Intel® PXA27x Processor Family

Developer's Manual

January 2006

Order Number: 280000-003

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTELR PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT.

Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The Intel® PXA27x Processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

I^2C is a two-wire communications bus/protocol developed by Philips. SMBus is a subset of the I^2C bus/protocol and was developed by Intel. Implementations of the I^2C bus/protocol or the SMBus bus/protocol may require licenses from various entities, including Philips Electronics N.V. and North American Philips Corporation.

This document and the software described in it are furnished under license and may only be used or copied in accordance with the terms of the license. The information in this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document. Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at http://www.intel.com.

Intel, the Intel logo, Intel StrataFlash, Intel XScale, and Wireless Intel SpeedStep are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © Intel Corporation, 2006. All Rights Reserved.

Contents:

Figures:

Tables:

Revision History

Date	Revision	Description
April 2004	-001	Initial release
October 2004	-002	Updated Section 2.2.5.1. with C0 CPUID and JTAG ID Updated Table 6-31 with the correct address for SA-1110 Updated Table 3-28 with new info for the CLKPWR register Updated Table 8-8 with SSPSP_X[MODE] description Updated Table 3-33 with CKEN[11] description Updated Table 24-2 with CKEN[11] description Updated Table 15-6 with the description of PRG_DONE Updated Section 15.8.9.2 step 4 of the MMC/SD/SDIO Block Data Read sequence Updated Table 3-17 with the PWER bit field definition. Updated Section 6.5.1.4 to add the CLK_MEM with SDCLK<2> or SDCL<1> change. Added note to section 3.5.7.2 for CLK_MEM with SDCL<1> or SDCLK<2> at 104 MHz Updated Section 21.4.4 Alarm bit data. Added section 8.4.11.1 and 8.4.11.2 Updated section 8.4.11 Added examples to section 15.7.2 and 15.7.3 Section 3.4.6.3.1 updated GPIO reset section from Edge triggered to Level triggered

Date	Revision	Description
January 2006	-003	Updates made based on “Dec 2005 Intel® PXA27x Processor Family Specification Update” LCD controller LCCR4 - Additional Functionality, Chapter 7, “LCCR4 Bit Definitions” Additional GPIOs for SSPTXD2 and SSPRXD2, Chapter 24, “GPIO Alternate Functions” Bit definition change in USB port 2 output 2 control register (UP2OCR), Chapter 12, “UP2OCR Bit Definitions” C5 additional supported product points, Chapter 3, “Clock Frequencies” Additional Trusted Platform Module (TPM) Documentation, Chapter 25, “Bit Positions for Primary Interrupt Sources”, Chapter 3, “Clock Enable Mappings for CKEN Bits”, Chapter 5, “DMA Quick Reference for On-Chip Peripherals”, Chapter 5, “DMA Controller Register Summary” Additional ForceSE0 mode in USB Host Ports, Chapter 12, “UP3OCR Bit Definitions” New CPU ID and JTAG ID values, Chapter 2, “Coprocessor: CPU ID and JTAG ID Values” LCD controller: LCCR3, Chapter 7, “LCD Controller Control Register 3 (LCCR3)” USB Client Controller: Removed section 12.6.3.8 “Host Port 2 D+ Pull-Up Bypass Enable“, Chapter 12, “Host Port 2 D– Pull-Up Enable” USB Client Controller: Removed bypass switch SW2 from Figure 12-22, Chapter 12, “D+ Pull-Up Resistors” Core: Incorrect register setting for supported product points, Chapter 3, “Clock Frequencies” GPIO: Wake-up sources updated, Chapter 2, “Intel® PXA27x Processor Signal Descriptions” Power Manager: Fault signals causes processor to enter Deep-Sleep, Chapter 2, “Intel® PXA27x Processor Signal Descriptions”

Date	Revision	Description
January 2006	-003	Processor ID register updated for the addition of the C5 stepping, Chapter 2, “Processor ID Register” SSPSCLKEN2 signal name correction, Chapter 8, “SSP Serial Port I/O Signal Descriptions” Interrupt Controller: ICPR bit definition correction, Chapter 25, “ICPR Bit Definitions” Operating System Timers: Resolution Clarification, Chapter 22, “Clock Generation for Channels 4–7”, Chapter 22, “Clock Generation for Channels 8–11” Memory Controller: SA1110 bit definitions correction, Chapter 6, “SA1110 Bit Definitions” RTC behavior after reset correction, Chapter 3, “Summary of Module Reset Functions” GPIO<38> alternate function 3 (out) definition correction, Chapter 24, “GPIO Alternate Functions” Memory Controller: Stacked Memory Clarification, Chapter 6, “Stacked SDRAM and Flash Memory”, Chapter 6, “MDCNFG Bit Definitions” Operating System Timers: Deep Sleep Mode Behavior Correction, Chapter 22, “OMCR4/5/6/7 Bit Definitions” Interrupt Controller: ICHP bit definition correction, Chapter 25, “ICHP Bit Definitions” Reset Manager: GPIO reset behavior updated, Chapter 3, “GPIO Reset” SYS_EN signal description correction, Chapter 2, “Intel® PXA27x Processor Signal Descriptions” Debug JTAG access timing correction, Chapter 26, “Code Download During a Cold Reset for Debug”, Chapter 26, “Code Download During a Warm Reset for Debug” USB host port wake-up behavior clarification, Chapter 20, “Typical List Structure” RTC register descriptions correction, Chapter 21, “RTC Stopwatch Alarm Registers (SWARx)” Power on reset timing correction, Chapter 3, “Invoking Power-On Reset” Clock manager and clock distribution block diagram updated, Chapter 3, “Clocks Manager and Clocks Distribution Block Diagram” Memory Controller Registers are not affected by GPIO Reset, Chapter 3, “Summary of Module Reset Functions”

Introduction 1

The Intel® PXA27x Processor Family (referred to as the *PXA27x processor* throughout this document) provides industry-leading multimedia performance, low-power capabilities, rich peripheral integration, and second-generation memory stacking. Designed from the ground up for wireless clients, it incorporates the latest Intel advances in mobile technology over its predecessor, the Intel® PXA255 processor. The Intel® PXA27x processor redefines scalability by operating from 104 MHz up to 624 MHz, providing enough performance for the most demanding mobile applications.

It is the first Intel® Personal Internet Client Architecture (PCA) processor to include Intel® Wireless MMX™ technology, enabling high performance, low-power multimedia acceleration with a general-purpose instruction set. Intel® Quick Capture technology provides one of the industry's most flexible and powerful camera interfaces for capturing digital images and video. While performance is key, power consumption is also a critical component. The new capabilities of Wireless Intel SpeedStep® technology provide a quantum leap forward in low-power operation.

The Intel® PXA27x Processor Family is available in both discrete and stacked versions, providing customers the flexibility to use the processor even when space is at a premium.

Members of the Intel® PXA27x Processor Family include:

- Intel® PXA270 processor in a 13x13mm VFBGA package
- Intel® PXA271 processor with 32 Mbytes of Intel StrataFlash® memory and 32 Mbytes of low-power SDRAM
- Intel® PXA272 processor with 64 Mbytes of Intel StrataFlash® memory

1.1 About This Manual

This manual is intended for experienced programmers of ARM* Architecture V5TE-compliant processors. This manual assumes that programmers have a working knowledge of the vocabulary and principles of embedded-systems programming.

Intel XScale® technology and the Intel® Wireless MMX™ media enhancement technology are not described in this manual. For more information, refer to Table 1-1.

1.1.1 Number Representation

All numbers in this document are decimal (base 10**)** unless designated otherwise. Hexadecimal numbers have a prefix of 0x, and binary numbers have a prefix of 0b. For example, 107 is represented as 0x6B in hexadecimal and 0b110_1011 in binary.

1.1.2 Naming Conventions

All signal and register bit names appear in uppercase. Active low items are prefixed with a lowercase “n”.

Bits within a signal name are enclosed in angle brackets:

EXTERNAL_ADDRESS<31:0>

nCS<1>

Bits within a register bit field are enclosed in square brackets:

REGISTER_BITFIELD[3:0]

REGISTER_BIT[0]

Bit fields in registers are identified this way: REGISTER[REGISTER_BITFIELD]

The following terms are used in this document:

clear—Program 0b0 into a single register bit.

set—Program 0b1 into a single register bit.

write—Program a hexadecimal value into a register bit field (more than one bit).

assert—Drive a signal to its active voltage level, either high or low.

de-assert—Drive a signal to its inactive voltage level, either high or low.

drive—Assert a voltage level onto a signal.

In register-definition tables:

Values shown in the “Reset” row have the following meanings:

0 = Bit clear

1 = Bit set

? = Bit is undefined.

***** = Bits whose value is determined by the state of an external pin

Abbreviations in the “Access” column have the following meanings:

R = Read-only

W = Write-only

R/W = Read and write

There are two special cases:

R/WC = R/W. To clear the bit, write 0b1 to it.

RC = Read-only. The bit is automatically cleared after it is read.

1.1.3 Data Types

In the context of the ARM* Architecture V5TE, a word consists of 32 bits. As a result, the following naming convention applies to the different data types in the PXA27x processor:

- 8 bits = byte (abbreviation **B**)
- 16 bits = half word (abbreviation **H**)
- 32 bits = word (abbreviation **W**)
- 64 bits = double word (abbreviation **D**)

1.1.4 Related Documents

Table 1-1 lists supplemental documentation for users of the Intel® PXA27x Processor Family. Contact an Intel representative for the latest revision of Intel documents without order numbers.

Table 1-1. Supplemental Documentation (Sheet 1 of 2)

Title
ARM Architecture Version 5T Specification (Document number ARM DDI 0100D-10) and *ARM Architecture Reference Manual* (ARM DDI 0100 or ISBN 0-201-73719-1)
ARM Developer Suite Developer Guide
ARM *Multi-ICE System Design Considerations, Application Note 72 (*ARM DAI 0072A)
Audio Codec '97 Component Specification, http://www.intel.com/labs/media/audio
Bluetooth SIG Inc. http://www.bluetooth.org
CF+ and CompactFlash Specification, Version 1.4, CompactFlash Association, http://www.compactflash.org
General information: http://developer.intel.com
GSM 11.11 *Specification of the Subscriber Identity Module-Mobile Equipment (SIM-ME) Interface, Version 3.16.0*, http://www.etsi.org. See also ISO standard 7816-3
I^2C-*Bus Specification*, Philips Semiconductors, http://www.phillipssemiconductors.com
I^2S *Bus Specification*, February 1986, Philips Semiconductors, http://www.phillipssemiconductors.com
IEEE Std. 1149.1-1990 Standard Test Access Port and Boundary-Scan Architecture, http://standards.ieee.org
Infrared Data Association Serial Infrared Physical Layer Specification Version 1.3, October 15, 1998, http://www.irda.org
256-Mbit 1.8 Volt Intel StrataFlash® Wireless Memory (L18/L30) Stacked-Chip Scale Package (x16) Datasheet, 252633
Intel XScale® Core Developer's Manual, 273473
Intel XScale® Microarchitecture for the PXA255 Processor Developer's Manual, 278796
Intel® *PC SDRAM Specification, Version 1.7*
Intel® PXA270 Processor Electrical, Mechanical, and Thermal Specification, 280002, and Intel® PXA27x Processor Family Electrical, Mechanical, and Thermal Specification, 280003
Intel® PXA27x Processor Family Design Guide, 280001
Intel® PXA27x Processor Family Optimization Guide, 280004
Intel® PXA27x Processor Family Power Requirements Application Note, 280005
Intel® Wireless MMX™ Technology Developer's Guide, 251793
International Telecommunication Union, *Recommendation ITU-R BT.656-4,* http://www.itu.int
ISO 7816-3 Smart Card Standard: Part 3: Electronic Signals and Transmission Protocols; *3G TS 31.101* Technical Specification, 3rd Generation Partnership Project, http://www.3gpp.org
MICROWIRE Serial Interface, National Semiconductor Application Note AN-452
MultiMediaCard System Specification Version 3.2, http://www.mmca.org
OpenHCI—Open Host Controller Interface Specification for USB, Release 1.0a, http://www.usb.org
PC Card Standard, Volume 2, Electrical Specification, PCMCIA/JEITA, http://www.pcmcia.org
SD Memory Card Specifications Part I, Physical Layer Specification, Version 1.01, *and Secure Digital Input/ Output (SDIO) Card Specification*, Version 1.0 (Draft 4), SD Association, http://www.sdcard.org

Table 1-1. Supplemental Documentation (Sheet 2 of 2)

Title
Sony Memory Stick Standard, Format Specification Version 1.3
UARTs are functionally compatible with the 16550A and 16750 industry standards. The 16550A was originally produced by National Semiconductor Inc. The 16750 is produced as the TL16C750 by Texas Instruments.
Universal Serial Bus Specification, Revision 1.1; *On-The-Go Supplement to Universal Serial Bus Specification Revision 2.0; Pull-Up/Pull-Down Resistors Engineering Change Notice to Universal Serial Bus 2.0 Specification;* http://www.usb.org

1.2 Product Overview

The PXA27x processor is an integrated system-on-a-chip microprocessor for high-performance, low-power, portable, handheld and handset devices. It incorporates the Intel XScale® technology with on-the-fly voltage and frequency scaling and sophisticated power management to provide industry-leading MIPs/mW performance. The PXA27x processor complies with the ARM* Architecture V5TE instruction set (excluding floating point instructions) and follows the ARM* programmer's model. The PXA27x processor also supports Intel® Wireless MMX™ integer instructions in applications such as those that accelerate audio and video processing.

The PXA27x processor provides a scalable, bidirectional data interface to a cellular baseband processor supporting seven logical channels. The OS timer channels, mobile scalable link (MSL) interface, and synchronous serial ports (SSPs) also accept an external network clock input so that they can be synchronized to the cellular network. The PXA27x processor also provides a Universal Subscriber Identity Module (USIM) card interface.

The PXA27x processor memory interface supports a variety of external memory types to allow design flexibility. Support for the connection of two companion chips permits a glueless interface to external devices.

The PXA27x processor also provides four 64-Kbyte banks of on-chip SRAM, which can be used for program code or multimedia data. Each bank can be configured to retain its contents when the processor enters a low-power mode.

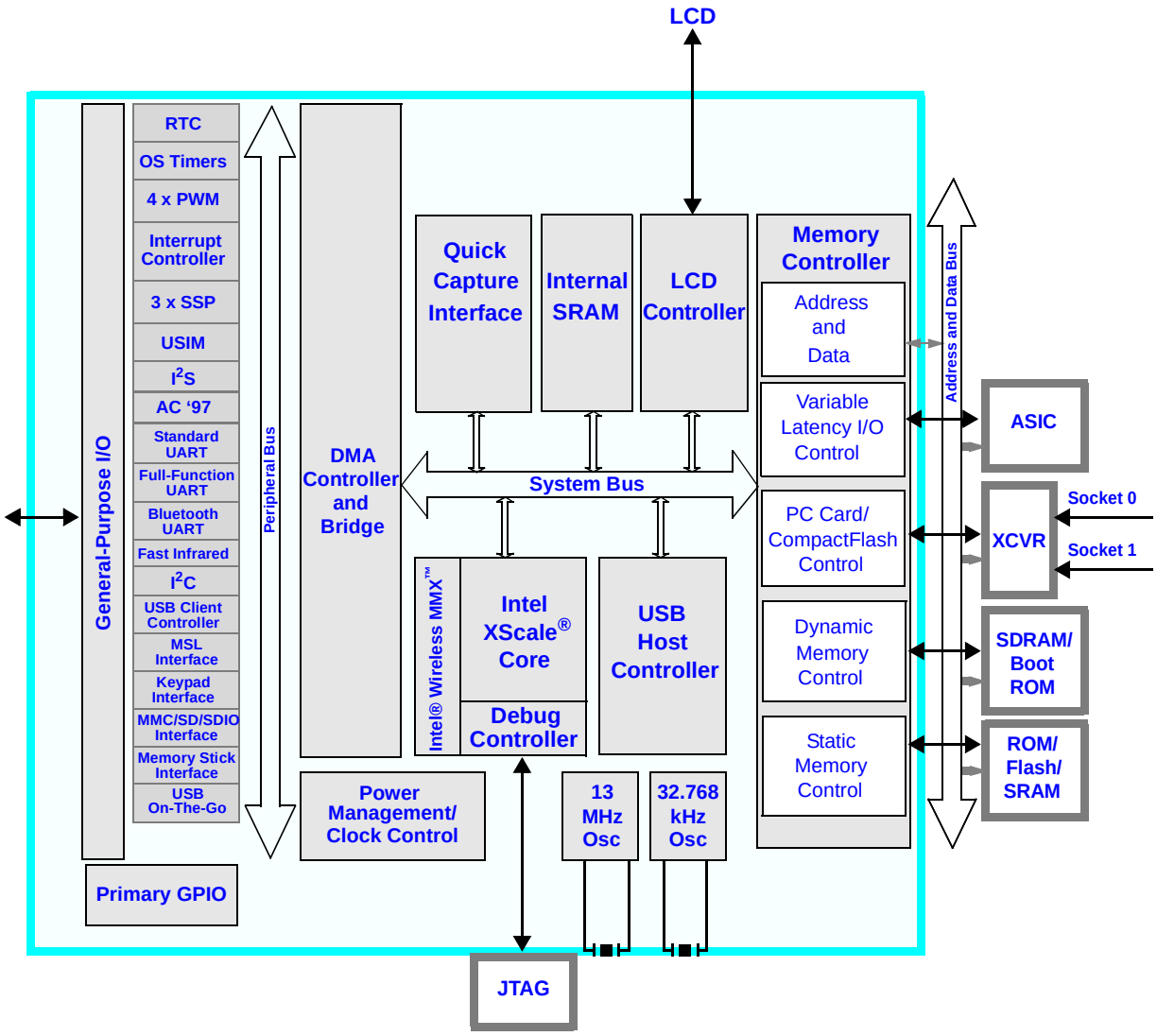
An integrated LCD panel controller provides support for displays up to 800 x 600 pixels. It permits 1-, 2-, and 4-bit gray scale and 8- or 16-bit color pixels. A 256-entry palette RAM provides flexibility in color mapping.

A set of serial devices and general system resources provides computational and connectivity capability for a variety of applications.

The PXA27x processor is designed for a high degree of backward compatibility with the Intel® PXA25x Applications Processor.

The PXA27x processor incorporates a comprehensive set of system and peripheral functions that make it useful in a variety of low-power applications. Figure 1-1 illustrates the system-on-a-chip processor. The diagram shows a primary system bus with the Intel XScale® core attached, along with an LCD controller, USB host controller, and 256 Kbytes of internal memory. The system bus is connected to a memory controller to allow communication with a variety of external memory or companion-chip devices, and it is also connected to a DMA controller/bridge to allow communication with the on-chip peripherals. The key features of all the sub-blocks are described in this section, with more detail provided in the respective chapters.

Figure 1-1. Intel® PXA27x Processor Block Diagram for a Typical System



1.2.1 Intel XScale® Technology

The Intel XScale® microarchitecture is based on a new core that complies with the ARM* Architecture V5TE. The microarchitecture surrounds the core with instruction and data memory management units; instruction, data, and mini-data caches; write, fill, pend, and branch-target buffers; power management, performance monitoring, debug, and JTAG units; coprocessor interface; multiply-accumulate coprocessor (MAC); and core memory bus.

1.2.1.1 Intel XScale® Microarchitecture Features

The features of Intel XScale® microarchitecture include:

- Seven- to eight-stage superpipelined RISC technology achieves high speed and ultra low power.
- Dynamic voltage management means voltage and frequency on-the-fly scaling to allow applications to use the right blend of performance and power.
- Media processing technology lets the MAC perform two simultaneous 16-bit single-instruction multiple-data (SIMD) multiplies with 40-bit accumulation for efficient media processing.
- Power management provides power savings with idle, deep-idle, standby, sleep, and deep-sleep modes.
- 128-entry branch target buffer keeps the pipeline filled with statistically correct branch choices.
- 32-Kbyte instruction cache keeps local copies of important instructions to enable high performance and low power.
- 32-Kbyte data cache keeps local copy of important data to enable high performance and low power.
- 2-Kbyte mini-data cache avoids “thrashing” of the 32-Kbyte data cache for frequently changing data streams.
- 32-entry instruction-memory management enables logical-to-physical address translation, access permissions, and instruction-cache attributes.
- 32-entry data memory management unit enables logical-to-physical address translation, access permissions, and data-cache attributes.
- Four-entry fill and pend buffers promote core efficiency by allowing “hit-under-miss” operation with data caches.
- Performance monitoring unit furnishes two 32-bit event counters and one 32-bit cycle counter for analysis of hit rates.
- Debug unit uses hardware breakpoints and 256-entry trace-history buffer (for flow change messages) to debug programs.
- 32-bit coprocessor interface provides a high-performance interface between core and coprocessors.
- Eight-entry write buffer allows the core to continue execution while data is written to memory.

See the *Intel XScale®* Core Developer’s Manual for additional information.

1.2.1.2 Coprocessor

The Intel XScale® core has attached to it a coprocessor to accelerate multimedia applications. This coprocessor is characterized by a 64-bit single-instruction multiple-data (SIMD) architecture and compatibility with the integer functionality of the Intel® Wireless MMX™ technology and streaming SIMD extensions (SSE) instruction sets. Key features of this coprocessor include:

- 30 new media-processing instructions
- 64-bit architecture up to eight-way SIMD

- 16 x 64-bit register file
- SIMD PSR flags with group-conditional execution support
- SIMD instruction support for sum of absolute differences (SAD) and multiply-accumulate (MAC) operations
- Instruction support for alignment and video operations
- Intel® MMX™ and SSE integer instruction compatibility
- Superset of existing media-processing instructions in the Intel XScale® core

See the Intel® Wireless MMX™ documentation, listed in Table 1-1, for more details.

1.2.2 Power Management

The PXA27x processor provides a rich set of flexible power-management controls for a wide range of usage models while enabling very low-power operation. The key features include the following:

- Five reset sources: power-on, hardware, watchdog, general-purpose I/O (GPIO), and exit from sleep, and deep-sleep modes (sleep-exit)
- Three clock speed controls to adjust frequency: turbo mode, half-turbo mode, fast-bus mode
- Switchable clock source
- Functional clock gating
- Programmable frequency-change capability, with multiple turbo settings without requiring the PLL to re-lock
- Power modes to control power consumption: normal (both run and turbo), idle, deep-idle, standby, sleep, and deep-sleep
- Programmable I^2C-based interface to support external power-regulator devices

See Chapter 3, “Clocks and Power Manager” for more details.

1.2.3 Internal Memory

Key features of the internal memory module include:

- 256 Kbytes of on-chip RAM arranged as four banks of 64 Kbytes
- Bank-by-bank power management for reduced power consumption
- Support for byte writes

See Chapter 4, “Internal Memory” for more details.

1.2.4 Interrupt Controller

The interrupt controller, which masks and prioritizes all on-chip interrupts, is accessed either through memory-mapped or coprocessor registers. The key features are as follows:

- Peripheral interrupt sources can be mapped to normal (IRQ) or fast (FIQ) interrupt request
- Each interrupt source can be enabled independently

- Priority mechanism to indicate highest priority interrupt
- Accessible from the coprocessor interface for fast access
- Accessible as a memory-mapped peripheral for backward compatibility.

See Chapter 25, “Interrupt Controller” for more details.

1.2.5 Operating-System Timers

The operating-system timers provide the following features:

- Single-counter operating at 3.25 MHz
- Four Match registers
- Watchdog function

Eight additional timer channels offer the following additional features:

- Eight independent channels, each consisting of:
  - Counter
  - Match register
  - Control register
- Independent clock for each counter, selectable by software:
  - 32.768-kHz clock for low power
  - 13-MHz clock for high accuracy
  - Externally supplied clock for network synchronization
- Counter resolutions of 1/32768th of a second, one millisecond, and one microsecond
- Periodic and one-shot timers
- Two external synchronization events
- Operation during reduced-power modes (standby, sleep, deep-sleep)

See Chapter 22, “Operating System Timers” for more details.

1.2.6 Pulse-Width Modulation Unit (PWM)

The PWM unit consists of four independent channels. Source data can be derived from memory (using DMA) or from CPU store. The following are key features:

- Four pulse-width modulated output channels
- Enhanced period control through 6-bit clock divider and 10-bit period counter
- 10-bit pulse control

See Chapter 23, “Pulse Width Modulator Controller” for more details.

1.2.7 Real-Time Clock (RTC)

The real-time clock is a 32-bit counter with trim control that runs off the 32.76- kHz crystal oscillator. The general features of the RTC are as follows.

- Timer
  - — User-programmable free-running counter
  - — User-programmable alarm register
  - — Resolution of one second
- Wristwatch
  - — User-programmable free-running counter displaying: time of the day in terms of hours, minutes, and seconds, day of week, week of month, day of month, month, and year
  - — User-programmable Alarm registers generate alarms in terms of hours, minutes, seconds, day of week, week of month, day of month, month, and year
  - — Resolution of one second
- Stopwatch
  - — User-programmable Counter register displays the time elapsed between two events in terms of hours, minutes, seconds, and one-hundredth of a second
  - — Two user-programmable Alarm registers generate alarms in terms of hours, minutes, seconds, and one-hundredth of a second
  - — Resolution of one-hundredth of a second
- Periodic interrupts
  - — User-programmable Alarm register generates periodic interrupts at regular intervals
  - — Resolution of one millisecond
- Trimmers
  - — User-programmable Trimmer register generates a precise 1-Hz clock for the timer section and the wristwatch section

See Chapter 21, “Real-Time Clock (RTC)” for more details.

1.2.8 General-Purpose I/O (GPIO)

Most of the peripheral pins on the PXA27x processor also double as GPIO pins. The general features of the GPIO are as follows:

- As inputs, they can be sampled or programmed to generate an interrupt from either a rising or falling edge.
- As outputs, they can be cleared or set individually and can be preprogrammed to either state when entering sleep mode.
- Each can be programmed to alternate functions to provide system flexibility.

See Section 24, “General-Purpose I/O Controller” for more details.

1.2.9 Memory Controller

The external-memory controller is based on a unified memory architecture (UMA), where all memory devices share a common address and data bus. The memory controller consists of four main units, each with its own dedicated control pins: dynamic memory, static memory, card interface, and companion chip. The UMA consists of the following:

- Interfaces to internal synchronous flash and SDRAM devices
- Interfaces to four partitions of SDRAM
- Interfaces to up to 1.0 Gbytes of SDRAM
- Supports 1.8-V JEDEC LP-SDRAM operation at 104 MHz
- Interfaces to six partitions of static memory. Four of these six partitions can be synchronous static memory (synchronous flash)
- Interfaces to up to 384 Mbytes of flash memory
- Interfaces to two sockets of PC Card memory
- Allows an alternate bus master to take control of the bus
- Places the SDRAMs into self-refresh mode before entering sleep, standby, deep-sleep, and frequency-change modes
- Provides signals and controls for fly-by DMA transfers
- Supports non-volatile memory configured as bank 0 from either 16- or 32-bit devices
- Provides three independent output clocks that can be turned on/off separately and can be programmed to be free-running. The clocks can be the same frequency or half the frequency of the input clock, CLK_MEM. One clock can also be programmed as one quarter of the input-clock frequency.
- Programmable power-down mode for saving power
- Compatibility with the SA-1110 companion chips.

See Chapter 6, “Memory Controller” for more details.

1.2.10 DMA Controller

The PXA27x processor provides the following DMA features:

- Supports memory-to-memory, peripheral-to-memory, and memory-to-peripheral transfers (the latter two transfers are supported only in flow-through mode)
- Supports fly-by and flow-through modes for transfers related to external companion chips. The external device can be a peripheral or a companion chip.
- Operates as a bridge for programmed I/O accesses to various peripheral devices
- Supports 32 channels and 63 peripheral-device requests, with the capability of pre-programming any request to any channel
- Uses a priority mechanism to process active channels (four channels with outstanding DMA requests at any given time)
- Operates in either the descriptor-fetch or the no-descriptor-fetch mode in each of the 32 channels.

- Supports special descriptor modes (descriptor comparison and descriptor branching)
- Retrieves trailing bytes in the receive peripheral-device buffers
- Supports programmable data-burst sizes (8, 16, or 32 bytes) and programmable peripheral device data widths (byte, half-word, or word)
- Supports up to (8 Kbytes minus 1 byte) of data transfer per descriptor; larger transfers can be performed by software chaining multiple descriptors
- Supports flow-control bits to process peripheral-device requests. Requests are processed only if the flow-control bit is set.

See Chapter 5, "DMA Controller" for more details.

1.2.11 Serial Ports

The PXA27x processor supports a rich set of serial controllers for general system use. All ports can be accessed through programmed I/O or through descriptor-based DMA transfers. Pins on ports not being used can be converted to GPIOs. The following sections describe these ports.

1.2.11.1 UARTS

The PXA27x processor has three UARTs: standard, Bluetooth*, and full-function.

All UARTs have the following features:

- Slow infrared asynchronous interface (up to 115.2 kbps) based on the IrDA standard
- Registers are compatible with the 16550 and 16750
- Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from the serial data
- Independently-controlled transmit, receive, line-status, and data-set interrupts
- Baud-rate generator allows division of clock by 1 to (2^{16}–1) and generates an internal 16X clock; baud rate can be programmed manually or automatically using automatic baud-rate detection circuitry.
- Fully programmable serial interface characteristics:
  - 5-, 6-, 7-, or 8-bit characters
  - Even, odd, or no parity detection
  - 1, 1½, or 2 stop-bit generation
  - Baud-rate generation up to 921.6 kbps
- False start-bit detection
- Complete status-reporting capability
- Break generation and detection
- Internal diagnostic capabilities include:
  - Loop-back controls for communications-link fault isolation
  - Break, parity, overrun, and framing error simulation

The Bluetooth and full-function UARTs have the following extra features:

- Modem control functions (nCTS, nRTS)
- Auto-flow capability controls data I/O without generating interrupts:
  - nRTS (output) controlled by UART receiver FIFO
  - nCTS (input) from modem controls UART transmitter
  - Full-function UART has hardware modem control functions (nDSR, nDTR, nRI, and nDCD)

See Chapter 10, "UARTs" for more details.

1.2.11.2 Fast-Infrared Communications Port

The fast-infrared communications port has the following features:

- Infrared Data Association (IrDA) compliant—See the *Infrared Data Association Serial Infrared Physical Layer Specification Version 1.3*
- 4 Mbps IrDA, 4 ppm modulation
- Two separate 128-byte receive and transmit FIFOs
- FIFOs can be serviced using DMA, processor interrupt, or polling
- Transmit/receive loop-back mode for internal diagnostics
- Separate transmit/receive data path supports full or half-duplex operation

See Chapter 11, "Fast Infrared Communications Port" for more details.

1.2.11.3 I^2C Serial Bus Port

The I^2C interface has the following features:

- I^2C compliant
- Multi-master and arbitration support
- Supports standard-mode operation at 100 kbps
- Supports fast-mode operation at 400 kbps.

See Chapter 9, "I^2C Bus Interface Unit" for more details.

1.2.11.4 AC '97 Codec Interface

The AC '97 Codec interface supports the following key features:

- Independent channels for stereo pulse code modulation (PCM) in, stereo PCM out, modem out, modem in, and mono MIC in
- All of the above channels support 16-bit samples only
- Supports multiple-sample-rate AC '97 2.0 Codecs (48-kHz and below). The AC '97 controller depends on the Codec to control the varying rate.
- Supports read/write access to AC '97 registers
- Secondary Codec support.

See Chapter 13, “AC ‘97 Controller” for more details.

1.2.11.5 I²S Audio Codec Interface

The I²S audio Codec interface supports the following key features:

- Record and playback of 64-bit stereo audio samples
- Each sample has two channels: audio left and audio right, each 32 bits wide.
- Each channel has 16 MSBs of valid data and 16 LSBs of padded zeros.
- Supports MSB-justified and normal-I²S modes
- Supports sampling frequencies of 48 kHz, 44.1 kHz, 22.05 kHz, 16 kHz, 11.025 kHz, and 8 kHz
- The bit-rate clock (BITCLK) can be configured to be either an input or an output. If configured as output, the processor also supplies an I²S system clock (SYSCLK), which is four times the BITCLK.

See Chapter 14, “Inter-IC Sound (I²S) Controller” for more details.

1.2.11.6 USB Client Controller

The USB client controller has the following key features:

- USB Revision 1.1 compliant—12 Mbps, half duplex
- 23 programmable endpoints
  - — Programmable endpoint type: bulk, isochronous, or interrupt
  - — Programmable endpoint direction: IN or OUT
  - — Programmable endpoint maximum packet size
  - — Programmable configuration, interface, and alternate interface setting numbers
- Endpoint 0 for control IN and OUT
- Four configurations:
  - — Three programmable configurations with up to seven interfaces
  - — Default configuration 0 with one interface and control endpoint 0
- Configurable 4-Kbyte memory for endpoint data storage

See Chapter 12, “USB Client Controller” for more details.

1.2.11.7 USB Host Controller

The USB host controller has the following key features:

- USB Rev. 1.1 compatible
- Supports both low-speed and full-speed USB devices
- Open Host Controller Interface (OHCI) Rev 1.0a compatible
- Root hub supports three downstream ports.

- Built-in DMA

See Chapter 20, “USB Host Controller” for more details.

1.2.11.8 Synchronous Serial Ports (SSP)

The three SSP ports support these protocols:

  - — Programmable serial protocol (PSP) with programmable frame sync and programmable start and stop delays
  - — National Semiconductor Microwire
  - — Texas Instruments Synchronous Serial Protocol (SSP)
  - — Motorola Serial Peripheral Interface (SPI) protocol.
- Up to 13-Mbps transfer rate
- Sample data formats from 4 to 32-bits of serial data
- Master or slave operation for both clock and frame sync signals
- Flexible clock-source selection from the 13-MHz master clock, the network clock input, or the dedicated SSP external clock input

See Chapter 8, “SSP Serial Ports” for more details.

1.2.12 LCD Panel Controller

The LCD controller supports these key features:

- Display modes
  - — Support for single- or dual-scan display modules
  - — Passive monochrome mode supports up to 256 gray-scale levels (8 bits)
  - — Active color mode supports up to 16777216 colors (24 bits)
  - — Passive color mode supports a total of 16777216 colors (24 bits)
  - — Support for LCD panels with an internal frame buffer
  - — Support for 8-bit (each) passive dual-scan color displays
  - — Support for up to 18-bit per pixel single-scan color displays without an internal frame buffer
  - — Support for up to 24-bit per pixel single-scan color displays with an internal frame buffer
- Base plane with software control of two overlay windows and a hardware cursor
- Color management:
  - — Up-scaling for YCbCr 4:2:0 and 4:2:2 to YCbCr 4:4:4
  - — Color space conversion CCIR 601—YCbCr 4:4:4 to RGB 8:8:8
  - — Conversion from true color, (RGB 8:8:8) to high color (RGB 5:5:5) and the various configurations of RGBT
- Support for display sizes from 1x1 to 800 x 600 pixels.

- 64-entry (by 24 bits) output FIFO
- Three 256-entry by 25-bits internal color-palette RAMs (one for each overlay and base) programmable to be automatically loaded at the beginning of each frame
- Command data RAM (16 x 9 bits) to hold command data
- Supports pixel depths of 2, 4, 8, 16, 18, and 24 bits per pixel (bpp) in RGB format
- Overlays supported with pixel depths of 16, 19, 24, and 25 bpp in RGBT format
- Provides one base layer plus two overlays for single-scan displays; maximum size of each overlay can equal the display size
- Integrated seven-channel DMA (one channel for base plane, one channel for Overlay 1 and three channels for Overlay 2, one channel for the hardware cursor, and one channel to for the command data)
- Hardware support for color-space conversion from YCbCr to RGB for video streams
- Supports hardware cursor for single-scan display
- Programmable toggle of AC bias-pin output (toggled by line count)
- Programmable pixel clock from 52.0 MHz to 25.4 kHz (104.0 MHz/2 to 13 MHz/512)
- Supports little-endian ordering of pixels in frame buffer
- Programmable wait-state insertion at beginning and end of each line
- Programmable polarity for output enable, frame clock, and line clock
- Programmable interrupts for input and output FIFOs (underrun)
- Six 16 x 64-bit input FIFOs: one for the base channel, one for Overlay 1, three for Overlay 2, and one for the hardware cursor; plus a seventh 4 x 52-bit input FIFO for command data for panels with internal frame buffer
- Backward-compatible with the Intel® PXA25x and Intel® PXA26x processor LCD controllers

See Chapter 7, “LCD Controller” for more details.

1.2.13 MultiMediaCard, SD Memory Card, and SDIO Card Controller

The MultiMediaCard/SD/SDIO controller provides the following key features:

- Data-transfer rates up to 19.5 Mbps for MMC, 1-bit SD/SDIO, and SPI mode data transfers
- Data-transfer rates up to 78 Mbps for 4-bit SD/SDIO data transfers
- A response FIFO
- Two transmit FIFOs and two receive FIFOs
- Two modes of operation: MMC/SD/SDIO mode and SPI mode. MMC/SD/SDIO mode supports MMC, SD, and SDIO communications protocols. SPI mode supports the SPI communications protocol.
- 1- and 4-bit data transfers are supported for SD and SDIO communications protocols.
- Controller turns clock on and off, based on status of FIFOs, to prevent overflows and underruns.

- Support for all valid MMC and SD/SDIO protocol data-transfer modes
- Interrupt-based application interface to control software interaction
- For stream writes, only data sizes of 10 bytes or more are allowed.
- Using the MMC communications protocol, multiple MMC cards are supported.
- Using the SD or SDIO communications protocol, one SD or SDIO card is supported.
- Using the SPI communications protocol, up to two MMC or SD/SDIO cards are supported. Mixed card types are supported for the SPI communications protocol only.

See Chapter 15, "MultiMediaCard/SD/SDIO Controller" for more details.

1.2.14 Memory Stick Host Controller

The Memory Stick host controller supports the following features:

- Compliance with the Sony Memory Stick standard
- Built-in transmit and receive FIFO buffers
- Built-in CRC calculation and checking
- Transfer clock up to 20 MHz
- Data transfer using programmed I/O, interrupt to processor, and DMA
- Automatic command execution when an interrupt from the Memory Stick is detected

See Chapter 17, "Memory Stick Host Controller" for more details.

1.2.15 Mobile Scalable Link (MSL) Interface

The MSL interface has the following key features:

- Two independent, high-speed, unidirectional links
- Scalable links with data-channel width options
- Asynchronous clocking from 0 to over 48 MHz per link
- Transfer rate per link up to 192 Mbps at 48 MHz
- Low-power electrical interface: 1.8 V (+20%/-5%), 2.5 V, 3.0 V and 3.3 V +10%/-10%
- Power management protocol and features
- 14 independent logical data channels for managing multiple simultaneous data streams
- Large 64-byte FIFOs for all data channels
- Round-robin FIFO service with independent enables and configuration options
- Single- or multiple-burst transfers
- Support for DMA-, interrupt-, or poll-driven operation

See Chapter 16, "Mobile Scalable Link (MSL) Interface" for more details.

1.2.16 Keypad Interface

The keypad interface has the following key features:

- Direct-keypad interface:
  - — Eight inputs
  - — Supports up to eight direct keys and up to two rotary encoders
    - Eight direct keys, or
    - Six direct keys and one rotary encoder (two pins for the rotary encoder), or
    - Four direct keys and two rotary encoders (two pins each for the two rotary encoders)
- Matrix-keypad interface:
  - — Eight scan outputs and eight inputs (returns)
  - — Supports up to 64 keys
  - — Supports manual and automatic scan
- Simultaneous operation of direct and matrix keypads
- Interrupt generated on keypad activity:
  - — Separate matrix- and direct-key interrupt enables
  - — One interrupt signal, generated by merging the matrix and direct interrupts.
- Continuous keypad polling
- Key-debounce logic for both matrix and direct keypads

See Chapter 18, “Keypad Interface” for more details.

1.2.17 Universal Subscriber Identity Module (USIM) Interface

The USIM interface has the following key features:

- Compatible with any USIM card that is compliant with standard *ISO 7816-3* and *3G TS 31.101* and operates in voltages of 1.8 V or 3 V
- Supports control lines for two-level voltage supply (1.8 V and 3 V)
- Supports USIM card reset-pin control (using reset-pin control and power-supply control, warm/cold reset can be software-initiated)
- Supports T = 0 and T = 1 protocols
- Programmable card clock frequency
- Supports any combination of the following clock-rate conversion factor *F*, and bit-rate adjustment factor *D*:
  - — *F* = {372, 512, 558}
  - — *D* = {1,2,4,8,16, 32, 12, 20}
- Auto-error signal in T = 0 receive mode
- Auto-character repeat in T = 0 transmit mode
- Transforms inverted format to regular format and vice-versa

- Programmable block-guard time period
- Programmable extra-guard time period
- Programmable character-waiting time period
- Programmable block-waiting time period
- Programmable time-out period
- Programmable CPU interrupt on an error-signal detection
- Programmable CPU interrupt when a smart card is connected

See Chapter 19, "Universal Subscriber ID Interface" for more details.

1.2.18 Quick Capture Camera Interface

The quick capture interface is a component of Intel® Quick Capture technology and has the following features:

- Parallel interface support for 8, 9, and 10 bits
- Serial interface support for 4-bit and 5-bit device connections
- Support for ITU-R BT.656-4 SAV and EAV embedded synchronization
- Pre-processed capture modes:
  - RGB 8:8:8, RGBT 8:8:8, RGB 6:6:6, RGB 5:6:5, RGB 5:5:5, RGBT 5:5:5, RGB 4:4:4 data formats
  - YCbCr 4:2:2 data format
  - RGB component precision reductions for RGB 8:8:8
- Raw capture modes including RGGB and CMYG
- Support for packing of 8-, 9-, and 10-bit raw pixel precision
- Support for both packed and planar data formatting for YCbCr 4:2:2 formats
- Programmable vertical and horizontal resolutions up to 2048 x 2048
- Two 8-entry (by 64 bits) and one 16-entry (by 64 bits) FIFOs
- Programmable sensor-clock output from 196.777 kHz to 52 MHz
- Programmable interface timing signals for internal and external synchronization
- Programmable interrupts for FIFO overflow, end-of-line, and end-of-frame
- Programmable frame-capture rate allows users to capture all frames or 1 out of every 2 to 8 frames

See Chapter 27, "Quick Capture Interface" for additional information.

1.2.19 Test Interface

The boundary-scan interface has the following features:

- JTAG interface

- Conforms to the IEEE Std. 1149.1–1990 and IEEE Std. 1149.1a-1993, *Standard Test Access Port and Boundary-Scan Architecture*
- Test-access port with dedicated pins: TDI, TMS, TCK, nTRST, and TDO
- Mini-instruction cache

For information on using the JTAG interface, see the "JTAG Debug Interface" chapter in the *Intel*® PXA27x Processor Family Design Guide.

1.3 Intel XScale® Microarchitecture Compatibility

The Intel XScale® microarchitecture complies with the ARM* Architecture V5TE. The PXA27x processor implements the integer instruction set of the ARM* Architecture V5TE.

Backward compatibility for user-mode applications is maintained with the first generation of Intel® StrongARM* products. Operating systems require modifications to match the specific Intel XScale® core hardware features and to take advantage of the performance enhancements added to this core.

Memory map and register locations are backward-compatible with the previous Intel XScale® microarchitecture hand-held products (see Section 1.3.1 for exceptions).

The Intel® Wireless MMX™ instruction set is compatible with the standard ARM* coprocessor instruction format.

1.3.1 Compatibility Exceptions

The USB client module is not backward-compatible with previous Intel XScale® microarchitecture hand-held products. The mapping of peripheral pins to GPIO is backward-compatible where possible.

System Architecture 2

This chapter outlines the core implementation, types of processor resets, and signal descriptions for the PXA27x processor.

2.1 Overview

The PXA27x processor is an implementation of the Intel XScale® microarchitecture, which is described in the *Intel XScale® Core Developer's Manual*. The characteristics of this particular implementation include the following:

- Several coprocessor registers
- Little-endian operation
- Semaphores and interrupts for processor control
- Multiple reset mechanisms
- Sophisticated power management
- Highly multiplexed pin usage

2.2 Intel XScale® Technology Implementation Options

The Intel XScale® core implementation used in the PXA27x processor includes the options outlined in the following subsections. Most of these options are specified within the coprocessor register space. Access to coprocessors other than CP14 and CP15 is controlled by the Coprocessor Access register (CPAR—see Section 2.2.5.4). To accommodate the functionality in the Intel XScale® core, registers in CP14 and CP15 have been added or augmented. To accommodate the Intel® Wireless MMX™ multimedia extensions, the registers in CP0 and CP1 have been added or augmented. For more information, refer to the following sources:

- Intel® Wireless MMX™ user documentation, listed in Table 1-1, "Supplemental Documentation" on page 1-3—Media-enhancement technology supported by the PXA27x processor
- *Intel XScale® Microarchitecture for the PXA250 and PXA210 Application Processors User's Manual*—Information on configuring the coprocessor registers
- Section 28.2.1, "Intel XScale® Microarchitecture Core Registers" on page 28-4—Complete list of coprocessor registers

The following subsections describe the coprocessor registers:

- Section 2.2.1, “Interrupt Controller Registers”
- Section 2.2.2, “Performance Monitoring Registers”
- Section 2.2.3, “Clock Configuration and Power Management Registers”
- Section 2.2.4, “Coprocessor Software Debug Registers”
- Section 2.2.5, “Coprocessor 15”

2.2.1 Interrupt Controller Registers

Access: Coprocessor 6

The interrupt controller registers can be accessed in either of two modes:

- Memory-mapped register access mode
- Coprocessor-register access mode. This mode results in significantly reduced interrupt latencies. Accessing the interrupt controller registers in coprocessor-register access mode must be performed in supervisor mode, as described in Section 25.4.1, “Accessing Interrupt Controller Registers” on page 25-4.

2.2.2 Performance Monitoring Registers

Access: Coprocessor 14—See Table 2-1

The performance-monitoring registers include four 32-bit performance counters, allowing four separate events to be monitored simultaneously. In addition, a 32-bit clock counter is available, which counts the number of core clock cycles. For additional information, refer to the Performance Monitoring section of the *Intel XScale® Core Developer’s Manual*.

Table 2-1. Performance Monitoring Registers (Sheet 1 of 2)

Name	Description	CRm	CRn	Instruction
PMNC	Performance Monitor Control register	0	1	Read: MRC p14, 0, Rd, c0, c1, 0 Write: MCR p14, 0, Rd, c0, c1, 0
CCNT	Clock Counter register	1	1	Read: MRC p14, 0, Rd, c1, c1, 0 Write: MCR p14, 0, Rd, c1, c1, 0
INTEN	Interrupt Enable register	4	1	Read: MRC p14, 0, Rd, c4, c1, 0 Write: MCR p14, 0, Rd, c4, c1, 0
FLAG	Overflow Flag register	5	1	Read: MRC p14, 0, Rd, c5, c1, 0 Write: MCR p14, 0, Rd, c5, c1, 0
EVTSEL	Event Selection register	8	1	Read: MRC p14, 0, Rd, c8, c1, 0 Write: MCR p14, 0, Rd, c8, c1, 0
PMN0	Performance Count register 0	0	2	Read: MRC p14, 0, Rd, c0, c2, 0 Write: MCR p14, 0, Rd, c0, c2, 0

Table 2-1. Performance Monitoring Registers (Sheet 2 of 2)

Name	Description	CRm	CRn	Instruction
PMN1	Performance Count register 1	1	2	Read: MRC p14, 0, Rd, c1, c2, 0 Write: MCR p14, 0, Rd, c1, c2, 0
PMN2	Performance Count register 2	2	2	Read: MRC p14, 0, Rd, c2, c2, 0 Write: MCR p14, 0, Rd, c2, c2, 0
PMN3	Performance Count register 3	3	2	Read: MRC p14, 0, Rd, c3, c2, 0 Write: MCR p14, 0, Rd, c3, c2, 0

2.2.3 Clock Configuration and Power Management Registers

Access: Coprocessor 14, Registers 6 and 7

The CCLKCFG (register 6) and PWRMODE (register 7) registers allow software to modify the clock and power-management modes. The valid operations are described in Section 3.8.3, "Coprocessor 14: Clock and Power Management" on page 3-103.

2.2.4 Coprocessor Software Debug Registers

Access: Coprocessor 14, registers 8 through 14
Coprocessor 15, register 14

These registers are used for software debug. See Table 26.6, "Register Summary" on page 26-46 for additional information.

2.2.5 Coprocessor 15

The following subsections describe the registers available in coprocessor 15:

- Section 2.2.5.1, "Processor ID Register"
- Section 2.2.5.2, "Processor Cache Type Register"
- Section 2.2.5.3, "Auxiliary Control Register (P-Bit)"
- Section 2.2.5.4, "Coprocessor Access Register"
- Section 2.2.5.5, "Additions to Coprocessor 15 Functionality"

2.2.5.1 Processor ID Register

Access: Coprocessor 15, Register 0, opcode_2 = 0

Table 2-2 shows the format and values presented in the read-only Processor ID register, which is accessible only in supervisor mode. It conforms with the values provided in the *ARM® Architecture Reference Manual*.

Table 2-2. Processor ID Register

Coprocessor 15 Register 0 opcode_2 = 0	Processor ID Register																Processor ID													
Bit	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
	Vendor								Arch Version								Core G			Core R			Prod ID						Prod R			
Reset	0	1	1	0	1	0	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0	0	0		†	†	†	†

Bits	Access	Name	Description
31:24	R	Vendor	Vendor = Intel (0x69 = “i” = Intel Corporation)
23:16	R	Arch Version	ARM* Architecture Version 5TE = 0b0000_0101
15:13	R	Core G	Core Generation Intel XScale® core = 0b010
12:10	R	Core R	Core Revision = 0b000 This field reflects revisions of core generations. Differences can include errata that dictate different operating conditions and software work-arounds.
9:4	R	Prod ID	0b01_0001 = PXA27x processor†
3:0	R	Prod R	Processor Stepping 0b0000 = A0 0b0001 = A1 0b0010 = B0 0b0011 = B1 0b0100 = C0 0b0111 = C5

† These values reflect the actual product identification and revision numbers embedded in the PXA27x processor.

Table 2-3. Coprocessor: CPU ID and JTAG ID Values

Stepping	CPU ID	JTAG ID
A0	0x69054110	0x09265013
A1	0x69054111	0x19265013
B0	0x69054112	0x29265013
B1	0x69054113	0x39265013
C0	0x69054114	0x49265013
C5	0x69054117	0x79265013

2.2.5.2 Processor Cache Type Register

Access: Coprocessor 15, Register 0, opcode_2 = 1

The Processor Cache Type register describes the cache configuration of the Intel XScale® core. The cache configuration for the PXA27x processor is described in the *Intel XScale® Core Developer's Manual*. The PXA27x processor cache configuration is identical to that of the Intel® PXA255 processor.

2.2.5.3 Auxiliary Control Register (P-Bit)

Access: Coprocessor 15, Register 1, opcode_2 = 1

Bit 1 of the Auxiliary Control register is defined as the page-table memory-attribute bit, or P-bit. It is not implemented in the PXA27x processor and must be written with 0b0. Similarly, the P-bit in the memory management unit (MMU) page table descriptor is not implemented and must be written with 0b0.

2.2.5.4 Coprocessor Access Register

Access: Coprocessor 15, Register 15, opcode_2 = 0, CRm = 1

The Coprocessor Access register (CPAR), defined in Table 2-4, controls access to all coprocessors other than CP14 and CP15. This register is accessible only in supervisor mode.

Example 2-1 demonstrates setting the CPAR while in supervisor mode.

Table 2-4. Processor CPAR Register

Coprocessor 15 Register 15 opcode_2 = 0 CRm = 1	Processor CPAR Register	Processor CPAR

User Settings																															
Bit	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																		CP13	CP12	CP11	CP10	CP9	CP8	CP7	CP6	CP5	CP4	CP3	CP2	CP1	CP0
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

Bits	Access	Name	Description
31:14	—	—	reserved
13:0	R/W	CPn	Coprocessor Access Rights 0 = Access denied. Any attempt to access the corresponding coprocessor generates an undefined exception, even in supervisor mode. 1 = Access allowed, including read and write access to the coprocessor.

Example 2-1. Enabling Access to CP0, CP1, and CP6

```
;; The following code sets bits 0, 1, and 6 of the CPAR.
;; This enables access to Intel® Wireless MMX(TM) media enhancements.
;; This enables access to interrupt controller coprocessor registers.

LDR R0, =0x0043                         ; Set bits 0,1, and 6.
MCR P15, 0, R0, C15, C1, 0              ; Move to CPAR.
CPWAIT                                  ; Wait for effect (Section 2.2.5.5).
```

2.2.5.5 Additions to Coprocessor 15 Functionality

At times, it is necessary to know exactly when a CP15 update takes effect. For example, when enabling memory address translation (turning on the MMU), it is vital to know when the MMU is actually guaranteed to be in operation. To address this need, a processor-specific code sequence is defined for the Intel XScale® core. Example 2-2 describes this sequence, CPWAIT.

Example 2-2. CPWAIT: Canonical Method to Wait for CP15 Update

```
;; The following macro should be used when software needs to be
;; assured that a CP15 update has taken effect.
;; It may only be used while in a privileged mode, because it
;; accesses CP15.

MACRO CPWAIT
    MRC P15, 0, R0, C2, C0, 0           ; arbitrary read of CP15
    MOV R0, R0                          ; wait for it
    SUB PC, PC, #4                      ; branch to next instruction
    ; At this point, any previous CP15 writes are
    ; guaranteed to have taken effect.
ENDM
```

When setting multiple CP15 registers, software can opt to execute CPWAIT only once, after a sequence of MCR instructions.

The CPWAIT sequence guarantees that CP15 updates are complete by the time the CPWAIT is complete. It is possible that a CP15 side effect might occur *before* CPWAIT completes or is issued. Use the technique shown in Example 2-2 to ensure that this does not affect the code correctness.

2.3 Endianness

Endianness is the convention that describes the order in which bits within a word are stored in memory.

The PXA27x processor operates in Little Endian mode only, in which the least significant byte (LSB) of a value is stored in memory at a lower address than the most significant byte (MSB). For example, the value 0x1234_5678 at address 0x0 in a little-endian system appears as shown in Table 2-5.

Table 2-5. Little-Endian Value Encoding

Address	0	1	2	3
Byte Value	0x78	0x56	0x34	0x12

2.4 I/O Ordering

The PXA27x processor uses queues that accept memory requests from the four internal masters: CPU core, DMA controller, USB host, and LCD controller. Operations issued by each master are completed in the order they are received. Operations from one master can be interrupted by operations from another master. The PXA27x processor does not provide a software method to control the order of operations from different masters.

Loads and stores to internal addresses are completed more quickly (generally) than those issued to external addresses. The difference in completion time allows one operation to be received before another operation, but completed after the second operation.

In the following sequence, the store to the address in r4 is completed before the store to the address in r2 because the first store waits for memory in the queue while the second is not delayed.

```
str r1, [r2]; store to external memory address [r2]
str r3, [r4]; store to internal (on-chip) memory address [r4]
```

If the two stores are control operations that must be completed in order, insert a load to an unbuffered, uncached memory page followed by an operation that depends on data from the load:

```
str r1, [r2]; first store issued
ldr r5, [r6]; load from external unbuffered, uncached address ([r2] if possible)
mov r5, r5; nop stalls until r5 is loaded
str r3, [r4]; second store completes in program order
```

2.5 Semaphores

The *swap* (SWP) and *swap-byte* (SWPB) instructions, as described in the ARM* V5TE architecture reference, can be used for semaphore manipulation. No on-chip master or process can access a memory location between the load and store portion of a SWP or SWPB to the same location.

Note: It is not possible for an external companion chip (through the use of the MBREQ/MBGNT handshake) to take ownership of the bus during a SWP or SWPB instruction. Therefore, software coherency management is not needed when external companion chips manipulate semaphores.

2.6 Interrupts

The interrupt controller is described in detail in Chapter 25, “Interrupt Controller”. All on-chip interrupts are enabled, masked, and routed to the core FIQ or IRQ. Each peripheral-unit interrupt is enabled or disabled at the source through an interrupt-enable bit. Generally, all interrupt bits in a unit are ORed together and present a single value to the interrupt controller.

Each interrupt goes through the Interrupt Controller Mask register. Then the Interrupt Controller Level register directs the interrupt into either the IRQ or FIQ. If an interrupt is taken, the Interrupt Controller Pending register can be read to identify the source. After identifying the interrupt source, software services the interrupt and clears it in the source unit before exiting the service routine.

Note: There is a delay between writing to a status bit and the interrupt actually being cleared by the system. Therefore, clear the interrupt early in the interrupt-service routine to allow the status bit time to clear before returning from the routine.

2.7 Reset

The PXA27x processor can be reset in any of five ways. Table 2-6 summarizes the effects of each kind of reset. See Section 3.4, “Reset Manager Operation” on page 3-6 for more descriptions and details of these resets:

- **Power-on reset**, equivalent to hardware reset, occurs at initial power-on when the power supply is detected on VCC_BATT.
- **Hardware reset** results from asserting nRESET, which forces all units into a reset state.
- **Watchdog reset** results from a time-out in the OS timer and can recover control from runaway code by resetting the processor and peripherals. Watchdog reset is disabled by default and must be enabled by software. For more information, see Chapter 22, “Operating System Timers”.
- **GPIO reset** is a “soft” reset, which preserves some of the registers and real-time clocks.
- **Sleep-exit reset** provides a reset to modules that have been powered down in sleep or deep-sleep mode so that they can recover properly when powered up to resume normal operation.

Each type of reset except sleep-exit affects the reset states of the processor pins. For details of these states, see the “Pin Usage” section in the *Intel® PXA270 Processor Electrical, Mechanical, and Thermal Specification* and *Intel® PXA27x Processor Family Electrical, Mechanical, and Thermal Specification* (*Intel® PXA27x Processor Family EMTS*).

Waking from sleep or deep-sleep mode causes a sleep-exit reset.

The Reset Controller Status register (see Section 3.8.1.10, “Reset Controller Status Register (RCSR)” on page 3-84) contains information that allows software to determine which reset has occurred.

Table 2-6. Effect of Each Type of Reset on Internal Register State

Unit	Sleep-Exit Reset	GPIO Reset	Watchdog Reset	Hardware Reset†
Core	reset	reset	reset	reset
Memory Controller	reset	preserved	reset	reset
Internal Memory	reset	reset	reset	reset
LCD Controller	reset	reset	reset	reset
DMA Controller	reset	reset	reset	reset
Full-Function UART	reset	reset	reset	reset
Bluetooth UART	reset	reset	reset	reset
Standard UART	reset	reset	reset	reset
I^2C	reset	reset	reset	reset
I^2S	reset	reset	reset	reset
AC '97	reset	reset	reset	reset
USB Client	reset	reset	reset	reset
USB Host	reset	reset	reset	reset
Infrared Communications Port	reset	reset	reset	reset
Quick Capture Interface	reset	reset	reset	reset
RTC	preserved	preserved	all reset except RTTR	reset
OS Timers	reset††	reset	reset	reset
PWM 0,1,2,3	reset	reset	reset	reset
Keypad Interface	All keypad register states are reset but can act as wake-up events	reset	reset	reset
MSL Interface	reset	reset	reset	reset
Interrupt Controller	reset	reset	reset	reset
GPIO	All GPIO register states are reset, but GPIO<116>,GPIO<113>, GPIO<102:93>,GPIO<91:90>, GPIO<83>, GPIO<53>, GPIO<40:34>, GPIO<31>, GPIO<17:9>, GPIO<4:3>, and GPIO<1:0> can act as wake-up events	reset	reset	reset

† The power-on reset state is the same as the hardware reset state.

†† Register takes its reset value in during sleep- or deep-sleep-exit unless the pwr_I2C island retains state, which occurs if PSLR[SL_PI] is set or nTRST is asserted before sleep or deep-sleep entry.

2.8 Internal Registers

All internal registers are mapped in physical memory space on 32-bit address boundaries. Most units allow only word accesses to that unit's internal registers; however, some units allow byte or half-word accesses. Refer to the unit chapter to determine which accesses are allowed. Internal register space must be mapped as non-cacheable.

Register space where a register is not specifically mapped is defined as *reserved space*. Reading or writing reserved space causes unpredictable results.

The PXA27x processor does not use all register bit locations. The unused bit locations are marked reserved and are allocated for future use. Write reserved bit locations with zeros. Ignore the values of these bits during reads, because they are unpredictable.

2.9 Selecting Peripherals or General-Purpose I/O

Most peripherals connect to the external pins through GPIOs. To use a peripheral connected through a GPIO, first configure the GPIO so that the preferred function is selected on the GPIO pins. By default, all GPIO pins function as inputs.

To allocate a peripheral to a pin, disable the GPIO function for that pin. Then, map the peripheral function onto the pin by selecting the proper alternate function for the pin. Some GPIOs have multiple alternate functions. After a function is selected for a pin, all other functions are excluded. For this reason, some peripherals are mapped to multiple GPIOs, as shown in Section 24.4.2, "GPIO Operation as Alternate Function" on page 24-3.

Note: Multiple mapping does not mean multiple instances of a peripheral—only that the peripheral is connected to the pins in several ways.

2.10 Power-On Reset and Boot Operation

Before the device using the PXA27x processor is powered on, the system must assert nRESET and nTRST. To allow the internal clocks to stabilize, all power supplies must be stable for a specified period before nRESET and nTRST are de-asserted. When nRESET is asserted, nRESET_OUT is asserted and can be used to reset other devices in the system.

