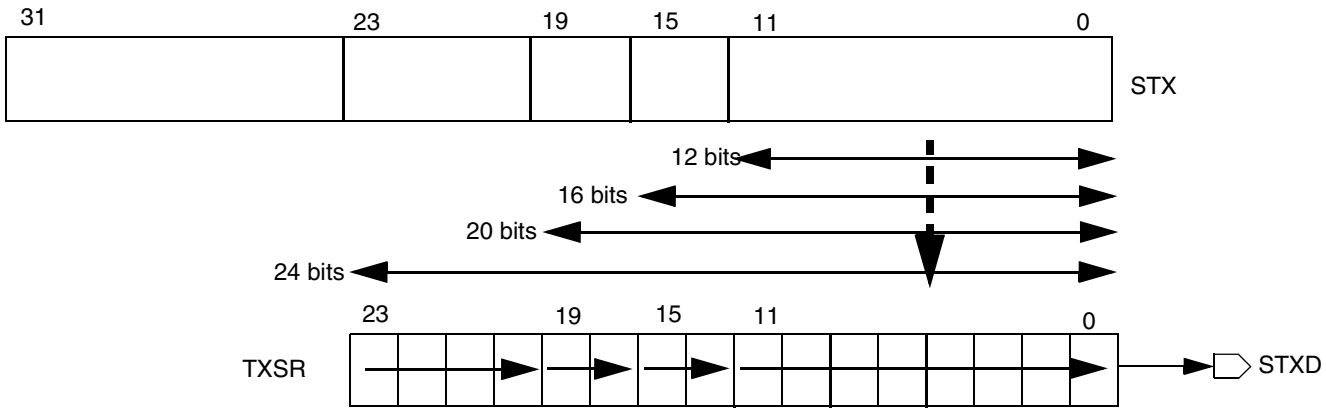


Figure 42-19. Transmit Data Path (TXBIT0=1, TSHFD=1) (LSB Alignment)

42.3.3.4 SSI Receive Data Registers 0 and 1 (SRX0/1)

See Figure 42-20 for an illustration of valid bits in SSI Receive Data Register and Table 42-11 for descriptions of the bit fields in the register.

0x1001_1008 (SRX0)
0x1001_100C (SRX1)

Access: User Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRX0/1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRX0/1[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-20. SSI Receive Data Registers

Table 42-11. SSI Receive Data Register Field Descriptions

Field	Description
31–0 SRX0 SRX1	SSI Receive Data. These bits store the data received by the SSI. These are implemented as the first word of their respective Rx FIFOs. These bits receive data from the RXSR depending on the mode of operation. In case both FIFOs are in use, data is transferred to each data register alternately. SRX1 can only be used in Two-Channel mode of operation.

42.3.3.5 SSI Receive FIFO 0 and 1 Registers

The SSI Receive FIFO Registers are 8x24-bit registers. These registers are not directly accessible by the end user (except in SSI test mode). They always accepts data from the Receive Shift Register (RXSR). The core is interrupted when the data level in either of the SSI Receive FIFOs reaches the selected threshold, if the associated interrupt is enabled.

42.3.3.6 SSI Receive Shift Register (RXSR)

The SSI Receive Shift Register (RXSR) is a 24-bit, shift register that receives incoming data from the serial receive data SRXD port. This register is not directly accessible by the end user. When a continuous clock is used, data is shifted in by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted in by the selected (internal/external) gated clock. Data is assumed to be received MSB first if the SHFD bit of the SRCR is cleared. If this bit is set, the data is received LSB first. Data is transferred to the appropriate SSI Receive Data Register (SRX0/1) or Receive FIFOs (if the receive FIFO is enabled and the corresponding SRX is full) after 8, 10, 12, 16, 18, 20, 22 or 24 bits have been shifted in depending on the WL[3:0] control bits. For receiving less than 24 bits of data, LSB bits are appended with zero. The following figures show the receiver loading and shifting operation. The figures show the working for some WL values, the same can be extended for other values.

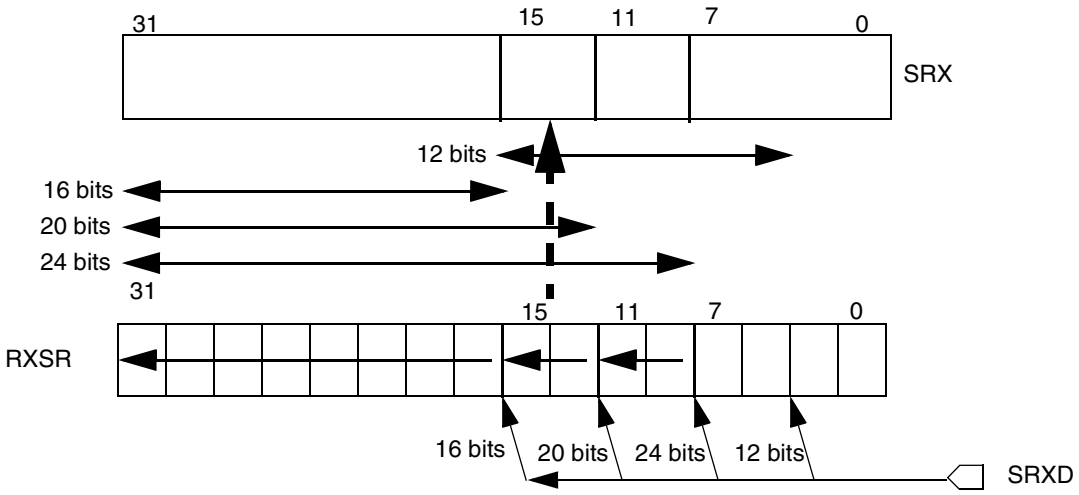


Figure 42-21. Receive Data Path (RXBIT0=0, RSHFD=0) (MSB Alignment)

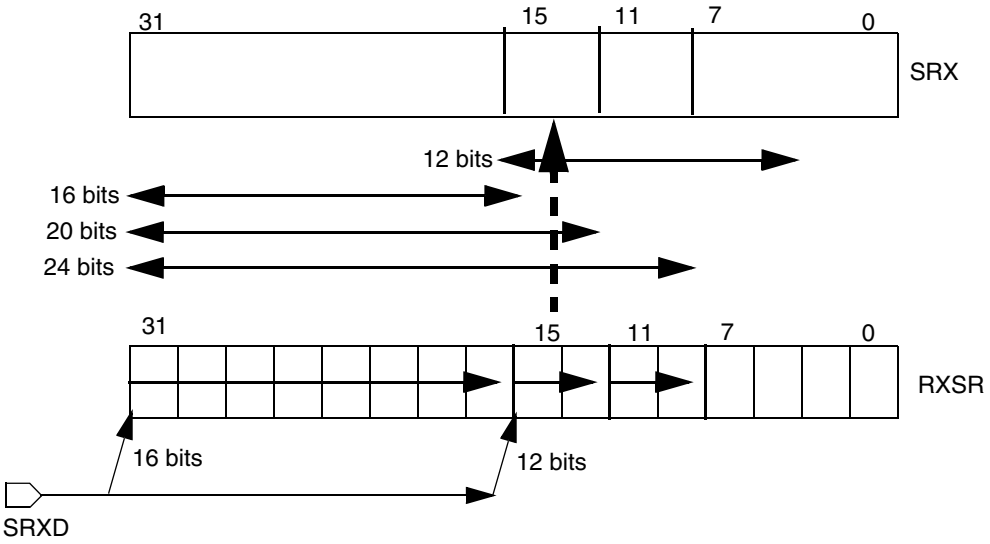


Figure 42-22. Receive Data Path (RXBIT0=0, RSHFD=1) (MSB Alignment)

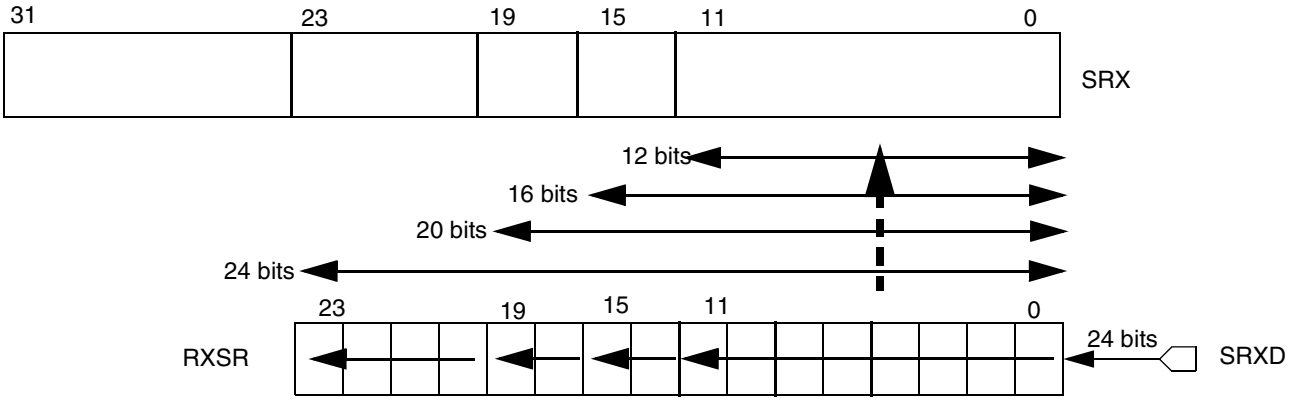


Figure 42-23. Receive Data Path (RXBIT0=1, RSHFD=0) (LSB Alignment)

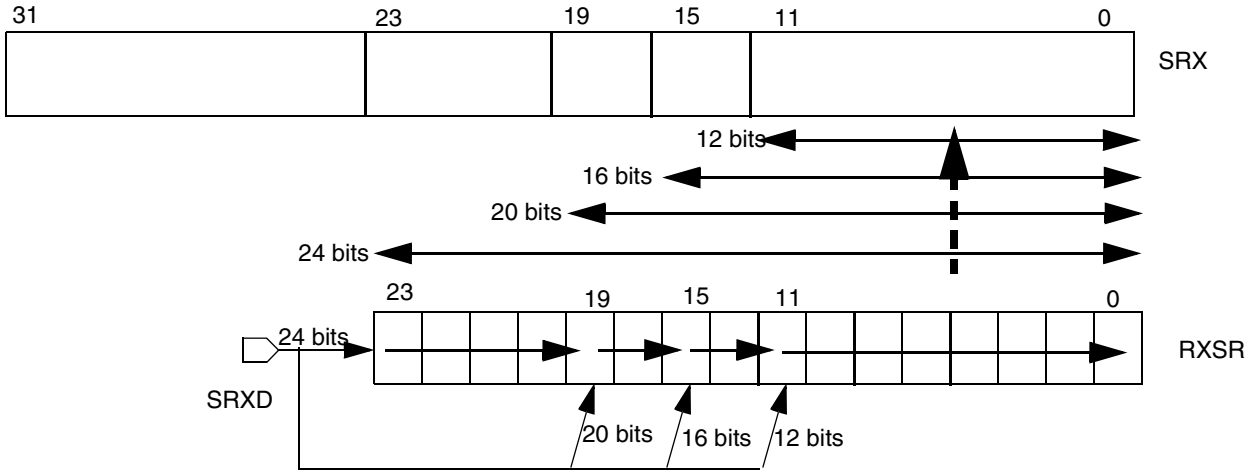


Figure 42-24. Receive Data Path (RXBIT0=1, RSHFD=1) (LSB Alignment)

42.3.3.7 SSI Control Register (SCR)

The SSI Control Register (SCR) is a 10-bit register used to set up the SSI. SSI reset is controlled by bit 0 in the SCR. SSI operating modes are also selected in this register (except AC97 mode, which is selected in SACNT register).

See Figure 42-25 for an illustration of valid bits in SSI Control Register and Table 42-12 for descriptions of the bit fields in the register.

0x1001_1010 (SCR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	CLK_IST	TCH_EN	SYS_CLK_EN	I^2S MODE[1:0]		SYN	NET	RE	TE	SSIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-25. SSI Control Register

Table 42-12. SSI Control Register Field Descriptions

Field	Description
31–10	Reserved
9 CLK_IST	Clock Idle State. This bit controls the idle state of the transmit clock port during SSI internal gated mode. 0 Clock idle state is ‘0’. 1 Clock idle state is ‘1’.
8 TCH_EN	Two-Channel Operation Enable. This bit allows SSI to operate in the two-channel mode. In this mode, 2 time slots should be used out of the possible 32. Any 2 time slots (from 0 to 31) can be selected. The data in the two time slots is alternately handled by the two data registers (0 and 1). While receiving, the RXSR transfers data to SRX0 and SRX1 alternately and while transmitting, data is alternately transferred from STX0 and STX1 to TXSR. If more than 2 slots are to be enabled, then for odd number of slots Two-Channel Operation is deprecated and TCH_EN should be cleared. This will ensure that all data is received and transmitted from one fifo, FIFO0. For an even number of slots, Two-Channel Operation can be enabled to optimize usage of both FIFOs or disabled as in the case of odd number of active slots. 0 Two-channel mode is disabled. 1 Two-channel mode is enabled.
7 SYS_CLK_EN	System Clock Enable. When set, this bit allows the SSI to output the SYS_CLK (ccm_ssi_clk) at the SRCK port, provided that network mode, synchronous mode, and transmit internal clock mode are set. The relationship between bit clock and SYS_CLK is determined by DIV2, PSR, and PM bits. This bit can be used to output the oversampling clock on SRCK port in I2S Master mode. 0 SYS_CLK is not output on the SRCK port. 1 SYS_CLK is output on the SRCK port.
6–5 I^2S MODE[1:0]	I^2S Mode Select. These bits allow the SSI to operate in Normal, I^2S Master or I^2S Slave mode. Refer to Section 42.1.2.4, “I2S Mode” for a detailed description of I^2S Mode of operation. Refer to Table 42-2 for details regarding settings.
4 SYN	Synchronous Mode. This bit controls whether SSI is in synchronous mode or not. In synchronous mode, the transmit and receive sections of SSI share a common clock port (STCK) and frame sync port (STFS). 0 Asynchronous mode is selected. 1 Synchronous mode is selected.
3 NET	Network Mode. This bit controls whether SSI is in network mode or not. 0 Network mode is not selected. 1 Network mode is selected.

Table 42-12. SSI Control Register Field Descriptions (continued)

Field	Description
2 RE	Receive Enable. This control bit enables the receive section of the SSI. When this bit is enabled, data reception starts with the arrival of the next frame sync. If data is being received when this bit is cleared, data reception continues until the end of the current frame and then stops. If this bit is set again before the second to last bit of the last time slot in the current frame, then reception continues without interruption. 0 Receive section is disabled. 1 Receive section is enabled.
1 TE	Transmit Enable. This control bit enables the transmit section of the SSI. It enables the transfer of the contents of the STX registers to the TXSR and also enables the internal transmit clock. The transmit section is enabled when this bit is set and a frame boundary is detected. When this bit is cleared, the transmitter continues to send data until the end of the current frame and then stops. Data can be written to the STX registers with the TE bit cleared (the corresponding TDE bit will be cleared). If the TE bit is cleared and then set again before the second to last bit of the last time slot in the current frame, data transmission continues without interruption. The normal transmit enable sequence is to write data to the STX register(s) and then set the TE bit. The normal disable sequence is to clear the TE and TIE bits after the TDE bit is set. In gated clock mode, clearing the TE bit results in the clock stopping after the data currently in TXSR has shifted out. When the TE bit is set, the clock starts immediately (for internal gated clock mode). 0 Transmit section is disabled. 1 Transmit section is enabled.
0 SSIEN	SSIEN—SSI Enable This bit is used to enable/disable the SSI. When disabled, all SSI status bits are preset to the same state produced by the power-on reset, all control bits are unaffected, the contents of Tx and Rx FIFOs are cleared. When SSI is disabled, all internal clocks are disabled (except register access clock). 0 SSI is disabled. 1 SSI is enabled.

42.3.3.8 SSI Interrupt Status Register (SISR)

The SSI Interrupt Status Register (SISR) is a 19-bit register used to monitor the SSI. This register is read-only and is used by the core to interrogate the status of the SSI. The status bits are described in the following table. Refer to Figure 42-26 for a list of SSI interrupt sources. All Receiver related interrupts are generated only if the Receiver is enabled (SCR[2]=1) and all Transmitter related interrupts are generated only if the Transmitter is enabled (SCR[1]=1).

SSI Interrupt Sources
SSI Receive Data with Exception Status 0/1
SSI Receive Data 0/1
SSI Receive Last Time Slot
SSI Transmit Data with Exception Status 0/1
SSI Transmit Data 0/1
SSI Transmit Last Time Slot
SSI AC97 Command Address Updated
SSI AC97 Command Data Updated
SSI AC97 Receive Tag Updated
SSI Receive Frame Sync

Figure 42-26. SSI Interrupts

NOTE

SSI Status flags are updated when SSI is enabled.

Refer to Section 42.4.3, “Receive Interrupt Enable Bit Description” and Section 42.4.4, “Transmit Interrupt Enable Bit Description” for interrupt source mapping.

All the flags in the SISR are updated after the first bit of the next SSI word has completed transmission or reception. Some status bits (ROE0/1 and TUE0/1) are cleared by reading the SISR followed by a read or write to either the SRX0/1 or STX0/1 registers.

See Figure 42-27 for an illustration of valid bits in SSI Interrupt Register and Table 42-13 for descriptions of the bit fields in the register.

0x1001_1014 (SISR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CMD AU	CMD DU	RXT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RDR1	RDR0	TDE1	TDE0	ROE1	ROE0	TUE1	TUE0	TFS	RFS	TLS	RLS	RFF1	RFF0	TFE1	TFE0
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1

Figure 42-27. SSI Interrupt Status Register

Table 42-13. SSI Interrupt Status Register Field Descriptions

Field	Description
31–19	Reserved
18 CMDAU	Command Address Register Updated. This bit causes the Command Address Updated interrupt (when CMDAU_EN bit is set). This status bit is set each time there is a difference in the previous and current value of the received Command Address. This bit is cleared on reading the SACADD register. 0 No change in SACADD register. 1 SACADD register updated with different value.
17 CMDDU	Command Data Register Updated. This bit causes the Command Data Updated interrupt (when CMDDU_EN bit is set). This status bit is set each time there is a difference in the previous and current value of the received Command Data. This bit is cleared on reading the SACDAT register. 0 No change in SACDAT register. 1 SACDAT register is updated with different value.
16 RXT	Receive Tag Updated. This status bit is set each time there is a difference in the previous and current value of the received tag. It causes the Receive Tag Interrupt (if RXT_EN bit is set). This bit is cleared on reading the SATAG register. 0 No change in SATAG register. 1 SATAG register is updated with different value.
15 RDR1	Receive Data Ready 1. This flag bit is set when SRX1 or Rx FIFO 1 is loaded with a new value and Two-Channel mode is selected. RDR1 is cleared when the Core reads the SRX1 register. If Rx FIFO 1 is enabled, RDR1 is cleared when the FIFO is empty. If RIE and RDR1_EN are set, a Receive Data 1 interrupt request is issued on setting of RDR1 bit in case Rx FIFO1 is disabled, if the FIFO is enabled, the interrupt is issued on RFF1 assertion. The RDR1 bit is cleared by POR and SSI reset. 0 No new data for Core to read. 1 New data for Core to read.
14 RDR0	Receive Data Ready 0. This flag bit is set when SRX0 or Rx FIFO 0 is loaded with a new value. RDR0 is cleared when the Core reads the SRX0 register. If Rx FIFO 0 is enabled, RDR0 is cleared when the FIFO is empty. If RIE and RDR0_EN are set, a Receive Data 0 interrupt request is issued on setting of RDR0 bit in case Rx FIFO0 is disabled, if the FIFO is enabled, the interrupt is issued on RFF0 assertion. The RDR0 bit is cleared by POR and SSI reset. 0 No new data for Core to read. 1 New data for Core to read.
13 TDE1	Transmit Data Register Empty 1. This flag is set whenever data is transferred to TXSR from STX1 register and Two-Channel mode is selected. If Tx FIFO1 is enabled, this occurs when there is at least one empty slot in STX1 or Tx FIFO1. If Tx FIFO1 is not enabled, this occurs when the contents of STX1 are transferred to TXSR. The TDE1 bit is cleared when the Core writes to STX1. If TIE and TDE1_EN are set, an SSI Transmit Data 1 interrupt request is issued on setting of TDE1 bit. The TDE1 bit is cleared by POR and SSI reset. 0 Data is available for transmission. 1 Data needs to be written by the Core for transmission.
12 TDE0	Transmit Data Register Empty 0. This flag is set whenever data is transferred to TXSR from STX0 register. If Tx FIFO 0 is enabled, this occurs when there is at least one empty slot in STX0 or Tx FIFO 0. If Tx FIFO 0 is not enabled, this occurs when the contents of STX0 are transferred to TXSR. The TDE0 bit is cleared when the Core writes to STX0. If TIE and TDE0_EN are set, an SSI Transmit Data 0 interrupt request is issued on setting of TDE0 bit. The TDE0 bit is cleared by POR and SSI reset. 0 Data is available for transmission. 1 Data needs to be written by the Core for transmission.

Table 42-13. SSI Interrupt Status Register Field Descriptions (continued)

Field	Description
11 ROE1	Receiver Overrun Error 1. This flag is set when the RXSR is filled and ready to transfer to SRX1 register or to Rx FIFO 1 (when enabled) and these are already full and Two-Channel mode is selected. If Rx FIFO 1 is enabled, this is indicated by RFF1 flag, else this is indicated by the RDR1 flag. The RXSR is not transferred in this case. The ROE1 flag causes an interrupt if RIE and ROE1_EN are set. The ROE1 bit is cleared by POR and SSI reset. It is also cleared by reading the SISR with ROE1 bit set, followed by reading the SRX1 register. Clearing the RE bit does not affect the ROE1 bit. 0 Default interrupt is issued to the Core. 1 Exception interrupt is issued to the Core.
10 ROE0	Receiver Overrun Error 0. This flag is set when the RXSR is filled and ready to transfer to SRX0 register or to Rx FIFO 0 (when enabled) and these are already full. If Rx FIFO 0 is enabled, this is indicated by RFF0 flag, else this is indicated by the RDR0 flag. The RXSR is not transferred in this case. The ROE0 flag causes an interrupt if RIE and ROE0_EN are set. The ROE0 bit is cleared by POR and SSI reset. It is also cleared by reading the SISR with ROE0 bit set, followed by reading the SRX0 register. Clearing the RE bit does not affect the ROE0 bit. 0 Default interrupt is issued to the Core. 1 Exception interrupt is issued to the Core.
9 TUE1	Transmitter Underrun Error 1. This flag is set when the TXSR is empty (no data to be transmitted), the TDE1 flag is set, a transmit time slot occurs and the SSI is in Two-Channel mode. When a transmit underrun error occurs, the previous data is retransmitted. In Network mode, each time slot requires data transmission (unless masked through STMSK register), when the transmitter is enabled (TE is set). The TUE1 flag causes an interrupt if TIE and TUE1_EN are set. The TUE1 bit is cleared by POR and SSI reset. It is also cleared by reading the SISR with TUE1 bit set, followed by writing to the STX1 register. 0 Default interrupt is issued to the Core. 1 Exception interrupt is issued to the Core.
8 TUE0	Transmitter Underrun Error 0. This flag is set when the TXSR is empty (no data to be transmitted), the TDE0 flag is set and a transmit time slot occurs. When a transmit underrun error occurs, the previous data is retransmitted. In Network mode, each time slot requires data transmission (unless masked through STMSK register), when the transmitter is enabled (TE is set). The TUE0 flag causes an interrupt if TIE and TUE0_EN are set. The TUE0 bit is cleared by POR and SSI reset. It is also cleared by reading the SISR with TUE0 bit set, followed by writing to the STX0 register. 0 Default interrupt is issued to the Core. 1 Exception interrupt is issued to the Core.
7 TFS	Transmit Frame Sync. This flag indicates the occurrence of transmit frame sync. Data written to the STX registers during the time slot when the TFS flag is set, is sent during the second time slot (in Network mode) or in the next first time slot (in Normal mode). In Network mode, the TFS bit is set during transmission of the first time slot of the frame and is then cleared when starting transmission of the next time slot. In Normal mode, this bit is always high. This flag causes an interrupt if TIE and TFS_EN are set. The TFS bit is cleared by POR and SSI reset. 0 No Occurrence of Transmit frame sync. 1 Transmit frame sync occurred during transmission of last word written to STX registers.
6 RFS	Receive Frame Sync. This flag indicates the occurrence of receive frame sync. In Network mode, the RFS bit is set when the first slot of the frame is being received. It is cleared when the next slot begins to be received. In Normal mode, this bit is always high. This flag causes an interrupt if RIE and RFS_EN are set. The RFS bit is cleared by POR and SSI reset. 0 No Occurrence of Receive frame sync. 1 Receive frame sync occurred during reception of next word in SRX registers.

Table 42-13. SSI Interrupt Status Register Field Descriptions (continued)

Field	Description
5 TLS	Transmit Last Time Slot. This flag indicates the last time slot in a frame. When set, it indicates that the current time slot is the last time slot of the frame. TLS is set at the start of the last transmit time slot and causes the SSI to issue an interrupt (if TIE and TLS_EN are set). TLS is cleared when the SISR is read with this bit set. The TLS bit is cleared by POR and SSI reset. 0 Current time slot is not last time slot of frame. 1 Current time slot is the last transmit time slot of frame.
4 RLS	Receive Last Time Slot. This flag indicates the last time slot in a frame. When set, it indicates that the current time slot is the last receive time slot of the frame. RLS is set at the end of the last time slot and causes the SSI to issue an interrupt (if RIE and RLS_EN are set). RLS is cleared when the SISR is read with this bit set. The RLS bit is cleared by POR and SSI reset. 0 Current time slot is not last time slot of frame. 1 Current time slot is the last receive time slot of frame.
3 RFF1	Receive FIFO Full 1. This flag is set when Rx FIFO1 is enabled, the data level in Rx FIFO1 reaches the selected Rx FIFO WaterMark 1 (RFWM1) threshold and the SSI is in Two-Channel mode. The setting of RFF1 only causes an interrupt when RIE and RFF1_EN are set, Rx FIFO1 is enabled and the Two-Channel mode is selected. RFF1 is automatically cleared when the amount of data in Rx FIFO1 falls below the threshold. The RFF1 bit is cleared by POR and SSI reset. When Rx FIFO1 contains 8 words, the maximum it can hold, all further data received (for storage in this FIFO) is ignored until the FIFO contents are read. 0 Space is available in Receive FIFO1. 1 Receive FIFO1 is full.
2 RFF0	Receive FIFO Full 0. This flag is set when Rx FIFO0 is enabled and the data level in Rx FIFO0 reaches the selected Rx FIFO WaterMark 0 (RFWM0) threshold. The setting of RFF0 only causes an interrupt when RIE and RFF0_EN are set and Rx FIFO0 is enabled. RFF0 is automatically cleared when the amount of data in Rx FIFO0 falls below the threshold. The RFF0 bit is cleared by POR and SSI reset. When Rx FIFO0 contains 8 words, the maximum it can hold, all further data received (for storage in this FIFO) is ignored until the FIFO contents are read. 0 Space is available in Receive FIFO0. 1 Receive FIFO0 is full.
1 TFE1	Transmit FIFO Empty 1. This flag is set when Tx FIFO1 is enabled, the data level in Tx FIFO1 falls below the selected Tx FIFO WaterMark 1 (TFWM1) threshold and the Two-Channel mode is selected. The setting of TFE1 only causes an interrupt when TIE and TFE1_EN are set, Tx FIFO1 is enabled and Two-Channel mode is selected. The TFE1 bit is automatically cleared when the data level in Tx FIFO1 becomes more than the amount specified by the watermark bits. The TFE1 bit is set by POR and SSI reset. 0 Transmit FIFO1 has data for transmission. 1 Transmit FIFO1 is empty.
0 TFE0	Transmit FIFO Empty 0. This flag is set when Tx FIFO0 is enabled and the data level in Tx FIFO0 falls below the selected Tx FIFO WaterMark 0 (TFWM0) threshold. The setting of TFE0 only causes an interrupt when TIE and TFE0_EN are set and Tx FIFO0 is enabled. The TFE0 bit is automatically cleared when the data level in Tx FIFO0 becomes more than the amount specified by the watermark bits. The TFE0 bit is set by POR and SSI reset. 0 Transmit FIFO0 has data for transmission. 1 Transmit FIFO0 is empty.

42.3.3.9 SSI Interrupt Enable Register (SIER)

The SSI Interrupt Enable Register (SIER) is a 23-bit register used to set up the SSI interrupts and DMA requests. See Figure 42-28 for an illustration of valid bits in SSI Interrupt Enable Register and Table 42-14 for descriptions of the bit fields in the register.

0x1001_1018 (SIER) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	RDMAE	RIE	TDMAE	TIE	CMDAU_EN	CMDDU_EN	RXT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RDR1_EN	RDR0_EN	TDE1_EN	TDE0_EN	ROE1_EN	ROE0_EN	TUE1_EN	TUE0_EN	TFS_EN	RFS_EN	TLS_EN	RLS_EN	RFF1_EN	RFF0_EN	TFE1_EN	TFE0_EN
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1

Figure 42-28. SSI Interrupt Enable Register

Table 42-14. SSI Interrupt Enable Register Field Descriptions

Field	Description
31–23	Reserved
22 RDMAE	Receive DMA Enable. This bit allows SSI to request for DMA transfers. When enabled, DMA requests are generated when any of the RFF0/1 bits in the SISR are set and if the corresponding RFEN bit is also set. If the corresponding FIFO is disabled, a DMA request is generated when the corresponding RDR bit is set. 0 SSI Receiver DMA requests are disabled. 1 SSI Receiver DMA requests are enabled.
21 RIE	Receive Interrupt Enable. This control bit allows the SSI to issue receiver related interrupts to the Core. Refer to Section 42.4.3, “Receive Interrupt Enable Bit Description” for a detailed description of this bit. 0 SSI Receiver Interrupt requests are disabled. 1 SSI Receiver Interrupt requests are enabled.
20 TDMAE	Transmit DMA Enable. This bit allows SSI to request for DMA transfers. When enabled, DMA requests are generated when any of the TFE0/1 bits in the SISR are set and if the corresponding TFEN bit is also set. If the corresponding FIFO is disabled, a DMA request is generated when the corresponding TDE bit is set. 0 SSI Transmitter DMA requests are disabled. 1 SSI Transmitter DMA requests are enabled.

Table 42-14. SSI Interrupt Enable Register Field Descriptions

Field	Description
19 TIE	Transmit Interrupt Enable. This control bit allows the SSI to issue transmitter data related interrupts to the Core. Refer to Section 42.4.4, "Transmit Interrupt Enable Bit Description" for a detailed description of this bit. 0 SSI Transmitter Interrupt requests are disabled. 1 SSI Transmitter Interrupt requests are enabled.
18–0 Enable Bits	Enable Bit. Each bit controls whether the corresponding status bit in SISR can issue an interrupt to the Core or not. 0 Status bit cannot issue an interrupt. 1 Status bit can issue an interrupt.

42.3.3.10 SSI Transmit Configuration Register (STCR)

The SSI Transmit Configuration Register (STCR) is a read/write control registers used to direct the transmit operation of the SSI. STCR controls the direction of the bit clock and frame sync ports, STCK and STFS. Interrupt enable bit for the transmit sections is provided in this control register. The Power on reset clears all STCR bits. However, SSI reset does not affect the STCR bits. The STCR bits are described in the following paragraphs. See Table 42-7 for the programming model of the SSI. The SSI Control Register (SCR) must first be set to enable interrupts. Next, the SSI interrupt bit in the Interrupt Enable Register (SIER) must be set to enable the interrupt. Finally the interrupt can be enabled from within the SSI.

See Figure 42-29 for an illustration of valid bits in SSI Transmit Configuration Register and Table 42-15 for descriptions of the bit fields in the register.

0x1001_101C (STCR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	TXBIT0	TFEN1	TFEN0	TFDIR	TXDIR	TSHFD	TSCKP	TFSI	TFSL	TEFS
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 42-29. SSI Transmit Configuration Register

Table 42-15. SSI Transmit Configuration Register Field Descriptions

Field	Description
31–10	Reserved
9 TXBIT0	Transmit Bit 0. This control bit allows SSI to transmit the data word from bit position 0 or 15/31 in the transmit shift register. The shifting data direction can be MSB or LSB first, controlled by the TSHFD bit. 0 Shifting with respect to bit 31 (if word length = 16, 18, 20, 22 or 24) or bit 15 (if word length = 8, 10 or 12) of transmit shift register (MSB aligned). 1 Shifting with respect to bit 0 of transmit shift register (LSB aligned).
8 TFEN1	Transmit FIFO Enable 1. This bit enables transmit FIFO 1. When enabled, the FIFO allows 8 samples to be transmitted by the SSI (per channel) (a 9th sample can be shifting out) before TDE1 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is transferred to the transmit shift register (provided the interrupt is enabled). 0 Transmit FIFO 1 disabled. 1 Transmit FIFO 1 enabled.
7 TFEN0	Transmit FIFO Enable 0. This bit enables transmit FIFO 0. When enabled, the FIFO allows eight samples to be transmitted by the SSI per channel (a 9th sample can be shifting out) before TDE0 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is transferred to the transmit shift register (provided the interrupt is enabled). 0 Transmit FIFO 0 disabled. 1 Transmit FIFO 0 enabled.
6 TFDIR	Transmit Frame Direction. This bit controls the direction and source of the transmit frame sync signal. Internally generated frame sync signal is sent out through the STFS port and external frame sync is taken from the same port. 0 Frame Sync is external. 1 Frame Sync generated internally.
5 TXDIR	Transmit Clock Direction. This bit controls the direction and source of the clock signal used to clock the TXSR. Internally generated clock is output through the STCK port. External clock is taken from this port. Refer to Table 42-5 for details of clock pin configurations. 0 Transmit Clock is external. 1 Transmit Clock generated internally.
4 TSHFD	Transmit Shift Direction. This bit controls whether the MSB or LSB will be transmitted first in a sample. **Note:** The CODEC device labels the MSB as bit 0, whereas the Core labels the LSB as bit 0. Therefore, when using a standard CODEC, Core MSB (CODEC LSB) is shifted in first (TSHFD cleared). 0 Data transmitted MSB first. 1 Data transmitted LSB first.
3 TSCKP	Transmit Clock Polarity. This bit controls which bit clock edge is used to clock out data for the transmit section. 0 Data clocked out on rising edge of bit clock. 1 Data clocked out on falling edge of bit clock.
2 TFSI	Transmit Frame Sync Invert. This bit controls the active state of the frame sync I/O signal for the transmit section of SSI. 0 Transmit frame sync is active high. 1 Transmit frame sync is active low.

Table 42-15. SSI Transmit Configuration Register Field Descriptions (continued)

Field	Description
1 TFSL	Transmit Frame Sync Length. This bit controls the length of the frame sync signal to be generated or recognized for the transmit section. The length of a word-long frame sync is same as the length of the data word selected by WL[3:0]. 0 Transmit frame sync is one-word long. 1 Transmit frame sync is one-clock-bit long.
0 TEFS	Transmit Early Frame Sync. This bit controls when the frame sync is initiated for the transmit section. The frame sync signal is deasserted after one bit-for-bit length frame sync and after one word-for-word length frame sync. In case of synchronous operation, the frame sync can also be initiated on receiving the first bit of data. 0 Transmit frame sync initiated as the first bit of data is transmitted. 1 Transmit frame sync is initiated one bit before the data is transmitted.

42.3.3.11 SSI Receive Configuration Register (SRCR)

The SSI Receive Configuration Register (SRCR) is a read/write control registers used to direct the receive operation of the SSI. SRCR controls the direction of the bit clock and frame sync ports, SRCK and SRFS. Interrupt enable bit for the transmit sections is provided in this control register. The Power on reset clears all SRCR bits. However, SSI reset does not affect the SRCR bits. See Figure 42-30 for an illustration of valid bits in SSI Receive Configuration Register and Table 42-16 for descriptions of the bit fields in the register.

0x1001_1020 (SRCR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	RXEXT	RXBIT0	RFEN1	RFEN0	RFDIR	RXDIR	RSHFD	RSCKP	RFSI	RFSL	REFS
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 42-30. SSI Receive Configuration Register

Table 42-16. SSI Receive Configuration Register Field Descriptions

Field	Description
31–11	Reserved
10 RXEXT	Receive Data Extension. This control bit allows SSI to store the received data word in sign extended form. This bit affects data storage only in case received data is LSB aligned (SRCR[9]=1) 0 Sign extension is turned off. 1 Sign extension is turned on.