When the system de-asserts nRESET, the processor de-asserts nRESET_OUT a specified time later, and the device attempts to boot from physical address location 0x0000_0000.

The BOOT_SEL pin is sampled when reset is de-asserted, which specifies the width of the memory device from which the processor attempts to boot. For more information on reading the boot-select pins, see Section 6.5.4, "Boot Time Default Configuration Register (BOOT_DEF)" on page 6-73.

2.11 Power Management

The PXA27x processor offers a number of modes to manage power in the system. These modes range widely in levels of power savings and functionality. The following modes are supported. Section 3.6, "Power Manager Operation" on page 3-34 describes these modes in detail:

- Turbo mode—Low-latency (nanoseconds). Switch between two pre-programmed frequencies.
- Run mode—Normal full-function mode. Peripherals are unaffected.
- Idle mode—Allows stopping the CPU clock during periods of processor inactivity. Resume through an interrupt back to full-frequency processing.
- Deep-idle mode—Allows stopping the CPU clock during periods of processor inactivity. Resume through an interrupt back to 13-MHz core frequency.
- Standby mode—Low-power mode where state is retained but no activity is allowed. Both PLLs are disabled. Recovery is through external and selected internal wake-up events.
- Sleep mode—Low-power mode that does not save states but keeps I/Os powered. The RTC, power manager, and clock modules are saved, except for Coprocessor 14.
- Deep-sleep mode—Low-power mode that uses even less power than sleep mode. Same as sleep mode, but I/Os are powered down to reduce system power.

Note: To maximize power savings, particularly in low-power modes, do not allow input pins to float. Do not allow output pins to be pulled or driven to opposite levels by external devices.

2.12 Signal Descriptions

Table 2-7 describes the PXA27x processor signals used by each interface. Most of the processor pins are multiplexed so that they can be configured for one of two, three, or four available functions using the GPIO alternate-function select registers. Some signals can be configured to appear on one of several different pins.

Note: For more details about the processor pins and signals, including reset states and alternate functions, see the “Pin Usage” section in the *Intel® PXA27x Processor Family EMTS*.

Table 2-7. Intel® PXA27x Processor Signal Descriptions (Sheet 1 of 10)

Signal Name	Type	Signal Descriptions
Memory Controller Signals		
MA<25:0>	Output	**Memory Address Bus**—Drives the requested address for external memory accesses.
MD<31:0>	Bidirectional	**Memory Data Bus**—Carries data to and from external memory devices.
nOE	Output	**Memory Output Enable**—Connect to the output enables of static memory devices to control data bus drivers.
nWE	Output	**Memory Write Enable**—Connect to the write enables of SDRAM and static memory devices.
DQM<3:0>	Output	**SDRAM DQM Data Byte Mask Control for Data Bytes 3 through 0**—Connect to the data output mask enables (DQM) for SDRAM. (DQM0 corresponds to MD<7:0>, DQM1 corresponds to MD<15:8>, and so forth.)
nSDRAS	Output	**SDRAM RAS**—Connect to the row address strobe (RAS) pins for all banks of SDRAM.
nSDCAS	Output	**SDRAM CAS**—Connect to the column address strobe (CAS) pins for all banks of SDRAM. Also functions as the active low address valid strobe for synchronous flash.
SDCKE	Output	**SDRAM Clock Enable**—Connect to the clock-enable pins of SDRAM. It is deasserted during sleep. SDCKE is always deasserted upon reset. The memory controller provides control register bits for de-assertion.

Table 2-7. Intel® PXA27x Processor Signal Descriptions (Sheet 2 of 10)

Signal Name	Type	Signal Descriptions
SDCLK0	Output	**SDRAM or Synchronous Static Memory Clocks**—Connect to the clock pins of SMROM and SDRAM-timing synchronous flash. Use SDCLK0 for all static memory partitions. Use SDCLK1 for SDRAM memory partitions 0/1. Use SDCLK2 for SDRAM memory partitions 2/3. SDCLK1 and SDCLK2 are generated by dividing the internal memory clock, CLK_MEM (configured in the Core Clock Configuration register) by 1 or 2. SDCLK0 is generated by dividing the internal memory clock by 1, 2, or 4. (The divide-by-4 option supports configurations where synchronous flash memory runs at half the frequency of SDRAM.)
SDCLK1		
SDCLK2		
nSDCS<3>	Output	**SDRAM Chip Selects**—Chip selects for SDRAM memory devices, individually programmable in the memory configuration registers.
nSDCS<2>		
nSDCS<1>		
nSDCS<0>		
nCS<5>	Output	**Static Chip Selects**—Chip selects to static memory devices such as ROM and flash, individually programmable in the memory configuration registers. nCS<5:0> can be used with variable-latency I/O devices. nCS<3:0> can be used with synchronous flash.
nCS<4>		
nCS<3>		
nCS<2>		
nCS<1>		
nCS<0>		
RDnWR	Output	**Read/Write**—Indicates that the current transaction is a read (high) or a write (low)
RDY	Input	**Variable Latency I/O Ready Pin**—An external variable-latency I/O (VLIO) device asserts RDY when it is ready to transfer data.
BOOT_SEL	Input	**Boot Select—**Configures the memory controller for the bus width of the boot memory.
MBREQ	Input	**Memory Controller Alternate Bus Master Request**—Allows an external device to request the memory bus from the memory controller.
MBGNT	Output	**Memory Controller Alternate Bus Master Grant**—The memory controller asserts MBGNT to allow an external device to control the memory bus.
DVAL<1>	Output	**Fly-by DMA Data Valid 1**—Asserted by the memory controller during a fly-by DMA transfer when the external companion chip should drive data onto the bus for writes or latch the data for reads.
DVAL<0>	Output	**Fly-by DMA Data Valid 0**—Asserted by the memory controller during a fly-by DMA transfer when the external companion chip should drive data onto the bus for writes or latch the data for reads.
PC Card and CompactFlash Control Signals		
nPOE	Output	**PC Card Output Enable**—Output enable for reads from PC Card memory and PC Card attribute space.
nPWE	Output	**PC Card Write Enable**—Enables writes to PC Card memory and PC Card attribute space. Also serves as the write enable signal for variable-latency I/O.
nPIOW	Output	**PC Card I/O Write**—Asserted for writes to PC Card I/O space.
nPIOR	Output	**PC Card I/O Read**—Asserted for reads from PC Card I/O space.
nPCE<2>	Output	**PC Card Enable 2**—Selects a PC Card. nPCE<2> enables the high byte lane, and nPCE<1> enables the low byte lane.
nPCE<1>	Output	**PC Card Enable 1**—Selects a PC Card. nPCE<2> enables the high byte lane and nPCE<1> enables the low byte lane.

Table 2-7. Intel® PXA27x Processor Signal Descriptions (Sheet 3 of 10)

Signal Name	Type	Signal Descriptions
nIOIS16	Input	**I/O Select 16**—Input from the PC Card card indicating that the data bus: 0 = Data bus is 8 bits wide 1 = Data bus is 16 bits wide
nPWAIT	Input	**PC Card Wait**—Driven low by the PC Card to insert wait states, which extend transfers to and from the PXA27x processor.
PSKTSEL	Output	**PC Card Socket Select**—Used by external steering logic to route control, address, and data signals to one of the two PC Card sockets. Active-low output enable that can be used as nOE for the data transceivers. Has the same timing as the address bus. In a single socket solution: 0 = Output enable selected 1 = Output enable not selected In a dual socket solution, the socket select: 0 = Socket 0 selected 1 = Socket 1 selected
nPREG	Output	**PC Card Register Select**—Functions as address bit 26 to select register space (I/O or attribute) or memory space. Has the same timing as the address bus.
LCD Controller Signals		
LDD<17:0>	Bidirectional	**LCD Display Data**—Transfers pixel information from the LCD controller to the external LCD panel. These pins become inputs driven by the panel during a read from a panel with an integrated frame buffer.
L_CS	Output	**LCD Chip Select**—Chip select signal for LCD panels with an internal frame buffer.
L_FCLK_RD	Output	**LCD Frame Clock**—Frame clock used by the LCD display module to signal the start of a new frame of pixels that resets the line pointers to the top of the screen. This pin is also the vertical synchronization signal for active (TFT) displays. This pin is the read signal during reads from a panel with an internal frame buffers.
L_LCLK_A0	Output	**LCD Line Clock**—Indicates the start of a new line. Also referred to as Hsync (or horizontal synchronization) for active panels. For LCDs with an internal frame buffer, this signal indicates a command or data transaction.
L_PCLK_WR	Output	**LCD Pixel Clock**—Pixel clock used by the LCD display module to clock the pixel data into the Line Shift register. In passive mode, the pixel clock toggles only when valid data is available on the data pins. In active mode, the pixel clock toggles continuously, and the AC bias pin is used as an output to signal when data is valid on the LCD data pins. This pin also functions as a write signal for LCD panels with an internal frame buffer.
L_VSYNC	Input	**LCD Refresh Sync**—Sync input driven by LCDs with an internal frame buffer
L_BIAS	Output	**LCD Bias Drive**—AC bias that signals the LCD display module to switch the polarity of the power supplies to the row and column axis of the screen to counteract DC offset. In active (TFT) mode, it is used as the output enable to signal when data should be latched from the data pins using the pixel clock.
Full-Function UART Signals		
FFRXD	Input	**Full-Function UART Receive Data**
FFTXD	Output	**Full-Function UART Transmit Data**
FFCTS	Input	**Full-Function UART Clear-to-Send**
FFDCD	Input	**Full-Function UART Data-Carrier-Detect**
FFDSR	Input	**Full-Function UART Data-Set-Ready**
FFRI	Input	**Full-Function UART Ring Indicator**

Table 2-7. Intel® PXA27x Processor Signal Descriptions (Sheet 4 of 10)

Signal Name	Type	Signal Descriptions
FFDTR	Output	**Full-Function UART Data-Terminal-Ready**
FFRTS	Output	**Full-Function UART Request-to-Send**
Bluetooth UART Signals		
BTRXD	Input	**Bluetooth UART Receive Data**
BTTXD	Output	**Bluetooth UART Transmit Data**
BTCTS	Input	**Bluetooth UART Clear-to-Send**
BTRTS	Output	**Bluetooth UART Request-to-Send**
Standard UART Signals		
STD_RXD	Input	**Receive Pin for Standard UART and Slow Infrared Functions**
STD_TXD	Output	**Transmit Pin for Standard UART and Slow Infrared Functions**
Fast Infrared Communications Port Signals		
ICP_RXD	Input	**IrDA Receive Data**—Receive data pin for the fast infrared port function
ICP_TXD	Output	**IrDA Transmit Data**—Transmit data pin for the fast infrared port function
MultiMediaCard (MMC) Controller Signals		
MMCLK	Output	**MultiMediaCard and SD/SDIO Card Bus Clock**
MMCMD	Bidirectional	**MultiMediaCard Command**: MMC and SD/SDIO:Bidirectional line for command and response tokens. SPI: Output for command and write data.
MMDAT<0>	Bidirectional	**MultiMediaCard Data 0**: MMC and SD/SDIO:Bidirectional line for read and write data. SPI: Input for response token and read data.
MMDAT<1>	Bidirectional	**MultiMediaCard Data 1**: SD/SDIO:Bidirectional line for read and write data. Used only for SD 4-bit data transfers and to signal SDIO interrupts to the controller. SPI: Used only to signal SDIO interrupts to the controller.
MMDAT<2>/ MMCCS<0>	Bidirectional	**MMC Chip Select 0**: SD/SDIO:Bidirectional line for read and write data. Used only for SD 4-bit data transfers. SPI: Chip select 0
MMDAT<3>/ MMCCS<1>	Bidirectional	**MMC Chip Select 1**: SD/SDIO:Bidirectional line for read and write data. Used only for SD 4-bit data transfers. SPI: Chip select 1
Synchronous Serial Port (SSP) Signals		
SSPSCLK	Bidirectional	**Synchronous Serial Port 1 Clock—**The serial bit-clock can be configured as an output (master-mode operation) or an input (slave-mode operation).
SSPSFRM	Bidirectional	**Synchronous Serial Port 1 Frame—**The serial frame sync can be configured as an output (master-mode operation) or an input (slave-mode operation).
SSPTXD	Output	**Synchronous Serial Port 1 Transmit Data—**Serial data driven out synchronously with the bit-clock.
SSPRXD	Input	**Synchronous Serial Port 1 Receive Data—**Serial data latched using the bit-clock.

Table 2-7. Intel® PXA27x Processor Signal Descriptions (Sheet 5 of 10)

Signal Name	Type	Signal Descriptions
SSPEXTCLK/ SSPSCLKEN	Input	**Synchronous Serial Port 1 External Clock—**This input supplies an external bit-clock or an external enable request for the internally generated bit-clock.
SSPSYSCLK	Output	**Synchronous Serial Port 1 System Clock**—When enabled, provides a reference clock at four times the port 1 bit-clock.
SSPSCLK2	Bidirectional	**Synchronous Serial Port 2 Clock—**The serial bit-clock can be configured as an output (master-mode operation) or an input (slave-mode operation).
SSPSFRM2	Bidirectional	**Synchronous Serial Port 2 Frame—**The serial frame sync can be configured as an output (master-mode operation) or an input (slave-mode operation).
SSPTXD2	Output	**Synchronous Serial Port 2 Transmit Data—**Serial data driven out synchronously with the bit-clock.
SSPRXD2	Input	**Synchronous Serial Port 2 Receive Data—**Serial data latched using the bit-clock.
SSPEXTCLK/ SSPSCLKEN2	Input	**Synchronous Serial Port 2 External Clock—**This input supplies an external bit-clock or an external enable request for the internally generated bit-clock.
SSPSYSCLK2	Output	**Synchronous Serial Port 2 System Clock**—When enabled, provides a reference clock at four times the port 2 bit-clock.
SSPSCLK3	Bidirectional	**Synchronous Serial Port 3 Clock—**The serial bit-clock can be configured as an output (master-mode operation) or an input (slave-mode operation).
SSPSFRM3	Bidirectional	**Synchronous Serial Port 3 Frame—**The serial frame sync can be configured as an output (master-mode operation) or an input (slave-mode operation).
SSPTXD3	Output	**Synchronous Serial Port 3 Transmit Data—**Serial data driven out synchronously with the bit-clock.
SSPRXD3	Input	**Synchronous Serial Port 3 Receive Data—**Serial data latched using the bit-clock.
SSPSYSCLK3	Output	**Synchronous Serial Port 3 System Clock**—When enabled, provides a reference clock at four times the port 3 bit-clock.
Single-Ended USB Signals		
USB_P2_5, USB_P3_5	Input	**USB D+ Positive Receiver Input—**Connects to the positive receiver output of an external USB transceiver.
USB_P2_3, USB_P3_3	Input	**USB D– Negative Receiver Input**—Connects to the negative receiver output of an external USB transceiver.
USB_P2_6, USB_P3_6	Output	**USB D+ Positive Driver Output—**Connects to the positive driver input of an external USB transceiver.
USB_P2_4, USB_P3_4	Output	**USB D– Negative Driver Output**—Connects to the negative driver input of an external USB transceiver.
USB_P2_1, USB_P3_1	Input	**USB Receiver Input**— Connects to the receiver difference signal output of an extoller USB transceiver.
USB_P2_2, USB_P3_2	Output	**USB Transceiver Output Enable—**Enables the output driver in an external USB transceiver.
USBHPEN<3:1>	Output	**USB Host Power Enable**—Controls power IC for USB host port.
USBHPWR<3:1>	Input	**USB Host Power Indicator**—Over-current indicator from USB power IC for USB host port.

Table 2-7. Intel® PXA27x Processor Signal Descriptions (Sheet 6 of 10)

Signal Name	Type	Signal Descriptions
Differential USB Signals		
USBC_P	Bidirectional	**USB Client Positive Line**—Differential signal connects to the USB client interface.
USBC_N	Bidirectional	**USB Client Negative Line**—Differential signal connects to the USB client interface.
USBH_P	Bidirectional	**USB Host Positive Line—**Differential signal connects to the USB host interface.
USBH_N	Bidirectional	**USB Host Negative Line—**Differential signal connects to the USB host interface.
Quick Capture Interface Signals		
CIF_MCLK	Output	**Quick Capture Interface Master Clock**
CIF_PCLK	Input	**Quick Capture Interface Pixel Clock**
CIF_DD<9:0>	Input	**Quick Capture Interface Data**
CIF_FV	Bidirectional	**Quick Capture Interface Frame Synchronization—**Vertical sync signal.
CIF_LV	Bidirectional	**Quick Capture Interface Line Synchronization—**Horizontal sync signal.
Universal Subscriber Identity Module (USIM) Interface Signals		
UIO	Bidirectional	**USIM I/O**—USIM data signal. The bidirectional pad is connected directly to the USIM card. When asserted, the I/O line is forced to V_{LOW}. When deasserted, the I/O line is pulled-up by a $20K\Omega$ pull-up resistor. If the USIM function is not used, the pull-up resistor is not needed, decreasing power consumption in standby mode.
UVS0	Output	**USIM Voltage Select 0**—This output goes high to disable the USIM card power and connect VCC_USIM to VSS_IO.
nUVS1	Output	**USIM Voltage Select 1**—This output goes low to enable the external USIM card power supply that provides 1.8 V on VCC_USIM.
nUVS2	Output	**USIM Voltage Select 2—**This output goes low to enable the external USIM card power supply that provides 3.0 V on VCC_USIM.
UCLK	Output	**USIM Clock**—USIM card clock signal.
nURST	Output	**USIM Reset**—USIM card reset signal.
UDET	Input	**USIM Card Detect**
UEN	Output	**USIM Enable**
Keypad Interface Signals		
KP_DKIN<7:0>	Input	**Keypad Direct Key Inputs**
KP_MKIN<7:0>	Input	**Keypad Matrix Key Inputs**
KP_MKOUT<7:0>	Output	**Keypad Matrix Key Outputs**
Memory Stick Host Controller Signals		
MSBS	Output	**Memory Stick Bus State**—Serial protocol bus-state signal.
MSSDIO	Bidirectional	**Memory Stick Data**—Serial protocol data signal.
nMSINS	Input	**Memory Stick Insert Signal**—Detects memory stick insertion and extraction.
MSSCLK	Output	**Memory Stick Serial Clock**—Serial protocol clock signal.
Mobile Scalable Link (MSL) Signals		
BB_OB_DAT<3:0>	Output	**MSL Outbound Data**—This bus carries up to four bits of parallel data to be transmitted to the baseband processor.
BB_OB_CLK	Output	**MSL Outbound Clock**—This clock provides timing for outbound transmissions to the baseband processor.

Table 2-7. Intel® PXA27x Processor Signal Descriptions (Sheet 7 of 10)

Signal Name	Type	Signal Descriptions
BB_OB_STB	Output	**MSL Outbound Strobe**—This signal qualifier indicates that a channel identifier is on the data pins when it is asserted, and that a data nibble is on the data pins when it is deasserted.
BB_OB_WAIT	Input	**MSL Outbound Wait**—This input provides flow control for the outbound link from the baseband processor.
BB_IB_DAT<3:0>	Input	**MSL Inbound Data**—This bus carries up to four-bits of parallel data received from the baseband processor.
BB_IB_CLK	Input	**MSL Inbound Clock**—This clock provides timing for inbound transmissions from the baseband processor.
BB_IB_STB	Input	**MSL Inbound Strobe**—This signal qualifier indicates that a channel identifier is on the data pins when it is asserted, and that a data nibble is on the data pins when it is deasserted.
BB_IB_WAIT	Output	**MSL Inbound Wait**—This output provides flow-control for the inbound link back to the baseband processor.
AC '97 Controller Signals		
AC97_RESET_n	Output	**AC '97 Reset**—Active-low Codec reset.
AC97_BITCLK	Input	**AC '97 Bit-Clock**—Bit-rate clock.
AC97_SYNC	Output	**AC '97 Sync**—Frame indicator and synchronizer.
AC97_SDATA_OUT	Output	**AC '97 Serial Data Out**—Serial audio data output to the Codec for digital-to-analog conversion.
AC97_SDATA_IN_0	Input	**AC '97 Serial Data In 0**—Serial audio data from the primary Codec analog-to-digital converter.
AC97_SDATA_IN_1	Input	**AC '97 Serial Data In 1**—Serial audio data from the secondary Codec analog-to-digital converter.
AC97_SYSCLK	Output	**AC '97 System Clock**—AC '97 system clock output.
I^2S Interface Signals		
I2S_SYSCLK	Output	**I^2S System Clock**—System clock running at four times the bit-clock, which is used by the Codec only.
I2S_BITCLK	Bidirectional	**I^2S Bit-rate Clock**—I^2S bit-rate clock.
I2S_SYNC	Output	**I^2S Sync**—Sync signal to identify left/right channel data.
I2S_SDATA_OUT	Output	**I^2S Serial Data Out**—Serial data output to the Codec digital-to-analog converter.
I2S_SDATA_IN	Input	**I^2S Serial Data In**—Serial data input from the Codec analog-to-digital converter.
I^2C Interface Signals		
SCL	Bidirectional	**I^2C Clock**—Serial clock.
SDA	Bidirectional	**I^2C Data**—Serial data/address bus.
Pulse Width Modulation (PWM) Signals		
PWM_OUT<3>	Output	**Pulse Width Modulation Channel 3**—Pulse width modulator channel 3 output.
PWM_OUT<2>	Output	**Pulse Width Modulation Channel 2**—Pulse width modulator channel 2 output.
PWM_OUT<1>	Output	**Pulse Width Modulation Channel 1**—Pulse width modulator channel 1 output.
PWM_OUT<0>	Output	**Pulse Width Modulation Channel 0**—Pulse width modulator channel 0 output.

Table 2-7. Intel® PXA27x Processor Signal Descriptions (Sheet 8 of 10)

Signal Name	Type	Signal Descriptions
DMA Signals		
DREQ<2>	Input	**DMA Request 2**—DMA request from an external companion chip.
DREQ<1>	Input	**DMA Request 1**—DMA request from an external companion chip.
DREQ<0>	Input	**DMA Request 0**—DMA request from an external companion chip.
GPIO Signals		
GPIO<120:0>	Bidirectional	**General-Purpose I/O**: GPIO<116, 113, 102:93, 91:90, 83, 53, 40:34, 31, 17:9, 4:3, 1:0> These signals can be configured as dedicated GPIO wake-up sources in sleep and standby modes. GPIO<3, 1:0> can be configured as deep-sleep wake-up sources. GPIO<1:0> are dedicated deep-sleep wake-up sources from nVDD_FAULT or nBATT_FAULT assertion. GPIO<1> can alternatively be configured as a GPIO reset input signal. GPIO<8:5>These pins are reserved for PWR_CAP use only. GPIO<120:117, 115:114, 112:103, 92, 89:84, 82:54, 52:41, 33:32, 30:18, 2> These additional GPIO signals cannot be configured to generate wake-up events. **NOTE:** GPIO<120:119> are available on the PXA271, PXA272 processors only.
Crystal and Clock Signals		
PXTAL_IN	Input	**Processor Crystal Input**—Can be connected to an external 13-MHz crystal or to an external clock source.
PXTAL_OUT	Analog	**Processor Crystal Output**—Can be connected to an external 13-MHz crystal or to an external clock source (which must be complementary to PXTAL_IN or floated).
TXTAL_IN	Input	**Timekeeping Crystal Input**—Clock input that is distributed to the timekeeping control system (32.768-kHz crystal or external clock source).
TXTAL_OUT	Analog	**Timekeeping Crystal Output**—Can be connected to an external 32.768-kHz crystal or to an external clock source (which must be complementary to TXTAL_IN or floated).
HZ_CLK	Output	**Real-Time 1 Hz Clock**—Real-time 1-Hz clock (after RTC trim adjustment).
CLK_PIO	Bidirectional	**Processor Clock Input/Output**—CLK_PIO can drive a buffered version of the PXTAL_IN oscillator input, or it can be used as a clock input alternative to PXTAL_IN. The 13-MHz processor clock from the oscillator is driven out when the POUT_EN bit of the OSCC register is set. When enabled, this clock is output in sleep mode, but it is always disabled in deep-sleep mode.
CLK_TOUT	Output	**Timekeeping Clock Output**—CLK_TOUT signal is an output that drives a buffered version of the TXTAL_IN oscillator input when the TOUT_EN bit of the OSCC register is set. When enabled, this clock is output in sleep mode, but it is always disabled in deep-sleep mode.
CLK_REQ	Bidirectional	**Clock Request**—CLK_REQ signal is an input during power-on or hardware reset that indicates if the process or oscillator clock input comes from PXTAL_IN (CLK_REQ = 0) or CLK_PIO (CLK_REQ = floating). If CLK_PIO is the processor oscillator input, then CLK_REQ becomes an output indicating when the processor oscillator is required.
CLK_EXT	Input	**External Network Clock**—This input accepts an external network clock, up to 13 MHz, that can be selected as the timing reference for the mobile scalable link (MSL) interface, SSP ports, and OS timer channels.
48_MHz	Output	**48-MHz Output Clock**—Generates peripheral timing from 312-MHz peripheral clock.
OS Timers Signals		
EXT_SYNC<1>	Input	**External Sync 1**—This input provides a reset for any timer channels enabled to use it.
EXT_SYNC<0>	Input	**External Sync 0**—This input provides a reset for any timer channels enabled to use it.

Table 2-7. Intel® PXA27x Processor Signal Descriptions (Sheet 9 of 10)

Signal Name	Type	Signal Descriptions
CHOUT<1>	Output	**Timer Channel Output 1**—Periodic clock output from timer channel 11.
CHOUT<0>	Output	**Timer Channel Output 0**—Periodic clock output from timer channel 10.
Miscellaneous Signals		
PWR_EN	Output	**Power Enable for Core Power Supply**—This output, when negated, signals the power supply to remove power from the external low-voltage power domains (VCC_CORE, VCC_SRAM, VCC_PLL) because the system is entering sleep or deep-sleep mode.
SYS_EN	Output	**Power Enable for System Peripheral Power Supply**—This output, when negated, signals the power supply to remove power from the external high-voltage power domains (VCC_IO, VCC_LCD, VCC_MEM, VCC_USIM, VCC_USB, and VCC_BB) because the system is entering deep-sleep mode.
nBATT_FAULT	Input	**Main Battery Fault—**This input signals that the main battery is low or removed. Assertion causes the PXA27x processor to enter deep-sleep mode or, if PMCR[BIDAE] is set, forces an imprecise-data abort, which cannot be masked. The PXA27x processor does not recognize a wake-up event while this signal is asserted.
nVDD_FAULT	Input	**VDD Fault**—This input signals that the main power source is going out of regulation. nVDD_FAULT causes the PXA27x processor to enter deep-sleep mode or, if PMCR[VIDAE] is set, forces an imprecise-data abort, which cannot be masked. nVDD_FAULT is ignored after a wake-up event until the power supply timer completes (approximately 10 ms).
nRESET	Input	**Reset**—This active-low, level-sensitive input starts the processor from the reset vector at address 0. Assertion causes the current instruction to terminate abnormally and causes a reset. When nRESET is driven high, the processor starts execution from address 0. nRESET must remain low until the power supply is stable and the internal 13-MHz oscillator has stabilized.
nRESET_OUT	Output	**Reset Out**—Asserted when nRESET is asserted, it deasserts after nRESET is deasserted but before the first instruction fetch occurs. nRESET_OUT is asserted during power-on, hardware, watchdog, and sleep-exit resets. It is configurable for GPIO reset.
PWR_SCL	Bidirectional[2]	**Power Manager I^{2}C Clock**—The power manager I^{2}C clock signal that connects to an external power controller.
PWR_SDA	Bidirectional[2]	**Power Manager I^{2}C Data**—The power manager I^{2}C data signal that connects to an external power controller.
PWR_CAP<3:0>	Analog	**Power Capacitor<3:0>**—Must be connected to external capacitors to achieve very low power in sleep mode.
PWR_OUT	Analog	**Power Out**—Low-voltage supply created for sleep and deep-sleep modes. It connects to an isolated external capacitor.
JTAG and Test Signals		
nTRST	Input	**JTAG Test Reset**—IEEE 1194.1 test reset.
TDI	Input	**JTAG Test Data Input**—Data from the JTAG controller is sent to the PXA27x processor using this signal. This pin has an internal pull-up resistor.
TDO	Output	**JTAG Test Data Output**—Data from the PXA27x processor is returned to the JTAG controller using this signal.
TMS	Input	**JTAG Test Mode Select**—Selects the test mode required from the JTAG controller. This pin has an internal pull-up resistor.
TCK	Input	**JTAG Test Clock**—For all transfers on the JTAG test interface.
TEST	Input	**Test Mode**—Reserved for manufacturing test. Must be grounded for normal operation.
TESTCLK	Input	**Test Clock**—Reserved for manufacturing test. Must be grounded for normal operation.

Table 2-7. Intel® PXA27x Processor Signal Descriptions (Sheet 10 of 10)

Signal Name	Type	Signal Descriptions
Power and Ground Signals		
VCC_BATT	Power	**Backup Battery Supply**—Connect to the backup battery supply. If a backup battery is not required, this pin can be connected to another 2.2-V to 3.8-V system supply.
VCC_CORE<13:0>	Power	**Positive Supply for Internal Logic**—Must be connected to a low-voltage, adjustable system power supply.
VSS_CORE<17:0>	Power	**Ground for Internal Logic**—Must be connected to the common ground plane on the PCB.
VCC_PLL	Power	**Positive Supply for PLLs and Oscillators**—Must be connected to a separate, fixed 1.3-V supply.
VSS_PLL	Power	**Ground for PLL**—Must be connected to the common ground plane on the PCB.
VCC_SRAM<3:0>	Power	**Positive Supply for Internal SRAM**—Must be connected to a separate, fixed 1.1-V supply.
VCC_LCD	Power	**Positive Supply for LCD I/O**—Must be connected to an external power supply for the LCD interface.
VSS_IO<4:0>	Power	**Ground for USB Transceiver Pins, USIM Pins, LCD Pins, and MSL Pins—**Must be connected to the common ground plane on the PCB.
VCC_IO<7:0>	Power	**Positive Supply for All CMOS I/O except the Memory Bus, PC Card Pins, USB Transceiver Pins, USIM Pins, LCD Pins, and MSL Pins—**Must be connected to an external power supply for pins on the I/O domain.[1]
VCC_USB	Power	**Positive Supply for USB Transceivers** (both host and client)—Must be connected to an external power supply for the USB interface.
VCC_USIM	Power	**Positive Supply for USIM Interface**—Must be connected to an external power supply for pins on the USIM power domain.
VCC_MEM<16:0>	Power	**Positive Supply for External Memory Interface**—Must be connected to an external power supply for the external memory interface.
VSS_MEM<16:0>	Power	**Ground for External Memory Interface**
VSS<3:0>	Power	**Ground for Sleep-Active Units and Oscillators**
VCC_BB	Power	**Positive Supply for Mobile Scalable Link**—Must be connected to an external power supply for the MSL. Must be externally tied to VCC_MEM in applications using PC Card or CompactFlash memory card interfaces.
VSS_BB	Power	**Ground for Mobile Scalable Link**

NOTES:

1. Two VCC_IO die pads are attached to the four corner balls identified as A23, A24, B23, and B24 in the *Intel® PXA27x Processor Family EMTS*.
2. Although these PWR I^2C signals are intended as outputs to power controllers, they can also be used generically.

Clocks and Power Manager 3

This chapter describes the clocks and power management unit and registers supported by the PXA27x processor.

3.1 Overview

The clocks and power manager unit administers the processor resets, clocks, power management, and controls external power management ICs (PMIC). Control over these features allows optimization of the processor's overall power consumption and performance for an individual application. This chapter describes the following management and control units:

- Reset Manager (Section 3.4)

  The reset manager places the processor into one of five reset states: power on, hardware, watchdog, GPIO, or sleep-exit reset.

- Clocks Manager (Section 3.5)

  The clocks manager contains all clock generation, gating, and frequency controls for the processor.

- Power Manager (Section 3.6)

  The power manager controls the following:

  — All internal power domains and external power-supply functionality

  — Entry and exit sequences for the different power modes: normal, idle, deep idle, standby, sleep, and deep sleep.

- Voltage Manager (Section 3.7)

  The voltage manager provides dynamic or static voltage management for the processor through the use of the power manager I^2C module, which is dedicated to communication with the external VCC_CORE regulator.

3.2 Features

Wireless Intel SpeedStep® technology includes the following features:

- Five reset sources: power-on, hardware, watchdog, GPIO, and exit from sleep and deep-sleep modes (sleep-exit)
- Multiple clock-speed controls to adjust frequency, including frequency change, turbo mode, half-turbo mode, fast-bus mode, memory clock, 13M mode, A-bit mode, and AC '97.
- Switchable clock source
- Functional-unit clock gating
- Programmable frequency-change capability
- One normal-operation power mode (also called *run mode*) and five low-power modes to control power consumption (idle, deep-idle, standby, sleep, and deep-sleep modes)

- Programmable I^2C-based external regulator interface to support changing dynamic core voltage, frequency change, and power-mode coupling.

3.3 Signal Descriptions

The following signals are inputs or outputs from the clocks and power manager unit. The use of some pins can be shared with general-purpose I/O (GPIO) functions. The GPIO functionality is described in Chapter 24, "General-Purpose I/O Controller".

Table 3-1. Clocks and Power Manager I/O Signal Descriptions (Sheet 1 of 2)

Name	Type	Definition
nRESET	Input	Signals the processor to enter hardware-reset state.
nRESET_OUT	Output	Signals the system that the processor is in any reset state (configurable for sleep- and deep-sleep-exit and GPIO resets).
GPIO<n>	Bidirectional	The GPIO<n> pins are used as standby and sleep wake-up sources. For the possible values of *n*, see Section 3.3.3.
GPIO<3>	Bidirectional	The GPIO<3> pin is used as a standby, sleep, and deep-sleep wake-up source.
GPIO<1:0>	Bidirectional	The GPIO<1:0> pins are used as standby and sleep/deep-sleep wake-up sources, and as deep-sleep wake-up sources after nBATT_FAULT or nVDD_FAULT is asserted.
PXTAL_IN	Input	Can be connected to an external 13-MHz crystal or to an external clock source. For more information, see Section 3.5.2.
PXTAL_OUT	Analog	Can be connected to an external 13-MHz crystal or to an external clock source. PXTAL_OUT must be either complementary to PXTAL_IN or floated.
CLK_PIO	Bidirectional	Can output a buffered version of the PXTAL_IN oscillator input or can be used as a clock-input alternative to PXTAL_IN.
TXTAL_IN	Input	Can be connected to an external 32.768-kHz crystal or to an external clock source. TXTAL_IN is distributed to the timekeeping control system and power-management unit. (See Section 3.5.3 for more information.)
TXTAL_OUT	Analog	Can be connected to an external 32.768-kHz crystal or to an external clock source. TXTAL_OUT must be either complimentary to TXTAL_IN or floated.
CLK_TOUT	Output	Drives a buffered and inverted version of the TXTAL_IN oscillator input.
CLK_REQ	Bidirectional	Input during power-on or hardware reset that indicates to the processor whether the processor oscillator clock input comes from PXTAL_IN (CLK_REQ low) or CLK_PIO (CLK_REQ floating). If CLK_PIO is the processor oscillator input, CLK_REQ becomes an output indicating when the processor oscillator is required. For more information, see Section 3.5.1.
CLK_EXT	Input	Can be used by the mobile scalable link (MSL) interface, SSP serial ports, or the operating system (OS) timer module as a clock input.
nBATT_FAULT	Input	Signals the processor that the main battery is low or has been removed from the system.
nVDD_FAULT	Input	Signals the processor that the main power supply is going out of regulation.
PWR_EN	Output	Enables the external low-voltage power domains: VCC_CORE, VCC_SRAM, VCC_PLL
SYS_EN	Output	Enables the external high-voltage power domains: VCC_IO, VCC_MEM, VCC_LCD, VCC_USB, VCC_USIM, VCC_BB
PWR_SCL	Input/Output	Power manager I^2C clock pin
PWR_SDA	Input/Output	Power manager I^2C data pin

Table 3-1. Clocks and Power Manager I/O Signal Descriptions (Sheet 2 of 2)

Name	Type	Definition
PWR_CAP<3:0>	Analog	The PWR_CAP pins connect to external capacitors that are used with on-chip DC-DC converter circuits to achieve very low power in sleep and deep-sleep modes.
PWR_OUT	Analog	Connects to an external isolated capacitor. See Section 3.6.2.3.
48_MHz	Output	48-MHz output clock—Divided down output generated from the 312-MHz peripheral clock. Generally used for board bring-up or debug purposes.

3.3.1 Hardware Reset (nRESET)

nRESET is an active-low input that signals the processor to enter hardware-reset state. The assertion of nRESET cannot be gated and causes the processor to enter a complete and unconditional reset state. While nRESET is asserted, the processor does not recognize any external events except CLK_REQ. (See Section 3.3.11 for more information.)

3.3.2 Internal Reset (nRESET_OUT)

nRESET_OUT is an active-low output that signals the system that the processor is in reset state. nRESET_OUT is asserted during power-on, hardware, watchdog, and sleep-exit resets.

If PSLR[SL_ROD] is clear (see Section 3.8.1.11), nRESET_OUT is also asserted during sleep and deep-sleep modes. See Section 3.4, Section 3.6.9, and Section 3.6.10 for descriptions of these modes.

If PCFR[GP_ROD] is clear (see Section 3.8.1.8), nRESET_OUT is also asserted during GPIO reset.

3.3.3 GPIO Wake-Up Sources

The GPIO<*n*> pins are used as standby and sleep wake-up sources. Possible values for *n* are:

```
116   113   102   101   100   99   98   97   96   95   94
93    91    90    83    53    40   39   38   37   36   35
34    31    17    16    15    14   13   12   11   10   9
4     3     1     0
```

See Table 3-17 and Table 3-28 for details. These pins contain internal resistive pull-downs or pull-ups that are enabled during power-on, hardware, watchdog, and GPIO resets and disabled when PSSR[RDH] is clear.

The GPIO<3> pin is used as a wake-up source for standby, sleep, and deep-sleep modes. This pin has an internal resistive pull-up that is enabled during power-on, hardware, watchdog, and GPIO resets and disabled when PSSR[RDH] is clear.

The GPIO<1:0> pins are used as dedicated standby, sleep, or deep-sleep wake-up sources. If nVDD_FAULT or nBATT_FAULT caused entry or re-entry into deep-sleep mode, then these GPIO pins are the only allowed wake-up sources. Therefore, if using nBATT_FAULT or nVDD_FAULT to indicate power-supply health, at least one of these GPIO pins must be programmed as an input.

3.3.4 GPIO Reset (nRESET_GPIO/GPIO<1>)

The nRESET_GPIO signal is an alternate function to GPIO<1>. If PCFR[GPR_EN] is set, then GPIO<1> functions as nRESET_GPIO. nRESET_GPIO is an active-low input that signals the processor to enter GPIO reset state. The GPIO<1> functionality, as described in Chapter 24, "General-Purpose I/O Controller", is enabled unless PCFR[GPR_EN] is set.

See Section 3.4.6 for more information about the GPIO reset function.

3.3.5 Processor Oscillator Input (PXTAL_IN)

PXTAL_IN is a clock input that is distributed to the processor control system. PXTAL_IN can be connected to an external 13-MHz crystal or to an external clock source. If OSCC[CRI] is set, PXTAL_IN is ignored and must be grounded. See Section 3.8.2.3 for details.

3.3.6 Processor Oscillator Output (PXTAL_OUT)

PXTAL_OUT is the output of the processor clock-control crystal oscillator amplifier. If PXTAL_IN is connected to an external 13-MHz crystal, then PXTAL_OUT must be connected to the other terminal of the crystal. If PXTAL_IN is connected to an external clock source, PXTAL_OUT must be driven with a signal complementary to PXTAL_IN or left floating. Noise causes performance degradation if PXTAL_OUT is floated. See Section 3.5.2 for more information.

3.3.7 Processor Clock Input/Output (CLK_PIO/GPIO<9>)

The CLK_PIO signal can be either an output that is driven with a buffered version of the PXTAL_IN oscillator input or an input used as an alternative to PXTAL_IN, based on OSCC[CRI] (see Section 3.8.2.3). CLK_PIO can be shared with the GPIO<9> function if the buffered processor clock input/output function is not required. The GPIO<9> functionality, as described in Chapter 24, "General-Purpose I/O Controller", is enabled unless OSCC[PIO_EN] or OSCC[CRI] is set. See Section 3.8.2.3 for OSCC register details.

3.3.8 Timekeeping Oscillator Input (TXTAL_IN)

TXTAL_IN is a clock input that is distributed to the processor timekeeping control system, which includes the real-time clock (RTC) and power manager. TXTAL_IN can be connected to an external 32.768-kHz crystal or to an external clock source. If OSCC[OON] and OSCC[CRI] are both clear (see Section 3.8.2.3), TXTAL_IN can be left unconnected or grounded. See Section 3.5.3 for more information.

3.3.9 Timekeeping Oscillator Output (TXTAL_OUT)

TXTAL_OUT is the output of the timekeeping control system's crystal oscillator amplifier. If TXTAL_IN is connected to an external 32.768-kHz crystal, then TXTAL_OUT must be connected to the other terminal of the crystal. If TXTAL_IN is connected to an external clock source, then TXTAL_OUT must be driven with a signal complementary to TXTAL_IN or left floating. Noise causes performance degradation if TXTAL_OUT is floated. See Section 3.5.3 for more information.

3.3.10 Timekeeping Clock Output (CLK_TOUT/GPIO<10>)

The CLK_TOUT signal is an output that drives a buffered version of the TXTAL_IN oscillator input. CLK_TOUT can be shared with the GPIO<10> function if the buffered timekeeping-clock-output function is not required. The GPIO<10> functionality, as described in Chapter 24, "General-Purpose I/O Controller", is enabled unless OSCC[TOUT_EN] is set. During deep-sleep mode, CLK_TOUT is held low if OSCC[TOUT_EN] is set.

The CLK_TOUT pin has an internal resistive pull-down that is enabled during power-on, hardware, watchdog, and GPIO resets and is disabled when PSSR[RDH] is clear.

3.3.11 Clock Request (CLK_REQ)

CLK_REQ is an input during power-on or hardware reset that is used for external clock source selection. See Section 3.5.1 for details of operation.

3.3.12 External Clock (CLK_EXT)

CLK_EXT is an input that can be used by the mobile scalable link (MSL), operating system (OS) timer, or SSP modules as a possible clock source for those modules.

Note: CLK_EXT does not function during standby and sleep modes if any OS timer is on. See Chapter 22, "Operating System Timers" for more information.

3.3.13 Battery Fault and VDD Fault (nBATT_FAULT, nVDD_FAULT)

nBATT_FAULT is an active-low input indicating that the main battery is low or has been removed from the system. nVDD_FAULT is an active-low input indicating that the main power supply is going out of regulation (for example, when an overload occurs).

Assertion of either of these power faults causes the processor to enter deep-sleep mode if PMCR[xIDAE] is 0, as described in Section 3.6.4, Section 3.6.10, and Section 3.6.11.

Once nBATT_FAULT or nVDD_FAULT has been asserted, the processor recognizes only GPIO<1:0> as wake-up sources.

For more information, refer to the *Intel*® PXA27x *Processor Family Power Requirements Application Note*.

3.3.14 Power Enable (PWR_EN)

PWR_EN is an active-high output that enables the external low-voltage core power supplies. Deasserting PWR_EN informs the external regulator that the processor is entering sleep or deep-sleep mode and that the *external low-voltage power domains* (which include VCC_CORE, VCC_SRAM, and VCC_PLL) can be removed.

3.3.15 System Power Enable (SYS_EN)

SYS_EN is an active-high output that enables the external high-voltage system power supplies. Deasserting SYS_EN informs the power supplies that the processor is entering deep-sleep mode and that the *external high-voltage power domains* (which include VCC_IO, VCC_LCD, VCC_MEM, VCC_USIM, VCC_USB, and VCC_BB) can be removed.

3.3.16 Power Manager I²C Clock (PWR_SCL/GPIO<3>)

PWR_SCL is the power manager I²C clock pin (see Chapter 9, "I²C Bus Interface Unit" for details of the I²C pins). PWR_SCL can be shared with the GPIO<3> function if the power manager I²C feature is not required. The GPIO functionality, as described in Chapter 24, "General-Purpose I/O Controller", is enabled unless PCFR[PI²C_EN] is set.

3.3.17 Power Manager I²C Data (PWR_SDA/GPIO<4>)

PWR_SDA is the power manager I²C data pin (see Chapter 9, "I²C Bus Interface Unit" for details of the I²C pins). PWR_SDA can be shared with the GPIO<4> function if the power manager I²C feature is not required. The GPIO functionality, as described in Chapter 24, "General-Purpose I/O Controller", is enabled unless PCFR[PI²C_EN] is set.

3.3.18 Power Manager Capacitor Pins (PWR_CAP<3:0>)

The PWR_CAP signals connect to external capacitors, which are used with on-chip DC-DC converter circuitry to achieve very low power in sleep and deep-sleep modes.

3.3.19 Power Manager Supply Output (PWR_OUT)

The PXA27x processor requires an external 0.1-µF capacitor connected to the PWR_OUT pin. This connection is the only connection or load allowed on the PWR_OUT pin. This function is not optional. The pin **must** be connected to a capacitor for correct operation.

3.3.20 48-MHz Output Clock (48_MHz)

This output signal is the divided-down output generated from the 312-MHz peripheral clock. It is generally used for board bring-up or debug.

3.4 Reset Manager Operation

3.4.1 Reset Types

The reset manager can place the PXA27x processor into one of five resets:

- Power-On Reset (Section 3.4.3 —An uncompromised, ungated, total and complete reset that is used when power is first applied to the VCC_BATT pin

- Hardware Reset (nRESET asserted) (Section 3.4.4)—An uncompromised, ungated, total, and complete reset that is used when absolutely no system information requires preservation
- Watchdog Reset (Section 3.4.5)—Enabled through the OS timer; resets all registers except those listed in Table 3-2.
- GPIO Reset (GPIO<1>) (Section 3.4.6)—Enabled with PCFR[GPR_EN], GPIO reset is an alternative to hardware reset. An external source can reset the processor while preserving the registers listed in Table 3-2.
- Sleep-Exit Reset (Section 3.6.9 and Section 3.6.10)—Provides resets to the modules that have been powered off during sleep or deep-sleep mode so that they recover properly when power is reapplied.

See Table 2-6, "Effect of Each Type of Reset on Internal Register State" on page 2-9 for the states of all processor modules during all resets.

3.4.2 Boot Sequences After Reset

The required boot sequence begins. All units in the processor (except those listed in Table 3-2) start with their predefined reset conditions. Software must examine the Reset Controller Status register (RCSR—see Table 3-23) to determine the reset source.

Each type of reset requires a boot sequence tailored to the machine states that are lost or retained during the reset. These reset states are defined in the discussion of each reset.

3.4.3 Power-On Reset

Power-on reset is invoked when a positive power supply is detected on the backup battery pin, VCC_BATT, and the nRESET pin is de-asserted.

3.4.3.1 Behavior During Power-On Reset

During power-on reset, the following conditions occur:

- All internal power domains except RTC remain powered off. (See Figure 3-2 for information on the power domains and what units are in each domain.)
- All internal registers and processes are held at their defined reset conditions.
- All clock sources are disabled; there is no activity inside the processor.
- The internal clocks are stopped and the chip is fully static.
- All pins assume their reset conditions.
- The nBATT_FAULT and nVDD_FAULT pins are ignored.
- nRESET_OUT pin is asserted when the power-on reset state occurs. For the states of processor pins during power-on reset, see the Pin Usage table in the *Intel® PXA270 Processor Electrical, Mechanical, and Thermal Specification* and *Intel® PXA27x Processor Family Electrical, Mechanical, and Thermal Specification* (*Intel® PXA27x Processor Family EMTS*).

3.4.3.2 Invoking Power-On Reset

When a positive transition is detected on the backup battery pin VCC_BATT, a power-on reset is invoked, and then nRESET is de-asserted after>10ms. VCC_BATT has to be completely powered off for more than 10 µs prior to invoking power-on reset. Typically, this occurs when VCC_BATT is asserted (the back-up battery is initially inserted into the system), before the initial system configuration. Power-on reset cannot be entered from any other mode, since the processor must be completely powered off first. See the *Intel® PXA27x Processor Family EMTS* for more details.

The following sequence occurs during power-on reset:

1. A positive transition is detected on VCC_BATT. Power must be applied in the following order:
   a. nRESET is asserted before or at the same time as VCC_BATT is asserted.
   b. VCC_BATT followed by the deassertion of nRESET (initiates the power-on reset).
   c. The high-voltage supplies VCC_IO, VCC_MEM, VCC_LCD, VCC_USB, VCC_USIM, and VCC_BB in any order except VCC_IO, which must be first.
      After the high-voltage supplies have been applied, VCC_IO must be maintained at a voltage as high as or higher than the other high-voltage supplies (*except* VCC_BATT and VCC_USB).
   d. VCC_CORE, VCC_SRAM, and VCC_PLL in any order.
2. The internal power domains are powered on.
3. The 13-MHz processor oscillator and internal PLL clock generators are enabled and wait for stabilization.
4. nRESET_OUT is deasserted.

Caution: The *Intel® PXA27x Processor Family EMTS* describes the power-supply ramp times, timing specifications and sequences and the delay between the power-supply transitions and the deassertion of nRESET_OUT. Any other supply power-on sequence might cause permanent damage to the processor.

Note: Deassertion of the nRESET pin must follow VCC_BATT detection (within specification) to initiate a power-on-reset. Removing and then re-applying VCC_BATT without the use of nRESET puts the processor into an undefined state.

3.4.4 Hardware Reset

Hardware reset is invoked when the nRESET pin is asserted. All units in the processor are reset to known states. Hardware reset is complete and total.

3.4.4.1 Behavior During Hardware Reset

During hardware reset, the following conditions occur:

- All internal registers and processes are held at their defined reset conditions.
- While nRESET is asserted, the only activity inside the processor is the stabilization of the 13-MHz processor oscillator and phase-locked loops.
- The remaining internal clocks are stopped and the chip is fully static.
- All pins assume their reset conditions, and the nBATT_FAULT and nVDD_FAULT pins are ignored.

- The nRESET_OUT pin is asserted when nRESET is asserted.

For the states of all processor pins during hardware reset, refer to the *Intel® PXA27x Processor Family EMTS*.

3.4.4.2 Invoking Hardware Reset

Hardware reset is invoked when an external source drives the nRESET input pin low. nRESET is unmaskable and is always enabled. Upon assertion of nRESET, hardware reset is entered regardless of the previous mode. For more information, see the *Intel® PXA27x Processor Family EMTS*.

The following sequence occurs during hardware reset:

1. The nRESET input is asserted for the period of time described in the *Intel® PXA27x Processor Family EMTS*. The nRESET_OUT signal is also asserted.
2. nRESET is deasserted.

Caution: VCC_BATT must be stable prior to the deassertion of nRESET. Otherwise, permanent damage to the PXA27x processor might result, and operation is undefined.

3. The 13-MHz processor oscillator and internal PLL clock generators wait for stabilization, if not stabilized already.
4. nRESET_OUT is deasserted.

3.4.5 Watchdog Reset

Watchdog reset is invoked when software fails to prevent the watchdog timer from expiring. In watchdog reset, all units in the PXA27x processor (except those listed in Table 3-2) are reset to their predefined reset states.

3.4.5.1 Behavior During Watchdog Reset

During watchdog reset, the following conditions occur:

- All units except those listed in Table 3-2 are held at their defined reset conditions.
- All pins assume their reset conditions, and the nBATT_FAULT and nVDD_FAULT pins are ignored.
- The PWR_EN and SYS_EN signals de-assert and the nRESET_OUT pin is asserted during watchdog reset.

A watchdog reset follows the same timing specifications as a power-on or hardware-reset sequence. Refer to the *Intel® PXA27x Processor Family EMTS* for timing specifications.

3.4.5.2 Invoking Watchdog Reset

Watchdog reset is invoked when OWER[WME] is set and OSMR3 matches the OS timer counter (see Chapter 22, “Operating System Timers”). When this occurs, watchdog reset is entered regardless of the previous mode. When watchdog reset is invoked, the nRESET_OUT pin is asserted.

The following sequence occurs during watchdog reset:

1. The watchdog reset source asserts. When this happens, OSMR3 matches the OS timer counter (see Section 22.4.2, “Compares and Matches” on page 22-4). The nRESET_OUT signal is asserted.
2. The watchdog reset source is reset automatically (OWER[WME] is reset (see Table 22-6, “OWER Bit Definitions” on page 22-16) as a result of propagation of the internal reset state.
3. The 13-MHz processor oscillator and internal PLL clock generators wait for stabilization, regardless of the state of the clocks at watchdog reset assertion.
4. The nRESET_OUT signal is deasserted. See the *Intel® PXA27x Processor Family EMTS* for timing details.

Note: If a watchdog reset occurs while the processor is in sleep mode, both the sleep-reset and the watchdog-reset indicator bits are set in the RCSR.

3.4.6 GPIO Reset

GPIO reset is invoked when PCFR[GPR_EN] is set (see Section 3.8.1.8) and the GPIO<1> pin is asserted low for more than 240 ns in normal and idle modes of operation.

In standby, sleep, and deep-sleep modes, GPIO reset is first treated as a wake-up event. To be recognized as a GPIO reset event, GPIO reset must be held asserted until nRESET_OUT is de-asserted as part of the normal wake up process. Holding GPIO Reset asserted until nRESET_OUT is de-asserted enables the internal power mode circuitry and all of the power supplies to stabilize.

In GPIO reset, all units in the PXA27x processor (except those listed in Table 3-2) are reset to their predefined reset states.

Note: If the flash reset signal is connected to the processor nRESET_OUT signal, then software must disable GPIO reset before putting the flash into a program or erase cycle. This ensures that a GPIO reset occurs only when the flash is in a read cycle and, if the GPIO reset occurs, then nRESET_OUT is held for at least 230 ns, which meets the flash requirements.

3.4.6.1 Enabling GPIO Reset

The nRESET_GPIO pin (GPIO<1>) has an internal pull-up that is active following any reset (except sleep- or deep-sleep-exit) until PSSR[RDH] (see Section 3.8.1.2) is cleared. Thus, PCFR[GPR_EN] must be written **before** PSSR[RDH] is cleared.

To enable the GPIO reset function, set PCFR[GPR_EN] (see Section 3.8.1.8). When PCFR[GPR_EN] is set, the GPIO reset function overrides GPIO<1> settings, regardless of its state. PCFR[GPR_EN] is not affected during standby, sleep, and deep-sleep modes, allowing use of the GPIO reset function in these modes. PCFR[GPR_EN] is not affected by GPIO reset.

GPIO reset is ignored while a frequency-change operation is in progress. If GPIO<1> remains asserted low for 240 ns after the frequency change completes, it is recognized as a GPIO reset.

3.4.6.2 Behavior During GPIO Reset

During GPIO reset, the following conditions occur:

- All units (except those listed in Table 3-2) are held at their predefined reset states.

- All pins assume their reset conditions, and the nBATT_FAULT and nVDD_FAULT pins are ignored. See the Pin Usage table in the *Intel® PXA27x Processor Family EMTS* for the states of all processor pins during GPIO reset.
- If PCFR[GP_ROD] is clear, the nRESET_OUT pin is asserted during GPIO reset.

3.4.6.3 Invoking GPIO Reset

3.4.6.3.1 Normal Power Mode

In normal power mode, GPIO reset is invoked when the function is properly configured (see Section 3.4.6.1) and nRESET_GPIO is asserted low for the time specified in the *Intel® PXA27x Processor Family EMTS*. When GPIO reset is invoked, nRESET_OUT is asserted if PCFR[GP_ROD] is clear.

In normal mode, the following sequence occurs during GPIO reset:

1. The GPIO reset source asserts the GPIO reset pin (GPIO<1>). The processor GPIO reset pin is level triggered. nRESET_OUT asserts if PCFR[GP_ROD] is clear.
2. If PCFR[GPR_EN] is set, the external GPIO reset source must be de-asserted.
3. If GPIO reset was entered while the core PLL was disabled or unlocked (such as during a power-mode or clock change), the core PLL clock generator waits for stabilization.
4. nRESET_OUT is de-asserted if PCFR[GP_ROD] is clear. See the *Intel® PXA27x Processor Family EMTS* for timing details.

3.4.6.3.2 Low-Power Modes

In the low-power modes (standby, sleep, and deep-sleep), if PCFR[GPR_EN] is set, the following GPIO reset sequence occurs:

1. The GPIO reset must be asserted for 100 µs to be treated first as a wake-up event.
2. Then, if held asserted for more than 1 ms after all power supplies are stable, it is recognized as a GPIO reset.
3. nRESET_OUT asserts if PCFR[GP_ROD] is clear.
4. If PCFR[GPR_EN] is set, the external GPIO reset source must be deasserted. (Failure to de-assert the reset source would force the GPR_EN bit to be cleared thus preventing GPIO<1> to be configured as the GPIO reset pin).
5. If GPIO reset was entered while the core PLL was disabled or unlocked (such as during a power-mode or clock change), the core PLL clock generator waits for stabilization.
6. nRESET_OUT is de-asserted if PCFR[GP_ROD] is clear. See the *Intel® PXA27x Processor Family EMTS* for timing details.

Note: If a GPIO reset occurs while the processor is in sleep or deep-sleep mode, both the corresponding sleep-reset and the GPIO-reset indicator bits are set in the Reset Controller Status register (RCSR).

Note: GPIO<1> is not recognized as a reset source again until configured to do so in software. Software must check the state of GPIO<1> before configuring as a reset to ensure that no spurious reset is generated.

3.4.7 Summary of Module Reset Sensitivity

The registers and functions of most modules assume their default (reset) values when entering any of the five reset states. Table 3-2 describes which modules are reset for the various reset modes.

Table 3-2. Summary of Module Reset Functions

Module	Register Name or Function	Sleep-Exit	GPIO	Watchdog	Hardware	Power-On
OS Timers	All registers and functions	x[1]	x	x	x	x
PWR_I^{2}C	All registers and functions	x[2]	x	x	x	x
Clocks and Power Manager	PGSR0, PGSR1, PGSR2, PGSR3	x	x	x	x	x
	PVCR, PCMD	x[2]	x	x	x	x
	PSSR, PSPR, PWER, PRER, PFER, PEDR, PKWR, PKSR		x	x	x	x
	All bits except GP_ROD in PCFR		x	x	x	x
	GP_ROD bit in PCFR			x	x	x
	CCCR, CKEN, CCSR			x	x	x
	OSCC				x	x
RTC	RTTR				x	x
RTC	Other registers and functions			x	x	x
I/O Pins	See the Pin Usage table in the *Intel® PXA27x Processor Family EMTS*.		x	x	x	x
MEMC	All registers and functions	x		x	x	x

NOTES:

x: Register takes its reset value when the corresponding reset is asserted

x[1]: Register takes its reset value in during sleep- or deep-sleep-exit unless the pwr_I^{2}C island retains state, which occurs if PSLR[SL_PI] is set or nTRST is asserted before sleep or deep-sleep entry.

x[2]: Register takes its reset value in during sleep- or deep-sleep-exit unless the pwr_I^{2}C island retains state, which occurs if PCFR[PI^{2}C_EN] is set or nTRST is asserted before sleep or deep-sleep entry.

Any module not listed takes the reset value for all of its registers.

3.4.8 Summary of Reset Sequences

All of the processor resets follow similar sequences. Table 3-3 summarizes the sequence for each type of reset and the differences between them.

Table 3-3. Summary of Reset Sequences (Sheet 1 of 2)

Description of Action	Clock Source	Latency (Cycles)	Sleep-Exit	GPIO	Watchdog	Hardware	Power-On
Reset source asserted	-		x	x	x	x	x
Reset source synchronized to clocks/power manager state machine	13M	3		x			
Reset asynchronously distributed; nRESET_OUT asserted	-		x[1]	x[3]	x	x	x
All I/O, CPU, peripheral modules take reset state (except as noted in the Pin Usage table in the *Intel® PXA27x Processor Family EMTS* and text in this chapter). All clock sources are driven from the 13-MHz processor oscillator.	-		x	x	x	x	x
PLLs, 13-MHz processor oscillator disabled	-				x	x	x

Table 3-3. Summary of Reset Sequences (Sheet 2 of 2)

Description of Action	Clock Source	Latency (Cycles)	Sleep-Exit	GPIO	Watchdog	Hardware	Power-On
13-MHz processor oscillator re-enabled, self-biases and stabilizes	13M	64k			x	x	x
Core PLL (CPLL) and peripheral PLL (PPLL) enabled	13M	3	x	x[2]	x	x	x
Core PLL and peripheral PLL self-bias	-	<100 µs	x	x[2]	x	x	x
Core PLL and peripheral PLL stabilize	13M	512	x	x[2]	x	x	x
All clock sources switched to functional source (PPLL, CPLL, processor oscillator)	13M	3	x	x	x	x	x
Final clock stabilization	CPLL	512	x	x	x	x	x
nRESET_OUT deasserted, boot sequence begins; RCSR written	-		x[1]	x[3]	x	x	x
NOTES:							
x: These steps apply if the corresponding reset has been asserted.							
x[1]: If PSLR[SL_ROD] is set, do not assert nRESET_OUT.							
x[2]: If PLL was disabled when GPIO reset was asserted, these steps apply.							
x[3]: If PCFR[GP_ROD] is set, do not assert nRESET_OUT.							