Table 42-16. SSI Receive Configuration Register Field Descriptions (continued)

Field	Description
9 RXBIT0	Receive Bit 0. This control bit allows SSI to receive the data word at bit position 0 or 15/31 in the receive shift register. The shifting data direction can be MSB or LSB first, controlled by the RSHFD bit. 0 Shifting with respect to bit 31 (if word length = 16, 18, 20, 22 or 24) or bit 15 (if word length = 8, 10 or 12) of receive shift register (MSB aligned). 1 Shifting with respect to bit 0 of receive shift register (LSB aligned).
8 RFEN1	Receive FIFO Enable 1. This bit enables receive FIFO 1. When enabled, the FIFO allows eight samples to be received by the SSI per channel (a 9th sample can be shifting in) before RDR1 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is received by the SSI (provided the interrupt is enabled). 0 Receive FIFO 1 disabled. 1 Receive FIFO 1 enabled.
7 RFEN0	Receive FIFO Enable 0. This bit enables receive FIFO 0. When enabled, the FIFO allows 8 samples to be received by the SSI (per channel) (a 9th sample can be shifting in) before RDR0 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is received by the SSI (provided the interrupt is enabled). 0 Receive FIFO0 is disabled. 1 Receive FIFO0 is enabled.
6 RFDIR	Receive Frame Direction. This bit controls the direction and source of the receive frame sync signal. Internally generated frame sync signal is sent out through the SRFS port and external frame sync is taken from the same port. 0 Frame Sync is external. 1 Frame Sync is generated internally.
5 RXDIR	Receive Clock Direction. This bit controls the direction and source of the clock signal used to clock the RXSR. Internally generated clock is output through the SRCK port. External clock is taken from this port. Refer to Table 42-5 for details on clock pin configurations. 0 Receive Clock is external. 1 Receive Clock is generated internally.
4 RSHFD	Receive Shift Direction. This bit controls whether the MSB or LSB will be received first in a sample. **Note:** The CODEC device labels the MSB as bit 0, whereas the Core labels the LSB as bit 0. Therefore, when using a standard CODEC, Core MSB (CODEC LSB) is shifted in first (RSHFD cleared). 0 Data received is MSB first. 1 Data received is LSB first.
3 RSCKP	Receive Clock Polarity. This bit controls which bit clock edge is used to latch in data for the receive section. 0 Data latched on falling edge of bit clock. 1 Data latched on rising edge of bit clock.
2 RFSI	Receive Frame Sync Invert. This bit controls the active state of the frame sync I/O signal for the receive section of SSI. 0 Receive frame sync is active high. 1 Receive frame sync is active low.

Table 42-16. SSI Receive Configuration Register Field Descriptions (continued)

Field	Description
1 RFSL	Receive Frame Sync Length. This bit controls the length of the frame sync signal to be generated or recognized for the receive section. The length of a word-long frame sync is same as the length of the data word selected by WL[3:0]. 0 Receive frame sync is one-word long. 1 Receive frame sync is one-clock-bit long.
0 REFS	Receive Early Frame Sync. This bit controls when the frame sync is initiated for the receive section. The frame sync is disabled after one bit-for-bit length frame sync and after one word-for-word length frame sync. 0 Receive frame sync is initiated one bit before the data is received. 1 Receive frame sync initiated as the first bit of data is received.

42.3.3.12 SSI Transmit and Receive Clock Control Registers (STCCR and SRCCR)

The SSI Transmit and Receive Control (STCCR and SRCCR) registers are 19-bit, read/write control registers used to direct the operation of the SSI. The Clock and Reset Module (CRM) can source the SSI clock (ccm_ssi_clk) from multiple sources and perform fractional division to support commonly used audio bit rates. The CRM can maintain the ccm_ssi_clk frequency at a constant rate even in cases where the ipg_clk frequency changes. These registers control the SSI clock generator, bit and frame sync rates, word length, and number of words per frame for the serial data. The STCCR register is dedicated to the transmit section, and the SRCCR register is dedicated to the receive section except in Synchronous mode, in which the STCCR register controls both the receive and transmit sections. Power-on reset clears all STCCR and SRCCR bits. SSI reset does not affect the STCCR and SRCCR bits. The control bits are described in the following paragraphs. Although the bit patterns of the STCCR and SRCCR registers are the same, the contents of these two registers can be programmed differently.

See Figure 42-31 for an illustration of valid bits in STCCR and SRCCR, and refer to Table 42-17 for descriptions of the bit fields for both SSI Transmit and Receive Clock Control registers.

0x1001_1024 (STCCR)
0x1001_1028 (SRCCR)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-31. SSI Transmit and Receive Clock Control Registers

Table 42-17. SSI Transmit and Receive Clock Control Register Field Descriptions

Field	Description
31–19	Reserved
18 DIV2	Divide By 2. This bit controls a divide-by-two divider in series with the rest of the prescalers. 0 Divider bypassed. 1 Divider used to divide clock by 2.
17 PSR	Prescaler Range. This bit controls a fixed divide-by-eight prescaler in series with the variable prescaler. It extends the range of the prescaler for those cases where a slower bit clock is required. 0 Prescaler bypassed. 1 Prescaler used to divide clock by 8.
16–13 WL3–WL0	Word Length Control. These bits are used to control the length of the data words being transferred by the SSI. These bits control the Word Length Divider in the Clock Generator. They also control the frame sync pulse length when the FSL bit is cleared. In I2S Master mode, the SSI works with a fixed word length of 32, and the WL bits are used to control the amount of valid data in those 32 bits. Refer to Table 42-18 for details of data word lengths supported by SSI. In AC97 Mode of operation, if word length is set to any value other than 16 bits, it will result in a word length of 20 bits.

Table 42-18. SSI Data Length

WL3	WL2	WL1	WL0	Number of Bits/Word	Supported in Implementation
0	0	0	0	2	No
0	0	0	1	4	No
0	0	1	0	6	No
0	0	1	1	8	Yes
0	1	0	0	10	Yes
0	1	0	1	12	Yes
0	1	1	0	14	No
0	1	1	1	16	Yes
1	0	0	0	18	Yes
1	0	0	1	20	Yes
1	0	1	0	22	Yes
1	0	1	1	24	Yes
1	1	0	0	26	No
1	1	0	1	28	No
1	1	1	0	30	No
1	1	1	1	32	No

Table 42-17. SSI Transmit and Receive Clock Control Register Field Descriptions

Field	Description
12–8 DC4–DC0	Frame Rate Divider Control. These bits are used to control the divide ratio for the programmable frame rate dividers. The divide ratio works on the word clock. In Normal mode, this ratio determines the word transfer rate. In Network mode, this ratio sets the number of words per frame. The divide ratio ranges from 1 to 32 in Normal mode and from 2 to 32 in Network mode. In Normal mode, a divide ratio of 1 (DC=00000) provides continuous periodic data word transfer. A bit-length frame sync must be used in this case. These bits can be programmed with values ranging from “00000” to “11111” to control the number of words in a frame.
7–0 PM7–PM0	Prescaler Modulus Select. These bits control the prescale divider in the clock generator. This prescaler is used only in Internal Clock mode to divide the internal clock (ccm_ssi_clk). The bit clock output is available at the clock port. A divide ratio from 1 to 256 (PM[7:0] = 0x00 to 0xFF) can be selected. Refer to Section 42.4.2.2, “DIV2, PSR and PM Bit Description” for details regarding settings.

42.3.3.13 SSI FIFO Control/Status Register (SFCSR)

See Figure 42-32 for an illustration of valid bits in SSI FIFO Control/Status Register and Table 42-19 for descriptions of the bit fields in the register.

0x1001_102C (SFCSR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RFCNT1[3:0]				TFCNT1[3:0]				RFWM1[3:0]				TFWM1[3:0]			
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RFCNT0[3:0]				TFCNT0[3:0]				RFWM0[3:0]				TFWM0[3:0]			
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

Figure 42-32. SSI FIFO Control/Status Register

Table 42-19. SSI FIFO Control/Status Register Field Descriptions

Field	Description
31–28 RFCNT1[3:0]	Receive FIFO Counter1. These bits indicate the number of data words in Receive FIFO 1. Settings for receive FIFO counter bits: 0000 0 data word in receive FIFO 0001 1 data word in receive FIFO 0010 2 data word in receive FIFO 0011 3 data word in receive FIFO 0100 4 data word in receive FIFO 0101 5 data word in receive FIFO 0110 6 data word in receive FIFO 0111 7 data word in receive FIFO 1000 8 data word in receive FIFO
27–24 TFCNT1[3:0]	Transmit FIFO Counter1. These bits indicate the number of data words in Transmit FIFO. Settings for transmit FIFO counter bits: 0000 0 data word in receive FIFO 0001 1 data word in receive FIFO 0010 2 data word in receive FIFO 0011 3 data word in receive FIFO 0100 4 data word in receive FIFO 0101 5 data word in receive FIFO 0110 6 data word in receive FIFO 0111 7 data word in receive FIFO 1000 8 data word in receive FIFO
23–20 RFWM1[3:0]	Receive FIFO Full WaterMark 1. These bits control the threshold at which the RFF1 flag will be set. The RFF1 flag is set whenever the data level in Rx FIFO 1 reaches the selected threshold. Settings for receive FIFO watermark bits: 0000 Reserved 0001 RFF set when at least one data word have been written to the Receive FIFO. Set when RxFIFO = 1,2,3,4,5,6,7,8 data words 0010 RFF set when more than or equal to 2 data word have been written to the Receive FIFO. Set when RxFIFO = 2,3,4,5,6,7,8 data words 0011 RFF set when more than or equal to 3 data word have been written to the Receive FIFO. Set when RxFIFO = 3,4,5,6,7,8 data words 0100 RFF set when more than or equal to 4 data word have been written to the Receive FIFO. Set when RxFIFO = 4,5,6,7,8 data words 0101 RFF set when more than or equal to 5 data word have been written to the Receive FIFO. Set when RxFIFO = 5,6,7,8 data words 0110 RFF set when more than or equal to 6 data word have been written to the Receive. Set when RxFIFO = 6,7,8 data words 0111 RFF set when more than or equal to 7 data word have been written to the Receive FIFO. Set when RxFIFO = 7,8 data words 1000 RFF set when 8 data word have been written to the Receive FIFO (default). Set when RxFIFO = 8 data words

Table 42-19. SSI FIFO Control/Status Register Field Descriptions (continued)

Field	Description
19–16 TFWM1[3:0]	Transmit FIFO Empty WaterMark 1. These bits control the threshold at which the TFE1 flag will be set. The TFE1 flag is set whenever the data level in Tx FIFO 1 falls below the selected threshold. Settings for transmit FIFO watermark bits: 0000 Reserved 0001 TFE set when there are more than or equal to 1 empty slots in Transmit FIFO. (default) Transmit FIFO empty is set when TxFIFO <= 7 data. 0010 TFE set when there are more than or equal to 2 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 6 data. 0011 TFE set when there are more than or equal to 3 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 5 data. 0100 TFE set when there are more than or equal to 4 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 4 data. 0101 TFE set when there are more than or equal to 5 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 3 data. 0110 TFE set when there are more than or equal to 6 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 2 data. 0111 TFE set when there are more than or equal to 7 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 1 data. 1000 TFE set when there are 8 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO=0 data.
15–12 RFCNT0[3:0]	Receive FIFO Counter 0. These bits indicate the number of data words in Receive FIFO 0.
11–8 TFCNT0[3:0]	Transmit FIFO Counter 0. These bits indicate the number of data words in Transmit FIFO 0.
7–4 RFWM0[3:0]	Receive FIFO Full WaterMark 0. These bits control the threshold at which the RFF0 flag will be set. The RFF0 flag is set whenever the data level in Rx FIFO 0 reaches the selected threshold.
3–0 TFWM0[3:0]	Transmit FIFO Empty WaterMark 0. These bits control the threshold at which the TFE0 flag will be set. The TFE0 flag is set whenever the data level in Tx FIFO 0 falls below the selected threshold.

Table 42-20 indicates the status of the Transmit FIFO empty flag, with different settings of the Transmit FIFO WaterMark bits and varying amounts of data in the Tx FIFO.

Table 42-20. Status of Transmit FIFO Empty Flag

Transmit FIFO Watermark (TFWM)	Number of data in TXFIFO								
	0	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1	0
2	1	1	1	1	1	1	1	0	0
3	1	1	1	1	1	1	0	0	0
4	1	1	1	1	1	0	0	0	0
5	1	1	1	1	0	0	0	0	0
6	1	1	1	0	0	0	0	0	0
7	1	1	1	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0

42.3.3.14 SSI Test Register (STR)

NOTE

SSI Test Register is designed for debugging purpose only, and is not intended for general user access.

See Figure 42-33 for an illustration of valid bits in SSI Test Register and Table 42-21 for descriptions of the bit fields for the register.

0x1001_1030 (STR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TEST	RCK2 TCK	RFS2 TFS	RXSTATE[4:0]					TXD2 RXD	TCK2 RCK	TFS2 RFS	TXSTATE[4:0]				
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Figure 42-33. SSI Test Register

Table 42-21. SSI Test Register Field Descriptions

Field	Description
31–16	Reserved
15 TEST	Test Mode. This bit enables the test features of SSI. When set, the RXSTATE and TXSTATE bit values get transferred to the state machine. When in test mode, the user can read the contents of a specific FIFO by writing an appropriate value to the RFCNT0/1 or TFCNT 0/1 bits (in SFCSR). 0 No effect on SSI operation. 1 SSI in Test Mode.
14 RCK2TCK	Receive Clock to Transmit Clock Loop Back. This bit connects SRCK (used as output) to STCK (used as input). 0 No effect on SSI operation. 1 SRCK to STCK loop back enabled.
13 RFS2TFS	Receive Frame to Transmit Frame Loop Back. This bit connects SRFS (used as output) to STFS (used as input). 0 No effect on SSI operation. 1 SRFS to STFS loop back enabled.
12–8 RXSTATE[4:0]	Receiver State Machine Status. These bits indicate the current status of the Receive State Machine.
7 TXD2RXD	Transmit Data to Receive Data Loop Back. This bit connects STXD to SRXD. 0 No effect on SSI operation. 1 STXD to SRXD loop back enabled.

Table 42-21. SSI Test Register Field Descriptions (continued)

Field	Description
6 TCK2RCK	Transmit Clock to Receive Clock Loop Back. This bit connects STCK (used as output) to SRCK (used as input). 0 No effect on SSI operation. 1 STCK to SRCK loop back enabled.
5 TFS2RFS	Transmit Frame to Receive Frame Loop Back. This bit connects STFS (used as output) to SRFS (used as input). 0 No effect on SSI operation. 1 STFS to SRFS loop back enabled.
4–0 TXSTATE	Transmitter State Machine Status. These bits indicate the current status of the Transmit State Machine.

42.3.3.15 SSI Option Register (SOR)

NOTE

SSI Option Register is designed for debugging purpose only, and is not intended for general user access.

See Figure 42-34 for an illustration of valid bits in SSI Option Register and Table 42-22 for descriptions of the bit fields for the register.

0x1001_1034 (SOR) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	CLKOFF	RX_CLR	TX_CLR	INIT	WAIT[1:0]		SYNRST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-34. SSI Option Register

Table 42-22. SSI Option Register Field Descriptions

Field	Description
31–7	Reserved
6 CLKOFF	Clock Off. This bit is used to turn off the ipg_clk to further reduce power consumption. 0 No effect on SSI operation. 1 Turn off ipg_clk.

Table 42-22. SSI Option Register Field Descriptions (continued)

Field	Description
5 RX_CLR	Receiver Clear. This bit flushes the contents of Rx FIFOs. It is always read as cleared (‘0’). 0 No effect on SSI operation. 1 Flush Rx FIFOs.
4 TX_CLR	Transmitter Clear This bit flushes the contents of Tx FIFOs. It is always read as cleared (‘0’). 0 No effect on SSI operation. 1 Flush Tx FIFOs.
3 INIT	Initialize. The setting of this bit causes the SSI state machine to reset. 0 No effect on SSI operation. 1 Initialize SSI state machine.
2–1 WAIT[1:0]	Wait. These bits control the number wait states to be added to all transactions between the Core and SSI. The value of these bits ranges from ‘00’ (no wait states) to ‘11’ (three wait states).
0 SYNRST	Frame Sync Reset. This bit automatically resets the accumulation of data in Receive Data Registers (SRX0/1) and Receive FIFOs (RXFIFO 0/1) on the next frame synchronization. 0 Data accumulation not affected. 1 Reset data accumulation on Frame Synchronization.

42.3.3.16 SSI AC97 Control Register (SACNT)

See Figure 42-35 for an illustration of valid bits in SSI AC97 Control Register and Table 42-23 for descriptions of the bit fields for the register.

0x1001_1038 (SACNT) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	FRDIV[5:0]						WR	RD	TIF	FV	AC97 EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-35. SSI AC97 Control Register

Table 42-23. SSI AC97 Control Register Field Descriptions

Field	Description
31–11	Reserved
10–5 FRDIV[5:0]	Frame Rate Divider. These bits control the frequency of AC97 data transmission/reception. They are programmed with the number of frames for which the SSI should be idle, after operating in one frame. Through these bits, AC97 frequency of operation, from 48 KHz (000000) to 1 KHz (101111) can be achieved. Sample Value: 001010 (10 Decimal) = SSI will operate once every 11 frames.
4 WR	Write Command. This bit specifies whether the next frame will carry an AC97 Write Command or not. The programmer should take care that only one of the bits (WR or RD) is set at a time. When this bit is set, the corresponding tag bits (corresponding to Command Address and Command Data slots of the next Tx frame) are automatically set. This bit is automatically cleared by the SSI after completing transmission of a frame. 0 Next frame will not have a Write Command. 1 Next frame will have a Write Command.
3 RD	Read Command. This bit specifies whether the next frame will carry an AC97 Read Command or not. The programmer should take care that only one of the bits (WR or RD) is set at a time. When this bit is set, the corresponding tag bit (corresponding to Command Address slot of the next Tx frame) is automatically set. This bit is automatically cleared by the SSI after completing transmission of a frame. 0 Next frame will not have a Read Command. 1 Next frame will have a Read Command.
2 TIF	Tag in FIFO. This bit controls the destination of the information received in AC97 tag slot (Slot #0). 0 Tag info stored in SATAG register. 1 Tag info stored in Rx FIFO 0.
1 FV	Fixed/Variable Operation. This bit selects whether the SSI is in AC97 Fixed mode or AC97 Variable mode. 0 AC97 Fixed Mode. 1 AC97 Variable Mode.
0 AC97EN	AC97 Mode Enable. This bit is used to enable SSI AC97 operation. Refer to Section 42.1.2.5 for details of AC97 operation. 0 AC97 mode disabled. 1 SSI in AC97 mode.

42.3.3.17 SSI AC97 Command Address Register (SACADD)

See Figure 42-36 for an illustration of valid bits in SSI AC97 Command Address Register and Table 42-24 for descriptions of the bit fields for the register.

0x1001_103C (SACADD) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	SACADD[18:16]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SACADD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-36. SSI AC97 Command Address Register

Table 42-24. SSI AC97 Command Address Register Field Descriptions

Field	Description
31–19	Reserved
18–0 SACADD	AC97 Command Address. These bits store the Command Address Slot information (bit 19 of the slot is sent in accordance with the Read and Write Command bits in SACNT register). These bits can be updated by a direct write from the Core. They are also updated with the information received in the incoming Command Address Slot. If the contents of these bits change due to an update, the CMDAU bit in SISR is set.

42.3.3.18 SSI AC97 Command Data Register (SACDAT)

See Figure 42-37 for an illustration of valid bits in SSI AC97 Command Data Register and Table 42-25 for descriptions of the bit fields for the register.

0x1001_1040 (SACDAT) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	SACDAT[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SACDAT[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-37. SSI AC97 Command Data Register

Table 42-25. SSI AC97 Command Data Register

Field	Description
31–20	Reserved
19–0 SACDAT	AC97 Command Data. The outgoing Command Data Slot carries the information contained in these bits. These bits can be updated by a direct write from the Core. They are also updated with the information received in the incoming Command Data Slot. If the contents of these bits change due to an update, the CMDDU bit in SISR is set. In case of an AC97 Read Command, these bits are cleared for transmission in Time Slot #2 of AC97 frame.

42.3.3.19 SSI AC97 Tag Register (SATAG)

See Figure 42-38 for an illustration of valid bits in SSI AC97 Tag Register and Table 42-26 for descriptions of the bit fields for the register.

0x1001_1044 (SATAG) Access: User Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SATAG[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-38. SSI AC97 Tag Register

(continued)

Table 42-26. SSI AC97 Tag Register Field Descriptions

Field	Description
31–16	Reserved
15–0 SATAG	AC97 Tag Value. Writing to this register (by the Core) sets the value of the Tx-Tag in AC97 fixed mode of operation. On a read, the Core gets the Rx-Tag Value received (in the last frame) from the CODEC. If TIF bit in SACNT register is set, the TAG value is also stored in Rx-FIFO in addition to SATAG register. When the received Tag value changes, the RXT bit in SISR register is set. Bits SATAG[1:0] convey the CODEC-ID. In current implementation only Primary CODECs are supported. Thus writing value 2'b00 to this field is mandatory.

42.3.3.20 SSI Transmit Time Slot Mask Register (STMSK)

See Figure 42-39 for an illustration of valid bits in SSI Transmit Time Slot Register and Table 42-27 for descriptions of the bit fields for the register.

0x1001_1048 (STMSK) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STMSK[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STMSK[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-39. SSI Transmit Time Slot Mask Register

Table 42-27. SSI Transmit Time Slot Mask Register Field Descriptions

Field	Description
31–0 STMSK	Transmit Mask. These bits indicate which slot has been masked in the current frame. The Core can write to this register to control the time slots in which the SSI transmits data. Each bit has info corresponding to the respective time slot in the frame. If a change is made to the register contents, the transmission pattern is updated from the next time slot. Transmit mask bits should not be used in I2S Slave mode of operation. 0 Valid Time Slot. 1 Time Slot masked (no data transmitted in this time slot).

42.3.3.21 SSI Receive Time Slot Mask Register (SRMSK)

See Figure 42-40 for an illustration of valid bits in SSI Receive Time Slot Mask Register and Table 42-28 for descriptions of the bit fields for the register.

0x1001_104C (SRMSK) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SRMSK[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SRMSK[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-40. SSI Receive Time Slot Mask Register

Table 42-28. SSI Receive Time Slot Mask Register Field Descriptions

Field	Description
31–0 SRMSK	Receive Mask. These bits indicate which slot has been masked in the current frame. The Core can write to this register to control the time slots in which the SSI receives data. Each bit has info corresponding to the respective time slot in the frame. If a change is made to the register contents, the reception pattern is updated from the next time slot. Receive mask bits should not be used in I2S Slave mode of operation. 0 Valid Time Slot. 1 Time Slot masked (no data received in this time slot).

42.3.3.22 SSI AC97 Channel Status Register (SACCST)

See Figure 42-41 for an illustration of valid bits in SSI AC97 Channel Status Register and Table 42-29 for descriptions of the bit fields for the register.

0x1001_1050 (SACCST) Access: User read-only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	SACCST[9:0]									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-41. SSI AC97 Channel Status Register

Table 42-29. SSI AC97 Channel Status Register Field Descriptions

Field	Description
9–0 SACCST	AC97 Channel Status. These bits indicate which data slot has been enabled in AC97 variable mode operation. This register is updated in case the core enables/disables a channel through a write to SACCEN/SACCDIS register or the external CODEC enables a channel by sending a ‘1’ in the corresponding SLOTREQ bit. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). The contents of this register only have relevance while the SSI is operating in AC97 variable mode. Writes to this register result in an error response on the IP interface. 0 Data channel is disabled. 1 Data channel is enabled.

42.3.3.23 SSI AC97 Channel Enable Register (SACCEN)

See Figure 42-42 for an illustration of valid bits in SSI AC97 Channel Enable Register and Table 42-30 for descriptions of the bit fields for the register.

0x1001_1054 (SACCEN) Access: User read-only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W							SACCEN[9:0]									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-42. SSI AC97 Channel Enable Register

Table 42-30. SSI AC97 Channel Enable Register Field Descriptions

Field	Description
9–0 SACCEN	AC97 Channel Enable Register. The Core writes a ‘1’ to these bits to enable an AC97 data channel. Writing a ‘0’ has no effect. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). Writes to these bits only have effect in the AC97 Variable mode of operation. These bits are always read as ‘0’ by the Core. 0 Write has no effect. 1 Write enables the corresponding data channel.

42.3.3.24 SSI AC97 Channel Disable Register (SACCDIS)

See Figure 42-43 for an illustration of valid bits in SSI AC97 Channel Disable Register and Table 42-31 for descriptions of the bit fields for the register.

0x1001_1058 (SACCEN) Access: User read-only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W							SACCDIS[9:0]									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-43. SSI AC97 Channel Disable Register

Table 42-31. SSI AC97 Channel Disable Register Field Descriptions

Field	Description
9–0 SACCDIS	AC97 Channel Disable Register. The Core writes a '1' to these bits to disable an AC97 data channel. Writing a '0' has no effect. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). Writes to these bits only have effect in the AC97 Variable mode of operation. These bits are always read as '0' by the Core. 0 Write has no effect. 1 Write disables the corresponding data channel.

42.4 Functional Description

42.4.1 SSI Architecture

The Synchronous Serial Interface (SSI) is connected to chip pads through the Digital Audio MUX (AUDMUX) module. The AUDMUX can be configured to connect the SSI module to the chip pads in various ways. Refer to Figure 42-1 for a block diagram of the SSI.

42.4.2 SSI Clocking

The SSI uses the following clocks:

- Bit clock—Used to serially clock the data bits in and out of the SSI port. This clock is either generated internally (from ccm_ssi_clk) or taken from external clock source (through the Tx/Rx clock ports).
- Word clock—Used to count the number of data bits per word (8, 10, 12, 16, 18, 20, 22 or 24 bits). This clock is generated internally from the bit clock.
- Frame clock (Frame Sync)—Used to count the number of words in a frame. This signal can be generated internally from the bit clock, or taken from external source (from the Tx/Rx frame sync ports).
- System clock—In master mode, this is an integer multiple of frame clock. This is ccm_ssi_clk. It is used in cases when SSI has to provide the clock.

Care should be taken to ensure that the bit clock frequency (either internally generated by dividing the ccm_ssi_clk or sourced from external device through Tx/Rx clock ports) is never greater than 1/4 of the ipg_clk frequency.

In Normal mode (SCR[6:5]=00), the bit clock, used to serially clock the data, is visible on the Serial Transmit Clock (STCK) and Serial Receive Clock (SRCK) ports. The word clock is an internal clock used to determine when transmission of an 8, 10, 12, 16, 18, 20, 22, or 24 bit word has completed. The word clock in turn then clocks the frame clock, which counts the number of words in the frame. The frame clock can be viewed on the STFS and SRFS frame sync ports, because a frame sync is generated after the correct number of words in the frame have passed. In master and synchronous mode, the unused port SRCK is used as Serial System Clock (SYS_CLK) enabled by the SCR register bit 15, SYS_CLK_EN. This Serial System Clock is an oversampling clock of the frame sync clock (STFS). In this mode, the word length (WL), Prescaler Range (PSR), Prescaler Modulus (PM) and Frame rate (DC) selects the ratio of SYS_CLK to sampling clock STFS. In case of I^2S mode, the oversampling clock ccm_ssi_clk can be made available

on this port if the SYS_CLK_EN bit is set. The relationship between the clocks and the dividers is shown in Figure 42-44. The bit clock can be received from an SSI clock port or can be generated from the ccm_ssi_clk through a divider, as shown in Figure 42-45.

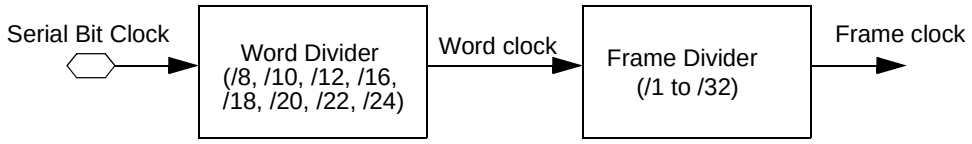


Figure 42-44. SSI Clocking

42.4.2.1 SSI Clock and Frame Sync Generation

Data clock and frame sync signals can be generated internally, or can be obtained from external sources. If internally generated, the SSI clock generator is used to derive bit clock and frame sync signals from the ccm_ssi_clk clock. The SSI clock generator consists of a selectable, fixed prescaler and a programmable prescaler for bit rate clock generation. In Gated Clock mode, the data clock is valid only when data is being transmitted. Otherwise the clock port is pulled to the inactive state. A programmable frame rate divider and a word length divider are used for frame rate sync signal generation.

Figure 42-45 shows a block diagram of the clock generator for the transmit section. The serial bit clock can be internal or external, depending on the Transmit Direction (TXDIR) bit in the SSI Transmit Configuration Register (STCR). The receive section contains an equivalent clock generator circuit.

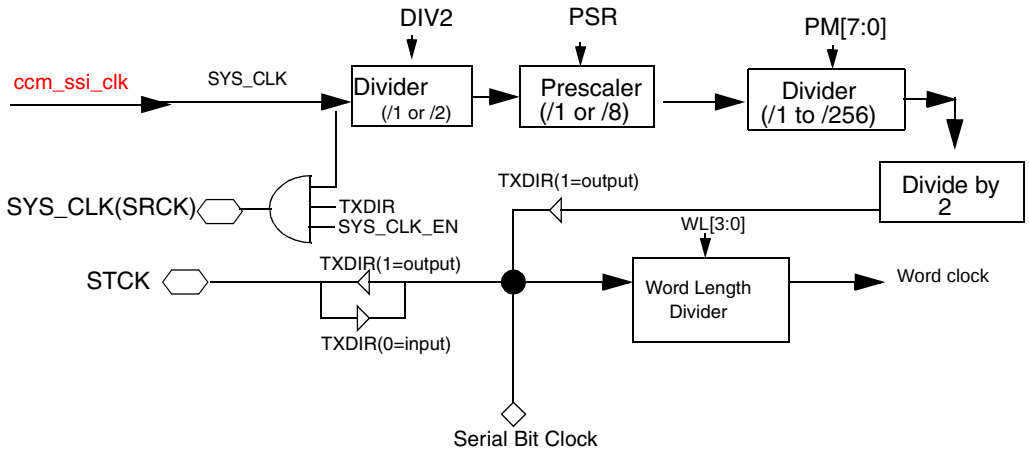


Figure 42-45. SSI Transmit Clock Generator Block Diagram

Figure 42-46 shows the Frame Sync Generator block for the transmit section. When internally generated, both receive and transmit frame sync are generated from the word clock and are defined by the Frame Rate Divider (DC[4:0]) bits and the Word Length (WL[3:0]) bits of the SSI Transmit Clock Control Register (STCCR). The receive section contains an equivalent circuit for the Frame Sync Generator.

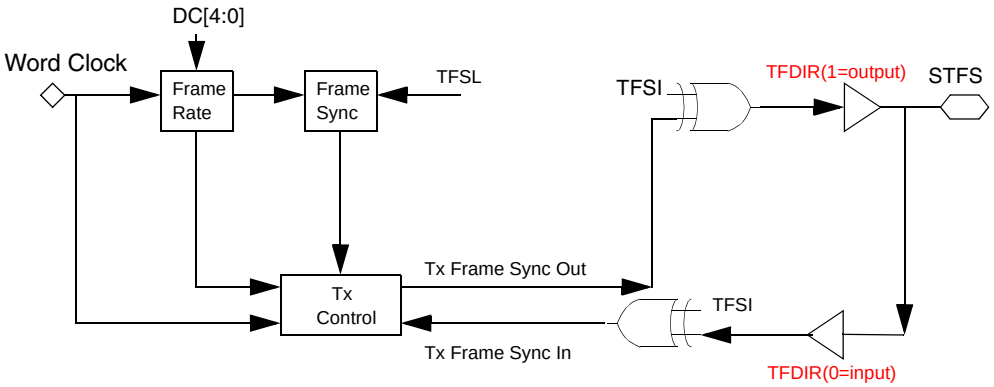


Figure 42-46. SSI Transmit Frame Sync Generator Block Diagram

42.4.2.2 DIV2, PSR and PM Bit Description

The bit clock frequency can be calculated from the SSI Serial System Clock (ccm_ssi_clk), using the equation in Figure 42-47.

NOTE

You must ensure that the bit-clock frequency must be l4 times the ipg_clk period. The oversampling clock frequency can go up to ipg_clk frequency. Bits DIV2, PSR and PM should not be all set to zero at the same time.

$$f_{INT\_BIT\_CLK} = f_{SYS\_CLK} / [(DIV2 + 1) \times (7 \times PSR + 1) \times (PM + 1) \times 2]$$

where PM=PM[7:0]

$$f_{FRAME\_SYN\_CLK} = (f_{INT\_BIT\_CLK}) / [(DC + 1) \times WL]$$

where DC = DC[4:0] and WL = 8, 10, 12, 16, 18, 20, 22, 24

Figure 42-47. SSI Bit Clock Equation

For example, if the SSI working clock SYS_CLK (ccm_ssi_clk) is 12.288Mhz, in 8-bit word Normal mode with DC[4:0] set to 1 (00001), PM[7:0] set to 47 (0010 1111), the PSR bit cleared, DIV2 bit set to 1, a bit clock rate of 12.288 Mhz /[1 x 4 x 48] = 64 kHz is generated. Since the 8-bit word rate is equal to one(that is, normal mode), the sampling rate (FS rate) would then be 64 kHz / [1 * 8] = 8kHz.

In next example, the SYS_CLK (ccm_ssi_clk) clock is 11.2896Mhz. A 16-bit word Network mode with DC[4:0] set to 1 (00001), PM[7:0] set to 3(0000 0011), the PSR bit is set to 0, DIV2 bit set to 0, and a 11.2896MHz SYS_CLK clock, a bit clock rate of 11.2896 Mhz 3/ [1 x 2x 4] = 1.4112MHz is generated. Since the 16-bit word rate is equal to two, the sampling rate (FS rate) would be 1.4112 MHz / [2 * 16] = 44.1 kHz.

Table 42-32 below shows programming examples for the clock dividers in CRM and SSI to support various bit clock (STCK) frequencies.

Table 42-32. SSI Bit Clock and Frame Rate as a Function of PSR, PM and DIV2

Bits/ Word	Words/ Frame	Ideal Frame rate (kHz)	PLL Freq (MHz)	SSIDIV (in CRM)	MCLK/ ccm_ssi_clk Freq (MHz)	DIV2	PSR	PM	WL	DC	Actual Bit_Clk Freq (kHz) STCK	Required Bit_Clk Freq (kHz) STCK	Error (Hz)
16	1	8	294.912	48	12.288	0	0	47	7	0	128	128	0
16	2	8	294.912	48	12.288	0	0	23	7	1	256	256	0
16	4	8	294.912	48	12.288	0	0	11	7	3	512	512	0
16	1	12	294.912	48	12.288	0	0	31	7	0	192	192	0
16	2	12	294.912	48	12.288	0	0	15	7	1	384	384	0
16	4	12	294.912	48	12.288	0	0	7	7	3	768	768	0
16	1	16	294.912	48	12.288	0	0	23	7	0	256	256	0
16	2	16	294.912	48	12.288	0	0	11	7	1	512	512	0
16	4	16	294.912	48	12.288	0	0	5	7	3	1024	1024	0
16	1	24	294.912	48	12.288	0	0	15	7	0	384	384	0
16	2	24	294.912	48	12.288	0	0	7	7	1	768	768	0
16	4	24	294.912	48	12.288	0	0	3	7	3	1536	1536	0
16	1	32	294.912	48	12.288	0	0	11	7	0	512	512	0
16	2	32	294.912	48	12.288	0	0	5	7	1	1024	1024	0
16	4	32	294.912	48	12.288	0	0	2	7	3	2048	2048	0
16	1	48	294.912	48	12.288	0	0	15	7	0	768	768	0
16	2	48	294.912	48	12.288	0	0	3	7	1	1536	1536	0
16	4	48	294.912	48	12.288	0	0	1	7	3	3072	3072	0
16	1	11.025	270.9504	48	11.2896	0	0	31	7	0	176.4	176.4	0
16	2	11.025	270.9504	48	11.2896	0	0	15	7	1	352.8	352.8	0
16	4	11.025	270.9504	48	11.2896	0	0	7	7	3	705.6	705.6	0
16	1	22.05	270.9504	48	11.2896	0	0	15	7	0	352.8	352.8	0
16	2	22.05	270.9504	48	11.2896	0	0	7	7	1	705.6	705.6	0
16	4	22.05	270.9504	48	11.2896	0	0	3	7	3	1411.2	1411.2	0
16	1	44.1	270.9504	48	11.2896	0	0	7	7	0	705.6	705.6	0
16	2	44.1	270.9504	48	11.2896	0	0	3	7	1	1411.2	1411.2	0
16	4	44.1	270.9504	48	11.2896	0	0	1	7	3	2822.4	2822.4	0

NOTE

The above table describes how various frame rates can be achieved with the PLL 0/1 supplying a frequency of 294.912 MHz and 270.9504 MHz (with WL and DC settings as shown). Using PLL2 requires that these input frequencies be lowered by a factor of 2 (and the dividers be changed accordingly). Using the MRCG allows the input frequency to be lowered by a factor of 4 (provided the dividers are changed accordingly). These clocks are recommended as convenient starting points but the system allows for other input clock frequencies as well. The table should be treated as an example and should be updated with the targeted and achievable values specific to the chip.

Table 42-34 below shows programming of the CRM and SSI dividers in order to generate the appropriate SYS_CLK and BIT_CLK frequencies for various sampling rates.In these examples, the master mode is selected either by setting I^2S master bit (SCR[6:5]=01) or individually programming the SSI in network, synchronous, transmit internal mode (the table specifically illustrates the I^2S mode frequencies/sample rates). The SYS_CLK clock is ccm_ssi_clk. The fixed I^2S frame rate of 64 bits per frame (word length (WL) can be any value) and DC of 1 are assumed.

Table 42-33. SSI System Clock, Bit Clock, Frame Clock in Master Mode

Bits/ Word	Words/ Frame	Ideal Sampling Rate (kHz)	Over Sampling Rate	PLL Freq (MHz)	SSIDIV (in CRM)	Target MCLK Freq (Mhz)	Actual MCLK Freq (Mhz)	Bits/ Word	Words/ Frame	Actual Sampling Rate (kHz) STFS	Error (Hz)
32	2	22.05	512	270.9504	48	11.2896	11.2896	32	2	44.117	17.65
32	2	24	512	294.912	48	12.288	12.288	32	2	22.058	8.82
32	2	32	384	294.912	48	12.288	12.288	32	2	11.029	4.41
32	2	32	512	294.912	36	16.384	16.384	32	2	48.000	0

NOTE

The above table describes how various frame rates can be achieved with the PLL0/1 supplying a frequency of 294.912 MHz and 270.9504 MHz. Using PLL2 requires that these input frequencies be lowered by a factor of 2 (and the dividers be changed accordingly). Using the MRCG allows the input frequency to be lowered by a factor of 4 (provided the dividers are changed accordingly). These clocks are recommended as convenient starting points but the system allows for other input clock frequencies as well. The above table should be treated as an example and should be updated with the targeted and achievable values specific to the chip.

42.4.3 Receive Interrupt Enable Bit Description

When the RIE and RE bit are set the program controller is interrupted when either of the SSI Receive FIFO Full (RFF0/1) bits in SISR is set (if the corresponding Receive FIFO is enabled). If Receive FIFO is not enabled, interrupt is generated when the corresponding SSI Receive Data Ready (RDR0/1) bit in the SISR is set. When the receive FIFO is enabled, maximum 8 values are available to be read (8 values per channel in Two Channel mode). If not enabled, then one value can be read from the SRX register (one each in case of Two Channel mode). If the RIE bit is cleared, these interrupt are disabled. However, the RFF0/1 and RDR0/1 bits still indicate the receive data register full condition. Reading the SRX registers clears the RDR bits, thus clearing the pending interrupt. Two receive data interrupts (two per channel in case of Two Channel mode) are available: receive data with exception status, and receive data without exception. Table 42-34 and Table 42-35 show the conditions under which these interrupts are generated.

Table 42-34. SSI Receive Data 1 Interrupts

Interrupt	RIE	ROE1	RFF1/RDR1
Receive Data 1 (with Exception Status)	1	1	1
Receive Data 1 (without exception)	1	0	1

Table 42-35. SSI Receive Data 0 Interrupts

Interrupt	RIE	ROE0	RFF0/RDR0
Receive Data 0 (with Exception Status)	1	1	1
Receive Data 0 (without exception)	1	0	1

42.4.4 Transmit Interrupt Enable Bit Description

The SSI Transmit Interrupt Enable (TIE) control bit allows interrupting the program controller for data transfer requirements of the SSI transmitter. When the TIE and TE bits are set, the program controller is interrupted when either of the SSI Transmit FIFO Empty (TFE0/1) flags in SISR are set (if corresponding Transmit FIFO is enabled). If the corresponding Transmit FIFO is not enabled, interrupt is generated when the corresponding SSI Transmit Data Register Empty (TDE0/1) flag in the SISR is set and Transmit Enable (TE) bit is set.

When Transmit FIFO 0 is enabled, 8 values can be written to the SSI (8 per channel in case of Two Channel mode, using Tx FIFO 1). If not enabled, then one value can be written to the STX0 register (one per channel in case of Two Channel mode, using STX1). When the TIE bit is cleared, all transmit interrupts are disabled. However, the TDE0/1 bits always indicate the corresponding STX register empty condition, even when the transmitter is disabled by the Transmit Enable (TE) bit (in the SCR). Writing data to the STX clears the corresponding TDE bit, thus clearing the interrupt. Two transmit data interrupts are available (four in case of Two Channel mode, two per channel): transmit data with exception status and transmit data without exceptions.Table 42-36 and Table 42-37 show the conditions under which these interrupts are generated.

Table 42-36. SSI Transmit Data 1 Interrupts

Interrupt	TIE	TUE1	TFE1/TDE1
Transmit Data 1 (with Exception Status)	1	1	1
Transmit Data 1 (without exception)	1	0	1

Table 42-37. SSI Transmit Data 0 Interrupts

Interrupt	TIE	TUE0	TFE0/TDE0
Transmit Data 0 (with Exception Status)	1	1	1
Transmit Data 0 (without exception)	1	0	1

42.4.5 IP Bus Interface

The SSI has an IP Bus interface compliant with SRS 3.0.2 in order to provide a control and data interface. This interface is used by both the processor and DMA controller.

42.4.5.1 Transfer Lengths Supported

The IP Bus interface of the SSI only supports 32-bit transfers with all SSI registers other than STX0, STX1, SRX0, and SRX1 (that is, the data registers). With the exception of the data registers, using 8-bit and 16-bit transactions could result in undesired behavior but will not result in a transfer bus error. The data registers (STX0, STX1, SRX0, and SRX1) support 8-bit, 16-bit, and 32-bit transfer lengths without restrictions.

42.4.5.2 Transfer Bus Errors

Transfer bus errors are generated upon response to the following:

- Write transfer to a read-only register.
- Read or write access to a register space beyond the SSI's last populated register in its memory map (up until the end of the SSI's allocated memory address range).

42.4.5.3 Clock Rate

The IP Bus clock frequency must be at least four times the serial bit clock frequency.

42.5 Initialization/Application Information

The SSI is affected by the following types of reset:

- Power-on Reset—The Power-on reset is generated by asserting the RESET port. The Power-on reset clears the SSIEN bit in SCR, which disables the SSI. All other status and control bits in the SSI are affected as described in SSI Programming Model in Section 42.3, "Memory Map and Register Definition."
- SSI Reset—The SSI reset is generated when the SSIEN bit in the SCR is cleared. The SSI status bits are preset to the same state produced by the Power-on reset. The SSI control bits are

unaffected. The control bits in the SCR are also unaffected. The SSI reset is useful for selective reset of the SSI without changing the present SSI control bits and without affecting the other peripherals.

The correct sequence to initialize the SSI is as follows:

1. Issue a Power-on or SSI reset (SCR[SSIEN]=0).
2. Disable SSI clocks (ipg_clk, ccm_ssi_clk).
3. Set all control bits for configuring the SSI (refer to Table 42-38).
4. Enable appropriate interrupts/DMA requests through SIER.
5. Set the SCR[SSIEN] bit (=1) to enable the SSI.
6. Enable SSI clocks (ipg_clk, ccm_ssi_clk), as required.
7. In case of AC97 mode, set the SACNT[AC97EN] bit after programming the SATAG register (if needed, for AC97 Fixed mode).
8. Set SCR[TE/RE] bits.

To ensure proper operation of the SSI, the programmer should use the Power-on or SSI reset before changing any of the following control bits listed in Table 42-38.

NOTE

These control bits should not be changed when SSI is enabled.

Table 42-38. SSI Control Bits Requiring SSI to be Disabled Before Change

Control Register	Bit
SCR	[9]=CLK_IST [8]=TCH_EN [7]=SYS_CLK_EN [6:5]=I^2S_MODE [4]=SYN [3]=NET
SIER	[22]=RDMAE [20]=TDMAE

Table 42-38. SSI Control Bits Requiring SSI to be Disabled Before Change (continued)

Control Register	Bit
SRCR STCR	[9]=RXBIT0 [9]=TXBIT0 [8]=RFEN1 [8]=TFEN1 [7]=RFEN0 [7]=TFEN0 [6]=RFDIR [6]=TFDIR [5]=RXDIR [5]=TXDIR [4]=RSHFD [4]=TSHFD [3]=RSCKP [3]=TSCKP [2]=RFSI [2]=TFSI [1]=RFSL [1]=TFSL [0]=REFS [0]=TEFS
SRCCR STCCR	[16]=WL3 [15]=WL2 [14]=WL1 [13]=WL0
SACNT	[1]=FV [10:5]=FRDIV

Chapter 43
Liquid Crystal Display Controller (LCDC)

The Liquid Crystal Display Controller (LCDC) provides display data for external gray-scale or color LCD panels. The LCDC is capable of supporting black-and-white, gray-scale, passive-matrix color (passive color or CSTN), and active-matrix color (active color or TFT) LCD panels.

43.1 Features

The LCDC provides the following features:

- Configurable AHB bus width (32-bit/64-bit).
- Support for single (non-split) screen monochrome or color LCD panels and self-refresh type LCD panels
- 16 simultaneous gray-scale levels from a palette of 16 for monochrome display
- Support for:
  - Maximum resolution of 800 × 600
  - Passive color panel:
    - 4 (mapped to RGB444) / 8 (mapped to RGB444) / 12 (RGB444) bits per pixel (bpp)
  - TFT panel:
    - 4 (mapped to RGB666) / 8 (mapped to RGB666) / 12 (RGB444) / 16 (RGB565) / 18 (RGB666) bpp
  - 16 and 256 colors out of a palette of 4096 colors for 4 bpp and 8 bpp CSTN display respectively
  - 16 and 256 colors out of a palette of 256K colors for 4 bpp and 8 bpp TFT display respectively
  - True 4096 colors for a 12 bpp display
  - True 64K colors for 16 bpp
  - True 256K colors for 18 bpp
  - 16-bit AUO TFT LCD Panel.
  - 24-bit AUO TFT LCD Panel.

Additional support details are shown in Table 43-1.

Table 43-1. Supported Panel Characteristics

Panel Type	Bit/Pixel	Panel Interface (Bits)	Number of Gray Level/Color
Monochrome	1	1, 2, 4, 8	black-and-white
	2	1, 2, 4, 8	4 out of palette of 16
	4	1, 2, 4, 8	16

Table 43-1. Supported Panel Characteristics (continued)

Panel Type	Bit/Pixel	Panel Interface (Bits)	Number of Gray Level/Color
CSTN	4, 8	12	16, 256 out of palette of 4096
	12	12	4096
TFT	4, 8	18	16, 256 out of palette of 256K
	12, 16, 18	12, 16, 18	4096, 64K, 256K

- Standard panel interface for common LCD drivers
- Panel interface of 1-, 2-, 4-, 8-bit for monochrome panels
- Panel interface of 12-, 16-, 18-bit for color panels
- For 4 bpp and 8 bpp, a palette table is used for re-mapping of data from memory, independent of the type of panel used. For 1 bpp, 2 bpp, 12 bpp, 16 bpp and 18 bpp, the palette table is by-passed.
- Interface to passive and active color panel (TFT)
- Supports timing requirements for Sharp 240 × 320 HR-TFT panel
- Hardware-generated cursor with blink, color, and size programmability
- Logical operation between color hardware cursor and background
- Hardware panning (soft horizontal scrolling)
- 8-bit pulse-width modulator for software contrast control
- Graphic window support for viewfinder function in color display
- Graphic window color keying for graphical hardware cursor
- 256 transparency levels for alpha blending between graphic window and background plane

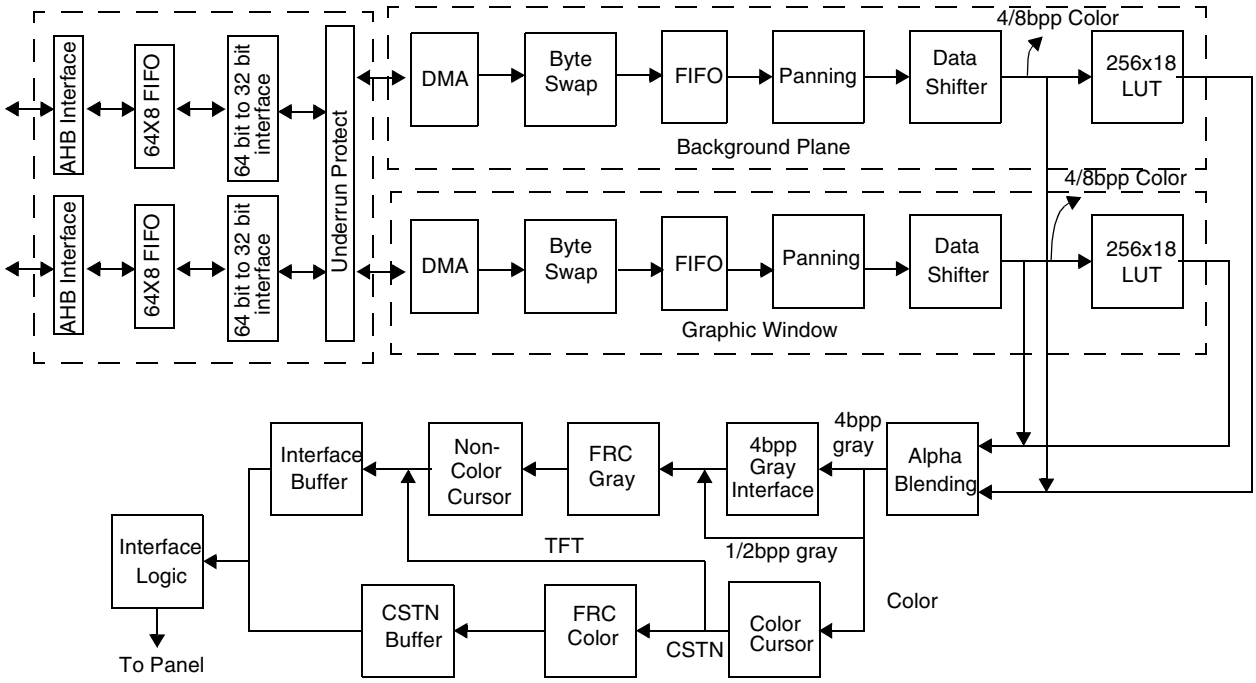


Figure 43-1. LCDC Block Diagram

43.2 Overview

The following sections describe the operation of LCDC with various industry standard LCD displays.

43.2.1 LCD Screen Format

The number of pixels forming the screen width and screen height of the LCD panel are software programmable. Figure 43-2 shows the relationship between the screen size and memory window.

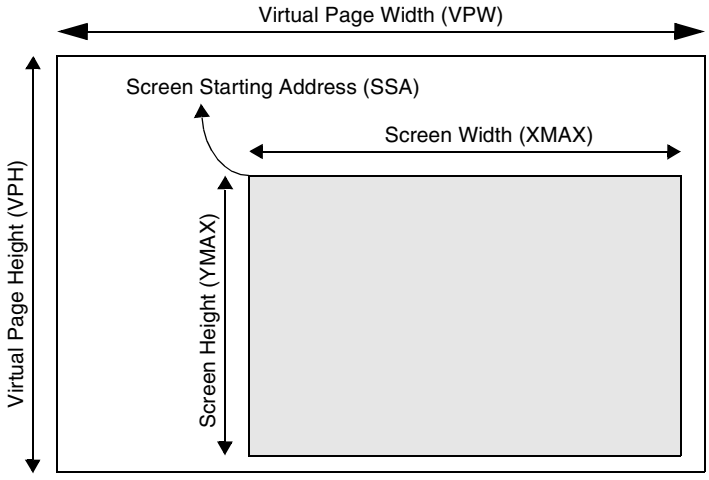


Figure 43-2. LCD Screen Format

Screen width (XMAX) and screen height (YMAX) parameters specify the LCD panel size. LCDC begins scanning the display memory at the location pointed by the screen starting address (SSA) register, represented by the shaded area in Figure 43-2, for display on the LCD panel. Maximum page width is specified by the virtual page width (VPW) parameter. Virtual page height (VPH) does not affect the LCDC and is limited only by memory size. By changing the SSA register, a screen-sized window can be vertically or horizontally scrolled (panned) anywhere inside the virtual page boundaries. Software must control the starting address in the SSA properly so that the scanning logic's system memory pointer (SMP) stays within the VPW and VPH limits to prevent display of strange artifacts on the screen. VPH is used by the programmer only for boundary checks. There is no VPH parameter internal to LCDC. VPW is used to calculate the RAM starting address representing the beginning of each displayed line. SSA sets the address of data for the first line of a frame. For each subsequent line, VPW is added to an accumulation initialized by the SSA to yield the starting address of that line.

43.2.2 Graphic Window on Screen

Graphic window is supported in LCD color panel screen for viewfinder and graphic hardware cursor functions. Similar to screen, virtual page width, graphic window start address and graphic window width and height are software programmable. Graphic window position on screen is specified by the graphic window position register. Figure 43-3 shows how the graphic window is configured and placed on the screen. Graphic window and background plane can be alpha blended. The alpha value is window basis which means all the pixels in the graphic window have the same level of transparency. There are a total of 256 levels of transparency to be configured. In addition, one of the pixel colors can be chosen for color keying in which the selected pixel color in the graphics window is made totally transparent. One of the applications can be a graphical hardware cursor.

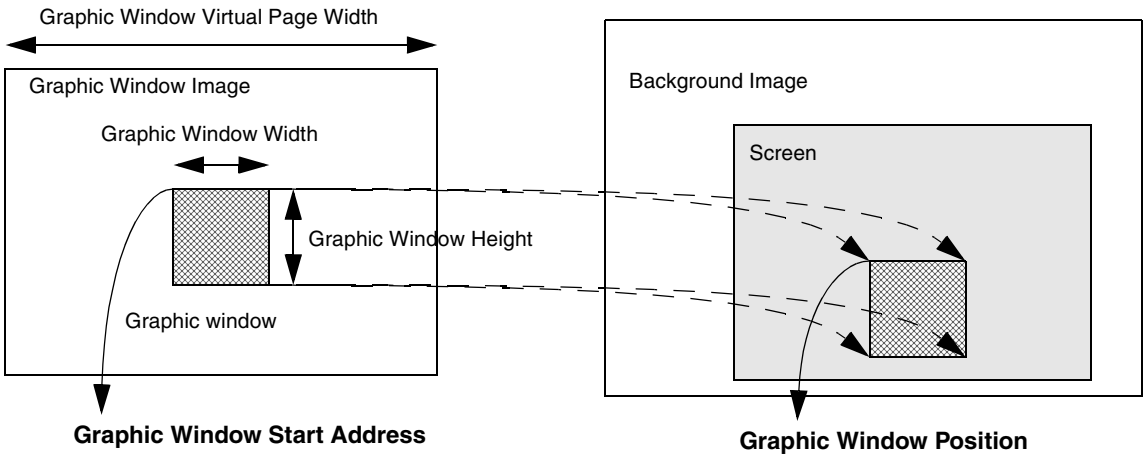


Figure 43-3. Graphic Window on Screen

NOTE

Graphic window and background images must have the same bpp setting.

43.2.3 Panning

Panning offset (POS) is expressed in bits, not pixels, so when operating in any mode other than 1 bpp, only even pixel boundaries are valid. In 12 bpp mode, pixels are aligned to 16-bit boundaries, and POS also must align to these boundaries. SSA and POS are located in isolated registers and are double buffered because they are dynamic parameters, likely to change when LCDC is running. New values of SSA and POS do not take effect until the beginning of next frame. A typical panning algorithm includes an interrupt at the beginning of frame. In the interrupt service routine, POS and/or SSA are updated (old values are internally latched). The updates take effect on the next frame.

43.2.4 Display Data Mapping

LCDC supports 1/2/4 bpp in monochrome mode and 4/8/12/16/18 bpp in color mode. System memory data mapping in 2/4/8/12/16/18 bpp modes is shown in Figure 43-4.

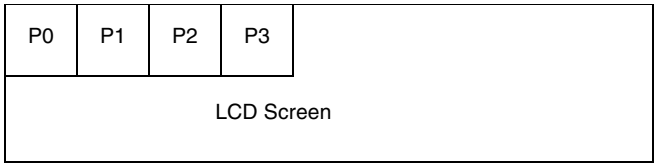


Figure 43-4. Pixel Location on Display Screen

NOTE

In 12 bpp mode, 16 bits of memory are used for each set of 12-bits, to leave 4-bits unused. In 18 bpp mode, 32 bits of memory are used for each pixel, leaving 14-bits unused. Refer to Figure 43-5.

1 bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P24	P25	P26	P27	P28	P29	P30	P31
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P16	P17	P18	P19	P20	P21	P22	P23
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P8	P9	P10	P11	P12	P13	P14	P15
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P0	P1	P2	P3	P4	P5	P6	P7

2 bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P12		P13		P14		P15	
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P8		P9		P10		P11	
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P4		P5		P6		P7	
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P0		P1		P2		P3	

4 bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P6				P7			
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P4				P5			
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P2				P3			
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P0				P1			

8 bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P3							
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P2							
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P1							
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P0							

12 bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
					Red1 [3]	Red1 [2]	Red1 [1]	Red1 [0]
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	Green1 [3]	Green1 [2]	Green1 [1]	Green1[0]	Blue1 [3]	Blue1 [2]	Blue1 [1]	Blue1 [0]
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
					Red0 [3]	Red0 [2]	Red0 [1]	Red0 [0]
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Green0 [3]	Green0 [2]	Green0 [1]	Green0 [0]	Blue0 [3]	Blue0 [2]	Blue0 [1]	Blue0 [0]

16 bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	Red1 [4]	Red1 [3]	Red1 [2]	Red1 [1]	Red1 [0]	Green1 [5]	Green1 [4]	Green1 [3]
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	Green1 [2]	Green1 [1]	Green1 [0]	Blue1 [4]	Blue1 [3]	Blue1 [2]	Blue1 [1]	Blue1 [0]
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Red0 [4]	Red0 [3]	Red0 [2]	Red0 [1]	Red0 [0]	Green0 [5]	Green0 [4]	Green0 [3]
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Green0 [2]	Green0 [1]	Green0 [0]	Blue0 [4]	Blue0 [3]	Blue0 [2]	Blue0 [1]	Blue0 [0]

Figure 43-5. Display Data Mapping 1 bpp Through 16 bpp Mode

Normal 18bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	Red [5]	Red [4]	Red [3]	Red [2]	Red [1]	Red [0]		
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Green [5]	Green [4]	Green [3]	Green [2]	Green [1]	Green [0]		
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Blue [5]	Blue [4]	Blue [3]	Blue [2]	Blue [1]	Blue [0]		

Microsoft PAL_BGR 18bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	Blue [5]	Blue [4]	Blue [3]	Blue [2]	Blue [1]	Blue [0]		
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Green [5]	Green [4]	Green [3]	Green [2]	Green [1]	Green [0]		
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Red [5]	Red [4]	Red [3]	Red [2]	Red [1]	Red [0]		

Figure 43-6. Display Data Mapping for 18 bpp Mode

AUS 24 bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	Blue[7]	Blue[6]	Blue [5]	Blue [4]	Blue [3]	Blue [2]	Blue [1]	Blue [0]
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Breen[7]	Green[6]	Green [5]	Green [4]	Green [3]	Green [2]	Green [1]	Green [0]
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Red[7]	Red[6]	Red [5]	Red [4]	Red [3]	Red [2]	Red [1]	Red [0]

Figure 43-7. Display Data Mapping for 24 bpp Mode

43.2.5 Black-and-White Operation

The 1 bpp mode is also known as black-and-white mode because each pixel is always either fully on or fully off.

43.2.6 Gray-Scale Operation

The LCDC generates a maximum of 16 gray levels. These gray levels are defined by 2 or 4 bits of display data for each pixel. Using 2 bpp, LCDC displays 4 shades of gray. Using 4 bpp, LCDC displays all 16 shades. The shades of gray are obtained by controlling the number of frames in which the pixel is “on” over a period of 16 frames. This method is known as frame rate control (FRC). For more information on FRC, see Section 43.2.8, “Frame Rate Modulation Control (FRC).” Mapping RAM use is shown in

Figure 43-8. While using 2 bpp, 2-bit code is mapped to one of the four gray levels, and while using 4 bpp, 4-bit code is mapped to one of the 16 gray levels. Because crystal formulations and driving voltages vary, visual gray effect may or may not be linearly related to the frame rate. A logarithmic scale such as 0, 1/4, 1/2 and 1 might be more pleasing than a linearly spaced scale such as 0, 5/16, 11/16 and 1 for certain graphics. Figure 43-8 illustrates gray-scale pixel generation. The flexible mapping scheme allows user to optimize the visual effect for a specific panel or application.

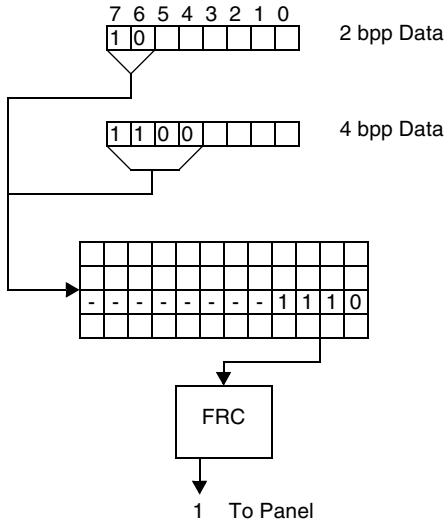


Figure 43-8. Gray-Scale Pixel Generation

43.2.7 Color Generation

Figure 43-9 shows an overview of an active matrix color pixel generation.

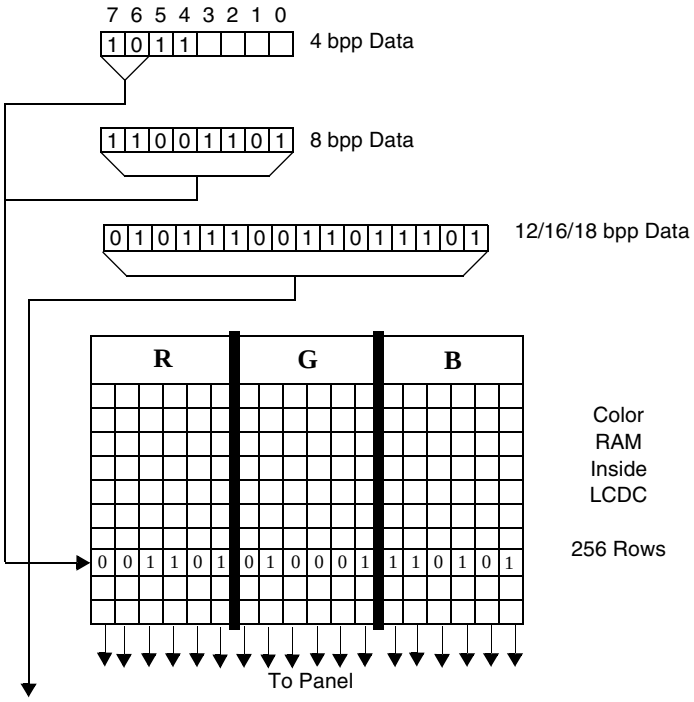


Figure 43-9. Active Matrix Color Pixel Generation

The value corresponding to each color pixel on the screen is represented by a 4-bit, 8-bit, 12-bit, 16-bit or 18-bit code in the display memory. For 4-bit and 8-bit modes, LCDC's color mapping RAM is used to map the data to a 12-bit and 18-bit RGB code for passive and active matrix color displays respectively. For 4-bit and 8-bit passive matrix color displays, 12-bit RGB code from mapping RAM is output to the FRC blocks that independently process the code corresponding to the red, green and blue components of each pixel to generate the required shade and intensity.

For 4-bit and 8-bit active matrix display, 18-bit output from mapping RAM is output to the panel. For 12-bit mode for passive matrix color display, mapping RAM is by-passed and output directly to the FRC block. For 12-bit, 16-bit and 18-bit active matrix color display, pixel data is simply moved from display memory to the LCDC output bus.

Figure 43-9 and Figure 43-10 illustrates active matrix and passive matrix color pixel generation.

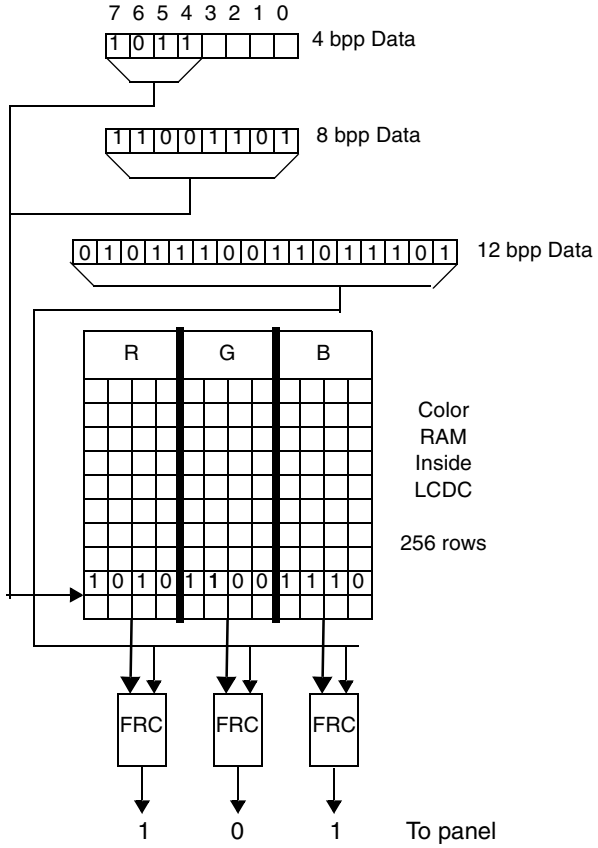


Figure 43-10. Passive Matrix Color Pixel Generation

43.2.8 Frame Rate Modulation Control (FRC)

Circuitry inside the LCDC generates intermediate gray-scale colors on the panel by adjusting the density of zeroes and ones that appear over the frames. LCDC can generate 16 simultaneous gray-scale levels.

Table 43-2. Gray Palette Density

Gray Code (Hexadecimal)	Density	Density (Decimal)
0	0	0
1	1/8	0.125
2	1/5	0.2
3	1/4	0.25
4	1/3	$0.\overline{333}$
5	2/5	0.4
6	4/9	$0.\overline{444}$
7	1/2	0.5
8	5/9	$0.\overline{555}$

Table 43-2. Gray Palette Density (continued)

Gray Code (Hexadecimal)	Density	Density (Decimal)
9	3/5	0.6
A	2/3	$0.\overline{666}$
B	3/4	0.75
C	4/5	0.8
D	7/8	0.875
E	14/15	$0.9\overline{33}$
F	1	1
Note: Overbars indicate repeating decimal numbers.		

43.2.9 Panel Interface Signals and Timing

LCDC continuously provides pixel data to the LCD panel via LCD panel interface. Panel interface signals are illustrated in Figure 43-11. Format, timing and polarity of panel interface signals are programmable. There are two basic modes, passive and active, selected by the TFT register bit. The user must also select either gray scale mode or color mode.

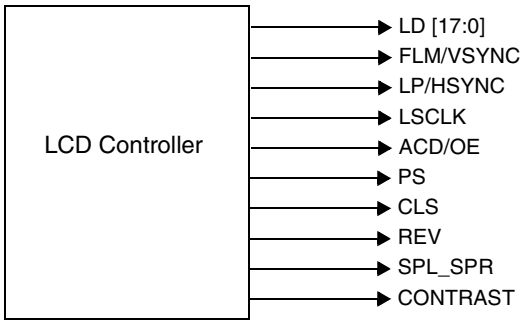


Figure 43-11. LCDC Interface Signals

SPL_SPR, PS, CLS and REV are other interface signals from LCDC. However, these signals are dedicated for Sharp HR-TFT 240 × 320 panels only.

43.2.9.1 Pin Configuration for LCDC

Figure 43-11 shows the signals used for the LCDC. These pins are multiplexed with other functions on the device, and must be configured for LCDC operation before they can be used.

NOTE

The user must ensure that data direction bits in GPIO are set to the correct direction for proper operation.

Table 43-3. Pin Configuration

Pin	Setting	Configuration Procedure
ACD/OE	Primary function of GPIO Port A[31]	1. Clear bit 31 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 31 of Port A General Purpose Register (GPR_A)
CONTRAST	Primary function of GPIO Port A[30]	1. Clear bit 30 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 30 of Port A General Purpose Register (GPR_A)
FLM/VSYNC	Primary function of GPIO Port A[29]	1. Clear bit 29 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 29 of Port A General Purpose Register (GPR_A)
LP/HSYNC	Primary function of GPIO Port A[28]	1. Clear bit 28 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 28 of Port A General Purpose Register (GPR_A)
SPL_SPR	Primary function of GPIO Port A[27]	1. Clear bit 27 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 27 of Port A General Purpose Register (GPR_A)
PS	Primary function of GPIO Port A[26]	1. Clear bit 26 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 26 of Port A General Purpose Register (GPR_A)
CLS	Primary function of GPIO Port A[25]	1. Clear bit 25 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 25 of Port A General Purpose Register (GPR_A)
REV	Primary function of GPIO Port A[24]	1. Clear bit 24 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 24 of Port A General Purpose Register (GPR_A)
LD [17:0]	Primary function of GPIO Port A[23:6]	1. Clear bits [23:6] of Port A GPIO In Use Register (GIUS_A) 2. Clear bits [23:6] of Port A General Purpose Register (GPR_A)
LSCLK	Primary function of GPIO Port A[5]	1. Clear bit 5 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 5 of Port A General Purpose Register (GPR_A)

43.2.9.2 Passive Matrix Panel Interface Signals

Figure 43-12 shows LCD interface timing for monochrome panels and Figure 43-13 shows LCD interface timing for passive matrix color panels. Signal polarities are shown positive, however it can be reversed by clearing the bits in Panel Configuration Register (PCR). Data bus timing for passive panels is controlled by shift clock (LSCLK), line pulse (LP), first line marker (FLM), alternate crystal direction (ACD) and line data (LD) signals. Operation of the panel interface is accomplished in the following steps:

1. LSCLK clocks the pixel data into the display driver's internal shift register.
2. LP signifies the end of current line of serial data and latches the shifted pixel data into a wide latch.
3. FLM marks the first line of the displayed page. LD (and the associated LP), enclosed by the FLM signal, marks the first line of the current frame.
4. ACD toggles after a pre-programmed number of FLM pulses. This signal refreshes the LCD panel.

NOTE

LD bus width is programmable to 1, 2, 4, or 8 bits in monochrome mode (COLOR bit in Panel Configuration register is set to 0). Data is justified to the least significant bits of the LD [17:0] bus. Passive color displays use a fixed 2-2/3 pixels of data per 8-bit vector as shown in Figure 43-13.