3.5 Clocks Manager Operation

The clocks manager contains all clock generation, gating, and frequency controls for the PXA27x processor. Clock generation and distribution consists of the following sources:

- 13-MHz processor oscillator—Creates the PLL reference clock and the functional clock for several units.
- 32.768-kHz timekeeping oscillator—Creates the low-power, low-frequency clock for timekeeping functions during low-power modes.
- Peripheral phase-locked loop (312 MHz)—Creates the fixed-frequency clocks for the peripheral bus and the peripheral units listed in Table 3-6.
- Core phase-locked loop (26–624 MHz)—Creates the programmable frequency clocks for the core, LCD controller, memory controller, and the system bus, although the core can run at 13 MHz in some modes.

Note: 624 MHz is available on the PXA270 processor only. See the *Intel® PXA270 Processor Electrical, Mechanical, and Thermal Specification* and *Intel® PXA27x Processor Family Electrical, Mechanical, and Thermal Specification* for supported frequency product points.

- Memory controller clock input—Sets the memory controller to run at the same frequency as the system bus. (CCCR[A] bit).
- Functional-unit clock gates—Enables and disables clocks to the functional units.

The clocks manager adjusts the frequencies of the internal clocks. Adjusting the frequency of the processor and peripheral system is one of the most effective methods of optimizing power consumption for an application's performance requirements. Applications might also require frequency changes for system reasons. The primary means of changing frequency are:

- Turbo/run selection—Changes the processor frequency relative to the system frequency. It is enabled by setting CLKCFG[T] (see Section 3.8.3.1).

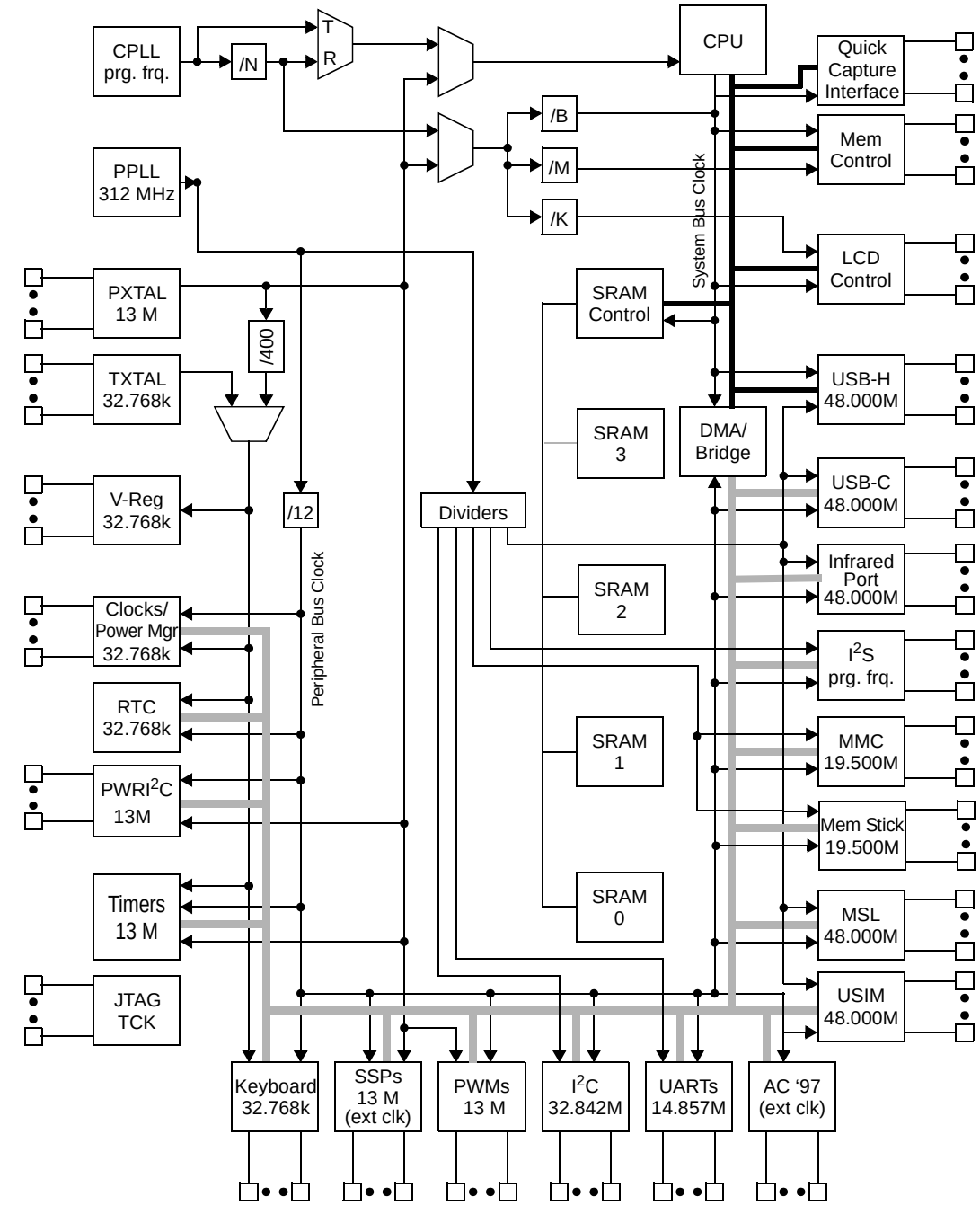
- Fast-bus mode—Changes the ratio of the system bus to the CPU run-mode frequency from 1:2 (reset value) to 1:1. It is enabled by setting CLKCFG[B].
- Frequency change—Changes the frequency of the entire system to a new, unrelated frequency with some interruption in system operation. It is enabled by setting CLKCFG[F].

Note: Refer to Section 3.5.7 for more information.

Figure 3-1 illustrates the clock distribution within the processor. The diagram uses the following definitions, where L, A, and N are programmed in the Core Clock Configuration register (see Table 3-31) and B is programmed in the Clock Configuration register (see Section 3.8.3.1):

```
    Turbo-mode frequency (T) = 13-MHz processor-oscillator frequency * L * N

    Run-mode frequency (R) = 13-MHz processor-oscillator frequency * L

    System-bus frequency = 13-MHz processor-oscillator frequency * L / B,
        where B = 1 (when in fast-bus mode) or B = 2 (when not in fast-bus mode)

For CCCR[A] = 0 (see Table 3-7):

    Memory-controller frequency = 13-MHz processor-oscillator frequency * L / M,
        where M = 1 (L = 2-10), M = 2 (L = 11-20), or M = 4 (L = 21-31)

    LCD frequency = 13-MHz processor-oscillator frequency * L / K,
        where K = 1 (L = 2-7), K = 2 (L = 8-16), or K = 4 (L = 17-31)

For CLKCFG[B] = 0 and CCCR[A] = 1 (see Table 3-7):

    Memory-controller frequency = 13-MHz processor-oscillator frequency * L / 2

    LCD frequency = 13-MHz processor-oscillator frequency * L / K,
        where K = 1 (L = 2-7), K = 2 (L = 8-16), or K = 4 (L = 17-31)

For CLKCFG[B] = 1 and CCCR[A] = 1 (see Table 3-7):

    Memory-controller frequency = 13-MHz processor-oscillator frequency * L

    LCD frequency = 13-MHz processor-oscillator frequency * L / K,
        where K = 1 (L = 2-7), K = 2 (L = 8-16), or K = 4 (L = 17-31)
```

Figure 3-1. Clocks Manager and Clocks Distribution Block Diagram



NOTE: ▬ Denotes system bus
▬ Denotes peripheral bus

3.5.1 External Clock Source Selection (CLK_REQ)

CLK_REQ is an input during power-on or hardware reset that sets or clears OSCC[CRI] (see Section 3.8.2.3). Table 3-4 summarizes the external clock selection as a function of OSCC[CRI], OSCC[OON], and CLK_REQ.

If CLK_REQ is driven low during power-on or hardware reset, OSCC[CRI] is cleared, and:

- The 13-MHz processor oscillator input is taken from PXTAL_IN.
- The 32.768-kHz timekeeping oscillator is software-enabled.
- CLK_PIO can be used as an output.
- CLK_REQ is an input.

If CLK_REQ is floated during power-on or hardware reset, OSCC[CRI] is set, and:

- The 13-MHz processor oscillator input is taken from the CLK_PIO pin.
- The 32.768-kHz timekeeping oscillator is enabled and driven out onto CLK_TOUT.
- PXTAL_IN must be grounded.
- CLK_REQ is an output indicating that the processor oscillator is requested from the external source.

If OSCC[CRI] is set, CLK_REQ is driven high after nRESET is de-asserted until the processor enters deep-sleep mode. During deep-sleep mode, CLK_REQ is driven low. After leaving deep-sleep mode, CLK_REQ is not resampled.

Note: Do not drive CLK_REQ high externally because the processor drives it low during deep-sleep mode.

If OSCC[OON] and OSCC[TOUT_EN] are set, CLK_TOUT does not start until the 32.768-kHz timekeeping oscillator is stable, which can take up to two seconds after power-on or hardware reset is asserted. See Section 3.5.2 for more information.

Refer to the *Intel® PXA27x Processor Family EMTS* for timing specifications.

Table 3-4. External Clock Source Selection

CLK_REQ†	OSCC[CRI]	OSCC[OON]	32.768-kHz Source	13-MHz Source
float	1	1	TXTAL_IN	CLK_PIO
0	0	0	Internally generated as PXTAL_IN / 400 = 32.5 kHz	PXTAL_IN
0	0	1††	TXTAL_IN	PXTAL_IN

NOTES:
† During power-on or hardware reset
†† Software-programmed

3.5.2 13-MHz Processor Oscillator

The 13-MHz processor oscillator provides the primary clock source for the PXA27x processor. The on-chip PLL frequency multipliers and several peripheral modules use the processor oscillator as a reference. If the application has not enabled the 32.768-kHz timekeeping oscillator, the processor oscillator also drives the real-time clock (RTC) and power manager.

The processor oscillator can be disabled during standby, sleep, and deep-sleep modes by setting PCFR[OPDE] (see Section 3.8.1.8), but only if the timekeeping oscillator is enabled and stabilized (OSCC[OOK] is set—see Section 3.8.2.3). If the timekeeping oscillator has not stabilized, the processor oscillator remains enabled for the duration of the low-power mode.

The processor oscillator can also serve as the clock source for the core, system bus, memory controller, and LCD controller by setting CCCR[CPDIS]. The peripheral PLL can be enabled by clearing CCCR[PPDIS] bit. The PPDIS bit can be set or cleared independently, regardless of the CPDIS setting. In this mode, the system bus frequency is 13 MHz regardless of the state of CLKCFG[B] (see Section 3.8.3.1). When selected, the processor oscillator is used with the same programming model as the core PLL, with the exception that the programming values are fixed, as shown in Table 3-5.

Table 3-5. Processor Oscillator Output Frequencies for 13-MHz Crystal

Run Mode Frequency (MHz)	System Bus Frequency B = 1 (MHz)	System Bus Frequency B = 0 (MHz)	CLK_MEM Frequency A = 1 (MHz)	CLK_MEM Frequency A = 0 (MHz)	LCD Frequency (MHz)
13	13	13	13	13	13 or 26†

NOTE:
† CCCR[LCD_26] determines the LCD frequency.

The 13-MHz processor oscillator input can be generated by using:

- An external crystal between the PXTAL_IN and PXTAL_OUT pins (lowest power consumption)
- An external clock source on the PXTAL_IN pin, or
- An external clock source on the CLK_PIO pin.

Observe the following precautions when driving the PXTAL_IN pin:

- OSCC[CRI] must be clear (see Section 3.8.2.3), and the CLK_REQ pin must be driven low during power-on and hardware reset.
- For best results, drive PXTAL_IN and PXTAL_OUT with complementary, terminated signals. Noise causes performance degradation if PXTAL_OUT is floated.
- Drive PXTAL_IN and PXTAL_OUT with a VSS-to-VCC_PLL rail-to-rail signal. If more than 10% overshoot or undershoot occurs, use a 1.5-kΩ series resistor or a voltage divider to reduce the level of VSS-to-VCC_PLL.

Observe the following precautions when driving the CLK_PIO pin:

- OSCC[CRI] must be set (the CLK_REQ pin must be floated during power-on and hardware reset).

- Drive CLK_PIO with a VSS-to-VCC_IO rail-to-rail signal. If more than 10% overshoot or undershoot occurs, use a 1.5-kΩ series resistor or a voltage divider to reduce the level of VSS-to-VCC_IO.
- Ground the PXTAL_IN pin, and leave the PXTAL_OUT pin floating.

If OSCC[CRI] is clear, setting OSCC[PIO_EN] drives a buffered version of the PXTAL_IN signal out to the external system on the CLK_PIO pin.

Note: The two pairs of crystal pins are located close to each other on the processor package. This arrangement is advantageous when crystals are connected to the pins because the low amplitudes and slow slew rates reduce noise coupling between the pins. If one of the crystals is replaced by an external signal source and the other is not, some degradation of the remaining crystal oscillator can result due to increased noise coupling. This effect can be reduced by limiting the slew rate on the pin(s) driven by the external source.

3.5.3 32.768-kHz Timekeeping Oscillator

The 32.768-kHz timekeeping oscillator is a low-power, low-frequency oscillator that clocks the real-time clock (RTC) and power manager. If OSCC[CRI] is set, the timekeeping oscillator is always enabled, and the CLK_TOUT pin drives a buffered version of the TXTAL_IN signal after the de-assertion of nRESET (for a period of up to 2.048 seconds (64,000 x 32-kHz clock cycles), this buffered signal might not be present because of the unknown stabilization time of the externally supplied clock source). This output can then be disabled by clearing OSCC[TOUT_EN].

If OSCC[CRI] is clear, the timekeeping oscillator and CLK_TOUT are disabled upon exit from power-on reset or hardware reset, and the RTC and power manager blocks use the 13-MHz processor oscillator divided by 400. Setting OSCC[OON] enables the timekeeping oscillator and configures the RTC and power manager to use the timekeeping oscillator after it has stabilized. CLK_TOUT is enabled by setting OSCC[TOUT_EN]. Doing so eliminates the need for the external (TXTAL) crystal oscillator, for cost savings in less power-sensitive applications.

Follow these steps to use the timekeeping oscillator and CLK_TOUT function:

1. Enable the timekeeping oscillator by setting OSCC[OON]. It is automatically set if OSCC[CRI] is set.

Note: OSCC[OON] cannot be cleared once it has been set, and the timekeeping oscillator cannot be disabled until a power-on or hardware reset occurs.

2. Wait for OSCC[OOK] to be set automatically. Do not attempt to enter standby, sleep, or deep-sleep modes with PCFR[OPDE] set until OSCC[OOK] is set. Otherwise, the 13-MHz processor oscillator is used and is not powered off.
3. Enable or disable the CLK_TOUT output as needed by setting or clearing OSCC[TOUT_EN], respectively. If OSCC[CRI] is set, OSCC[TOUT_EN] is set automatically after power-on reset or hardware reset.

For lowest power consumption, connect a 32.768-kHz crystal between the TXTAL_IN and TXTAL_OUT pins. However, some systems might have other clock sources of the same frequency, so the overall system cost is reduced by driving the TXTAL crystal pins with one of those clock sources.

Observe the following precautions when driving the TXTAL_IN and TXTAL_OUT pins:

- For best results, drive TXTAL_IN and TXTAL_OUT with complementary, terminated signals. Noise causes performance degradation if TXTAL_OUT is floated.

- Drive TXTAL_IN and TXTAL_OUT with VSS-to-VCC_PLL rail-to-rail signals. If more than 10% overshoot or undershoot is experienced, use a 1.5-kΩ series resistor or a voltage divider to reduce the level to VSS-to-VCC_PLL.

3.5.4 Peripheral Phase-Locked Loop (312 MHz)

When the CCCR[PPDIS] bit is clear (see Section 3.8.2.1), the peripheral PLL generates a fixed-frequency clock source (312 MHz) that is used in several peripherals, as shown in Table 3-6.

To save power, the peripheral PLL can be disabled by initiating a frequency change with CCCR[PPDIS] set. The frequencies of the peripheral clocks then derive from the 13-MHz processor oscillator. However, doing so causes those peripheral modules with strict bandwidth or protocol-related frequency requirements not to work. When the peripheral PLL is re-enabled, the exit sequence from the frequency change lengthens for up to 150 µs.

The generated PLL frequencies, listed in Table 3-6, are not exact, because of the choice of crystal frequency and the lack of a least common multiple between peripheral units. The 13-MHz crystal maintains the clock frequency for each unit within the unit's clock tolerance.

Note: The following configuration is not supported:

— CCCR[CPDIS] = 0 and CCCR[PPDIS] = 1

Table 3-6. Peripheral PLL Output Frequencies for 13-MHz Crystal

Units	Divide Ratio	Required Frequency (MHz)	Actual Frequency (MHz)	Systemic Error Due to Divide (%)	Total Error with PLL/ Osc Jitter (%)	Frequency, PPDIS = 1 (MHz)
Peripheral Bus	24/2	26.000	26.000	0.000	±0.025	13
USB-H, USB-C, Infrared Port, USIM	13/2	48.000	48.000	0.000	±0.025	13
MSL	13/2	48.000	48.000	0.000	±0.025	13
I^2C	19/2	33.333	32.842	–1.474	–1.500 to –1.450	13
MMC	32/2	20.000	19.500	–0.500	–2.525 to –2.475	13
UARTs	42/2	14.746	14.857	+0.754	+0.725 to +0.780	13
I^2S (48.000k)	51/2	12.288	12.235	–0.429	–0.455 to –0.400	13
I^2S (44.100k)	55/2	11.290	11.346	+0.493	+0.465 to +0.520	13
I^2S (22.050k)	(37*3)/2	5.645	5.622	–0.411	–0.440 to –0.385	13
I^2S (16.000k)	(38*4)/2	4.096	4.105	+0.226	+0.200 to +0.255	13
I^2S (11.025k)	(37*6)/2	2.822	2.811	–0.411	–0.440 to –0.385	13
I^2S (8.000k)	(38*8)/2	2.048	2.053	+0.226	+0.200 to +0.255	13
SD/SDIO (8.000k)	32/2	25.000	19.500	–5.5	–2.525 to –2.475	13

3.5.5 Core Phase-Locked Loop (Programmable)

When the CCCR[CPDIS] bit is clear (see Section 3.8.2.1), the core PLL serves as the clock source for the core, system bus, memory controller, and LCD controller. The core PLL generates three output frequencies:

- Run-mode frequency = 13-MHz processor oscillator * L

- Turbo-mode frequency = run-mode frequency * N
- Memory/LCD controller frequencies = predefined divisor of run-mode frequency based on L, where Table 3-7 gives the values of L and N along with the output frequencies. See Section 3.8.2.1 for information on programming the L and N factors.

Note: Do not exceed the maximum specified frequency for the applied VCC_CORE voltage. Observe the T_{CASE} specification. See the *Intel® PXA27x Processor Family EMTS* for details.

To reduce power, the core PLL can be disabled by performing a frequency change with CCCR[CPDIS] set. Refer to Table 3-5 for details. Setting the PLL early enable bit, CCCR[PLL_EARLY_EN], prior to a frequency change reduces the frequency change time. Refer to the *Intel® PXA27x Processor Family EMTS* for timing details.

Note: The following configuration is not supported:

— CCCR[CPDIS] = 0 and CCCR[PPDIS] = 1

Table 3-7. Clock Frequencies

Note: Refer to the *Intel® PXA27x Processor Family Specification Update* for any changes to the supported frequency points in Table 3-7.

Core Run Freq (MHz)	CLKCFG[T]	Core Turbo Freq (MHz)	CLKCFG[T]	CLKCFG[HT]	CCCR[L]	CCCR[2N]	System Bus (MHz)	CLKCFG[B]	CLK_MEM (MHz)	CCCR[A]	SDCLK<2:1> SDRAM Clocks (MHz)	MDREFR[KxDB2]†††	Synchronous Flash (MHz)	MDREFR[K0DB4]	MDREFR[K0DB2]	LCD (MHz)	C0 Stepping	C5 Stepping
13†	X	—	X	X	X	X	13	X	13	X	13	0	13	0	0	13 or 26††	X	X
91†††	0	—	0	0	7	2	45.5	0	91	0	45.5	1	22..75	1	1	91	X	X
104	0	104	1	0	8	2	104	1	104	1	104	0	52	0	1	52	X	X
104	0	156	1	1	8	6	104	1	104	1	104	0	52	0	1	52	X	X
104	0	208	1	0	8	4	104	1	104	1	104	0	52	0	1	52	X	X
208	0	208	1	0	16	2	208	1	104	0	104	0	52	0	1	104		X
208	0	208	1	0	16	2	208	1	208	1	104	1	52	1	X	104	X	X
104	0	312	1	0	8	6	104	1	104	1	104	0	52	0	1	52	X	X
208	0	312	1	0	16	3	208	1	104	0	104	0	52	0	1	104		X
208	0	312	1	0	16	3	208	1	208	1	104	1	52	1	X	104	X	X
208	0	416	1	0	16	4	208	1	104	0	104	0	52	0	1	104		X
208	0	416	1	0	16	4	208	1	208	1	104	1	52	1	X	104	X	X
208	0	520	1	0	16	5	208	1	104	0	104	0	52	0	1	104		X
208	0	520	1	0	16	5	208	1	208	1	104	1	52	1	X	104	X	X

Table 3-7. Clock Frequencies

Note: Refer to the *Intel® PXA27x Processor Family Specification Update* for any changes to the supported frequency points in Table 3-7.

Core Run Freq (MHz)	CLKCFG[T]	Core Turbo Freq (MHz)	CLKCFG[T]	CLKCFG[HT]	CCCR[L]	CCCR[2N]	System Bus (MHz)	CLKCFG[B]	CLK_MEM (MHz)	CCCR[A]	SDCLK<2:1> SDRAM Clocks (MHz)	MDREFR[KxDB2]††††	Synchronous Flash (MHz)	MDREFR[K0DB4]	MDREFR[K0DB2]	LCD (MHz)	C0 Stepping	C5 Stepping
208	0	624†††††	1	0	16	6	208	1	104	0	104	0	52	0	1	104		X
208	0	624†††††	1	0	16	6	208	1	208	1	104	1	52	1	X	104	X	X

NOTES:

† Not a PLL clock frequency. Refer to Section 3.5.7.7.

†† Use CCCR[LCD_26] to control this setting. See Table 3-31.

††† L = 7 (Core = 91.0 MHz) is used for hardware boot-up frequency only and must not be used for normal operation.

†††† KxDB2 represents K1DB2 and K2DB2

††††† 624 MHz is available on the PXA270 processor only. See the *Intel® PXA270 Processor Electrical, Mechanical, and Thermal Specification* and *Intel® PXA27x Processor Family Electrical, Mechanical, and Thermal Specification* for supported frequency product points.

3.5.6 Functional-Unit Clock Gating

Functional-unit clock gating allows software to enable and disable the clock to individual modules (peripherals) within the processor, which is an advantage for low-power system designs.

The Clock Enable register (CKEN—see Section 3.8.2.2) enables and disables the clocks to individual modules. The CKEN register values override the clock-gating functionality that might be present in some modules. These values are set only when entire modules are not being used. After power-on reset or hardware reset, software must disable the clocks to inactive modules.

If a module is temporarily quiescent but does not have clock gating functionality, use CKEN to disable the module's clock. With its clock disabled, register reads from that module return undefined values, and register writes to that module have no effect.

3.5.7 Modifying Clock Frequencies

3.5.7.1 General Procedure for Clock-Frequency Changes

Note: In this document, the term *CPU clock* refers to both the Intel XScale® microarchitecture and the Wireless MMX coprocessor clocks.

Modifying any of the processor clock frequencies requires programming two registers, in the following order:

1. The Core Clock Configuration register (CCCR—see Section 3.8.2.1) contains the clock configuration information. For details of the frequency changes, see the following sections:

   Section 3.5.7.3 — Changing Core Frequency

   Section 3.5.7.4 — Turbo Mode

Section 3.5.7.6 — Fast-Bus Mode

2. Coprocessor 14, register C6 (CLKCFG—see Section 3.8.3.1) initiates the changes programmed in CCCR when any of the following bits are written to CLKCFG. It is legal to change the settings of any of these bits when writing to CLKCFG, in which case all of the changes in CCCR are implemented.

   CLKCFG[B]—Fast-Bus Mode. When B is set, the system-bus frequency is equal to the run-mode frequency indicated in CCCR. When B is clear, the system-bus frequency is equal to half the run-mode frequency indicated in the CCCR. It is illegal to set B if CCCR[CPDIS] is set. When B is modified, the core PLL is stopped, and then restarted with the new CCCR settings. See Section 3.5.7.6 for details.

   CLKCFG[F] —Core Frequency Change. When F is set and CCCR[xPDIS] are clear, the core PLL is stopped, and then restarted with the new CCCR settings. If the core PLL is disabled (CCCR[CPDIS] set), the PLL is not restarted.

   F remains set after the frequency change, and there is no need to clear it. Clearing F does not initiate a frequency change. To initiate a new frequency change, set F again. See Section 3.5.7.3 for details.

   CLKCFG[T]—Turbo Mode. When T is set, the CPU operates at the turbo frequency; when clear, the CPU operates at the run-mode frequency. If only T is set, F is clear, and B is not altered, then the core PLL is not stopped. See Section 3.5.7.4 for details.

   CLKCFG[HT]—Half-Turbo Mode. When HT is set, whether T is set or clear, the CPU operates at the turbo frequency divided by two; when HT is clear, and T is clear, the CPU operates at the run-mode frequency; when HT is clear, and T is set, the CPU operates at the turbo frequency. If only HT is set, F is clear, and B is not altered, then the core PLL is not stopped. See Section 3.5.7.4 for details.

3.5.7.2 Special Considerations for Clock-Frequency Changes

Changing a clock frequency generally incurs the following stoppages, latencies, and special requirements (except half-turbo and turbo mode changes). Table 3-8 summarizes the latencies in terms of clock cycles.

- All interrupts to the CPU are held, causing latencies for the peripherals.
- All current instructions, including incomplete fetches, are completed.
- All outstanding stores are completed.
- The CPU clock stops. The stoppage time incurred by each type of frequency change is given in the “Preparations” subsection for the specific change.
- Depending on the requested change sequence, the system-bus clock can be halted.
- Depending on system use, some peripherals might need reconfiguration to account for the new frequencies (depending on system configuration and peripheral unit usage).

Note: (1) Each of the clock-frequency change sequences can impose additional limitations and requirements. For details of a specific type of clock-frequency change, see the subsections below.

(2) When a write to CLKCFG initiates more than one clock-frequency change at the same time,

- The total latency time is the maximum required by any of the changes initiated. In other words, the change sequences run concurrently.
- Preparations for **all** requested changes must be completed before writing to CLKCFG. See the corresponding “Preparations” section for each type of frequency change in the following subsections.

- Refer to section 6.5.1.4 for CLK_MEM changes while SDCLK<1> or SDCLK<2> is at 104 MHz

Table 3-8 summarizes the clock-frequency change sequences. The total latency for each sequence is the sum of the latencies at each step.

Table 3-8. Summary of Clock-Frequency Change Sequences

Description of Action	Unit	Latency (cycles)	Type of Change		
			Turbo	Fast Bus	Core Freq.
Write to CP14 CLKCFG register (software); interrupts are held.	CPU	1 CPU	X	X	X
All current instructions, including incomplete fetches, are completed.	CPU	? CPU	X	X	X
All outstanding stores are completed.	CPU	? SB	X	X	X
Wait for synchronization. The CPU clock halts.	CPU	<20 CPU	X	X	X
Deny all bus requests from LCD, USB-H, DMA, CPU, and memory controllers.	PM	1 SB	—	X	X
Complete all memory control transactions (allow memory controller bus requests). Place SDRAM in self-refresh mode.	MEM	? SB	—	—	X
Synchronize clocks and power manager. Switch clock sources (done in hardware).	PM	<4 13M <8 SB	—	—	X
Core-frequency change sequence splits					
Disable PLLs if appropriate (xPDIS set).	PM	2 13M	—	—	X
Enable PLLs if appropriate (xPDIS clear). Wait for PLLs to lock.	PM	2000 13M	—	—	X
Synchronize clocks and power manager.	PM	<2 13M <4 SB	—	—	X
Core-frequency change sequence merges					
Set PLL and/or SB dividers.	PM	2 SB	—	X	X
Release system bus for all transactions.	PM	1 SB	—	X	X
Wait for synchronization.	PM	<20 CPU	X	X	X
Enable clocks, enable interrupts to the CPU, and begin execution.	PM	2 SB	X	X	X

NOTES:

X	Step is followed in the corresponding clock mode.
—	Not applicable
?	Variable—depends on system/software configuration.
13M	13-MHz processor oscillator
CPU	CPU
MEM	Memory controller
PM	Power manager
SB	System bus

3.5.7.3 Changing Core Frequency

The core clock frequency can be changed in several ways:

- Selecting the 13-MHz clock source
- Changing the core PLL frequency
- Enabling turbo or half-turbo mode

The following parameters in the Core Clock Configuration register (CCCR) control the core clock frequency (see Section 3.8.2.1 for register details):

- The run-mode-to-oscillator ratio bit, CCCR[L], determines the run frequency by multiplying the external crystal oscillator input by L.
- The core PLL disable bit, CCCR[CPDIS=1], turns off the core PLL to reduce power consumption. The peripheral PLL can remain enabled, if required (CCCR[PPDIS=0]). If the core PLL is disabled, all internal processor core units derive their clocks from the 13-MHz processor oscillator. Disabling the peripheral PLL, CCCR[PPDIS=0], forces all peripherals to derive their clocks from the 13-MHz processor oscillator. The PLLs can be enabled or disabled independently by the CCCR[xPDIS] bits.
- The turbo mode bit, CLKCFG[T], selects either turbo mode or run mode.
- The turbo-mode-to-run-mode ratio bit, CCCR[2N], determines the turbo frequency by multiplying the run frequency by N.
- The half-turbo mode bit, CLKCFG[HT], selects whether the core frequency is equal to the run or turbo frequency (dependent on the setting of CLKCFG[T]).

Table 3-7 summarizes the core PLL frequencies as functions of L and N.

3.5.7.3.1 Preparations for Core-Frequency Change

Before performing a core-frequency change, review Section 3.5.7.1 and Section 3.5.7.2.

Table 3-8 summarizes the specific steps that take place during a core-frequency change. The frequency-change sequence imposes the following limitations and requirements:

- All interrupts to the CPU are held, causing latencies for the peripherals.
- All current instructions, including incomplete fetches, are completed.
- All outstanding memory-controller transactions are completed.
- The system-bus clock stops for up to 150 µs. No new system-bus transactions are allowed during the change sequence. During this time,
  - The LCD controller cannot transmit data to the LCD panel. If the panel cannot tolerate the 150-µs latency (for example, if it does not have a built-in frame buffer), disable the LCD controller before initiating the core-frequency change.
  - The USB host controller cannot access its own FIFOs or configuration registers, which might require disabling the USB host controller before initiating the core-frequency change.
- SDRAMs are automatically placed in self-refresh mode.
- The CPU clock stops for up to 150 µs.

The suspension of DMA activity and CPU interrupt service can cause overrun or underrun in some peripheral modules. If a peripheral module cannot tolerate the 150-µs system-bus clock latency, disable the module **before** initiating the core-frequency change.

Note: (1) If the PLLs are to be turned off using the xPDIS bits, then set the xPDIS bits **before** the frequency change, and clear the xPDIS bits **after** the frequency change.

(2) For best results, set both of the PMCR[xIDAE] bits in the Power Manager Control register while changing the core frequency (to allow software controlled entry into deep sleep, if nBATT_FAULT or nVDD_FAULT asserts, rather than immediate entry to deep sleep) (see Section 3.8.1.1 for register details). If the PLLs are disabled by setting the xPDIS bits, and then a standby mode entry is initiated, the CCCR[xPDIS] bits must not be altered until after exit from standby mode.

Table 3-9 summarizes the action required for each module when changing the core frequency.

Table 3-9. Required Actions Before and After Core-Frequency Change

Module	Before PLL Frequency Change (Core PLL = Clock Source)	After PLL Frequency Change (Core PLL = Clock Source)
Memory Controller	—	Reconfigure for new clock speed.
LCD Controller Quick Capture Interface	Disable unless LCD panel is tolerant of 150-µs pause in data transmittal.	Reconfigure for new clock speed and re-enable.
USB Host	Disable if intolerant of 150-µs system-bus clock stoppage and DMA/interrupt latency.	—
USB Client Infrared Port MSL I^2C Bus Interface Unit MMC UARTs AC'97 I^2S SSP Serial Ports	Disable if intolerant of 150-µs DMA/interrupt latency.	—
Keypad OS Timers RTC	Interrupts held until completion of frequency change.	—

3.5.7.3.2 Initiating Core-Frequency Change

To initiate a frequency-change operation:

1. Write the preferred values for L, N, PPDIS, and CPDIS to CCCR and T and HT to CLKCFG.
2. Set CLKCFG[F]. When this write occurs, the new core-frequency change sequence begins and the values in CCCR are applied.

Note: Do not set CLKCFG[HT] while performing a frequency change.

3.5.7.3.3 Behavior During Core-Frequency Change

While the core-frequency change sequence is executing, the following occurs before the switch in CPU clock speed is made:

1. All processor activity is stopped and all interrupt requests to the processor are held.

2. All new DMA and LCD activity is suspended.
3. All CPU loads are completed and CPU stores are sent to the system bus.
4. All outstanding memory-controller transactions are completed.
5. The memory controller places the SDRAM in self-refresh mode and drives the nRAS/nSDCS<3:0> and nCAS/DQM<3:0> pins to their self-refresh state.
6. The system bus clocks are stopped for a period of at least four cycles of the system bus clock (at the new frequency).

The DMA controller and CPU experience a stoppage of the system bus clock for up to four system-bus cycles (during a clock-source change) or up to 150 µs (during a core PLL frequency change with the core PLL selected as the clock source), plus the time it takes the memory controller to finish its outstanding transactions. The total period of DMA and CPU inactivity depends on the source/destination of memory controller transactions and the type of frequency change. The lack of DMA service means that peripheral modules can experience overrun or underrun in their FIFOs, and the lack of CPU activity means increased interrupt latency.

If a power fault (nVDD_FAULT or nBATT_FAULT) is asserted during the period of CPU inactivity, the frequency-change sequence completes first. Then, the processor enters deep-sleep mode as described in Section 3.6.4.

3.5.7.3.4 Completion of Core-Frequency Change Sequence

After the clock speed has been changed, the CPU continues execution at the next instruction after the write to CLKCFG. interrupt requests are no longer held, and any interrupts that occurred during the change sequence are sent to the CPU.

CLKCFG remains in the state that was written to it. The values in it and the Clock Configuration Status register (CCSR—see Section 3.8.2.4) provide data about the operating frequencies of the core PLL, which determines the frequencies of the CPU, LCD controller, memory controller, and system bus.

The SDRAMs are automatically brought out of self-refresh mode. Before enabling external transactions to devices with frequency-dependent configurations, reconfigure the memory controller to account for the new clock frequency. See Chapter 6, “Memory Controller” for more information on memory configuration.

If required, reconfigure the LCD controller pixel clock to account for the new frequency. If the LCD controller was disabled before the frequency change, re-enable it. See Chapter 7, “LCD Controller” for more information on LCD configuration.

3.5.7.4 Turbo Mode

The processor CPU frequency depends on the setting of the T-bit in the Clock Configuration register (CLKCFG—see Section 3.8.3.1):

- CLKCFG[T] set—CPU operates at the turbo frequency.
- CLKCFG[T] clear—CPU operates at the run frequency.

The turbo-mode and run-mode frequencies depend on the values in the Core Clock Configuration register (CCCR—see Section 3.8.2.1 for register details). Table 3-7 lists the turbo-mode frequencies.

The increased latency for entering or exiting turbo mode is an interruption in execution while current instructions and loads are completed and stores are sent to the system bus. This latency varies depending on the number and destination and source of the stores and loads, respectively, as well as other bus activity and cacheability of the interrupt handler. Interrupt requests are held until the frequency is changed, resulting in longer and less predictable interrupt latency. The increased latency is greater when performing a frequency change (F-bit) or fast-bus change (B-bit) in the same write as the T-bit in the CLKCFG register.

Also refer to Section 3.5.7.5 for half-turbo mode details.

3.5.7.4.1 Preparations for Turbo-Mode Entry and Exit

While entering or exiting turbo mode, the CPU clock halts, and interrupt requests to the CPU are held for up to eight core clock cycles.

The value of N is the ratio of the *turbo-mode-frequency* to the *run-mode frequency*. N is determined by CCCR[2N]. This value must have been loaded beforehand into the core PLL by means of a core-frequency change (see Section 3.5.7.3) and must be reflected in the Core Clock Status register (CCSR—see Section 3.8.2.4). The value reflected in CCSR[2N_S] is the value that is actually used when CLKCFG[T] is set.

Because entering or exiting turbo mode does not stop the system-bus clocks and does not alter any of the peripheral clocks (including the LCD and memory controller), no special steps are required with respect to the peripherals or memory controller.

3.5.7.4.2 Initiating Turbo-Mode Change

Follow these steps to enter or exit turbo mode:

1. Complete the preparations described in Section 3.5.7.4.1.
2. Set or clear CLKCFG[T]. When this write occurs, the CPU frequency switches to the indicated mode (turbo or normal run).

Note: If CCSR[2N_S] has the value of 0x2, the turbo-mode change sequence occurs, but the core frequency does not change.

3.5.7.4.3 Behavior During Turbo-Mode Change

While the processor is entering or exiting turbo mode, all processor activity is stopped and all interrupt requests are held. All CPU-initiated loads are completed, and all stores are sent to the system bus before the switch in CPU clock speed is made. Peripheral and memory-controller activity continues without interruption or change in behavior.

A possible issue during the entry or exit from turbo mode is the additional interrupt latency caused by holding interrupt requests while all CPU loads are completed and CPU stores are sent to the system bus.

If a power fault (nVDD_FAULT or nBATT_FAULT) is asserted during the period of CPU inactivity, the turbo-mode change sequence completes first. Then, the processor enters deep-sleep mode as described in Section 3.6.4.

3.5.7.4.4 Completion of Turbo-Mode Change Sequence

After the clock speed has been changed, the CPU continues execution at the next instruction after the write to CLKCFG. interrupt requests are no longer held, and any interrupts that occurred during the change sequence are sent to the CPU.

CLKCFG remains in the state that was written to it. The values in it and the Clock Configuration Status register provide information about the current operating frequencies of the CPU, LCD controller, memory controller, and system bus.

3.5.7.5 Half-Turbo Mode

The processor CPU frequency depends on the setting of the HT bit in the Clock Configuration register (CLKCFG—see Section 3.8.3.1):

- CLKCFG[HT] set—the CPU operates at the turbo-mode frequency divided by two.
- CLKCFG[HT] clear—the CPU operates at the run frequency if the T-bit is clear or at the turbo frequency if the T-bit is set.

Note: Half-turbo mode can be invoked only when the CCSR reflects 2*N values of 6 or 8.

The half-turbo-mode and run-mode frequencies depend on the values in the Core Clock Configuration register (CCCR—see Section 3.8.2.1 for register details). Table 3-7 lists the turbo-mode frequencies.

The increased latency for entering or exiting half-turbo mode is an interruption in execution while current instructions and loads are completed and stores are sent to the system bus. This latency varies depending on the number and destination and source of the stores and loads, respectively, as well as other bus activity and cacheability of the interrupt handler. Interrupt requests are held until the frequency is changed, resulting in longer and less predictable interrupt latency. Do not set the HT-bit in CLKCFG while performing a core PLL frequency change. After a frequency change has been performed, writing to the T-bit and the HT-bit at the same time results in the CPU frequency at the turbo-mode frequency divided by two.

3.5.7.5.1 Preparations for Half-Turbo Mode Entry and Exit

While entering or exiting half-turbo mode, the CPU clock halts, and interrupt requests to the CPU are held for up to eight core clock cycles.

The value of N is the ratio of the *turbo-mode-frequency* to the *run-mode frequency*. N is determined by CCCR[2N]. This value must have been loaded beforehand into the core PLL by means of a core-frequency change (see Section 3.5.7.3) and must be reflected in the Core Clock Status register (CCSR—see Section 3.8.2.4). The value reflected in CCSR[2N_S] is the value that is actually used when CLKCFG[HT] is set.

Because entering or exiting half-turbo mode does not stop the system-bus clocks and does not alter any of the peripheral clocks (including the LCD and memory controller), no special steps are required with respect to the peripherals or memory controller.

3.5.7.5.2 Initiating Half-Turbo Mode Change

Follow these steps to enter or exit half-turbo mode:

1. Complete the preparations described in Section 3.5.7.4.1.
2. Set or clear CLKCFG[HT] as follows:
   a. If CLKCFG[HT] is set, the CPU frequency switches to the half-turbo mode
   b. if CLKCFG[HT] is clear and T is clear, the CPU frequency is normal run-mode frequency
   c. if CLKCFG[HT] is clear and T is set, the CPU frequency is normal turbo frequency.

3.5.7.5.3 Behavior During a Half-Turbo Mode Change

While the processor is entering or exiting half-turbo mode, all processor activity is stopped and all interrupt requests are held. All CPU-initiated loads are completed and all stores are sent to the system bus before the switch in CPU clock speed is made. Peripheral and memory-controller activity continues without interruption or change in behavior.

A possible issue during the entry or exit from half-turbo mode is the additional interrupt latency caused by holding interrupt requests while all CPU loads are completed and CPU stores are sent to the system bus.

If a power fault (nVDD_FAULT or nBATT_FAULT) gets asserted during the period of CPU inactivity, the turbo-mode change sequence completes first. Then, the processor enters deep-sleep mode as described in Section 3.6.4.

3.5.7.5.4 Completion of Half-Turbo Mode Change Sequence

After the clock speed has been changed, the CPU continues execution at the next instruction after the write to CLKCFG. interrupt requests are no longer held, and any interrupts that occurred during the change sequence are sent to the CPU.

CLKCFG remains in the state that was written to it. The values in it and the Clock Configuration Status register provide information about the current operating frequencies of the CPU, LCD controller, memory controller, and system bus.

3.5.7.6 Fast-Bus Mode

The processor system-bus frequency depends on the setting of the B-bit in the Clock Configuration register (CLKCFG—see Section 3.8.3.1):

- CLKCFG[B] set—System bus operates at the full run-mode frequency.
- CLKCFG[B] clear—System bus operates at one-half the run-mode frequency.

The system-bus frequency is relative to the run-mode frequency indicated in the Core Clock Configuration register (CCCR—see Section 3.8.2.1 for register details and Table 3-7 for a summary of the frequencies).

Certain values of CCCR[2N] are illegal when CLKCFG[B] is set. For details, see Table 3-7.

The increased latency for a fast-bus mode change is a stoppage of LCD, memory controller, system bus and core clocks, lasting up to 150 µs. Additionally, an interruption in execution occurs while all outstanding processor or memory controller transactions are completed. Finally, some re-configuration of the LCD controller and memory controller might be required. Interrupt requests to the CPU are held until the new frequency is enacted, resulting in longer and less predictable interrupt latency.

3.5.7.6.1 Preparations for Fast-Bus-Mode Entry and Exit

Entering and exiting fast-bus mode incurs the same latencies and requires the same precautions as for a core frequency change. Thus, to prepare for a fast-bus mode change, follow the instructions in Section 3.5.7.3.1.

3.5.7.6.2 Initiating Fast-Bus Mode Change

Follow these steps to enter or exit fast-bus mode:

1. Complete the preparations described in Section 3.5.7.3.1.
2. Set or clear CLKCFG[B]. When this write occurs, the system-bus frequency switches to the indicated mode relative to the CPU run-mode frequency.

3.5.7.6.3 Behavior During Fast-Bus Mode Change

System behavior during the fast-bus-mode change sequence is identical to the behavior during a core frequency change, as described in Section 3.5.7.3.3.

3.5.7.6.4 Completion of Fast-Bus-Mode Change Sequence

After the system-bus speed has been changed, the peripheral continues execution at the next instruction after the write to CLKCFG. interrupt requests are no longer held, and any that occurred during the change sequence are sent to the peripheral.

CLKCFG remains in the state that was written to it. The values in it and the Clock Configuration Status register (CCSR—see Section 3.8.2.4) provide data about the operating frequencies of the peripheral, LCD controller, memory controller, and system bus.

3.5.7.7 13M Mode

The processor enters 13M mode when (1) the CPDIS is set, (2) the PPDIS is clear, and (3) a frequency change operation is performed by writing to the F-bit in the CLKCFG register. In this mode (CPDIS set and PPDIS clear), only the CPU core PLL is turned off, forcing all of the internal clocks to be derived from the 13-MHz oscillator. However, the peripheral PLL is still enabled and continues to provide the 312-MHz clock to all peripherals. Peripherals using external clocking are not altered; they continue to receive an external clock.

Allowing the peripheral PLL to continue running while the core PLL is disabled (forcing the CPU to run at 13 MHz) maintains peripheral functionality while using a lower CPU frequency.

Software can disable (optionally) both CPU core PLL and peripheral PLL (CPDIS and PPDIS are set); however, the increased latency for this is a stoppage of certain peripherals that cannot operate at 13 MHz.

Note: The LCD clock frequency (L_CLK) can be configured to 26 MHz while the processor is in 13M mode or deep idle by setting CCCR[LCD_26].

Follow these steps to avoid stoppage of the LCD clock while exiting 13M mode (when CCCR[CPDIS] is set):

1. Remain in 13M mode, but early-enable the PLL (CCCR[CPDIS] = 1 and CCCR[PLL_EARLY_EN] = 1) to allow the PLL to be started early.
2. Read CCCR and compare to make sure that the data was correctly written.
3. Verify that CCSR[CPLOCK] and CCSR[PPLOCK] bits are both set, indicating that the PLLs are locked. Proceed to the next step only when this condition is true. At this point, the processor is still in 13M mode, but the PLLs are running.

Note: Because the PLLs do not lock in less than 120 µs, software could use an OS timer to generate an interrupt after 120 µs. When the 120-µs timer interrupt occurs, the software could then begin polling CCSR[CPLOCK] and CCSR[PPLOCK]. This interrupt-driven delay would allow normal processing of other tasks at 13 MHz to continue during the time the PLLs are locking.

4. Exit 13M mode by writing 0x00 to CCCR[CPDIS,PPDIS] but maintaining CCCR[PLL_EARLY_EN] = 1. The CCCR[PLL_EARLY_EN] bit is cleared automatically after the frequency change.
5. Perform the frequency change only by setting the CLKCFG register F-bit.

Note: Entering idle mode from 13M mode puts the processor into deep-idle mode. Refer to Section 3.6.7 for details.

3.5.7.7.1 Prerequisites for Changing to 13M Mode

See Section 3.5.7.3 for details because entering or exiting 13M mode is the same as a frequency change. If software also disables the peripheral PLL, CCCR[PPDIS=1], certain peripherals cannot function in this mode—see Table 3-6 for details.

Note: Software can change into 13M mode from run mode only.

3.5.7.7.2 Initiating 13M Mode Change Sequence

To enter or exit 13M mode, perform a frequency change after setting the CCCR[CPDIS] bit. When this bit is written and a frequency change is performed by setting the F-bit in CLKCFG, the processor enters or exits 13M mode.

Note: Other bits in CLKCFG cannot be changed while entering or exiting 13M mode. While in 13M mode, software must not write to CLKCFG [B, HT, T].

3.5.7.7.3 Behavior During 13M Mode Change Sequence

Entering and exiting 13M mode is the same as a frequency change. See Section 3.5.7.3 for details.

3.5.7.7.4 Completion of 13M Mode Change Sequence

After the clock speed has been changed, the peripheral continues execution at the next instruction after the write to CLKCFG. interrupt requests are no longer held, and any interrupts that occurred during the change sequence are sent to the peripheral.

CLKCFG remains in the state that was written to it. The values in it and the Clock Configuration Status register provide information about the current operating frequencies of the peripheral, LCD controller, memory controller, and system bus.

3.5.8 Summary of Clock Modes

The clock modes follow the sequences shown in Table 3-10. The total latency of each sequence is the sum of the latencies at each step.

Table 3-10. Summary of Clock Mode Sequences

Description of Action	Unit	Latency (cycles)	Half-Turbo/Turbo Chg.	Fast-Bus Chg.	Freq. Chg.
Write to CP14 CLKCFG register (software); interrupts gated.	CPU	1 CPU	x	x	x
All current instructions, including incomplete fetches, completed.	CPU	? CPU	x	x	x
All outstanding stores pending in CPU and memory controller are completed.	CPU	? SB	x	x	x
Wait for synchronization, Halt CPU clock.	Clks	<20 CPU	x	x	x
Deny all new bus requests (from LCD, USB-H, DMA, CPU).	PM	1 SB		x	x
Complete all memory controller transactions. Place SDRAM in self-refresh mode.	MEM	? SB			x
Synchronize clocks and power manager. Switch clock sources.	PM	<4 13M <8 SB			x
Frequency change sequence splits					
Disable PLLs if appropriate (xPDIS set).	PM	2 13M			x
Enable PLLs if appropriate (xPDIS clear) and wait for PLL to lock.	PM	2k* 13M			x
Synchronize clocks and power manager.	PM	<2 13M <4 SB			x
Frequency change completion sequence merges here					
Set PLL and/or SB divider.	PM	2 SB		x	x
Release bus for all transactions.	PM	1 SB		x	x
Wait for synchronization.	PM	<20 CPU	x	x	x
Enable clocks, interrupts to CPU and begin execution.	PM	2 SB	x	x	x

NOTES:

x	Must follow this step in the corresponding clock mode.
—	Not applicable
?	Variable—depends on system/software configuration.
13M	13-MHz processor oscillator
CPU	CPU
MEM	Memory controller
PM	Power manager
SB	System bus frequency

3.6 Power Manager Operation

The power manager controls all internal power domains, external power-supply functionality, and the entry and exit sequences for the processor power modes. The functional units within the power manager are:

- Power domains—Provide the connectivity and biasing to different regions for the various power modes. All units within a power domain receive the same power supply and must be powered on and off together. (The power domains are defined on page 3-5 and illustrated in Figure 3-2.)
- Sleep-mode power supply—Provides power during sleep mode to the 32.768-kHz timekeeping oscillator, real-time clock (RTC), and power manager.
- Power manager I^2C interface—Provides a hardware-controlled interface to the external regulator for voltage management.

The processor power modes are listed in order of recovery time back to normal mode, with idle mode being the fastest and deep-sleep mode being the slowest:

- Normal mode—All internal power domains and external power supplies are fully powered and functional. The processor clocks are running.
- Idle mode—See Section 3.6.6 for more information. Clocks to the CPU are disabled; recovery is through interrupt assertion.
- Deep-idle mode—See Section 3.6.6 for more information. Clocks to the CPU are disabled; recovery is through interrupt assertion. When CCCR[CPDIS] is set, the mode is referred to as *deep-idle*.
- Standby mode—See Section 3.6.8 and Section 3.8.1.12 for more information. All internal power domains except VCC_RTC and VCC_OSC are placed in a low-power mode where state is retained but no activity is allowed. The clock sources can be disabled. Some of the internal power domains can be powered off, and both PLLs are disabled. Recovery is through external and selected internal wake-up events.
- Sleep mode—See Section 3.6.9 and Section 3.8.1.11 for more information. All internal power domains except VCC_RTC and VCC_OSC (both are internal supplies) can be powered off. All clock sources, except those used by the real-time clock (RTC) and the power manager, are disabled. The PXA27x processor PWR_EN[1] output pin de-asserts to optionally disable the external low-voltage power supplies to the processor's low-voltage domains. The remaining power domains are placed in a low-power state where state is retained but no activity is allowed. Recovery is through external and selected internal wake-up events. Because the program counter is invalid, recovery requires a system reboot (the program counter restarts from 0x0, so the core begins execution starting at the reset vector).
- Deep-sleep mode—See Section 3.6.10 for more information. All internal power domains except VCC_RTC and VCC_OSC can be powered off. All clock sources, except those used by the RTC and the power manager, are disabled. The PXA27x processor PWR_EN[1] output pin de-asserts to optionally disable the external low-voltage power supplies. The processor SYS_EN[1] output pin also de-asserts to optionally disable the external high-voltage power domains. All power domains are powered directly from the backup battery pin, VCC_BATT. The remaining power domains are placed in a low-power state where state is retained but no activity is allowed. Recovery is through external and selected internal wake-up events.

1. PWR_EN and SYS_EN are de-asserted by hardware during the entry into sleep and deep-sleep modes. The system's power management IC must be configured to correctly control the system's power supplies.

Because the program counter is invalid, recovery requires a system reboot (the program counter restarts from 0x0, so the core begins execution starting at the reset vector).

Table 3-11 summarizes the action taken in each power mode. Figure 3-2 shows an overview of the power domains and units internal to the processor. Figure 3-3 summarizes the operational modes.

Note: VCC_BATT must be on at all times.

Table 3-11. Summary of Module Power and Clocks by Power Mode

Module	Clocks					Power				
	Normal	Idle	Standby	Sleep	Deep Sleep	Normal†	Idle†	Standby†	Sleep††	Deep Sleep†,††††
CPU (Processor Core)	On	Off	Off	Off	Off	On	On	St	Off	Off
Cache Contents	—	—	—	—	—	P	P	P	NP	NP
SRAMs (Internal SRAM Banks 0, 1, 2, and 3)	On	On	Off	Off	Off	On	On	St/Off	St[6]/Off	Off
Peripherals (All Units Not Otherwise Mentioned)	C[1]	C[1]	Off	Off	Off	On	On	St	Off	Off
OS Timer and Power Manager I^2C (PI) Power Domain	C[1]	C[1]	C[1]/Off	Off	Off	On	On	On/St/Off	On/St[6]/Off	St[7]/Off
Peripheral PLL	C[2]	C[2]	Off	Off	Off	On	On	Off	Off	Off
Core PLL	C[3]	C[3]	Off	Off	Off	On	On	Off	Off	Off
Real-Time Clock, Clocks/Power Manager	On	On	On	On	On	On	On	On	On[8]	On[9]
13-MHz Processor Oscillator	On	On	C[4]	C[4]	C[4]	On[8]	On[8]	On[8]	On[8]	On[9]
32.768-kHz Timekeeping Oscillator	C[5]	C[5]	C[5]	C[5]	C[5]	On[8]	On[8]	On[8]	On[8]	On[9]

NOTES:

On—Clock or power supply is active and fully functional.

Off—Clock is disabled or power supply is powered off.

St—Standby—power supply is in a low-power state-retaining mode; no activity is allowed.

C[1]—Configurable, using the clock-enable bits in CKEN.

C[2]—Configurable, using CCCR[PPDIS]; off if processor oscillator is off.

C[3]—Configurable, using CCCR[CPDIS]; off if processor oscillator is off.

C[4]—Configurable, using PCFR[OPDE]; on if OSCC[OOK] is clear.

C[5]—Configurable, using OSCC[OON].

St[6]—Standby state is powered by VCC_IO through the internal voltage regulator.

St[7]—Standby state is powered by VCC_BATT through the internal voltage regulator.

On[8]—Powered by VCC_IO through the internal voltage regulator.

On[9]—Powered by VCC_BATT through the internal voltage regulator.

P—Contents preserved.

NP—Contents not preserved.

† All power supplies that are being used must be enabled at all times.

†† When PWR_EN is de-asserted, optionally remove external low-voltage power domains.

†††When SYS_EN is de-asserted, optionally remove external high-voltage power domains.

Figure 3-2. Power Manager and Internal Power Domain Block Diagram

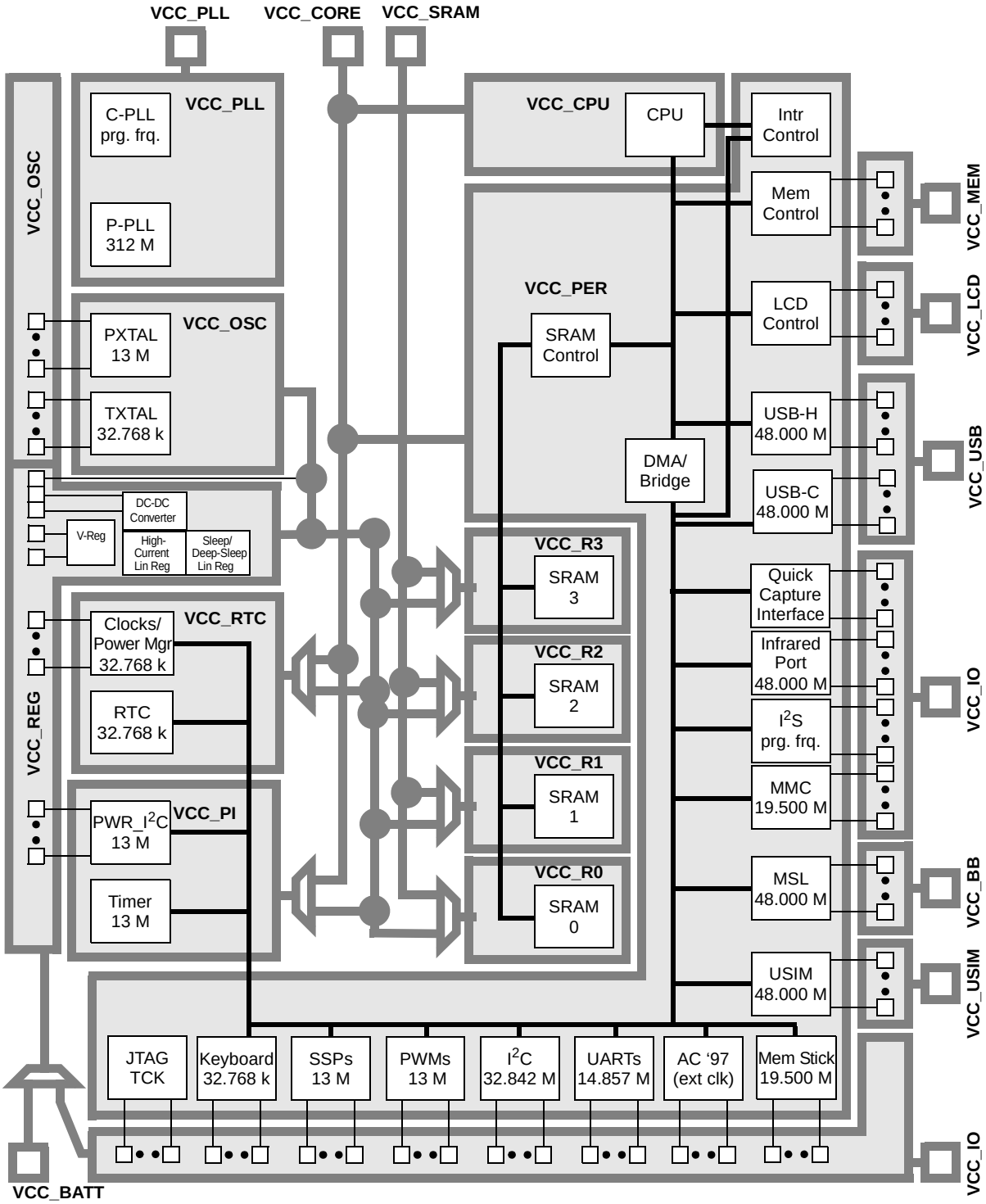
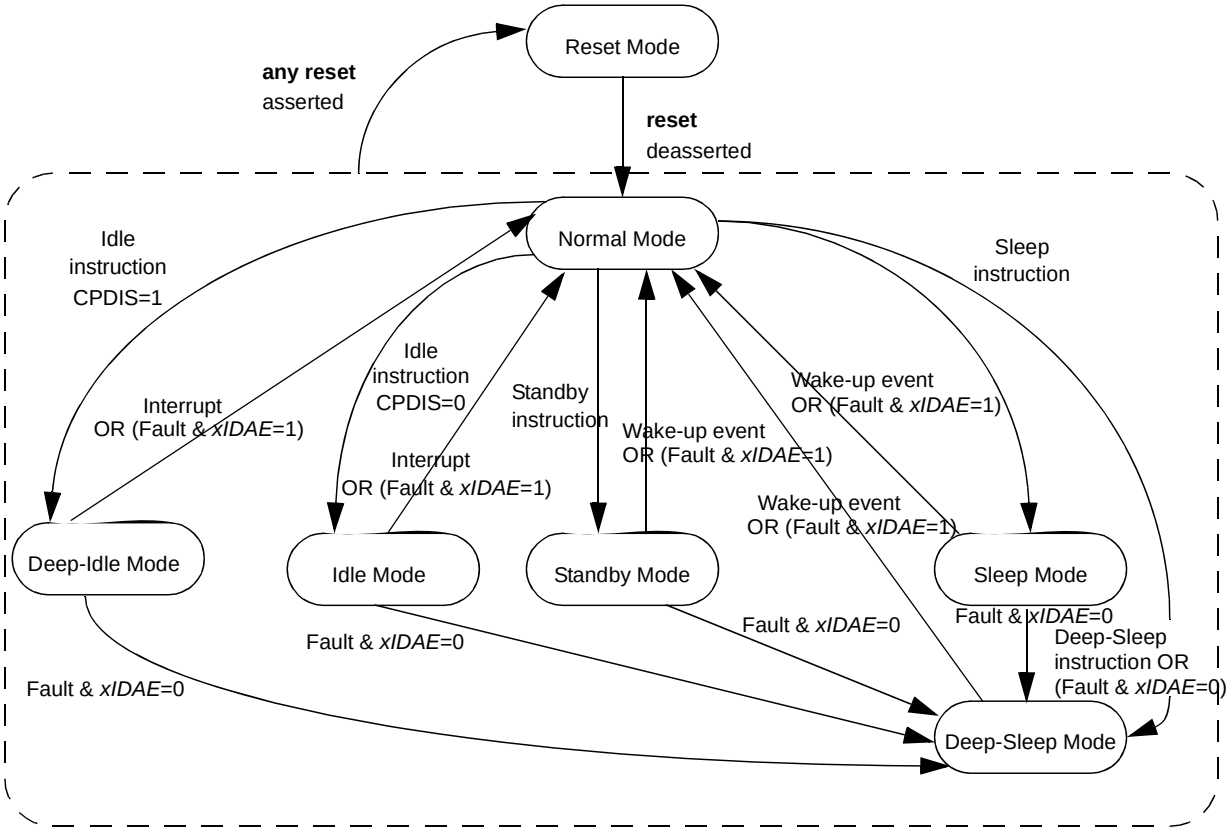


Figure 3-3. Overview of Power Manager Modes of Operation



3.6.1 Power Domains

The PXA27x processor has seven internal power domains (shown in Figure 3-2) and four I/O power supplies. All units within a power domain receive the same power and are powered on and off together. These domains are generated from one of seven externally applied power supplies, described in the *Intel® PXA27x Processor Family EMTS*.