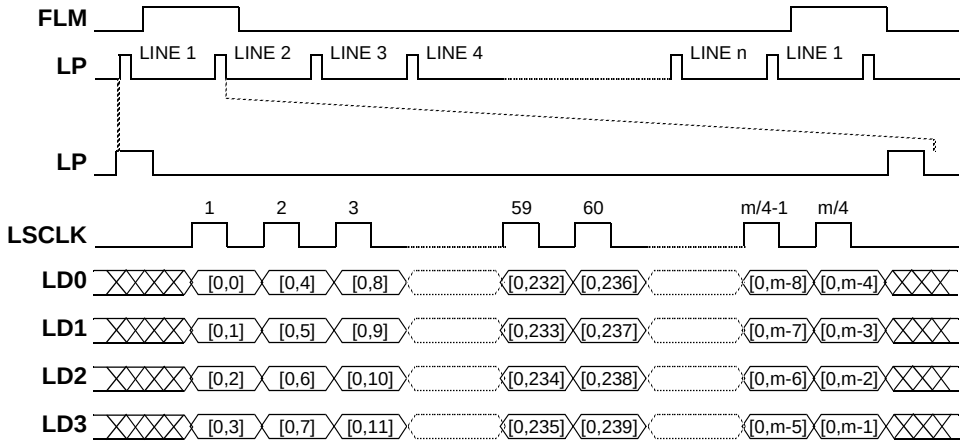


Figure 43-12. LCDC Interface Timing for 4-Bit Data Width Gray-Scale Panels

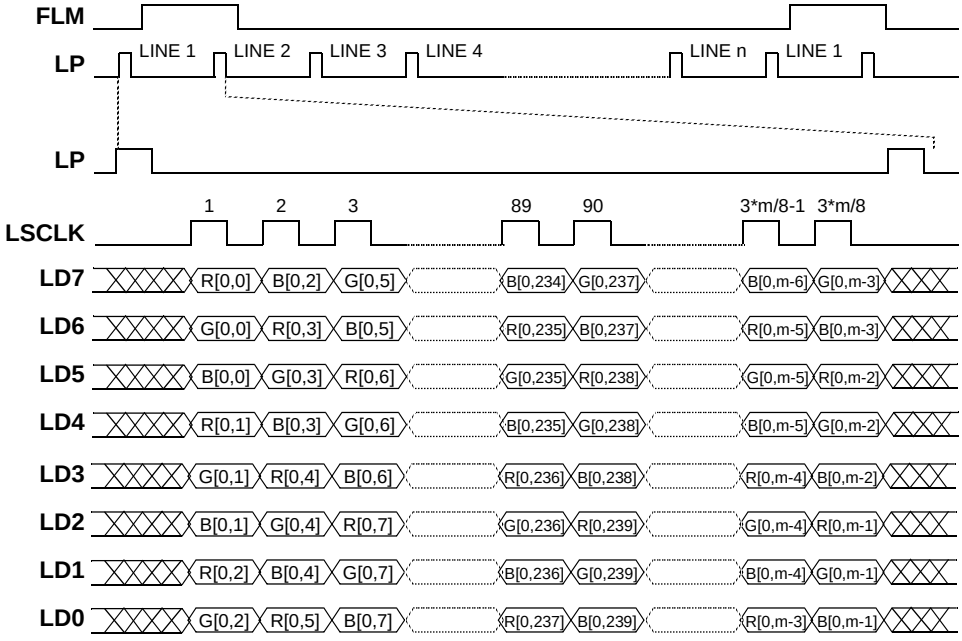


Figure 43-13. LCDC Interface Timing for 8-Bit Data Passive Matrix Color Panels

43.2.9.3 Passive Panel Interface Timing

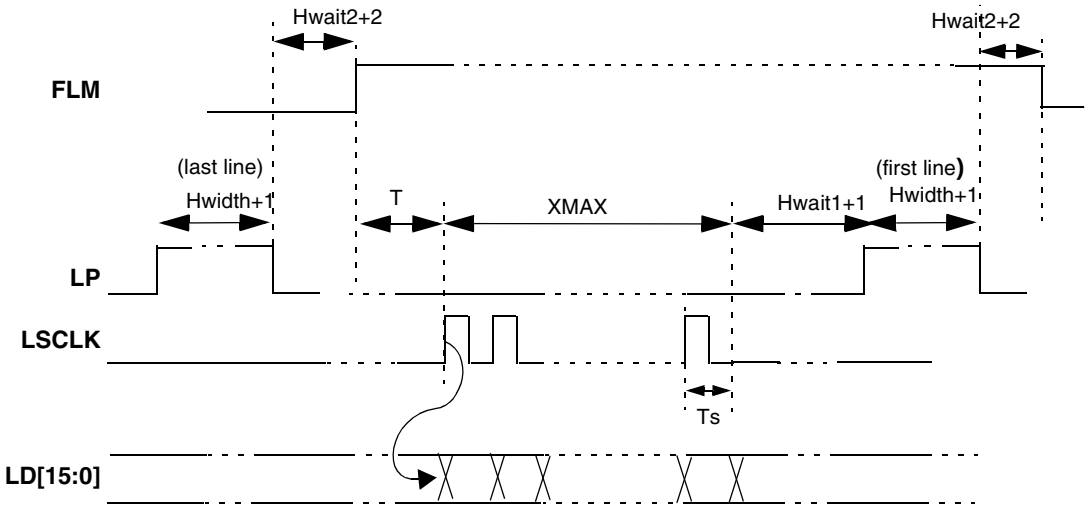
Figure 43-14 shows horizontal timing (timing of one line), including both line pulse (LP) and data. LP width and delay, both before and after LP are programmable. The parameters used for panel interface timing are:

- XMAX (X size) defines number of pixels per line. XMAX is the total number of pixels per line.
- H_WAIT_1 defines the delay from end of data output to the beginning of LP.
- H_WIDTH (horizontal sync pulse width) defines width of FLM pulse, and it must be at least 1.
- H_WAIT_2 defines the delay from end of LP to the beginning of data output.

NOTE

All parameters are defined in unit of pixel clock period, unless otherwise stated.

43.2.10 8 bpp Mode Color STN Panel



When it is in CSTN mode or monochrome mode with bus width = 1, T = 1 SCLK period.
When it is in monochrome mode with bus width = 2, 4 and 8, T = 1, 2 and 4 SCLK period respectively.

Figure 43-14. Horizontal Sync Pulse Timing in Passive Mode

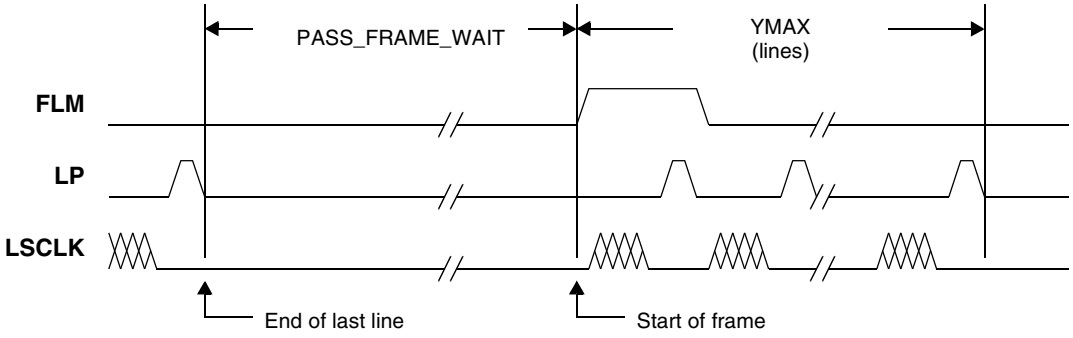


Figure 43-15. Vertical Sync Pulse Timing in Passive Mode

43.2.10.1 Active Matrix Panel Interface Signals

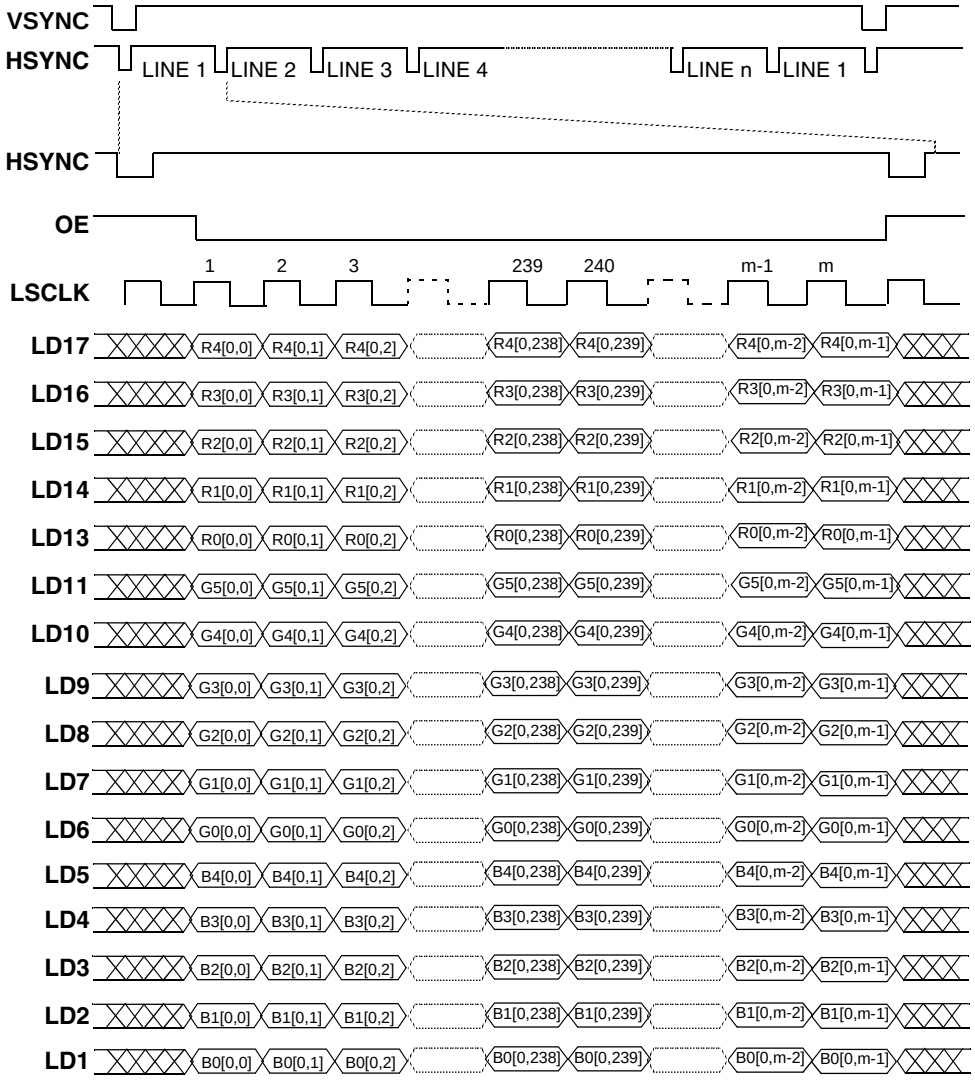


Figure 43-16. LCDC Interface 16-Bit Timing for Active Matrix Color Panels

Figure 43-16 and Figure 43-17 shows LCD interface timing for an active matrix color TFT panel. In these figures, signals are shown with negative polarity (FLMPOL=1, LPPOL=1, CLKPOL=0, OEPOL=1). In TFT mode, LSCLK is automatically inverted. The panel interface timing for active matrix panels is sometimes referred to as a “digital CRT” and is controlled by shift clock (LSCLK), horizontal sync pulse (HSYNC, LP pin in passive mode), vertical sync pulse (VSYNC, FLM pin in passive mode), output enable (OE, ACD pin in passive mode), and line data (LD) signals. Sequence of events for active matrix interface timing is:

1. LSCLK latches data into the panel on its negative edge (when positive polarity is selected). In active mode, LSCLK runs continuously.
2. HSYNC causes the panel to start a new line.
3. VSYNC causes the panel to start a new frame. It always encompasses at least one HSYNC pulse.

4. OE functions as an output enable signal to the CRT. This output enable signal is similar to blanking output in a CRT and enables the data to be shifted onto the display. When disabled, data is invalid and the trace is off.

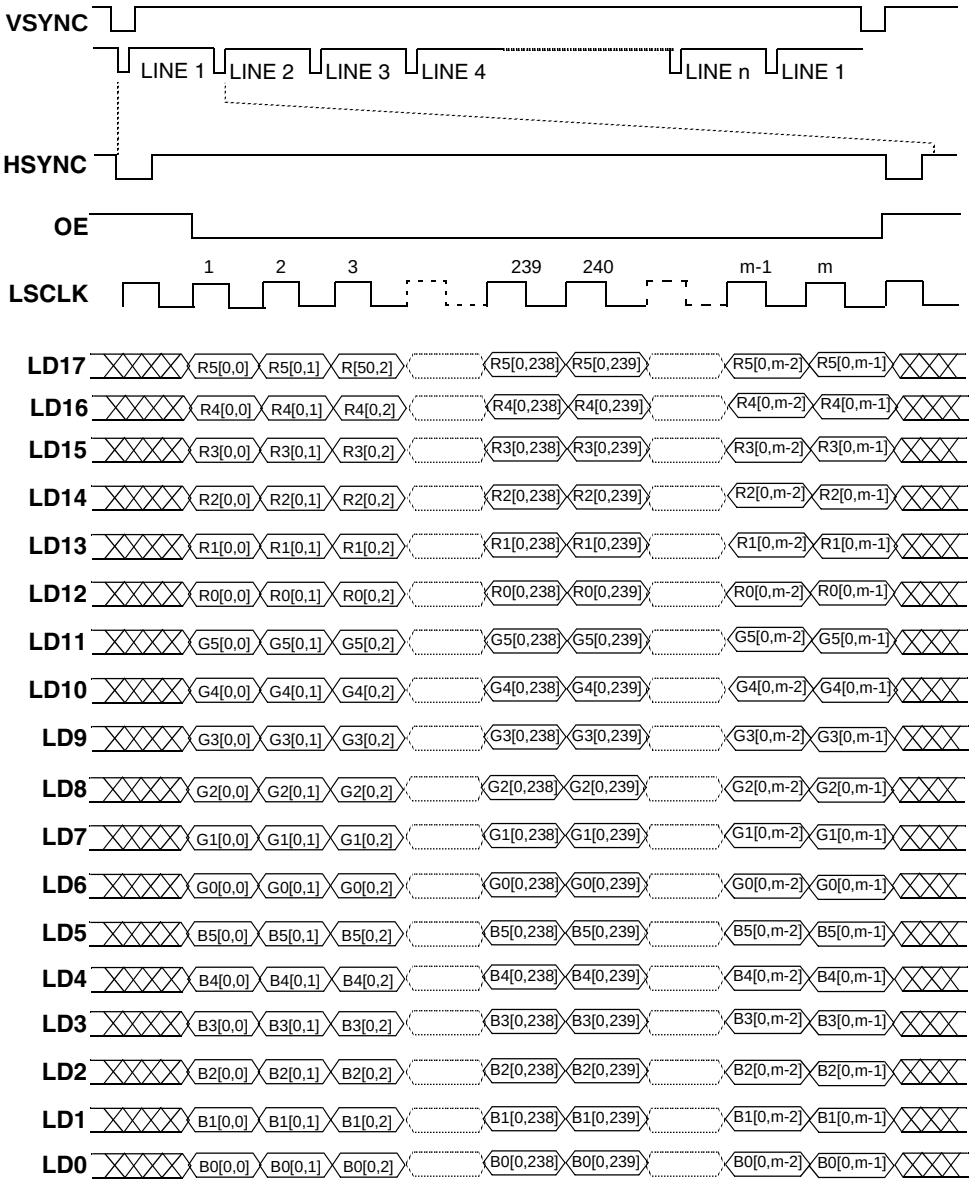


Figure 43-17. LCDC Interface 18-Bit Timing for Active Matrix Color Panels

In 4-bit and 8-bit mode, LD [17:12] bits define red, LD [11:6] bits define green and LD [5:0] bits define blue. In 12-bit mode, LD[17:14] bits define red, LD[11:8] bits define green and LD[5:2] bits define blue. In 16-bit mode, LD [17:13] bits define red, LD [11:6] bits define green and LD [5:1] bits define blue.

The actual TFT color channel assignments are shown in Table 43-4. The unused bits are fixed at 0.

Table 43-4. TFT Color Channel Assignments

	LD 17	LD 16	LD 15	LD 14	LD 13	LD 12	LD 11	LD 10	LD 9	LD 8	LD 7	LD 6	LD 5	LD 4	LD 3	LD 2	LD 1	LD 0
4 bpp	R5	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1	B0
8 bpp	R5	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1	B0
12 bpp	R3	R2	R1	R0	—	—	G3	G2	G1	G0	—	—	B3	B2	B1	B0	—	—
16 bpp	R4	R3	R2	R1	R0	—	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	—
18 bpp	R5	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1	B0
16 bpp (For AUS mode only)	—	—	—	—	—	—	—	—	—	—	R4	R3	R2	R1	R0	R4	R3	R2[1]
	—	—	—	—	—	—	—	—	—	—	G5	G4	G3	G2	G1	G0	G5	G4[2]
	—	—	—	—	—	—	—	—	—	—	B4	B3	B2	B1	B0	B4	B3	B2[3]
24 bpp (For AUS mode only)	—	—	—	—	—	—	—	—	—	—	R7	R6	R5	R4	R3	R2	R1	R0[1]
	—	—	—	—	—	—	—	—	—	—	G7	G6	G5	G4	G3	G2	G1	G0[2]
	—	—	—	—	—	—	—	—	—	—	B7	B6	B5	B4	B3	B2	B1	B0[3]

Note1: Data output at clock 3*N +1.
Note2: Data output at clock 3*N + 2.
Note3: Data output at clock 3*N +3.

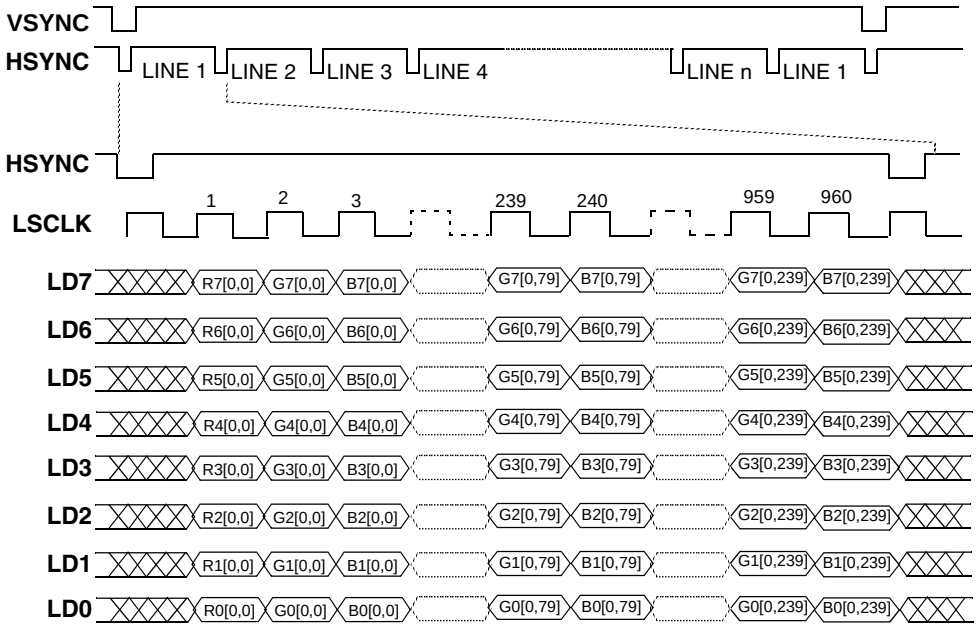


Figure 43-18. LCDC Interface 24-bpp Timing For AUS Mode

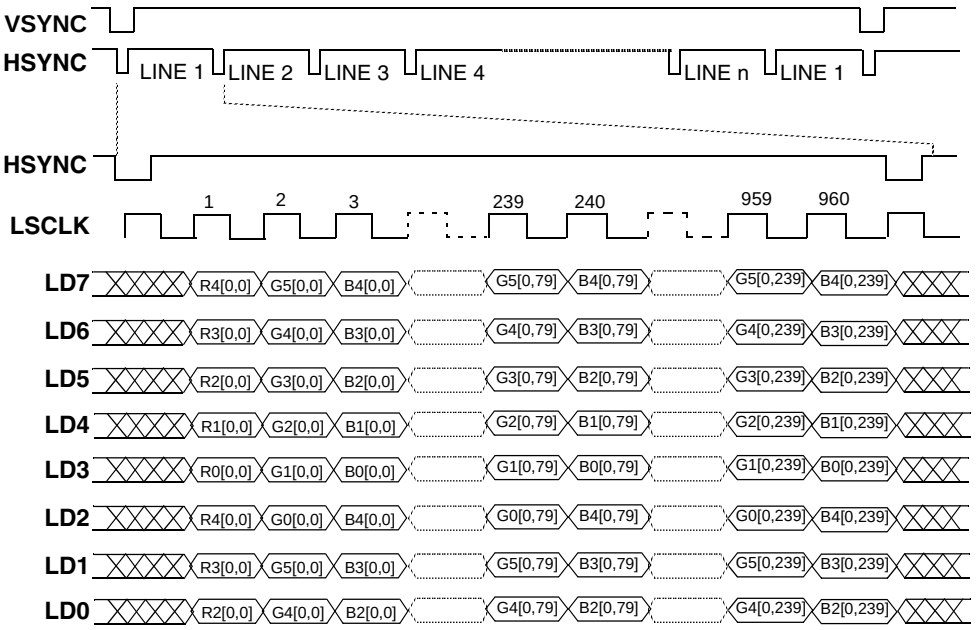


Figure 43-19. LCDC Interface 16-bpp Timing For AUS Mode

43.2.10.2 Active Panel Interface Timing

Figure 43-20 shows horizontal timing (timing of one line), including both horizontal sync pulse and data. The width of HSYNC and delays, both before and after HSYNC are programmable. The timing signal parameters are defined as follows:

- H_WIDTH defines the width of HSYNC pulse and must be at least 1.
- H_WAIT_2 defines the delay from end of HSYNC to the beginning of the OE pulse.
- H_WAIT_1 defines the delay from end of OE to the beginning of the HSYNC pulse.
- XMAX defines the (total) number of pixels per line.

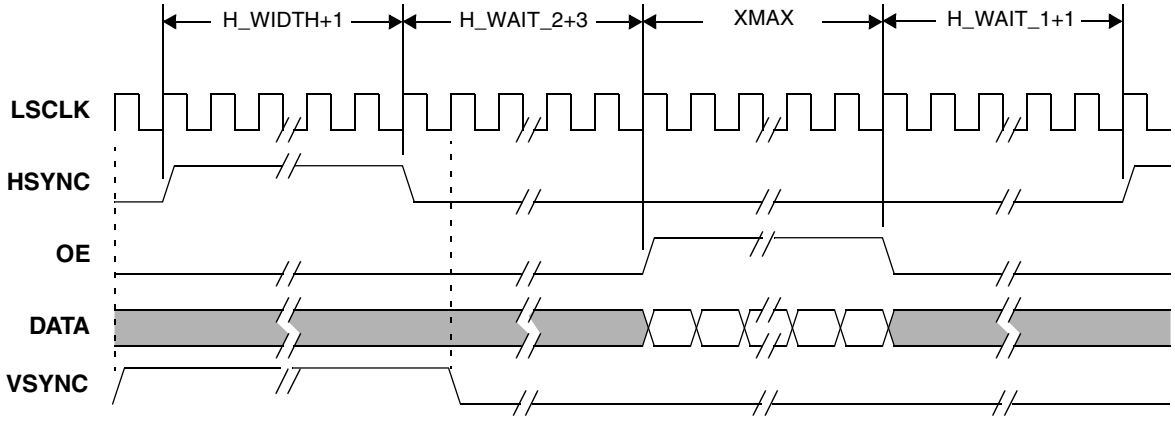


Figure 43-20. Horizontal Sync Pulse Timing in TFT Mode

Note: All parameters are defined in pixel periods, not LSCLK periods.

Figure 43-21 shows vertical timing (timing of one frame). The delay from end of one frame until the beginning of the next is programmable. The memory timing signal parameters are:

- V_WAIT_1 is a delay measured in lines. For V_WAIT_1 = 1 there is a delay of one HSYNC (time = one line period) before VSYNC. HSYNC pulse is output during the V_WAIT_1 delay.
- For V_WIDTH (vertical sync pulse width) = 0, VSYNC encloses one HSYNC pulse. For V_WIDTH = 2, VSYNC encloses two HSYNC pulses.
- V_WAIT_2 is a delay measured in lines. For V_WAIT_2 = 1, there is a delay of one HSYNC (time = one line period) after VSYNC. The HSYNC pulse is output during the V_WAIT_2 delay.

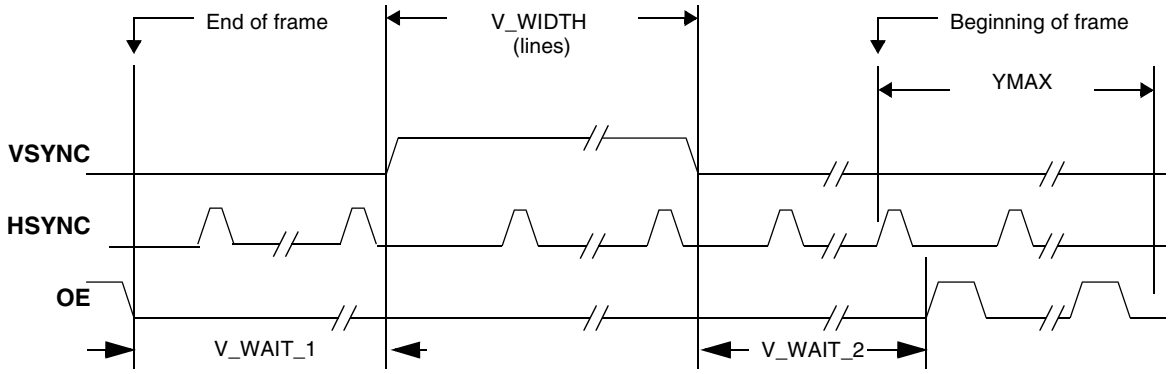


Figure 43-21. Vertical Sync Pulse Timing TFT Mode

43.3 Memory Map and Register Definitions

The LCDC memory space contains 21 32-bit registers for display parameters, a read-only status register, and two 256 × 18 Color Mapping RAMs—one for the graphic window and the other for the background plane. The color mapping RAMs are physically located inside the Palette Lookup Table module. Table 43-5 summarizes these registers and their addresses. The LCDC only supports word access. Byte and halfword access is undefined.

Table 43-5. LCDC Memory Map

Address	Register	Access	Reset Value	Section/Page
0x1002_1000 (LSSAR)	LCDC Screen Start Address Register	R/W	0x0000_0000	43.3.2/43-24
0x1002_1004 (LSR)	LCDC Size Register	R/W	0x0000_0000	43.3.3/43-24
0x1002_1008 (LVPWR)	LCDC Virtual Page Width Register	R/W	0x0000_0000	43.3.4/43-25
0x1002_100C (LCPR)	LCDC Cursor Position Register	R/W	0x0000_0000	43.3.5/43-26
0x1002_1010 (LCWHB)	LCDC Cursor Width Height and Blink Register	R/W	0x0101_FFFF	43.3.6/43-27
0x1002_1014 (LCCMR)	LCDC Color Cursor Mapping Register	R/W	0x0000_0000	43.3.7/43-28
0x1002_1018 (LPCR)	LCDC Panel Configuration Register	R/W	0x0000_0000	43.3.8/43-29
0x1002_101C (LHCR)	LCDC Horizontal Configuration Register	R/W	0x0000_0000	43.3.9/43-31
0x1002_1020 (LVCR)	LCDC Vertical Configuration Register	R/W	0x0400_0000	43.3.10/43-32
0x1002_1024 (LPOR)	LCDC Panning Offset Register	R/W	0x0000_0000	43.3.11/43-33
0x1002_1028 (LSCR)	LCDC Sharp Configuration Register	R/W	0x400C_0373	43.3.12/43-34

Table 43-5. LCDC Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x1002_102C (LPCCR)	LCDC PWM Contrast Control Register	R/W	0x0000_0000	43.3.13/43-36
0x1002_1030 (LDCR)	LCDC DMA Control Register	R/W	0x8004_0060	43.3.14/43-37
0x1002_1034 (LRMCR)	LCDC Refresh Mode Control Register	R/W	0x0000_0000	43.3.15/43-38
0x1002_1038 (LICR)	LCDC Interrupt Configuration Register	R/W	0x0000_0000	43.3.16/43-39
0x1002_103C (LIER)	LCDC Interrupt Enable Register	R/W	0x0000_0000	43.3.17/43-40
0x1002_1040 (LISR)	LCDC Interrupt Status Register	Read	0x0000_0000	43.3.18/43-41
0x1002_1050 (LGWSAR)	LCDC Graphic Window Start Address Register	R/W	0x0000_0000	43.3.19/43-43
0x1002_1058 (LGWVPWR)	LCDC Graphic Window Size Register	R/W	0x0000_0000	43.3.20/43-43
0x1002_1058 (LGWVPWR)	LCDC Graphic Window Virtual Page Width Register	R/W	0x0000_0000	43.3.21/43-44
0x1002_105C (LGWPOR)	LCDC Graphic Window Panning Offset Register	R/W	0x0000_0000	43.3.22/43-44
0x1002_1060 (LGWPR)	LCDC Graphic Window Position Register	R/W	0x0000_0000	43.3.23/43-45
0x1002_1064 (LGWCR)	LCDC Graphic Window Control Register	R/W	0x0000_0000	43.3.24/43-46
0x1002_1068 (LGWDCR)	LCDC Graphic Window DMA Control Register		0x8004_0060	43.3.25/43-47
0x1002_1080 (LAUSCR)	LCDC Aus mode Control Register	R/W	0x0000_0000	43.3.26/43-48
0x1002_1084 (LAUSCCR)	LCDC Aus mode Cursor Control Register	R/W	0x0000_0000	43.3.27/43-49

43.3.1 Register Summary

Figure 43-22 shows the key to the register fields, and Table 43-6 shows the register figure conventions.

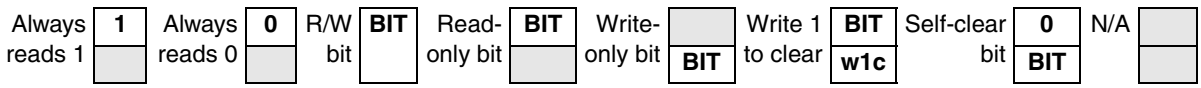


Figure 43-22. Key to Register Fields

Table 43-6. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.

Table 43-6. Register Figure Conventions (continued)

Convention	Description
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 43-7 provides an overview of the fields for all of the registers.

Table 43-7. LCDC Register Summary

Name	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
0x1002_1000 (LSSAR)	Screen Start Address High – SSA H															
	Screen Start Address Low – SSA L															
0x1002_1004 (LSR)	0	0	0	Bus Size	0	0	Screen Width – XMAX									
							Screen Height – YMAX									
0x1002_1008 (LVPWR)																
							Virtual Page Width – VPW									
0x1002_100C (LCPR)	CC			OP			Cursor X Position – CXP									
							Cursor Y Position – CYP									
0x1002_1010 (LCWHB)	BK_EN			Cursor Width – CW								Cursor Height – CH				
									BD							
0x1002_1014 (LCCMR)															CUR_COL_R[5:4]	
	Cursor Red – CUR_COL_R[3:0]				Cursor Green – CUR_COL_G						Cursor Blue – CUR_COL_B					

Table 43-7. LCDC Register Summary (continued)

Name	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
0x1002_1018 (LPCR)	TFT	COLOR	Bus Width PBSIZ		Bits Per Pixel BPIX			PIXPOL	FLMPOL	LP POL	CLKPOL	OEPOL	SCLKIDLE	END_SEL	END_BYTE_SWAP	REV_VS
	ACD SEL	Crystal Direction Toggle – ACD							SCLK SEL	SHARP	Pixel Clock Divider – PCD					
0x1002_101C (LHCR)	Horizontal Sync Width – H_WIDTH															
	Horizontal Wait 1 – H_WAIT_1								Horizontal Wait 2 – H_WAIT_2							
0x1002_1020 (LVCR)	Vertical Sync Width – V_WIDTH															
	Vertical Wait 1 – V_WAIT_1								Vertical Wait 2 – V_WAIT_2							
0x1002_1024 (LPOR)	0	0	0	0	0	0	0	0	0	0	0					
												Panning Offset – POS				
0x1002_1028 (LSCR)	PS_RISE_DELAY								CLS_RISE_DELAY							
					REV_TOGGLE_DELAY				Gray 2				Gray 1			
0x1002_102C (LPCCR)		0	0	0	0			CLS High Width								
	LD MSK					SCR		CC_EN	Pulse Width – PW							
0x1002_1030 (LDCR)	BURST	0	0	0	0	0	0	0	0	DMA High Mark – HM						
										DMA Trigger Mark – TM						
0x1002_1034 (LRMCR)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
																SELF_REF
0x1002_1038 (LICR)	0	0	0	0	0	0	0	0	0	0	0		0		0	
												GW_INT_CON		INTSYN		INTCON
0x1002_103C (LIER)	0	0	0	0	0	0	0	0								
									GW_UDR_ERR_EN	GW_ERR_RES_EN	GW_EOF_EN	GW_BOF_EN	UDR_ERR_EN	ERR_RES_EN	EOF_EN	BOF_EN

Table 43-7. LCDC Register Summary (continued)

Name	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
0x1002_1040 (LISR)	0	0	0	0	0	0	0	0								
									GW_UDR_ERR	GW_ERR_RES	GW_EOF	GW_BOF	UDR_ERR	ERR_RES	EOF	BOF
0x1002_1050 (LGWSAR)	Graphic Window Start Address High – GWSA H															
	Graphic Window Start Address Low – GWSA L															
0x1002_1054 (LGWSR)	0	0	0	0	0	0	Graphic Window Width – GWW									
							Graphic Window Height – GWH									
0x1002_1058 (LGWVPWR)	0	0	0	0	0	0										
							Graphic Window Virtual Page Width – GWVPW									
0x1002_105C (LGWPOR)	0	0	0	0	0	0	0	0	0	0	0					
												Graphic Window Panning Offset – GWPO				
0x1002_1060 (LGWPR)	0	0	0	0	0	0	Graphic Window X Position – GWXP									
							Graphic Window Y Position – GWYP									
0x1002_1064 (LGWCR)	Graphic Window Alpha Value – GWAV								GWCKE	GWE	GW_RVS				Graphic Window Color Key Red – GWCKR[5:4]	
	Graphic Window Color Key Red – GWCKR[3:0]				Graphic Window Color Key Green – GWCKG						Graphic Window Color Key Blue – GWCKB					
0x1002_1068 (LGWDCR)	GWBT	0	0	0	0	0	0	0	0	Graphic Window High Mark – GWHM						
										Graphic Window Low Mark – GWLM						
0x1002_1080 (LAUSCR)	AUS Mode								Graphic Window Color Key Red – AGWCKR[7:0] (AUS mode only)							
	Graphic Window Color Key Green – AGWCKG[7:0] (AUS mode only)								Graphic Window Color Key Blue – AGWCKB[3:0] (AUS mode only)							
0x1002_1084 (LAUSCCR)									Cursor Red – ACUR_COL_R [7:0] (AUS mode only)							
	Cursor Green – ACUR_COL_G [7:0] (AUS mode only)								Cursor Blue– ACUR_COL_B [7:0] (AUS mode only)							
0x1002 1800 — 0x1002 1BFC (BPLUT)															(R[5:4])	
	First RAM Location of Background Plane LUT (R [3:0], G [5:0], B [5:0])															
															(R[5:4])	
	Last RAM Location of Background Plane LUT (R [3:0], G [5:0], B [5:0])															

Table 43-7. LCDC Register Summary (continued)

Name	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
0x1002 1C00 — 0x1002 1FFC (GWLUT)																
	First RAM Location of Graphic Window LUT (R[3:0], G[5:0], B[5:0])															
															(R[5:4])	
	Last RAM location of Graphic Window LUT (R[3:0], G[5:0], B[5:0])															

43.3.2 LCDC Screen Start Address Register (LSSAR)

The LSSAR specifies the LCD screen start address. See Figure 43-23 for bit assignments and Table 43-8 for field descriptions.

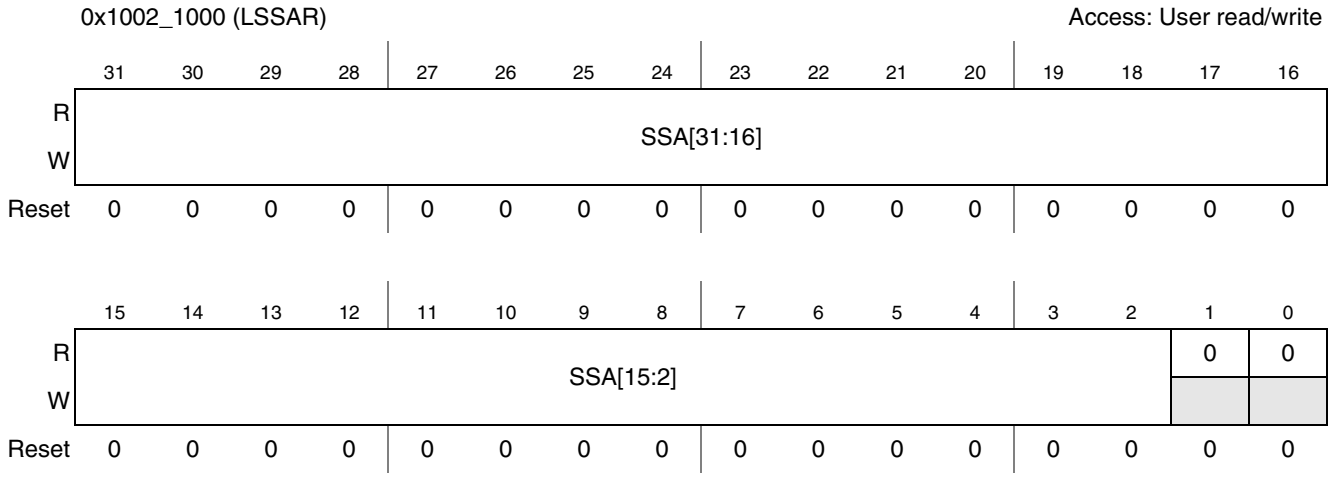


Figure 43-23. LCDC Screen Start Address Register (LSSAR)

Table 43-8. Screen Start Address Register Field Description

Field	Description
31–2 SSA	Screen Start Address of LCD Panel. Holds pixel data for a new frame from SSA address. This field must start at a location that enables a complete picture to be stored in a 4 Mbyte memory boundary (A [21:0]). A [31:22] has a fixed value for a picture's image.
1–0	Reserved. These bits are reserved and should read 0.

43.3.3 LCDC Size Register (LSR)

The LSR defines the height and width of the LCD screen. See Figure 43-24 for bit assignments and Table 43-9 for field descriptions.

0x1002_1004 (LSR) Access: User read/write

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

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

Figure 43-24. LCDC Size Register (LSR)

Table 43-9. LCDC Size Register Field Descriptions

Field	Description
31–29	Reserved. These bits are reserved and should read 0.
28 Bus Size	Bus Size. Determine LCDC bus size. 0 AHB bus of LCDC is 64-bit 1 AHB bus of LCDC is 32-bit.
25–20 XMAX	Screen Width Divided by 16. Holds screen x-axis size, divided by 16. For black-and-white panels (1 bpp), XMAX [20] is ignored, forcing the x-axis of the screen size to be a multiple of 32 pixels/line.
19–10	Reserved. These bits are reserved and should read 0.
9–0 YMAX	Screen Height. Specifies the height of LCD panel in terms of pixels or lines. Lines are numbered from 1 to YMAX for a total of YMAX lines.

43.3.4 LCDC Virtual Page Width Register (LVPWR)

The LVPWR defines virtual page width for LCD panel. See Figure 43-25 for bit assignments and Table 43-10 for field descriptions.

0x1002_1008 (LVPWR) Access: User read/write

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

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

Figure 43-25. LCDC Virtual Page Width Register (LVPWR)

Table 43-10. LCDC Virtual Page Width Register Field Descriptions

Field	Description
31–10	Reserved. These bits are reserved and should read 0.
9–0 VPW	Virtual Page Width. Defines virtual page width of LCD panel. VPW bits represent the number of 32-bit words required to hold the data for one virtual line. VPW is used in calculating the starting address representing the beginning of each displayed line.

43.3.5 LCDC Cursor Position Register (LCPR)

LCPR is used to determine the starting position of the cursor on the LCD panel. Figure 43-26 shows the register; Table 43-11 provides its field descriptions.

0x1002_100C (LCPR) Access: User read/write

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

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

Figure 43-26. LCDC Cursor Position Register (LCPR)

Table 43-11. LCDC Cursor Position Register Field Descriptions

Field	Description
31–30 CC	Cursor Control. Controls the cursor format and the type of arithmetic operations. When OP = 0 00 Transparent, cursor is disabled 01 1 for non-color displays; color defined in LCDC Color Cursor Mapping Register for color displays 10 Reversed, INV background for non-color displays; INV color defined in LCDC Color Cursor Mapping Register for color displays 11 0 for non-color displays; AND between background and cursor for color displays When OP = 1, for color mode only 00 Transparent, cursor is disabled 01 OR between background and cursor 10 XOR between background and cursor 11 AND between background and cursor
29	Reserved. This bit is reserved and should read 0.
28 OP	Arithmetic Operation Control. Enables/disables arithmetic operations between the background and cursor. 0 = Disable Arithmetic Operation 1 = Enable Arithmetic Operation
27–26	Reserved. These bits are reserved and should read 0.
25–16 CXP	Cursor X Position. Represents the cursor's horizontal starting position X in pixel count (from 0 to XMAX).
15–10	Reserved. These bits are reserved and should read 0.
9–0 CYP	Cursor Y Position. Represents the cursor's vertical starting position Y in pixel count (from 0 to YMAX).

43.3.6 LCDC Cursor Width Height and Blink Register (LCWHB)

LCWHB is used to determine the cursor's width and height, and how it blinks. Figure 43-27 shows the register; Table 43-12 provides its field descriptions.

0x1002_1010 (LCWHB) Access: User read/write

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

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

Figure 43-27. LCDC Cursor Width Height and Blink Register (LCWHB)

Table 43-12. LCDC Cursor Width Height and Blink Register Field Descriptions

Field	Description
31 BK_EN	Blink Enable. Determines whether the blink enable cursor will blink or remain steady. 0 Blink is Disabled 1 Blink is Enabled
30–29	Reserved. These bits are reserved and should read 0.
28–24 CW	Cursor Width. Specifies the width of hardware cursor in pixels. This field can be any value between 1 and 31 (setting this field to zero disables the cursor)
23–21	Reserved. These bits are reserved and should read 0.
20–16 CH	Cursor Height. Specifies the height of hardware cursor in pixels. This field can be any value between 1 and 31 (setting this field to zero disables the cursor)
15–8	Reserved. These bits are reserved and should read 0.
7–0 BD	Blink Divisor. Sets the cursor blink rate. A 32 Hz clock from RTC is used to clock the 8-bit up counter. When the counter value equals BD, the cursor toggles on/off. Hence larger the BD, slower the cursor will blink. The faster cursor blinking rate is when BD is 0.

43.3.7 LCDC Color Cursor Mapping Register (LCCMR)

LCCMR defines the cursor color in passive or TFT color modes. If bpp mode setting is smaller than 18bpp, cursor color component bits should be put in the MSBs. For example, if color cursor of RGB=(0x1a, 0x26, 0x05) is wanted, CUR_COL_R, CUR_COL_G and CUR_COL_B should be set to 0x34, 0x26 and 0x0a. Figure 43-28 shows the register; Table 43-13 provides its field descriptions.

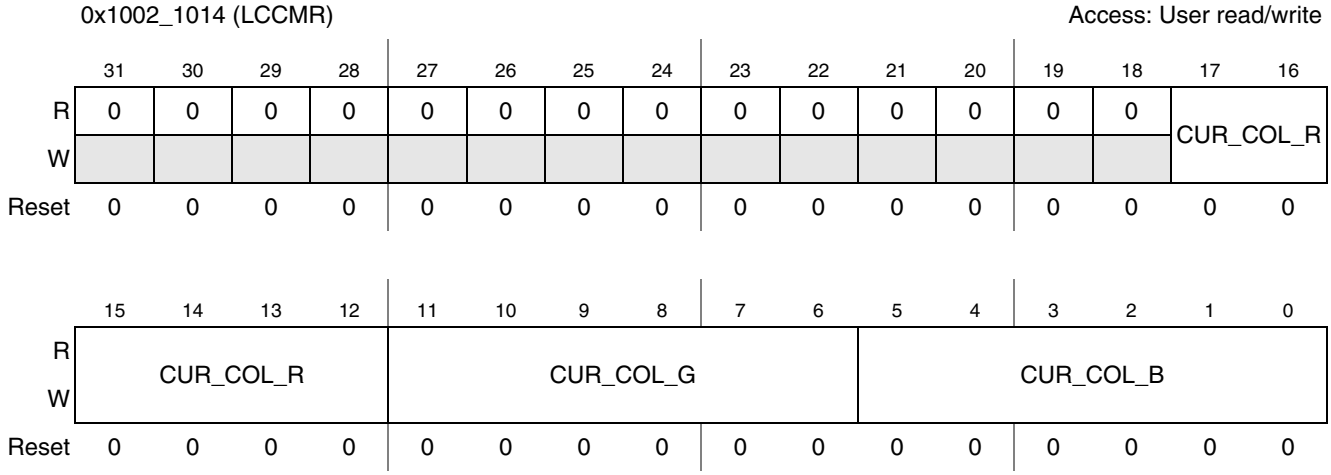


Figure 43-28. LCDC Color Cursor Mapping Register (LCCMR)

Table 43-13. LCDC Color Cursor Mapping Register Field Descriptions

Field	Description
31–18	Reserved. These bits are reserved and should read 0.
17–12 CUR_COL_R	Cursor Red Field. Defines the red component of the cursor color in color mode. 000000 = No red … 111111 = Full red
11–6 CUR_COL_G	Cursor Green Field. Defines the green component of the cursor color in color mode. 000000 = No green … 111111 = Full green
5–0 CUR_COL_B	Cursor Blue Field. Defines the blue component of the cursor color in color mode. 000000 = No blue … 111111 = Full blue

43.3.8 LCDC Panel Configuration Register (LPCR)

The LPCR defines all properties of the LCD screen. See Figure 43-29 for bit assignments and Table 43-14 for field descriptions.

0x1002_1018 (LPCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TFT	COL OR	PBSIZ		BPIX			PIX POL	FLM POL	LP POL	CLK POL	OE POL	SCLK IDLE	END_ SEL	SWA P_ SEL	REV_ VS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ACD SEL	ACD							SCLK SEL	SHAR P	PCD					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-29. LCDC Panel Configuration Register (LPCR)

Table 43-14. LCDC Panel Configuration Register Field Descriptions

Field	Description
31 TFT	Interfaces to TFT Display. Controls the format and timing of output control signals. Active and passive displays use different signal timing formats as described in Section 43.2.9, “Panel Interface Signals and Timing.” TFT also controls the use of FRC in color mode. 0 LCD panel is a Passive display 1 LCD panel is an Active display: “digital CRT” signal format, FRC is bypassed. 0 0 Monochrome 0 1 CSTN 1 0 – 1 1 TFT
30 COLOR	Interfaces to Color Display. Activates 3 channels of FRC in passive mode to allow the use of special 2 2/3 pixels per output vector format. 0 LCD panel is a Monochrome display. 1 LCD panel is a Color display. 0 0 Monochrome 0 1 CSTN 1 0 – 1 1 TFT
29–28 PBSIZ	Panel Bus Width. Specifies the panel bus width. Applicable for monochrome monitors. For passive color panels, only 8-bit panel bus width is supported. 00 1-bit 10 4-bit 11 8-bit
27–25 BPIX	Bits Per Pixel. Indicates the number of bits per pixel in memory. 000 1 bpp, FRC bypassed 001 2 bpp 010 4 bpp 011 8 bpp 100 12 bpp (16-bits of memory used) 101 16 bpp 110 18bpp (32-bits of memory used) 111 Reserved **Note:** To set normal 18bpp mode: BPIX = 110, END_SEL = 0 and SWAP_SEL = X (don’t care) To set Microsoft PAL_BGR 18bpp mode: BPIX = 110, END_SEL = 1 and SWAP_SEL = 1
24 PIXPOL	Pixel Polarity. Sets the pixels polarity. 0 Active High 1 Active Low
23 FLMPOL	First Line Marker Polarity. Sets the polarity of first line marker symbol. 0 Active High 1 Active Low
22 LPPOL	Line Pulse Polarity. Sets the polarity of line pulse signal. 0 Active High 1 Active Low
21 CLKPOL	LCD Shift Clock Polarity. Sets the polarity of active edge of LCD shift clock. 0 Active Negative edge of LSCLK (in TFT mode, Active on Positive edge of LSCLK) 1 Active Positive edge of LSCLK (in TFT mode, Active on Negative edge of LSCLK)
20 OEPOL	Output Enable Polarity. Sets the output enable signal polarity. 0 Active High 1 Active Low

Table 43-14. LCDC Panel Configuration Register Field Descriptions (continued)

Field	Description
19 SCLKIDLE	LSCLK Idle Enable. Enables/disables LSCLK when VSYNC is idle in TFT mode. 0 Disable LSCLK 1 Enable LSCLK
18 END_SEL	Endian Select. Selects the image download into memory as big or little endian format. 0 Little Endian 1 Big Endian
17 SWAP_SEL	Swap Select. LCDC operates in big endian mode internally. Swap Select controls the swap of data before operation in little endian mode. 0 16 bpp, 12 bpp mode 1 8 bpp, 4 bpp, 2 bpp, 1 bpp mode **Note:** When SWAP_SEL = 0, byte 3 (bits 31–24), byte 2 (bits 23–16), byte 1 (bits 15–8), byte 0 (bits 7–0) data swapped to byte 1, byte 0, byte 3 and byte 2 respectively. When SWAP_SEL = 1, byte 3, byte 2, byte 1, byte 0 data swapped to byte 0, byte 1, byte 2 and byte 3 respectively.
16 REV_VS	Reverse Vertical Scan. Selects the vertical scan direction as normal or reverse image flips along the x-axis). SSA register must be changed accordingly. 0 Vertical scan in Normal direction 1 Vertical scan in Reverse direction
15 ACDSEL	ACD Clock Source Select. Selects the clock source used by alternative crystal direction counter. 0 Use FRM as Clock Source for ACD count 1 Use LP/HSYN as Clock Source for ACD count
14–8 ACD	Alternate Crystal Direction. Toggles ACD signal once every 1–16 FLM cycles based on the value specified in this field. The actual number of FLM cycles between toggles is the programmed value plus one. For active mode (TFT=1), this parameter is not used.
7 SCLKSEL	LSCLK Select. Selects whether to enable or disable LSCLK in TFT mode when there is no data output. 0 Disable OE and LSCLK in TFT mode when no data output 1 Always enable LSCLK in TFT mode even if there is no data output
6 SHARP	Sharp Panel Enable. Enables/disables signals for Sharp HR-TFT 240 x 320 panels. 0 Disable Sharp signals 1 Enable Sharp signals
5–0 PCD	Pixel Clock Divider. Holds clock divider value. LCDC_CLK (PERCLK3) is divided by N (PCD plus one) to yield the pixel clock rate. Values of 1 to 63 yield N=2 to 64. Pixel clock rate is faster than LSCLK by a factor equal to the data bus width for monochrome display. For all other displays, pixel clock rate is the same as LSCLK. PCD value must be set such that LSCLK frequency is at least one third or one fourth of HCLK frequency in TFT and CSTN modes respectively, otherwise LD will be incorrect.

43.3.9 LCDC Horizontal Configuration Register (LHCR)

LHCR defines the horizontal sync pulse timing. For detailed settings, refer to the *i.MX27 Multimedia Applications Processor Data Sheet*. Figure 43-30 shows the register; Table 43-15 provides its field descriptions.

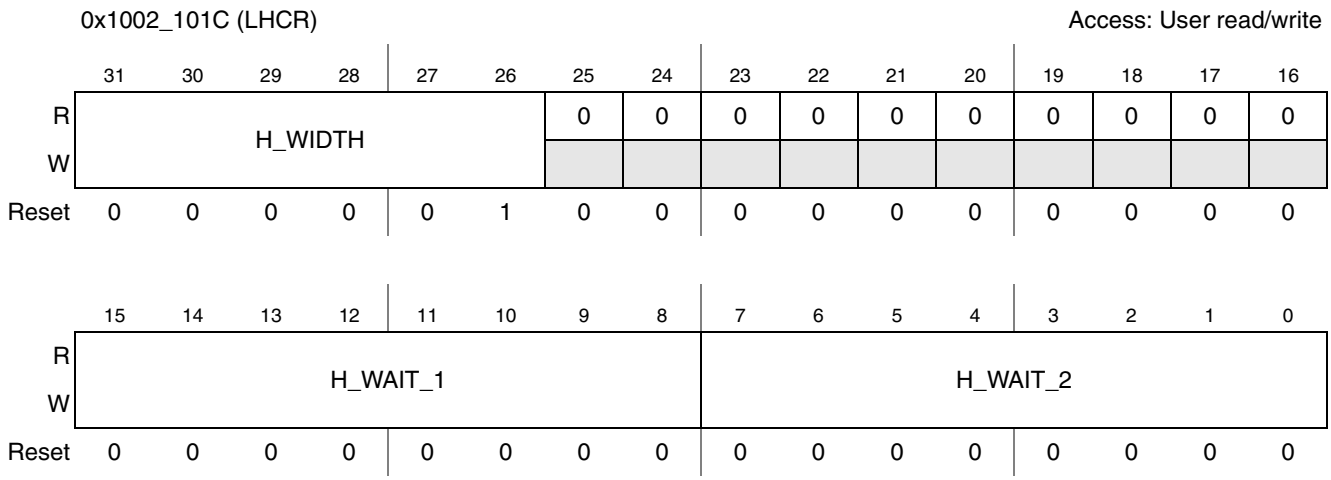


Figure 43-30. LCDC Horizontal Configuration Register (LHCR)

Table 43-15. LCDC Horizontal Configuration Register Field Description

Field	Description
31–26 H_WIDTH	Horizontal Sync Pulse Width. Specifies number of SCLK periods for which HSYNC is activated. The active time is equal to (H_WIDTH + 1) of the SCLK periods.
25–16	Reserved. These bits are reserved and should read 0.
15–8 H_WAIT_1	Wait Between OE and HSYNC. In TFT mode, it specifies the number of SCLK periods between the end of OE signal and the beginning of HSYNC. Total delay time equals (H_WAIT_1 + 1) of SCLK periods. In CSTN mode, it specifies the number of SCLK periods between the last display data and the beginning of HSYNC signal. Total delay time equals (H_WAIT_1 + 1) of SCLK periods.
7–0 H_WAIT_2	Wait Between HSYNC and Start of Next Line. In TFT mode, it specifies the number of SCLK periods between the end of HSYNC and the beginning of OE signal. Total delay time equals (H_WAIT_2 + 3). In CSTN mode, it specifies the number of SCLK periods between the end of HSYNC and the first display data in each line. Total delay time equals (H_WAIT_2 + 2) of SCLK periods.

43.3.10 LCDC Vertical Configuration Register (LVCR)

LVCR defines the vertical sync pulse timing. For detailed settings, refer to the *i.MX27 Multimedia Applications Processor Data Sheet*. Figure 43-31 shows the register; Table 43-16 provides its field descriptions.

0x1002_1020 (LVCR) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	V_WAIT_1								V_WAIT_2							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-31. LCDC Vertical Configuration Register (LVCR)

Table 43-16. LCDC Vertical Configuration Register Field Descriptions

Field	Description
31–26 V_WIDTH	Vertical Sync Pulse Width. Specifies the width, in lines, of VSYNC pulse for active mode (TFT =1). For a value of “000001”, vertical sync pulse encompasses one HSYNC pulse. For a value of “000002”, vertical sync pulse encompasses two HSYNC pulses, and so on. For passive mode (TFT=0) and non-color mode, see Figure 43-14.
25–16	Reserved. These bits are reserved and should read 0.
15–8 V_WAIT_1	Wait Between Frames 1. Defines the delay, in lines, between the end of OE pulse and the beginning of VSYNC pulse for active mode (TFT=1). This field has no meaning in passive non-color mode. The actual delay is (V_WAIT_1). In passive color mode, this field is the delay, measured in virtual clock periods, between the last line of the frame to the beginning of next frame.
7–0 V_WAIT_2	Wait Between Frames 2. Defines the delay, in lines, between the end of VSYNC pulse and the beginning of OE pulse of the first line in active mode (TFT=1). The actual delay is V_WAIT_2) lines. Set this field to zero for passive non-color mode. The minimum value of this field is 0x01.

43.3.11 LCDC Panning Offset Register (LPOR)

The LCDC Panning Offset Register (LPOR) sets up panning for the image. Figure 43-32 shows the register; Table 43-17 provides its field descriptions.

0x1002_1024 (LPOR) Access: User read/write

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

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

Figure 43-32. LCDC Panning Offset Register (LPOR)

Table 43-17. LCDC Panning Offset Register Field Descriptions

Field	Description
31–5	Reserved. These bits are reserved and should read 0.
4–0 POS	Panning Offset. Defines the number of bits that the data from memory is panned to the left before processing. POS is read by the LCDC once at the beginning of each frame. For example, in 4bpp mode, setting POS = 16 shifts 16-bits, which means panning the image by 4 pixels left. **Note:** Shifting data more than 32 bits should use LSSAR register setting. 18 bpp panning should use LSSAR register setting. To achieve panning of the final image by N bits:

Bits Per Pixel	POS	Effective # of pixels Panned on Image
1	N	N
2	2N	N
4	4N	N
8	8N	N
12/16	16N	N

43.3.12 LCDC Sharp Configuration Register (LSCR)

For 2 bpp modes, full black and full white are the two predefined display levels. The other two intermediate gray-scale shading densities can be adjusted within the LSCR. LSCR also controls the relative delay timing of CLS, REV and PS. For detail Sharp panel settings, refer to the *i.MX27 Multimedia Applications Processor Data Sheet*. A TFT timing diagram that shows the relationship between these signals is shown in the Figure 43-34.

0x1002_1028 (LSCR) Access: User read/write

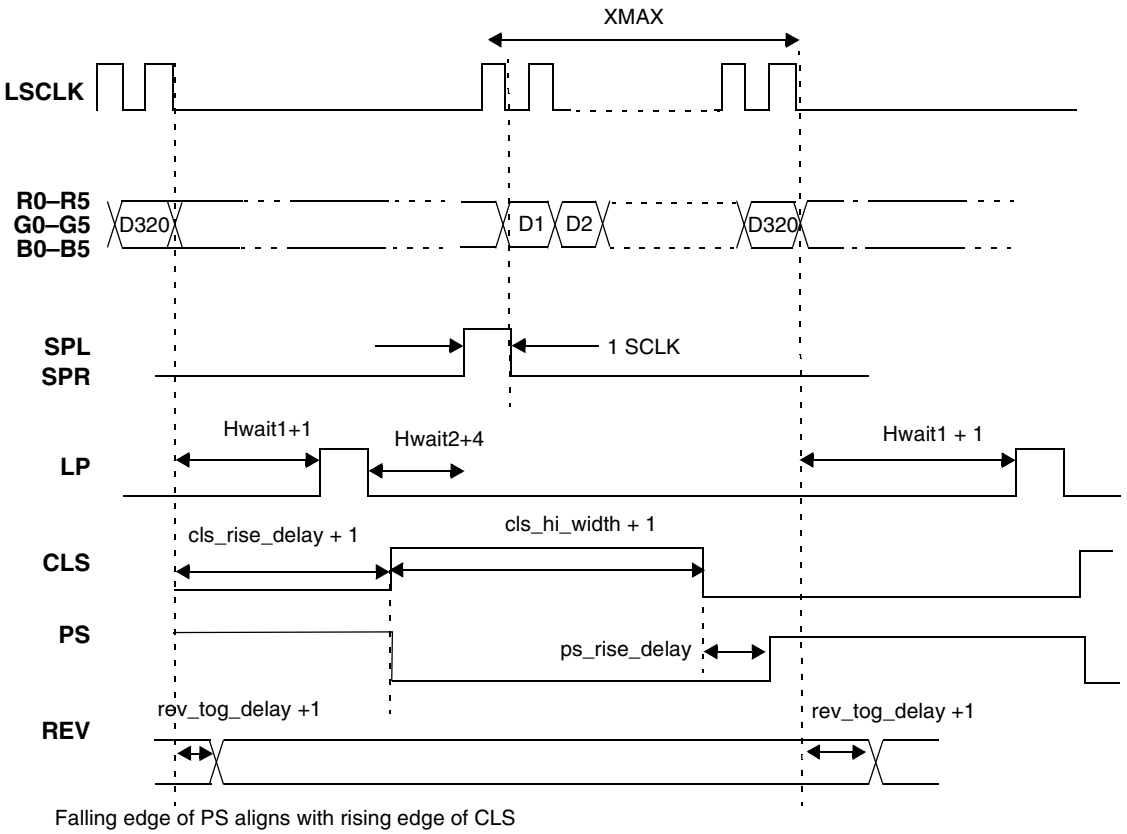
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PS_RISE_DELAY						0	0	CLS_RISE_DELAY							
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0

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

Figure 43-33. LCDC Sharp Configuration Register (LSCR)

Table 43-18. LCDC Sharp Configuration Register Field Descriptions

Field	Description
PS_RISE_DELAY 31–26	PS Rise Delay. Controls the rising edge delay of PS relative to the falling edge of CLS. Total delay time equals to PS_RISE_DELAY SCLK periods. 00000 0 LSCLK period 00001 1 LSCLK period … 11111 63 LSCLK periods
25–24	Reserved. These bits are reserved and should read 0.
CLS_RISE_DELAY 23–16	CLS Rise Delay. Controls the rising edge delay of CLS relative to the last LD of the line. Total delay time equals to (CLS_RISE_DELAY + 1) SCLK periods. 00000000 10 LSCLK periods 00000001 2 LSCLK period … 11111111 256 LSCCLK periods
15–12	Reserved. These bits are reserved and should read 0.
REV_TOGGLE_DELAY 11–8	REV Toggle Delay. Controls the transition delay of REV relative to the last LD of the line. Total delay time equals to (REV_TOGGLE_DELAY + 1) SCLK periods. 0000 1 LSCLK period 0001 2 LSCLK period … 1111 16 LSCLK periods
GRAY 2 7–4	Gray-Scale 2. Represents one of the two gray-scale shading densities. This field is programmable to any value between 0 and 16 (0 and 16 are already defined as two of the four colors)
GRAY 1 3–0	Gray-Scale 1. Represents the other gray-scale shading density. This field is programmable to any value between 0 and 16 (0 and 16 are already defined as two of the four colors)



Falling edge of PS aligns with rising edge of CLS
The rising edge delay of PS is programmed by PS_RISE_DELAY
CLS_HI_WIDTH is equal to PWM_SCR0 • 256 + PWM_WIDTH in units of LSCLK.
SPL/SPR pulse width is fixed and aligned to the first data of the line.
REV toggles every LP period.

Figure 43-34. Horizontal Timing in Device_Number

43.3.13 LCDC PWM Contrast Control Register (LPCCR)

LPCCR is used to control the signal output at the contrast pin, which controls contrast of the LCD panel.

0x1002_102C (LPCCR) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LDMSK	0	0	0	0	SCR		CC_EN	PW							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-35. LCDC PWM Contrast Control Register (LPCCR)

Table 43-19. LCDC PWM Contrast Control Register Field Description

Field	Description
31–25	Reserved. These bits are reserved and should read 0.
24–16 CLS_HI_WIDTH	CLS High Pulse Width. Controls the pulse width of CLS in units of SCLK. The actual pulse width = CLS_HI_WDITH +1.
15 LDMSK	LD Mask. Enables/disables the LD output to zero for Sharp TFT panel power-off sequence. 0 LD [15:0] is normal 1 LD [15:0] always equals 0
14–11	Reserved. These bits are reserved and should read 0.
10–9 SCR	Source Select. Selects the input clock source for PWM counter. PWM output frequency is equal to the frequency of input clock divided by 256. 00 Line pulse 01 Pixel clock 10 LCD clock 11 Reserved
8 CC_EN	Contrast Control Enable. Enables/disables the contrast control function. 0 Contrast control is off 1 Contrast control is on
7–0 PW	Pulse-Width. Controls the pulse-width of the built-in PWM, which controls the contrast of LCD screen.

43.3.14 LCDC DMA Control Register (LDCR)

There is a 128×32 bit line buffer in the LCDC that stores DMA data from system memory. LDCR controls DMA burst length and when to trigger a DMA burst in terms of number of data bytes left in the pixel buffer.

0x1002_1030 (LDCR) Access: User read/write

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

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

Figure 43-36. LCDC DMA Control Register (LDCR)

Table 43-20. LCDC DMA Control Register Field Descriptions

Field	Description
31 BURST	Burst Length. Determines whether the burst length is fixed or dynamic. 0 Burst length is dynamic 1 Burst length is fixed
30–23	Reserved. These bits are reserved and should read 0.
22–16 HM	DMA High Mark. Establishes the high mark for DMA requests. For dynamic burst length, after DMA request is made, data is loaded and pixel buffer continues to be filled until the number of empty words left in DMA FIFO is equal to the high mark minus 2. Minimum HM setting in dynamic burst is 3. For fixed burst length, burst length (in words) of each request is equal to the DMA high mark setting and its value must be larger than TM.
15–7	Reserved. These bits are reserved and should read 0.
6–0 TM	DMA Trigger Mark. Sets the low-level mark in the pixel buffer to trigger a DMA request. The low-level mark equals the number of words left in the pixel buffer.

NOTE

The dynamic mode is recommend, and also the TM and HM reset value are recommend in the dynamic mode. The same recommendation is for the graphic DMA control.

43.3.15 LCDC Refresh Mode Control Register (LRMCR)

LRMCR is used to control the refresh characteristics.

0x1002_1034 (LRMCR) Access: User read/write

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

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

Figure 43-37. LCDC Refresh Mode Control Register (LRMCR)

Table 43-21. LCDC Refresh Mode Control Register Field Descriptions

Field	Description
31–1	Reserved. These bits are reserved and should read 0.
0 SELF_REF	Self-Refresh. Enables/disables self-refresh mode. 0 Disable self-refresh 1 Enable self-refresh

NOTE

On entering self-refresh mode, LSCLK and LD [17:0] signals stay low. HYSN and VSYN operate normally.

Except for SSA, BGLUT and GWLUT registers, all configurations must be performed before enabling the LCDC to avoid a malfunction.

SSA must always match the address range of the RAM selected. If the user wants to switch between various types of RAM, LCDC must be disabled before switching.

43.3.16 LCDC Interrupt Configuration Register (LICR)

LICR is used to configure the interrupt conditions.

0x1002_1038 (LICR) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	GW_INT_ CON	0	INT SYN	0	INT CON
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-38. LCDC Interrupt Configuration Register (LICR))

Table 43-22. LCDC Interrupt Configuration Register Field Descriptions

Field	Description
31–5	Reserved. These bits are reserved and should read 0.
4 GW_INT_CON	Graphic Window Interrupt Condition. Determines if an interrupt condition is set at the beginning or end of graphic window condition. 0 Interrupt flag is set when end of graphic window is reached 1 Interrupt flag is set when beginning of graphic window is reached
3	Reserved. This bit is reserved and should read 0.

Table 43-22. LCDC Interrupt Configuration Register Field Descriptions

Field	Description
2 INTSYN	Interrupt Source. Determines if an interrupt flag is set during last/first data of frame loading or on last/first data of frame output to the LCD panel. **Note:** There is a latency between loading last/first data of the frame to output to the LCD panel. 0 Interrupt flag is set on loading the last/first data of frame from memory 1 Interrupt flag is set on output of the last/first data of frame to LCD panel **Table 43-23. INTSYN/INTCON Settings** INTSYN / INTCON / Description: 0 / 0 / Interrupt flag is set on loading last data of frame from memory. 0 / 1 / Interrupt flag is set on loading first data of frame from memory. 1 / 0 / Interrupt flag is set on output of last data of frame to LCD panel. 1 / 1 / Interrupt flag is set on output of first data of frame to LCD panel.
1	Reserved. This bit is reserved and should read 0.
0 INTCON	Interrupt Condition. Determines if an interrupt condition is set at the beginning or end of frame condition. 0 Interrupt flag is set when the End of Frame (EOF) is reached. 1 Interrupt flag is set when the Beginning of Frame (BOF) is reached.

43.3.17 LCDC Interrupt Enable Register (LIER)

LIER is used to enable the LCDC interrupt pin generated to ARM926EJ-S Interrupt Controller (AITC). When interrupt is masked, LCDC will not generate the interrupt request to AITC, but its status can still be observed in the Interrupt Status Register.

0x1002_103C (LIER) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	GW_UDR_ERR_EN	GW_ERR_RES_EN	GW_EOF_EN	GW_BOF_EN	UDR_ ERR_EN	ERR_ RES_EN	EOF_EN	BOF_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-39. LCDC Interrupt Enable Register (LIER)

Table 43-24. LCDC Interrupt Enable Register Field Description

Field	Description
31–8	Reserved. These bits are reserved and should read 0.
7 GW_UDR_ERR_EN	Graphic Window Under Run Error Interrupt Enable. Used to enable or mask the graphic window under-run error interrupt to AITC.
6 GW_ERR_RES_EN	Graphic Window Error Response Interrupt Enable. Used to enable or mask the graphic window error response interrupt to AITC.
5 GW_EOF_EN	Graphic Window End of Frame Interrupt Enable. Used to enable or mask the graphic window end of frame interrupt to AITC.
4 GW_BOF_EN	Graphic Window Beginning of Frame Interrupt Enable. Used to enable or mask the graphic window beginning of frame interrupt to AITC.
3 UDR_ERR_EN	Under Run Error Interrupt Enable. Used to enable or mask the background plane under-run error interrupt to AITC.
2 ERR_RES_EN	Error Response Interrupt Enable. Used to enable or mask the background plane error response interrupt to AITC.
1 EOF_EN	End of Frame Interrupt Enable. Used to enable or mask the background plane end of frame interrupt to AITC.
0 BOF_EN	Beginning of Frame Interrupt Enable. Used to enable or mask the background plane beginning of frame interrupt to AITC.

Bit Value Meaning: 0 = mask interrupt, 1 = enable interrupt

43.3.18 LCDC Interrupt Status Register (LISR)

The read-only LISR indicates whether an interrupt has occurred or not. The status bit is set whenever the interrupt condition is met. If any bit in this register is set and the corresponding bit in LIER is set, LCDC interrupt pin is asserted to AITC. The status bit is cleared by reading the register.

0x1002_1040 (LISR) Access: User read-only

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	GW_UDR_ERR	GW_ERR_RES	GW_EOF	GW_BOF	UDR_ ERR	ERR_ RES	EOF	BOF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-40. LCDC Interrupt Status Register (LISR)

Table 43-25. LCDC Interrupt Status Register Field Descriptions

Field	Description
31–8	Reserved. These bits are reserved and should read 0.
7 GW_UDR_ERR	Graphic Window Under Run Error. Indicates whether the LCDC FIFO in graphic window plan has hit an under-run condition. This is when the data output rate is faster than the data input rate to the FIFO of the graphic window plan. Under-run can cause erroneous data output to LD. LD data output rate must be adjusted to prevent this error.
6 GW_ERR_RES	Graphic Window Error Response. Indicates whether the LCDC has issued a read data request in graphic window and has received a response from the memory controller not equal to ‘OK.’ It is cleared by reading the status register, at power on reset, or when the LCDC is disabled.
5 GW_EOF	Graphic Window End of Frame. Indicates whether the end of graphic window has reached. It is cleared by reading the status register, at power on reset, or when the LCDC is disabled.
4 GW_BOF	Graphic Window Beginning of Frame. Indicates whether the beginning of graphic window has reached. It is cleared by reading the status register, at power on reset, or when the LCDC is disabled.
3 UDR_ERR	Under Run Error. Indicates whether the LCDC FIFO has hit an under-run condition. This is when the data output rate is faster than the data input rate to the FIFO. Under-run can cause erroneous data output to LD. LD data output rate must be adjusted to prevent this error.
2 ERR_RES	Error Response. Indicates whether the LCDC has issued a read data request and has received a response from the memory controller not equal to ‘OK’. It is cleared by reading the status register, at power on reset, or when the LCDC is disabled.
1 EOF	End of Frame. Indicates whether the end of frame has reached. It is cleared by reading the status register, at power on reset, or when the LCDC is disabled.
0 BOF	Beginning of Frame. Indicates whether the beginning of frame has reached. It is cleared by reading the status register, at power on reset, or when the LCDC is disabled.

Bit Value Meaning: 0 = Interrupt has not occurred, 1 = Interrupt has occurred

43.3.19 LCDC Graphic Window Start Address Register (LGWSAR)

LGWSAR defines the starting address of the graphic window image. See Figure 43-3.

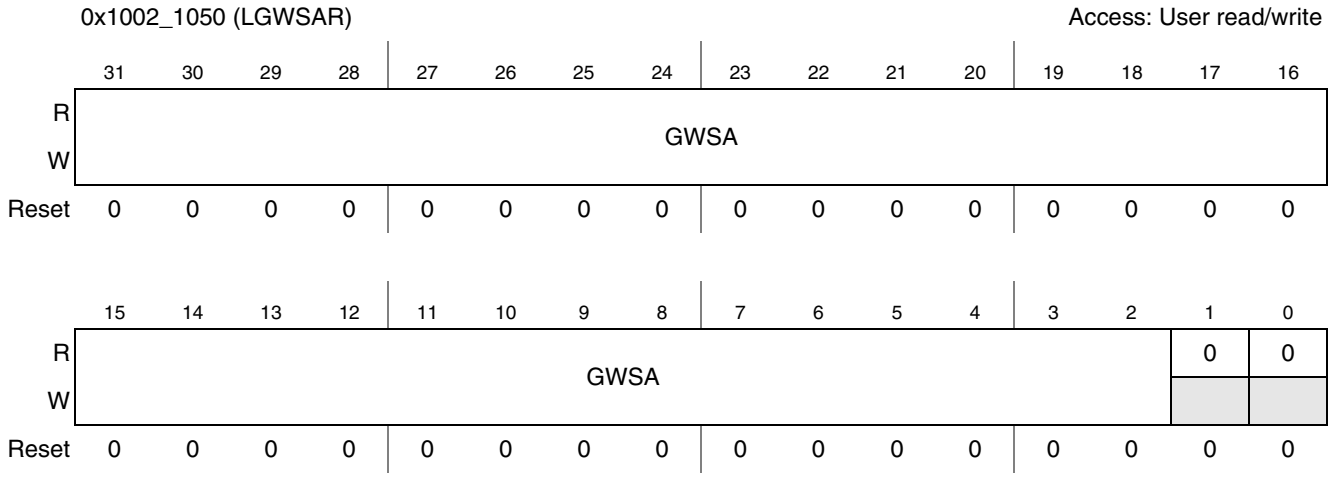


Figure 43-41. LCDC Graphic Window Start Address Register (LGWSAR)

Table 43-26. LCDC Graphic Window Start Address Register Field Descriptions

Field	Description
31–2 GWSA	Graphic Window Start Address on LCD Screen. Holds the starting address of the graphic window picture. This field must start at a location that enables a complete graphic window picture to be stored in a 4Mbyte memory boundary (A[21:0]). A[31:22] has a fixed value for the graphic window picture's image.
1–0	Reserved. These bits are reserved and should read 0.