The external power supplies create the internally-generated power domains. Refer to the *Intel® PXA27x Processor Family EMTS* for more information.

3.6.2 Internal Voltage Regulators

To achieve lowest system power in sleep and deep-sleep modes, the external power supplies can be disabled to prevent external regulator overhead power. Because the real-time clock (RTC), 32.768-kHz timekeeping oscillator, and power manager circuits must remain active, the PXA27x processor contains three internal voltage regulators, described in the sections that follow:

- High-Current Linear Regulator (Section 3.6.2.1)
- Sleep/Deep-Sleep Linear Regulator (Section 3.6.2.2)
- Sleep/Deep-Sleep DC-DC Converter (Section 3.6.2.3)

3.6.2.1 High-Current Linear Regulator

The external power supplies can be disabled in sleep and deep-sleep modes. In these modes, however, there might be many current active loads:

- The real-time clock, 32.768-kHz timekeeping oscillator, and power manager are always active in these modes.
- The 13-MHz processor oscillator, OS timer, and power I^2C units are selectively active, based on software settings.
- In sleep mode only, the internal SRAM banks can be placed in a state that retains data at the expense of some current leakage.

The high-current linear regulator is always active in any of the following cases:

- The sleep/deep-sleep linear regulator and DC-DC converter are disabled by clearing PCFR[L1_EN] and PCFR[DC_EN]. For register details, see Section 3.8.1.8
- The 13-MHz processor oscillator is enabled.
- PCFR[OPDE] is set, attempting to disable the 13-MHz oscillator, but the 32.768-kHz oscillator has not yet stabilized (OSCC[OOK] is clear) before entering the low-power mode.
- Any of the internal SRAM banks or the power manager I^2C power domains are in a state-retaining mode during sleep mode.
- The power manager I^2C power domain is in a state-retaining mode during deep-sleep mode.

3.6.2.2 Sleep/Deep-Sleep Linear Regulator

The sleep/deep-sleep linear regulator cannot supply high current. This regulator is used when all of the following conditions apply (the sequence of setting the conditions is unimportant):

- The sleep/deep-sleep DC-DC converter is disabled (PCFR[DC_EN] is clear) and the sleep/deep-sleep linear regulator is enabled (PCFR[L1_EN] is set). For register details, see Section 3.8.1.8.
- The 13-MHz processor oscillator is disabled, and the 32-kHz oscillator has stabilized (OSCC[OOK] is set) before entering the low-power mode.
- The internal SRAM banks and power manager I^2C power domain do not retain state during sleep or deep-sleep modes.
- The power manager I^2C power domain and the OS timers are inactive.

3.6.2.3 Sleep/Deep-Sleep DC-DC Converter

There are very few active loads during the lowest power sleep or deep-sleep modes. In these cases, the processor achieves best low-power efficiency when the internal DC-DC converter creates the internal supply. The DC-DC converter is used when all of the following conditions apply (the sequence of setting these conditions is unimportant):

- The sleep/deep-sleep DC-DC converter is enabled (PCFR[DC_EN] is set) and the sleep/deep-sleep linear regulator is disabled (PCFR[L1_EN] is clear). For register details, see Section 3.8.1.8.
- The 13-MHz processor oscillator is disabled and the 32-kHz oscillator has stabilized (OSCC[OOK] is set) before entering the low-power mode.

- The internal SRAM banks and power manager I^2C power domain are not in state-retaining modes during sleep or deep-sleep.
- The power manager I^2C power domain and the OS times are inactive.

The sleep/deep-sleep DC-DC converter requires the following external components:

- 0.1-µf capacitor connected between the PWR_OUT pin and ground
- 0.1-µf capacitor connected between the PWR_CAP0 and PWR_CAP1 pins
- 0.1-µf capacitor connected between the PWR_CAP2 and PWR_CAP3 pins

These capacitors must be ceramic, unpolarized capacitors with low equivalent series resistance (ESR).

Note: No other connections are allowed on the PWR_OUT and PWR_CAP<3:0> pins. Failure to adhere to this requirement can result in high currents, with associated potential for high temperature and permanent damage to the processor and the system.

3.6.3 Power Manager I^2C Interface

The PXA27x processor contains a dedicated I^2C module for communicating with an external regulator. See Chapter 9, "I^2C Bus Interface Unit" for a full description. The only differences between the power manager I^2C (PWR_I2C) and the standard I^2C interfaces are the register addresses, which are summarized in Section 9.6.

3.6.4 Power Faults and Imprecise-Data Abort

Upon assertion of nBATT_FAULT or nVDD_FAULT, the processor enters the low-power deep-sleep mode, either automatically or under the control of a fault handler. See Section 3.6.10 for details of deep-sleep mode. See Section 3.3.13 for descriptions of the fault signals.

After a power fault, the ensuing action occurs in one of three ways, depending on the settings of the corresponding BIDAE and VIDAE bits and the IAS bit in the Power Manager Control register (PMCR—see Section 3.8.1.1 for register details):

- xIDAE clear—The processor immediately enters deep-sleep mode without issuing an imprecise data abort or an interrupt to the core. All data and processor states are lost.
- xIDAE set, IAS clear—An imprecise data abort is issued to the processor core, which immediately jumps to a pre-defined interrupt vector address. The imprecise data abort handler must be in place to manage deep-sleep entry. Refer to the chapter "Handling Processor Exceptions" in the *ARM* Developer Suite Developer Guide* for more information.
- xIDAE set, IAS set—An interrupt is issued to the processor core. The interrupt handler must be in place to manage deep-sleep entry. See Chapter 25, "Interrupt Controller" for more information about the processor interrupts.

Note: When a fault occurs—regardless of the method used to manage this fault—the processor must enter deep-sleep mode. The software handler must not attempt to place the processor into any other mode when a fault signal asserts.

In the last two cases (xIDAE set), the abort handler typically preserves critical cache and other data before initiating the deep-sleep entry with a write to the PWRMODE register.

The time available for the abort handler to take other actions, such as completion of current routines, depends upon how long the external regulator system can continue to supply power after the assertion of a power fault.

In some cases, such as during sleep mode while some units are retaining states, the wake-up latency and the abort handler's execution require more time than is available while the external supply ramps down. In these cases, clear xIDAE to force an immediate deep-sleep entry.

3.6.5 Modifying Power Modes

Coprocessor 14, register C7 (PWRMODE—see Section 3.8.3.2 for register details) initiates the following power-mode changes when the corresponding bit is set:

- PWRMODE[VC] initiates a voltage-change sequence. See Section 3.7.2 for details.
- PWRMODE[M] initiates the power mode corresponding to the value written to this bit field: normal, idle, deep idle, standby, sleep, or deep sleep. Section 3.6 summarizes these power modes, and the following subsections describe them in detail.

Legal values can be written to both VC and M in the same write operation. When a write to PWRMODE occurs, the processor implements the requested power-mode change and clears PWRMODE automatically.

Note: (1) If CPSR[I] and CPSR[F] are set to mask the interrupt in the core before changing the power mode, the processor continues with the power-mode change despite any pending interrupts. In this case, if the power mode is sleep or deep sleep, and if there was a pending interrupt, the interrupt information is lost after waking from the mode. To avoid this situation, software must clear CPSR[I] and CPSR[F] before entering the power mode.

(2) Any two writes to CLKCFG or PWRMODE must be separated by six 13-MHz cycles. This requirement is achieved by reading CCCR and then comparing its value to the CLKCFG or PWRMODE register.

(3) If the memory controller is configured for synchronous external flash memory, it loses the synchronous configuration upon entry into sleep or deep-sleep mode. If PSLR[SL_ROD] is set, then nRESET_OUT does not get asserted. Thus, the external flash memory is still configured as synchronous, whereas the memory controller is configured for asynchronous operation. Software must manage this potential mismatch. See Section 6.5.2.1, "Synchronous Static Memory Configuration Register (SXCNFG)" on page 6-57 for more information.

3.6.6 Idle Mode

Idle mode allows stopping only the CPU clock during periods of processor inactivity, while continuing to monitor interrupt service requests. Idle mode is entered when PWRMODE[M] = 0b001. Generation of all other clocks remains unchanged so that, when an interrupt occurs, the CPU is quickly re-activated at the point where it entered idle mode. During idle mode, all other on-chip resources are active.

This mode is called *deep-idle mode* when CCCR[CPDIS] is set and PWRMODE[M] = 0b001.

3.6.6.1 Preparation for Idle Mode

Before entering idle mode, enable any interrupts to be used as wake-ups from idle mode.

Idle mode does not stop the bus clocks and does not alter any of the peripheral clocks, including the memory and LCD controllers. Thus, no additional steps are required (with respect to the peripheral clocks) before entering idle mode.

3.6.6.2 Entering Idle Mode

To enter idle mode, first complete the preparations listed in Section 3.6.6.1. Then, write the appropriate value to the M bit field in the PWRMODE register (see Section 3.8.3.2). When the write to PWRMODE occurs, the following steps occur in order:

1. All processor activity is stopped. All interrupt requests to the CPU core are held.
2. All CPU loads are completed. CPU stores are sent to the system bus.
3. The CPU clock is halted.
4. Interrupts are no longer held and are recognized as wake-up sources from idle mode.

3.6.6.3 Behavior During Idle Mode

During idle mode, all peripherals and system resources are fully operational, except that the CPU clock is stopped.

The only difference from normal peripheral operation is that any enabled interrupt can awaken the processor from idle mode, regardless of the state of ICMR (see Section 25.5.4, “Interrupt Controller Mask Registers (ICMR and ICMR2)” on page 25-19).

If normal interrupt masking is required, disable this feature by setting the disable idle mask bit, ICCR[DIM] (see Section 25.5.6, “Interrupt Controller Control Register (ICCR)” on page 25-27).

If ICCR[DIM] is clear and idle-mode wake-ups from a specific unit are not wanted, the unit’s interrupt must be disabled at the unit level.

Note: (1) Additional interrupt latency is caused by holding interrupt requests while the CPU loads are completed and CPU stores are sent to the system bus. This latency varies with the number, source, and destination of loads and stores, as well as with other bus activity.

(2) The watchdog timer, if enabled, is functional during idle mode and generates a watchdog reset if OSMR register 3 matches the OS timer counter.

3.6.6.4 Exiting Idle Mode

Idle mode ends with the assertion of either of the following idle-mode wake-up events:

- Any enabled interrupt, regardless of the state of ICMR.
- Assertion of nBATT_FAULT or nVDD_FAULT (see Section 3.6.4 for details of the ensuing action).

Following the assertion of an idle-mode wake-up event, the following occurs:

1. The CPU clock is restarted.
2. The processor continues execution at the next instruction after the write to PWRMODE or at the entry point to the imprecise-data abort or interrupt handler.

Note: Any two writes to the CLKCFG or PWRMODE registers must be separated by at least six 13-MHz cycles. To meet this requirement, read CCCR and then compare its value to the CLKCFG or PWRMODE register.

3.6.7 Deep-Idle Mode

Deep-idle mode is a combination of 13M mode and the processor idle feature. Deep-idle mode is a transition into idle mode from 13M mode. Refer to Section 3.6.6 for idle mode and Section 3.5.7.7 for 13M mode.

Refer to the *Intel® PXA27x Processor Family EMTS* for appropriate VCC_CORE voltage setting in deep-idle mode.

When CCCR[CPDIS] is set and PWRMODE[M] = 0b001, this mode is referred to as deep-idle.

3.6.8 Standby Mode

Standby mode is a low-power mode in which power consumption is reduced below the normal static power consumption while the processor retains state. All processor activity stops, except for the real-time clock (RTC) and the clocks and power manager. Since internal activity has stopped, recovery from standby mode must be through an external or RTC event. At recovery, execution resumes at the instruction following the write to the PWRMODE register (see Section 3.8.3.2).

Unit state retention during standby mode is as follows (see the block diagram in Figure 3-2):

- The following units always retain state:
  - CPU powered by VCC_CPU (internal domain)
  - Peripheral units powered by VCC_PER (internal domain)
- The following units may optionally retain state:
  - Internal SRAM banks, powered by VCC_SRAM or VCC_REG (internal domain) (depending on low-power mode)
  - Power manager I^2C unit and 13-MHz timer, powered by VCC_PI (internal domain)
- The PLLs are automatically disabled in standby mode.

Note: Any two writes to the CLKCFG or PWRMODE registers must be separated by six 13-MHz cycles. To meet this requirement, read of CCCR and then compare its value to the CLKCFG or POWERMODE register.

Refer to the *Intel® PXA27x Processor Family EMTS* for appropriate VCC_CORE voltage setting.

3.6.8.1 Preparation for Standby Mode

Before entering standby mode, complete the following steps:

1. Set the standby-mode unit-retention bits in the Standby Configuration register (PSTR) for any units that must retain state during standby mode (see Section 3.8.1.12 for register details).
2. Configure the memory controller to ensure that SDRAM contents are maintained during standby mode (all required boot sequences must be complete). See Chapter 6, “Memory Controller” for details.

3. Stop or disable all peripheral units except the RTC and, optionally, the OS timer. Also, if the keypad is configured as the wake-up source, do not disable the keypad controller.

   The other peripherals do not function normally because the clocks are stopped. This includes the LCD controller; thus, unless the external LCD can sustain operation without constant pixel information, disable the external LCD and the LCD controller.

4. Program the following registers to enable the standby-mode wake-up sources:

   Section 3.8.1.4 — Power Manager Wake-Up Enable Register (PWER)

   Section 3.8.1.15 — Power Manager Keyboard Wake-Up Enable Register (PKWR)

   Section 3.8.1.5 — Power Manager Rising-Edge Detect Enable Register (PRER)

   Section 3.8.1.6 — Power Manager Falling-Edge Detect Enable Register (PFER)

5. **If low-power operation is required**, set PCFR[OPDE] to disable the 13-MHz processor oscillator (see Section 3.8.1.8 for register details). In this case, wait until PCFR[OOK] is set before entering standby mode.

   If fast wake-up is required, clear PCFR[OPDE] to keep the oscillator on during standby.

6. After exiting standby mode, the code/data fetches by the core must not resume from the memory space that cannot be accessed due the PSSR[PH] bit being set by hardware.

Note: The GPIO block is not reset in standby mode. Hence, the GPIO alternate functions are restored automatically after standby mode to their states immediately preceding standby mode.

3.6.8.2 Entering Standby Mode

To enter standby mode, first complete the preparations listed in Section 3.6.8.1. Then, write the appropriate value to the M-bit field in the PWRMODE register (see Section 3.8.3.2). The following steps occur when the write to PWRMODE occurs:

1. All processor activity is stopped. All interrupt requests to the processor are held.
2. All CPU loads are completed. All CPU stores are sent to the system bus.
3. The CPU clock is halted.
4. All new transactions from the USB host, LCD controller, and DMA controller are ignored.
5. The memory controller completes all outstanding transactions in its buffers.
6. The memory controller places the SDRAM in self-refresh mode and drives the nRAS/nSDCS<3:0> and nCAS/DQM<3:0> pins to their self-refresh states.
7. All PLLs are disabled.
8. If PCFR[OPDE] and PCFR[OOK] are set, the 13-MHz processor oscillator is disabled.
9. The circuits that activate the low-current mode are energized for the following units:
   - units in the VCC_CPU and VCC_PER power domains (see the block diagram in Figure 3-2)
   - units selected by the PSTR[standby mode unit retention] bits.
10. Units not selected by the PSTR[standby mode unit retention] bits are powered off.

3.6.8.3 Behavior in Standby Mode

In standby mode, all clocks are disabled except those for the power manager and the RTC. No interrupts are recognized. No external pin transitions are recognized other than valid wake-up signals, reset signals, nBATT_FAULT, and nVDD_FAULT.

The power manager watches for wake-up events that were programmed prior to entering standby mode (see Section 3.6.8.1 for more information about wake-ups). Refer to the *Intel® PXA27x Processor Family EMTS* for GPIO timing specifications.

Deep-sleep entry due to a power fault (nVDD_FAULT or nBATT_FAULT asserted) normally occurs as described in Section 3.6.4. But in standby mode, with the corresponding PMCR[xIDAE] bit set, the imprecise data abort or interrupt is not issued immediately. Instead, the power-fault event appears to the power manager first as a standby-mode wake-up. The power-fault abort is sent to the processor core only after exit from standby mode is complete. Thus, there is additional latency between the assertion of a power fault and its recognition.

If this additional latency is unacceptable, clear the corresponding PMCR[xIDAE] bit. In this case, entry into deep-sleep mode occurs immediately, but controlled entry using software is not possible.

3.6.8.4 Exiting Standby Mode

The following occurs after the assertion of a pre-programmed standby-mode wake-up event or the assertion of nBATT_FAULT or nVDD_FAULT with the corresponding PMCR[xIDAE] bit set:

1. If standby mode was entered with PCFR[OPDE] set, the 13-MHz processor oscillator gets re-enabled and allowed to stabilize.

   If PCFR[OPDE] is clear, the 13-MHz processor oscillator is already enabled and has stabilized.
2. If any of the standby-mode unit-retention bits are clear, power is restored to the selected unit.
3. The PLLs are restarted according to the corresponding values in the Core Clock Configuration register and allowed to stabilize.
4. The CPU clock is restarted. Interrupts are no longer held.
5. The processor resumes execution at the next instruction after the write to PWRMODE or at the entry point to the imprecise data abort or interrupt handler.
6. The standby-mode configuration is automatically cleared in the PWRMODE register.
7. The SDRAM must be brought out of self-refresh mode, which requires that the SDRAM controller be switched to its idle state. See Chapter 6, “Memory Controller” for details on configuring the SDRAM interface.

If nBATT_FAULT or nVDD_FAULT is asserted:

- If the corresponding PMCR[xIDAE] bit is set, the regular standby-mode exit sequence occurs. The abort handler can then enter deep-sleep mode under controlled conditions, allowing software to save critical data. See Section 3.6.4 for more information about the abort handler.
- If the corresponding PMCR[xIDAE] bit is clear, the processor exits standby mode without performing steps 1-7 and immediately enters deep-sleep mode.

3.6.9 Sleep Mode

Sleep mode offers even lower power consumption by powering off most units. The increased latency for this low-power mode is that all states are lost. There is no activity inside the processor, except for the units programmed to retain their state in the PSLR register, the real-time clock, and the clocks and power manager. Because internal activity has stopped, recovery from sleep mode must occur through an external or a real-time clock event. All processor states are reset, and recovery begins with the required boot sequence (see Section 3.4 for information about boot sequences).

In sleep mode, the external low-voltage power domains can be disabled externally to further lower the system power consumption. See Figure 3-2 for information on the power domains.

3.6.9.1 Preparation for Sleep Mode

Follow these steps before entering sleep mode:

1. For units that must retain their states during sleep, set the appropriate sleep-mode unit-retention bits in the Sleep Mode Configuration register (PSLR—see Section 3.8.1.11). The units that can retain state are the internal SRAM banks and the PI power domain (timer and power I^2C). The PLLs are disabled automatically.
2. Program PSLR[SYS_DEL] and PSLR[PWR_DEL] for the number of 32.768-kHz timekeeping oscillator cycles required for the external power supplies to stabilize (the default reset value is 125-ms delay for each).
3. For lowest power consumption, enable the sleep/deep-sleep DC-DC converter (see Section 3.6.2.3) by setting PCFR[DC_EN].

   If PCFR[DC_EN] is clear and PCFR[L1_EN] is set, the sleep/deep-sleep linear regulator is enabled. If both PCFR[DC_EN] and PCFR[L1_EN] are clear, the high-current linear regulator is enabled.

Note: Do **not** set both PCFR[DC_EN] and PCFR[L1_EN] at the same time.

4. Disable the LCD controller, unless the external LCD panel has a built-in frame buffer and can operate while the pixel clock is stopped. See Chapter 7, “LCD Controller” for more information.
5. Configure the appropriate power manager registers for the sleep-mode wake-up sources:

   Section 3.8.1.4 — Power Manager Wake-Up Enable Register (PWER)

   Section 3.8.1.15 — Power Manager Keyboard Wake-Up Enable Register (PKWR)

   Section 3.8.1.5 — Power Manager Rising-Edge Detect Enable Register (PRER)

   Section 3.8.1.6 — Power Manager Falling-Edge Detect Enable Register (PFER)

   Section 3.8.1.9 — Power Manager GPIO Sleep-State Registers (PGSRx)

Note: As the PGSRx registers get loaded onto the processor GPIO outputs, initialize these to the correct state. For example, pins related to the static and synchronous-memory chip selects need to be de-asserted, and PC Card control pins must be either de-asserted or floated.

6. **For lowest-power operation**, set PCFR[OPDE] to disable the 13-MHz processor oscillator (see Section 3.8.1.8 for register details). In this case, wait until PCFR[OOK] is set before entering sleep mode.

   For fastest wake-up, clear PCFR[OPDE] to keep the oscillator running during sleep mode.
7. Prepare for possible power faults (assertion of nVDD_FAULT or nBATT_FAULT and subsequent deep-sleep entry) as described in Section 3.6.4.

3.6.9.2 Entering Sleep Mode

Note: The GPIO block is reset in sleep mode. Therefore, the GPIO alternate functions must be re-programmed after sleep exit.

Entry into sleep mode occurs when the sleep configuration is written to the M-bits in the PWRMODE register (see Section 3.8.3.2).

The following sequence occurs when the sleep configuration is written to PWRMODE:

1. All processor activity stop and all interrupt requests to the processor are ignored.
2. All CPU loads are completed and CPU stores are sent to the system bus.
3. The CPU clock is halted.
4. All new transactions from the USB host controller, LCD controller, and DMA controller are ignored.
5. The memory controller completes all outstanding transactions in its buffers.
6. The memory controller places the SDRAM in self-refresh mode and drives the nRAS/nSDCS<3:0> and nCAS/DQM<3:0> pins to their self-refresh state.
7. The PLL clock sources and their outputs are disabled.
8. The power manager switches the GPIO output pins to the sleep states programmed in the PGSR registers.
9. If PCFR[OPDE] and OSCC[OOK] are set, the 13-MHz processor oscillator is disabled.
10. An internal reset (reflected externally by the assertion of the nRESET_OUT pin if PSLR[SL_ROD] is clear) is generated to the CPU, to peripheral logic powered by VCC_PER (internal domain), and to all units not selected by the sleep-mode unit-retention bits.
11. The low current state retention circuitry is enabled for the units that are selected by the sleep-mode unit-retention bits.
12. The units not selected by the sleep-mode unit-retention bits are powered off.
13. The power supply to the clocks and power manager, RTC, and any units selected by the sleep-mode unit-retention bits is switched from VCC_CORE to VCC_OSC (internal domain). The internal SRAM banks selected by the sleep-mode unit-retention bits are switched from VCC_SRAM to VCC_OSC (internal domain).
14. The PWR_EN pin is de-asserted. Optionally, disable the external low-voltage power domains to minimize power consumption.

3.6.9.3 Behavior in Sleep Mode

In sleep mode, all clocks are disabled to the processor and to all peripherals except the RTC. However, if the keypad is configured to be the wake-up source, do not disable the keypad controller. No interrupts are recognized, and no external pin transitions other than valid wake-up signals, reset signals, and the power fault (nVDD_FAULT and nBATT_FAULT) signals are recognized. The nVDD_FAULT pin is ignored if PSLR[IVF] is set.

The power manager watches for pre-programmed wake-up events that the CPU configures prior to entering sleep mode. Refer to the *Intel® PXA27x Processor Family EMTS* for GPIO timing specifications.

The processor does not recognize the imprecise data abort or interrupt during sleep mode. Therefore, if the corresponding PMCR[xIDAE] bit is set, the assertion of nVDD_FAULT or nBATT_FAULT appears to the processor as a wake-up event from sleep mode. Once the processor has exited sleep mode, an imprecise data abort or interrupt is reported to the core. See Section 3.6.4 for more information on these power faults and their management.

If nVDD_FAULT or nBATT_FAULT is asserted and if the corresponding PMCR[xIDAE] bit is set, the processor does not issue the imprecise data abort or interrupt until sleep-mode exit completes. If this additional latency is unacceptable, the corresponding PMCR[xIDAE] bit must be cleared before entering sleep mode. In this case, however, assertion of the power fault results in the immediate loss of all processor states, with no software-controlled entry into deep-sleep mode.

3.6.9.4 Sleep Exit

The following occurs after the assertion of a pre-programmed sleep-mode wake-up event while the nVDD_FAULT and nBATT_FAULT pins are not asserted (nVDD_FAULT is ignored if PSLR[IVF] is set):

1. PWR_EN is asserted, enabling the external low-voltage power domains.
2. The processor waits the number of 32.768-kHz timekeeping oscillator cycles specified by the PSLR[PWR_DEL] bits.
3. The power supply to the clocks and power manager, RTC, and any units selected by the sleep-mode unit-retention bits in PSLR is switched from VCC_OSC to VCC_CORE or from VCC_OSC to VCC_SRAM.
4. If sleep mode was entered with PCFR[OPDE] set, the 13-MHz processor oscillator is enabled and allowed to stabilize.
5. If any of the sleep-mode unit-retention bits are clear, power to the selected units is restored.
6. The PLLs are restarted with the corresponding values in the Core Clock Configuration register and allowed to stabilize.
7. The sleep-mode configuration in the PWRMODE register is cleared.
8. The nRESET_OUT pin, if asserted, is de-asserted, indicating that the processor is about to perform a fetch from the reset-vector location. The processor internal reset is de-asserted.
9. The CPU begins the required boot sequence (see Section 3.4 for information about boot sequences), which includes the following items:
    a. Software must bring the SDRAM out of self-refresh mode, which requires that the SDRAM controller be switched to its idle state. See Chapter 6, “Memory Controller” for details on configuring the SDRAM interface.
    b. All units in the PXA27x processor, except those listed in Table 3-2, begin with their predefined reset conditions.
    c. Software must examine the Reset Controller Status register (RCSR[SMR]) to determine that the reset source was a sleep-exit reset and the Sleep Status register (PSSR[SSS]) to determine the reason for being in sleep mode.
    d. If the Scratch Pad register (PSPR) was used for saving any general processor state during sleep mode, the state can be recovered.

Note: If sleep mode was entered while the processor was in turbo, half-turbo or fast-bus mode, the sleep-mode exit returns the processor to normal run mode. If sleep mode was entered while in 13M mode, the sleep-mode exit returns the processor to 13M mode.

3.6.10 Deep-Sleep Mode

Deep-sleep mode offers the lowest power consumption by powering most units off. The increased latency for this low-power mode is that all state is lost and there is no activity inside the processor, except for the real-time clock (RTC) and the clocks and power manager. Because internal activity has stopped, recovery from deep-sleep mode must be through an external event or an RTC event. Because all state has been lost, the state of the processor is reset, and recovery begins with the required boot sequence (see Section 3.4 for information about boot sequences).

In deep-sleep mode, all the power supplies (VCC_CORE, VCC_SRAM, VCC_PLL, VCC_IO, VCC_LCD, VCC_USIM, VCC_USB, VCC_BB, and VCC_MEM) excluding VCC_BATT can be powered off for minimized power consumption.

If deep-sleep mode was entered with a software write to the PWRMODE register, deep-sleep can be exited by correctly programming the PWER, PFER and PRER registers for the RTC wake-up event and a wake-up event on GPIO<3>, GPIO<1>, or GPIO<0>. If deep-sleep mode was entered due to a power fault (assertion of nBATT_FAULT or nVDD_FAULT), then exit from deep-sleep is limited to a wake-up event on GPIO<1> or GPIO<0>. The PWER, PFER, and PRER registers are automatically forced to their reset values in this case.

3.6.10.1 Preparation for Deep-Sleep Mode

Complete the following steps before entering deep-sleep mode:

1. For any units that must retain state during deep-sleep mode, set the sleep-mode unit-retention bits in the Sleep Mode Configuration register (see Section 3.8.1.11). Only the units in the PI power domain (RTC and power manager I^2C) can retain state. The PLLs are disabled automatically.
2. Program the SYS_DEL and PWR_DEL bits in PSLR for the number of 32.768-kHz timekeeping oscillator cycles required to stabilize the external power supplies (the reset value is 125ms delay each).
3. For lowest power consumption, enable the sleep/deep-sleep DC-DC converter (see Section 3.6.2.3) by setting PCFR[DC_EN].

   If PCFR[DC_EN] is clear and PCFR[L1_EN] is set, the sleep/deep-sleep linear regulator is enabled. Otherwise, the high-current linear regulator is enabled.

Note: Do **not** set PCFR[DC_EN] and PCFR[L1_EN] at the same time.

4. The memory controller sends the self-refresh command to the SDRAM banks. Ensure that the supply to the SDRAM is not removed if state retention is required during deep-sleep mode. See Chapter 6, “Memory Controller” for more details.
5. Disable the LCD controller. LCD operation during deep-sleep mode is possible only with an external LCD panel that has a built-in frame buffer.
6. Initialize the appropriate power manager registers to determine the deep-sleep wake-up sources:
   - Power Manager Wake-Up Enable register (PWER)
   - Power Manager Falling-Edge Detect Enable and Power Manager Rising-Edge Detect Enable registers (PFER and PRER)
   - Power Manager GPIO Sleep-State registers (PGSR0, PGSR1, PGSR2 and PGSR3).

Note: Because the PGSRx registers get loaded onto the GPIO outputs, be careful to initialize these to the correct states—for example, pins related to the static and synchronous memory chip selects, PC Card control, and so forth must be floated. When SYS_EN is de-asserted and the power domains are powered off, all GPIOs float. However, if software is configured to maintain power during deep-sleep mode, the GPIOs default to the pull-up, pull-down reset state as indicated in the *Intel® PXA27x Processor Family EMTS* Pin Usage section for reset states and the GPIO AC Timing Specification section for timing details.

7. **For lowest-power operation**, set PCFR[OPDE] to disable the 13-MHz processor oscillator (see Section 3.8.1.8 for register details). In this case, wait until PCFR[OOK] is set before entering sleep mode.
8. **For fastest wake-up**, clear PCFR[OPDE] to keep the oscillator running during sleep mode.

3.6.10.2 Entering Deep-Sleep Mode

Entry into deep-sleep mode occurs at any of the following deep-sleep entry events:

- The deep-sleep configuration is written to the mode bits in the PWRMODE register (see Section 3.8.3.2).
- The pin nBATT_FAULT or nVDD_FAULT is asserted while the corresponding xIDAE bit is clear.
- nBATT_FAULT or nVDD_FAULT is asserted while exiting from deep-sleep mode.

The following sequence occurs when the deep-sleep configuration is written:

1. All processor activity is stopped and all interrupt requests to the processor are ignored.
2. All CPU loads are completed, and CPU stores are sent to the system bus.
3. The CPU clock is halted.

The sequence begins here if nBATT_FAULT or nVDD_FAULT is asserted while the corresponding xIDAE bit is clear:

4. All new transactions from the LCD controller or DMA controller are ignored.
5. The memory controller completes all outstanding transactions in its buffers.
6. The memory controller places the SDRAM in self-refresh mode and drives the nRAS/nSDCS<3:0> and nCAS/DQM<3:0> pins to their self-refresh state.

If nVDD_FAULT or nBATT_FAULT is asserted during the sleep or deep-sleep exit sequence, deep-sleep mode is re-entered here:

7. The power manager switches the GPIO output pins to the sleep state programmed in registers PGSR0-3.
8. If the deep-sleep sequence was entered because of the assertion of nVDD_FAULT or nBATT_FAULT, regardless of the state of the corresponding xIDAE bit, the following actions occur:
   a. All wake-ups detected at this point are cleared (all GPIO edge-detects and the RTC alarm interrupt).
   b. The power manager wake-up source registers (PWER, PRER and PFER) are loaded with the value 0x0000_0003, the reset state after a hardware reset, which limits the potential wake-up sources to a rising or falling edge on GPIO<0> or GPIO<1>. This wake-up fault state prevents spurious events from causing an unwanted wake-up when a problem develops with the main battery or a power supply.

9. The PLL clock sources and their outputs are disabled.
10. If PCFR[OPDE] and OSCC[OOK] are set, the 13-MHz processor oscillator is disabled.
11. An internal reset (reflected externally by the assertion of the nRESET_OUT pin if PSLR[SL_ROD] is clear) is generated to the CPU, to peripheral logic powered by VCC_PER, and to all units not selected by the sleep mode unit retention bits.
12. The low-current state retention circuitry is enabled for the units that are selected by the sleep-mode unit-retention bits.
13. The units not selected by the sleep-mode unit-retention bits are powered off.
14. The power supply (VCC_REG) for the regulator that generates VCC_OSC is switched from VCC_IO to VCC_BATT.
15. The power supply to the clocks and power manager, RTC, and any units selected by the sleep-mode unit-retention bits is switched from VCC_CORE to VCC_OSC.
16. The PWR_EN pin is de-asserted. Disable the external low-voltage power domains to minimize power consumption.
17. The SYS_EN pin is de-asserted. Disable the external high-voltage power domains to minimize power consumption. If any of these power supplies is disabled, then all of the external low-voltage power domains must also be disabled.

3.6.10.3 Behavior in Deep-Sleep Mode

In deep-sleep mode, all clocks to the processor and to all peripherals (except the RTC) are disabled. Therefore, no interrupts are recognized, and no external pin transitions other than valid wake-up signals, reset signals, and the nBATT_FAULT signal are recognized. The nVDD_FAULT pin is ignored until the appropriate point in the deep-sleep wake-up sequence. The nVDD_FAULT pin is ignored if PSLR[IVF] is set.

The power manager watches for wake-up events that were programmed prior to entering deep-sleep mode. Refer to the *Intel® PXA27x Processor Family EMTS* for GPIO timing specifications.

In deep-sleep mode, the external GPIO wake-up sources are limited to GPIO<3:0>, unless the deep-sleep entry was caused by nBATT_FAULT or nVDD_FAULT assertion. In this case, the wake-ups are limited to GPIO<1:0>.

The imprecise data abort or interrupt is not recognized in deep-sleep mode. If nBATT_FAULT is asserted during deep-sleep mode, the processor remains in deep-sleep mode.

3.6.10.4 Exiting Deep-Sleep Mode

The following occurs after the assertion of a pre-programmed deep-sleep mode wake-up event while the nBATT_FAULT pin is not asserted:

1. SYS_EN is asserted, enabling the external high-voltage power domains.
2. The processor waits the number of 32.768-kHz timekeeping oscillator cycles specified by PSLR[SYS_DEL].

   If PSLR[PSSD] is set, the processor shortens the wake-up sequence by asserting PWR_EN as soon as all of the high-voltage power supplies signal that they are powered on.
3. The PWR_EN signal is asserted, enabling the external low-voltage power domains.
4. The processor waits the number of 32.768-kHz timekeeping oscillator cycles specified by PSLR[PWR_DEL].

If PSLR[PSSD] is set, the processor shortens the wake-up sequence by cutting short the PSLR[PWR_DEL] counting as soon as all of the low-voltage power supplies signal that they are powered on.

Beyond this point, if nVDD_FAULT is asserted, the processor switches back into deep-sleep mode.

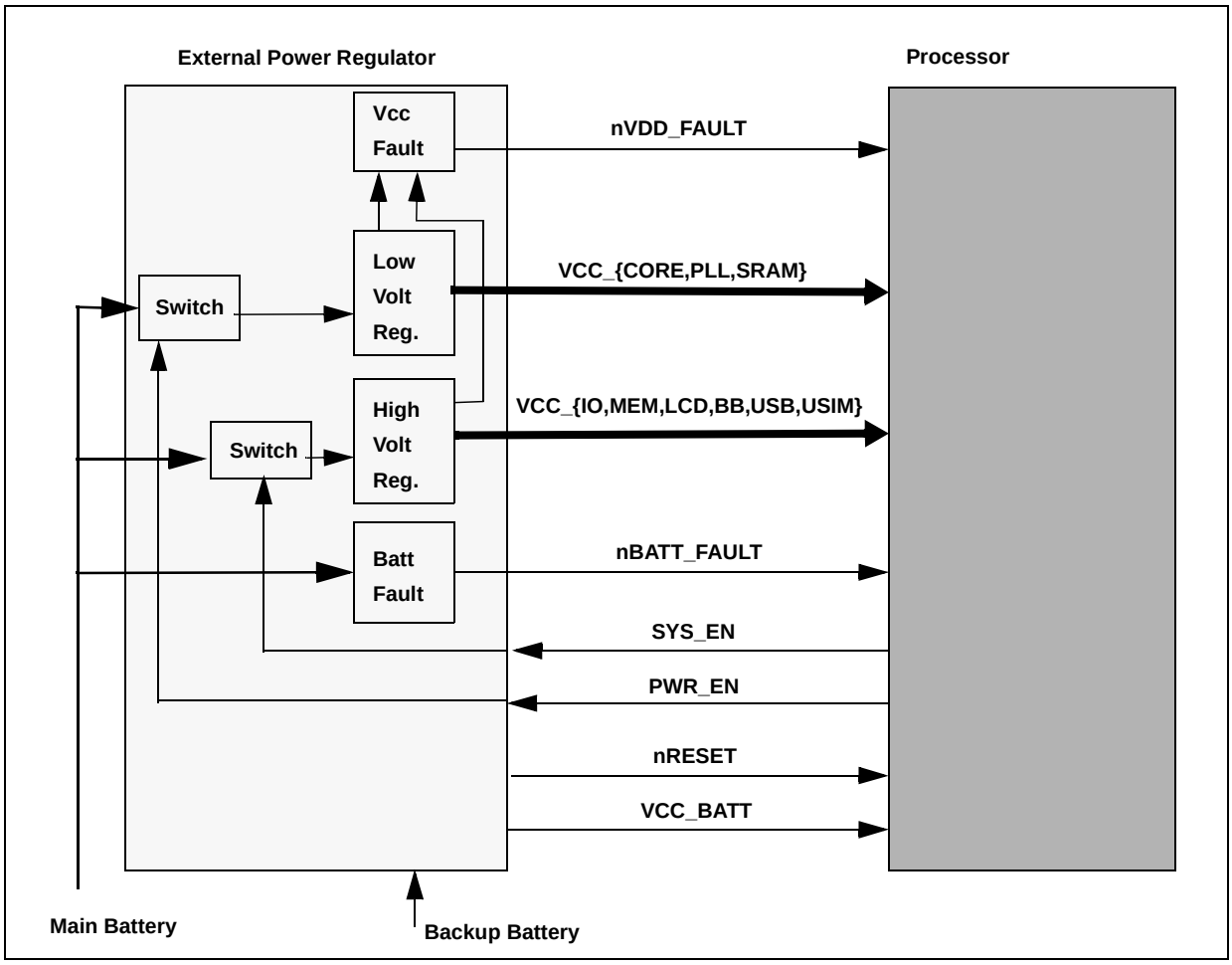
5. The power supply to the clocks and power manager, the RTC, and any units selected by the PSLR[sleep-mode unit-retention] bits is switched from VCC_OSC to VCC_CORE.
6. The power supply (VCC_REG) to the regulator that generates VCC_OSC is switched from VCC_BATT to VCC_IO.
7. If deep-sleep mode was entered with PCFR[OPDE] set, the 13-MHz processor oscillator is enabled and allowed to stabilize.
8. If any of the PSLR[sleep-mode unit-retention] bits are clear, power to the selected unit is restored. If any of the PSLR[sleep-mode unit-retention] bits are set, the low-current state-retention circuitry is disabled, and power to the selected unit is restored.
9. The PLLs are reprogrammed with the corresponding values in the Core Clock Configuration register and allowed to stabilize.
10. The deep-sleep configuration in the PWRMODE register is cleared.
11. If it is asserted, the nRESET_OUT pin is de-asserted, indicating that the processor is about to perform a fetch from the reset-vector location. The processor's internal reset is de-asserted.
12. The CPU begins the required boot sequence (see Section 3.4 for information about boot sequences), which includes the following items:
    a. Software must bring the SDRAM out of self-refresh mode, which requires that the SDRAM controller be switched to its idle state. See Chapter 6, "Memory Controller" for details on configuring the SDRAM interface.
    b. All processor units, except those listed in Table 3-2, begin with their predefined reset states.
    c. Software must examine the Reset Controller Status register (RCSR[SMR]) to determine that the reset source was a sleep-exit reset from deep-sleep mode and the Sleep Status register (PSSR) to determine the reason for being in deep-sleep mode.
    d. If the Scratch Pad register (PSPR) was used for saving any general processor states during deep-sleep mode, the states can be recovered.

Note: If deep-sleep mode was entered while the processor was in turbo, half-turbo or fast-bus mode, the deep-sleep exit returns the processor to normal run mode. If deep-sleep mode was entered while the processor was in 13M mode, the deep-sleep exit returns the processor to13M mode.

3.6.11 Initial Power-On and Deep-Sleep Exit Sequence

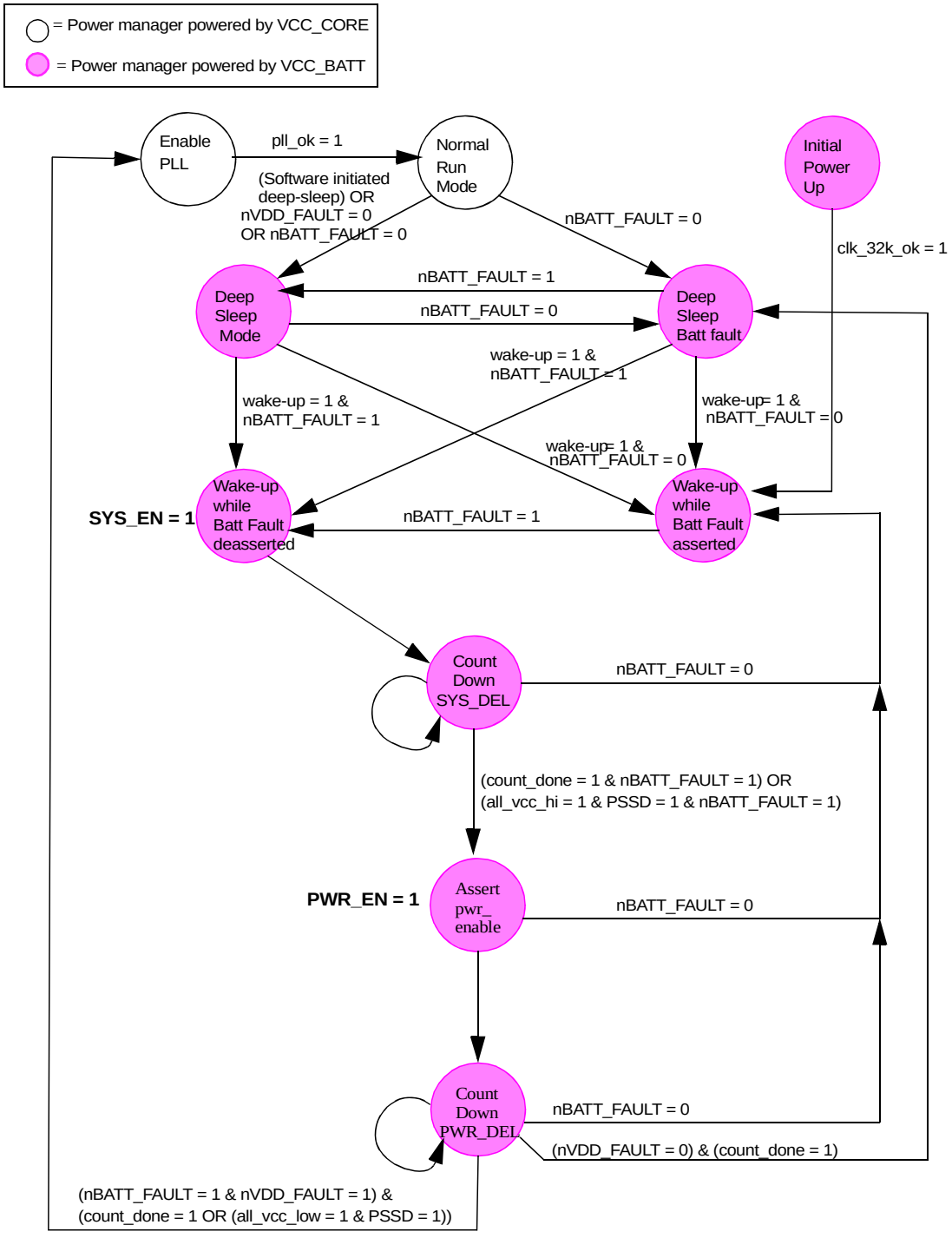
As shown in Figure 3-4, the external voltage regulator supplies the high-voltage and low-voltage power supplies to the processor. The external voltage regulator also sources the nBATT_FAULT and nVDD_FAULT signals to the processor. The processor's SYS_EN and PWR_EN signals control the high (VCC_IO, VCC_MEM, VCC_LCD, VCC_BB, VCC_USB, and VCC_USIM) and low (VCC_CORE, VCC_SRAM, VCC_PLL) voltages, respectively.

Figure 3-4. Typical System Diagram



The state diagram in Figure 3-5 shows typical steps taken by the power manager while initially powering up and while exiting from deep-sleep mode. Based on the wake-up events and states of the nBATT_FAULT and nVDD_FAULT signals, the processor exits from deep-sleep mode and enters the normal power mode. Refer to the *Intel® PXA27x Processor Family EMTS* for timing information on the initial power-on sequence and the deep-sleep exit sequence.

Figure 3-5. Initial Power-On and Deep-Sleep Exit States



3.6.12 Summary of Power Modes

The power modes follow the entry and exit sequences shown in Table 3-12. The total latency of the entry into each sequence is the sum of the latencies at each step. The remaining latency visible to software depends on latency in the boot up or interrupt service routine.

Table 3-12. Summary of Power and Clock Mode Sequences (Sheet 1 of 2)

Description of Action	Unit	Latency (cycles)	Idle	Standby	Sleep	Deep Sleep
Write to CP14 PWRMODE register (software); interrupts gated off.	CPU	1 CPU	x	x	x	x
All current instructions, including incomplete fetches, completed.	CPU	? CPU	x	x	x	x
All outstanding stores completed.	CPU	? SB	x	x	x	x
Wait for synchronization. Halt CPU clock.	CPM	< 20 CPU	x	x	x	x
Entry point for sleep/deep-sleep entry, xIDAE bits clear						
Deny all bus requests (from LCD, USB-H, DMA, CPU, or memory controller).	CPM	1 SB		x	x	x
Complete all memory controller transactions (allow memory controller requests). Place SDRAM in self-refresh mode.	MEM	? SB		x	x	x
Switch GPIO output pins to sleep state in PGSR registers.	CPM	1 SB			x	x
Re-Entry point for sleep/deep-sleep entry, Fault pin asserted during exit sequence						
Clear all wake-up sources; set PWER, PRER, PFER to 0x0000 0003 if entered from FAULT pin.	CPM	1 SB			x	x
Synchronize clocks and power manager. Switch clock sources if needed.	CPM	<4 13M <8 SB		x	x	x
Disable PLLs if appropriate (OPDE set, xPDIS set).	CPM	2 13M		x	x	x
Switch clocks and power manager from 13M to 32k clock if OPDE set and disable 13M oscillator.	CPM	3 32k		x	x	x
Assert reset to internal units if appropriate. Assert nRESET_OUT.	CPM	1 32k		x	x x	x x
Power off/standby selected units according to the mode's retention units if any.	CPM	2 32k		x	x	x
Deassert PWR_EN.	CPM	2 32k			x	x
Deassert SYS_EN.	CPM	2 32k				x
Power/clock mode entry sequence completed; waiting for external wake-up event						
Enable wake-up events/interrupts to CPU.	CPM	1 32k	x	x	x	x
Wake-up/interrupt received in clocks and power manager.	CPM	—	x	x	x	x
External wake-up synchronized to clocks and power manager clock.	CPM	2 32k		x	x	x
Assert SYS_EN.	CPM	1 32k				x
Wait for ramp time set by SYS_DEL.	CPM	? 32k				x
Assert PWR_EN.	CPM	1 32k			x	x
Wait for ramp time set by PWR_DEL.	CPM	? 32k			x	x
The 13-MHz processor oscillator stabilizes if OPDE is set.	CPM	64k* 13M		x	x	x
Switch CPM to 13-MHz clock if OPDE set.	CPM	2* 13M		x	x	x
Power on internal units.	CPM	? 13M		x	x	x
Enable PLLs if appropriate (xPDIS clear) and wait for PLL to lock.	CPM	1290 * 13M		x	x	x
Synchronize clocks and power manager.	CPM	<2 13M <4 SB		x	x	x

Table 3-12. Summary of Power and Clock Mode Sequences (Sheet 2 of 2)

Description of Action	Unit	Latency (cycles)	Idle	Standby	Sleep	Deep Sleep
Enable I/O, clocks to all units, set PLL divider.	CPM	3 SB		x	x	x
Release bus for all transactions.	CPM	1 SB		x	x	x
Wait for synchronization.	CPM	<20 CPU	x	x	x	x
Enable clocks, interrupts to CPU and begin execution.	CPM	2 SB	x	x	x	x
Deassert internal reset, nRESET_OUT.	CPM	1 SB			x	x

NOTES:

x	Step is followed in the corresponding power mode
?	Variable
13M	13-MHz processor oscillator
32k	32.768-kHz timekeeping oscillator
CPM	Clocks and power manager
CPU	CPU
MEM	Memory controller
SB	System bus

3.7 Voltage Manager Operation

The voltage manager provides dynamic and static voltage management to the processor through the use of an I^2C module (PWR_I^2C) dedicated to communication with the external regulator. The voltage manager provides the following features:

- Static (halted) or dynamic (operational) voltage change
- Up to 32 I^2C commands automatically sent to external PWR_I^2C module
- Programmable delay between commands

The voltage manager consists of two primary components:

- Dedicated power manager I^2C module (PWR_I^2C)
- Command sequencer

3.7.1 Power Manager I^2C and Restrictions

The dedicated I^2C module used by the voltage manager is nearly identical to the I^2C described in Chapter 9, "I^2C Bus Interface Unit". The power manager I^2C (PWR_I^2C) is optimized for connection to the external voltage regulator only. The power manager I^2C is a full-featured I^2C module that can connect to other units, although operation during a voltage-change operation may be limited.

Refer to the *Intel® PXA27x Processor Family EMTS* for voltage-change timing specifications.

3.7.1.1 Programming Restrictions

Except for the ICCR fixed selections, power manager I^2C is a full-function I^2C capable of all normal operations, including master and slave, receive and transmit operation. The power manager I^2C supports standard-speed operation of 40 kbits/sec and fast-speed operation of 160 kbits/sec. When used with the voltage-change sequencer, the power manager I^2C operates at standard speed. When used apart from the voltage-change sequencer, the power manager I^2C supports all I^2C specifications. The PI^2C_EN bit in register PCFR (see Section 3.8.1.8) must be set to use the power manager I^2C, either with or without the voltage-change sequencer.

The following condition applies when the voltage-change sequencer is operating, indicated by hardware setting the PVCR[VCSA] (see Section 3.8.1.13):

- The power manager I^2C registers (PCMDx and PVCR) are not writable and reads return unknown values.

Thus, software must check PVCR[VCSA] before reading or writing to the power manager I^2C registers and must read the registers following a write to ensure that the write occurred.

The following restrictions apply to the voltage-change sequencer:

- The sequencer allows a maximum of 32 commands in the transmission of data to the external regulator.
- The sequencer allows only master-transmitter operations to a single, predefined slave.
- The sequencer does not send interrupts to the CPU.
- Only the standard-speed operation at 40 kbits/sec is used.

3.7.2 Voltage-Change Sequencer

The voltage manager contains a voltage-change sequencer, which automatically sends commands to the external regulator when triggered by the voltage-change mode. The sequencer can send up to 32 commands, which can be categorized as *dynamic commands* and *static commands*.

Dynamic commands are executed when the core is running. The power manager requests the voltage manager for command execution. The sequencer starts sending out the commands as soon as the request is received. The voltage manager acknowledges the power manager after completing all of the commands.

Static commands are executed after clocks to the processor are disabled. Static commands are transmitted by the voltage manager when voltage change is coupled with a frequency change (see Section 3.7.6.3) or a power-mode change (see Section 3.7.6.4).

3.7.2.1 Voltage-Change Sequencer Controls

The following control bits in PCFR (see Section 3.8.1.8), PVCR (see Section 3.8.1.13), and PCMD (see Section 3.8.1.17) affect execution of a voltage-change sequence:

- Frequency/voltage change bit (PCFR[FVC])—When set, a frequency-change sequence also triggers a voltage-change sequence.
- Read Pointer in PVCR—These bits point to the PCMD register location that contains the command to be sent out. The command sequence can start from any PCMD register by programming these bits accordingly. After a command is sent out, the read pointer increments

to point to the next PCMD register location. The read pointer is not incremented if the current command is the last command, as indicated by PCMD[LC] set.

- Delay command execution bit (PCMD[DCE])—If DCE is set in the current PCMD, a counter (set by the command delay bits in PVCR) waits for a programmable number of 13-MHz processor-oscillator cycles before continuing execution of the command. This is useful if a longer period between commands is required (for example, to allow for stabilization).
- Multi-byte command bit (PCMD[MBC])—If set, the voltage-change sequencer continues sending bytes to the slave with no delay or handshaking with the power manager until a command with PCMD[MBC] clear is executed. PCMD[SQC] and PCMD[LC] must be the same for all bytes of a multi-byte command except for the last byte, which has PCMD[MBC] clear and PCMD[LC] set.
- Last command bit (PCMD[LC])—When clear, the voltage-change sequencer expects the PCMD register at the next higher address to contain an additional command. If PCMD[LC] is clear in PCMD31, the PVCR read pointer rolls over to PCMD0 after executing the command in PCMD31. When PCMD[LC] is set, the voltage-change sequencer considers the current command the last one and finishes after execution completes. Each voltage-change command sequence must be terminated by setting PCMD[LC] for the last command in the sequence. The PVCR read pointer is not incremented if PCMD[LC] is set.

3.7.2.2 Static Voltage-Change Sequence Configurations

Execution of the voltage-change sequence is controlled by the following configurations set by the SQC bits in PCMDx:

- Continue configuration (PCMD[SQC] set to Continue)—Execution of this command is automatic when the command is read from PCMDx.
- Pause configuration (PCMD[SQC] set to Pause)—When this command is read, the voltage manager sends a trigger to the power manager after executing the command. The PVCR read pointer is incremented, but execution of the next command pauses until the power manager issues a new request to the voltage manager.

3.7.2.3 Prerequisites for Voltage-Change Sequence

Configure the following items before initiating an automatic voltage-change sequence:

1. Set PCFR[PI^{2}C_EN].
2. If the voltage-change sequence is to be used in conjunction with a clock frequency change, set PCFR[FVC].
3. Program the required delay between commands into PVCR[Command Delay].
4. Program the address for the external voltage regulator into PVCR[Slave Address].
5. Configure the external voltage regulator as a slave with the same address that is specified in PVCR[Slave Address].
6. Set all units on the power manager I^{2}C bus to slave-receive mode. They must be prevented from transmitting on the I^{2}C bus during the voltage-change sequence.
7. Load PCMDx with the commands to be sent.
8. Write the starting command location to PVCR[Read Pointer].

3.7.2.4 Sequence Initiation

The first command in the Power Manager I^{2}C Command register file (PCMDx) is initiated immediately when the power manager sends a request to the voltage manager, triggered by either of the following events:

- 0b1 is written to PWRMODE[VC])
- 0b1 is written to CLKCFG[F] while PCFR[FVC] is set.

Once the voltage-change sequence is initiated, reads from and writes to the power manager I^{2}C module are ignored until the voltage-change sequence is complete. All registers return unknown values if read, writes are ignored, and interrupts from power manager I^{2}C are directed to the automatic voltage-change sequencer instead of to the interrupt controller. This state is indicated when PVCR[VCSA]. is set.

Each command is executed from PCMDx in order of increasing addresses, starting with the address in PVCR[Read Pointer]. The command is executed if all of the following events have occurred:

1. PWRMODE[VC] is set, or 0b1 is written to CLKCFG[F], while PCFR[FVC] is set.
2. All previous (lower-address) commands have completed execution but did not set their LC configuration.
3. If the Pause configuration was set for the previous command, then a request has been received from the power manager.

Once initiated, PVCR[VCSA] is set and remains set until the voltage-change sequence is complete. While VCSA is set, the power manager I^{2}C registers ignore writes and return undefined values if read.

3.7.2.5 Command Execution

Each command is set up as a master-mode transmission to the slave device (the slave address is stored in (PVCR[Slave Address]). Because all commands use the same transmission type and slave address, automatic communication is limited to a single device. The voltage-change sequence occurs automatically, as follows.

1. If Pause is set in the previously transmitted PCMDx command, wait for a request from the power manager.
2. If PCMDx[DCE] is set in the current command, count the specified number of 13-MHz processor oscillator cycles.
3. Write the slave address to the Power I^{2}C Data Buffer register, PIDBR[7:1]. Clear PIDBR[0]. See Section 9.5.4, “I^{2}C Data Buffer Register (IDBR, PIDBR)” on page 9-29 for PIDBR register details.
4. In the I^{2}C Control register, set the transmit-empty interrupt-enable (ITEIE), I2C unit enable, (IUE), SCL enable (SCLEA), transfer byte (TB), and start (START) bits. This sends one byte in master-transmit mode to the power manager I^{2}C bus. See Section 9.5.1, “I^{2}C Control Registers (ICR, PICR)” on page 9-23 for PICR register details.
5. Wait for the IDBR transmit-empty interrupt, which automatically triggers the next step.
6. Read the Power I^{2}C Status register (PISR), looking for the IDBR transmit-empty (ITE) and unit-busy (UB) bits to be set. See Section 9.5.2, “I^{2}C Status Registers (ISR, PISR)” on page 9-26 for PISR register details.
7. Read PISR to clear PISR[ITEIE].

8. Write the currently-pointed-to PCMDx[Command Data] to PIDBR[7:0].
9. Set PICR[ITEIE], PICR[IUE], PICR[SCLEA], and PICR[TB]. If PCMDx[SQC] is set to Pause for this command (indicating a pause after the command transmission), **or** if PCMDx[MBC] is clear and PCMDx[LC] is set (indicating the last command of the sequence), set PICR[STOP]. This sends the command byte in master-transmit mode to the power I^2C bus with a stop bit to terminate the communication.
10. Wait for the IDBR transmit-empty interrupt, which automatically triggers the next step.
11. Read PISR, looking for PISR[ITE] and PISR[UB] to be set.
12. If the current PCMDx[MBC] bit is set, repeat steps 7-11 until a command is executed in which PCMDx[MBC] is clear.
13. If the current PCMDx[LC] or PCMDx[Pause] is set:
    a. Trigger the power manager to begin modification or power-off of the power supplies.
    b. Trigger the power manager to re-enable the clocks.
    c. Exit the voltage-change sequence and clear PVCR[VCSA].
14. If the current PCMDx[LC] is clear (PCMDx[LC] is set to Continue), increment PVCR[Read Pointer] and execute the next command. (repeat steps 1-14).

3.7.3 External Voltage Regulator Requirements

The external voltage regulator must meet the requirements described in the *Intel® PXA27x Processor Family EMTS*.

3.7.4 Sending Commands Using Voltage-Change Sequencer

The voltage-change sequencer sends single-byte, multi-byte, or sets of single- and multi-byte commands to the external regulator. These sequences can be used individually or as part of complex power-mode voltage changes.

3.7.4.1 Single-Byte Command Voltage Change

Power manager I^2C commands can be sent to the external regulator at any time. Sending a single command to the external regulator's I^2C module is efficient, using direct software control of the power manager I^2C bus. This sequence is the basic building block for more complex change sequences. To send a single command to the external regulator, use the following sequence, where:

n = number of the PCMD register containing the single-byte command (see Section 3.8.1.17)

1. Clear PCFR[FVC] (see Section 3.8.1.8) and PVCR[Command Delay] (see Section 3.8.1.13). These functions are not required when sending a single command to the external regulator.
2. Program PVCR[Slave Address] with the external regulator's I^2C address. Because PCMD register *n* contains the single-byte command, write *n* to PVCR[Read Pointer].
3. In PCMD*n*, clear the SQC, MBC and DCE bits. These functions are not required when sending a single command to the external regulator.
4. Set PCMD*n*[LC], which indicates that this is the last command.
5. Program PCMD*n*[Command Data] with the data to be sent to the external regulator.

6. Because PCMD*n*[LC] is set, the voltage-change sequencer ignores the remaining PCMD registers.
7. Enable power manager I^2C by setting PCFR[PI^{2}C_EN] **before** writing 0b1 to PWRMODE[VC]. PCFR[PI^{2}C_EN] must remain set as long as the power manager I^2C is being used on the I^2C bus.
8. Execute the voltage-change sequence by setting PWRMODE[VC]. See Section 3.8.3.2 for register details.

The voltage-change sequence begins as soon as PWRMODE[VC] is written with 0b1. One byte (PCMD*n*[Command Data]) is sent to the external regulator. When the sequence is complete, PVCR[VCSA] is cleared.

3.7.4.2 Single, Multiple-Byte Command Voltage-Change

The voltage-change sequencer can send a multi-byte I^2C command to the external regulator at any time. A multi-byte command is a single command that contains more than one byte of PCMDx[Command Data]. These bytes are sent without delay or re-arbitration for the I^2C bus between bytes, even if multiple masters are present on the bus. This sequence is a basic building block for more complex sequences. To send a single, multi-byte command to the external regulator, use the following sequence, where:

n = number of the PCMD register containing the multi-byte command (see Section 3.8.1.17)

m = number of bytes to be sent

1. Clear PCFR[FVC] (see Section 3.8.1.8) and PVCR[Command Delay] (see Section 3.8.1.13). These functions are not required when sending a single command to the external regulator.
2. Program PVCR[Slave Address] with the external regulator's I^2C address. Because PCMD register *n* contains the multi-byte command, write *n* to PVCR[Read Pointer].
3. In registers PCMD*n* through PCMD*n+m–1*, clear the SQC and DCE bits. These functions are not required when sending a single command to the external regulator.
4. In registers PCMD*n* through PCMD*n+m–2*, set the MBC bit. Setting MBC informs the sequencer logic that the next highest PCMD register contains an additional byte of Command Data to be sent as part of the command.
5. Clear PCMD*n+m–1*[MBC]. Clearing MBC informs the sequencer logic that the last byte of Command Data for the command is in the current register.
6. In registers PCMD*n* through PCMD*n+m–1*, set LC. This indicates that the first command in the PCMD is also the last (although the command contains several bytes).
7. In registers PCMD*n* through PCMD*n+m–1*, program Command Data with the data to be sent to the external regulator.
8. Because PCMD*n+m–1*[LC] is set and PCMD*n+m–1*[MBC] is cleared, the voltage-change sequencer ignores the remaining PCMD registers.
9. Enable power manager I^2C by setting PCFR[PI^{2}C_EN] **before** writing 0b1 to PWRMODE[VC]. PCFR[PI^{2}C_EN] must remain set as long as the power manager I^2C is being used on the I^2C bus.
10. Execute the voltage-change sequence by setting PWRMODE[VC]. See Section 3.8.3.2 for register details.

The voltage-change sequence begins as soon as PWRMODE[VC] is written with 0b1. *m* bytes (PCMD*n* through PCMD*n+m–1 Command Data*) are sent to the external regulator. When the sequence is complete, PVCR[VCSA] is cleared.

3.7.4.3 Multiple Single-Byte Command Voltage Change

The voltage-change sequencer can send up to 32 I^2C commands to the external regulator, with programmable delays between commands. Such commands control the ramp rate of the external regulator.

Regulators designed specifically for dynamic voltage control of the processor have built-in ramp control that can be specified with a minimum number of I^2C commands. However, multiple I^2C commands might be required to control the ramp beyond the regulator's built-in capability, or for regulators without built-in ramp control.

Use the following sequence to send multiple single-byte commands to the external regulator, where:

n = number of the PCMD register containing the first command (see Section 3.8.1.17)

m = number of commands to be sent

1. Clear PCFR[FVC] (see Section 3.8.1.8). This function is not required when sending multiple commands to the external regulator.
2. Program PVCR[Command Delay] with the required delay between single-byte commands (this delay can control the ramp rate). See Section 3.8.1.13 for register details.
3. Program PVCR[Slave Address] with the external regulator's I^2C address. Because PCMD register *n* contains the first single-byte command, write *n* to PVCR[Read Pointer].
4. In registers PCMD*n* through PCMD*n+m–1*, clear the SQC and MBC bits. These functions are not required when sending multiple single-byte commands to the external regulator.
5. Set PCMD*n+m–1*[LC]. This indicates that PCMD*n+m–1* contains the last command.
6. In registers PCMD*n* through PCMD*n+m–1*, program Command Data with the data to be sent to the external regulator.

   For each command that is to be delayed by PVCR[Command Delay], set PCMDx[DCE].
7. Because PCMD*n+m–1*[LC] is set, the sequencer logic ignores the remaining PCMD registers.
8. Enable power manager I^2C by setting PCFR[PI^{2}C_EN] **before** writing 0b1 to PWRMODE[VC]. PCFR[PI^{2}C_EN] must remain set as long as the power manager I^2C is being used on the I^2C bus.
9. Execute the voltage-change sequence by setting PWRMODE[VC]. See Section 3.8.3.2 for register details.

The voltage-change sequence begins as soon as PWRMODE[VC] is written with 0b1. One byte is sent to the external regulator for each PCMD register, for a total of *m* bytes. Commands with PCMD[DCE] set are delayed by PVCR[Command Delay] before being sent. When the sequence is complete, PVCR[VCSA] is cleared.

3.7.4.4 Multiple Single- and Multi-Byte Command Voltage Change

Depending on the type of external regulator used, some commands may need to be single-byte commands and others multiple-byte commands. Additionally, some commands may require a delay between commands, while other commands may need to be sent immediately following the previous command. Use the following sequence to send multiple, single, and multi-byte commands to the external regulator, where:

n = number of the PCMD register containing the first command (see Section 3.8.1.17)

m = number of commands to be sent

t = total number of command bytes to be sent

1. Clear PCFR[FVC] (see Section 3.8.1.8). This function is not required when sending multiple commands to the external regulator.
2. Program PVCR[Command Delay] with the required delay between commands (this delay can control the ramp rate). For register details, see Section 3.8.1.13.
3. Program PVCR[Slave Address] with the external regulator's I^2C address.
4. Because PCMD register *n* contains the first command of the sequence, write *n* to PVCR[Read Pointer].
5. For registers PCMD*n* through PCMD*n+t–1*, clear the SQC bits, which are not required when sending multiple commands to the external regulator.
6. For each multi-byte command of length *m* starting at location *x*:
   a. Set MBC in registers PCMD*x* through PCMD*x+m–2*. MBC set indicates that the following PCMD register contains an additional byte of command data.
   b. Clear MBC in PCMD*x+m*. MBC clear indicates that the current PCMD register contains the last byte of command data for the multi-byte command.
   c. If a delay is required before executing the multi-byte command, set PCMD*x*[DCE].
7. For each single-byte command starting at PCMD*x*, clear PCMD*x*[MBC].
8. If the last command of the sequence is a multi-byte command of *m* bytes,
   a. For registers PCMD*n+t–m-2* through PCMD*n+t–1*, set LC.
   b. For registers PCMD*n+t–m-2* through PCMD*n+t–2*, set MBC.
   c. Clear PCMD*n+t–1*[MBC]. MBC clear indicates that the last command is contained in registers PCMD*n+t–m-2* through PCMD*n+t–1*.
9. For registers PCMD*n* through PCMD*n+t–1*, program command data with the data to be sent to the external regulator for each command.
10. Because PCMD*n+t–1*[LC] is set, the sequencer logic ignores the remaining PCMD registers.
11. Enable power manager I^2C by setting PCFR[PI^{2}C_EN] **before** writing 0b1 to PWRMODE[VC]. PCFR[PI^{2}C_EN] must remain set as long as the power manager I^2C is being used on the I^2C bus.
12. Execute the voltage-change sequence by setting PWRMODE[VC]. See Section 3.8.3.2 for register details.

The voltage-change sequence begins as soon as PWRMODE[VC] is written with 0b1.

For each single-byte command, one byte of command data is sent to the external regulator. For each multi-byte command, multiple bytes are sent, as specified by the MBC bits. A total of *t* bytes is sent. Commands with PCMD[DCE] set are delayed by PVCR[Command Delay] before being sent.

PVCR[VCSA] is cleared when the sequence is complete.

3.7.5 Behavior During Power-Fault Assertion

If nBATT_FAULT or nVDD_FAULT is asserted while the voltage manager is transmitting a command and the corresponding xIDAE bits (BIDAE, VIDAE) are clear in the Power Manager Control register (PMCR), the command sequence is aborted with a STOP condition on the I^2C bus after completing the next command in the sequence. In this case, programmed delay between commands, if any, is ignored. Even if the command being transmitted is part of a multi-byte command, the behavior is the same as above.

If the nBATT_FAULT or nVDD_FAULT occurs during the programmed delay between commands while the corresponding PMCR[xIDAE] bits are clear, the sequence terminates after executing the command following the delay.

If nBATT_FAULT or nVDD_FAULT is asserted while the corresponding PMCR[xIDAE] bits are set, the command sequence is not terminated.

3.7.6 Using the Voltage Manager

The power manager I^2C bus and voltage-change sequencer are normally used to control the voltage applied to the internal logic (supplied by VCC_CORE) based on frequency. The voltage can be adjusted at any time relative to software execution, provided that the applied voltage meets the requirements for the frequency in use at the time. However, the most efficient use of the voltage manager is achieved when used in conjunction with the power-mode and frequency controls.

3.7.6.1 Voltage Change at Initialization

Voltage regulators for the PXA27x processor must power on with the correct default voltages. For details, see the *Intel® PXA27x Processor Family EMTS*. During the boot sequence, the power manager I^2C can be used to adjust the regulator output voltages to meet the system requirements. These adjustments can be made with direct software access to the power manager I^2C, so that use of the voltage-change sequencer is not necessary.

3.7.6.2 Coupling Voltage Change with Turbo Modes

Adjusting the processor frequency and voltage according to application requirements achieves optimal system power consumption. The fastest way of adjusting the processor frequency is through the use of turbo mode (see Section 3.5.7.4) or half-turbo mode (see Section 3.5.7.5), which adjusts the frequencies of the CPU and the system bus without affecting operation of the peripheral modules. Follow these steps to adjust the voltage accordingly:

To raise the voltage and frequency:

1. At boot-up, set the turbo mode to run mode ratio (CCCR[2N]) to the required values. If required, execute a frequency change.

2. Program the PCMD and the PVCR registers to send the types of commands necessary to raise the voltage. For example, for a controlled ramp, follow the procedure given in Section 3.7.4.3.
3. Initiate a voltage-change sequence by setting PWRMODE[VC].
4. Wait for PVCR[VCSA] to clear by periodically polling the register.
5. Enable turbo mode (or half-turbo mode) by writing to the appropriate bits in the CLKCFG register.

To lower the voltage and frequency:

1. At boot-up, set the turbo mode to run mode ratio (CCCR[2N]) to the required values. If required, execute a core frequency change, turbo-mode change, or fast-bus mode change.
2. Program the PCMD and the PVCR registers to send the types of commands necessary to lower the voltage. For example, for a controlled ramp, follow the procedure given in Section 3.7.4.3.
3. Exit turbo mode (or half-turbo mode) by writing to the appropriate bits in the CLKCFG register.
4. Initiate a voltage-change sequence by setting PWRMODE[VC].
5. The voltage change is complete when PVCR[VCSA] is clear.

3.7.6.3 Coupling Voltage Change with Frequency Change

A frequency change (clock source change or core PLL frequency change) can be used to change the frequency of the CPU, system bus, memory controller, and LCD controller to a value not available with turbo or fast-bus modes. This change can be coupled with a voltage change in a similar way as turbo mode. The only additional requirement is that PCFR[FVC] be set. Similarly, voltage change can be coupled with fast-bus mode. See sections Section 3.5.7.3 and Section 3.5.7.6 for details of these frequency changes.

In the case of a core PLL frequency change where the core PLL is also the core clock source (CCCR[CPDIS] = 0), there is a delay while the PLL re-locks. Use the following sequence to change the voltage and frequency to reduce the overall delay caused by the frequency change and the voltage change as well as the software overhead required to do both:

1. Set the turbo-mode-to-run-mode ratio (CCCR[2N]) to the required values.
2. Program the PCMD registers with the required commands. For each command, set PCMDx[SQC] to Continue. For the last command, set PCMD[LC].

   The first command is executed as soon as the power manager asserts the request. After the last command is executed, the voltage manager triggers the clocks manager to perform a frequency change.
3. Set PCFR[FVC].
4. Initiate a frequency-change sequence by writing to CLKCFG[F] (or CLKCFG[B] if fast-bus mode is to be used).
5. The frequency-change sequence exits at the new voltage and frequency.

Note: Observe the appropriate frequency/voltage specification (refer to the *Intel® PXA27x Processor Family EMTS* for details) for the VCC_CORE power domain when initiating any frequency or frequency-coupled voltage changes.

3.7.6.4 Coupling Voltage Change with Power-Mode Changes

Low-power modes (deep idle, idle, standby, sleep, and deep sleep) can reduce power consumption significantly by gating clocks, reducing leakage, or powering off large sections of the processor. Before entering these modes, the voltage can be adjusted for optimum power consumption.

The voltage can be changed in anticipation of entry into one of these power modes in the same way that frequency and voltage can be changed, requiring a frequency change and a voltage change before the power mode is changed and again after the power-mode change is finished, potentially increasing the latency of the power-mode change. Power-mode changes can be coupled more efficiently with voltage change by using the following sequences.

To raise the voltage during a power-mode change:

1. Set the turbo mode to run mode ratio (CCCR[2N]) to the required value.
2. Program the PCMD registers in the following order:
   a. The first set of commands consists of those that may be required by the external voltage regulator but do not actually change the voltage. These commands all have the SQC bits set to Continue. The first command is executed as soon as the power manager asserts a request to the voltage manager.
   b. The second set of commands consists of those that actually raise the voltage; it can be a ramp or a single command. Because the voltage must be raised before the power mode is entered, these commands all have the SQC bits set to Continue, except the last command of this set, which has the SQC bits set to Pause. After execution of this command, the voltage manager sends a trigger to the power manager to enter the power mode.
   c. The third set of commands consists of those that again lower the voltage after the power-mode change is complete (these commands are executed after the power-mode wake-up source is asserted and the PWR_EN and SYS_EN pins and associated timers have been asserted). The last command in this set must have the PCMDx[LC] set. After executing the last command, the voltage manager again sends a trigger to the power manager, signaling the end of the voltage-change sequence.
3. Initiate a voltage-change sequence and power-mode change concurrently by writing to PWRMODE[VC] and PWRMODE[M] simultaneously.

This sequence raises the voltage, enters the power mode, exits the power mode according to its normal exit mechanism, and then lowers the voltage.

To lower the voltage during a power-mode change:

1. Set the turbo-mode-to-run-mode ratio (CCCR[2N] to the required value.
2. Program the PCMD registers in the following order:
   a. The first set of commands consists of those that may be required by the external voltage regulator but do not actually change the voltage. These commands all have the SQC bits set to Continue. The first command is executed as soon as power manager asserts a request to the voltage manager.
   b. The second set of commands consists of those that actually lower the voltage; it can be a ramp or a single command. Since clocks were already stopped before the power manager asserted the request, these commands have the SQC bits set to Continue, except the last command of this set, which has the SQC bits set to Pause. After the execution of this command, the voltage manager sends a trigger to the power manager to enter the power mode.

c. The third set of commands consists of those that again raise the voltage after the power mode is complete (these commands are executed after the power-mode wake-up source is asserted and the PWR_EN and SYS_EN pins and associated timers have been asserted). The last command in this set must have the LC bit set. After executing the last command, the voltage manager sends a trigger to the power manager signaling the end of voltage-change sequence.

3. Initiate a voltage-change sequence and power-mode change concurrently by writing to PWRMODE[VC] and PWRMODE[M] at the same time.

This sequence lowers the voltage, enters the power mode, exits the power mode according to its normal exit mechanism, and then raises the voltage.

3.7.6.5 Alternate Method: Voltage, Frequency, and Power-Mode Changes

An alternative means of performing a voltage change coupled with a frequency change does not require the use of the PWRMODE[VC] or PVCR[VCSA] functionality. The following sequences illustrate this alternative method, which is as efficient as using the automatic power manager I^2C coupling described in Section 3.7.6.3.

To raise the voltage and frequency (for example, enter turbo mode or increase the value of L):

1. Perform the voltage change by instructing the external power-management device to raise the voltage to the final level.
2. Optionally, the external regulator can generate an interrupt after the transfer is completed.
3. If necessary, add a delay to allow for the time required by the external device to change the voltage.
4. Perform the frequency change.

To lower the voltage and frequency (for example, exit turbo mode or decrease the value of L):

1. Perform the frequency change.
2. Perform the voltage change by instructing the external power-management device to lower the voltage to the final level.
3. Optionally, the external regulator can generate an interrupt after the transfer is completed.

To couple a voltage change with a power-mode change, use a sequence that is similar to the ones shown above.

3.8 Register Descriptions

The following sections describe the registers used by the clocks and power manager:

Section 3.8.1 — Power Manager Registers

Section 3.8.2 — Clocks Manager Registers

Section 3.8.3 — Coprocessor 14: Clock and Power Management

3.8.1 Power Manager Registers

The power manager uses the following 32-bit registers:

- Section 3.8.1.1 — Power Manager Control Register (PMCR) selects whether nVDD_FAULT and nBATT_FAULT cause immediate entry into sleep mode or cause an imprecise data abort. PMCR also indicates whether an imprecise data abort has occurred.
- Section 3.8.1.2 — Power Manager Sleep Status Register (PSSR) contains status bits that indicate whether sleep or standby mode was invoked. PSSR also identifies the states of certain I/O pins after resets.
- Section 3.8.1.3 — Power Manager Scratch-Pad Register (PSPR) is a general-purpose register that stores processor data during all power modes.
- Section 3.8.1.4 — Power Manager Wake-Up Enable Register (PWER), Section 3.8.1.5 — Power Manager Rising-Edge Detect Enable Register (PRER), and Section 3.8.1.6 — Power Manager Falling-Edge Detect Enable Register (PFER) program the sleep wake-up sources in the system.
- Section 3.8.1.7 — Power Manager Edge-Detect Status Register (PEDR) indicates which GPIO pin caused a wake-up from standby, sleep, or deep-sleep mode.
- Section 3.8.1.8 — Power Manager General Configuration Register (PCFR) controls various configurable functions and the disable status of modules during power modes in the processor.
- Section 3.8.1.9 — Power Manager GPIO Sleep-State Registers (PGSRx) program the values loaded onto GPIO outputs when the processor switches into sleep or deep-sleep mode.
- Section 3.8.1.10 — Reset Controller Status Register (RCSR) indicates the source that caused a reset.
- Section 3.8.1.11 — Power Manager Sleep Configuration Register (PSLR), and Section 3.8.1.12 — Power Manager Standby Configuration Register (PSTR) control the power features of sleep and standby modes.
- Section 3.8.1.13 — Power Manager Voltage Change Control Register (PVCR) holds configuration data for the external voltage regulator.
- Section 3.8.1.17 — Power Manager I^2C Command Register File (PCMDx) is a bank of registers that hold the I^2C commands and sequence information for use in a voltage-change sequence.
- Section 3.8.1.15 — Power Manager Keyboard Wake-Up Enable Register (PKWR) and Section 3.8.1.16 — Power Manager Keyboard Level-Detect Status Register (PKSR) detect and manage keyboard wake-up events.
- Section 9.5.1 — I^2C Control Registers (ICR, PICR), Section 9.5.2 — I^2C Status Registers (ISR, PISR), Section 9.5.3 — I^2C Slave Address Registers (ISAR, PISAR), Section 9.5.4 — I^2C Data Buffer Register (IDBR, PIDBR), and Section 9.5.5 — I^2C Bus Monitor Registers (IBMR, PIBMR) control the power manager I^2C interface.

3.8.1.1 Power Manager Control Register (PMCR)

PMCR, defined in Table 3-13, controls processor behavior when nVDD_FAULT or nBATT_FAULT is asserted. There are two imprecise data-abort-enable bits (xIDAE), one for each fault:

BIDAE—Battery fault (nBATT_FAULT)

VIDAE—VCC fault (nVDD_FAULT)

If a battery or VCC fault occurs and the corresponding xIDAE bit is clear, the processor enters deep-sleep mode. If a battery or VCC fault occurs and the corresponding xIDAE bit is set, the processor enters or remains in normal mode and sends an imprecise data abort or interrupt to the core. If both faults are asserted and one of the xIDAE bits is clear, the processor enters deep-sleep mode.

Depending on PMCR[IAS], either an interrupt or an abort is sent to the core. The imprecise -data abort handler or the interrupt handler for this case can store critical data before entering deep-sleep mode through software or before clearing xIDAE to let the processor enter deep-sleep mode.

If a battery or VCC fault occurs:

- **If the corresponding xIDAE bit is clear**, indicating immediate entry into deep-sleep mode, the PI power domain and SRAMs are powered off regardless of the PSLR register unit-retention settings. If voltage-manager commands are being sent on the power manager I^2C (PWR_I^2C) bus, the processor stops sending the voltage-manager commands after completing the next command, and the rest of the sequence is terminated. In this case, PMCR[xIDAS] and PMCR[INTRS] are not set.
- **If the corresponding xIDAE bit is set** and voltage-manager commands are being sent on the PWR_I^2C bus, the processor completes all the sequences and then sends the abort or interrupt to the processor core. The PMCR[xIDAS] bits contain the status of whichever fault was asserted.

  If PMCR[IAS] is clear, an imprecise data abort is reported to the core. If PMCR[IAS] is set, an interrupt is sent to the interrupt controller unit. PMCR[INTRS] contains the status of the interrupt.

  Write 0b1 to clear the PMCR[xIDAS] or PMCR[INTRS] bits.

Note: Use xIDAE = 0b0 with caution because the processor enters deep-sleep mode without letting the core or other on-chip peripherals complete their tasks.

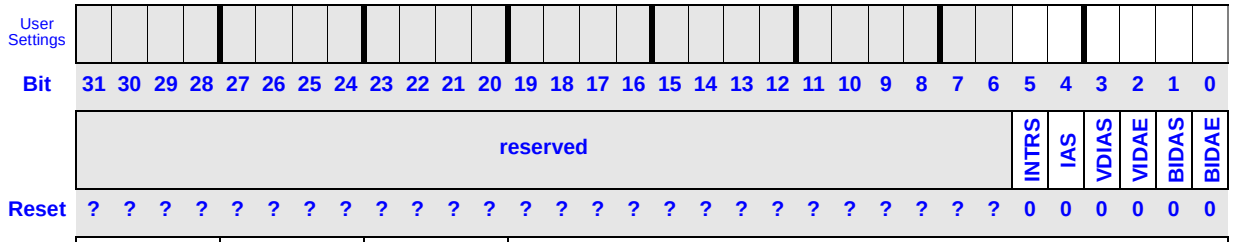
Refer to the PSLR register (Section 3.8.1.11) for the definition of the IVF bit, which can allow ignoring of the VCC fault during entry into sleep mode.

Power-on, hardware, watchdog, and GPIO resets return the PMCR bits to their reset values, as shown in Table 3-13.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-13. PMCR Bit Definitions

Physical Address 0x40F0_0000 — PMCR — Clocks and Power Manager



Bits	Access	Name	Description
31:6	—	—	reserved
5	R/WC†	INTRS	Interrupt Status The interrupt is sent from the PWR_I^{2}C block to the interrupt controller, which in turn sends it to the core if the appropriate masks are enabled. For more information, see Chapter 25, "Interrupt Controller". 0 = No interrupt was reported to the interrupt controller. 1 = An interrupt was reported to the interrupt controller.
4	R/W	IAS	Interrupt/Abort Select 0 = Send an abort to the core on a fault with the corresponding xIDAE bit set. 1 = Send an interrupt to the core on a fault with the corresponding xIDAE bit set.
3	R/WC†	VIDAS	Imprecise-Data-Abort Status for nVDD_FAULT 0 = No data abort occurred since the last time VIDAS was cleared by software or reset; or, the data abort was not due to assertion of nVDD_FAULT with VIDAE set. 1 = The data abort was due to the assertion of nVDD_FAULT with VIDAE set.
2	R/W	VIDAE	Imprecise-Data-Abort Enable for nVDD_FAULT 0 = Allow immediate entry into deep-sleep mode when nVDD_FAULT is asserted. 1 = Force an imprecise-data-abort signal to the CPU. This allows software entry into deep-sleep when nVDD_FAULT is asserted. **NOTE:** VIDAE and BIDAE must be identical.
1	R/WC†	BIDAS	Imprecise-Data Abort Status for nBATT_FAULT 0 = No data abort occurred since the last time IDAS was cleared by software or reset; or, the data abort was not due to assertion of nBATT_FAULT with BIDAE set. 1 = The data abort was due to the assertion of nBATT_FAULT with BIDAE set.
0	R/W	BIDAE	Imprecise-Data Abort Enable for nBATT_FAULT 0 = Allow immediate entry into deep-sleep when nBATT_FAULT is asserted. 1 = Force an imprecise-data-abort signal to the CPU. This allows software entry into deep-sleep when nBATT_FAULT is asserted. **NOTE:** BIDAE and VIDAE must be identical.

NOTE:
† Write 0b1 to this bit to clear it. Writing 0b0 has no effect.

3.8.1.2 Power Manager Sleep Status Register (PSSR)

PSSR, defined in Table 3-15, contains the following status flags:

- Read Disable Hold (RDH) is set during any reset and during sleep and deep-sleep. RDH indicates that all processor GPIO input paths are disabled. If RDH was set due to a power-on, hardware, watchdog, GPIO reset, or a deep-sleep entry, then a resistive pullup or pulldown in the pad is enabled and remains enabled until RDH is cleared. Software must clear this bit for any GPIO input pin to be enabled. The following is a summary of RDH operation:
  - Hardware, power-on, GPIO, and watchdog reset, or deep-sleep mode:
    - The RDH bit is set.
    - The receivers of all GPIO pins are disabled until RDH bit is cleared.
    - The pullups/pulldowns on the GPIO pins are enabled.
    - Any GPIOs that could be wake ups are not affected.
  - Deep-sleep mode:
    - The RDH bit gets set.
    - The receivers of all GPIO pins are disabled if PCFR[RO] is clear. The receivers of all GPIO pins are not disabled if PCFR[RO] is set.
    - The pullups/pulldowns on the GPIO pins are enabled.
    - Any GPIOs that could serve as wakeups are not affected.
  - Sleep mode:
    - The RDH bit is set.
    - The receivers of all GPIO pins are disabled if PCFR[RO] is clear**.** The receivers of all GPIO pins are not disabled if PCFR[RO] is set.
    - The pullups/pulldowns on the GPIO pins are not enabled.
    - Any GPIOs that could be wakeups are not affected.
  - Other power modes (standby, idle, and deep idle):
    - The RDH bit is not set.
    - The receivers of all GPIO pins are not disabled.
    - The pullups/pulldowns on the GPIO pins are not enabled.
    - Any GPIOs that could be wakeups are not affected.

Table 3-14 shows the effect of RDH operation in low-power modes.

Table 3-14. RDH Operation in Low-Power Modes

Mode	Effect of RDH with PCFR[RO] Bit Clear	Effect of RDH with PCFR[RO] Bit Set
Idle Modes and Standby Mode	**No Effect**	**No Effect**
Hardware/ Power-On/ Watchdog/ GPIO Reset	• The RDH bit gets set. • The receivers of all GPIO pins are disabled until RDH bit is clear. • The pullups/downs on the GPIO pins are enabled.	• The RDH bit gets set. • The receivers of all GPIO pins are disabled until RDH bit is clear. • The pullups/downs on the GPIO pins are enabled.
Deep-Sleep Mode	• The RDH bit gets set. • The receivers of all GPIO pins are **disabled** until RDH bit is clear. • The pullups/downs on the GPIO pins are enabled. • Any GPIOs that could be wakeups are not affected.	• The RDH bit gets set. • The receivers of all GPIO pins are **enabled** after exiting the mode even though the RDH bit is set • The pullups/downs on the GPIO pins are enabled. • Any GPIOs that could be wakeups are not affected.
Sleep Mode	• The RDH bit gets set. • The receivers of all GPIO pins are **disabled** until RDH bit is clear. • The pullups/downs on the GPIO pins are **not** enabled. • Any GPIOs that could be wakeups are not affected.	• The RDH bit gets set. • The receivers of all GPIO pins are **enabled** after exiting the mode even though the RDH bit is set • The pullups/downs on the GPIO pins are **not** enabled. • Any GPIOs that could be wakeups are not affected.

- Peripheral Control Hold (PH) is set upon entry into standby and sleep mode if PCFR[PO] is clear. It indicates that the GPIO pins are retaining their states. PH is clear during deep sleep. If PCFR[PO] is set, PH is cleared automatically after exiting the mode.
- Standby-mode Status (STS) is set when standby mode is entered as a result of setting the standby mode configuration in the PWRMODE register (coprocessor 14, register C7—see Section 3.8.3.2).
- VCC Fault Status (VFS) is set when the assertion of nVDD_FAULT invokes deep sleep.
- Battery Fault Status (BFS) is set when the assertion of nBATT_FAULT invokes deep sleep.
- Software Sleep Status (SSS) is set when sleep mode is entered as a result of setting the sleep-mode configuration in the PWRMODE register (coprocessor 14, register C7—see Section 3.8.3.2). Bit PSSR[SSS] is not set for deep-sleep mode. If it is necessary to determine if an exit from deep-sleep is being performed, then:
  - — If the reset handler is entered, the RCSR bits are clear, and PSSR[SSS] is clear, then an exit from deep-sleep mode is being performed, or
  - — Software can use the power manager scratch-pad register to store an indication that a deep-sleep entry was performed.
- USB On-The-Go (OTG) Peripheral Control Hold (OTGPH) is set when sleep mode is entered as a result of setting the sleep mode configuration in the PWRMODE register (coprocessor 14, register C7). It indicates that the OTG pad is retaining its state. Upon exit from sleep mode, and before clearing OTGPH, software must configure the USB OTG pad, UDC, and UHC to the state they were in before entering sleep mode. The USB OTG pad pullup and pulldown resistor settings must be restored before clearing OTGPH to avoid invalid changes in the USB OTG D+ and D– signals (USBC D+ and D–) after exit from sleep mode. The OTGPH is also set when standby mode is entered and is cleared automatically when standby mode is exited.

The status flags are cleared by writing 0b1 to them. Writing 0b0 has no effect. Power-on, hardware, watchdog, and GPIO resets return the PSSR bits to their reset values, as shown in Table 3-15. RDH is set as a result of sleep or deep-sleep entry.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-15. PSSR Bit Definitions (Sheet 1 of 2)

Physical Address 0x40F0_0004 — **PSSR** — **Clocks and Power Manager**

	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
User Settings																																
Bit	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																									OTGPH	RDH	PH	STS	VFS	BFS	SSS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	0[1]	0[2]	0[2]	0[2]	0[2]

Bits	Access	Name	Description
31:7	—	—	reserved
6	R/WC[3]	OTGPH	OTG Peripheral Control Hold 0 = OTG pad is not holding its state. 1 = OTG pad is holding its state.
5	R/WC[3]	RDH	Read Disable Hold 0 = GPIO pins are configured according to their GPIO configuration (see Chapter 24, “General-Purpose I/O Controller”). 1 = The receivers of all GPIO pins are disabled. If RDH is set as a result of any reset except sleep, resistive pull-downs are enabled until RDH is cleared. RDH must be cleared by software after the peripheral and GPIO interfaces have been configured but before they are actually used.
4	R/WC[3]	PH	Peripheral Control Hold 0 = GPIO pins are configured according to their GPIO configuration. 1 = GPIO pins are held in their sleep-mode states. PH is set upon entry into sleep or standby mode. PH must be cleared by software after the peripherals and GPIO interfaces have been configured but before they are actually used. PH is clear during deep-sleep.
3	R/WC[3]	STS	Standby Mode Status 0 = The processor has not been placed in standby mode by configuring the PWRMODE register since STS was cleared by a reset or by software. 1 = The processor was placed in standby mode by configuring the PWRMODE register.
2	R/WC[3]	VFS	VCC Fault Status 0 = nVDD_FAULT has not been asserted since it was last cleared by a reset or by software. 1 = nVDD_FAULT has been asserted and caused the processor to enter deep-sleep mode. **NOTE:** This bit is not set by the assertion of nVDD_FAULT while the processor is in deep-sleep mode.

Table 3-15. PSSR Bit Definitions (Sheet 2 of 2)

Physical Address 0x40F0_0004 — PSSR — Clocks and Power Manager

	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
User Settings																																
Bit	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																									OTGPH	RDH	PH	STS	VFS	BFS	SSS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	0[1]	0[2]	0[2]	0[2]	0[2]

Bits	Access	Name	Description
1	R/WC[3]	BFS	Battery Fault Status 0 = nBATT_FAULT has not been asserted since it was last cleared by a reset or by software. 1 = nBATT_FAULT has been asserted and caused the processor to enter deep-sleep mode.
0	R/WC[3]	SSS	Software Sleep Status 0 = The processor has not been placed in sleep mode by configuring the PWRMODE register since SSS was last cleared by a reset or by the software. 1 = The processor was placed in sleep mode by configuring the PWRMODE register.

NOTES:
1. Exit from deep-sleep mode clears this bit.
2. Exit from sleep or deep-sleep mode does not clear or set this bit.
3. To clear this bit, write 0b1 to it.

3.8.1.3 Power Manager Scratch-Pad Register (PSPR)

PSPR saves processor configuration information in any preferred format. PSPR is a holding register that is powered during sleep and deep-sleep modes and is cleared by power-on, hardware, watchdog, and GPIO resets. Any value can be written to it while in normal mode. The value can be read once sleep or deep-sleep mode is exited. Use the register value to retain processor configuration prior to invoking sleep or deep-sleep modes. See Table 3-16 for details.

Table 3-16. PSPR Bit Definitions

Physical Address 0x40F0_0008 — PSPR — Clocks and Power Manager

	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
User Settings																																
Bit	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
	SP31	SP30	SP29	SP28	SP27	SP26	SP25	SP24	SP23	SP22	SP21	SP20	SP19	SP18	SP17	SP16	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
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†

Bits	Access	Name	Description
31:0	R/W	SP[n]†	Scratch Pad Register bit n, where n = [31:0]

NOTES:
† Exit from sleep or deep-sleep mode does not clear or set this bit.

3.8.1.4 Power Manager Wake-Up Enable Register (PWER)

PWER, defined in Table 3-17, selects whether the corresponding wake-up sources cause a wake-up from standby, sleep, or deep-sleep mode. Only GPIO<3, 1:0> can cause a wake-up from deep-sleep mode. Possible wake-up sources that can be programmed in PWER are RTC alarm, timer event, USB host wake-up event, USB client wake-up event, MSL port, USIM card insertion detection and GPIO<35, 15:9, 4:3, 1:0> wake-ups. Also, one of GPIO<31, 113> and one of GPIO<53, 40, 38, 36> can be programmed to cause a wake-up from standby or sleep.

Additional wake-up sources for standby and sleep are available through the PKWR register, see Section 3.8.1.15.

For a GPIO to serve as a wake-up source from these modes:

- It must be programmed as an input in the GPIO Pin-Direction registers (see Section 24.5.1 on page 24-11).
- Either or both of the corresponding bits in PRER and PFER must be set.

Refer to the *Intel® PXA27x Processor Family EMTS* for GPIO timing specifications.

When nVDD_FAULT or nBATT_FAULT is asserted, PWER assumes its reset value, enabling only GPIO<1:0> as wake-up sources.

Note: If a wake-up source is programmed as an output, the corresponding bit in PWER must be clear. Otherwise, unpredictable behavior occurs; unknown values can be read from PFER and PRER after waking up from the mode.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-17. PWER Bit Definitions (Sheet 1 of 3)

Physical Address 0x40F0_000C — **PWER** — **Clocks and Power Manager**

Bit	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
	WERTC	WEP1	reserved	WEUSBH2	WEUSBH1	WEUSBC	WBB	WE35	WEUSIM	reserved		WEMUX3	WEMUX2				WE15	WE14	WE13	WE12	WE11	WE10	WE9	reserved				WE4	WE3	reserved	WE1	WE0
Reset	0†	0†	?	0†	0†	0†	0†	0†	0†	?	?	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	?	?	?	?	0†	0†	0†	1†	1†

Bits	Access	Name	Description
31	R/W	WERTC†	Wake-up Enable for RTC Standby, Sleep, or Deep-Sleep Mode 0 = Disable wake-up due to RTC alarm. 1 = Enable wake-up due to RTC alarm.
30	R/W	WEP1†	Wake-up Enable for PI Power Domain Standby or Sleep Mode 0 = Disable wake-up due to timer wake-up event. 1 = Enable wake-up due to timer wake-up event.
29	—	—	reserved
28	R/W	WEUSBH2†	Wake-up Enable for USB Host Port 2 Standby or Sleep Mode 0 = Disable wake-up due to USB host port 2. 1 = Enable wake-up due to USB host port 2.

Table 3-17. PWER Bit Definitions (Sheet 2 of 3)

Physical Address 0x40F0_000C — **PWER** — **Clocks and Power Manager**

	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
User Settings																																
Bit	WERTC	WEP1	reserved	WEUSBH2	WEUSBH1	WEUSBC	WBB	WE35	WEUSIM	reserved		WEMUX3		WEMUX2			WE15	WE14	WE13	WE12	WE11	WE10	WE9	reserved				WE4	WE3	reserved	WE1	WE0
Reset	0†	0†	?	0†	0†	0†	0†	0†	0†	?	?	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	?	?	?	?	0†	0†	0†	1†	1†

Bits	Access	Name	Description
27	R/W	WEUSBH1†	Wake-up Enable for USB Host Port 1 Standby or Sleep Mode 0 = Disable wake-up due to USB host port 1. 1 = Enable wake-up due to USB host port 1.
26	R/W	WEUSBC†	Wake-up Enable for USB Client Port Standby or Sleep Mode 0 = Disable wake-up due to USB client port. 1 = Enable wake-up due to USB client port.
25	R/W	WBB†	Wake-up Enable for a Rising Edge from MSL (GPIO<83>) for Standby or Sleep Mode 0 = Disable wake-up due to MSL port. 1 = Enable wake-up due to MSL port.
24	R/W	WE[35]†	Wake-up Enable for Standby or Sleep Mode 0 = Disable wake-up due to GPIO<35> edge detect. 1 = Enable wake-up due to GPIO<35> edge detect.
23	R/w	WEUSIM†	Wake-up Enable for Rising or Falling Edge from UDET (GPIO<116>) for Standby or Sleep Mode 0 = Disable wake-up due to USIM card detect port. 1 = Enable wake-up due to USIM card detect port.
22:21	—	—	reserved
20:19	R/W	WEMUX3†	Wake-up Enable for GPIO<31> or GPIO<113> for Standby and Sleep Modes Both rising and falling edges are recognized as wake-ups. 00 = No wake-up enabled 01 = Enable wake-up due to GPIO<31> 10 = Enable wake-up due to GPIO<113 > 11 = Reserved **NOTE:** Only one of these two GPIOs can be configured to cause the processor to wake up at any given time.
18:16	R/W	WEMUX2†	Wake-up Enable for GPIO<36>, GPIO<38>, GPIO<40>, or GPIO<53> for Standby and Sleep Modes Both rising and falling edge is recognized as wake-up 000 = No wake-up enabled 001 = Enable Wake-up due to GPIO<38> 010 = Enable Wake-up due to GPIO<53> 011 = Enable Wake-up due to GPIO<40> 100 = Enable Wake-up due to GPIO<36> Other settings = Reserved **NOTE:** Only one of these four GPIOs can be configured to cause the processor to wake up at any given time.

Table 3-17. PWER Bit Definitions (Sheet 3 of 3)

Physical Address 0x40F0_000C — **PWER** — **Clocks and Power Manager**

	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
User Settings																																
Bit	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
	WERTC	WEP1	reserved	WEUSBH2	WEUSBH1	WEUSBC	WBB	WE35	WEUSIM	reserved		WEMUX3		WEMUX2			WE15	WE14	WE13	WE12	WE11	WE10	WE9	reserved				WE4	WE3	reserved	WE1	WE0
Reset	0†	0†	?	0†	0†	0†	0†	0†	0†	?	?	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	?	?	?	?	0†	0†	0†	1†	1†

Bits	Access	Name	Description
15:9	R/W	WE[n]†	Wake-up Enable n for Standby or Sleep, n = 9 to 15 0 = Disable wake-up due to GPIO<n> edge detect. 1 = Enable wake-up due to GPIO<n> edge detect.
8:5	—	—	reserved
4	R/W	WE4	Wake-up Enable for Standby or Sleep Modes 0 = Disable wake-up due to GPIO<4> edge detect. 1 = Enable wake-up due to GPIO<4> edge detect.
3	R/W	WE3	Wake-up Enable for Standby, Sleep, or Deep-sleep Modes 0 = Disable wake-up due to GPIO<3> edge detect. 1 = Enable wake-up due to GPIO<3> edge detect.
2	—	—	reserved
1:0	R/W	WE[n]	Wake-up Enable n for Standby, Sleep, or Deep-sleep Modes, n = 0 to 1 0 = Disable wake-up due to GPIO<n> edge detect. 1 = Enable wake-up due to GPIO<n> edge detect.

NOTES:

† Exit from sleep or deep-sleep mode does not clear or set this bit.

3.8.1.5 Power Manager Rising-Edge Detect Enable Register (PRER)

PRER, defined in Table 3-18, selects whether the GPIO pin enabled in PWER causes a wakeup when a rising edge is detected on that pin. When nVDD_FAULT or nBATT_FAULT is asserted, PRER assumes its reset value, enabling rising edges on GPIO<1:0> to act as wake-up sources.

Note: Refer to the *Intel® PXA27x Processor Family EMTS*, "GPIO AC Timing Specifications" for the minimum pulse duration to guarantee edge detection.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-18. PRER Bit Definitions

Physical Address 0x40F0_0010 — **PRER** — **Clocks and Power Manager**

	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
User Settings																																
Bit	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							RE35	reserved								RE15	RE14	RE13	RE12	RE11	RE10	RE9	reserved				RE4	RE3	reserved	RE1	RE0
Reset	?	?	?	?	?	?	?	0†	?	?	?	?	?	?	?	?	0†	0†	0†	0†	0†	0†	0†	?	?	?	?	0†	0†	?	1†	1†

Bits	Access	Name	Description
31:25	—	—	reserved
24	R/W	RE35†	Standby or Sleep-mode Rising-Edge Wake-Up Enable 0 = Disable wake-up due to GPIO<35> rising-edge detect. 1 = Enable wake-up due to GPIO<35> rising-edge detect.
23:16	—	—	reserved
15:9	R/W	RE[n]†	Standby or Sleep Mode Rising-Edge Wake-Up Enable n, where n = 15:9 0 = Disable wake-up due to GPIO<n> rising-edge detect. 1 = Enable wake-up due to GPIO<n> rising-edge detect.
8:5	—	—	reserved
4	R/W	RE[4]†	Standby or Sleep Mode Rising-Edge Wake-Up Enable 0 = Disable wake-up due to GPIO<4> rising-edge detect. 1 = Enable wake-up due to GPIO<4> rising-edge detect.
3	R/W	RE[3]†	Standby, Sleep, or Deep-sleep Rising-Edge Wake-Up Enable 0 = Disable wake-up due to GPIO<3> rising-edge detect. 1 = Enable wake-up due to GPIO<3> rising-edge detect.
2	—	—	reserved
1:0	R/W	RE[n]†	Standby, Sleep, or Deep-Sleep Rising-Edge Wake-Up Enable, n = 0 to 1 0 = Disable wake-up due to GPIO<n> rising-edge detect. 1 = Enable wake-up due to GPIO<n> rising-edge detect.

NOTES:
† Exit from sleep or deep-sleep mode does not clear or set this bit.

3.8.1.6 Power Manager Falling-Edge Detect Enable Register (PFER)

PFER, defined in Table 3-19, selects whether the GPIO pin enabled in PWER causes a wakeup when a falling edge is detected on that pin. When nVDD_FAULT or nBATT_FAULT is asserted, PFER assumes its reset value, enabling falling edges on GPIO<1:0> to act as wake-up sources. Refer to the *Intel® PXA27x Processor Family EMTS* for GPIO timing specifications.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-19. PFER Bit Definitions

Physical Address 0x40F0_0014 — **PFER** — **Clocks and Power Manager**

	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
User Settings																																
Bit	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							RE35	reserved								RE15	RE14	RE13	RE12	RE11	RE10	RE9	reserved				RE4	RE3	reserved	RE1	RE0
Reset	?	?	?	?	?	?	?	0†	?	?	?	?	?	?	?	?	0†	0†	0†	0†	0†	0†	0†	?	?	?	?	0†	0†	?	1†	1†

Bits	Access	Name	Description
31:25	—	—	reserved
24	R/W	RE35†	Standby or Sleep Mode Falling-Edge Wake-Up Enable 0 = Disable wake-up due to GPIO<35> falling-edge detect. 1 = Enable wake-up due to GPIO<35> falling-edge detect.
23:16	—	—	reserved
15:9	R/W	FE[n]†	Standby or Sleep Mode Falling-Edge Wake-Up Enable n, where n = 15:9 0 = Disable wake-up due to GPIO<n> falling-edge detect. 1 = Enable wake-up due to GPIO<n> falling-edge detect.
8:5	—	—	reserved
4	R/W	FE[n]†	Standby or Sleep Falling-Edge Wake-Up Enable, n = 4 0 = Disable wake-up due to GPIO<n> falling-edge detect. 1 = Enable wake-up due to GPIO<n> falling-edge detect.
3	R/W	FE[n]†	Standby or Sleep Falling-Edge Wake-Up Enable, n = 3 0 = Disable wake-up due to GPIO<n> falling-edge detect. 1 = Enable wake-up due to GPIO<n> falling-edge detect.
2	—	—	reserved
1:0	R/W	FE[n]†	Standby or Sleep Falling-Edge Wake-Up Enable, n =0 to 1 0 = Disable wake-up due to GPIO<n> falling-edge detect. 1 = Enable wake-up due to GPIO<n> falling-edge detect.

NOTES:
† Exit from sleep or deep-sleep mode does not clear or set this bit.

3.8.1.7 Power Manager Edge-Detect Status Register (PEDR)

PEDR, defined in Table 3-20, indicates which GPIO pin (enabled through the PWER, PRER and PFER registers) caused a wakeup from standby, sleep or deep-sleep mode. These bits can be set only by a rising edge, falling edge, or either on the given GPIO pin, depending on the settings in the PRER and PFER registers. These bits are cleared by writing 0b1 to them. Writing 0b0 to any status bit has no effect. Refer to the *Intel® PXA27x Processor Family EMTS* for GPIO timing specifications.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-20. PEDR Bit Definitions (Sheet 1 of 2)

Physical Address 0x40F0_0018 — **PEDR** — **Clocks and Power Manager**

Bit	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
	EDRTC	EDP1	reserved	EDUSBH2	EDUSBH1	EDUSBC	EDBB	ED35	reserved			EDMUX3	reserved		EDMUX2	reserved	ED15	ED14	ED13	ED12	ED11	ED10	ED9	reserved				ED4	ED3	reserved	ED1	ED0
Reset	0†	0†	?	0†	0†	0†	0†	0†	?	?	?	0†	?	?	0†	?	0†	0†	0†	0†	0†	0†	0†	?	?	?	?	0†	0†	?	0†	0†

Bits	Access	Name	Description
31	R/W	EDRTC	Standby, Sleep, or Deep-Sleep Wake-Up from RTC 0 = Wake-up due to RTC source not detected. 1 = Wake-up due to RTC source is detected.
30	R/W	EDP1	Standby or Sleep Wake-Up from PI Power Domain 0 = Wake-up due to PI source not detected. 1 = Wake-up due to PI source is detected
29	—	—	reserved
28	R/W	EDUSBH2	Standby or Sleep Wake-Up from USB Host Port 2 0 = Wake-up due to USB host port 2 not detected. 1 = Wake-up due to USB host port 2 is detected.
27	R/W	EDUSBH1	Standby or Sleep Wake-Up from USB Host Port 1 0 = Wake-up due to USB host port 1 not detected. 1 = Wake-up due to USB host port 1 is detected.
26	R/W	EDUSBC	Standby or Sleep Wake-Up from USB Client: 0 = Wake-up due to USB client not detected. 1 = Wake-up due to USB client is detected.
25	R/W	EDBB	Standby or Sleep Wake-Up from MSL 0 = Wake-up due to MSL not detected. 1 = Wake-up due to MSL is detected.
24	R/W	ED35	GPIO<35> Standby or Sleep Edge Detect Status 0 = Wake-up due to edge on GPIO<35> not detected. 1 = Wake-up due to edge on GPIO<35> is detected.
23:21	—	—	reserved
20	R/W	EDMUX3	Standby or Sleep Edge Detect Status, where n = the value programmed in PWER[WEMUX3] 0 = Wake-up due to edge on GPIO<n> not detected. 1 = Wake-up due to edge on GPIO<n> is detected.
19:18	—	—	reserved

Table 3-20. PEDR Bit Definitions (Sheet 2 of 2)

Physical Address 0x40F0_0018 — PEDR — Clocks and Power Manager

	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
User Settings																																
Bit	EDRTC	EDP1	reserved	EDUSBH2	EDUSBH1	EDUSBC	EDBB	ED35	reserved			EDMUX3	reserved		EDMUX2	reserved	ED15	ED14	ED13	ED12	ED11	ED10	ED9	reserved				ED4	ED3	reserved	ED1	ED0
Reset	0†	0†	?	0†	0†	0†	0†	0†	?	?	?	0†	?	?	0†	?	0†	0†	0†	0†	0†	0†	0†	?	?	?	?	0†	0†	?	0†	0†

Bits	Access	Name	Description
17	R/W	EDMUX2	Standby or Sleep Edge Detect Status, where n = the value programmed in PWER[WEMUX2] 0 = Wake-up due to edge on GPIO<n> not detected. 1 = Wake-up due to edge on GPIO<n> is detected.
16	—	—	reserved
15:9	R/W	ED[n]	Standby or Sleep Edge Detect Status, n = 9 to 15 0 = Wake-up due to edge on GPIO<n> not detected. 1 = Wake-up due to edge on GPIO<n> is detected.
8:5	—	—	reserved
4	R/W	ED4	Standby or Sleep Edge Detect Status 0 = Wake-up due to edge on GPIO<4> not detected. 1 = Wake-up due to edge on GPIO<4> is detected.
3	R/W	ED3	Standby, Sleep, or Deep-Sleep Edge Detect Status 0 = Wake-up due to edge on GPIO<3> not detected. 1 = Wake-up due to edge on GPIO<3> is detected.
2	—	—	reserved
1:0	R/W	ED[n]	Standby, Sleep, or Deep-Sleep Edge Detect Status, n = 0 through 1 0 = Wake-up due to edge on GPIO<n> not detected. 1 = Wake-up due to edge on GPIO<n> is detected.

NOTES:
† Exit from sleep or deep-sleep mode does not clear or set this bit.

3.8.1.8 Power Manager General Configuration Register (PCFR)

PCFR, defined in Table 3-21, contains the following bits to configure various functions within the processor:

- RDH Override (RO)—In sleep and deep-sleep modes, the receivers of all GPIO pins are disabled if RO is clear. The receivers of all GPIO pins are not disabled if the RO bit is set.
- PH Override (PO)—PSSR[PH] is set upon entry into standby and sleep mode if PO is clear. It indicates that the GPIO pins are retaining their states. If the PO bit is set, the PH bit is cleared automatically after exiting the low-power mode.
- GPIO Reset Disable (GPROD) enables/disables assertion of nRESET_OUT during GPIO reset.
- Sleep-Mode Sleep/Deep-Sleep Linear Regulator Enable (L1_EN) enables the sleep/deep-sleep linear regulator during sleep mode if other conditions are also met (see Section 3.6.2.3).

- Frequency/Voltage Change (FVC) controls initiation of the voltage-change sequence during a frequency change.
- Sleep/Deep-Sleep Dc-DC Converter Enable (DC_EN) controls use of the sleep/deep-sleep linear regulator and the PWR_CAP pins during sleep mode if other conditions are also met (see Section 3.6.2.3).
- Power Manager I^2C Enable (PI^2C_EN) controls usage of the power manager I^2C interface.
- nRESET_GPIO Pin Enable (GRP_EN) controls usage of the pin, either as GPIO or as a GPIO reset input.
- Float PC Card (FP) and Float Status (FS) control the state of the PC Card control pins and the static-memory control pins during sleep mode.
- Oscillator Power-Down Enable (OPDE) selects whether the 13-MHz processor oscillator is powered off in standby or sleep mode.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-21. PCFR Bit Definitions (Sheet 1 of 2)

Physical Address 0x40F0_001C — **PCFR** — **Clocks and Power Manager**

Bit	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
User Settings																																
	reserved																RO	PO	reserved	GPROD	L1_EN	FVC	reserved		DC_EN	PI^2C_EN	reserved	GPR_EN	reserved	FS	FP	OPDE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0†	0†	0†	?	?	0†	0†	?	0†	?	0†	0†	0†

Bits	Access	Name	Description
31:16	—	—	reserved
15	R/W	RO	RDH Override Overrides the default value of PSSR[RDH] in sleep and deep-sleep modes. 0 = The receivers of all GPIO pins are disabled. 1 = The receivers of all GPIO pins are enabled.
14	R/W	PO	PH Override Overrides the default value of PSSR[PH]. 0 = PSSR[PH] is set upon entry into standby and sleep mode, which indicates that the GPIO pins are retaining their states. 1 = PSSR[PH] is automatically cleared after exiting the low-power mode.
13	—	—	reserved
12	R/W	GPROD	GPIO nRESET_OUT Disable 0 = nRESET_OUT is asserted during GPIO reset. 1 = nRESET_OUT is not asserted during GPIO reset. **NOTE:** GPIO reset does not clear this bit.
11	R/W	L1_EN	Sleep Mode Sleep/Deep-Sleep Linear Regulator Enable 0 = The sleep/deep-sleep linear regulator is not used. 1 = The sleep/deep-sleep linear regulator is used if all enable conditions are met. **NOTE:** Do **not** set both L1_EN and DC_EN simultaneously.

Table 3-21. PCFR Bit Definitions (Sheet 2 of 2)

Physical Address 0x40F0_001C — **PCFR** — **Clocks and Power Manager**

	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
User Settings																																
Bit	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																RO	PO	reserved	GPROD	L1_EN	FVC	reserved		DC_EN	PI²C_EN	reserved	GPR_EN	reserved	FS	FP	OPDE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0†	0†	0†	?	?	0†	0†	?	0†	?	0†	0†	0†

Bits	Access	Name	Description
10	R/W	FVC	Frequency/Voltage Change 0 = The voltage-change sequencer is not initiated during a frequency change. 1 = The voltage-change sequencer is initiated during a frequency change if power manager I²C is enabled.
9:8	—	—	reserved
7	R/W	DC_EN	Sleep/Deep-Sleep DC-DC Converter Enable 0 = The DC-DC converter is not used. 1 = The DC-DC converter is used if all enable conditions are met. **NOTE:** Do **not** set both L1_EN and DC_EN simultaneously.
6	R/W	PI²C_EN	Power Manager I²C Enable 0 = Power manager I²C is not used; the PWR_SDA and PWR_SCL pins can be used as GPIO. 1 = The PWR_SDA and PWR_SCL pins are driven by power manager I²C, regardless of the GPIO configuration.
5	—	—	reserved
4	R/W	GPR_EN	nRESET_GPIO Pin Enable 0 = nRESET_GPIO is not used; the pin can be used as GPIO. 1 = The nRESET_GPIO pin is use as a GPIO reset input, regardless of the GPIO configuration.
3	—	—	reserved
2	R/W	FS	Float Static Chip Selects (nCS<5:1>) During Sleep Mode 0 = The static chip select pins are driven by the appropriate PGSRx bits. 1 = The static chip select pins are floated in sleep or deep-sleep mode. Float Static Chip Selects (nCS<0>) During Sleep Mode 0 = The static chip select pin is at the pre-sleep defined value. 1 = The static chip select pins are floated in sleep or deep-sleep mode.
1	R/W	FP	Float PC Card Pins During Sleep or Deep-Sleep Mode 0 = The PC Card pins are driven by the appropriate PGSRx bits. 1 = The nPOE, nPWE, nPIOW, nPIOR, nPCE<2:1>, PSKTSEL, and nPREG pins are floated during sleep or deep-sleep mode.
0	R/W	OPDE	13-MHz Processor Oscillator Power-Down Enable 0 = Do not stop the oscillator during standby, sleep, or deep-sleep mode. 1 = Stop the oscillator during standby, sleep, or deep-sleep mode if bit OOK in the Oscillator Configuration register is set.

NOTES:
† Exit from sleep or deep-sleep mode does not clear or set this bit.

3.8.1.9 Power Manager GPIO Sleep-State Registers (PGSRx)

PGSR0, PGSR1, PGSR2, and PGSR3, defined in Table 3-22, allow for selecting the output state (the driven value) of each GPIO pin when the processor enters and while it is in sleep mode, or when it enters deep-sleep mode (until the power supplies are removed). The values programmed in PGSRx are not driven out during standby mode.

When a transition to sleep or deep-sleep mode is required (either through software or through the assertion of the nBATT_FAULT or nVDD_FAULT pins), the contents of the PGSRx registers are loaded into the GPIO Output Data registers, which are normally controlled by software through the GPSR (set) and GPCR (clear) registers. Only the pins already configured as outputs reflect the new state; however, all bits of the output registers are loaded. After the processor re-enters normal mode from sleep or deep-sleep mode, these GPIO pins retain their programmed sleep state until software clears PSSR[PH]. If PSSR[PH] is clear and a pin direction switches from input to an output, the pin is floated.

These registers are reset upon sleep or deep-sleep exit, so the configuration of the GPIO pins must be read from external memory. However, the GPIO state itself is not changed until software resets PSSR[PH].

In deep-sleep mode, GPIO<120:11> float if the external high-voltage power domains are removed. (GPIO<120:119> are available on the PXA271, PXA272 processors only.)

If these supplies remain enabled, the GPIOs default to the pullup, pulldown reset state. Refer to the *Intel® PXA27x Processor Family EMTS* Pin Usage section for reset states and the “GPIO States in Deep-Sleep Mode” section for timing details.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-22. PGSR0/1/2/3 Bit Definitions

Physical Address				Clocks and Power Manager
0x40F0_0020	PGSR0			
0x40F0_0024	PGSR1			
0x40F0_0028	PGSR2			
0x40F0_002C	PGSR3			

	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
User Settings																																
Bit	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
PGSR0	SS31	SS30	SS29	SS28	SS27	SS26	SS25	SS24	SS23	SS22	SS21	SS20	SS19	SS18	SS17	SS16	SS15	SS14	SS13	SS12	SS11	SS10	SS9	SS8	SS7	SS6	SS5	SS4	SS3	SS2	SS1	SS0
PGSR1	SS63	SS62	SS61	SS60	SS59	SS58	SS57	SS56	SS55	SS54	SS53	SS52	SS51	SS50	SS49	SS48	SS47	SS46	SS45	SS44	SS43	SS42	SS41	SS40	SS39	SS38	SS37	SS36	SS35	SS34	SS33	SS32
PGSR2	SS95	SS94	SS93	SS92	SS91	SS90	SS89	SS88	SS87	SS86	SS85	SS84	SS83	SS82	SS81	SS80	SS79	SS78	SS77	SS76	SS75	SS74	SS73	SS72	SS71	SS70	SS69	SS68	SS67	SS66	SS65	SS64
PGSR3	reserved							SS120††	SS119††	SS118	SS117	SS116	SS115	SS114	SS113	SS112	SS111	SS110	SS109	SS108	SS107	SS106	SS105	SS104	SS103	SS102	SS101	SS100	SS99	SS98	SS97	SS96
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

Bits	Access	Name	Description
31:0	R/W	SS[n]	Sleep State of GPIO<n> 0 = If programmed as an output, the pin is driven and held low during the transition to and while in sleep mode or is driven and held low during the transition to deep-sleep mode. 1 = If programmed as an output, the pin is driven and held high during the transition to and while in sleep mode or is driven and held high during the transition to deep-sleep mode.

NOTES:

† Reset values of reserved bits are undefined.

†† Supported on the PXA271, PXA272 processors only.

3.8.1.10 Reset Controller Status Register (RCSR)

RCSR, defined in Table 3-23, indicates the last causes of a reset. Table 3-2 details the behavior of different processor modules during each of the reset sources: power on, hardware, watchdog, sleep or deep sleep, and GPIO.