43.3.20 LCDC Graphic Window Size Register (LGWSR)

LGWSR defines the height and width of the graphic window on the LCD screen.

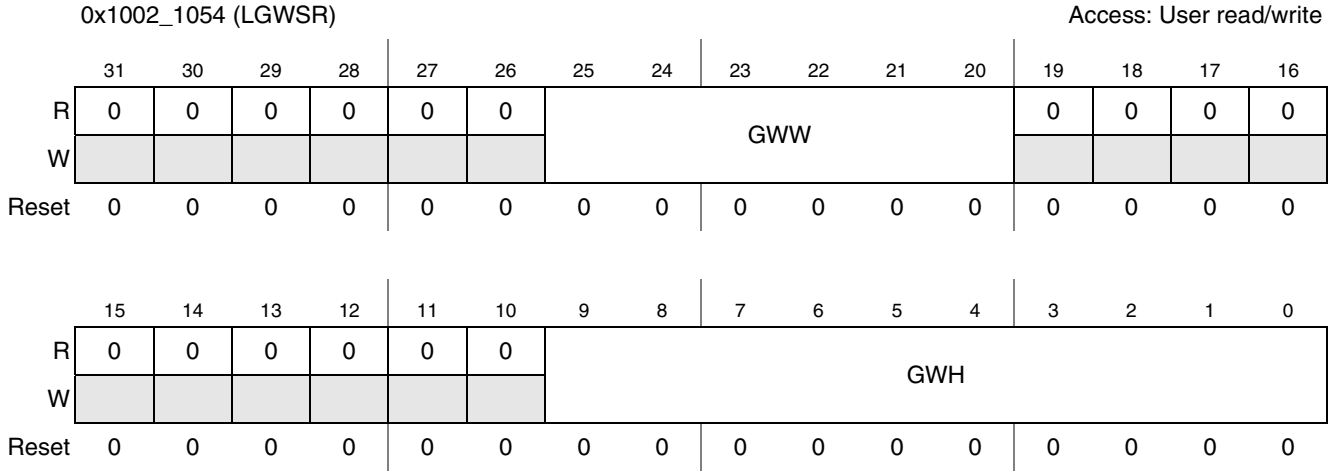


Figure 43-42. LCDC Graphic Window Start Address Register (LGWSAR)

Table 43-27. LCDC Graphic Window Size Register Field Descriptions

Field	Description
31–26	Reserved. These bits are reserved and should read 0.
25–20 GWW	Graphic Window Width Divided by 16. Holds graphic window x-axis size divided by 16. For black-and-white panels (1 bpp), GW_XMAX [20] is ignored, forcing the x-axis of the screen size to be a multiple of 32 pixels/line. **Note:** Graphic window size cannot be set to 0.
19–10	Reserved. These bits are reserved and should read 0.
9–0 GWH	Graphic Window Height. Specifies height of the graphic window in terms of pixels or lines. The lines are numbered from 1 to GW_YMAX for a total of GW_YMAX lines. **Note:** Graphic window size cannot be set to 0.

43.3.21 LCDC Graphic Window Virtual Page Width Register (LGWVPWR)

LGWVPWR defines the virtual page width for the graphic window picture on the LCD screen. See Figure 43-3.

0x1002_1058 (LGWVPWR) Access: User read/write

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

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

Figure 43-43. LCDC Graphic Window Virtual Page Width Register (LGWVPWR)

Table 43-28. LCDC Graphic Window Virtual Page Width Register Field Descriptions

Field	Description
31–10	Reserved. These bits are reserved and should read 0.
9–0 GWVPW	Graphic Window Virtual Page Width. Defines the virtual page width of the graphic window picture. VPW bits represent the number of 32-bit words required to hold the data for one virtual line. GWVPW is used in calculating the starting address representing the beginning of each line of the graphic window picture.

43.3.22 LCDC Graphic Window Panning Offset Register (LGWPOR)

LGWPOR sets up panning for the image.

0x1002_105C (LGWPOR) Access: User read/write

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

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

Figure 43-44. LCDC Graphic Window Panning Offset Register (LGWPOR)

Table 43-29. LCDC Graphic Window Panning Offset Register Field Descriptions

Filed	Description
31–5	Reserved. These bits are reserved and should read 0.
4–0 GWPO	Graphic Window Panning Offset. Defines the number of bits that the graphic window data from memory is panned to the left before processing. POS is read by the LCDC once at the beginning of each frame. For example, in 4 bpp mode, setting POS = 16 shifts 16 bits which means panning the image by 4 pixels left. **Note:** Shifting data more than 32 bits in graphic window should use LGWSAR register setting. 18 bpp graphic window panning should use LGWSAR register setting. To achieve panning of the final image by N bits: **Table 43-30.**

Bits Per Pixel	GWPO	Effective number of pixels panned
1	N	N
2	2N	N
4	4N	N
8	8N	N
12/16	16N	N

43.3.23 LCDC Graphic Window Position Register (LGWPR)

LGWPR is used to determine the starting position of the graphic window on the LCD panel.

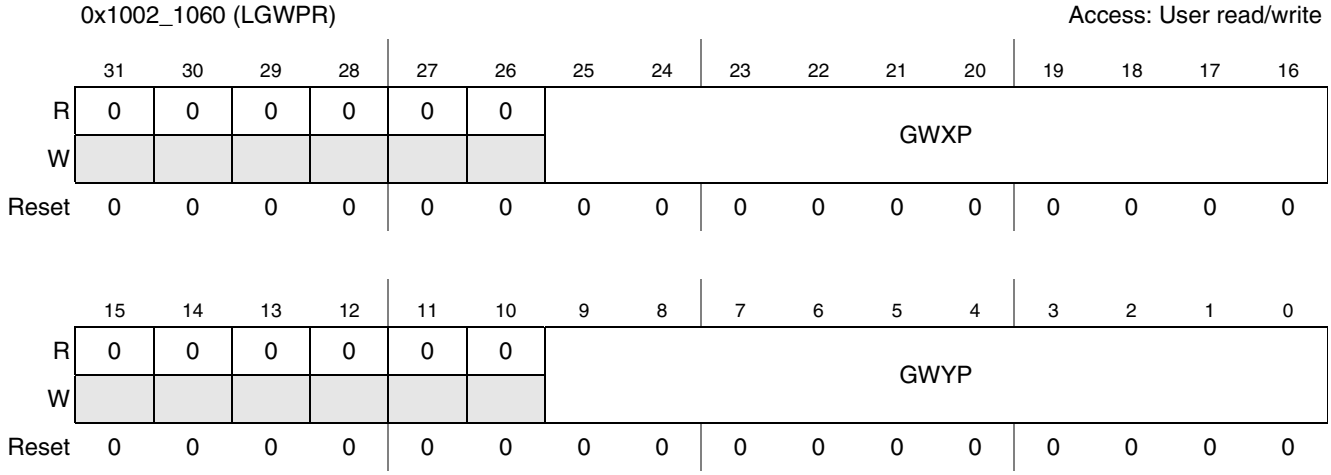


Figure 43-45. LCDC Graphic Window Position Register (LGWPR)

Table 43-31. LCDC Graphic Window Position Register Field Descriptions

Field	Description
31–26	Reserved. These bits are reserved and should read 0.
25–16 GWXP	Graphic Window X Position. Represents the graphic window's horizontal starting position in pixel count (from 0 to XMAX).
15–10	Reserved. These bits are reserved and should read 0.
9–0 GWYP	Graphic Window Y Position. Represents the graphic window's vertical starting position in line (from 0 to YMAX).

43.3.24 LCDC Graphic Window Control Register (LGWCR)

LGWCR defines the behaviors of Graphic Window.

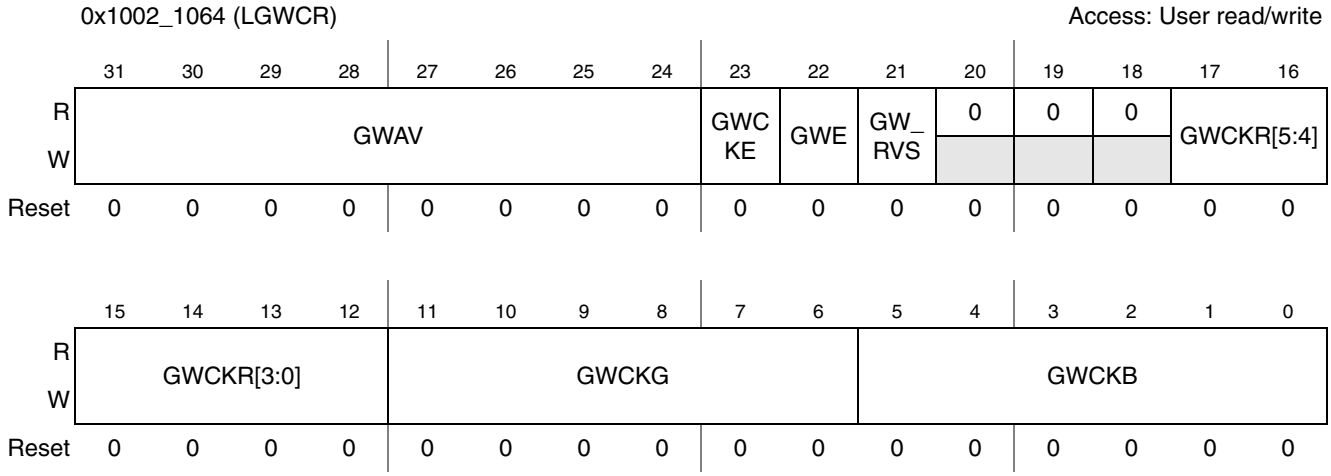


Figure 43-46. LCDC Graphic Window Control Register (LGWCR)

Table 43-32. LCDC Graphic Window Control Register Field Descriptions

Field	Description
31–24 GWAV	Graphic Window Alpha Value. Defines the graphic window alpha value used for alpha blending between graphic window and background plane 0 Graphic Window Totally Transparent, that is, not displayed on LCD screen 1 Graphic Window Totally Opaque, that is, completely visible on LCD screen
23 GWCKE	Graphic Window Color Keying Enable. Enable/disable graphic window color keying. 0 Disable Color Keying of Graphic Window 1 Enable Color Keying of Graphic Window
22 GWE	Graphic Window Enable. Enable/disable graphic window displayed on screen. 0 Disable Graphic Window on Screen 1 Enable Graphic Window on Screen
21 GW_RVS	Graphic Window Reverse Vertical Scan. Selects the graphic window vertical scan direction as normal or reverse (graphic window image flips along the x-axis). LGWSAR must be changed accordingly. 0 Vertical Scan in Normal Direction 1 Vertical Scan in Reverse Direction
20–18	Reserved. These bits are reserved and should read 0.
17–12 GWCKR	Graphic Window Color Keying Red Component 000000 No red … 111111 Full red
11–6 GWCKG	Graphic Window Color Keying Green Component 000000 No green … 111111 Full green
5–0 GWCKB	Graphic Window Color Keying Blue Component 000000 No blue … 111111 Full blue

43.3.25 LCDC Graphic Window DMA Control Register (LGWDCR)

There is a 128×32 bit line buffer in the LCDC that stores graphic window data from system memory. LGWDCE controls the DMA burst length and when to trigger a DMA burst in terms of the number of data bytes left in the pixel buffer.

0x1002_1068 (LGWDCR) Access: User read/write

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

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

Figure 43-47. LCDC Graphic Window DMA Control Register (LGWDCR)

Table 43-33. LCDC DMA Control Register Field Descriptions

Field	Description
31 GWBT	Graphic Window DMA Burst Type. Determines whether burst length is fixed or dynamic in graphic window plane. 0 Burst length is dynamic. 1 Burst length is fixed.
30–23	Reserved—These bits are reserved and should read 0.
22–16 GWHM	Graphic Window DMA High Mark. Establishes high mark for DMA requests. For dynamic burst length, once DMA request is made, data is loaded and the graphic window FIFO continues to be filled until the number of empty words left in the graphic window FIFO is equal to the high mark minus 2. Minimum HM setting in dynamic burst is 3. For fixed burst length, burst length (in words) of each request is equal to the DMA high mark setting and its value must be larger than TM.
15–7	Reserved. These bits are reserved and should read 0.
6–0 GWTM	Graphic Window DMA Low Mark. Sets low level mark in the graphic window FIFO to trigger a DMA request. Low level mark equals the number of words left in the pixel buffer.

43.3.26 LCDC AUS Mode Control Register (LAUSCR)

LCDC AUS Mode Control Register is used to determine the behavior of LCDC in AUS mode.

0x1002_1080 (LAUSCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	AUS Mode	0	0	0	0	0	0	0	Graphic Window Color Key Red – AGWCKR[7:0] (AUS mode only)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Graphic Window Color Key Green – AGWCKG[7:0] (AUS mode only)								Graphic Window Color Key Blue – AGWCKB[7:0] (AUS mode only)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-48. LCDC AUS Mode Control Register (LAUSCR)

Table 43-34. LCDC AUS Mode Control Register Field Descriptions

Field	Description
31 AUS Mode	AUS Mode Control. Determines whether LCDC enters AUS mode or not. When BPIX = 101, supports 16bpp AUO panel. When BPIX = 110, supports 24bpp AUO panel. 0 Normal Mode 1 AUS Mode
30–24	Reserved. These bits are reserved and should read 0.
23–16 AGWCKR	AUS Graphic Window Color Keying Red Component. Defines the red component of graphic window color keying. (AUS mode only)
15–8 AGWCKG	AUS Graphic Window Color Keying Green Component. Defines the green component of graphic window color keying. (AUS mode only)
7–0 AGWCKB	AUS Graphic Window Color Keying Blue Component. Defines the Blue component of graphic window color keying.(AUS mode only)

43.3.27 LCDC AUS Mode Cursor Control Register (LAUSCCR)

LCDC AUS Mode Cursor Control Register is used to determine the color map of cursor in AUS mode.

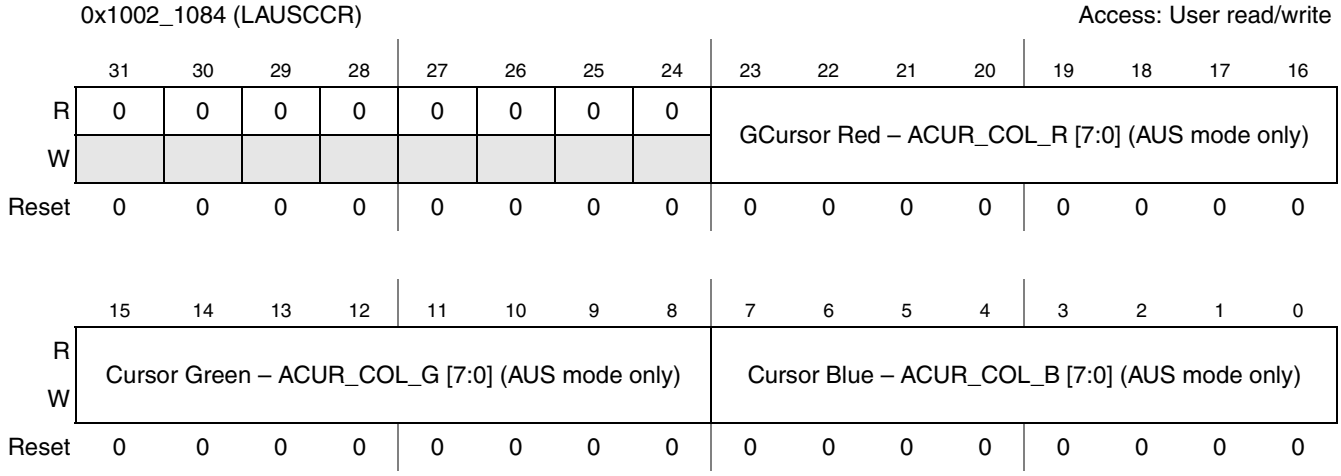


Figure 43-49. LCDC AUS Mode Cursor Control Register (LAUSCCR)

Table 43-35. LCDC AUS Mode Cursor Control Register Field Descriptions

Field	Description
31–24	Reserved. These bits are reserved and should read 0.
23–16 ACUR_COL_R	AUS Cursor Red Field. Defines the red component of cursor color in color mode.(AUS mode only)
15–8 ACUR_COL_G	AUS Cursor Red Field. Defines the green component of cursor color in color mode.(AUS mode only)
7–0 ACUR_COL_B	AUS Cursor Red Field. Defines the Blue component of cursor color in color mode.(AUS mode only)

43.3.28 BGLUT and GWLUT

There are 2 separate mapping RAMs in LCDC: background lookup table (BGLUT) and graphic window lookup table (GWLUT). BGLUT is for background plane and the mapping table is addressable from 0x10021800–0x10021BFC. GWLUT is for graphic window and its mapping table is addressable from 0x10021C00–0x10021FFC. These mapping RAMs are used for mapping 4-bit codes for grayscale to 16 gray shades, and for mapping 4-bit color and 8-bit color to 16 colors and 256 colors, respectively, out of either a palette of 4096 (passive panels) or a palette of 256K (active panels).

Mapping RAM contains 256 entries and each entry is 18-bits wide. Each RAM entry uses 4 bytes of address space. RAM is accessed with word transactions only and the address must be word aligned. Unimplemented bits are read as 0. Byte or halfword access to the RAM corrupts its contents. All read and write data use the least significant 12 or 18 bits.

In 4 bpp mode, first sixteen RAM entries are used. In 8 bpp mode, all 256 RAM entries are used. The color RAM is not initialized at reset. Only the following settings use the mapping RAMs:

- 4 bpp gray-scale mode
- 4 bpp passive matrix color mode

- 8 bpp passive matrix color mode
- 4 bpp active matrix color mode
- 8 bpp active matrix color mode

43.3.28.1 Four Bits Per Pixel Gray-Scale Mode

In 4bpp gray-scale mode, a 4-bit code represents a gray-scale level. The first 16 mapping RAM entries must be written to define the codes for all 16 combinations.

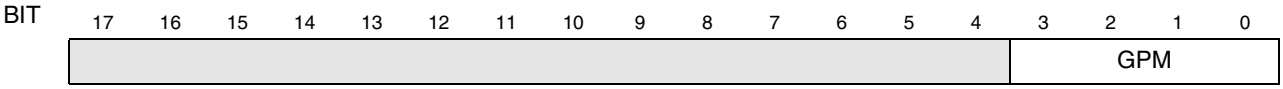


Table 43-36. Four Bits Per Pixel Gray-Scale Mode

Bits	Field	Description
11–4	Reserved	These bits are reserved and should read 0.
3–0	GPM	Gray Palette Map. Represents the gray-scale level for a given pixel code.

43.3.28.2 Four Bits Per Pixel Passive Matrix Color Mode

In 4bpp passive matrix color mode, a 4-bit code represents a 12-bit color. Because just 4-bits are used to encode the color, a maximum of 16 colors can be selected out of a palette of 4096. The first 16 mapping RAM entries must be written to define the codes for the 16 available combinations.

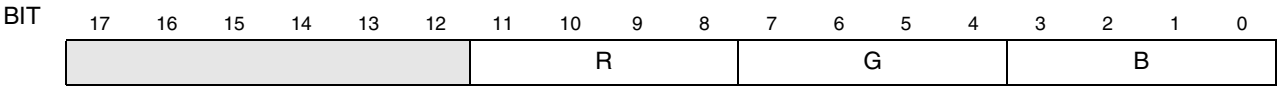


Table 43-37. Four Bits Per Pixel Passive Matrix Color Mode

Bits	Field	Description
11–8	R	Red Level (color display). Represents the red component level in the color.
7–4	G	Green Level (color display). Represents the green component level in the color.
3–0	B	Blue Level (color display). Represents the blue component level in the color.

43.3.28.3 Eight Bits Per Pixel Passive Matrix Color Mode

In 8bpp passive matrix color mode, an 8-bit code represents a 12-bit color. Because 8-bits are used to encode the color, a maximum of 256 colors can be selected out of a palette of 4096. All 256 mapping RAM entries must be written to define the codes for the 256 available combinations.

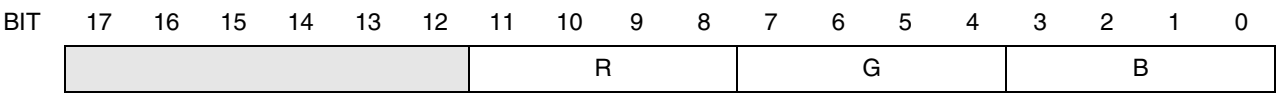


Table 43-38. Eight Bits Per Pixel Passive Matrix Color Mode

Bits	Field	Description
11–8	R	Red Level (color display). Represents the red component level in the color.
7–4	G	Green Level (color display). Represents the green component level in the color.
3–0	B	Blue Level (color display). Represents the blue component level in the color.

43.3.28.4 Four Bits Per Pixel Active Matrix Color Mode

In 4bpp active color mode, a 4-bit code represents a 18-bit color. Because just 4-bits are used to encode the color, a maximum of 16 colors can be selected out of a palette of 256K. The first 16 mapping RAM entries must be written to define the codes for the 16 available combinations.

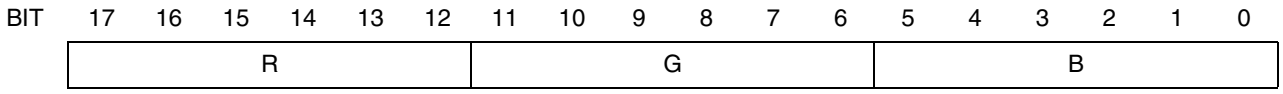


Table 43-39. Four Bits Per Pixel Active Matrix Color Mode

Bits	Field	Description
17–12	R	Red Level (color display). Represents the red component level in the color.
11–6	G	Green Level (color display). Represents the green component level in the color.
5–0	B	Blue Level (color display). Represents the blue component level in the color.

43.3.28.5 Eight Bits Per Pixel Active Matrix Color Mode

In 8bpp active color mode, an 8-bit code represents an 18-bit color. Because 8-bits are used to encode the color, a maximum of 256 colors can be selected out of a palette of 256K. All 256 mapping RAM entries must be written to define the codes for the 256 available combinations.

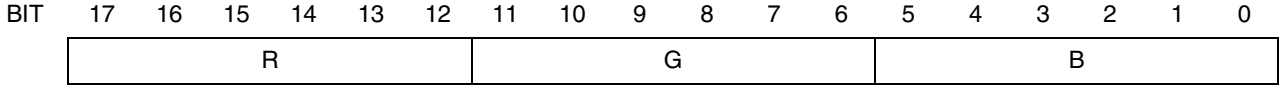


Table 43-40. Eight Bits Per Pixel Active Matrix Color Mode

Bits	Field	Description
17–12	R	Red Level (color display). Represents the red component level in the color.
11–6	G	Green Level (color display). Represents the green component level in the color.
5–0	B	Blue Level (color display). Represents the blue component level in the color.

Chapter 44
Smart Liquid Crystal Display Controller (SLCDC)

The SLCDC module transfers data from display memory buffer to external display device. DMA transfers data transparently with minimal software intervention. DMA Bus utilization is controllable and deterministic.

As cellular phone displays become larger and more colorful, demands on the processor increase. More CPU power is needed to render and manage the image. The role of the display controller is to reduce the CPU's involvement in the data transfer from memory to the display device so that CPU can concentrate on image rendering. DMA is used to optimize the transfer. Embedded control information needed by the display device is automatically read from a second buffer in system memory and inserted into the data stream at the proper time to completely eliminate the CPU's role in the transfer.

A typical scenario for a cellular phone display is to have display image rendered in main system memory. After the image is complete, CPU triggers the SLCDC module to transfer the image to the display device. Image transfer is accomplished by burst DMA which steals bus cycles from CPU. Cycle stealing behavior is programmable and bus use can be kept within predefined bounds. After transfer is complete, a maskable interrupt is generated indicating the status. For animated displays, it is suggested to implement a 2-buffer ping-pong scheme so that when DMA is fetching data from 1 buffer, next image is rendered into the other.

Several display sizes and types are used in various products that uses SLCDC. SLCDC module has the capability of directly interfacing to the selected display devices. Both serial and parallel interfaces are supported. SLCDC module only supports write to the display controller. SLCDC read operations from the display controller are not supported.

44.1 SLCDC Module Pin List

Table 44-1 is a list of the SLCDC module pins.

Table 44-1. SLCDC Module Pin List

Pin Name	Direction	Description
LCD Interfaces		
SLCDC_LCD_DATA[15:0]	Output	Data bus used to write information to external LCD controller.
SLCDC_LCD_CS	Output	Used as a chip select for external display controller in serial mode. In parallel mode, used as write strobe for the external display controller. This signal polarity is programmable.
SLCDC_LCD_RS	Output	LCD Register Select signal that indicates to the LCD device whether data being written is display data or control data. This signal polarity is programmable.

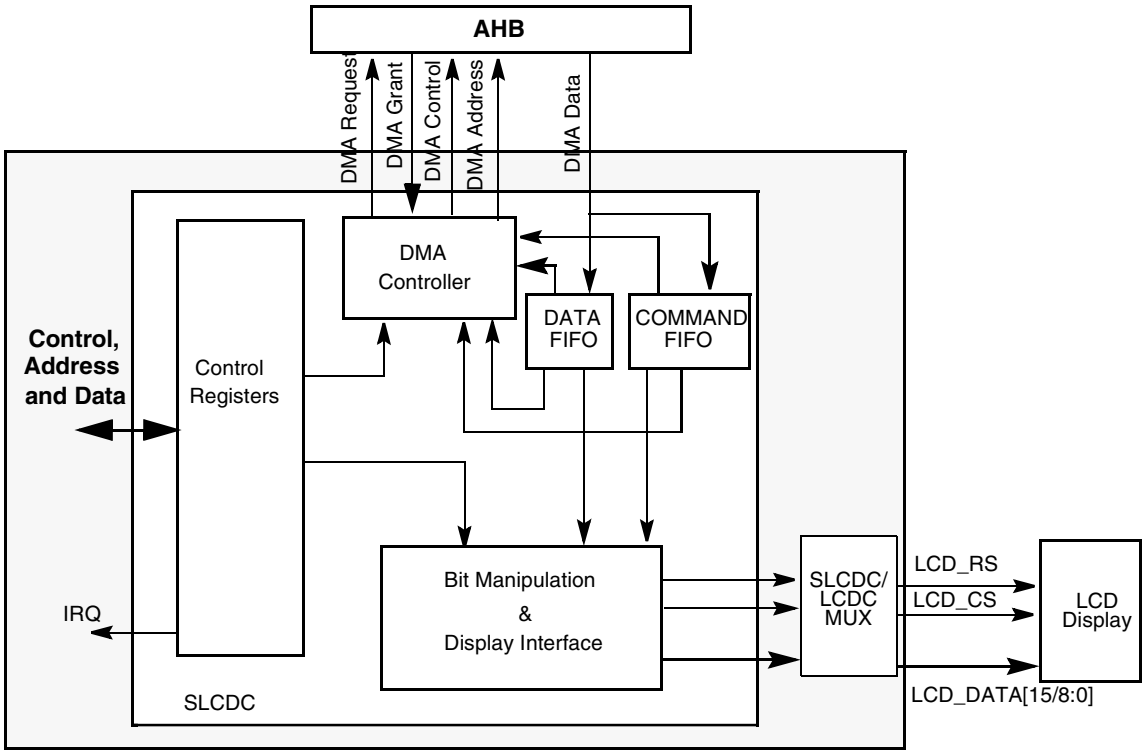


Figure 44-1. SLCDC System Diagram

44.2 Functional Description

The purpose of SLCDC is to transparently and efficiently transfer image data from system memory to an external LCD controller. System memory can be either internal or external memory. Figure 44-1 shows the block diagram of SLCDC module within the system. SLCDC module contains a DMA controller that is designed to operate as an alternate master. SLCDC DMA interface requests control of AHB to do 32-bit reads from system memory.

The purpose of DMA within SLCDC is to transfer image and control data from system memory to SLCDC FIFO where it is formatted and sent out to the external LCD controller. DMA only performs read accesses from system memory. The data is collected in 32-bit words and stored in two internal FIFOs. Data is pulled from the FIFOs, as needed, and put in correct format for the external LCD controller.

SLCDC has both control and status registers which are accessible via IP bus. These registers are used to store information about the type of LCD controller attached to the system. SLCDC can be configured to write image data to an external LCD controller via a 4-line serial, 3-line serial, an 8-bit or 16-bit parallel interface.

During automatic transfers, SLCDC is responsible for properly sequencing display data and control strings in a data stream that is sent to the LCD controller to fill its internal display RAM. Display data is typically written to the external controller in pages. Each write to the controller fills one column of the current page. SLCDC assumes that the controller automatically increments its display RAM pointers after each write.

NOTE

Changing SLCDC configuration in middle of transfer can cause corruption of data stream output to LCD. SLCDC configuration must only be changed when GO bit and BUSY bit of SLCDC control/status register are cleared.

44.2.1 Word Size Definition

Due to increased complexity of color displays, SLCDC supports 16- or 8-bit transfers of command or display data. This is controlled by WRD_DEF_COM and WRD_DEF_DAT bits in LCD transfer configuration register. All registers in SLCDC are defined in words.

WRD_DEF_DAT controls definition of word for all data registers:

- DATA_BUFFER_SIZE
- LCD_CONFIG

WRD_DEF_COM controls the definition of word for all command registers:

- COMMAND_BUFFER_SIZE

WRD_DEF_WRITE controls the definition of LCDDAT for the register:

- LCD_WRITE_DATA

44.2.2 Image Endianness

In the examples of automatic transfers shown in this document, all memory images are either big endian or 32-bit little endian. However, it is possible that user may have an image that is stored in half word (16-bit) or byte (8-bit little endian). SLCDC can compensate for this by using the IMGEND bits in the LCD_TRANSFER_CONFIG register. These bits causes SLCDC to convert 16- or 8-bit little endian image to a big endian image. The resultant big endian decode is used for the rest of the data processing.

Table 44-2. Image Endianness

IMGEND	Conversion	32-Bit Word Data Stored in Memory	32-Bit Data After Conversion
00	Big endian or 32-bit little endian to big endian	0x12345678	0x12345678
01	16-bit little endian to big endian	0x56781234	0x12345678
10	8-bit little endian to big endian	0x78563412	0x12345678

44.2.3 Accessing the LCD Controller

SLCDC provides two methods of accessing the external display device.

- The first, and preferred method, is through automatic writes handled by SLCDC.
- The second, is through direct IP register access via LCD write data register.

44.2.3.1 Automatic SLCDC Transfers

SLCDC is designed to take a block of data from system memory and write it, without software intervention, to an external display controller. This is done by giving SLCDC a starting address and a data block size (in words). This method can be used for both LCD display data as well as command data. Transfers are initiated by setting the GO bit in SLCDC control/status register. Data transfer is done in the background. Upon completion, a maskable interrupt is generated. Automatic transfers are accomplished in one of three ways:

- A block of data is transferred to the LCD controller with control strings automatically inserted to navigate through the LCD controller's internal RAM. This mode makes use of a second buffer in system memory to store the software commands necessary for LCD controller RAM addressing.
- A block of data is transferred from system memory to the display controller while tying the LCD_RS pin low.
- A block of data is transferred from system memory to the display controller while tying the LCD_RS pin high.

Automatic transfer mode control bits (AUTOMODE[1:0]) of SLCDC control/status register configures the SLCDC for one of the three supported automatic transfer modes.

44.2.3.1.1 Automatic Display Data Transfers (AUTOMODE[1:0][1]=10)

SLCDC is designed to display a frame buffer from system memory to external LCD controller without any software intervention while transferring. To accomplish this, SLCDC must automatically transmit control strings for LCD controller RAM addressing at appropriate times in the frame buffer data stream. LCD controller RAMs are typically organized into pages. After RAM page is filled, controller's page address must be set to the next page. SLCDC contains counters that monitor the number of display data words written to the LCD controller to determine when the current page of RAM is filled. After this page is filled, SLCDC inserts a control string into the data stream to set controller RAM's page address to next page. SLCDC requires this control string be stored in a command buffer in system memory. The organization of command buffer is discussed later in this section. To correctly move a complete frame of image data from system memory to the LCD controller, SLCDC must be programmed with the following LCD information:

- Command word definition (8-bit or 16-bit words)
- Data word definition (8-bit or 16-bit words)
- Number of words in a page of LCD image data
- Number of words in the LCD frame data buffer
- Type of LCD interface (serial or parallel)
- Data clock polarity of the LCD (for serial interfaces only)
- Endianness of the image (big endian, little 32-bit, little 16-bit, or little 8-bit)
- Chip-select polarity for the LCD
- Transfer clock speed requirements of the LCD
- Control string needed at the start of each page of LCD display data (stored in a separate command buffer in system memory)

1. Automode [1:0] = 11 is reserved.

SLCDC uses number of words per page of LCD image data information (stored in WORDPPAGE[12:0] bits of LCD CONFIG register) to determine when a set of addressing commands must be sent to the LCD. When the current display RAM page is full, SLCDC must increment the page address and reset the column address of the LCD controller to 0. This is accomplished by reading the next control string from the command buffer and transmitting it to the LCD controller. SLCDC continues to send display data and command strings as needed until the controller's display RAM is filled. Transfer is finished when the number of data words transferred equals the number of words in the buffer as defined by DATABUFSIZ bits in DATA BUFFER SIZE registers.

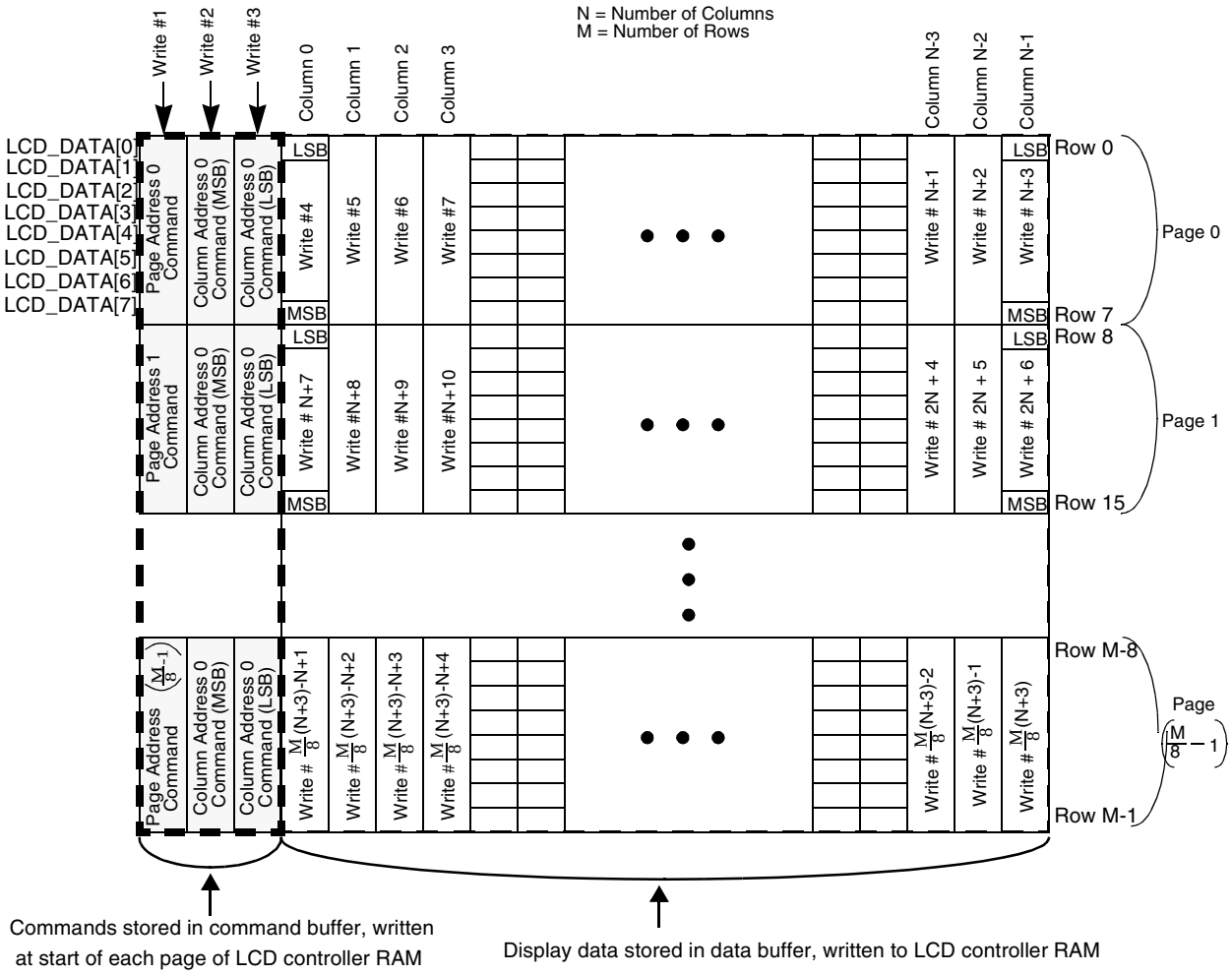


Figure 44-2. Sample LCD Controller Memory Mapping (Monochrome) (8-Bit Command/8-Bit Data)

Figure 44-2 shows an example of how an external LCD controller RAM is filled during automatic SLCDC writes for a monochrome display. Command and data word sizes are both defined as bytes. In this example, LCD controller requires three "command" words (bytes) to set the page and column address. Setting GO bit of control/status register starts the data transfer from system memory to the LCD controller. The first word written to LCD sets the page address to 0. The next two words written are control strings that set the LCD column address to 0. Word write #4 starts transfer of image data to the LCD. The first word of display data maps to the display column 0. The MSB of the word maps to row 7 and the LSB maps to row 0.