Each RCSR status bit is set by its specific source of reset. To clear a bit, write 0b1 to it.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-23. RCSR Bit Definitions

Physical Address 0x40F0_0030	RCSR	Clocks and Power Manager

User Settings																															
Bit	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																												GPR	SMR	WDR	HWR
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0[1]	0[1]	0[1]	0[1]

Bits	Access	Name	Description
31:4	—	—	reserved
3	R/WC[2]	GPR	GPIO Reset 0 = GPIO reset has not occurred since the last time software or power-on reset cleared this bit. 1 = GPIO reset has occurred since the last time software or power-on reset cleared this bit.
2	R/WC[2]	SMR	Sleep-Exit Reset from Sleep or Deep-Sleep Mode 0 = Sleep-exit reset has not occurred since the last time software or power-on reset cleared this bit. 1 = Sleep-exit reset has occurred since the last time software or power-on reset cleared this bit.
1	R/WC[2]	WDR	Watchdog Reset 0 = Watchdog reset has not occurred since the last time software or power-on reset cleared this bit. 1 = Watchdog reset has occurred since the last time software or power-on reset cleared this bit.
0	R/WC[2]	HWR	Hardware/Power-On Reset 0 = Hardware or power-on reset has not occurred since the last time software or power-on reset cleared this bit. 1 = hardware or power-on reset has occurred since the last time software or power-on reset cleared this bit.

NOTES:
1. Power-on reset clears this bit.
2. To clear this bit, write 0b1 to it.

3.8.1.11 Power Manager Sleep Configuration Register (PSLR)

PSLR, defined in Table 3-24, provides the following functions that control the processor behavior in both sleep and deep-sleep modes:

- System Power-Supply Ramp Delay (SYS_DEL) adjusts the power-on time (in 32.768-kHz timekeeping oscillator cycles) from the assertion of SYS_EN to the assertion of PWR_EN.
- Power-Supply Ramp Delay (PWR_DEL) adjusts the power-on time (in 32.768-kHz timekeeping oscillator cycles) from the assertion of PWR_EN to the continuation of the sleep-exit sequence.
- Sleep-Mode Shorten Wake-up Delay Disable (PSSD) shortens the sleep-mode wake-up delay when all power supplies have powered on.
- Ignore nVDD_FAULT in Sleep Mode (IVF) prevents nVDD_FAULT from affecting the sleep or deep-sleep state.

- Sleep-Mode nRESET_OUT Disable (SL_ROD) prevents the nRESET_OUT pin from asserting upon entry into from sleep or deep-sleep.
- Sleep-Mode Unit Retention bits (SL_Rn) select which units retain their states in sleep. All units except the RTC and power manager are reset; state retention affects only memories and the PI power domain.

See Table 3-11 for more details about the possible states of each unit in sleep mode.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-24. PSLR Bit Definitions (Sheet 1 of 2)

Physical Address 0x40F0_0034 — **PSLR** — **Clocks and Power Manager**

Bit	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
User Settings																																
	SYS_DEL				PWR_DEL				PSSD	IVF	reserved	SL_ROD	reserved								SL_R3	SL_R2	SL_R1	SL_R0	reserved				SL_PI		reserved	
Reset	1[1]	1[1]	0[1]	0[1]	1[1]	1[1]	0[1]	0[1]	0[1]	0[1]	?	0[1]	?	?	?	?	?	?	?	?	0[1]	0[1]	0[1]	0[1]	?	?	?	?	0[1]	0[1]	?	?

Bits	Access	Name	Description
31:28	R/W	SYS_DEL	External High-Voltage Power Domains Ramp Delay Sets the number of 32.768-kHz timekeeping oscillator cycles between assertion of SYS_EN (system power supplies enabled) and assertion of PWR_EN to 2^n (reset value = 0b1100 for 125 ms delay). 0b0000-0b1100: n = SYS_DEL 0b1101-0b1111: n = 12
27:24	R/W	PWR_DEL	External Low-Voltage Power Domains Ramp Delay Sets the number of 32.768-kHz timekeeping oscillator cycles between assertion of PWR_EN and the sleep or deep-sleep recovery sequence to 2^n (reset value = 0b1100 for 125 ms delay). 0b0000-0b1100: n = PWR_DEL 0b1101-0b1111: n = 12
23	R/W	PSSD	Sleep-Mode Shorten Wake-up Delay Disable 0 = Do not shorten the wake-up delay (SYS_DEL or PWR_DEL) if all the corresponding power supplies are detected to have powered on. 1 = Shorten the wake-up delay (SYS_DEL or PWR_DEL) if all the corresponding power supplies are detected to have powered on.
22	R/W	IVF	Ignore nVDD_FAULT in Sleep Mode and Deep-Sleep Mode 0 = nVDD_FAULT is not gated off in sleep and deep-sleep mode. 1 = No action taken when nVDD_FAULT occurs in sleep or deep-sleep mode. The nVCC_FAULT is gated off when IVF is set and PWR_EN is deasserted.
21	—	—	reserved
20	R/W	SL_ROD	Sleep-Mode/Deep-Sleep Mode nRESET_OUT Disable 0 = nRESET_OUT is asserted upon entry into sleep or deep-sleep mode. 1 = nRESET_OUT is not asserted upon entry into sleep or deep-sleep mode, but the CPU and on-chip peripherals are reset.
19:12	—	—	reserved

Table 3-24. PSLR Bit Definitions (Sheet 2 of 2)

Physical Address 0x40F0_0034 — **PSLR** — **Clocks and Power Manager**

	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
User Settings																																
Bit	SYS_DEL				PWR_DEL				PSSD	IVF	reserved	SL_ROD	reserved								SL_R3	SL_R2	SL_R1	SL_R0	reserved				SL_PI		reserved	
Reset	1[1]	1[1]	0[1]	0[1]	1[1]	1[1]	0[1]	0[1]	0[1]	0[1]	?	0[1]	?	?	?	?	?	?	?	?	0[1]	0[1]	0[1]	0[1]	?	?	?	?	0[1]	0[1]	?	?

Bits	Access	Name	Description
11	R/W	SL_R3[2]	Sleep Mode Unit Retention—Internal SRAM Bank 3 0 = SRAM bank 3 is powered off in sleep mode. 1 = SRAM bank3 retains state (memory array only) in sleep mode.
10	R/W	SL_R2[2]	Sleep Mode Unit Retention—Internal SRAM Bank 2 0 = SRAM bank 2 is powered off in sleep mode. 1 = SRAM bank 2 retains state (memory array only) in sleep mode.
9	R/W	SL_R1[2]	Sleep Mode Unit Retention—Internal SRAM Bank 1 0 = SRAM bank 1 is powered off in sleep mode. 1 = SRAM bank 1 retains state (memory array only) in sleep mode.
8	R/W	SL_R0[2]	Sleep Mode Unit Retention—Internal SRAM Bank 0 0 = SRAM bank 0 is powered off in sleep mode. 1 = SRAM bank 0 retains state (memory array only) in sleep mode.
7:4	—	—	reserved
3:2	R/W	SL_PI	Sleep or Deep-sleep Mode Unit Retention—PI Power Domain 00 = The PI power domain is powered off in sleep and deep-sleep mode. 01 = The PI power domain retains state in sleep and deep-sleep mode 10 = The PI power domain is active with clocks running during sleep mode. **Do not use this setting for deep-sleep mode.** 11 = reserved
1:0	—	—	reserved

NOTES:
1. Exit from sleep or deep-sleep mode does not clear or set this bit.
2. If this bit is set, then the PMCR[xIDAE] bits must also be set. Do not set this bit when entering deep-sleep mode.

3.8.1.12 Power Manager Standby Configuration Register (PSTR)

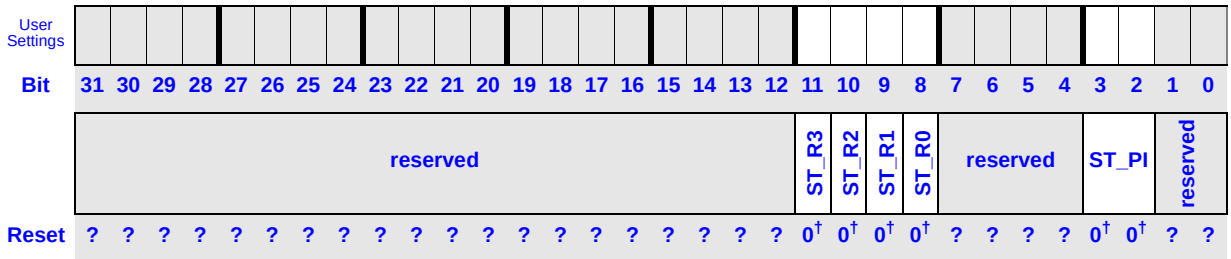
PSTR, defined in Table 3-25, contains the standby-mode unit-retention bits. These bits select which units retain state and operation in standby mode. Registers of disabled units take their reset values at standby exit. The CPU and most peripherals retain state in standby mode.

See Table 3-11 for more details about the possible states of each unit in standby mode.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-25. PSTR Bit Definitions

Physical Address 0x40F0_0038	PSTR	Clocks and Power Manager



Bits	Access	Name	Description
31:12	—	—	reserved
11	R/W	ST_R3	Standby-Mode Retain State —Internal SRAM Bank 3 0 = SRAM bank 3 is powered off in standby mode. 1 = SRAM bank 3 retains state in standby mode.
10	R/W	ST_R2	Standby-Mode Retain State—Internal SRAM Bank 2 0 = SRAM bank 2 is powered off in standby mode. 1 = SRAM bank 2 retains state in standby mode.
9	R/W	ST_R1	Standby-Mode Retain State—Internal SRAM Bank 1 0 = SRAM bank 1 is powered off in standby mode. 1 = SRAM bank 1 retains state in standby mode.
8	R/W	ST_R0	Standby-Mode Retain State—Internal SRAM Bank 0 0 = SRAM bank 0 is powered off in standby mode. 1 = SRAM bank 0 retains state in standby mode.
7:4	—	—	reserved
3:2	R/W	ST_PI	Standby-Mode Retain State—PI Power Domain 00 = The PI power domain is powered off in standby mode. 01 = The PI power domain retains state in standby mode. 10 = The PI power domain is active with clocks running in standby mode. 11 = reserved
1:0	—	—	reserved

NOTES:
† Exit from sleep or deep-sleep mode does not clear or set this bit.

3.8.1.13 Power Manager Voltage Change Control Register (PVCR)

PVCR contains configuration bits that control the automatic voltage-change sequence, as described in Table 3-26).

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-26. PVCR Bit Definitions

Physical Address 0x40F0_0040 — **PVCR** — **Clocks and Power Manager**

Bit	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								Read Pointer				reserved					VCSA	reserved		Command Delay				Slave Address							
Reset	?	?	?	?	?	?	?	0†	0†	0†	0†	0†	?	?	?	?	?	0†	?	?	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†

Bits	Access	Name	Description
31:25	—	—	reserved
24:20	R/W	Read Pointer	A programmed value of N indicates that the voltage-change sequencer is pointing to register PCMD[N]. When written, N points to the PCMDx register that contains the first command of the sequence. When read, N points to the PCMDx register that is to be or is being transmitted.
19:15	—	—	reserved
14	R	VCSA	Voltage-Change Sequencer Active 0 = No voltage-change sequence is in progress. 1 = The voltage-change sequencer is actively communicating with the power manager I^2C bus and the external regulator. Software cannot read from or write to the power manager I^2C registers.
13:12	—	—	reserved
11:7	R/W	Command Delay	Command Delay When DCE is set in a PCMD register, the command waits M number of 13-MHz processor oscillator cycles before execution, allowing delay between I^2C commands. $M = 2^N$, where N is the value programmed as Command delay for 0b00000 < N < 0b11001 $M = 2^{24}$ for N ≥ 0b11001
6:0	R/W	Slave Address	Slave Address The seven-bit address of the external regulator's I^2C module.

NOTES:
† Exit from sleep or deep-sleep mode does not clear or set this bit if the PI power domain is not powered off.

3.8.1.14 Power Manager USIM Card Control/Status Register (PUCR)

PUCR, defined in Table 3-27, contains bits for automatic detection of the USIM card. When the EN_UDET bit is set, the processor automatically detects insertion or removal of the USIM card if a rising or falling edge is seen on the UDET (GPIO<11>] alternate function) pin. Depending on the USIM114 or USIM115 bits, the UEN card-control signal is asserted or de-asserted.

The USIM card auto-detect mechanism is available in all power modes except deep-sleep mode. See Section 3.8.1.4 regarding wake-up from sleep and standby mode in response to a falling or rising edge on UDET.

- USIM Detect Status (UDETS)—If a rising or falling edge is detected on UDET, then the previous status is toggled. Software can also forcefully set or reset the status of the card. If UDETS is set, the processor asserts the UEN signal, depending on the USIM115 or USIM114 bits.
- USIM115 allows UEN functionality for GPIO<115>.
- USIM114 allows UEN functionality for GPIO<114>.
- Enable USIM Card Detect (EN_UDET) enables automatic detection of the USIM card based on the UDET signal's rising or falling edge.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-27. PUCR Bit Definitions (Sheet 1 of 2)

Physical Address 0x40F0_004C — **PUCR** — **Clocks and Power Manager**

	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
User Settings																																
Bit	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																										UDETS	Reserved	USIM115	USIM114	Reserved	EN_UDET
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0†	?	0†	0†	?	0†

Bits	Access	Name	Description
31:6	—	—	reserved
5	R/W	UDETS	USIM Detect Status If a rising or falling edge is detected on UDET, then the previous status is toggled. Software can also forcefully set or reset the status of the card. 0 = USIM card is not detected, deassert the nUVS1/UEN and UVS0/nUEN signals 1 = USIM card is detected, assert the nUVS1/UEN and UVS0/nUEN signals depending on the USIM115 and USIM114 bits.
4	—	—	reserved
3	R/W	USIM115	Allow UVS or UEN Functionality for GPIO<115> 0 = The pad is a GPIO 1 = The pad is used for UEN functionality regardless of the corresponding alternate function configuration in GAFR3_U[AF115]. The GPDR[PD115] bit must be set.

Table 3-27. PUCR Bit Definitions (Sheet 2 of 2)

Physical Address 0x40F0_004C — **PUCR** — **Clocks and Power Manager**

	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
User Settings																																
Bit	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																										UDETS	Reserved	USIM115	USIM114	Reserved	EN_UDET
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0†	?	0†	0†	?	0†

Bits	Access	Name	Description
2	R/W	USIM114	Allow UVS or UEN Functionality for GPIO<114> 0 = The pad is a GPIO 1 = The pad is used for UEN functionality regardless of the corresponding alternate function configuration in GAFR3_U[AF114]. The GPDR[PD114] bit must be set.
1	—	—	reserved
0	R/W	EN_UDET	Enable USIM Card Detect 0 = Disable USIM card detect. 1 = Enable USIM card detect. The GPIO<116> pad is used for UDET regardless of the alternate function configuration in GAFR3_U[AF116]. The GPDR3[PD116] bit must be cleared.

NOTES:

† Sleep and deep-sleep mode do not clear or set these bits.

3.8.1.15 Power Manager Keyboard Wake-Up Enable Register (PKWR)

PKWR, defined in Table 3-28, selects whether or not the corresponding keypad-related GPIO pin causes a wake-up from standby or sleep mode. To serve in this manner, the GPIO must be programmed as an input in the GPDR (see Section 24.5.1, "GPIO Pin-Direction Registers (GPDR)" on page 24-11). Refer to the *Intel® PXA27x Processor Family EMTS* for GPIO timing specifications. When nVDD_FAULT or nBATT_FAULT is asserted, PKWR assumes its reset value, enabling only GPIO<1:0> as wake-up sources and disabling all bits in PKWR.

Additional wake-up sources for standby and sleep are available through the PWER register (see Section 3.8.1.4).

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-28. PKWR Bit Definitions

Physical Address 0x40F0_0050 — **PKWR** — **Clocks and Power Manager**

	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
User Settings																																
Bit	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												WE102	WE101	WE100	WE99	WE98	WE97	WE96	WE95	WE94	WE93	WE91	WE90	WE39	WE38	WE37	WE36	WE34	WE17	WE16	WE13
Reset	?	?	?	?	?	?	?	?	?	?	?	?	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†

Bits	Access	Name	Description
25:16	—	—	reserved
19:0	R/W	WE[n]	Standby or Sleep-Mode Wake-Up Enable n 0 = Disable wake-up due to high level on GPIO<n>. 1 = Enable wake-up due to high level on GPIO<n>.

NOTES:
† Exit from sleep mode does not clear or set this bit.

3.8.1.16 Power Manager Keyboard Level-Detect Status Register (PKSR)

PKSR, defined in Table 3-29, indicates which GPIO pin caused a wake-up from standby or sleep mode. The GPIO pin must be enabled in PKWR (see Section 3.8.1.15).

These bits are set only by a high level on the GPIO pin. The pin must be configured as an input (see Section 24.5.1, "GPIO Pin-Direction Registers (GPDR)" on page 24-11) and as a keyboard input (see Section 24.5.4, "GPIO Alternate Function Register (GAFR)" on page 24-23).

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-29. PKSR Bit Definitions

Physical Address 0x40F0_0054												PKSR																Clocks and Power Manager							
User Settings																																	
Bit	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												ED102	ED101	ED100	ED99	ED98	ED97	ED96	ED95	ED94	ED93	ED91	ED90	ED39	ED38	ED37	ED36	ED34	ED17	ED16	ED13
Reset	?	?	?	?	?	?	?	?	?	?	?	?	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†

Bits	Access	Name	Description
31:20	—	—	reserved
19:0	R/WC††	LD[n]	Standby or Sleep Mode Level Detect Status n 0 = Do not detect wake-up due to high level on GPIO<n>. 1 = Detect wake-up due to high level on GPIO<n>.

NOTES:

† Exit from sleep or deep-sleep mode does not clear or set this bit.

†† To clear this bit, write 0b1 to it.

3.8.1.17 Power Manager I²C Command Register File (PCMDx)

The Power Manager I²C Command Register File is a set of 32 identical registers (PCMD0–31) that indicate the type of transaction to be completed on the power manager I²C bus during a voltage-change sequence. Each register represents a single command. Commands are executed starting with the PCMD register indicated by the read pointer field in PVCR. Each register contains a 13-bit field that holds five configuration bits and eight data bits. See Table 3-30 for bit definitions.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-30. PCMD0–31 Bit Definitions

Physical Address 0x40F0_0080 through 0x40F0_00FC	PCMD0 through PCMD31	Clocks and Power Manager

User Settings																															
Bit	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																			MBC	DCE	LC	SQC		Command Data							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†	0†

Bits	Access	Name	Description
31:13	—	—	reserved
12	R/W	MBC	Multi-Byte Command 0 = The corresponding data is sent either as a single-byte command or as the last byte of a multi-byte command. 1 = The corresponding data is part of a multi-byte command (except the last byte). The next byte (in the next higher order address PCMD) is sent without delay or handshaking with the power manager.
11	R/W	DCE	Delay Command Execution 0 = Execute the corresponding command without waiting for the delay counter to time out. 1 = Execute the corresponding command only after the delay counter has timed out.
10	R/W	LC	Last Command 0 = There are additional valid commands after this one. 1 = The command in this register is the last one.
9:8	R/W	SQC	Sequence Configuration 0b00 = Continue; the next command is automatically read. 0b01 = Pause*; the* next command is not executed until the power manager asserts a request. 0b10 = reserved 0b11 = reserved
7:0	R/W	Command Data	Power Manager I²C Command Data to be Sent to External Regulator

NOTES:
† Exit from sleep or deep-sleep mode does not clear or set this bit if the PI power domain is not powered off.

3.8.2 Clocks Manager Registers

The clocks manager uses the following registers:

- Section 3.8.2.1 — Core Clock Configuration Register (CCCR)
- Section 3.8.2.2 — Clock Enable Register (CKEN)
- Section 3.8.2.3 — Oscillator Configuration Register (OSCC)
- Section 3.8.2.4 — Core Clock Status Register (CCSR)
- Section 3.8.3.1 — Clock Configuration Register (CLKCFG), coprocessor 14

Table 3-39 summarizes the registers associated with the clocks manager and the physical addresses to access them.

3.8.2.1 Core Clock Configuration Register (CCCR)

The core clock is the base clock from which the CPU, memory-controller, LCD-controller, and system-bus frequencies are derived. CCCR, defined in Table 3-31, controls these frequencies with the following bits:

- Run-Mode-to-Oscillator Ratio (L) creates the nominal run-mode frequency by multiplying the 13-MHz processor oscillator by L.
- Turbo-Mode-to-Run-Mode Ratio (N) creates the nominal turbo-mode frequency by multiplying the run-mode frequency by N.

Note: CCCR[2N] must be programmed with *twice* the value of N.

- Peripheral PLL Disable (PPDIS) turns off the peripheral PLL when not needed. If disabled, the peripheral PLL is not selected as the clock source and renders many peripheral units nonfunctional. If the peripheral PLL is disabled, all peripherals use the 13-MHz clock source.
- Core PLL Disable (CPDIS) turns off the core PLL when not needed. If disabled, the core PLL is not selected as the clock source, and the 13-MHz clock is used.
- LCD Clock Frequency (LCD_26) selects between 13 and 26 MHz.
- Enable PLLs Early (PLL_EARLY_EN)—In 13M mode, enables the core and peripheral PLLs early, while remaining fully operational at 13 MHz. This feature can save time while waiting for the PLLs to lock.

These configurations are not loaded instantaneously; a frequency change is required to enact any changes. When these changes are enacted, they can be ignored if inconsistent (see Section 3.5.7.3.1 for restrictions on CPDIS and PPDIS). The actual value configured in the processor is reflected in the Core Clock Status register (see Section 3.8.2.4). When enabled, the critical frequencies are:

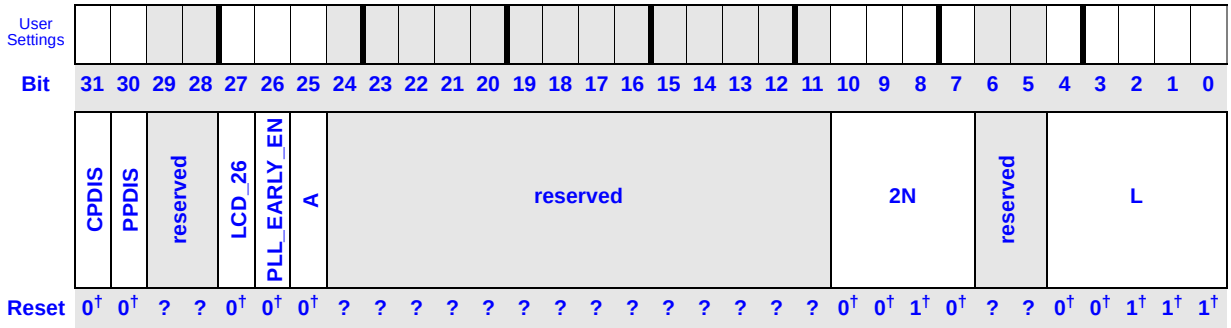
```
    Turbo-mode frequency (T) = 13-MHz processor-oscillator frequency * L * N

    Run-mode frequency (R) = 13-MHz processor-oscillator frequency * L

    System-bus frequency = 13-MHz processor-oscillator frequency * L / B,
        where B = 1 (when in fast-bus mode) or B = 2 (when not in fast-bus mode)

For CCCR[A] = 0 (see Table 3-7):

    Memory-controller frequency = 13-MHz processor-oscillator frequency * L / M,
        where M = 1 (L = 2-10), M = 2 (L = 11-20), or M = 4 (L = 21-31)

    LCD frequency = 13-MHz processor-oscillator frequency * L / K,
        where K = 1 (L = 2-7), K = 2 (L = 8-16), or K = 4 (L = 17-31)

For CLKCFG[B] = 0 and CCCR[A] = 1 (see Table 3-7):
```

```
    Memory-controller frequency = 13-MHz processor-oscillator frequency * L / 2

    LCD frequency = 13-MHz processor-oscillator frequency * L / K,
        where K = 1 (L = 2-7), K = 2 (L = 8-16), or K = 4 (L = 17-31)

For CLKCFG[B] = 1 and CCCR[A] = 1 (see Table 3-7):

    Memory-controller frequency = 13-MHz processor-oscillator frequency * L

    LCD frequency = 13-MHz processor-oscillator frequency * L / K,
        where K = 1 (L = 2-7), K = 2 (L = 8-16), or K = 4 (L = 17-31)
```

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-31. CCCR Bit Definitions (Sheet 1 of 2)

Physical Address 0x4130_0000 — **CCCR** — **Clocks and Power Manager**



Bits	Access	Name	Description
31	R/W	CPDIS	Core PLL Output Disable 0 = Core PLL is enabled after frequency change. 1 = Core PLL is disabled after frequency change if not selected as a clock source.
30	R/W	PPDIS	Peripheral PLL Output Disable 0 = Peripheral PLL is enabled after frequency change. 1 = Peripheral PLL is disabled after frequency change if not selected as a clock source.
29:28	—	—	reserved
27	R/W	LCD_26	LCD Clock Frequency in Deep-Idle or 13M Mode 0 = LCD clock frequency is 13 MHz. 1 = LCD clock frequency is 26 MHz.
26	R/W	PLL_EARLY_EN	Early PLL Enable In 13M mode, allows software to enable the PLLs ahead of time, while remaining fully operational in 13M mode. When the core and peripheral PLLs have locked (CCSR[CPLK] and CCSR[PPLK] set), software can perform the frequency change. This bit is automatically cleared after a frequency change exiting from 13M mode. 0 = Do not enable the core and peripheral PLLs early. 1 = Enable the core and peripheral PLLs early (all units and the core remain operational at 13 MHz). **NOTE:** Write to this bit only when the processor is in 13M mode and the core PLL has been disabled by setting CPDIS. In normal run mode, writing to this bit causes unpredictable results.

Table 3-31. CCCR Bit Definitions (Sheet 2 of 2)

Physical Address 0x4130_0000 — CCCR — Clocks and Power Manager

	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
User Settings																																
Bit	CPDIS	PPDIS	reserved		LCD_26	PLL_EARLY_EN	A	reserved														2N			reserved			L				
Reset	0†	0†	?	?	0†	0†	0†	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0†	0†	1†	0†	?	?	0†	0†	1†	1†	1†

Bits	Access	Name	Description
25	R/W	A	Alternate Setting for Memory Controller Clock 0 = Memory controller clock (CLK_MEM) clock frequency is as specified in Table 3-7. 1 = Memory controller clock (CLK_MEM) clock frequency is the same as the system bus frequency. ***Note:*** Refer to section 6.5.1.4 for changing CLK_MEM while SDCLK <1> or <2> is at 104 MHz
24:11	—	—	reserved
10:7	R/W	2N	Turbo-Mode-to-Run-Mode Ratio, N (Reset value 0b010 for N = 1) 0b000–0b010 = ratio (N) = 1 0b011–0b110 = ratio (N) = (2N / 2) 0b111 = reserved **NOTE:** Program this field with *twice* the value of N.
6:5	—	—	reserved
4:0	R/W	L	Run-Mode-to-Oscillator Ratio (Reset value 0b00111 for L=7) 0b00000–0b00010 = ratio = 2 0b00011–0b11110 = ratio = L 0b11111 = reserved

NOTES:
† GPIO, sleep, and deep-sleep resets do not clear or set this bit.

3.8.2.2 Clock Enable Register (CKEN)

CKEN enables or disables the clock to most of the peripheral units. For lowest power consumption, any unit that is not being used must have its clock disabled by clearing the appropriate bit. See Table 3-32 for register details and Table 3-33 for clock-enable-to-CKEN bit mappings.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-32. CKEN Bit Definitions

Physical Address 0x4130_0004																															
CKEN																															
Clocks and Power Manager																															
User Settings																															
Bit	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
	CKEN31	reserved						CKEN24	CKEN23	CKEN22	CKEN21	CKEN20	CKEN19	CKEN18	CKE717	CKEN16	CKEN15	CKEN14	CKEN13	CKEN12	CKEN11	CKEN10	CKEN9	CKEN8	CKEN7	CKEN6	CKEN5	CKEN4	CKEN3	CKEN2	CKEN1	CKEN0
Reset	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†

Bits	Access	Name	Description
31	R/W	CKEN[31]	AC ‘97 Controller Configuration This bit works in conjunction with CKEN[2], as described in Table 13-7, “AC ‘97 Configuration” on page 13-13.
30:25	—	—	reserved
24:0	R/W	CKEN[n]	Clock Enable 0 = Clock to the unit is disabled. 1 = Clock to the unit is enabled. See Table 3-33 for module mapping.

NOTES:
† GPIO, sleep, and deep-sleep resets do not clear or set this bit.

Table 3-33. Clock Enable Mappings for CKEN Bits

Name	Description	Name	Description
CKEN[25]	TPM Unit Clock Enable	CKEN[11]	USB Client Unit Clock Enable 48-MHz Clock Enable ***Note:*** Also enables the 48-MHz clock on alternate functions GPIO<11:!12>
CKEN[24]	Quick Capture Interface Clock Enable		
CKEN[23]	SSP1 Unit Clock Enable	CKEN[10]	USB Host Unit Clock Enable
CKEN[22]	Memory Controller	CKEN[9]	OS Timer Unit Clock Enable
CKEN[21]	Memory Stick Host Controller	CKEN[8]	I^2S Unit Clock Enable
CKEN[20]	Internal Memory Clock Enable	CKEN[7]	BTUART Unit Clock Enable
CKEN[19]	Keypad Interface Clock Enable	CKEN[6]	FFUART Unit Clock Enable
CKEN[18]	USIM Unit Clock Enable	CKEN[5]	STUART Unit Clock Enable
CKEN[17]	MSL Interface Unit Clock Enable	CKEN[4]	SSP3 Unit Clock Enable
CKEN[16]	LCD Controller Clock Enable	CKEN[3]	SSP2 Unit Clock Enable
CKEN[15]	Power Manager I^2C Unit Clock Enable	CKEN[2]	AC '97 Controller Clock Enable
CKEN[14]	I^2C Unit Clock Enable	CKEN[1:0]	0b00 = All PWMs disabled† 0b01 = All PWMs enabled 0b10 = All PWMs enabled 0b11 = All PWMs enabled
CKEN[13]	Infrared Port Clock Enable		
CKEN[12]	MMC Controller Clock Enable		

NOTE:

† Individual PWMs can be independently disabled for lower power consumption within the PWM controller using PWMDCRx[FD] (see Section 23.5.2, "PWM Duty Cycle Registers (PWMDCRx)" on page 23-8.

3.8.2.3 Oscillator Configuration Register (OSCC)

OSCC, defined in Table 3-34, controls the 13-MHz processor and 32.768-kHz timekeeping oscillator configurations as follows:

- Processor (13-MHz) Oscillator Stabilization Delay (OSD) defines the programmable delay to let the 13-MHz processor oscillator to stabilize. Refer to the *Intel® PXA27x Processor Family EMTS* for more details on recommended settings. The delay times associated with the OSD are approximate.
- Clock Request Input (CRI) status—External processor oscillator:
  - **If CRI is clear**, the clock is supplied on the PXTAL_IN and PXTAL_OUT. Observe the following requirements:
    - CLK_PIO is configurable as a GPIO or buffered output of PXTAL_IN.
    - PXTAL_IN is used as the processor-oscillator input.
    - PXTAL_IN must be connected to a crystal or clock source.
    - PXTAL_OUT must be connected to a crystal or clock source or floated.
    - CLK_REQ is an input.
    - OON and TOUT_EN is cleared out of power-on or hardware reset.

— **If CRI is set**, the clock is supplied externally. Observe the following requirements:
  - CLK_PIO is used as the processor-oscillator input.
  - PXTAL_IN must be grounded.
  - PXTAL_OUT must be floated.
  - CLK_REQ is an output.
  - A crystal or clock source must be connected to TXTAL_IN.
  - OON and TOUT_EN is set out of power-on or hardware reset.

- Processor-Oscillator Output Enable (PIO_EN) and Timekeeping-Oscillator Output Enable (TOUT_EN)—When set, enables the respective oscillator-derived clocks as outputs on the corresponding pin. When clear, the GPIO function is active.
- 32.768-kHz Timekeeping Oscillator On (OON) enables the timekeeping oscillator. OON can be set only by software and cleared only by power-on or hardware reset.
- 32.768-kHz Timekeeping Oscillator OK (OOK) sets 2-3 seconds after OON is set; switches the clock source to the RTC and power manager from the 13-MHz processor oscillator divided by 400 to the timekeeping oscillator. OOK can be set only by the stabilization timer and cleared only by power-on or hardware reset.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-34. OSCC Bit Definitions (Sheet 1 of 2)

Physical Address 0x4130_0008 — **OSCC** — **Clocks and Power Manager**

	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
User Settings																																
Bit	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																									OSD		CRI	PIO_EN	TOUT_EN	OON	OOK
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?[1]	0[2]	?[1]	?[1]	?[1]

Bits	Access	Name	Description
31:7	—	—	reserved
6:5	R/W	OSD	Processor (13-MHz) Oscillator Stabilization Delay 00 = Approximately 5 ms 01 = Approximately 3 ms 10 = Approximately 375 µs 11 = Approximately 1 µs
4	R	CRI	Clock Request Input (External Processor Oscillator) Status 0 = The processor oscillator must be supplied using PXTAL_IN and PXTAL_OUT (the CLK_REQ pin was driven low during hardware or watchdog reset). 1 = The processor oscillator is supplied externally (the CLK_REQ pin was floated during hardware or watchdog reset). **NOTE:** See the requirements for each of these conditions (listed in Section 3.8.2.3).

Table 3-34. OSCC Bit Definitions (Sheet 2 of 2)

Physical Address 0x4130_0008 — **OSCC** — **Clocks and Power Manager**

	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
User Settings																																
Bit	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																									OSD		CRI	PIO_EN	TOUT_EN	OON	OOK
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?[1]	0[2]	?[1]	?[1]	?[1]

Bits	Access	Name	Description
3	R/W	PIO_EN	13-MHz Processor Oscillator Output Enable 0 = CLK_PIO is not used; pin can be used as GPIO. 1 = CLK_PIO is driven at the same frequency as the processor oscillator, regardless of the GPIO configuration. This configuration is ignored if CRI is set.
2	R/W	TOUT_EN	Timekeeping (32.768 kHz) Oscillator Output Enable 0 = CLK_TOUT is not used; pin can be used as GPIO. 1 = CLK_TOUT is driven at the same frequency as the timekeeping oscillator, regardless of the GPIO configuration. If CRI is set, TOUT_EN is set out of power-on or hardware reset but can be set or cleared at any time.
1	R/W[3]	OON	Timekeeping (32.768 kHz) Oscillator On (Write once only) 0 = The timekeeping oscillator is disabled. 1 = The timekeeping oscillator is enabled. Once written, OON cannot be cleared except by power-on or hardware reset. If CRI is set, OON is set out of power-on or hardware reset and cannot be cleared.
0	R[4]	OOK	Timekeeping (32.768 kHz) Oscillator OK 0 = The timekeeping oscillator is disabled or not stable, and clocks (divided by 112) the RTC and power manager. 1 = The timekeeping oscillator has been enabled (OON = 1) and stabilized; it clocks the RTC and power manager. If CRI is set, OOK is set out of power-on or hardware reset.

NOTES:
1. This bit is set or cleared at power-on or hardware reset, based on the state of the CLK_REQ pin during these resets. It is unaffected by watchdog, GPIO, or sleep and deep-sleep resets.
2. This bit is unaffected by watchdog, GPIO, or sleep and deep-sleep resets.
3. This bit can be set only by software and cleared only by power-on or hardware reset.
4. This bit can be set only by the stabilization timer and cleared only by power-on or hardware reset.

3.8.2.4 Core Clock Status Register (CCSR)

CCSR, defined in Table 3-35, provides the current status of the core clock system. This read-only register is loaded whenever the following occur: frequency change, exit from standby, or exit from sleep or deep-sleep. The bits map functionally to the corresponding bits in the Core Clock Configuration register (CCCR), with the addition of two bits, CPLCK and PPLCK:

- Core PLL Lock (CPLCK) indicates whether the core PLL is ready to use
- Peripheral PLL Lock (PPLCK) indicates whether the peripheral PLL is ready to use

This is a read-only register. Ignore reads from reserved bits.

Table 3-35. CCSR Bit Definitions

Physical Address 0x4130_000C — **CCSR** — **Clocks and Power Manager**

	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
User Settings																																
Bit	CPDIS_S	PPDIS_S	CPLCK	PPLCK	reserved																		2N_S			reserved		L_S				
Reset	0†	0†	1†	1†	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0†	1†	0†	?	?	0†	0†	1†	1†	1†

Bits	Access	Name	Description
31	R	CPDIS_S	Core PLL Output Disable (CPDIS) Status Updated from CCCR at frequency change and standby, sleep, or deep-sleep exit.
30	R	PPDIS_S	Peripheral PLL Output Disable (PPDIS) Status Updated from CCCR at frequency change and standby, sleep, or deep-sleep exit.
29	R	CPLCK	Core PLL Lock 0 = Core PLL is not locked. Using it as a clock source incurs a time penalty to make the switch while it locks. 1 = Core PLL is locked and ready to use.
28	R	PPLCK	Peripheral PLL Lock 0 = Peripheral PLL is not locked. Using it as a clock source incurs a time penalty to make the switch while it locks. 1 = Peripheral PLL is locked and ready to use.
27:10	—	—	reserved
9:7	R	2N_S	Turbo-Mode-to-Run-Mode Ratio (N) Status Updated from CCCR at frequency change and standby, sleep, or deep-sleep exit. (Reset value = 0b010 for N = 1) **NOTE:** The value in this field reflects *twice* the value of N.
6:5	—	—	reserved
4:0	R	L_S	Run-Mode to 13-MHz Processor Oscillator Ratio (L) Status Updated from CCCR at frequency change and standby, sleep, or deep-sleep exit. (Reset value = 0b0_0111 for L = 7)

NOTES:
† GPIO, sleep, and deep-sleep resets do not set this bit.

3.8.3 Coprocessor 14: Clock and Power Management

Coprocessor 14 contains two registers that control the power modes and power sequences in the PXA27x processor:

- CP14 register 6—CLKCFG register
- CP14 register 7—PWRMODE register

Note: CP14 registers can be accessed only in supervisor mode. Power-on, hardware, watchdog, GPIO, sleep, and deep-sleep resets clear these registers.

3.8.3.1 Clock Configuration Register (CLKCFG)

The CLKCFG register, defined in Table 3-36, controls the following modes and sequences:

- Turbo mode (See Section 3.5.7.4)
- Frequency-change sequence (See Section 3.5.7.3)
- Fast-bus mode (See Section 3.5.7.6)
- 13M mode (See Section 3.5.7.7)
- Half-Turbo mode (See Section 3.5.7.5)

Note: Any two writes to CLKCFG or PWRMODE registers must be separated by six 13-MHz cycles. This requirement is achieved by performing the write to the CLKCFG or POWERMODE register, performing a read of CCCR, and then comparing the value in the CLKCFG or POWERMODE register to the written value until it matches.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-36. CLKCFG Bit Definitions (Sheet 1 of 2)

Coprocessor 14 Register CR6																CLKCFG												Clocks and Power Manager															
User Settings																																										
Bit	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																												B	HT	F	T
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0

Bits	Access	Name	Description
31:4	—	—	reserved
3	R/W	B	Fast-Bus Mode 0 = System bus frequency is half of the run-mode frequency. 1 = System bus frequency is equal to the run-mode frequency.

Table 3-36. CLKCFG Bit Definitions (Sheet 2 of 2)

Coprocessor 14 Register CR6 — CLKCFG — Clocks and Power Manager

	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
User Settings																																
Bit	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																												B	HT	F	T
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0

Bits	Access	Name	Description
2	R/W	HT	Half-Turbo Mode 0 = The core frequency is equal to run or turbo clock, depending on the value of T bit. 1 = The core frequency is turbo clock divided by 2. Half-turbo mode can only be invoked when the CCSR reflects the 2*N values of 6 or 8.
1	R/W	F	Frequency Change A change sequence is initiated when F is set. 0 = No frequency change is performed. 1 = Frequency-change sequence begins when written.
0	R/W	T	Turbo Mode 0 = Run mode 1 = Turbo mode

3.8.3.2 Power Mode Register (PWRMODE)

Writing to PWRMODE, defined in Table 3-37, initiates entry into the following power modes:

- Idle mode (see Section 3.6.6)
- Deep-idle mode (seeSection 3.6.7)
- Standby mode (see Section 3.6.8)
- Sleep mode (see Section 3.6.9)
- Deep-sleep mode (see Section 3.6.10)
- Voltage-change mode (see Section 3.7)

The reset mode is normal power mode, in which the CPU is fully active.

Note: Any two writes to CLKCFG or PWRMODE must be separated by six 13-MHz cycles. This requirement is achieved by performing the write to the CLKCFG or POWERMODE register, performing a read of CCCR, and then comparing the value in the CLKCFG or POWERMODE register to the preferred value until it matches.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 3-37. PWRMODE Bit Definitions

Coprocessor 14 Register CR7 — PWRMODE — Clocks and Power Manager

User Settings

Bit	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																												VC	M		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0

Bits	Access	Name	Description
31:4	—	—	reserved
3	R/W	VC	Voltage Change 0 = No voltage change is performed. 1 = Voltage-change sequence begins when written. This bit is cleared upon completion of the voltage-change mode. It can be written in conjunction with any of the modes specified by M.
2:0	R/W	M	Mode 0b000 = Normal mode 0b001 = Idle mode 0b001 = Deep-idle mode† 0b010 = Standby mode 0b011 = Sleep mode 0b100 = reserved 0b110 = reserved 0b111 = Deep-sleep mode These bits are cleared (return to normal mode) upon exit from the specified mode.

NOTE:
† CCCR[CPDIS] must be set

3.9 Register Summary

Table 3-38 summarizes the registers and memory mapping associated with the power manager. See Chapter 9, "I²C Bus Interface Unit" for the power manager I²C register descriptions.

Table 3-39 summarizes the registers and memory mapping associated with the clocks manager.

Table 3-40 summarizes the coprocessor 14 clocks and power registers, which control the power modes and power sequences for the PXA27x processor.

Table 3-38. Power Manager Register Summary (Sheet 1 of 2)

Address	Name	Description	Page
0x40F0_0000	PMCR	Power Manager Control register	3-68
0x40F0_0004	PSSR	Power Manager Sleep Status register	3-70
0x40F0_0008	PSPR	Power Manager Scratch Pad register	3-73
0x40F0_000C	PWER	Power Manager Wake-Up Enable register	3-74
0x40F0_0010	PRER	Power Manager Rising-Edge Detect Enable register	3-77

Table 3-38. Power Manager Register Summary (Sheet 2 of 2)

Address	Name	Description	Page
0x40F0_0014	PFER	Power Manager Falling-Edge Detect Enable register	3-78
0x40F0_0018	PEDR	Power Manager Edge-Detect Status register	3-79
0x40F0_001C	PCFR	Power Manager General Configuration register	3-80
0x40F0_0020	PGSR0	Power Manager GPIO Sleep State register for GPIO<31:0>	3-83
0x40F0_0024	PGSR1	Power Manager GPIO Sleep State register for GPIO<63:32>	3-83
0x40F0_0028	PGSR2	Power Manager GPIO Sleep State register for GPIO<95:64>	3-83
0x40F0_002C	PGSR3	Power Manager GPIO Sleep State register for GPIO<120:96>	3-83
0x40F0_0030	RCSR	Reset Controller Status register	3-84
0x40F0_0034	PSLR	Power Manager Sleep Configuration register	3-85
0x40F0_0038	PSTR	Power Manager Standby Configuration register	3-88
0x40F0_0040	PVCR	Power Manager Voltage Change Control register	3-89
0x40F0_0044–0x40F0_0048	—	reserved	—
0x40F0_004C	PUCR	Power Manager USIM Card Control/Status register	3-90
0x40F0_0050	PKWR	Power Manager Keyboard Wake-Up Enable register	3-92
0x40F0_0054	PKSR	Power Manager Keyboard Level-Detect Status register	3-93
0x40F0_0058–0x40F0_007C	—	reserved	—
0x40F0_0080–0x40F0_00FC	PCMD0–PCMD31	Power Manager I^2C Command register File	3-94

Table 3-39. Clocks Manager Register Summary

Physical Address	Name	Description	Page
0x4130_0000	CCCR	Core Clock Configuration register	3-95
0x4130_0004	CKEN	Clock Enable register	3-98
0x4130_0008	OSCC	Oscillator Configuration register	3-99
0x4130_000C	CCSR	Core Clock Status register	3-101
0x4130_0010–0x413F_FFFC	—	reserved	—

Table 3-40. Coprocessor 14 Clocks and Power Register Summary

Address	Name	Description	Page
CP14 register CR6	CLKCFG	Clock Configuration register	3-103
CP14 register CR7	PWRMODE	Power Mode register	3-104

Internal Memory 4

This chapter describes the internal memory of the PXA27x processor.

4.1 Overview

The PXA27x processor provides 256 Kbytes of internal memory-mapped SRAM. The SRAM is divided into four banks, each consisting of 64 Kbytes.

4.2 Features

- 256 Kbytes of on-chip SRAM arranged as four banks of 64 Kbytes
- Bank-by-bank power management for reduced power consumption
- Byte write support

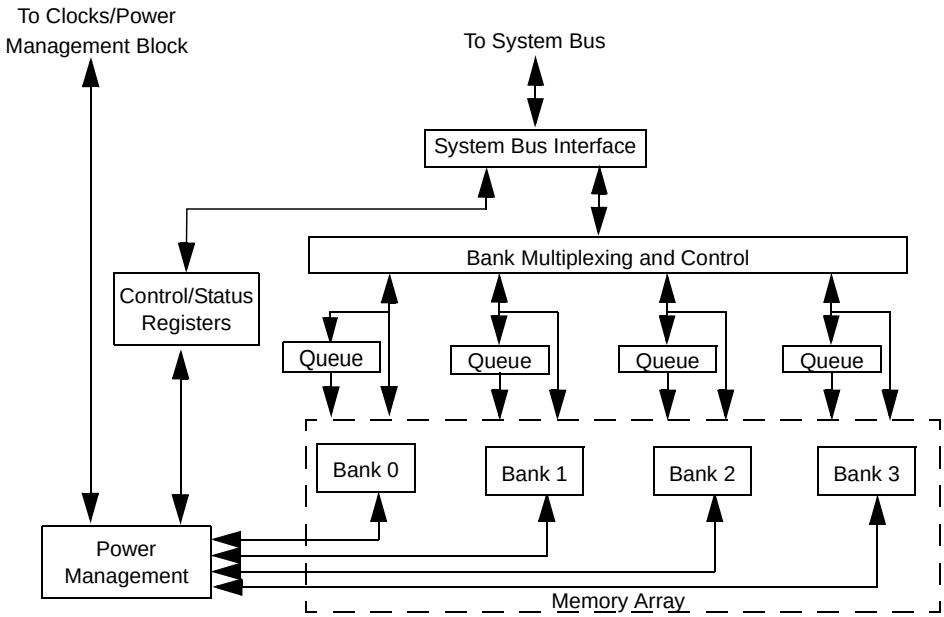
4.3 Signal Descriptions

No I/O signals are associated with the internal SRAM block.

4.4 Operation

The internal memory module has six major blocks: the system-bus interface, control and status registers, power management block, memory-bank multiplexing and control, queues, and the four SRAM banks. Figure 4-1 shows the internal memory block diagram.

Figure 4-1. Internal Memory Block Diagram



4.4.1 SRAM Array and Queue

The SRAM array module consists of four banks of 8-K x 64-bit memory arrays. Each memory bank has a dedicated single-entry queue and 8 K x 64 bits for data storage. If a memory bank is in standby mode, the access request is stored in the queue while the memory bank is placed in run mode. The access is completed when the memory bank has entered run mode. If a memory bank is in run mode and the queue does not contain any pending access requests, the queue is bypassed and the memory is accessed normally. Refer to Section 4.4.3 for more details on memory-bank operation during various power modes.

4.4.2 System Bus Interface

All accesses to the internal memory locations are initiated and completed using the system bus.

4.4.3 Power Management

The internal-memory power-management block shares control of the memory-bank power modes with the PXA27x processor power manager. The processor has six power modes of operation: normal (run and turbo), idle, deep idle, standby, sleep, and deep sleep**.** The status for the internal memory in each of these modes is as follows:

- In both run and turbo modes, the internal memory is enabled, and the memory-bank power modes are controlled by the processor power manager.
- In idle and deep-idle modes, the internal memory is enabled, and the memory-bank power modes are controlled by the processor power manager.

- In standby mode, the internal memory is placed either in standby mode or powered off, as specified by the PSTR register (see Section 3.8.1.12, “Power Manager Standby Configuration Register (PSTR)” on page 3-88).
- In sleep mode, the internal memory banks go into standby mode or are powered off, as specified by the PSLR register (see Section 3.8.1.11, “Power Manager Sleep Configuration Register (PSLR)” on page 3-85).
- In deep-sleep mode, the internal memory is always powered off. The internal memory cannot be accessed in deep-sleep mode.

Table 4-1 lists the power modes for the power manager, the power modes of the internal memory and the memory banks, and which power manager has control of the internal memory block and memory-bank power modes.

Table 4-1. Power Modes, Internal Memory, and Memory Banks

Intel® PXA27x Processor			Power-Mode Control Block for Memory Banks	Control Register
Power Modes	Internal Memory Block	Memory Banks		
Run and Turbo	Run mode	Run mode	Processor power manager	—
Idle and Deep-Idle	Run mode	Run mode	Processor power manager	—
Standby	Standby mode	Retain state or Off	Processor power manager	PSTR
Sleep	Off	Retain state or Off	Processor power manager	PSLR
Deep-Sleep	Off	Off	Processor power manager	—

All the memory banks are reset to run mode and enter run mode when the processor power manager places the internal memory block into run mode. If the processor enters sleep mode, each memory bank is powered off, and all data in the memory banks is lost unless the sleep-mode retention bits PSLR[SL_Rx] are set. If the processor enters deep-sleep mode, the memory bank is powered off and all data is lost.

4.5 Register Descriptions

Internal memory has no associated registers.

4.6 Register Summary

Table 4-2 lists the internal memory banks and their memory-mapped locations.

Table 4-2. Internal Memory Register Summary

Address	Name	Description	Page
0x5800_0000– 0x5BFF_FFFC	—	reserved	—
0x5C00_0000– 0x5C00_FFFC	Memory Bank 0	64-Kbyte SRAM	—
0x5C01_0000– 0x5C01_FFFC	Memory Bank 1	64-Kbyte SRAM	—
0x5C02_0000– 0x5C02_FFFC	Memory Bank 2	64-Kbyte SRAM	—
0x5C03_0000– 0x5C03_FFFC	Memory Bank 3	64-Kbyte SRAM	—
0x5C04_0000– 0x5C7F_FFFC	—	reserved	—
0x5C80_0000– 0x5FFF_FFFC	—	reserved	—

DMA Controller 5

This chapter describes the DMA controller for the PXA27x processor.

5.1 Overview

The PXA27x processor contains a direct-memory access (DMA) controller that transfers data to and from memory in response to requests generated by peripheral devices or companion chips. The peripheral devices and companion chips do not directly supply addresses and commands to the memory controller. Instead, the states required to manage a data stream are maintained in 32 DMA channels, DMA[31:0], in the DMA controller.

The DMA controller supports flow-through and fly-by transfers as shown in Table 5-1.

Figure 5-1 provides an overview of the DMA controller. Table 5-2 provides a signal listing.

5.2 Features

The DMA controller provides the following features:

- Memory-to-memory data transfers in flow-through mode only.
- Data transfers for peripheral-bus peripherals (PBP). Supported types are PBP-to-memory and memory-to-PBP transfers, both in flow-through mode only.
- Data transfers for internal-bus peripherals (IBPs) such as the quick capture interface. The only supported transfer types are IBP-to-memory and flow-through mode.
- Three external companion-chip-related transfers: two in fly-by and flow-through modes and one in only flow-through mode. The external device might also be an external peripheral, instead of a companion chip. A *companion chip* is defined as a device that has the ability to control the external bus, whereas an external peripheral must be controlled by the memory controller.
- 32 channels, 68 PBP requests, 3 IBP requests, and 3 external device requests. Allows any request to a channel to be pre-programmed.
- A priority mechanism to process active channels (four channels with outstanding DMA requests at any given time).
- Each of the 32 channels can operate for descriptor-fetch or no-descriptor-fetch transfers (see Section 5.4.2).
- Special descriptor modes (descriptor comparison and descriptor branching).
- Retrieval of trailing bytes in the receive peripheral-device buffers.
- Programmable data-burst sizes (8, 16, or 32 bytes) and programmable peripheral device data widths (byte, half word, or word)
- Up to (8 Kbytes –1) bytes of data transfer per descriptor. Larger transfers can be performed by chaining multiple descriptors.

- Flow-control bits to process requests from peripheral devices. Requests are not processed unless the flow-control bit is set.

Table 5-1. DMA Support Matrix

	Internal Memory	External Memory	Internal Peripheral	External Peripheral	Companion Chip
Companion Chip	Flow through	Flow through or fly by	Flow through	Flow through or fly by	Flow through or fly by
External Peripheral	Flow through	Flow through	Flow through	Flow through	
Internal Peripheral	Flow through	Flow through	—		
External Memory	Flow through	Flow through			
Internal Memory	Flow through				

NOTE: A *companion chip* is defined as a device that has the ability to control the external bus, whereas an *external peripheral* must be controlled by the memory controller.

5.3 Signal Descriptions

The DREQ<2:0>, PREQ<67:0>, IREQ<2:0>, and DMA_IRQ signals are controlled by the DMA controller. The DVAL<1:0> signals are driven by the memory controller, as indicated in Figure 5-1.

Table 5-2. External DMA Controller I/O Signal Descriptions

Signal Name	Direction	Description
DREQ<2:0>	Input	External Device Request Lines The DMA controller detects the positive edge of this signal to log a request. The external device asserts the DREQ<2:0> signals when a DMA transfer request is required. The DREQ<2:0> signals must remain asserted for four CLK_MEM cycles to allow the DMA controller to recognize the low-to-high transition (see Figure 5-2). When the DREQ<2:0> signals are de-asserted, they must remain so for at least four CLK_MEM cycles. The DMA controller registers the transition from low to high to identify a new request. The external device need not wait until the completion of the data transfer before asserting the next request. It can have up to 31 outstanding requests on each of the DREQ<2:0> pins. The number of pending requests are logged in status registers, DRQSRx. Requests on pins DREQ<1:0> can be used for data transfers in both fly-by and flow-through modes. Requests on pins DREQ<2> can be used for data transfers in flow-through mode only.
DVAL<1:0>	Output	External Data Valid signals for fly-by transfers. The memory controller asserts DVAL to notify the companion chip that data must be driven or is valid.

5.4 Operation

The DMA controller can be configured to transfer data using flow-through or fly-by DMA.

All addresses used by the DMA controller must be physical memory addresses and not virtual memory addresses.

Software must ensure cache coherency when it configures the DMA channels. The DMA controller does not check the cache, so target and source addresses must be configured as non-cacheable in the memory management unit.

The DMA controller has 32 configurable channels. Figure 5-1 shows how the controller uses the signals.

Figure 5-1. DMA Controller Block Diagram

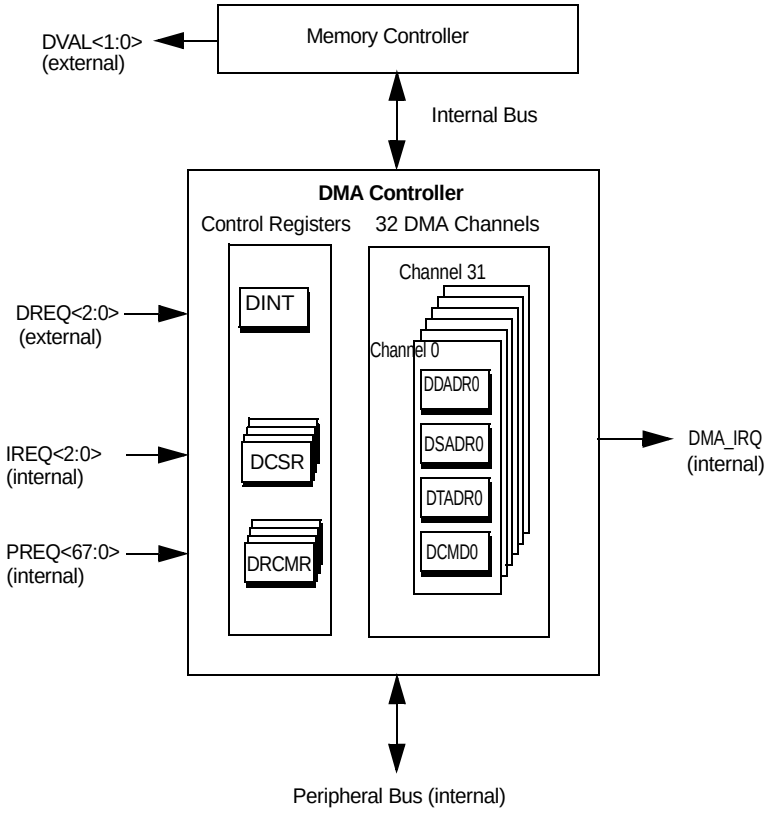
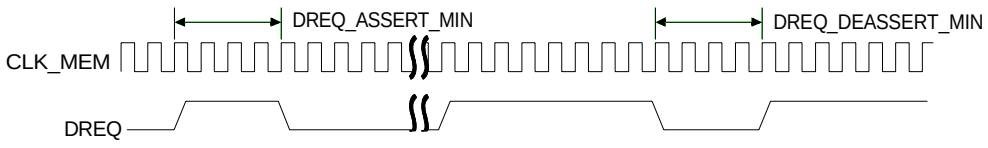


Figure 5-2. DREQ Timing Requirements



5.4.1 DMA Channels

The 32-channel DMA is controlled by four 32-bit registers. Each channel can be configured to service any kind of transfer. Each channel is serviced in increments of that device's burst size and delivered in the granularity of the device port width. The burst size and port width for each device are programmed in the channel registers and are based on the device FIFO depth and bandwidth requirements. When multiple channels are actively executing, each channel is serviced with a burst of data. After each burst of data, the DMA controller performs a context switch to another active channel. The DMA controller performs context switches based on whether a channel is active, whether its target device is currently requesting service, and the channel's priority.

5.4.1.1 DMA Channel-Priority Scheme

The DMA channel-priority scheme helps ensure that peripherals are serviced according to their bandwidth requirements. Assign a high priority to peripherals with high-bandwidth requirements and a lower priority to peripherals with lower bandwidth requirements. This practice ensures that high-bandwidth peripherals are serviced more often than low-bandwidth peripherals.

The DMA channels are divided internally into four sets of eight channels each. The channels in each set get a round-robin priority. Set 0 has highest priority and set 3 has lowest. Program modules with the most severe latency requirements are in set 0. Sets 2 and 3 are low-priority sets. Program memory-to-memory moves and low-bandwidth peripherals to use set 2 and 3. Refer to Table 5-3 for details.

When all the channels are running concurrently, set 0 is serviced four out of every eight consecutive channel-servicing instances, set 1 is serviced twice, and sets 2 and 3 are each serviced one time.

For example, if all the channels request data transfers, the sets are prioritized in following order: set 0, set 1, set 0, set 2, set 0, set 1, set 0, set 3. After eight channel-servicing instances, the pattern repeats. The channels in each set are given a round-robin priority.

Table 5-3. Channel Priority

Set	Channels	Priority	Number of Times Served
0	0, 1, 2, 3, 16, 17, 18, 19	Highest	4 / 8
1	4, 5, 6, 7, 20, 21, 22, 23	Higher than 2 and 3. Lower than 0,	2 / 8
2	8, 9, 10, 11, 24, 25, 26, 27	Higher than 3. Lower than 0 and 1.	1 / 8
3	12, 13, 14, 15, 28, 29, 30, 31	Lowest	1 / 8

5.4.1.2 Concurrent Active Channels

The DMA controller can have up to four outstanding (active) DMA requests at a time. The DMA controller has four 32-byte internal buffers to hold descriptor information or data fetched from memory.

The DMA controller can have one of two transfer combinations at a time:

- Up to four outstanding transfers on the internal bus
- Three outstanding transfers on the internal bus and one outstanding transfer on the peripheral bus

After the peripheral bus is allocated to an active channel, no other channel can use the peripheral bus until the current active channel completes the transfer on the peripheral bus. The internal bus can be allocated successively to four active channels.

5.4.1.3 Channel States

The following states apply to the DMA channels:

- Uninitialized—Occurs after a reset. DCSRx[STOPINTR] is set when uninitialized.
- Valid descriptor, not running—Occurs when either a valid descriptor has been loaded in the DDADRx register during a descriptor-fetch transfer or valid DSADRx, DTADRx, and DCMDx registers have been programmed during a no-descriptor-fetch transfer, but the corresponding run bit, DCSRx[RUN], is not set. For a no-descriptor-fetch transfer, DCSRx[STOPINTR] is not cleared when the DSADRx, DTADRx and DCMDx registers are programmed; for a descriptor-fetch transfer, DCSRx[STOPINTR] is cleared when the DMA controller updates the DDADRx register.
- Descriptor fetch, running—For a descriptor-fetch transfer, after programming DDADRx and setting DCSRx[RUN], four words of descriptors are fetched from the memory and DCSRx[STOPINTR] continues to be clear. For a no-descriptor-fetch transfer, after programming the DSADRx, DTADRx and DCMDx registers and setting DCSRx[RUN], the corresponding channel clears the DCSRx[STOPINTR], skips the “descriptor-fetch, running” state and enters the “Wait for Request” or “Transfer Data” state (see Figure 5-3).
- Wait for request—Occurs as the channel waits for a request before it starts to transfer data; DCSRx[STOPINTR] is clear.
- Transfer data—Transferring data to/from source/target; DCSRx[STOPINTR] is clear.
- Channel error—Error in the channel. The channel remains in the stopped state until software clears the error condition, re-initializes the channel, and sets the DCSRx[RUN] bit and the DCSRx[BUSERRINTR] bit. See Section 5.5.9 and Section 5.5.8 for details.
- Stopped—The channel is stopped. DCSRx[STOPINTR] is set. For a no-descriptor-fetch transfer, a stopped channel is re-initialized by updating the DSADRx, DTADRx and DCMDx registers, then setting DCSRx[RUN]. For a descriptor-fetch transfer, a stopped channel is re-initialized by updating the DDADRx register and setting DCSRx[RUN].

Table 5-4. Channel States Based on Software Configuration

Descriptor Mode	Software Configuration	DCSRx[Run]	DCSRx[StopIntr]	Resulting Channel State
Descriptor-Fetch Mode.	Power-up	0	1	Uninitialized
	Write to DDADRx before DCSRx[Run] is set (recommended flow).	0	0	Valid descriptor, not running.
	Set DCSRx[Run] after writing to DDADRx (recommended flow).	1	0	Descriptor fetch, running.
	Set DCSRx[Run] before writing to DDADRx (*not recommended*).	1	1	Invalid. *This configuration flow is not recommended.*
	Write to DDADRx after DCSRx[Run] is set (*not recommended*).	1	0	Descriptor fetch, running. *This configuration flow is not recommended.*
	Stop running channel by clearing DCSRx[Run] and DCSRx[MaskRun]	0	0 -> 1	Channel, if not immediately, eventually switches to a stopped state (identified by DCSRx[StopIntr] toggling from low to high').
No-Descriptor-Fetch Mode.	Power-on	0	1	Uninitialized
	Write to DSADRx, DTADRx and DCMDx before DCSRx[Run] is set (recommended flow).	0	1	Valid descriptor, not running.
	Set DCSRx[Run] after configuring DSADRx, DTADRx, and DCMDx (recommended flow).	1	0	Wait for Request, running.
	Set DCSRx[Run] before configuring DSADRx, DTADRx, and DCMDx (*not recommended*).	1	0	Wait for Request, running. *Channel uses current DSADRx, DTADRx and DCMDx for the transfer and may lead to unpredictable results.*
	Stop running channel, by clearing DCSRx[Run] and DCSRx[MaskRun].	0	0 -> 1	Channel, if not immediately, eventually switches to a stopped state (identified by DCSRx[StopIntr] toggling from low to high).

5.4.2 DMA Descriptors

A DMA descriptor is a four-word (32-bit) block, aligned on a 16-byte boundary in memory:

Word [0] contains a value for the DDADRx register and a single flag bit (STOP).

Word [1] contains a value for the DSADRx register.

Word [2] contains a value for the DTADRx register.

Word [3] contains a value for the DCMDx register.

The DMA controller can operate in two distinct modes based on the DCSRx[NODESCFETCH] bit:

- Descriptor-fetch transfer
- No-descriptor-fetch transfer

5.4.2.1 Descriptor-Fetch Transfer Operation

Descriptor-fetch transfers (DCSRx[NODESCFETCH] = 0) operate in the following manner:

Software must first clear the DCSRx[RUN] bit and then clear the DCSRx[NODESCFETCH] bit. Software must write a valid descriptor address to the DDADRx register and then set DCSRx[RUN]. Doing so in this order enables the DMA controller to fetch the four-word descriptor (if the memory is already set up with the descriptor chain) from the memory that the DDADRx register indicates. The channel either waits for a request or starts the data transfer, as determined by the DCMDx[FLOW] source and target bits. After the channel transfers a number of bytes equal to the smaller of DCMDx[SIZE] and DCMDx[LEN], it either waits for the next request or continues with the data transfer until the DCMDx[LEN] reaches zero. The channel stops or continues with a new descriptor fetch from the memory, as determined by the DDADRx[STOP] bit. Figure 5-3 summarizes this operation.

If an error occurs during the fetch operation, the channel enters the stopped state and remains there unless the software clears the error condition, re-initializes the channel, and sets the DCSRx[RUN] register.

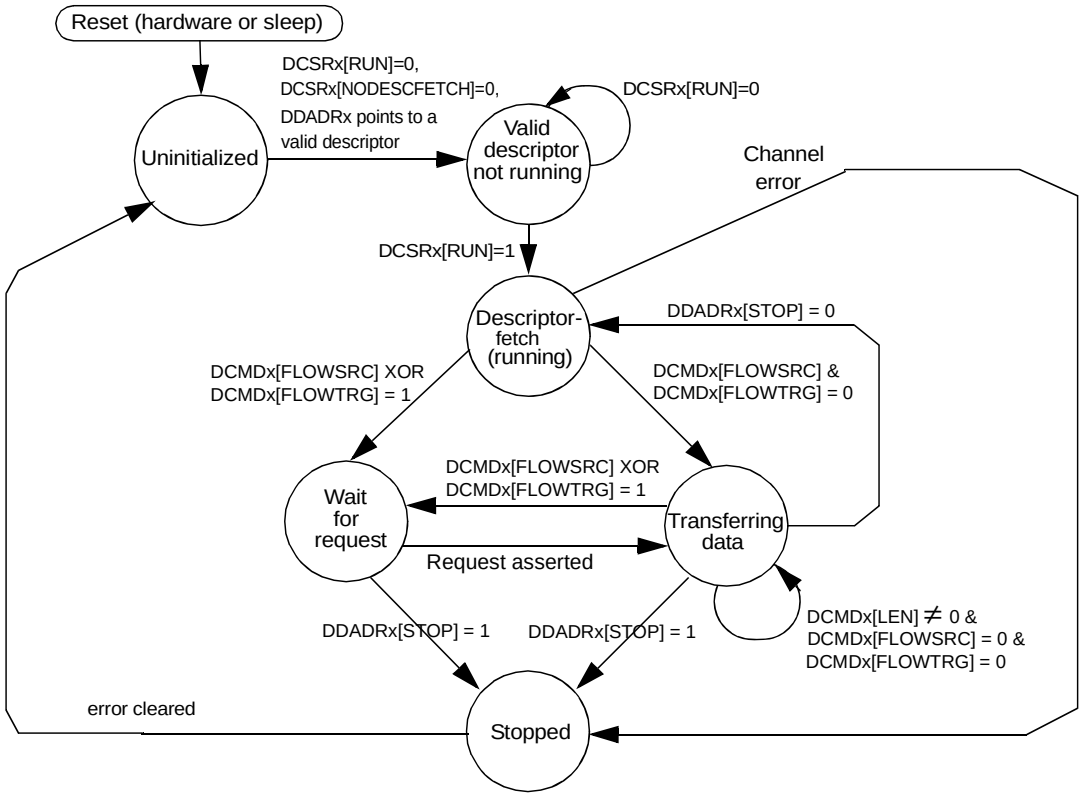
When a channel switches between a descriptor-fetch transfer and a no-descriptor-fetch transfer, it must be stopped before the mode switch.

For a descriptor-fetch transfer, the software must load the DDADRx register and set the DCSRx[RUN] bit. The channel-descriptor fetch does not occur unless the DCSRx[RUN] bit is set.

Although software loads the DDADRx register, the DSADRx, DTADRx, and DCMDx registers must be loaded indirectly from DMA descriptors. The DMA descriptor indicated in the DDADRx register is loaded into the registers of the associated DMA channel after all of the data described by a descriptor has been transferred and when a write to the DCSRx[RUN] register switches the channel from stopped to running.

Bit [0] (STOP) of word [0] of a DMA descriptor (the low bit of the DDADRx field) marks the special descriptor, which resides at the end of a descriptor list. The value of the stop bit does not affect the loading of the fields of a descriptor into channel registers in any way; however, if a descriptor with the stop bit set is loaded into a channel register, then the channel stops after completely transferring the data pertaining to that channel.

Figure 5-3. Descriptor-Fetch Transfer Channel State Diagram

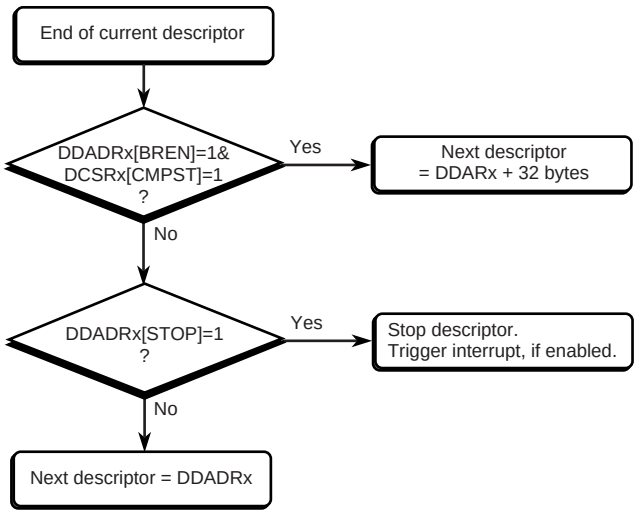


5.4.2.1.1 Descriptor Branching

If DDADRx[BREN] and DCSRx[CMPST] are set, the DMA controller fetches the next descriptor from (the address in the DDADRx register + 32 bytes). If either of the bits is clear, the DMA controller fetches the next descriptor from the address in the DDADRx register.

DDADRx[BREN] is relevant only for descriptor-fetch transfers (DCSRx[NODESCFETCH] is cleared).

Figure 5-4. Flow Chart for Descriptor Branching



A9378-01

5.4.2.2 No-Descriptor-Fetch Transfer Operation

The typical no-descriptor-fetch transfer (DCSRx[NODESCFETCH] = 1) operation follows:

The channel is in an uninitialized state after reset. Software must first clear DCSRx[RUN] and then set DCSRx[NODESCFETCH]. Software must write a valid source address to the DSADRx register, a target address to the DTADRx register, and a command to the DCMDx register. The DDADRx register is reserved in this mode and must not be written. Next, the DCSRx[RUN] bit must be set. A descriptor fetch is not performed. Depending on the DCMDx[FLOW] source and target bits, the channel either waits for the request or starts the data transfer. After transferring the number of bytes equal to the smaller of DCMDx[SIZE] and DCMDx[LEN], the channel either waits for the next request or continues with the data transfer until the DCMDx[LEN] reaches zero, depending on the priority of the enabled DMA channels. When DCMDx[LEN] reaches zero, the channel stops. Figure 5-5 summarizes this operation.

Setting DCSR[STOPIRQEN] to trigger a stop interrupt might cause the DMA controller to trigger the stop interrupt prematurely, possibly even before the channel enters active run mode. See Table 5-17 for more information on the RUN condition.

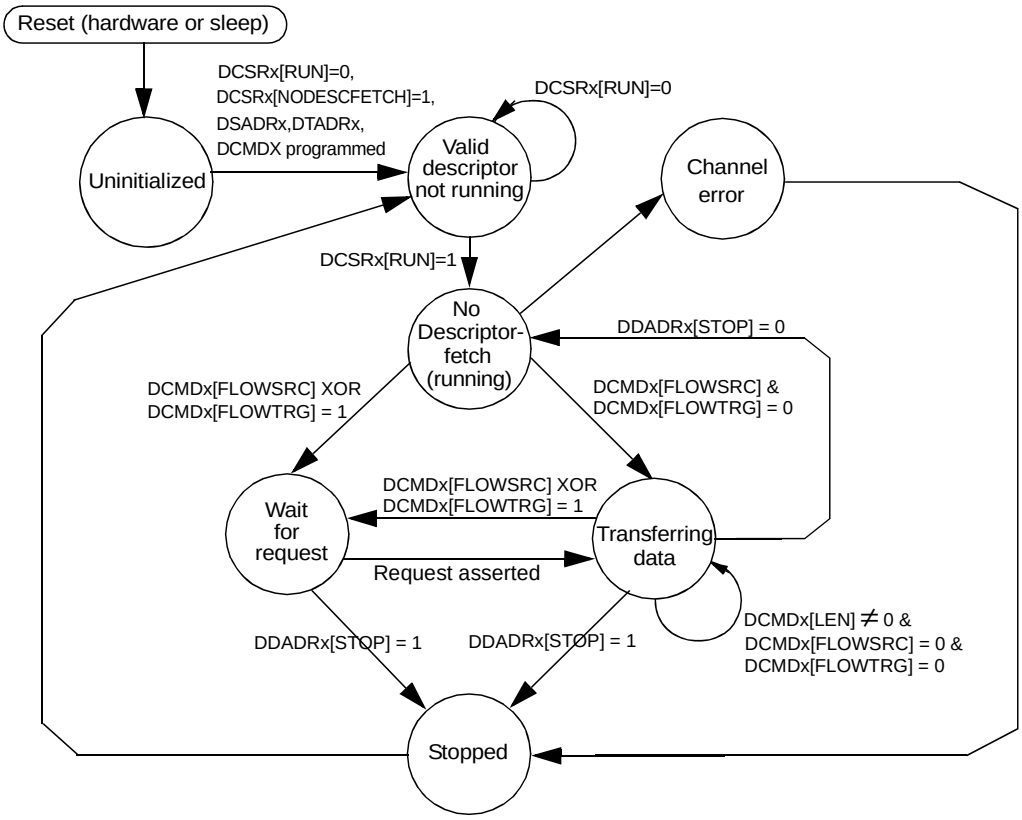
To detect stoppage of a channel as the result of an end-of-receive (EOR) from a peripheral, software must validate the stoppage by checking the status of DCSR[EORINT]. If the EOR condition stopped the channel, then DCSR[EORINT] is set. See Table 5-17 for more information on the EORINT bit.

Detect normal channel stoppage by using the end interrupt, **not** the stop interrupt. See Table 5-17 for more information on the end interrupt.

If an error occurs during the fetch operation, the channel enters the stopped state and remains there unless software clears the error condition and sets the DCSRx[RUN] bit field. When a channel switches between a descriptor-fetch transfer and a no-descriptor-fetch transfer, it must be stopped before the switch occurs.

It is possible to use a few DMA channels for a descriptor-fetch transfer and the remaining channels in a no-descriptor-fetch transfer simultaneously. Do not program the DDADRx register of the channels during a no-descriptor-fetch transfer.

Figure 5-5. No-Descriptor-Fetch Transfer Channel State Diagram



5.4.3 Transferring Data

The on-chip peripherals connected to the DMA on the peripheral bus operate in flow-through transfers (For details, refer to Section 5.4.3.1.1). Although the source or destination of a DMA transfer is usually a peripheral intended to be used as a source or sink of DMA data, the DMA controller can transfer data to or from any memory location through memory-to-memory moves.

High-performance external DMA devices, such as companion chips, are directly connected to the data pins of the memory SDRAMs and operate in fly-by mode. Such devices can achieve high data-transfer rates and are restricted to transfers with alignments and lengths that match that of the memory. These external DMA devices also work for flow-through transfers.

5.4.3.1 Servicing Internal Peripherals

The PXA27x processor has two types of internal peripherals: the peripheral bus peripherals (PBP) and the internal bus peripherals (IBP). Peripherals such as AC ’97, MSL, and UART are examples of PBP. The quick-capture interface is the only IBP in the processor. The DMA controller provides DMA request to channel map registers (DRCMRx) that contain five bits of channel number for

each of the possible DMA requests. These possible peripheral requests are mapped to 32 available channels. Peripherals on the peripheral bus assert the appropriate peripheral request signal (PREQx); peripherals on the internal bus assert the appropriate internal request signal (IREQx). IREQ signals are sampled directly on every system bus clock (SYSCLK). The PREQ signals are sampled on every peripheral clock (PCLK) and synchronized to the internal bus clock domain. The synchronization delay might cause an IREQ to be processed ahead of a PREQ, even if the latter was mapped to a channel of higher priority. If any of the PREQ signals are sampled non-zero, a look-up is performed on the corresponding bits of the DRCMRx register, which enables the request to be mapped to one of the channels. If the on-chip peripheral address resides in the DSADRx register, the DCMDx[FLOWSRC] bit must be set, which allows the processor to wait for the request before it initiates the transfer. If the on-chip peripheral address resides in the DTADRx register, then the DCMDx[FLOWTRG] bit must be set.

If DCMDx[IRQEN] is set, a DMA interrupt is requested at the end of the last cycle associated with the byte that caused DCMDx[LEN] to decrement to zero.

Note: Internal peripherals, whether IBP or PBP, cannot be serviced with fly-by transfers.

5.4.3.1.1 Servicing Internal Peripherals Using Flow-Through DMA Read Cycles

A flow-through DMA read begins when an on-chip peripheral sends a request on the PREQ bus to a channel in the DMA controller while the channel is running and is configured for flow-through reads. The number of bytes to be transferred is specified using DCMDx[SIZE]. When the request is recognized, the following process begins:

1. The DMA controller prompts the memory controller to read the required number of bytes addressed by DSADRx into a 32-byte buffer in the DMA controller.
2. The DMA controller transfers the data to the peripheral device addressed in DTADRx[31:0]. The DCMDx[WIDTH] specifies the width of the internal peripheral to which the transfer is being made.
3. At the end of the transfer, DSADRx is increased and DCMDx[LEN] is decreased by the smaller of DCMDx[LEN] and DCMDx[SIZE].

For a flow-through DMA read from an internal peripheral, use the following settings for the DMA controller register bits:

DCMDx[FLYBYS] and DCMDx[FLYBYT] = 0
DSADRx[SRCADDR] = memory address
DTADRx[TRGADDR] = internal peripheral address
DCMDx[INCSRCADDR] = 1
DCMDx[INCTRGADDR] = 0
DCMDx[FLOWSRC] = 0
DCMDx[FLOWTRG] = 1

5.4.3.1.2 Servicing Internal Peripherals Using Flow-Through DMA Write Cycles

A flow-through DMA write begins when an on-chip peripheral sends a request on the PREQ bus to a channel in the DMA controller while the channel is running and is configured for flow-through writes. The number of bytes to be transferred is specified using DCMDx[SIZE]. When the request is recognized the following process begins:

1. The DMA controller processes the request by transferring the required number of bytes from the peripheral device addressed by DSADRx[31:0] into a DMA controller buffer.

2. The DMA controller transfers the data to the memory controller on the internal bus. The DCMDx[WIDTH] specifies the width of the internal peripheral to which the transfer is being made.
3. At the end of the transfer, DTADRx is increased and DCMDx[LEN] is decreased by the smaller of DCMDx[LEN] and DCMDx[SIZE].

For a flow-through DMA write from an internal peripheral, use the following settings for the DMA controller register bits:

DCMDx[FLYBYS] and DCMDx[FLYBYT] = 0
DSADRx[SRCADDR] = internal peripheral address
DTADRx[TRGADDR] = memory address
DCMDx[INCSRCADDR] = 0
DCMDx[INCTRGADDR] = 1
DCMDx[FLOWSRC] = 1
DCMDx[FLOWTRG] = 0

5.4.3.2 Servicing External Companion Chips

External companion chips can be serviced using DREQ<1:0>, using flow-through transfers (DCMDx[FLYBYS] and DCMDx[FLYBYT] = 0) or fly-by transfers (DCMDx[FLYBYS] or DCMDx[FLYBYT] = 1). External companion chips can be serviced using DREQ<2>, using flow-through transfers (DCMDx[FLYBYS] and DCMDx[FLYBYT] = 0) only. The DMA controller provides DMA requests to the channel map registers (DRCMRx) that contain five bits of channel number for each of the possible DMA requests. The companion-chip requests are DREQ<2:0>. A request can be mapped to any of the eight channels in a priority set. The DMA controller detects a request by identifying a low-to-high transition on the DREQ (the timing information is explained in Section 5.3). The DMA controller then performs a lookup on the corresponding bits of the DRCMRx register. The lookup maps the request to one of the channels. If the external companion-chip address resides in the DSADRx register, the DCMDx[FLOWSRC] bit must be set. If the external companion-chip address resides in the DTADRx register, the DCMDx[FLOWTRG] bit must be set. This lookup allows the application processor to wait for the request before it initiates the transfer.

If DCMDx[IRQEN] is set, a DMA interrupt is requested at the end of the last cycle associated with the byte that caused DCMDx[LEN] to decrement to zero.

Refer to Section 5.3 for the DVAL<1:0> timing and other related information.

5.4.3.2.1 Servicing Companion Chips: Flow-Through DMA Read Cycles

A flow-through DMA read begins when an external companion chip sends a request, on the DREQ<2:0> bus, to a channel in the DMA controller while the channel is running and is configured for flow-through reads. The number of bytes to be transferred is specified using DCMDx[SIZE]. When the request is recognized the following process begins:

1. The DMA controller prompts the memory controller to read the required number of bytes addressed by DSADRx[31:0] into a 32-byte buffer in the DMA controller.
2. The DMA controller transfers the data to the external device addressed in DTADRx[31:0].
3. At the end of the transfer, DSADRx is increased and DCMDx[LEN] is decreased by the smaller of DCMDx[LEN] and DCMDx[SIZE].

For a flow-through DMA read from an external companion chip, use the following settings for the DMA controller register bits:

DCMDx[FLYBYS] and DCMDx[FLYBYT] = 0
DSADRx[SRCADDR] = memory address
DTADRx[TRGADDR] = companion chip address
DCMDx[INCSRCADDR] = 1
DCMDx[INCTRGADDR] = 0
DCMDx[FLOWSRC] = 0
DCMDx[FLOWTRG] = 1

5.4.3.2.2 Servicing Companion Chips: Flow-Through DMA Write Cycles

A flow-through DMA write begins when an external companion chip sends a request, on the DREQ bus, to a channel in the DMA controller while the channel is running and is configured for flow-through writes. The number of bytes to be transferred is specified using DCMDx[SIZE]. When the request becomes the highest priority request and the following process begins:

1. The DMA controller transfers the required number of bytes from the peripheral device addressed by DSADRx into a DMA controller buffer.
2. The DMA controller transfers the data to the memory controller on the internal bus.
3. At the end of the transfer, DTADRx is increased and DCMDx[LEN] is decreased by the smaller of DCMDx[LEN] and DCMDx[SIZE].

For a flow-through DMA write to an external companion chip, use the following settings for the DMA controller register bits:

DCMDx[FLYBYS] and DCMDx[FLYBYT] = 0
DSADRx[SRCADDR] = companion chip address
DTADRx[TRGADDR] = memory address
DCMDx[INCSRCADDR] = 0
DCMDx[INCTRGADDR] = 1
DCMDx[FLOWSRC] = 1
DCMDx[FLOWTRG] = 0

5.4.3.2.3 Servicing Companion Chips: Fly-By DMA Read Cycles

A fly-by DMA read begins when an external companion chip sends a request on the DREQ<1:0> bus to a channel in the DMA controller while the channel is running and is configured for fly-by reads. The number of bytes to be transferred is specified using DCMDx[SIZE]. DREQ<2> must not be used for the fly-by mode.

The following process when the request becomes the highest priority request:

1. The DMA controller prompts the memory controller to read the required number of bytes addressed by DSADRx.
2. Rather than latching the data in the DMA, the DMA controller provides an external strobe signal that allows the external device to latch the data while it is valid on the external bus. DTADRx is not used during this transfer.
3. At the end of the transfer, DSADRx is increased and DCMDx[LEN] is decreased by the smaller of DCMDx[LEN] and DCMDx[SIZE].

For a fly-by DMA read from an external companion chip, use the following settings for the DMA controller register bits:

DCMDx[FLYBYS] = 1
DCMDx[FLYBYT] = 0

DSADRx[SRCADDR] = external memory address
DCMDx[INCSRCADDR] = 1
DCMDx[FLOWSRC] = 0
DCMDx[FLOWTRG] = 1

Note: The PXA27x processor does not support fly-by transfers between the internal SRAM and an external companion chip. Fly-by transfers can be used only for data transfers between SDRAM and the companion chip.

5.4.3.2.4 Servicing Companion Chips: Fly-By DMA Write Cycles

A fly-by DMA write begins when an external companion chip sends a request on the DREQ bus to a channel in the DMA controller while the channel is running and is configured for fly-by write transfers. The number of bytes to be transferred is specified using DCMDx[SIZE]. When this request is recognized the following process begins:

1. The DMA controller signals the memory controller that it has received a request for a fly-by write operation.
2. The memory controller signals the companion chip (using the DVAL signals) to drive its data on to the external bus.
3. The required number of bytes are transferred from the companion chip into the DTADRx[TRGADDR] memory location. DSADRx is not used during this transfer.
4. At the end of the transfer, DTADRx is increased and DCMDx[LEN] is decreased by the smaller of DCMDx[LEN] and DCMDx[SIZE].

For a fly-by DMA write to an external companion chip, use the following settings for the DMA controller register bits:

DCMDx[FLYBYS] = 0
DCMDx[FLYBYT] = 1
DTADRx[TRGADDR] = external memory address
DCMDx[INCTRGADDR] = 1
DCMDx[FLOWSRC] = 1
DCMDx[FLOWTRG] = 0

Note: The PXA27x processor does not support fly-by transfers between the internal SRAM and an external companion chip. Fly-by mode can only be used for data transfers between SDRAM and the companion chip.

5.4.3.3 Memory-to-Memory Moves

Memory-to-memory moves do not involve request signals. For a memory-to-memory move, the processor writes to the DCSRx[RUN] bit indicated by the channel that is configured to perform a memory-to-memory move. The DCMDx[FLOWSRC] and DCMDx[FLOWTRG] bits must be cleared as soon as the descriptor is fetched. The transfer begins without waiting for any request signals.

If DCMDx[IRQEN] is set, a DMA interrupt is requested at the end of the last cycle associated with the byte that caused DCMDx[LEN] to decrement to zero.

Note: Memory-to-memory moves cannot be performed using fly-by mode. They can be performed from internal SRAM to external memory and vice versa.

5.4.3.3.1 Memory-to-Memory Moves: Flow-Through DMA Read/Write Cycles

A flow-through DMA memory-to-memory read or write begins when the processor writes to the DCSRx[RUN] bit. If the channel is in a descriptor-fetch transfer, it fetches the four-word descriptor. The FLOWSRC and FLOWTRG bits must be cleared for a memory-to-memory move. The channel begins to transfer data without waiting for any request signals. The number of bytes to be transferred is specified using DCMDx[SIZE]. Processing begins as follows:

1. The DMA controller prompts the memory controller to read the required number of bytes addressed by DSADRx into a 32-byte buffer in the controller.
2. The DMA controller generates a write cycle to the to the location addressed by DTADRx.
3. At the end of the transfer, DSADRx and DTADRx are increased and DCMDx[LEN] is decreased by the smaller of DCMDx[LEN] and DCMDx[SIZE].

Use the following settings for the DMA controller register bits for flow-through memory-to-memory moves:

DCMDx[FLYBYS] and DCMDx[FLYBYT] = 0
DSADRx[SRCADDR]= memory address
DTADRx[TRGADDR]= memory address
DCMDx[INCSRCADDR] = 1
DCMDx[INCTRGADDR] = 1

5.4.4 Programming Tips

This section provides information concerning software requirements, instruction ordering, and misaligned memory accesses.

5.4.4.1 Software Management Requirements

The information that must be maintained on a per-stream basis (for example, the memory address, the peripheral address, the transfer count, and the implied direction of data flow), is maintained in descriptor registers in the DMA controller. The descriptor registers are loaded from memory locations specified by the software. Multiple DMA descriptors can be chained together in a list. This allows a DMA channel to transfer data to and from a number of separate locations. The descriptor-based DMA design allows descriptors to be dynamically added to an active DMA channel-descriptor chain, which is particularly useful in applications that involve network-transmit lists and network-receiver buffer-free lists.

Each demand for data that a peripheral generates is a memory data read or write. A peripheral must not request a DMA transfer unless it is prepared to read or write the full data block (8, 16, or 32 bytes) and is able to handle trailing bytes that can occur at the end of a DMA transfer.

5.4.4.2 Programmed I/O Operations

The processor can read and write the peripheral registers and FIFOs on the peripheral bus. The peripherals' internal registers must be accessed using word-access loads and stores. Internal register and FIFO space must be mapped as non-cacheable. Byte and half-word accesses to internal registers are not allowed. The FIFOs of some of the peripherals on the peripheral bus are accessed using byte, half-word, or word-access loads and stores. Refer to the individual peripheral chapters for details.

5.4.4.3 Instruction Ordering

The DMA controller executes programmed I/O instructions in the order specified by the software.

References to internal addresses generally complete more quickly than those issued to external addresses, which means that memory accesses can be sent in one order and completed in a different order.

The DMA controller ensures that memory references made by a single DMA channel are presented to memory in order and that the descriptor fetches occur between the data blocks. The order in which the accesses are completed cannot be guaranteed unless the channels refer to only one type of memory (either to the external memory or to the internal SRAM).

The channel references must not reference both internal and external memory in a DMA descriptor chain for the following operations:

- Self-modifying DMA descriptor chains.
- Channels that write data blocks followed by status blocks while another channel (generally the processor) polls a field in the status block.

5.4.4.4 Misaligned Memory Accesses

The DMA controller is a 64-bit device that can access memory on byte-aligned boundaries. The DMA controller can encounter misaligned (not aligned to 64-bit boundary) addresses while it access memory.

Only the following type of data transfers can access misaligned addresses:

1. Memory-to-memory transfers.
2. Memory-to-peripheral transfers or peripheral-to-memory transfers. In this case, the peripheral addresses are 32-bit-aligned.

Note: All companion-chip-related transfers must use 64-bit aligned addresses for both source and target locations.

Addresses must be 64-bit aligned in the compare-descriptor mode.

The DMA controller employs channel-specific alignment buffers that hold either the leading or the lagging misaligned data. These buffers must be empty when the DMA controller performs a context switch to service the next pending channel. When the DMA controller completes a descriptor transfer, it ensures that all of the data in the alignment buffers is properly flushed to its respective targets.

Restricting memory addresses to 8-byte boundaries can be helpful because the DMA controller encounters overhead while it works with misaligned data. Align the source and target addresses to 32-byte boundaries for optimal DMA controller and memory controller performance.

By default, during data transfers the DMA controller forces the least significant three bits of all external addresses to zeros and the least significant two bits of all peripheral addresses to zeros. Software must activate the alignment register to activate byte-aligned addressing. See Table 5-19 for details.

5.4.5 Fly-By Transfers

The PXA27x processor supports fly-by transfers for external devices. Fly-by transfers are achieved through four external pins: DREQ<1:0> and DVAL<1:0>. Fly-by transfers allow the external device to drive data on the memory data bus by three-stating this bus during writes. For fly-by transfers, requests are only made to or from external SDRAM.

The memory controller interfaces with the DVAL<1:0> pins. The DMA controller interfaces with the DREQ<1:0> pins to detect fly-by DMA requests. Such requests are encoded on the transfer to the memory controller from the DMA. When the memory controller processes this transfer, the external pins DVAL<1:0> are asserted appropriately when the data is available to the external device for reads or when the external device must drive the data for writes. Figure 5-7 describes fly-by DMA mode.

Read data must be latched on the rising edge of the SDCLK that is being used when DVAL is asserted (SDCLK<1> for SDRAM partitions 0/1 or SDCLK<2> for SDRAM partitions 2/3).

For writes, DVAL is driven out two SDCLKs early, so a delayed version must be used to generate the write data to SDRAM. Write data must be driven on the rising edge of the SDCLK that is being used when a two-clock delayed DVAL is asserted (SDCLK<1> for SDRAM partitions 0/1 or SDCLK<2> for SDRAM partitions 2/3) and must be released on the rising edge of the same SDCLK when the two-clock delayed DVAL is de-asserted.

Figure 5-6. Fly-By Transfer Diagram

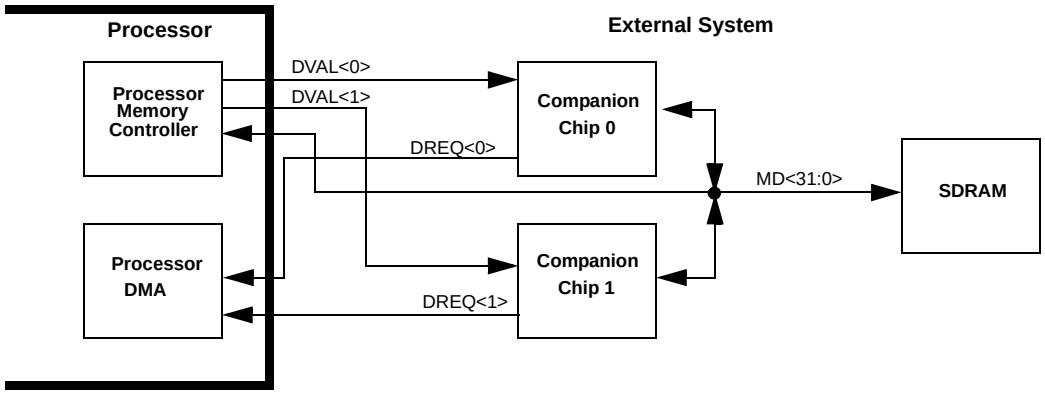
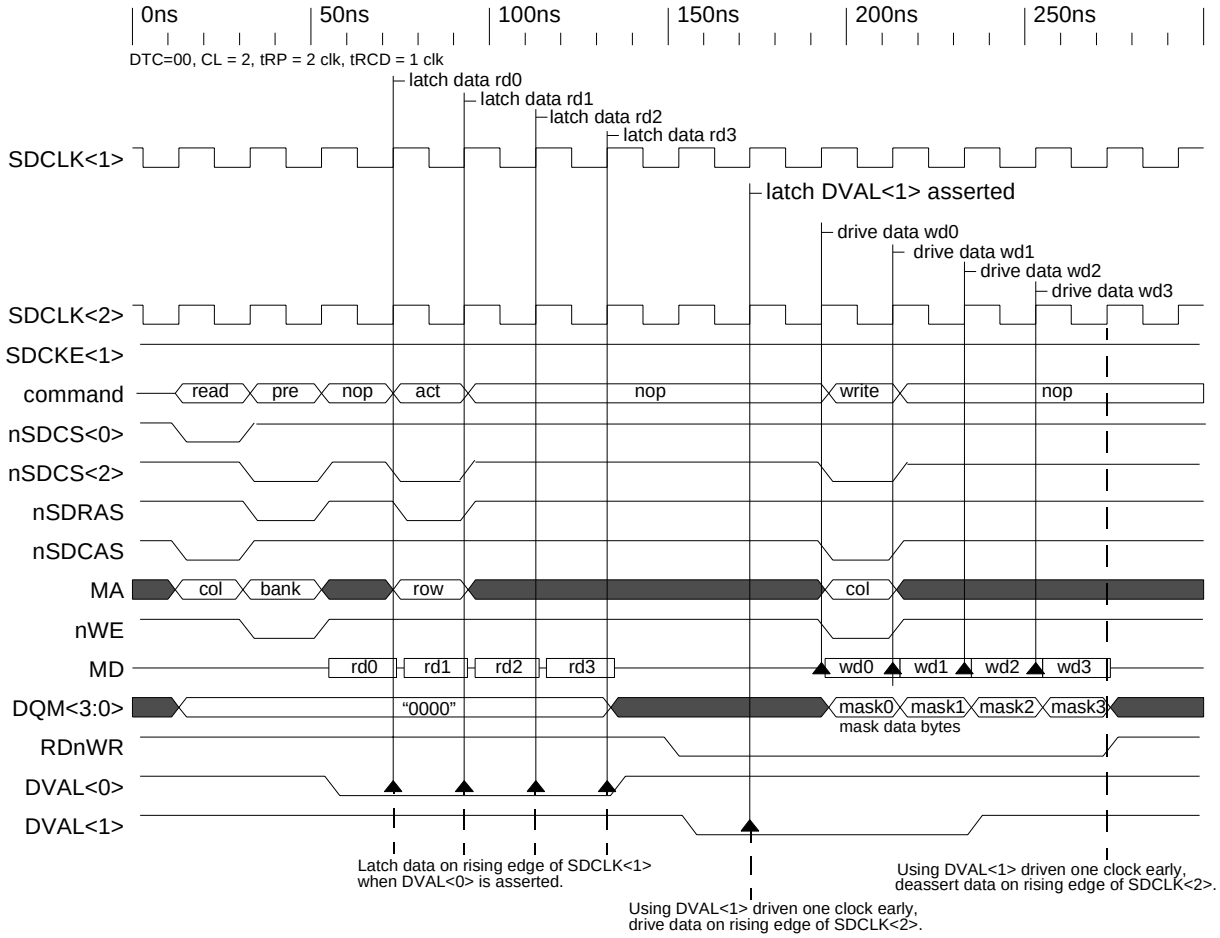


Figure 5-7. Real-Time Fly-By DMA Operation for SDRAM



5.4.6 How DMA Handles Trailing Bytes

DMA normally transfers bytes equal to the transaction size specified by DCMDx[SIZE]. But if the descriptor is reaching its end, the number of trailing bytes in the DCMDx[LEN] field could be smaller than the transfer size. The DMA can transfer the exact number of trailing bytes if the DCMDx[FLOWSRC] and DCMDx[FLOWTRG] bits are both 0, or if it receives a corresponding request from the on-chip/off-chip peripheral or companion chip. The following cases are possible:

- Memory-to-memory moves: The DMA transfers bytes equal to the smaller of DCMDx[LEN] or DCMDx[SIZE].
- Companion-chip-related transfers (flow through or fly by): The companion chip must assert the request to allow the DMA to handle the trailing bytes. If the request is asserted, the DMA transfers a number of bytes equal to the smaller of DCMDx[LEN] and DCMDx[SIZE].
- Memory-to-on-chip-peripheral transfers: Most of the application processor peripherals send a request for trailing bytes. Refer to the appropriate chapter of this document for details of a specific peripheral's operation. The DMA transfers a number of bytes equal to the smaller of DCMDx[LEN] and DCMDx[SIZE].

- On-chip-peripheral-to-memory transfers: Special hand-shaking signals and interrupts are employed for transferring trailing bytes from an on-chip peripheral to memory. The conditions that use the hand-shaking signals and interrupts are explained below:
  - — End of Packet (EOP)—The peripheral receives its last data sample from an external Codec and detects an EOP based on its receive protocol. Any remaining data samples in the peripheral receive FIFO are treated as trailing bytes. The peripheral can be programmed to initiate a DMA request, even if it has fewer bytes than its receive trigger threshold. The DMA responds to this request and reads out the trailing bytes. CPU intervention is not required as long as the descriptor chain has not ended.
  - — Time-Out (TO)—Peripherals that do not support EOP protocols use a time-out mechanism to determine if they have received their last data sample. Refer to the peripheral chapters for details of the time-out implementation. Any remaining data samples in the peripheral receive FIFO are treated as trailing bytes. The peripheral can be programmed to initiate a DMA request, even if it has fewer bytes than its receive trigger threshold. The DMA responds to the DMA request and reads out the trailing bytes. CPU intervention is not required as long as the descriptor chain has not ended.

Note: When a peripheral signals either an EOP or a TO, the DMA controller sets the end-of-receive (EOR) status bit in the corresponding channel's Control Status register (DCSRx). See Table 5-17 for details.

- — End-of-Descriptor Chain (EOC)—Indicates that a DMA channel is at the end of its last descriptor. After the current transfer, DCMDx[LEN] = 0 and DDADRx[STOP] = 1. DMA signals the peripheral on an EOC and the peripheral interrupts the CPU to retrieve any trailing bytes. Refer to the individual peripheral chapters for details on the processor interrupt implementation. EOC is the only trailing-bytes case that requires programmed I/O to retrieve data.
- — Request after channel stops (RAS)—Status bit in the DMA controller control status register (DCSRx). This bit is set when a peripheral asserts a DMA request after the channel to which the peripheral is mapped has stopped. See Table 5-17 for details.

The following cases illustrate the handling of various trailing bytes using EOR, EOC, and RAS.

Case 1—The peripheral signals a DMA request to service trailing bytes in its receive FIFO (RxFIFO). The current descriptor's DCMDx[Len] is equal to or greater than the trailing-bytes count.

1. The peripheral signals a receive DMA request.
2. The DMA controller responds and reads out all trailing bytes including the last byte.
3. The peripheral signals an EOR.
4. The DMA controller transfers all trailing bytes to the channel's target and then updates DCSRx[EORInt].
5. The DMA channel can be configured to stop, jump, or just wait for another request after receiving EOR, depending on DCSRx[EORStopEn] and DCSRx[EORJmpEn]. The EORINT bit must be cleared before restarting a channel.
6. DCSRx[EORInt] set indicates that all trailing bytes were read and transferred to the required target.

Case 2—The peripheral signals a DMA request to signal an EOR only (RxFIFO is empty). The current descriptor's DCMDx[Len] is greater than zero

1. The peripheral signals a receive DMA request.

2. The peripheral responds to the DMA controller that the RxFIFO is empty, and the DMA controller does read any data from the RxFIFO.
3. The peripheral signals an EOR.
4. The DMA controller transfers any previously read data (temporarily stored in a local DMA buffer) to the channel's target and then updates DCSRx[EORInt].
5. The DMA channel can be configured to stop, jump, or just wait for another request after receiving EOR, depending on DCSRx[EORStopEn] and DCSRx[EORJmpEn]. The EORINT bit must be cleared before restarting a channel.
6. DCSRx[EORInt] set indicates that all trailing bytes were read and transferred to the required target.

Case 3—The peripheral signals a DMA request to service trailing bytes in its RxFIFO. The current descriptor's DCMDx[Len] is less than the trailing-bytes count. More descriptors are available in the descriptor chain

1. The peripheral signals a receive DMA request.
2. The DMA controller responds and reads only the number of trailing bytes programmed by DCMDx[Len].
3. The current descriptor's DCMDx[Len] decrements to zero. This condition forces the DMA controller to transfer all the data read from the peripheral's RxFIFO to the channel's target.
4. The DMA controller fetches the next descriptor.
5. Because the peripheral still has trailing bytes in its RxFIFO, it must make another request. The peripheral must continue to issue such requests until the DMA controller reads out all the trailing bytes and until an EOR is signaled.
6. DCSRx[EORInt] set indicates that all trailing bytes were read and transferred to the required target. The EORINT bit must be cleared before restarting a channel.

Case 4—The peripheral signals a DMA request to service trailing bytes in its RxFIFO. The current descriptor's DCMDx[Len] is less than the trailing-bytes count. No more descriptors are available in the descriptor chain.

1. The peripheral signals a receive DMA request.
2. The DMA controller responds and reads only the number of trailing bytes programmed by DCMDx[Len].
3. The current descriptor's DCMDx[Len] decrements to zero. This condition forces the DMA controller to transfer all the data read from the peripheral RxFIFO to the channel target.
4. The DMA channel stops, as there are no more descriptors in the descriptor chain.
5. If the DMA controller signaled an end-of-chain (EOC) while reading the last byte, the peripheral must set a status bit in a local (peripheral) status register. This setting indicates for software to handle the remainder of the trailing bytes in the peripheral's RxFIFO.
6. If the DMA controller fails to signal an EOC, the peripheral signals a DMA request. Because the DMA channel has already stopped, the DMA controller sets DCSRx[RAS]. This setting indicates for software to handle the remainder of the trailing bytes in the peripheral's RxFIFO.

Note: The DMA controller signals an EOC only if the descriptor chain's last descriptor's DCMDx[Len] is greater than zero.

Peripherals that require the DMA services for processing trailing bytes must support the above-mentioned signals. The DMA services can process trailing bytes on the following peripherals:

- BTUART, FFUART, STUART, fast infrared port
- AC ‘97 controller
- SSP1, SSP2, SSP3
- USB device controller (UDC)
- Mobile Scalable Link (MSL)
- USIM
- MMC and SDIO (supports TO; does not support EOC)
- Memory Stick (supports TO; does not support EOC)
- I^2S
- Quick capture interface

5.4.7 Quick Reference to DMA Programming

Table 5-5, Table 5-6, Table 5-7, and Table 5-8 tabulate width, alignment and address increment modes for various DMA data transfers.

Table 5-5. Configuration for Peripheral Bus Peripheral (PBP) Related Data Transfers

Source	Target	Source Alignment (Bytes)	Target Alignment (Bytes)	DCMD [IncSrcAddr] (Binary)	DCMD [IncTrgAddr] (Binary)	DCMD [Width] (Binary)
PBP	Memory	4	1	0 or 1	1	01, 10 or 11
Memory	PBP	1	4	1	0 or 1	01, 10 or 11
PBP	Expansion memory FIFO	4	1	0 or 1	0	01, 10 or 11
Expansion memory FIFO	PBP	1	4	0	0 or 1	01, 10 or 11

NOTE:
Memory refers to all types of memory explained in Chapter 6, “Memory Controller” (including internal memory, external memory, variable-latency I/O memory, and expansion memory implemented in non-FIFO mode).

If a memory address is byte-aligned, then the DALGN register must be programmed. Refer to Section 5.5.10 for further details.

If either DCMDx[IncSrcAddr] or DCMDx[IncTrgAddr] is set, then the DMA controller increments the source or target address after each transaction by a number equal to the width of the peripheral bus peripheral (PBP). For example, if the DMA controller is transferring 32 bytes of data from memory to a 4-byte wide PBP, then the DMA controller writes 4 bytes to the PBP 8 times and increments the target address by 4 bytes after each of the 8 transactions.

Table 5-6. Configuration for Memory-to-Memory Data Transfers

Source	Target	Source Alignment (Bytes)	Target Alignment (Bytes)	DCMD [IncSrcAddr] (Binary)	DCMD [IncTrgAddr] (Binary)	DCMD[Width] (Binary)
Memory	Memory	1	1	1	1	00
Expansion memory FIFO	Memory	1	1	0	1	00
Memory	Expansion memory FIFO	1	1	1	0	00
Expansion memory FIFO	Expansion memory FIFO	1	1	0	0	00

NOTE:
Memory refers to all types of memory explained in Chapter 6, "Memory Controller" (including internal memory, external memory, variable-latency I/O memory, and expansion memory implemented in non-FIFO mode).

If a memory address is byte-aligned, then DALGN register must be programmed. Refer to Section 5.5.10 for further details.

Table 5-7. Configuration for Internal Bus Peripheral (IBP) Related Data Transfers

Source	Target	Source Alignment (Bytes)	Target Alignment (Bytes)	DCMD [IncSrcAddr] (Binary)	DCMD [IncTrgAddr] (Binary)	DCMD [Width] (Binary)
IBP	Memory	8	8	0 or 1	1	00
Memory	IBP	8	8	1	0 or 1	00
IBP	Expansion memory FIFO	8	8	0 or 1	0	00
Expansion memory FIFO	IBP	8	8	0	0 or 1	00

NOTE:
Memory refers to all types of memory explained in Chapter 6, "Memory Controller" (including internal memory, external memory, variable-latency I/O memory, and expansion memory implemented in non-FIFO mode).

If either DCMDx[IncSrcAddr] or DCMDx[IncTrgAddr] is set, then the DMA controller increments the source or target address after each bursting transaction by a number equal to the transaction burst size (8, 16 or 32 bytes) or DCMDx[Len]. The latter is used if DCMDx[Len] is less than the burst size. For example, if the DMA controller is transferring 48 bytes of data from memory to IBP in bursts of 32 bytes, then the DMA controller increments the target address by 32 after the first burst and then by 16 after the second burst.

Table 5-8. Configuration for Companion Chip (CC) Related Data Transfers

Source	Target	Source Alignment (Bytes)	Target Alignment (Bytes)	DCMD [IncSrcAddr] (Binary)	DCMD [IncTrgAddr] (Binary)	DCMD[Width] (Binary)
CC or External Peripheral	Memory	8	8	0 or 1	1	00
Memory	CC or External Peripheral	8	8	1	0 or 1	00
CC or External Peripheral	Expansion memory FIFO	8	8	0 or 1	0	00
Expansion memory FIFO	CC or External Peripheral	8	8	0	0 or 1	00

NOTE:
Memory refers to all types of memory explained in Chapter 6, "Memory Controller" (including internal memory, external memory, variable-latency I/O memory, and expansion memory implemented in non-FIFO mode).

For flow-through data-transfer mode, any memory can be used. For fly-by data-transfer mode, the internal memory must not be used as either a source or a target.

The companion chip or external peripheral must be connected as a variable-latency I/O memory.

If either DCMDx[IncSrcAddr] or DCMDx[IncTrgAddr] is set, then the DMA controller increments the source or target address, after each bursting transaction, by a number equal to the transaction burst size (8, 16 or 32 bytes) or DCMDx[Len]. The latter is used if DCMDx[Len] is less than the burst size. For example, if the DMA is transferring 48 bytes of data from memory to the companion chip in bursts of 32 bytes, then the DMA controller increments the target address by 32 after the first burst and then by 16 after the second burst.

Table 5-9 provides a quick reference for programming the DMA controller for the on-chip peripherals.

Table 5-9. DMA Quick Reference for On-Chip Peripherals (Sheet 1 of 4)

Unit	Function	FIFO Address	Width (Bytes)	DCMDx Width (Binary)	Burst Size (Bytes)	Source or Target	DRCMRx
I^2S	receive	0x4040_0080	4	11	8, 16, 32	Source	0x4000_0108
	transmit	0x4040_0080	4	11	8, 16, 32, or trailing	Target	0x4000_010C
BTUART	receive	0x4020_0000	1 or 4	01 or 11	8, 16, 32, or trailing	Source	0x4000_0110
	transmit	0x4020_0000	1 or 4	01 or 11	8, 16, 32, or trailing	Target	0x4000_0114
FFUART	receive	0x4010_0000	1 or 4	01 or 11	8, 16, 32, or trailing	Source	0x4000_0118
	transmit	0x4010_0000	1 or 4	01 or 11	8, 16, 32, or trailing	Target	0x4000_011C

Table 5-9. DMA Quick Reference for On-Chip Peripherals (Sheet 2 of 4)

Unit	Function	FIFO Address	Width (Bytes)	DCMDx Width (Binary)	Burst Size (Bytes)	Source or Target	DRCMRx
AC '97	Microphone	0x4050_0060	4	11	8, 16, 32	Source	0x4000_0120
	Modem receive	0x4050_0140	4	11	8, 16, 32	Source	0x4000_0124
	Modem transmit	0x4050_0140	4	11	8, 16, 32	Target	0x4000_0128
	Audio receive	0x4050_0040	4	11	8, 16, 32	Source	0x4000_012C
	Audio transmit	0x4050_0040	4	11	8, 16, 32	Target	0x4000_0130
MSL	Receive 1	0x4140_0004	4	11	8, 16, 32, or trailing	Source	0x4000_01C0
	Transmit 1	0x4140_0004	4	11	8, 16, 32, or trailing	Target	0x4000_01C4
	Receive 2	0x4140_0008	4	11	8, 16, 32, or trailing	Source	0x4000_01C8
	Transmit 2	0x4140_0008	4	11	8, 16, 32, or trailing	Target	0x4000_01CC
	Receive 3	0x4140_000C	4	11	8, 16, 32, or trailing	Source	0x4000_01D0
	Transmit 3	0x4140_000C	4	11	8, 16, 32, or trailing	Target	0x4000_01D4
	Receive 4	0x4140_0010	4	11	8, 16, 32, or trailing	Source	0x4000_01D8
	Transmit 4	0x4140_0010	4	11	8, 16, 32, or trailing	Target	0x4000_01DC
	Receive 5	0x4140_0014	4	11	8, 16, 32, or trailing	Source	0x4000_01E0
	Transmit 5	0x4140_0014	4	11	8, 16, 32, or trailing	Target	0x4000_01E4
	Receive 6	0x4140_0018	4	11	8, 16, 32, or trailing	Source	0x4000_01E8
	Transmit 6	0x4140_0018	4	11	8, 16, 32, or trailing	Target	0x4000_01EC
	Receive 7	0x4140_001C	4	11	8, 16, 32, or trailing	Source	0x4000_01F0
	Transmit 7	0x4140_001C	4	11	8, 16, 32, or trailing	Target	0x4000_01F4
USIM	Receive	0x4160_0000	1	01	8 or trailing	Source	0x4000_01F8
	Transmit	0x4160_0004	1	01	8 or trailing	Target	0x4000_01FC

Table 5-9. DMA Quick Reference for On-Chip Peripherals (Sheet 3 of 4)

Unit	Function	FIFO Address	Width (Bytes)	DCMDx Width (Binary)	Burst Size (Bytes)	Source or Target	DRCMRx
USB	Endpoint 0	0x4060_0300	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0160
	Endpoint A	0x4060_0304	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0164
	Endpoint B	0x4060_0308	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0168
	Endpoint C	0x4060_030C	4	11	8, 16, 32, or trailing	Source or Target	0x4000_016C
	Endpoint D	0x4060_0310	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0170
	Endpoint E	0x4060_0314	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0174
	Endpoint F	0x4060_0318	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0178
	Endpoint G	0x4060_031C	4	11	8, 16, 32, or trailing	Source or Target	0x4000_017C
	Endpoint H	0x4060_0320	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0180
	Endpoint I	0x4060_0324	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0184
	Endpoint J	0x4060_0328	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0188
	Endpoint K	0x4060_032C	4	11	8, 16, 32, or trailing	Source or Target	0x4000_018C
	Endpoint L	0x4060_0330	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0190
	Endpoint M	0x4060_0334	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0194
	Endpoint N	0x4060_0338	4	11	8, 16, 32, or trailing	Source or Target	0x4000_0198
	Endpoint P	0x4060_033C	4	11	8, 16, 32, or trailing	Source or Target	0x4000_019C
	Endpoint Q	0x4060_0340	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01A0
	Endpoint R	0x4060_0344	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01A4
	Endpoint S	0x4060_0348	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01A8
	Endpoint T	0x4060_034C	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01AC
	Endpoint U	0x4060_0350	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01B0
	Endpoint V	0x4060_0354	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01B4
	Endpoint W	0x4060_0358	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01B8
	Endpoint X	0x4060_035C	4	11	8, 16, 32, or trailing	Source or Target	0x4000_01BC
STUART	Endpoint	0x4070_0000	1 or 4	01 or 11	8, 16, 32, or trailing	Source	0x4000_014C
	Endpoint	0x4070_0000	1 or 4	01 or 11	8, 16, 32, or trailing	Target	0x4000_0150
Fast Infrared Port	Endpoint	0x4080_000C	1 or 4	01 or 11	8, 16, 32, or trailing	Source	0x4000_0144
	Endpoint	0x4080_000C	1 or 4	01 or 11	8, 16, 32, or trailing	Target	0x4000_0148
SSP1	Endpoint	0x4100_0010	1, 2, or 4	01, 10, or 11	8, 16, 32, or trailing	Source	0x4000_0134
	Endpoint	0x4100_0010	1, 2, or 4	01, 10, or 11	8, 16, 32, or trailing	Target	0x4000_0138
MMC/SDIO	Receive	0x4110_0040 (width = 1 byte) or 0x4110_0140 (width = 4 bytes)	1 or 4	01 or 11	32 or trailing	Source	0x4000_0154
	Transmit	0x4110_0044 (width = 1 byte) or 0x4110_0144 (width = 4 bytes)	1 or 4	01 or 11	32 or trailing	Target	0x4000_0158

Table 5-9. DMA Quick Reference for On-Chip Peripherals (Sheet 4 of 4)

Unit	Function	FIFO Address	Width (Bytes)	DCMDx Width (Binary)	Burst Size (Bytes)	Source or Target	DRCMRx
SSP2	Receive	0x4170_0010	1, 2, or 4	01, 10, or 11	8, 16, 32 or trailing	Source	0x4000_013C
	Transmit	0x4170_0010	1, 2, or 4	01, 10, or 11	8, 16, 32 or trailing	Target	0x4000_0140
Memory Stick	Receive	0x4180_0018	1	01	8 or trailing	Source	0x4000_1100
	Transmit	0x4180_001C	1	01	8 or trailing	Target	0x4000_1104
SSP3	Receive	0x4190_0010	1, 2, or 4	01, 10, or 11	8, 16, 32 or trailing	Source	0x4000_1108
	Transmit	0x4190_0010	1, 2, or 4	01, 10, or 11	8, 16, 32 or trailing	Target	0x4000_110C
Quick Capture Interface	Receive 1	0x50000028	8	00	8, 16, 32 or trailing	Source	0x4000_1110
	Receive 2	0x50000030	8	00	8, 16, 32 or trailing	Source	0x4000_1114
	Receive 3	0x50000038	8	00	8, 16, 32 or trailing	Source	0x4000_1118
Trusted Platform Module (TPM)	Receive	0x43000008	4	11	8, 16, 32	Source	0x4000_111C
	Transmit 1	0x43000000	4	11	8, 16, 32	Target	0x4000_1120
	Transmit 2	0x43000004	4	11	8, 16, 32	Target	0x4000_1124

5.4.8 Programming Examples

Example 5-1. Setting Up and Starting a Channel

The following code example shows how to set up and start a channel to transfer LEN words that start at the address in the DSADRx register to peripheral address in the DTADRx register. In this example, the stop bit in the DDADRx register is set so that the DMA channel stops after it completely transfers LEN bytes of data associated with this descriptor.

```
// build real descriptor
desc[0].ddadr = STOP;
desc[0].dsadr = DSADR;
desc[0].dtadr = DTADR;
desc[0].dcmd = DCMD;
// start the channel
DMANEXT[CHAN] = &desc[0];
DRUN = 1;
```

Example 5-2. Creating a Zero-Length Descriptor

The following code example shows how to initialize a descriptor list for a channel that is running:

```
// Allocate a new descriptor, and make it an End-
// Descriptor whose "ddadr" field points back at itself
newDesc = New Desc();
newDesc->ddadr = newDesc | STOP;
// make it a zero length descriptor
newDesc->dcmd = ZERO;
// Start the channel
DMANEXT[CHAN] = newDesc;
DRUN = 1;
```

The channel starts up, loads the descriptor into its registers, detects that the transfer length is zero and the stop bit is set, and stops the channel. No data transfer occurs in this case. To restart the channel, write to its DDADRx register, then set the DCSRx[RUN] bit.

Example 5-3. Initializing a Channel to Be Used by the Direct DMA Master

The most effective way to move data between a peripheral device and memory is to use the built-in descriptor-based DMA system. In most applications, a free DMA channel is readily available.

Some applications demand true direct-memory access. Each application has different requirements, so a descriptor-based DMA may be best for some applications while a non-descriptor-based DMA is best for others. For applications that cannot tolerate the time required to fetch a descriptor before each DMA transfer, choose the non-descriptor-based DMA method. For applications that can tolerate it, a descriptor-based DMA method can reduce the amount of core intervention.

By programming the descriptor-based DMA system, it is possible to offer true direct-memory access to devices that require it.

In the following example, the companion chip requires the DMA to read four descriptor words, program a channel as indicated by descriptor, and continue the transfer based on the new descriptor. Such a companion chip makes the following requirements:

1. When the companion chip asserts DREQ, fetch four descriptor words from one of its ports.
2. Based on the information contained in the descriptor, transfer data from the source address to the destination address without waiting for another request from the companion chip.
3. Transfer the number of bytes mentioned in DCMDx[LEN], then return to Step 1.

Such an external device can use a constant descriptor in memory:

A channel that is used by a direct DMA master must be initialized in a special way. The following example of code shows how to initialize such a channel:

```
struct {          long ddadr;
                  long dsadr;
                  long dtadr;
                  short len;
                  short dcmd;
} desc[2];
desc[0].ddadr = &desc[1];
desc[0].dsadr = I_ADR + I_DESC_OFFS;
desc[0].dtadr = &desc[1].dsadr;
desc[0].len = 8;
desc[0].dcmd = CMD_IncTrgAdr | CMD_FlowThru;
desc[1].ddadr = &desc[0];
desc[1].dtadr = I_ADR + I_DATA_OFFS;
desc[1].dsadr = 0;
desc[1].len = 0;
desc[1].dcmd = 0;
```

When the external device has data to transfer, it makes a DMA request in the standard way. The DMA controller wakes up and reads four words from the device's I_DESC_OFFS address. The DMA controller only transfers four words because the first descriptor has an eight-byte count. The four words supplied by the external device are written over the DSADRx, DTADRx, and DCMDx registers of the next descriptor. The DMA controller then steps into the next (dynamically

modified) descriptor and uses the devices's I_DATA_OFFS address to process the requested transfer. When the data transfer is complete, the DMA controller steps back to the first descriptor and the process repeats.

Example 5-4. Adding a Descriptor to End-of-Descriptor List (Channel Running)

Note: The following example assumes that a descriptor-fetch transfer is active.

DMA descriptor lists are used as queues of full buffers for network transmitters and as queues of empty buffers for network receivers. Because each buffer can be small, on-the-fly manipulation of DMA descriptor lists must be as efficient as possible.

To add a descriptor to the end of a descriptor list for a running channel:

1. Write DCSRx[RUN] = 0.
2. Wait for the channel to stop. The stop status is indicated in DCSRx[STOPINTR].
3. Create the end descriptor in the memory with the stop bit set.
4. Manipulate the DDADRx register of the last descriptor of the current chain in the memory to ensure that the DDADRx register points to the newly created end descriptor in step 3.
5. Create a new descriptor with values in the DDADRx, DSADRx, DTADRx, and DCMDx registers that match those in the stopped DMA channel. The new descriptor is the next descriptor for this descriptor list.
6. Examine the DMA channel registers and determine if the channel stopped in the last descriptor of the chain. If it did, manipulate the DDADRx register of this descriptor so that it points to the newly created end descriptor.
7. Program the channel DDADRx register with the next descriptor created in Step 5.
8. Set DCSRx[RUN] = 1.