Words of display data continue to be written to LCD until SLCDC's internal word counter equals the value stored in WORDPPAGE[12:0] bits of LCD CONFIG register. At this point, SLCDC issues a command string, taken from the command buffer, to increment the page address and reset the column address of LCD. This sequence continues until LCD controller's data RAM is filled. After display buffer has been filled, IRQ bit of SLCDC control/status register is set. Also, BUSY bit and GO bit is cleared. An interrupt is generated if IRQEN bit is set and the system is configured for SLCDC interrupts.

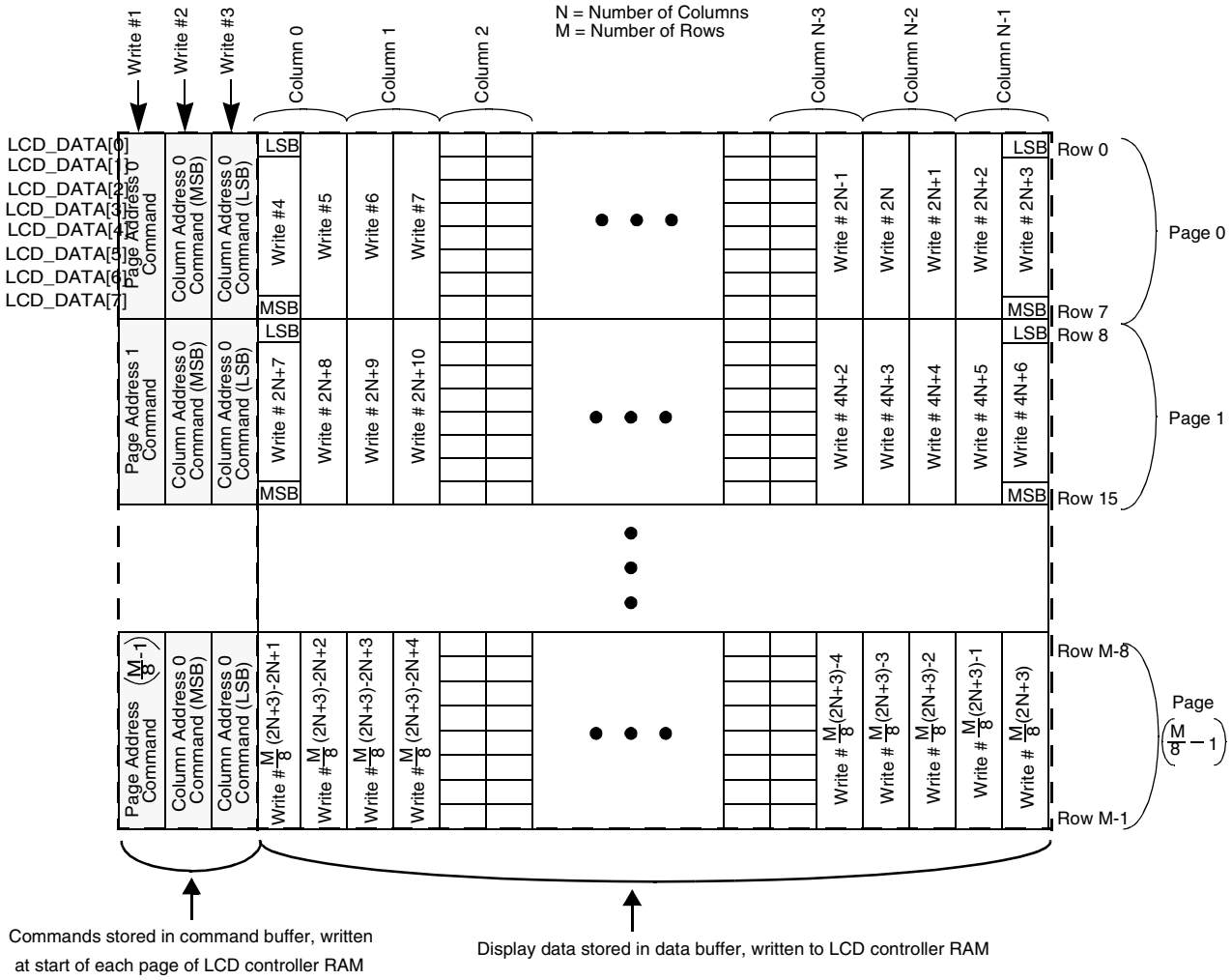


Figure 44-3. SLCDC LCD Controller Memory Mapping (2-Bit Color/Gray Scale) (8-Bit Command/8-Bit Data)

Figure 44-3 shows the mapping for a 2-bit color/gray scale display. Command and data word sizes are defined as bytes. For this configuration, each column of data within the page requires 2 word writes. The sequence remains the same as 1-bit per pixel case shown in Figure 44-2, however twice as many display data writes are required to fill the page.

SLCDC is designed to deal with a large variety of LCD controllers from multiple vendors. While the hardware interface to the external controller is somewhat standard, software interface can vary from chip to chip. To deal with the software differences between LCD controllers, SLCDC makes use of a separate

command buffer stored in system memory to go along with the display data buffer. Figure 44-4 shows the organization of the information in system memory. LCD page and column command buffer should be set up to contain the commands necessary to navigate through the LCD controller's internal RAM. Each command in the buffer is preceded by a tag. SLCDC uses the tags to determine the LCD_RS pin value during the command word transmission.

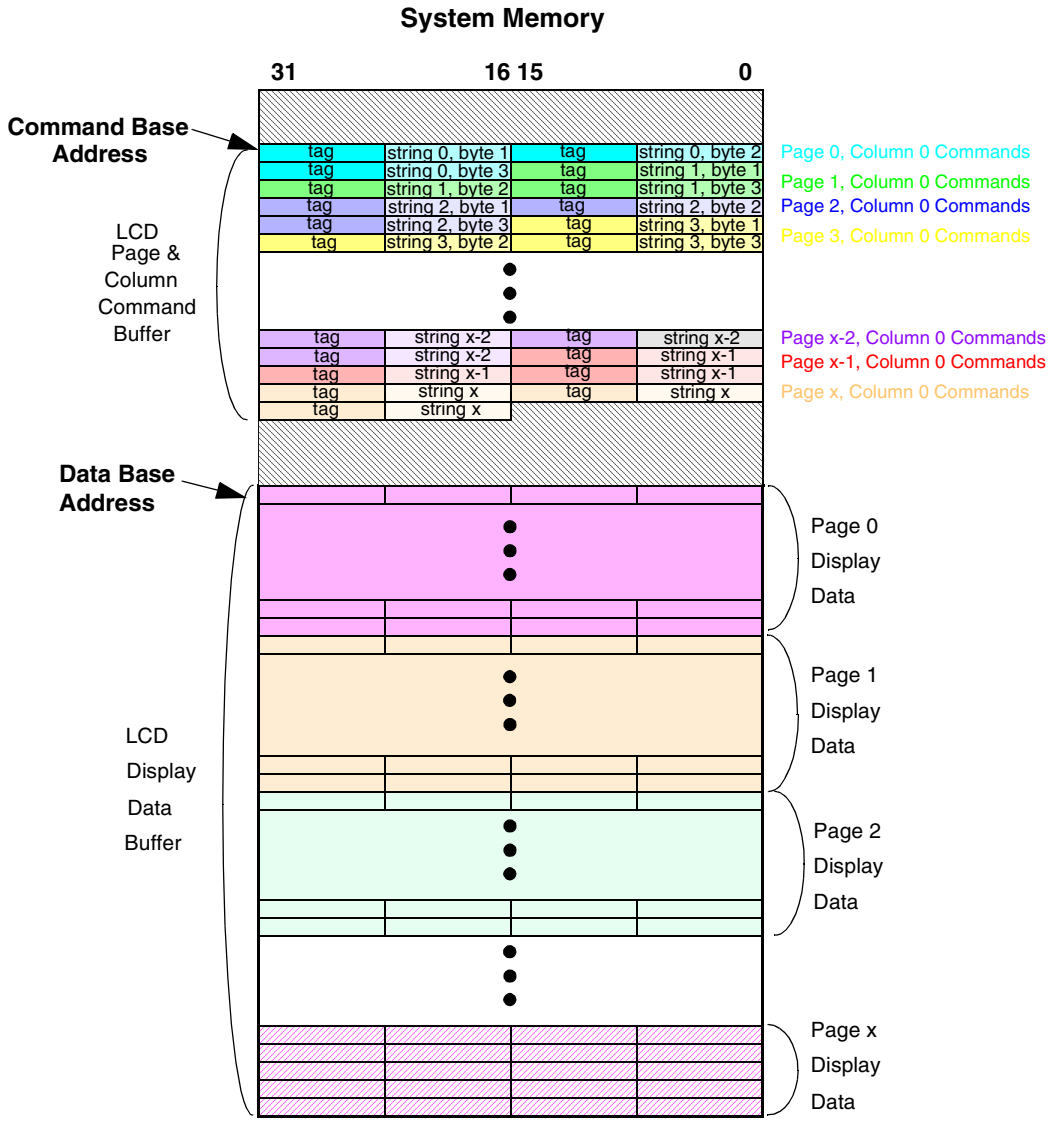


Figure 44-4. Automatic Display Data Transfer Memory Configuration (8-Bit Command/8-Bit Data)

Figure 44-5 shows how the commands and tags are organized in system memory. In this example, command string length is 3, that is, three words are required to be transmitted to the LCD controller at the start of each page of LCD RAM. Each of the command words have a tag placed adjacent to it in the memory. The command is placed immediately after its tag in the memory. Currently SLCDC only uses the least-significant bit of the tag field. This bit, labeled "RS" in Figure 44-5, sets the LCD_RS pin value while the command word is being transferred to the LCD controller. If RS tag bit is set to "1", LCD_RS is driven

to 1 during the transfer of the corresponding command byte. Similarly, if RS tag bit is set to “0”, LCD_RS is driven to 0 during the transfer of the corresponding command byte.

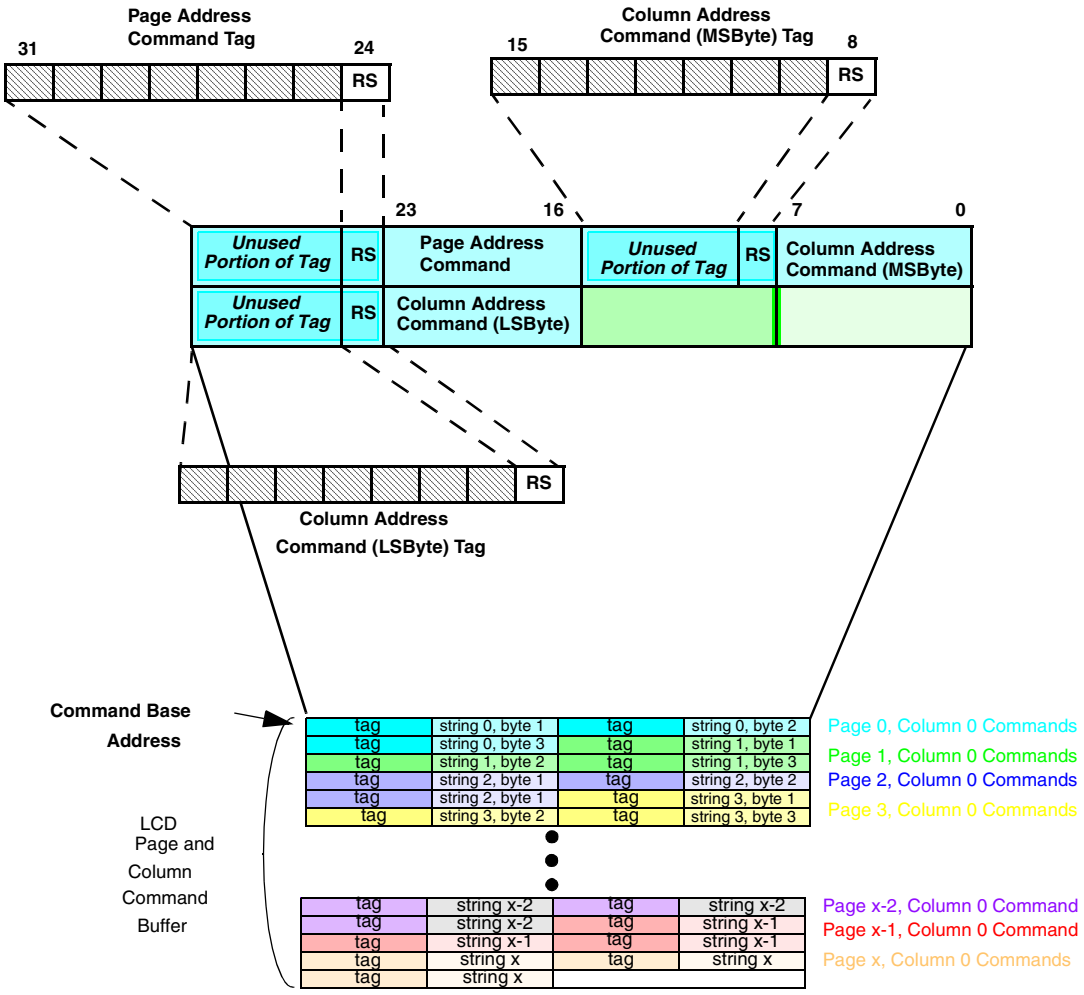


Figure 44-5. Command Buffer Tag Organization (8-Bit Words)

Figure 44-6 shows a 16-bit command and 16-bit data transaction for a 16-bit color display. In this example, command length is 3. Each LCD data transfer is for one display pixel. The mapping of display pixels for 16-bit color is generally “RRRRGGGGBBBBXXX” (R=red G=green B=blue X=ignored). Check your display to see what the 16-bit color mapping is. In this case, a page consists of a row of pixels. Three 16-bit words of command data are transferred to the display first, then 16-bit transfers of data will occur until the WORDPPAGE are transferred. Three more command words are sent designating the next page, and the cycle repeats itself.

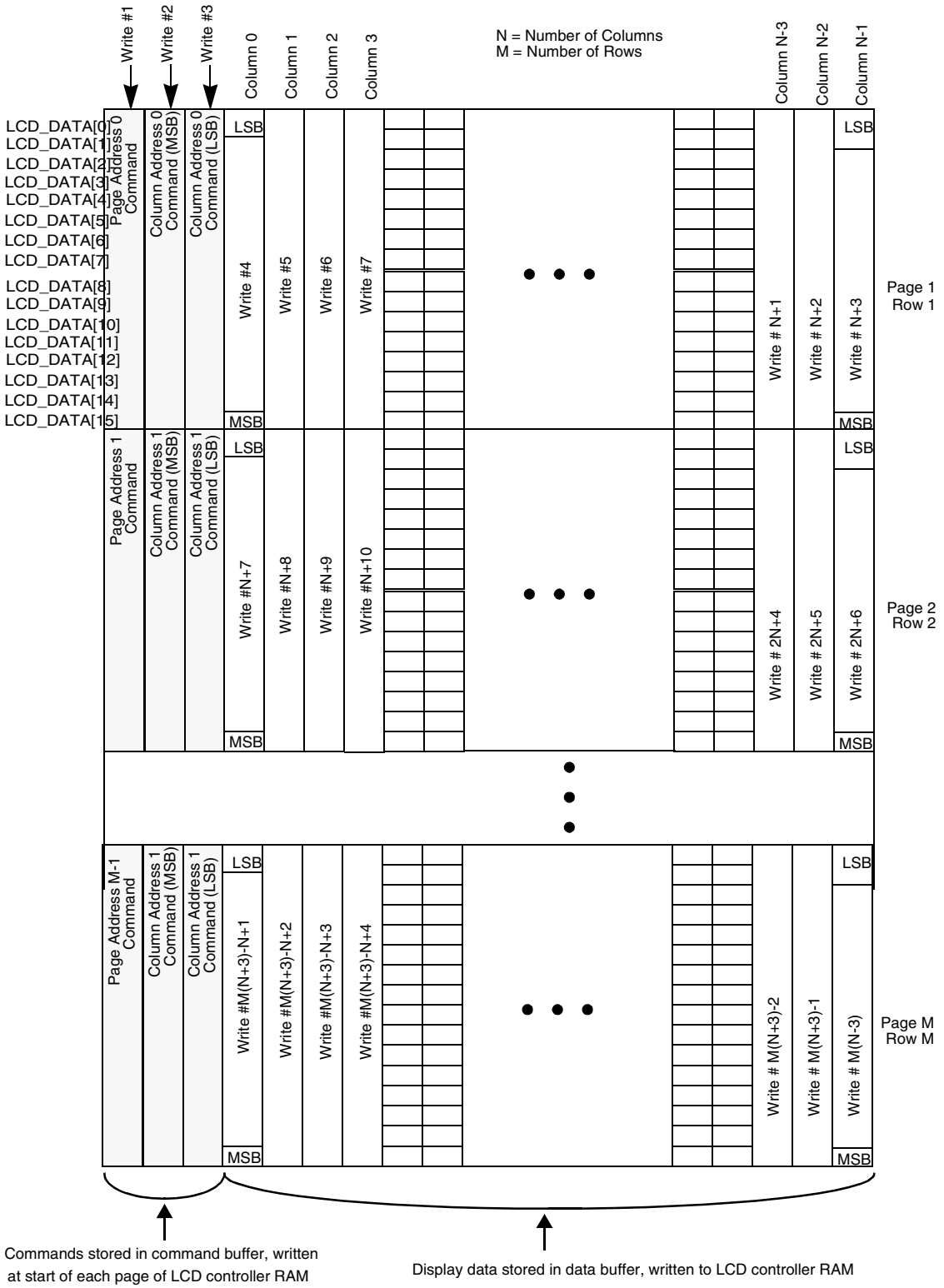


Figure 44-6. Sample LCD Controller Memory Map (16-Bit Color) (16-Bit Command/16-Bit Data)

Figure 44-7 shows how a 16-bit command word is stored in system memory. The 16-bit command is stored in the least significant half-word of memory (15–0). Tag is stored in the most significant half-word of memory (31–16). Currently only bit 16 of the tag is used for RS.

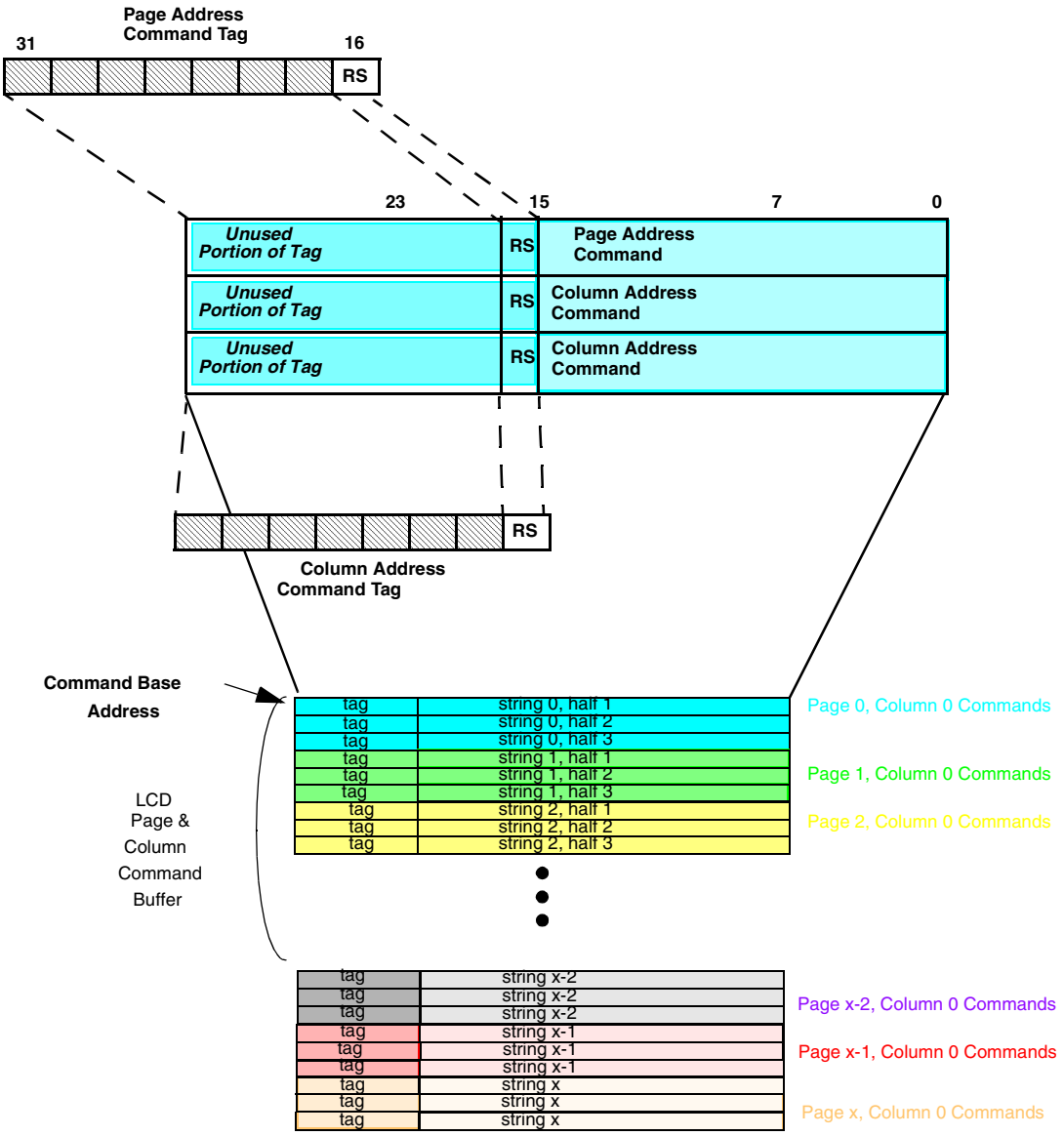


Figure 44-7. Command Buffer Tag Organization (16-Bit Words)

Figure 44-8 shows how 16-bit word commands and 16-bit word data is stored in system memory. Because command data is 17-bits long (16-bits of command + 1-bit tag), each command takes up 1 word of system memory. Each 16-bit word of data can be placed in a half-word of system memory.

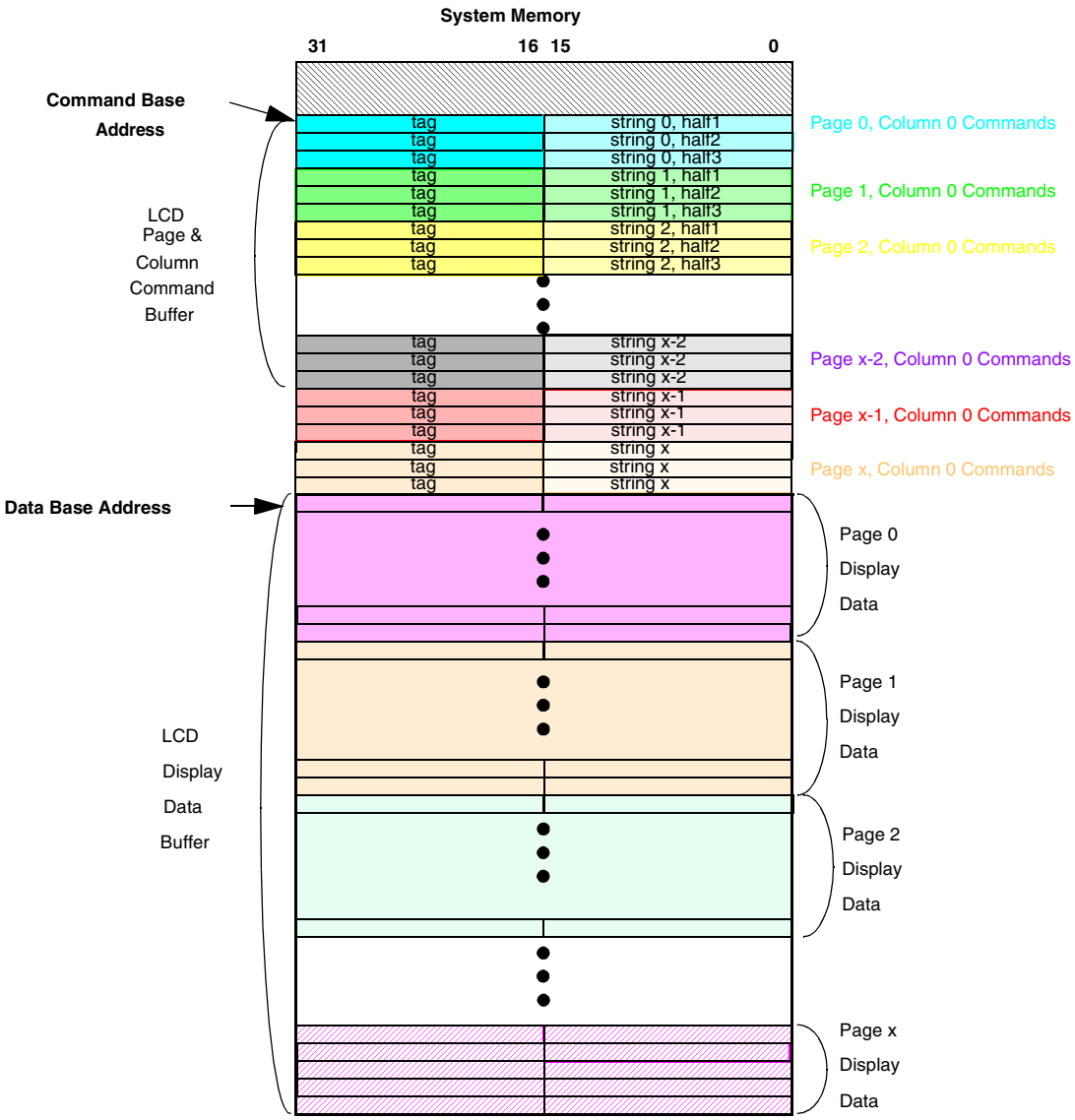


Figure 44-8. Automatic Display Data Transfer Memory Configuration (16-Bit Command/16-Bit Data)

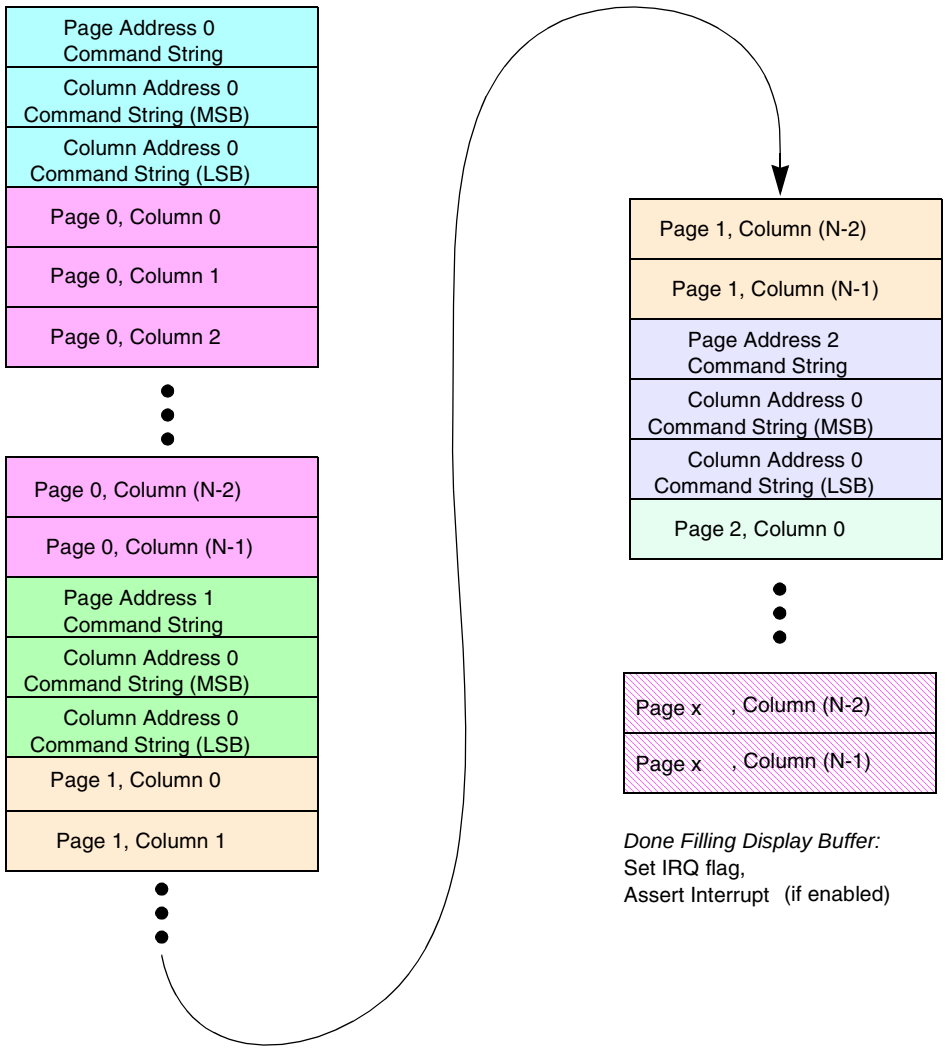


Figure 44-9. SLCDC Automatic Mode Write Sequence (Monochrome Display)

Figure 44-9 and Figure 44-10 show the sequence in which the data words are written to the LCD controller when AUTOMODE[1:0] =10. These figures illustrate how the command strings, taken from the command buffer, are interleaved with display data taken from the data buffer.

It is important to note that different combinations of word definitions are allowed. For example, there can be 8-bit command words and 16-bit data words. In this case, commands would be stored in system memory much like Figure 44-5, and data would be stored like Figure 44-8. Refer Section 44.3, “LCD Controller Interface” to determine how mixed transfers occur.

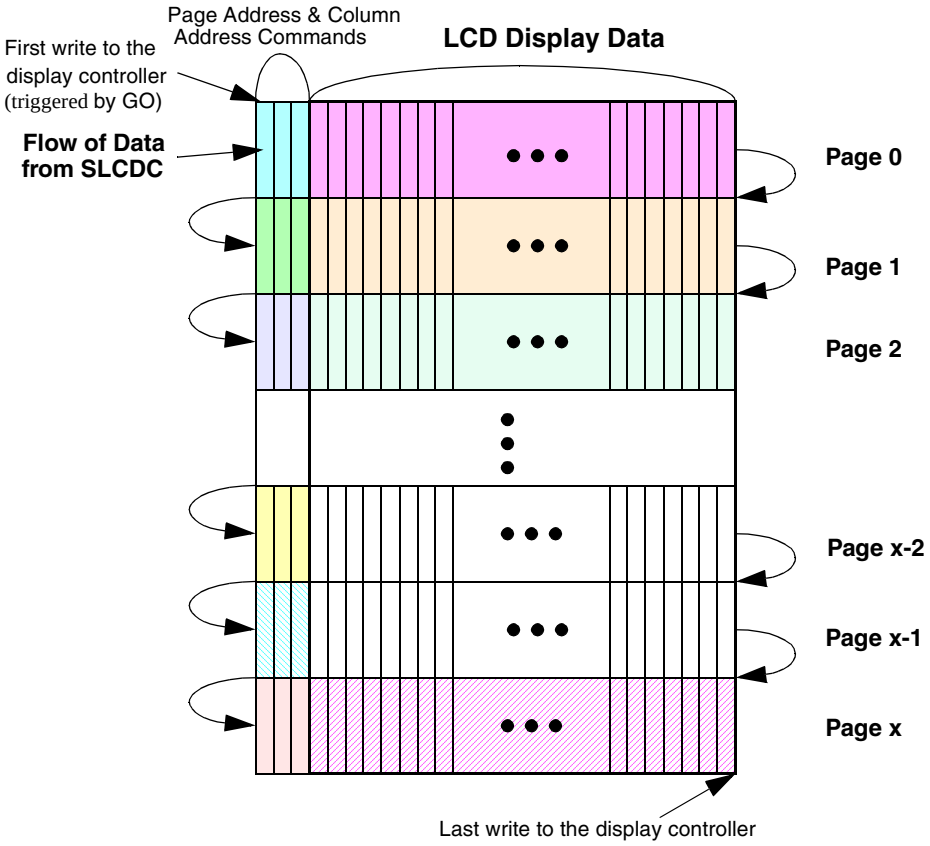


Figure 44-10. SLCDC Automatic Mode Data Flow (AUTOMODE[1:0] = 10)

44.2.3.1.2 Automatic Command Data Transfers (AUTOMODE[1:0]=00)

SLCDC provides a convenient method of moving data from system memory to an external device, like a LCD controller. When AUTOMODE[1:0] bits = 00, SLCDC ignores the command buffer and transfers the contents of specified data buffer to SLCDC output pins, without inserting the page and column addressing control strings. When AUTOMODE[1:0] = 00, LCD_RS output is held to 0 during the entire transfer. This method is used for sending a block of commands to the LCD controller for initialization. The buffer transmitted is defined by DATABASEADR[16:0] bits of DATA BASE ADDRESS register and DATABUFSIZE[16:0] bits of DATA BUFFER SIZE register. Setting GO bit of SLCDC control/status register starts the transfer and sets the BUSY status bit. After the number of words transferred to the external device is equal to the buffer size defined by the DATA BUFFER SIZE register, BUSY and GO bit will be cleared, and IRQ flag will be set. A system interrupt is generated if IRQEN bit is set.

44.2.3.1.3 Automatic Command Data Transfers (AUTOMODE[1:0]=01)

SLCDC provides a convenient general purpose method of moving data from system memory to an external device, like a LCD controller. When AUTOMODE[1:0] bits = 01, SLCDC ignores the command buffer and transfers the contents of the specified data buffer to the SLCDC output pins without inserting page and column addressing control strings. When AUTOMODE[1:0] = 01, LCD_RS output is held to 1 during the entire transfer. The buffer transmitted is defined by DATABASEADR[31:0] bits of DATA BASE ADDRESS register and DATABUFSIZ[16:0] bits of DATA BUFFER SIZE register. Setting GO bit of

SLCDC control/status register starts the transfer and sets the BUSY status bit. After the number of words transferred to the external device is equal to the buffer size defined by DATA BUFFER SIZE register, BUSY and GO bit is cleared, and IRQ flag is set. A system interrupt is generated if IRQEN bit is set.

44.2.3.2 Direct Register Access

Single words are written to the external LCD controller via 9-bit LCD WRITE DATA register. Any write to the LCD WRITE DATA register results in a write to the external display controller, provided that SLCDC is not currently in middle of a transfer (indicated by BUSY=1 in control/status register). The value written to LCDDAT[15:0] bits of LCD write data register are transmitted according to settings stored in LCD transfer configuration register, that is, transfer is done according to the settings of WORDDEFWRITE, XFRMODE, CSPOL, and SCKPOL bits of LCD transfer configuration register.

Typical LCD controllers use a register select signal to distinguish between display data writes and control command writes. RS bit of LCD WRITE DATA register determines whether a write is presented to the external controller as a command string or a display data string. LCD_RS pin is held to the value stored in the RS bit during byte transfer. Since transfer to the external device is slow compared to the system clock rates, BUSY bit in the control/status register is used to indicate that a transfer is in progress. A transfer initiated by a write to the WRITE DATA register causes an interrupt if IRQEN bit of SLCDC control/status register is set.

WORDDEFWRITE in LCD transfer configuration register determines if an 8- or 16-bit transfer occurs. An 8-bit transfer causes LCDDAT[7:0] bits to be transferred, and LCDDAT[15:8] to be ignored. Another direct write cannot be done until the previous transfer has completed (BUSY=0).

44.2.4 Aborting SLCDC Transfers

ABORT bit in SLCDC control/status register is used to terminate a SLCDC transfer that is in progress. Termination occurs gracefully. Any byte transfer that is in progress when ABORT is asserted is completed before the SLCDC goes to an idle state. BUSY status bit clears upon entry to the idle state. IRQ flag will also assert, indicating that the abort is complete. A SLCDC interrupt is generated, if enabled.

44.2.5 Low-Power Mode Operation

SLCDC does not contain any control registers to program SLCDC behavior during low-power modes. Any power savings in the usage of SLCDC are gained by reducing the system clock to a lower rate.

44.2.6 Memory Map

Table 44-3. SDLC Memory Map

Address	Register	Access	Reset Value	Section/Page
0x1002_2000 (DATABASEADR)	Data Buffer Base Address Register	R/W[1]	0x0000_0000	44.2.9/44-18
0x1002_2004 (DATABUFSIZE)	Data Buffer Size register	R/W	0x0000_0000	44.2.10/44-18

Table 44-3. SDLC Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x1002_2008 (COMBASEADR)	Command Base Address register	R/W	0x0000_0000	44.2.11/44-19
0x1002_200C (COMBUFSIZ)	Command Buffer Size register	R/W	0x0000_0000	44.2.12/44-19
0x1002_2010 (COMSTRINGSIZ)	Command String Size register	R/W	0x0000_0000	44.2.13/44-20
0x1002_2014 (FIFOCONFIG)	FIFO Configuration register	R/W	0x0000_0000	44.2.14/44-21
0x1002_2018 (LCDCONFIG)	LCD Controller Configuration register	R/W	0x0000_0000	44.2.15/44-21
0x1002_201C (LCDTRANSCONFIG)	LCD Transfer Configuration register	R/W	0x0000_0000	44.2.16/44-22
0x1002_2020 (SLCDCCONTROL/STATUS)	Smart LCD Control/Status register	R/W	0x0000_0000	44.2.17/44-23
0x1002_2024 (LCDCLOCKCONFIG)	LCD Clock Configuration register	R/W	0x0000_0000	44.2.18/44-26
0x1002_2028 (LCD WRITE DATA)	LCD Write register	R/W	0x0000_0000	44.2.19/44-26

[1] Some R/W registers may contain some read-only or write-only bits.

44.2.7 Register Summary

The conventions in Figure 44-11 and Table 44-4 serve as a key for register summary and individual register diagrams.

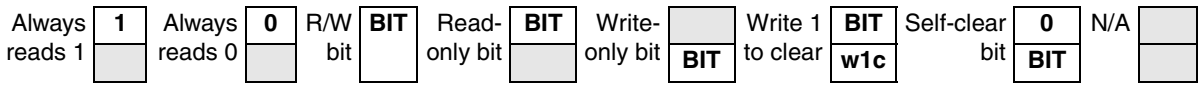


Figure 44-11. Key to Register Fields

Table 44-4. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.

Table 44-4. Register Conventions (continued)

Convention	Description
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

44.2.8 SLDC Register Descriptions

This section provides detailed descriptions of various SLCDC registers. Table 44-5 provides the detail summary for SLCDC registers.

Table 44-5. SLCDC Register Summary

Name		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
0x1002_2000 (DATABASEADR)	R	DATABASEADR[31:2]															
	W																
	R	DATABASEADR[31:2]														0	0
	W																
0x1002_2004 (DATABUFSIZE)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																
	R	DATABUFSIZ[16:0]															
	W																
0x1002_2008 (COMBASEADR)	R	COMBASEADR[31:2]															
	W																
	R	COMBASEADR[31:2]														0	0
	W																
0x1002_200C (COMBUFSIZ)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	COMBUFSIZ[16:0]
	W																
	R	COMBUFSIZ[16:0]															
	W																

Table 44-5. SLCDC Register Summary (continued)

Name		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
0x1002_2010 (COMSTRINGSIZ)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	COMSTRINGSIZ[7:0]							
	W																
0x1002_2014 (FIFOCONFIG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BURST	
	W																
0x1002_2018 (LCDCONFIG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	WORDPPAGE[12:0]												
	W																
0x1002_201C (LCDTRANSCONFIG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IMGEND	
	W																
	R	0	0	0	0	0	0	0	0	0	0	WORD DEF WRITE	WOR DDEF DAT	WOR DDEF COM	XFR MOD E	CS POL	SCK POL
	W																
0x1002_2020 (SLCDCCONTROL/STATUS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	AUTOM ODE [1:0]		0	0	PR OT1	IRQ EN	IRQ	UNDR FLOW	IRQ	0	BUS Y	0	0
	W										W1C	W1C	W1C			ABO RT	GO
0x1002_2024 (LCDCLOCKCONFIG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	DIVIDE[5:0]					
	W																
0x1002_2028 (LCD WRITE DATA)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RS
	W																
	R	LCDDAT[15:0]															
	W																

44.2.9 Data Buffer Base Address Register (DATABASEADR)

Address 0x1002_2000 (DATABASEADR) Access: User read/write

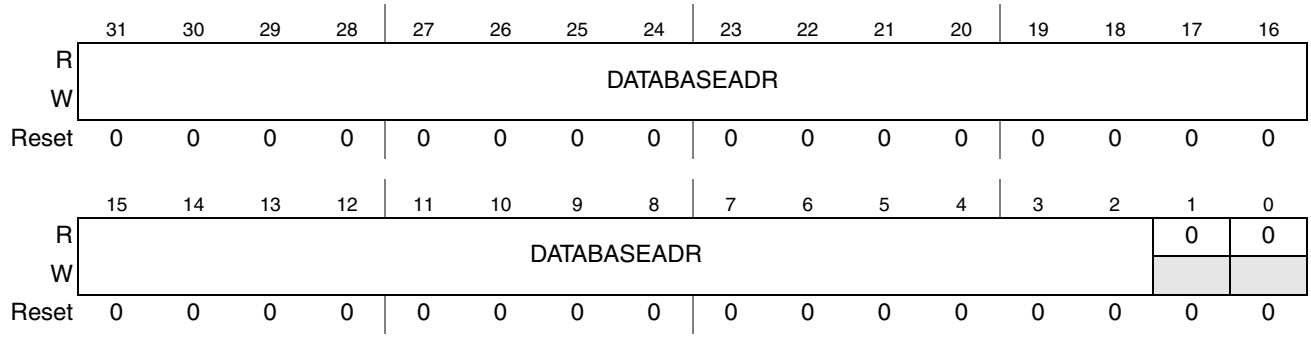


Figure 44-12. Data Buffer Base Address register

Table 44-6. Data Buffer Base Address Register Field Description

Field	Description
31–2 DATABASEADR	Data Buffer Base Address. This register contains the pointer in R-AHB address space to the start of LCD data buffer. This value is used to form the 32-bit data buffer base address. Data buffer base address value is forced to be word-aligned because the two LSBs are forced to 0. For example, data buffer base address [1:0] = 00.
1–0	Reserved. These bits are reserved and should read zero.

44.2.10 Data Buffer Size Register (DATABUFSIZE)

Address 0x1002_2004 (DATABUFSIZE) Access: User read/write

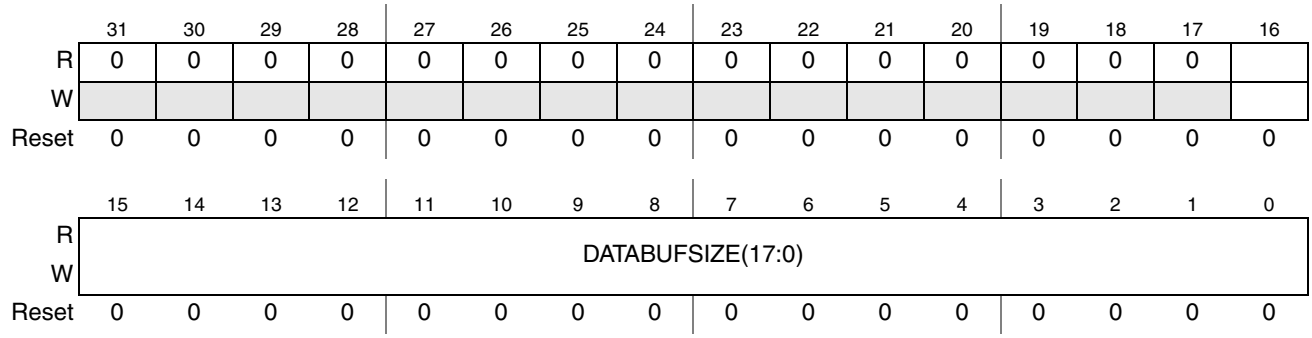


Figure 44-13. Data Buffer Size register

Table 44-7. Data Buffer Size Register Description

Field	Description
31–17	Reserved. These bits are reserved and should read zero.
16–0 DATABUFSIZ	Data Buffer Size. This register defines the length (in words) of LCD image or control buffer stored in the system memory. This value is used to determine when DMA transfer of data buffer from system memory to LCD controller is complete.

NOTE

SLCDC DMA address generator width is 17-bits. Therefore, it is possible that a data buffer can be incorrectly addressed by SLCDC, if it extends beyond a 128 kbyte boundary. Therefore, it is recommended that data buffer base address be set at the beginning of a 128 kbyte boundary. The sum of DATABASEADR[16:0] and DATABUFSIZ[16:0] must not exceed a multiple of 128K.

44.2.11 Command Buffer Base Address Register (COMBASEADR)

Address 0x1002_2008 (COMBASEADR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	COMBASEADR(31:2)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COMBASEADR(31:2)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-14. Command Buffer Base Address Register

Table 44-8. Command Base Address Register Field Description

Field	Description
31–2 COMBASEADR	Command Buffer Base Address. This register contains the pointer in R-AHB address space to the start of LCD command buffer that contains the LCD controller page and column address commands. This buffer is used when SLCDC is configured for transfers with automatic page address and column address command insertion. The command buffer base address value is forced to be word-aligned because the two LSBs are forced to 0. For example, command buffer base address [1:0] = 00.
1–0	Reserved. These bits are reserved and should read zero.

44.2.12 Command Buffer Size Register (COMBUFSIZ)

Address 0x1002_200C (COMBUFSIZ) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	COMB UFSIZ (16:0)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COMBUFSIZ (16:0)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-15. Command Buffer Size Register

Table 44-9. Command Buffer Size Register Field Description

Field	Description
31–17	Reserved. These bits are reserved and read zero.
16–0 COMBUFSIZ	Command Buffer Size. This register defines the length (in words) of LCD control buffer stored in system memory. This value is used to determine when DMA transfer of command buffer from system memory to the LCD controller is complete.

NOTE

SLCDC DMA address generator width is 17-bits. Therefore, it is possible that a command buffer can be incorrectly addressed by the SLCDC if it extends beyond a 128 kilobyte boundary. Therefore, it is recommended that the command buffer base address be set at the beginning of a 128 kilobyte boundary. The sum of COMBASEADR[16:0] and COMBUFSIZ[16:0] must not exceed a multiple of 128K.

44.2.13 Command String Size Register (COMSTRINGSIZ)

Address 0x1002_2010 (COMSTRINGSIZ) Access: User read/write

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

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

Figure 44-16. Command String Size Register

Table 44-10. Command String Size Register Field Description

Field	Description
31–8	Reserved. These bits are reserved and read zero.
7–0 COMSTRINGSIZ	Command String Size. Length (in words) of the LCD command string made up of LCD controller page and column address commands. This command string is used when SLCDC begins to fill a new page of the LCD controller's RAM. **Note:** These strings are only transmitted to the LCD controller when SLCDC is configured for transfers with automatic page and column address command insertion. Command strings are stored along with tags in the LCD command buffer located in system memory. These bits represent the number of words that must be sent to the LCD controller to set the page and column address, not the number of words needed to store the corresponding commands and tags in system memory.

44.2.14 FIFO Configuration Register (FIFOCONFIG)

Address 0x1002_2014 (FIFOCONFIG) Access: User read/write

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

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

Figure 44-17. FIFO Configuration Register

Table 44-11. FIFO Configuration Register Description

Field	Description
31–3	Reserved. These bits are reserved and should read zero.
2–0 BURST	DMA Burst Length. Length (in 32-bit words) of the DMA burst. that is, these bits determine the number of 32-bit word reads that occur when SLCDC has ownership of R-AHB bus. 000 Burst length set to 1 32-bit word 001 Burst length set to 2 32-bit word ... 111 Burst length set to 8 32-bit words

44.2.15 LCD Controller Configuration Register (LCDCONFIG)

Address 0x1002_2018 (LCDCONFIG) Access: User read/write

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

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

Figure 44-18. LCD Controller Configuration Register

Table 44-12. LCD Controller Configuration Register Field Description

Field	Description
31–13	Reserved. These bits are reserved and should read zero.
12–0 WORDPPAGE	LCD Bytes Per Page. Number of words per page for external LCD controller connected to SLCDC module. These bits are used by SLCDC to determine when control characters must be sent to the controller. 0 = 0 words per LCD page, 1 = 1 word per LCD page … 8191 = 8191 words per LCD page

44.2.16 LCD Transfer Configuration Register (LCDTRANSCONFIG)

Address 0x1002_201C (LCDTRANSCONFIG) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	WORDDEFWRITE	WORDDEFDAT	WORDDEFCOM	XFRMODE	CSPOL	SKCPOL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-19. LCD Transfer Configuration Register

Table 44-13. LCD Transfer Configuration Register Field Description

Field	Description
31–18	Reserved. These bits are reserved and should read zero.
17–16 IMGEND	Image Endianness. This field defines the image endianness. 00 Big endian or 32-bit little endian 01 16-bit little endian 10 8-bit little endian 11 Reserved
15–6	Reserved. These bits are reserved and should read zero.
5 WORDDEFWRITE	Word Define, Write Data. It defines word for the LCD WRITE DATA register transfers. 0 8-bit data words 1 16-bit data words
4 WORDDEFDAT	Word Define, Data. It defines word for LCD data registers and data transfers. All registers associated with data transfers (DATA BUFFER SIZE, LCD CONFIG) must take this into consideration. 0 8-bit data words 1 16-bit data words
3 WORDDEFCOM	Word Define, Command. It defines word for LCD command registers and command data transfers. All registers associated with data transfers (COMMAND BUFFER SIZE) must take this into consideration. 0 8-bit command words 1 16-bit command words
2 XFRMODE	Image Data Transfer Width. This bit selects the width of LCD display interface data bus. Data transfers to the LCD controller can be done in a serial or parallel manner. 0 word serial transfers 1 word parallel transfers

Table 44-13. LCD Transfer Configuration Register Field Description (continued)

Field	Description
1 CSPOL	Chip Select Polarity. Selects LCD_CS signal polarity. It affects the LCD_CS pin behavior. 0 LCD_CS is asserted low. Therefore, SLCDC will drive LCD_CS to 0 during serial transfer of data to external LCD controller. In parallel mode, LCD_CS will normally be high. LCD data will change when LCD_CS goes low. LCD_CS rising edge is used by the LCD controller to latch the parallel data. 1 LCD_CS is asserted high. Therefore, SLCDC will drive LCD_CS to 1 during serial transfer of data to external LCD controller. In parallel mode, LCD_CS will normally be low. LCD data will change when LCD_CS goes high. LCD_CS falling edge is used by LCD controller to latch the parallel data.
0 SCKPOL	Serial Data Clock Polarity. It selects whether serial data transitions on rising/falling edge of serial data clock during transfers of data in serial mode. It has no effect during parallel data transfers. 0 Serial data will transition on the rising edge of serial clock. Therefore, data should be latched by the display device on the falling edge of serial clock. 1 Serial data will transition on the falling edge of serial clock. Therefore, data should be latched by the display device on the rising edge of serial clock.

44.2.17 SLCDC Control/Status Register (SLCDCCONTROL/STATUS)

0x1002_2020 (SLCDCCONTROL/STATUS) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	AUTOMODE		0	0	PROT1	IRQEN	IRQ	UNDRFLOW	TEA	0	BUSY	ABORT	GO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-20. SLCDC Control/Status Register

Table 44-14. SLCDC Control/Status Register Field Description

Field	Description
31–13	Reserved. These bits are reserved and should read zero.
12–11 AUTOMODE	Automatic Transfer Mode. These bits control how SLCDC transfers the data buffer to the LCD controller upon being triggered by the GO bit. DATA BASE ADDRESS bits designate the location of transferred data buffer in the system memory. Transfers triggered by GO bit can be configured to automatically insert page address and column address commands in the data stream to navigate through the LCD controller's RAM. This is done without CPU intervention. SLCDC can be also be configured to send the data buffer to the LCD controller without inserting any command strings. In this mode, the value of the LCD_RS signal can be configured to be either 1 or 0 during the transfer. 00 SLCDC transfers the data buffer defined by DATA BASE ADDRESS and DATA BUFFER SIZE without inserting control strings. With this setting, LCD_RS signal is tied to 0 during the entire transfer. 01 SLCDC transfers the data buffer defined by DATA BASE ADDRESS and DATA BUFFER SIZE without inserting control strings. With this setting, LCD_RS signal is tied to 1 during the entire transfer. 10 When transmitting data buffer defined by DATA BASE ADDRESS and DATA BUFFER SIZE, SLCDC inserts command strings to set the LCD controller's page and column addresses. These command strings are taken from the buffer defined by COMMAND BASE ADDRESS and COMMAND STRING SIZE. This insertion of command strings is done at beginning of each page of LCD controller RAM. 11 Reserved
10–9	Reserved. These bits are reserved and should read zero.
8 PROT1	R-AHB Protection Code. This bit controls the HPROT[1] signal value sent to R-AHB each time a SLCDC access is done. This value controls the access type done on the bus (user or privileged). SLCDC_hprot[0] output of SLCDC is tied to 1 to indicate a data access. PROT1 bit resets to 0. 0 SLCDC_hprot[1:0] driven to 01 indicating SLCDC accesses will be user data accesses. 1 SLCDC_hprot[1:0] driven to 11 indicating SLCDC accesses will be privileged data accesses.
7 IRQEN	Interrupt Enable. This bit controls whether a SLCDC interrupt is generated upon completion of SLCDC transfer or not. SLCDC transfers can end in three ways: 1) When a transfer error occurs on the R-AHB. 2) When SLCDC completes the data transfer to the LCD controller. 3) When a SLCDC transfer is aborted by setting ABORT bit in SLCDC control/status register. SLCDC transfers are initiated by assertion of GO bit or a write to LCD WRITE DATA register. Clearing IRQEN bit prevents an interrupt from being generated when SLCDC completes a transfer. 0 SLCDC interrupt is masked. No interrupt is generated from SLCDC. 1 SLCDC interrupt is generated upon completion of a SLCDC transfer.
6 IRQ	SLCDC Interrupt Flag. This bit indicates that an interrupt condition occurred in SLCDC. Interrupt flag is set when a SLCDC transfer is completed or aborted. SLCDC transfers are initiated by assertion of GO bit or a write to LCD WRITE DATA register. This bit is cleared by writing a 1 to it. 0 SLCDC interrupt is not pending 1 Interrupt condition occurred in SLCDC.

Table 44-14. SLCDC Control/Status Register Field Description (continued)

Field	Description
5 UNDRFLOW	SLCDC FIFO Underflow. This bit indicates that SLCDC FIFO became empty during data transfer from system memory to LCDC. Although FIFO underflow does not cause an error in SLCDC transfer, it indicates the output portion of SLCDC had to wait for FIFO to be filled during transfer. This bit is cleared by writing a 1 to it. **Note:** Underflow flag will not set until some data has been read into the SLCDC FIFOs. It only sets when FIFOs does not refill fast enough for the SLCDC bit manipulator. In this case, SLCDC bit manipulator stalls while it waits for data to be read from system memory into the FIFOs. It is also not set in case when a SLCDC transfer is initiated, but SLCDC is never granted control of bus, and thus able to fill the FIFOs. FIFOs must go from a non-empty state to an empty state to set it. 0 No underflow occurred in SLCDC FIFO. 1 SLCDC FIFO became empty during data transfer to the LCD controller. This causes the output portion of SLCDC to wait, delaying data transfer to the LCD controller.
4 TEA	SLCDC DMA Transfer Error. This bit indicates that SLCDC DMA received a transfer error acknowledge from R-AHB bus while trying to read data from system memory. This is indicated by HRESP0=1 and HREADY=1 on the R-AHB bus. This is a fatal condition and causes SLCDC to terminate its transfer. This bit is cleared by writing a 1 to it. 0 No transfer error acknowledge error occurred. 1 A transfer error acknowledge was received by the SLCDC DMA from the R-AHB.
3	Reserved. This bit is reserved and should read zero.
2 BUSY	SLCDC Busy. This bit indicates that SLCDC is in the process of transferring image buffer from system memory to LCD controller. Data transfer is initiated by setting the GO bit or by writing data to the LCD write data register. This bit is a read-only bit and clears after data transfer to the LCD controller is completed. 0 SLCDC idle. 1 SLCDC in process of transferring image buffer.
1 ABORT	Abort SLCDC Transfer. This bit causes SLCDC to abort the current data transfer. Termination occurs gracefully, that is, ongoing byte transfer to the LCD controller is completed before SLCDC goes to an idle state. This bit is cleared automatically when SLCDC has transitioned to the idle state. 0 Do not abort the current data transfer. 1 Abort the current SLCDC data transfer.
0 GO	Start SLCDC Transfer. This bits starts automatic transfer of image data from system memory to the external LCD controller. This bit always reads 1 until data transfer is complete. After data transfer is complete or has been aborted, GO bit will read 0. **Note:** Writing 1 to this bit has no effect if BUSY bit is set. 0 Do not start SLCDC transfer. 1 Start SLCDC image data transfer.

44.2.18 LCD Clock Configuration Register (LCDCLOCKCONFIG)

Address 0x1002_2024 (LCDCLOCKCONFIG) Access: User read/write

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

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

Figure 44-21. LCD Clock Configuration Register

Table 44-15. LCD Clock Configuration Register Field Description

Field	Description
31–6	Reserved. These bits are reserved and should read zero.
5–0 DIVIDE	LCD Clock Divide Value. This bit sets the divide ratio used to generate the LCD clock from HCLK_SLCDC clock. Setting DIVIDE to 0 disables the LCD_CLK by gating HCLK_SLCDC clock from the fractional divider. These bits must be set to some non-zero value before LCD transfers can occur. Frequency relations: 0 LCD_CLK off 1 LCD_CLK = HCLK_SLCDC clock*1/128 2 LCD_CLK = HCLK_SLCDC clock*2/128 … 63 LCD_CLK=HCLK_SLCDC clock*63/128

44.2.19 LCD Write Data Register (LCDWRITEDATA)

0x1002_2028 (LCD WRITE DATA) Access: User read/write

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

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

Figure 44-22. LCD Write Data Register

Table 44-16. LCD Write Data Register Field Description

Field	Description
31–17	Reserved. These bits are reserved and should read zero.
16 RS	LCD Register Select. Determines the value placed on LCD_RS pin during byte transfer from WRITE DATA register 0 LCD_RS signal is tied to 0 during transfer. 1 LCD_RS signal is set to 1 during transfer.
15–0 LCDDAT	LCD Controller Data. Data written is transferred to the external LCD controller. This register gives direct write access to the LCD controller. Data is determined to be command or display data via RS bit. Data written to this register is only transferred to the LCD controller if SLCDC is not in middle of another transfer (indicated by the BUSY bit of control/status register (Section 44.2.17, "SLCDC Control/Status Register (SLCDCCONTROL/STATUS)"). A read of these bits gives the last value written to this register (or the reset value), and not a value from the LCD controller. The amount of data that is transferred is determined by WORDDEFWRITE bit in the LCD transfer configuration register (Section 44.2.16, "LCD Transfer Configuration Register (LCDTRANSCONFIG)"). During 8-bit word transfers, LCDDAT[7:0] is transferred, and LCDDAT[15:8] is ignored.

44.3 LCD Controller Interface

SLCDC transfers data from the display memory buffer in system memory to the external display device. Transfers can be done via a 4-wire serial, 3-wire serial, 8-bit parallel, or a 16-bit parallel interface.

44.3.1 Serial Interface

In serial mode (XFRMODE bit in LCD Transfer Configuration Register is 0), data is transmitted to the display device via LCD_CS, LCD_DATA[7:6] and LCD_RS. Data is transmitted on LCD_DATA[7], serial clock toggles on LCD_DATA[6] and LCD_RS tells the display whether it is display data or command data. LCD_CS is used as a chip select signal. LCD_CS signal is asserted during all transfers. LCD_CS signal polarity is programmable using the CSPOL bit of LCD Transfer Configuration Register (Section 44.2.16, "LCD Transfer Configuration Register (LCDTRANSCONFIG)").

Figure 44-23 shows timing associated with the transfer of one byte of data to the display. The polarity of the serial data clock on LCD_DATA[6] pin can be configured using SCKPOL bit of LCD Transfer Configuration Register (Section 44.2.16, "LCD Transfer Configuration Register (LCDTRANSCONFIG)"). This bit must be set in accordance with the external display device requirements.

If external device latches the serial data on rising edge of serial clock, SCKPOL must be set to 1. If external device latches the serial data on falling edge of serial clock, SCKPOL must be cleared. The command and data word definitions determines how long the data string is in serial mode.

Figure 44-23 shows 8-bit transfers. A serial transfer of 16-bits has same timing as 8-bit transfers, however 16-bits of data is transferred. It is possible to have a command defined as 8-bits and serial defined as 16-bits (or vice versa). In this case, when a command is being transferred, 8-bits of data are sent, and when data is being transferred, 16-bits of data is sent.

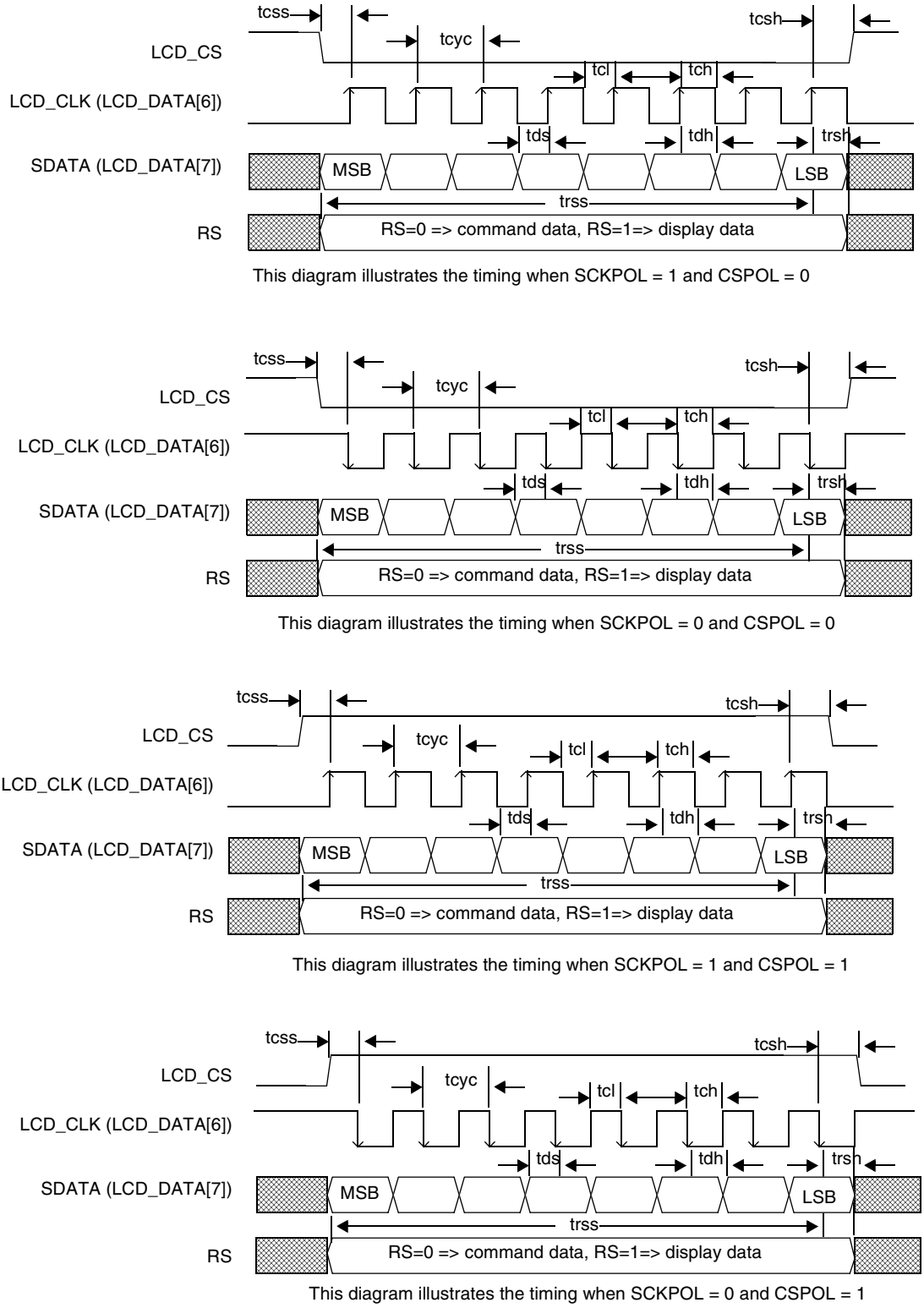


Figure 44-23. SLCDC Serial Transfers to LCD Device

Table 44-17. SLCDC Serial Interface Timing

Symbol	Parameter	Minimum (ns)	Typical (ns)	Maximum (ns)
t_{css}	Chip select setup time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{csh}	Chip select hold time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{cyc}	Serial clock cycle time	39 (±) t_{prop}	—	2641
t_{cl}	Serial clock low pulse	18 (±) t_{prop}	—	—
t_{ch}	Serial clock high pulse	18 (±) t_{prop}	—	—
t_{ds}	Data setup time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{dh}	Data hold time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{rss}	Register select setup time	(15 * t_{cyc} / 2) (±) t_{prop}	—	—
t_{rsh}	Register select hold time	(t_{cyc} / 2) (±) t_{prop}	—	—

t_{prop} is the propagation delay from SLCDC outputs to the pin outputs.

44.3.2 Parallel Interface

SLCDC module can be configured to execute parallel transfers of display data from the system memory to an external display device. Figure 44-24 shows the timing associated with an SLCDC parallel transfer to an external LCD device. The timing is based on the frequency of LCD_CLK, which is programmable via LCD Clock Configuration Register (Section 44.2.18, “LCD Clock Configuration Register (LCDCLOCKCONFIG)”).

In parallel mode, LCD_CS pin is used as a write strobe by the external LCD controller. LCD_CS latching edge occurs one LCD_CLK cycle after data is made available to the display on the LCD_DATA[15:0] pins. LCD_CS signal polarity is programmable using the CSPOL bit in the LCD transfer configuration register. Data transfers of 16- or 8-bit are determined by the word definition bits in the LCD transfer configuration register (Section 44.2.16, “LCD Transfer Configuration Register (LCDTRANSCONFIG)”). All 8-bit data transfer uses LCD_DATA[7:0]. LCD_DATA[15:8] shows 0’s. It is possible for there to be a 8-bit command data transfers and 16-bit display data transfers (and vice versa). In this case, when command data is transferred, only LCD_DATA[7:0] is used, and display data uses LCD_DATA[15:0].

Table 44-18. SLCDC Parallel Interface Timing

Symbol	Parameter	Minimum (ns)	Typical (ns)	Maximum (ns)
t_{cyc}	Parallel clock cycle time	78 (±) t_{prop}	—	4923
t_{ds}	Data setup time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{dh}	Data hold time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{rss}	Register select setup time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{rsh}	Register select hold time	(t_{cyc} / 2) (±) t_{prop}	—	—

t_{prop} is the propagation delay from the SLCDC outputs to the pin outputs.

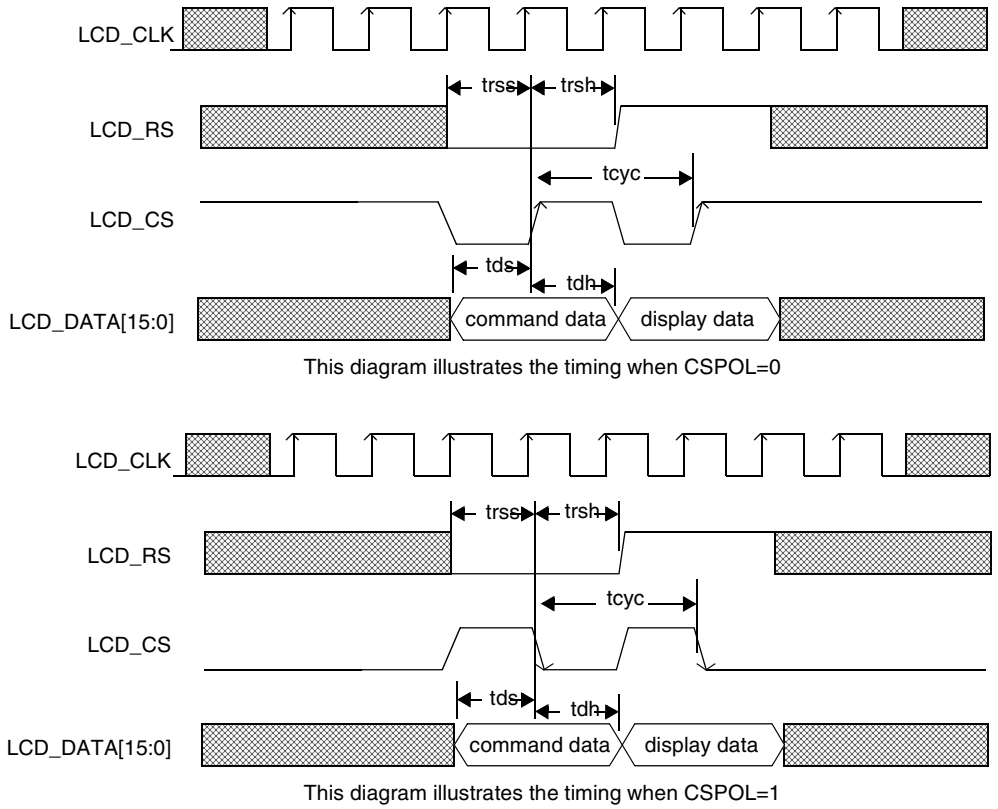


Figure 44-24. SLCDC Parallel Transfers to LCD Device

44.4 LCD Clock Configuration

SLCDC uses a fractional divider to generate the LCD data clock from HCLK_SLCDC clock signal. The fractional divider is programmed by setting a divide value using DIVIDE[5:0] bits of the LCD clock Configuration Register (Section 44.2.18, "LCD Clock Configuration Register (LCDCLOCKCONFIG)). LCD_CLK frequency is calculated using Equation 44-1:

Eqn. 44-1

$$LCD\_CLK = \frac{HCLKx(DivideValue)}{128}$$

Divide value is taken directly from DIVIDE[5:0] bits of the LCD clock Configuration Register (Section 44.2.18, "LCD Clock Configuration Register (LCDCLOCKCONFIG)"). Setting DIVIDE[5:0] to 0, disables the LCD_CLK signal. These bits must be set to some non-zero value before LCD transfers can occur.

Table 44-19. LCD_CLK Frequency Range

HCLK_SLCDC Clock Frequency (MHz)	Minimum LCD_CLK Frequency (MHz)	Maximum LCD_CLK Frequency (MHz)	Resolution of Step (kHz)
52	0	25.59	406.25
26	0	12.8	203.13
16.8	0	8.27	131.25

44.5 R-AHB Interface and SLCDC FIFOs

SLCDC uses a DMA interface to access system memory without CPU intervention. When SLCDC needs data, DMA requests AHB control and reads 32-bit words from the system memory. The number of 32-bit words read during the burst is determined by the value stored in the BURST[2:0] bits of the FIFO Configuration Register (Section 44.2.14, “FIFO Configuration Register (FIFOCONFIG)”). AHB control is relinquished by DMA after the burst is complete. SLCDC generates idle cycles when data transfer is complete. The amount of R-AHB bandwidth required by SLCDC can be controlled by the system software. The frequency of LCD frame updates and the rate of LCD_CLK affect how often the SLCDC requests the use of R-AHB.

Data read by the DMA interface is stored in two 32-bit × 8 word FIFOs that are used to buffer data between the DMA and the SLCDC display interface. One FIFO is used to buffer data from the command buffer and the other is used to buffer data from the display data buffer. When one FIFO has enough room to store the data read during a DMA burst, it will request servicing. Because DMA burst length is programmable, the occupancy level of the FIFOs can vary when servicing is requested.