Example 5-5. Using Software Implementation of Full and Empty Bits

This example shows how to use the compare-descriptor and branch-descriptor modes to allow the software to implement full and empty bits. As shown in Figure 5-8, this implementation can create a circular descriptor chain to transfer a large amount of data. The full and empty bits are maintained in a memory table and are cleared prior to a DMA transfer. The DMA controller uses the compare-descriptor mode to determine that the full and empty bits are cleared before it performs a data transfer. If either bit is set, the DMA controller uses the branch-descriptor mode to branch to a descriptor in which the DDADRx[STOP] bit is set, which terminates the transfer. If the full and empty bits are both cleared, the DMA controller continues to perform a regular data transfer. After the DMA controller completes the data transfer, it updates the full and empty memory table by setting the full and empty bits, which indicates a successful transfer.

This example assumes the following memory addresses:

```
SRC0, SRC1: source-address locations
TRG0, TRG1: target-address locations
FEUPDT: address pointing to data 0x0000_0003. Bit]1:0] = [Full:Empty].
```

In the example, the memory table contains the following full- and empty-bit information before the DMA transfers. Only the two least significant bits are valid. Bit[1:0] = [Full:Empty].

```
FETBL0: 0x0000_0000
FETBL1: 0x0000_0000
```

After successful DMA transfers, the same memory table contains the following information:

```
FETBL0: 0x0000_0003
FETBL1: 0x0000_0003
```

The following code is a chain of descriptors for full- and empty-bit implementation:

First Descriptor Set

First Descriptor

```
//Compare and Branch Descriptor modes enabled.
//No data transferred by this descriptor.
//Source is indirectly addressed and target is directly addressed
//On a successful compare of &FETBL0 with 0x0000,
// Descriptor chain branches to desc[1] + 4*32bits, i.e desc[2].
//If Compare fails, then descriptor chain jumps to desc[1].
//Desc[1] stops the channel as Full and Empty bits were not both 0.
desc[0].ddadr = &desc[1], BrEn = 1;
desc[0].dsadr = FETBL0;
desc[0].dtadr = 0x0000;
desc[0].dcmd = CmpEn=1, AddrMode = b01;
```

Second Descriptor

```
//Error setting descriptor, which stops the channel as
//(Full:Empty) != 0b00
//No data transferred. Stop interrupt triggered.
desc[1].ddadr = (Stop = 1);
desc[1].dsadr = ignored;
desc[1].dtadr = ignored;
desc[1].dcmd = Len=0;
```

Third Descriptor

```
//Data transferring descriptor
desc[2].ddadr = &desc[3];
desc[2].dsadr = SRC0;
desc[2].dtadr = TRG0;
desc[2].dcmd = Len=4K bytes;
```

Fourth Descriptor

```
//FullEmpty table updating descriptor
desc[3].ddadr = &desc[4];
desc[3].dsadr = FEUPDT;
desc[3].dtadr = FETBL0;
desc[3].dcmd = Len=4 bytes;
```

Second Descriptor Set

First Descriptor

```
//Compare and Branch Descriptor modes enabled.
//No data transferred by this descriptor.
//Source is indirectly addressed and target is directly addressed
//On a successful compare of &FETBL1 with 0x0000,
```

```
// Descriptor chain branches to desc[5] + 4*32bits, i.e desc[6].
//If Compare fails, then descriptor chain jumps to desc[5].
//Desc[5] stops the channel as Full and Empty bits were not both 0.
desc[4].ddadr = &desc[5], BrEn = 1;
desc[4].dsadr = FETBL1;
desc[4].dtadr = 0x0000;
desc[4].dcmd = CmpEn=1, AddrMode = b01;
```

Second Descriptor

```
//Error setting descriptor, which stops the channel as
//(Full:Empty) != 0b00
//No data transferred. Stop interrupt triggered.
desc[5].ddadr = (Stop = 1);
desc[5].dsadr = ignored;
desc[5].dtadr = ignored;
desc[5].dcmd = Len=0;
```

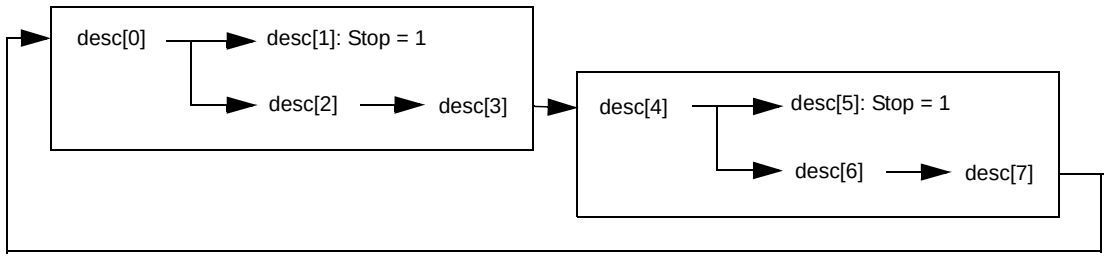
Third Descriptor

```
//Data transferring descriptor
desc[6].ddadr = &desc[7];
desc[6].dsadr = SRC1;
desc[6].dtadr = TRG1;
desc[6].dcmd = Len=4K bytes;
```

Fourth Descriptor

```
//FullEmpty table updating descriptor
//Notice that the chain jumps back to desc[0].
//All ok, if software has loaded new data at the address pointed to by
//desc[0] and has also updated FETBL0
desc[7].ddadr = &desc[0];
desc[7].dsadr = FEUPDT;
desc[7].dtadr = FETBL1;
desc[7].dcmd = Len=4 bytes;
```

Figure 5-8. Descriptor Chain for Software Implementation of Full and Empty Bits



5.5 Register Descriptions

This section describes the DMA controller registers. Table 5-21 summarizes the register names, addresses, and descriptions.

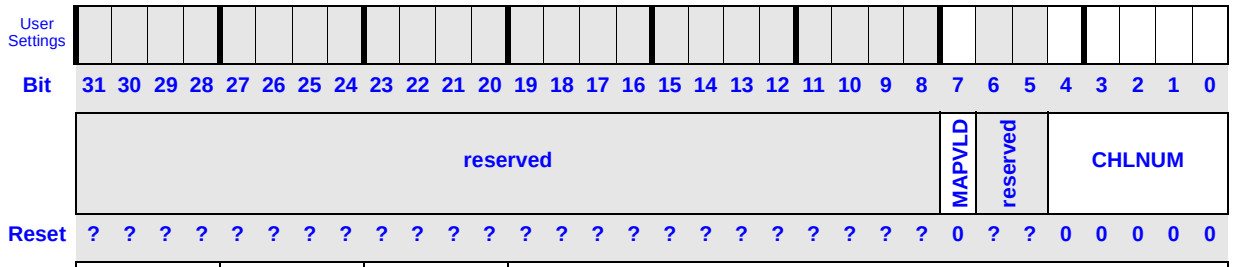
5.5.1 DMA Request to Channel Map Register (DRCMRx)

These registers map the DMA request to a channel.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 5-10. DRCMR0–63, DRCMR64–70, and DRCMR74 Bit Definitions

Physical Address 0x4000_0100–0x4000_01FC 0x4000_1100–0x4000_1118 0x4000_1128	DRCMR0–DRCMR63 DRCMR64–DRCMR70 DRCMR74	DMA Controller



Bits	Access	Name	Description
31:8	—	—	reserved
7	R/W	MAPVLD	Map Valid Channel Defines whether the request is mapped to a valid channel. If the bit is set, the request is mapped to a valid channel indicated by DRCMRx[CHLNUM]. If the bit is cleared, the request is unmapped. This bit can also mask the request. 0 = Request is unmapped. 1 = Request is mapped to a valid channel indicated by DRCMRx[4:0].
6:5	—	—	reserved
4:0	R/W	CHLNUM	Channel Number Indicates the valid channel number if DRCMRx[MAPVLD] is set. Do not map two active requests to the same channel, since it produces unpredictable results.

5.5.2 DMA Descriptor Address Registers (DDADRx)

These registers contain the memory address of the next descriptor for a channel. The address must be aligned to a 128-bit (4-word) boundary. DDADRx must not contain the address of any other internal peripheral register or DMA register, which causes a bus error.

The DDADRx register is reserved if the channel is performing a no-descriptor-fetch transaction.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 5-11. DDADR0–31 Bit Definitions

Physical Address 0x4000_02x0–0x4000_03x0 — DDADR0–DDADR31 — DMA Controller

	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
User Settings																																
Bit	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
	Descriptor Address																												reserved		BREN	STOP
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
31:4	R/W	Descriptor Address	Descriptor Address Contains address of next descriptor
3:2	—	—	reserved
1	R/W	BREN	Enable Descriptor Branching Works with DCSRx[CMPST] to determine which descriptor is fetched next. If both DDADRx[BREN] and DCSRx[CMPST] are set, the DMA controller fetches the next descriptor from (DDADRx + 32 bytes). If either of the bits is cleared, DMA controller fetches the next descriptor from the DDADRx register. DDADRx[BREN] is relevant only for descriptor-fetch transactions (DCSRx[NODESCFETCH] = 0). See Figure 5-4 for details about the operational flow. 0 = Disable descriptor branching. Fetch the next descriptor from register DDADRx. 1 = Enable descriptor branching.
0	R/W	STOP	Stop Channel If DDADRx[STOP] is cleared, a new descriptor fetch based on register DDADRx and other control information is initiated after the current descriptor is completely processed. 0 = Run channel. 1 = Stop channel after completely processing this descriptor and before fetching the next descriptor (DCMDx[LEN] = 0).

5.5.3 DMA Source Address Register (DSADRx)

These registers are read-only for descriptor-fetch transactions and read/write for no-descriptor-fetch transactions.

The registers (Table 5-12) contain the source address of the current descriptor for a channel. The source address is the address of the on-chip peripheral or the address of a memory location. The bits in this register are undefined at power on.

If the source address is the address of a companion chip or external peripheral, the source address must be aligned to an eight-byte boundary; bits [2:0] of the address are reserved.

If the source address is the address for an on-chip peripheral, then the address must be 32-bit aligned, so bits [1:0] of the address are reserved. DSADR cannot contain addresses of any other internal DMA registers, as they cause a bus error.

If the source address is the address of a memory location, and if the alignment register is properly configured, the address can be aligned to a byte boundary (see Section 5.5.10). Improper configuration of the alignment register defaults the source address to an eight-byte boundary.

Other restrictions on byte boundary alignment could apply for special DMA operations (see Section 5.4.4.4).

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 5-12. DSADR0–31 Bit Definitions

Physical Address 0x4000_02x4–0x4000_03x4	DSADR0–DSADR31	DMA Controller

User Settings																															
Bit	**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**
	SRCADDR																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:2	R/W	SRCADDR	Source address of the on-chip peripheral or address of a memory location.
2	R/W	SRCADDR or reserved	SRCADDR[2], if DSADRx[SRCADDR] is a memory location and alignment register is configured. See Section 5.5.10 for programming details and restrictions. SRCADDR[2], if DSADRx[SRCADDR] is an on-chip peripheral. RESERVED for all companion-chip or external peripheral related transfers. RESERVED for special DMA modes, such as compare modes.
1:0	R/W	SRCADDR or reserved	SRCADDR[1:0], if DSADRx[SRCADDR] is a memory location and alignment register is configured. See Section 5.5.10 for programming details and restrictions. RESERVED, if DSADRx[SRCADDR] is an on-chip peripheral. RESERVED for all companion-chip or external peripheral related transfers. RESERVED for special DMA modes, such as compare modes

5.5.4 DMA Target Address Registers (DTADRx)

These registers (Table 5-13) are read only for descriptor-fetch transfers and read/write for no-descriptor-fetch transfers.

The registers contain the target address of the current descriptor for a channel. The target address is the address of the on-chip peripheral or the address of a memory location. The bits in this register are undefined at power on.

If the target address is the address of a companion chip or external peripheral, the target address must be aligned to an eight-byte boundary; bits [2:0] of the address are reserved.

If the target address is the address for an on-chip peripheral, then the address must be 32-bit aligned; bits [1:0] of the address are reserved. DTADR cannot contain addresses of any other internal DMA registers as they do cause a bus error.

If the target address is the address of a memory location, and if the alignment register is properly configured, the address can be aligned to a byte boundary (see Section 5.5.10). Improper configuration of the alignment register defaults the target address to an eight-byte boundary.

Other restrictions on byte boundary alignment could apply for special DMA operations (see Section 5.4.4.4).

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 5-13. DTADR0–31 Bit Definitions

Physical Address 0x4000_02x8–0x4000_03x8	DTADR0–DTADR31	DMA Controller

User Settings																															
Bit	**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**
	TRGADDR																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:2	R/W	TRGADDR	Target address of the on-chip peripheral or address of a memory location.
2	R/W	TRGADDR or reserved	TRGADDR[2], if DTADRx[TRGADDR] is a memory location and alignment register is configured. See Section 5.5.10 for programming details and restrictions. TRGADDR[2], if DTADRx[TRGADDR] is an on-chip peripheral. RESERVED for all companion-chip or external peripheral related transfers. RESERVED for special DMA modes, such as compare modes.
1:0	R/W	TRGADDR or reserved	TRGADDR[1:0], if target address is a memory location and alignment register is configured. See Section 5.5.10 for programming details and restrictions. RESERVED if target address is an on-chip peripheral. RESERVED for all companion-chip or external peripheral related transfers. RESERVED for special DMA modes, such as compare modes.

5.5.5 DMA Command Registers (DCMDx)

These registers (Table 5-14) are read only for descriptor-fetch transfers and read/write for no-descriptor-fetch transfers.

The registers contain the command and length of the current descriptor for a channel.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 5-14. DCMD0–31 Bit Definitions (Sheet 1 of 4)

Physical Address 0x4000_02xC–0x4000_03xC																															
DCMD0–DCMD31																															
DMA Controller																															
User Settings																															
Bit	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
	INCSRCADDR	INCTRGADDR	FLOWSRC	FLOWTRG	reserved		CMPEN	reserved	ADDRMODE	STARTIRQEN	EndIrqEn	FLYBYS	FLYBYT	reserved	SIZE		WIDTH		reserved	LEN												
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

Bits	Access	Name	Description
31	R/W	INCSRCADDR	Source Address Increment If the source address is an internal peripheral FIFO address or external I/O address, the address is not incremented on each successive access. In these cases, DCMDx[INCSRCADDR] must be cleared. 0 = Do not increment source address. 1 = Increment source address. **NOTE:** The only valid memory types for this mode are VLIO and Card devices. With DMA transfers from other memory types this bit field must be set.
30	R/W	INCTRGADDR	Target Address Increment If the target address is an internal peripheral FIFO address or external I/O address, the address is not incremented on each successive accesses. In these cases, DCMDx[INCTRGADDR] must be cleared. 0 = Do not increment target address. 1 = Increment target address. **NOTE:** The only valid memory types for this mode are VLIO and Card devices. With DMA transfers to other memory types this bit field must be set.
29	R/W	FLOWSRC	Source Flow Control Must be set if the source is an on-chip peripheral or external companion chip. Setting both the FLOWSRC and FLOWTRG bits causes unpredictable behavior. 0 = Do not wait for request signals associated with this channel. Start the data transfer if it is legal to do so. 1 = Wait for a request signal before initiating the data transfer.

Table 5-14. DCMD0–31 Bit Definitions (Sheet 2 of 4)

Physical Address 0x4000_02xC–0x4000_03xC | **DCMD0–DCMD31** | **DMA Controller**

	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
User Settings																																
Bit	INCSRCADDR	INCTRGADDR	FLOWSRC	FLOWTRG	reserved		CMPEN	reserved	ADDRMODE	STARTIRQEN	EndIrqEn	FLYBYS	FLYBYT	reserved	SIZE		WIDTH		reserved	LEN												
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

Bits	Access	Name	Description
28	R/W	FLOWTRG	Target Flow Control Must be set if the target is an on-chip peripheral or external companion chip. Setting both the FLOWSRC and FLOWTRG bits causes unpredictable behavior. 0 = Do not wait for request signals associated with this channel. Start the data transfer if it is legal to do so. 1 = Wait for a request signal before initiating the data transfer.
27:26	—	—	reserved
25	R/W	CMPEN	Descriptor Compare Enable Must be cleared for normal DMA operations. Setting the bit enables the descriptor-compare mode, in which the DMA controller treats the current descriptor as a special case and compares data that corresponds to the source and target fields. DCMDx[ADDRMODE] determines the addressing mode before the Compare operation. 0 = DMA does not perform any address-compare operations. 1 = DMA recognizes the current descriptor as a special case and compares data based on the source address and target address fields. If the compare is true, the channel's DCSRx[CMPST] bit is set. If the compare is false, DCSRx[CMPST] is cleared.
24	—	—	reserved
23	R/W	ADDRMODE or reserved	Addressing Mode Controls the addressing mode for descriptor comparison and is valid only in the descriptor compare mode (DCMDx[CMPEN] = 1). Reserved if DCMDx[CMPEN] = 0. If DCMDx[CMPEN] is set, the bits specify the addressing modes of the source address and target address fields. If either field contains an address, the DMA controller fetches the data at that address and uses it for the compare operation. 0 = Source address field contains address, and target address field contains address. 1 = Source address field contains address, and target address field contains data.

Table 5-14. DCMD0–31 Bit Definitions (Sheet 3 of 4)

Physical Address 0x4000_02xC–0x4000_03xC	DCMD0–DCMD31	DMA Controller

Bit	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
	INCSRCADDR	INCTRGADDR	FLOWSRC	FLOWTRG	reserved		CMPEN	reserved	ADDRMODE	STARTIRQEN	EndIrqEn	FLYBYS	FLYBYT	reserved	SIZE		WIDTH		reserved	LEN													
User Settings																																
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

Bits	Access	Name	Description
22	R/W	STARTIRQEN or reserved	Start Interrupt Enable Indicates that the interrupt is enabled as soon as the descriptor is loaded. In no-descriptor-fetch transfers, this bit is reserved. 0 = Interrupt not triggered after descriptor is loaded. 1 = Set interrupt bit for that channel in the DINT[CHLINTR] when the descriptor (4 words) for the channel is loaded.
21	R/W	EndIrqEn	End Interrupt Enable 0 = Interrupt is not triggered when LENGTH decrements to zero. 1 = Set the DINT interrupt bit for the channel when LENGTH decrements to zero.
20	R/W	FLYBYS	Fly-By Source Use for external companion-chip support only. If this bit is set, the source is a memory address and the target is the external companion chip. Do not set the FLYBYS and FLYBYT bits at the same time. 0 = Flow-through 1 = Fly-by
19	R/W	FLYBYT	Fly-By Target Use for external companion chip support only, If this bit is set, the target is a memory address, and the source is the external companion chip. Do not set the FLYBYS and FLYBYT bits at the same time. 0 = Flow-through 1 = Fly-by
18	—	—	reserved
17:16	R/W	SIZE	Maximum Burst Size of Each Data Transfer 0b00 = reserved 0b01 = 8 Bytes 0b10 = 16 Bytes 0b11 = 32 Bytes The size **must** be less than or equal to the serviced peripheral's FIFO trigger threshold to properly handle the respective FIFO's trailing bytes

Table 5-14. DCMD0–31 Bit Definitions (Sheet 4 of 4)

Physical Address 0x4000_02xC–0x4000_03xC | DCMD0–DCMD31 | DMA Controller

Bit	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
User Settings																																
	INCSRCADDR	INCTRGADDR	FLOWSRC	FLOWTRG	reserved		CMPEN	reserved	ADDRMODE	STARTIRQEN	EndIrqEn	FLYBYS	FLYBYT	reserved	SIZE		WIDTH		reserved	LEN												
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

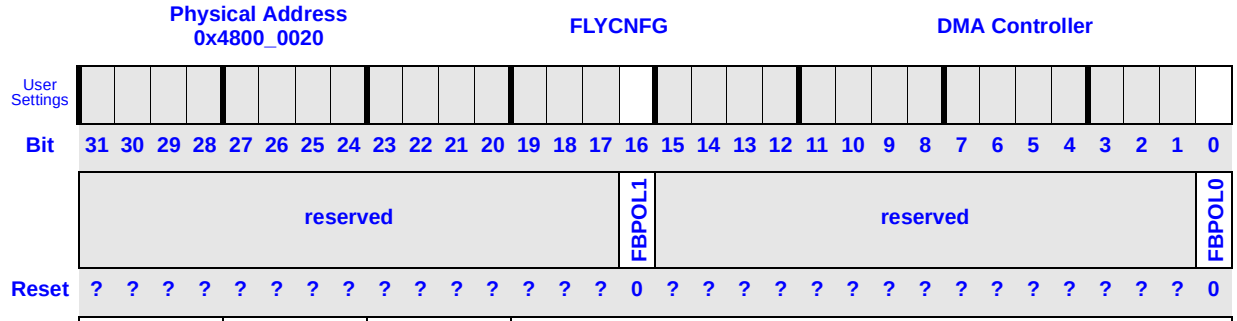
Bits	Access	Name	Description
15:14	R/W	WIDTH or reserved	Width of On-Chip Peripheral Reserved field for operations that do not involve on-chip peripherals, such as memory-to-memory moves and companion-chip-related operations. WIDTH must be 0b00 for memory-to-memory moves or companion-chip-related operations. 0b00 = reserved for on-chip peripheral-related transactions 0b01 = 1 byte 0b10 = Half-word (2 bytes) 0b11 = Word (4 Bytes)
13	—	—	reserved
12:0	R/W	LEN	Length of Transfer in Bytes LEN = 0 means zero bytes for descriptor-fetch transactions. LEN = 0 is an invalid setting for no-descriptor-fetch transactions. Programming LEN = 0 in the descriptor-fetch mode when DCMD[CmpEn] is clear (normal data transfer mode) causes the channel to immediately discard the descriptor after it is fetched from memory. If the descriptor chain has more descriptors, the channel fetches the next valid descriptor. The channel stops if the descriptor chain has no more descriptors. The maximum transfer length is (8K – 1) bytes. If the transfer is of the memory-to-memory type, the length of the transfer may be any value (except for the DCMDx[Len] = 0 restriction in no-descriptor-fetch mode) up to a maximum of (8K – 1) bytes. If the transfer involves an external peripheral (or a companion chip), then the length of the transfer must be an integer multiple of the peripheral's FIFO threshold (or water-mark). If the transfer involves any of the on-chip peripherals, the length of the transfer must be as follows: • For AC '97 and MMC/SDIO, LEN must be an integer multiple of 32 bytes. • For the quick capture interface, LEN must be an integer multiple of 8 bytes. • For all other on-chip peripherals, LEN must be an integer multiple of the peripheral's sample width (DCMDx[WIDTH]). **NOTE:** LEN is ignored in the compare descriptor mode (when DCMD[CmpEn] is set).

5.5.6 DMA Fly-By Configuration Register (FLYCNFG)

For user software, this register (Table 5-15) is read/write for polarity control of the DVAL<1:0> signal used during fly-by DMA transfers.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 5-15. FLYCNFG Bit Definitions



Bits	Access	Name	Description
31:17	—	—	reserved
16	R/W	FBPOL1	Fly-By DMA DVAL<1> Polarity 0 = Active low 1 = Active high
15:1	—	—	reserved
0	R/W	FBPOL0	Fly-By DMA DVAL<0> Polarity 0 = Active low 1 = Active high

5.5.7 DREQ<2:0> Status Register (DRQSR0/1/2)

DRQSR0/1/2 (Table 5-16) logs the number of pending requests made by an external companion chip on the corresponding DREQ<0>, DREQ<1> or DREQ<2> pin. The register reflects the status of a 5-bit counter that is controlled by the DMA controller in the following manner:

- The DMA controller increments the counter each time the external companion chip toggles the DREQ<2:0> pin from low to high (positive-edge trigger). The external companion chip must follow the rules outlined in Section 5.4.1.2.
- For a write to an external peripheral or companion chip, the DMA controller decreases the counter after it completes the write.
- For a read from an external peripheral or companion chip, the DMA controller decreases the counter after it sends the corresponding write request to the memory controller that completes the read-write transaction.

The external companion chip or external peripheral must not have more than 31 pending requests at a given time.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 5-16. DRQSR0/1/2 Bit Definitions

Physical Address			DMA Controller
0x4000_00E0	DRQSR0		
0x4000_00E4	DRQSR1		
0x4000_00E8	DRQSR2		

Bit	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
User Settings																																
	reserved																							CLR	reserved			REQPEND				
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	?	0	0	0	0	0

Bits	Access	Name	Description
31:9	—	—	reserved
8	W	CLR	Clear Pending Requests Writing 0b1 to this bit clears DRQSRx[REQPEND] and thereby clears all pending requests made by the external DMA request pin DREQx. Writing 0b0 to this bit has no effect. This bit could be used for clearing the requests if the channel mapped to DREQx was prematurely stopped by software. CLR must be set only after the mapped channel has stopped (DCSRx[STOPINTR] set). Clearing the requests of a running channel results in unpredictable behavior. 0 = No effect on DRQSRx[REQPEND]. 1 = Clear all pending requests registered in DRQSRx[REQPEND]
7:5	—	—	reserved
4:0	R	REQPEND	Requests Pending Indicates the number of pending requests on DREQx.

5.5.8 DMA Channel Control/Status Registers (DCSRx)

These read/write registers (Table 5-17) contains the control and status bits for the channels.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 5-17. DCSR0–31 Bit Definitions (Sheet 1 of 6)

Physical Address 0x4000_0000–0x4000_007C — **DCSR0–DCSR31** — **DMA Controller**

	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
User Settings																																
Bit	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
	RUN	NODESCFETCH	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	SETCMPST	CLRCMPST	RASIrqEn	MaskRun	reserved											CMPST	EORINTR	REQPEND	reserved			RASIntr	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
31	R/W	RUN	Run Allows software to start or stop the channel. If the run bit is cleared in the middle of the burst, the burst is completed before the channel stops. Setting RUN starts the stopped channel. If the channel is in a descriptor-fetch transfer and RUN is set before writing a valid descriptor address to register DDADRx, no-descriptor fetch occurs. This bit is reset as soon as it is cleared and when the channel stops normally. DCSRx[STOPINTR] must be polled to see the status of the channel or to set DCSRx[STOPIRQEN] and expect an interrupt after the channel stops. After the channel stops, DCSRx[STOPINTR]is set. STOPINTR must be polled to read the channel status or to set DCSRx[STOPIRQEN] and expect an interrupt after the channel stops. 0 = Stops the channel. 1 = Starts the channel.
30	R/W	NODESCFETCH	No-Descriptor-Fetch Controls whether or not a channel has a descriptor. If NODESCFETCH is set, the channel is considered a simple channel with no descriptors. In this case, the DMA does not initiate descriptor fetches when software sets DCSRx[RUN] or when the byte count for the current transfer reaches zero. To program the channel for a no-descriptor-fetch transfer, software must set DCSRx[NODESCFETCH], then write to the individual DSADRx, DTADRx, and DCMDx registers for that channel. The DDADRx register is not used in this mode and must not be written. DCSRx[RUN] must be set to allow the channel to start the transfer. If DCSRx[NODESCFETCH] is cleared, the DMA controller initiates descriptor-fetches when software writes to the DDADRx register, when the byte count for the current transfer reaches zero, and when DDADRx[STOP] is cleared. 0 = Descriptor-fetch transfer 1 = No-descriptor-fetch transfer

Table 5-17. DCSR0–31 Bit Definitions (Sheet 2 of 6)

Physical Address 0x4000_0000–0x4000_007C **DCSR0–DCSR31** **DMA Controller**

	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
User Settings																																
Bit	RUN	NODESCFETCH	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	SETCMPST	CLRCMPST	RASIrqEn	MaskRun	reserved											CMPST	EORINTR	REQPEND	reserved			RASIntr	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
29	R/W	STOPIRQEN	Stop Interrupt Enabled Enables the interrupt when DCSRx[STOPINTR] is set. If STOPINTEN is cleared, no interrupt is generated after the channel stops. DCSRx[STOPINTR] is set after system reset deassertion. Thus, if STOPINTR is set before the channel is started, an interrupt is generated. 0 = No interrupt if the channel is in uninitialized or stopped state. 1 = Interrupt enabled if the channel is in uninitialized or stopped state.
28	R/W	EORIRQEN	End-of-Receive Interrupt Enable Setting this bit triggers an interrupt on an EOR. Clearing it does not generate an EOR-related interrupt. 0 = Interrupt not triggered even if DCSRx[EORINT] is set. 1 = Enable interrupt if DCSRx[EORINT] is set.
27	R/W	EORJMPEN	Jump to Next Descriptor on EOR Controls the descriptor flow when the mapped peripheral signals an EOR to the DMA controller. Setting EORJMPEN causes the DMA controller to jump to the next descriptor on an EOR. Clearing EORJMPEN causes the DMA controller to wait for the peripheral's next receive request. See Figure 5-9 for the behavior of the descriptor during this condition. This control bit has no effect on the channel for no-descriptor-fetch transfers (DCSRx[NODESCFETCH] set). The DMA controller completes the peripheral-to-memory data transfer on an EOR, regardless of EORJMPEN. 0 = DMA controller continues to hold the current descriptor and simply waits until the mapped peripheral makes another receive request. 1 = DMA controller jumps to the channel's next descriptor on receiving an EOR from the mapped peripheral.
26	R/W	EORSTOPEN	Stop Channel on EOR Has no effect on the channel for no-descriptor-fetch transfers (DCSRx[NODESCFETCH] set). The DMA controller completes the peripheral-to-memory data transfer on an EOR, regardless of DCSRx[EORSTOPEN]. Setting EORSTOPEN causes the DMA controller to stop the channel on an EOR and set the corresponding DCSRx[STOPINTR] bit. If DCSRx[STOPIRQEN] is set when EORSTOPEN is set, an interrupt occurs. See Figure 5-9 for the behavior of the descriptor. 0 = DMA controller continues to hold the current descriptor and waits until the mapped peripheral makes another receive request. 1 = DMA controller stops the channel that received an EOR from the mapped peripheral.

Table 5-17. DCSR0–31 Bit Definitions (Sheet 3 of 6)

Physical Address 0x4000_0000–0x4000_007C | **DCSR0–DCSR31** | **DMA Controller**

Bit	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
User Settings																																
	RUN	NODESCFETCH	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	SETCMPST	CLRCMPST	RASIrqEn	MaskRun	reserved											CMPST	EORINTR	REQPEND	reserved			RASIntr	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
25	W	SETCMPST	Set Descriptor Compare Status Partially controls DCSRx[CMPST]. Setting this bit sets DCSRx[CMPST]. Clearing SETCMPST has no effect on DCSRx[CMPST]. Software can set DCSRx[CMPST] even if the descriptor is not configured in the compare mode (DCMDx[CMPEN] = 0). 0 = no effect on DCSRx[CMPST] 1 = set DCSRx[CMPST]
24	W	CLRCMPST	Clear Descriptor Compare Status Partially controls DCSRx[CMPST]. Setting CLRCMPST clears DCSRx[CMPST]. Clearing CLRCMPST has no effect on DCSRx[CMPST]. Software can set DCSRx[CMPST] even if the descriptor is not configured in the compare mode. 0 = no effect on DCSRx[CMPST] 1 = clear DCSRx[CMPST]
23	R/W	RASIrqEn	Request After Channel Stopped Interrupt Enable 0 = Interrupt not triggered when a peripheral asserts a DMA request after the channel has stopped. 1 = Set interrupt bit for that channel in the DINT[CHLINTR] when a peripheral asserts a DMA request after the channel has stopped.
22	W	MaskRun	Mask Run Mask DCSR[Run] during a programmed I/O write to the DCSR register. 0 = Software (programmed I/O write) can modify DCSR[RUN] during a write transaction in which DCSR[MaskRun] is 0. 1 = Software (programmed I/O write) cannot modify DCSR[RUN] during a write transaction in which DCSR[MaskRun] is 1.
21:11	—	—	reserved

Table 5-17. DCSR0–31 Bit Definitions (Sheet 4 of 6)

Physical Address 0x4000_0000–0x4000_007C | **DCSR0–DCSR31** | **DMA Controller**

	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
User Settings																																
Bit	RUN	NODESCFETCH	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	SETCMPST	CLRCMPST	RASIrqEn	MaskRun	reserved											CMPST	EORINTR	REQPEND	reserved			RASIntr	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
10	R/W	CMPST	Descriptor Compare Status Indicates the most recent status of the source and target compare operation. CMPST is set on a successful compare of the source and target fields. An unsuccessful comparison clears CMPST. Refer to the description of DCMDx[ADDRMODE] for the various addressing modes used for this comparison. For details regarding the descriptor compare mode, refer to DCMDx[CMPEN] in Table 5-14. The DMA controller updates CMPST only in descriptor compare mode (DCMDx[CMPEN] = 1). CMPST can be set and cleared by setting DCSRx[SETCMPST] and DCSRx[CLRCMPST], respectively. If software attempts to concurrently set and clear CMPST by setting both DCSRx[SETCMPST] and DCSRx[CLRCMPST], DCSRx[SETCMPST] has higher precedence. Modifying this bit after DCSRx[RUN] is set and the channel is actively running leads to faulty behavior of the descriptor chain. The channel must be stopped before setting or clearing CMPST. 0 = Indicates an unsuccessful address compare in descriptor compare mode. 1 = Indicates a successful compare of the current descriptor's source and target addresses in descriptor compare mode.
9	R/W	EORINT	End of Receive Indicates the status of the mapped peripheral's receive data. EORINT is set after the DMA controller reads out the last trailing sample from the peripheral's receive FIFO. Figure 5-9 illustrates the behavior of the descriptor during this condition. To clear EORINT, write 0b1 to the bit. EORINT pertains only to internal peripherals. 0 = DMA continues with current descriptor because the internal peripheral is still actively receiving data. 1 = Channel mapped internal peripheral has no data remaining in its receive FIFO and has completed all receive transactions. Refer to the description of DCSRx[EORJMPEN] for the behavior of the DMA controller during this condition. **NOTE:** The EORINT bit must be cleared before restarting a channel.

Table 5-17. DCSR0–31 Bit Definitions (Sheet 5 of 6)

Physical Address 0x4000_0000–0x4000_007C — DCSR0–DCSR31 — DMA Controller

Bit	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
	RUN	NODESCFETCH	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	SETCMPST	CLRCMPST	RASIrqEn	MaskRun	reserved											CMPST	EORINTR	REQPEND	reserved			RASIntr	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
User Settings																																
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
8	R	REQPEND	Request Pending Indicates a pending request for the DMA channel. REQPEND is cleared for a channel if that channel has no pending request or the request has just been issued to the memory interface in case of a read or write from the external companion chip to memory. If DREQ assertion sets REQPEND and DCSRx[RUN] is cleared to stop that channel, REQPEND and the internal registers that hold the DREQ assertion information, do not remain set. If the channel is restarted, REQPEND must be reset by a descriptor that transfers dummy data (for example, a memory-to-memory transfer from a temporary location to another temporary location). 0 = No request is pending for the channel. 1 = A request is pending for the channel.
7:5	—	—	reserved
4	R/W	RASIntr	Request After Channel Stopped 0 = No interrupt 1 = Interrupt caused due to a request made by the peripheral after the respective channel stopped.
3	R	STOPINTR	Stop Interrupt Reflects the channel's state. If the channel is uninitialized or stopped, STOPINTR is set. If DCSRx[STOPIRQEN] is set, the DMA controller generates an interrupt. Software must clear DCSRx[STOPIRQEN] to reset the interrupt. Reprogramming DDADRx and setting DCSRx[RUN] restarts the channel. 0 = The channel is running 1 = The channel is in uninitialized or stopped state.
2	R/W	ENDINTR	End Interrupt Indicates that the current descriptor finished successfully and that DCMDx[ENDIRQEN] is set. Set ENDINTR to reset the corresponding interrupt. Clearing ENDINTR has no effect. 0 = No interrupt 1 = Interrupt was caused due to successful completion of the current transaction and DCMDx[LEN] = 0.
1	R/W	STARTINTR	Start Interrupt Indicates that the current descriptor was loaded successfully and that DCMDx[STARTIRQEN] is set. Set STARTINTR to reset the corresponding interrupt. Clearing the bit has no effect. 0 = No interrupt 1 = Interrupt was caused due to successful descriptor fetch.

Table 5-17. DCSR0–31 Bit Definitions (Sheet 6 of 6)

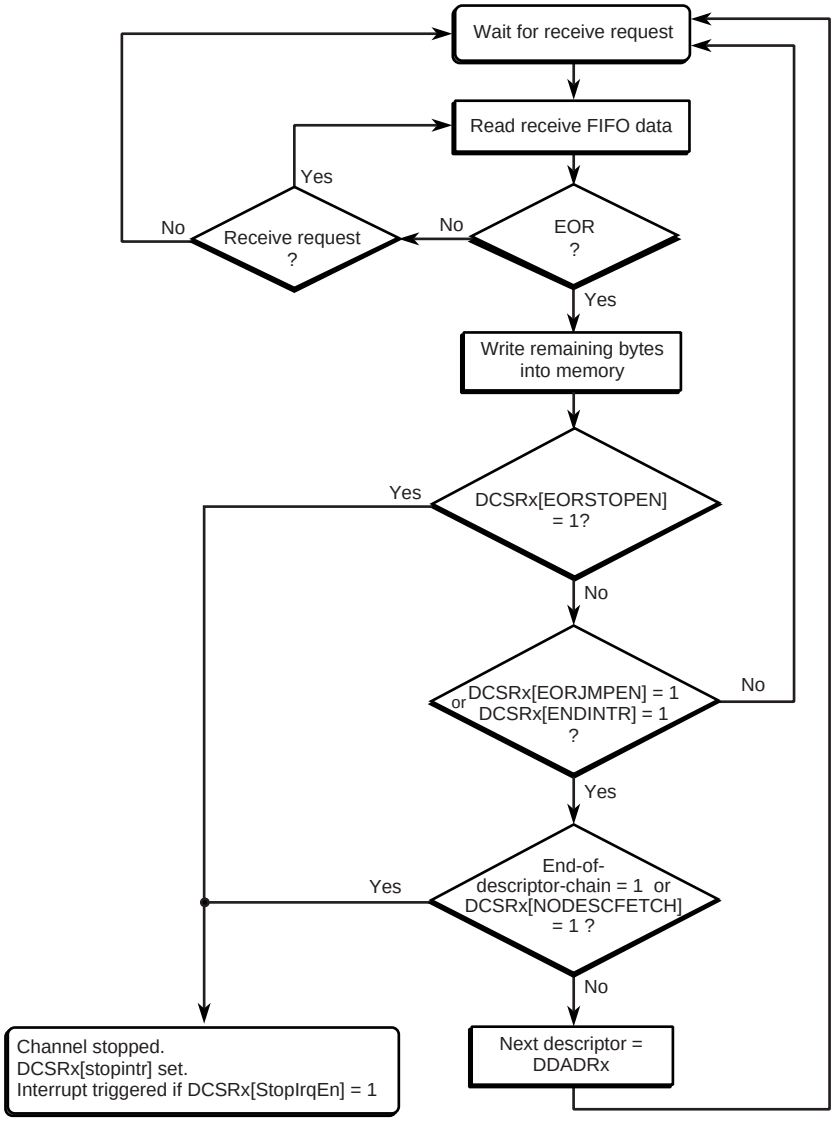
Physical Address 0x4000_0000–0x4000_007C — **DCSR0–DCSR31** — **DMA Controller**

Bit	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
User Settings																																
	RUN	NODESCFETCH	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	SETCMPST	CLRCMPST	RASIrqEn	MaskRun	reserved											CMPST	EORINTR	REQPEND	reserved			RASIntr	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
0	R/W	BUSERRINTR	Bus Error Interrupt Indicates that an error occurred during a data transfer on the internal bus. The error may be due to a bad descriptor source or target address (any address that is in the non-burstable or reserved space and can cause a bus error on the system bus). Set BUSERRINTR to reset the corresponding interrupt. Clearing this bit has no effect. Only one error per channel is logged. The channel that caused the error is not updated at the end of the transfer and is not accessible until it is reprogrammed and the corresponding run bit is set. 0 = No interrupt 1 = Interrupt was caused by bus error.

Figure 5-9 shows the descriptor behavior on end-of-receive (EOR).

Figure 5-9. Descriptor Behavior on End-of-Receive (EOR)



Note:
Fetching the NextDescriptor can be different if branching mode is enabled. Refer to the DDADRx[BREN] description for further details.

A9379-01

5.5.9 DMA Interrupt Register (DINT)

Read-only register DINT (Table 5-18) logs the interrupt information for each channel. An interrupt is generated if any of the following conditions occurs:

- Any transaction error occurs on the internal bus associated with the relevant channel.
- The current transfer finishes successfully and the DCMDx[ENDIRQEN] bit is set.
- The current descriptor is loaded successfully and the DCMDx[STARTIRQEN] bit is set.
- DCSRx[STOPERQEN] is set and the channel is in an uninitialized or stopped state.
- DCSRx[EORIRQEN] is set and DCSRx[EORINT] is set (EOR signaled by a peripheral).
- DCSRx[RASIrqEn] is set and the peripheral makes a DMA request after the channel has stopped.

All DMA controller interrupts, except the one that corresponds to DCSRx[STOPINTR], are cleared when software sets the respective error bit in DCSRx[STOPINTR].

Table 5-18. DINT Bit Definitions

Physical Address 0x4000_00F0								DINT																DMA Controller							
User Settings																															
Bit	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
	CHLINTR31	CHLINTR30	CHLINTR29	CHLINTR28	CHLINTR27	CHLINTR26	CHLINTR25	CHLINTR24	CHLINTR23	CHLINTR22	CHLINTR21	CHLINTR20	CHLINTR19	CHLINTR18	CHLINTR17	CHLINTR16	CHLINTR15	CHLINTR14	CHLINTR13	CHLINTR12	CHLINTR11	CHLINTR10	CHLINTR9	CHLINTR8	CHLINTR7	CHLINTR6	CHLINTR5	CHLINTR4	CHLINTR3	CHLINTR2	CHLINTR1	CHLINTR0
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

Bits	Access	Name	Description
31:0	R	CHLINTRx	Channel Interrupt Indicates that DMA channel *x* has been interrupted: 0 = No interrupt 1 = Interrupt

5.5.10 DMA Alignment Register (DALGN)

DALGN (Table 5-19) activates byte alignment for source and target addresses. Each bit in the register corresponds to a DMA channel. By default, during data transfers, the DMA controller forces the least-significant three bits for all external addresses to zeros and the least-significant two bits of all peripheral addresses to zeros.

Setting a channel-specific bit in DALGN causes the corresponding channel to access the complete user-specified address (none of the LSB bits of the address are forced to zeros). For example, if channel 31 is programmed to transfer data involving a misaligned address, software must write 0b1 to bit 31 of DALGN.

Clearing a bit position in the DALGN register causes the DMA controller to treat the corresponding channel as the default, a 64-bit aligned channel; the source and target addresses are forced to zeros, as explained earlier.

DALGN must be updated before setting DCSR[Run] and then must not be altered until the channel stops.

Table 5-19. DALGN Bit Definitions

Physical Address 0x4000_00A0 — **DALGN** — **DMA Controller**

	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
User Settings																																
Bit	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
	DALGN31	DALGN30	DALGN29	DALGN28	DALGN27	DALGN26	DALGN25	DALGN24	DALGN23	DALGN22	DALGN21	DALGN20	DALGN19	DALGN18	DALGN17	DALGN16	DALGN15	DALGN14	DALGN13	DALGN12	DALGN11	DALGN10	DALGN9	DALGN8	DALGN7	DALGN6	DALGN5	DALGN4	DALGN3	DALGN2	DALGN1	DALGN0
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

Bits	Access	Name	Description
31:0	R/W	DALGNx	Alignment Control for Channel X 0 = Source and target addresses of channel x are default aligned (internal peripherals default to 4 byte alignment; external bus addresses default to 8 byte alignment). 1 = Source and target addresses of channel x are as defined by user (byte-aligned).

5.5.11 DMA Programmed I/O Control Status Register (DPCSR)

DPCSR, defined in Table 5-20, is used for activating and monitoring posted writes and split reads on the system bus, when software uses programmed I/O (PIO) instructions to access the peripheral address domain via the DMA bridge.

Setting DPCSR[BrgSplit] activates the following DMA behavior:

- If the PIO transaction is a read from a peripheral-address domain, the DMA split responds to the read instruction. The DMA bridge releases the system bus, then uses microcoded instructions to read data from the peripheral bus. Once the read completes across the peripheral bus, the DMA controller completes the split transaction by recapturing the system bus and completing the PIO read transaction. The core is stalled until the read is returned, because this is programmed I/O. Any PIO transactions (reads or writes) that occur while the current PIO read transaction is between the split response and the split completion are retried.
- If the PIO transaction is a write instruction to a peripheral-address domain, the DMA posts the write instruction. The DMA bridge indicates to the system bus that the PIO write is complete and then releases the system bus. The actual write transaction is then sent across the peripheral bus using microcoded instructions. If software requires that a write complete on the peripheral bus before continuing, then software must write the address, then immediately read the same address, which guarantees that the address has been updated before allowing the core to continue execution. Any PIO instructions (reads or writes) between the time the write gets posted on the system bus and until the time the actual write completes across the peripheral bus are retried.

Clearing DPCSR[BrgSplit] de-activates the posted write and split response behavior. A write transaction on the system bus is completed only after the write is sent across the peripheral bus. The targeted address location is guaranteed to be updated by the time the transaction completes on the system bus. A read transaction on the system bus is completed only after the DMA bridge receives the data from across the peripheral bus. There are no split responses, split completions, or retries in this mode.

Note: (1) If software requires that a write complete on the peripheral bus before continuing, then software must write the address, then immediately read the same address, which guarantees that the address has been updated before allowing the core to continue execution. Users must perform this read-after-write transaction to make sure the processor is in a correct state before the core continues execution.

(2) This control bit must be modified only when DPCSR[BrgBusy] is clear (no pending peripheral PIO transactions). Modifying this control bit when a PIO transaction is still pending might lead to unpredictable results and is therefore not recommended.

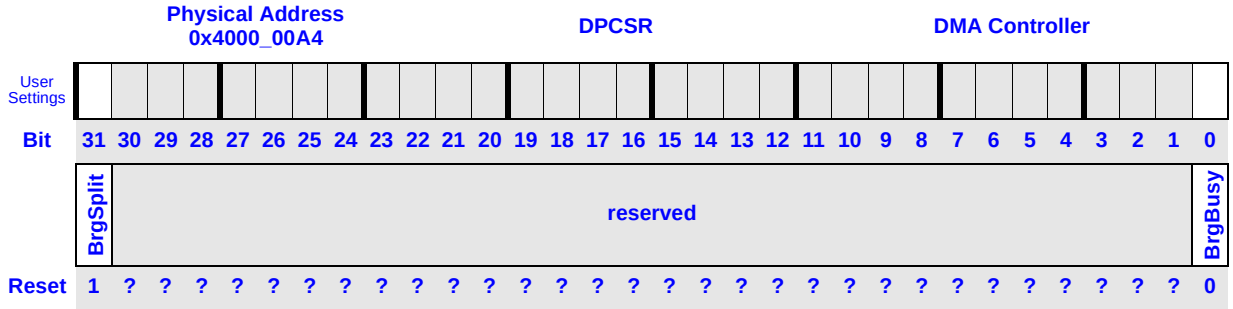
(3) The PIO transactions are always completed in the order they were issued, regardless of DPCSR[BrgSplit].

(4) DPCSR[BrgSplit] is set by default (reset value).

DPCSR[BrgBusy] is a status bit which, when set, indicates a pending PIO transaction across the peripheral bus. Any further PIO transactions on the system bus are retried when DPCSR[BrgBusy] is set. This bit cannot be modified by software. When DPCSR[BrgBusy] is clear, there are no pending PIO transactions across the peripheral bus. A new PIO transaction is not retried in this case.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 5-20. DPCSR Bit Definitions



Bits	Access	Name	Description
31	R/W	BrgSplit	Activate Posted Writes and Split Reads 0 = Deactivate posted writes, split responses, split completions and retries. 1 = Activate posted writes, split responses, split completions and retries.
30:1	—	—	reserved
0	R	BrgBusy	Bridge Busy Status 0 = No pending PIO transactions across peripheral bus. 1 = Pending PIO transaction across peripheral bus.

5.6 Register Summary

Table 5-21 summarizes the DMA controller registers.

Table 5-21. DMA Controller Register Summary (Sheet 1 of 8)

Address	Name	Description	Page
0x4000_0000	DCSR0	DMA Control/Status register for Channel 0	5-41
0x4000_0004	DCSR1	DMA Control/Status register for Channel 1	5-41
0x4000_0008	DCSR2	DMA Control/Status register for Channel 2	5-41
0x4000_000C	DCSR3	DMA Control/Status register for Channel 3	5-41
0x4000_0010	DCSR4	DMA Control/Status register for Channel 4	5-41
0x4000_0014	DCSR5	DMA Control/Status register for Channel 5	5-41
0x4000_0018	DCSR6	DMA Control/Status register for Channel 6	5-41
0x4000_001C	DCSR7	DMA Control/Status register for Channel 7	5-41
0x4000_0020	DCSR8	DMA Control/Status register for Channel 8	5-41
0x4000_0024	DCSR9	DMA Control/Status register for Channel 9	5-41
0x4000_0028	DCSR10	DMA Control/Status register for Channel 10	5-41
0x4000_002C	DCSR11	DMA Control/Status register for Channel 11	5-41
0x4000_0030	DCSR12	DMA Control/Status register for Channel 12	5-41
0x4000_0034	DCSR13	DMA Control/Status register for Channel 13	5-41
0x4000_0038	DCSR14	DMA Control/Status register for Channel 14	5-41
0x4000_003C	DCSR15	DMA Control/Status register for Channel 15	5-41

Table 5-21. DMA Controller Register Summary (Sheet 2 of 8)

Address	Name	Description	Page
0x4000_0040	DCSR16	DMA Control/Status register for Channel 16	5-41
0x4000_0044	DCSR17	DMA Control/Status register for Channel 17	5-41
0x4000_0048	DCSR18	DMA Control/Status register for Channel 18	5-41
0x4000_004C	DCSR19	DMA Control/Status register for Channel 19	5-41
0x4000_0050	DCSR20	DMA Control/Status register for Channel 20	5-41
0x4000_0054	DCSR21	DMA Control/Status register for Channel 21	5-41
0x4000_0058	DCSR22	DMA Control/Status register for Channel 22	5-41
0x4000_005C	DCSR23	DMA Control/Status register for Channel 23	5-41
0x4000_0060	DCSR24	DMA Control/Status register for Channel 24	5-41
0x4000_0064	DCSR25	DMA Control/Status register for Channel 25	5-41
0x4000_0068	DCSR26	DMA Control/Status register for Channel 26	5-41
0x4000_006C	DCSR27	DMA Control/Status register for Channel 27	5-41
0x4000_0070	DCSR28	DMA Control/Status register for Channel 28	5-41
0x4000_0074	DCSR29	DMA Control/Status register for Channel 29	5-41
0x4000_0078	DCSR30	DMA Control/Status register for Channel 30	5-41
0x4000_007C	DCSR31	DMA Control/Status register for Channel 31	5-41
0x4000_0080–0x4000_009C	—	reserved	—
0x4000_00A0	DALGN	DMA Alignment register	5-49
0x4000_00A4	DPCSR	DMA Programmed I/O Control Status register	5-51
0x4000_00A8–0x4000_00DC	—	reserved	—
0x4000_00E0	DRQSR0	DMA DREQ<0> Status register	5-40
0x4000_00E4	DRQSR1	DMA DREQ<1> Status register	5-40
0x4000_00E8	DRQSR2	DMA DREQ<2> Status register	5-40
0x4000_00EC	—	reserved	—
0x4000_00F0	DINT	DMA Interrupt register	5-48
0x4000_00F4–0x4000_00FC	—	reserved	—
0x4000_0100	DRCMR0	Request to Channel Map register for DREQ<0> (companion chip request 0)	5-31
0x4000_0104	DRCMR1	Request to Channel Map register for DREQ<1> (companion chip request 1)	5-31
0x4000_0108	DRCMR2	Request to Channel Map register for I^2S receive request	5-31
0x4000_010C	DRCMR3	Request to Channel Map register for I^2S transmit request	5-31
0x4000_0110	DRCMR4	Request to Channel Map register for BTUART receive request	5-31
0x4000_0114	DRCMR5	Request to Channel Map register for BTUART transmit request.	5-31
0x4000_0118	DRCMR6	Request to Channel Map register for FFUART receive request	5-31
0x4000_011C	DRCMR7	Request to Channel Map register for FFUART transmit request	5-31
0x4000_0120	DRCMR8	Request to Channel Map register for AC ’97 microphone request	5-31

Table 5-21. DMA Controller Register Summary (Sheet 3 of 8)

Address	Name	Description	Page
0x4000_0124	DRCMR9	Request to Channel Map register for AC '97 modem receive request	5-31
0x4000_0128	DRCMR10	Request to Channel Map register for AC '97 modem transmit request	5-31
0x4000_012C	DRCMR11	Request to Channel Map register for AC '97 audio receive request	5-31
0x4000_0130	DRCMR12	Request to Channel Map register for AC '97 audio transmit request	5-31
0x4000_0134	DRCMR13	Request to Channel Map register for SSP1 receive request	5-31
0x4000_0138	DRCMR14	Request to Channel Map register for SSP1 transmit request	5-31
0x4000_013C	DRCMR15	Request to Channel Map register for SSP2 receive request	5-31
0x4000_0140	DRCMR16	Request to Channel Map register for SSP2 transmit request	5-31
0x4000_0144	DRCMR17	Request to Channel Map register for ICP receive request	5-31
0x4000_0148	DRCMR18	Request to Channel Map register for ICP transmit request	5-31
0x4000_014C	DRCMR19	Request to Channel Map register for STUART receive request	5-31
0x4000_0150	DRCMR20	Request to Channel Map register for STUART transmit request	5-31
0x4000_0154	DRCMR21	Request to Channel Map register for MMC/SDIO receive request	5-31
0x4000_0158	DRCMR22	Request to Channel Map register for MMC/SDIO transmit request	5-31
0x4000_015C	—	reserved	—
0x4000_0160	DRCMR24	Request to Channel Map register for USB endpoint 0 request	5-31
0x4000_0164	DRCMR25	Request to Channel Map register for USB endpoint A request	5-31
0x4000_0168	DRCMR26	Request to Channel Map register for USB endpoint B request	5-31
0x4000_016C	DRCMR27	Request to Channel Map register for USB endpoint C request	5-31
0x4000_0170	DRCMR28	Request to Channel Map register for USB endpoint D request	5-31
0x4000_0174	DRCMR29	Request to Channel Map register for USB endpoint E request	5-31
0x4000_0178	DRCMR30	Request to Channel Map register for USB endpoint F request	5-31
0x4000_017C	DRCMR31	Request to Channel Map register for USB endpoint G request	5-31
0x4000_0180	DRCMR32	Request to Channel Map register for USB endpoint H request	5-31
0x4000_0184	DRCMR33	Request to Channel Map register for USB endpoint I request	5-31
0x4000_0188	DRCMR34	Request to Channel Map register for USB endpoint J request	5-31
0x4000_018C	DRCMR35	Request to Channel Map register for USB endpoint K request	5-31
0x4000_0190	DRCMR36	Request to Channel Map register for USB endpoint L request	5-31
0x4000_0194	DRCMR37	Request to Channel Map register for USB endpoint M request	5-31
0x4000_0198	DRCMR38	Request to Channel Map register for USB endpoint N request	5-31
0x4000_019C	DRCMR39	Request to Channel Map register for USB endpoint P request	5-31
0x4000_01A0	DRCMR40	Request to Channel Map register for USB endpoint Q request	5-31
0x4000_01A4	DRCMR41	Request to Channel Map register for USB endpoint R request	5-31
0x4000_01A8	DRCMR42	Request to Channel Map register for USB endpoint S request	5-31
0x4000_01AC	DRCMR43	Request to Channel Map register for USB endpoint T request	5-31
0x4000_01B0	DRCMR44	Request to Channel Map register for USB endpoint U request	5-31
0x4000_01B4	DRCMR45	Request to Channel Map register for USB endpoint V request	5-31

Table 5-21. DMA Controller Register Summary (Sheet 4 of 8)

Address	Name	Description	Page
0x4000_01B8	DRCMR46	Request to Channel Map register for USB endpoint W request	5-31
0x4000_01BC	DRCMR47	Request to Channel Map register for USB endpoint X request	5-31
0x4000_01C0	DRCMR48	Request to Channel Map register for MSL receive request 1	5-31
0x4000_01C4	DRCMR49	Request to Channel Map register for MSL transmit request 1	5-31
0x4000_01C8	DRCMR50	Request to Channel Map register for MSL receive request 2	5-31
0x4000_01CC	DRCMR51	Request to Channel Map register for MSL transmit request 2	5-31
0x4000_01D0	DRCMR52	Request to Channel Map register for MSL receive request 3	5-31
0x4000_01D4	DRCMR53	Request to Channel Map register for MSL transmit request 3	5-31
0x4000_01D8	DRCMR54	Request to Channel Map register for MSL receive request 4	5-31
0x4000_01DC	DRCMR55	Request to Channel Map register for MSL transmit request 4	5-31
0x4000_01E0	DRCMR56	Request to Channel Map register for MSL receive request 5	5-31
0x4000_01E4	DRCMR57	Request to Channel Map register for MSL transmit request 5	5-31
0x4000_01E8	DRCMR58	Request to Channel Map register for MSL receive request 6	5-31
0x4000_01EC	DRCMR59	Request to Channel Map register for MSL transmit request 6	5-31
0x4000_01F0	DRCMR60	Request to Channel Map register for MSL receive request 7	5-31
0x4000_01F4	DRCMR61	Request to Channel Map register for MSL transmit request 7	5-31
0x4000_01F8	DRCMR62	Request to Channel Map register for USIM receive request	5-31
0x4000_01FC	DRCMR63	Request to Channel Map register for USIM transmit request	5-31
0x4000_0200	DDADR0	DMA Descriptor Address register for Channel 0	5-32
0x4000_0204	DSADR0	DMA Source Address register for Channel 0	5-33
0x4000_0208	DTADR0	DMA Target Address register for Channel 0	5-34
0x4000_020C	DCMD0	DMA Command Address register for Channel 0	5-35
0x4000_0210	DDADR1	DMA Descriptor Address register for Channel 1	5-32
0x4000_0214	DSADR1	DMA Source Address register for Channel 1	5-33
0x4000_0218	DTADR1	DMA Target Address register for Channel 1	5-34
0x4000_021C	DCMD1	DMA Command Address register for Channel 1	5-35
0x4000_0220	DDADR2	DMA Descriptor Address register for Channel 2	5-32
0x4000_0224	DSADR2	DMA Source Address register for Channel 2	5-33
0x4000_0228	DTADR2	DMA Target Address register for Channel 2	5-34
0x4000_022C	DCMD2	DMA Command Address register for Channel 2	5-35
0x4000_0230	DDADR3	DMA Descriptor Address register for Channel 3	5-32
0x4000_0234	DSADR3	DMA Source Address register for Channel 3	5-33
0x4000_0238	DTADR3	DMA Target Address register for Channel 3	5-34
0x4000_023C	DCMD3	DMA Command Address register for Channel 3	5-35
0x4000_0240	DDADR4	DMA Descriptor Address register for Channel 4	5-32
0x4000_0244	DSADR4	DMA Source Address register for Channel 4	5-33
0x4000_0248	DTADR4	DMA Target Address register for Channel 4	5-34
0x4000_024C	DCMD4	DMA Command Address register for Channel 4	5-35

Table 5-21. DMA Controller Register Summary (Sheet 5 of 8)

Address	Name	Description	Page
0x4000_0250	DDADR5	DMA Descriptor Address register for Channel 5	5-32
0x4000_0254	DSADR5	DMA Source Address register for Channel 5	5-33
0x4000_0258	DTADR5	DMA Target Address register for Channel 5	5-34
0x4000_025C	DCMD5	DMA Command Address register for Channel 5	5-35
0x4000_0260	DDADR6	DMA Descriptor Address register for Channel 6	5-32
0x4000_0264	DSADR6	DMA Source Address register for Channel 6	5-33
0x4000_0268	DTADR6	DMA Target Address register for Channel 6	5-34
0x4000_026C	DCMD6	DMA Command Address register for Channel 6	5-35
0x4000_0270	DDADR7	DMA Descriptor Address register for Channel 7	5-32
0x4000_0274	DSADR7	DMA Source Address register for Channel 7	5-33
0x4000_0278	DTADR7	DMA Target Address register for Channel 7	5-34
0x4000_027C	DCMD7	DMA Command Address register for Channel 7	5-35
0x4000_0280	DDADR8	DMA Descriptor Address register for Channel 8	5-32
0x4000_0284	DSADR8	DMA Source Address register for Channel 8	5-33
0x4000_0288	DTADR8	DMA Target Address register for Channel 8	5-34
0x4000_028C	DCMD8	DMA Command Address register for Channel 8	5-35
0x4000_0290	DDADR9	DMA Descriptor Address register for Channel 9	5-32
0x4000_0294	DSADR9	DMA Source Address register for Channel 9	5-33
0x4000_0298	DTADR9	DMA Target Address register for Channel 9	5-34
0x4000_029C	DCMD9	DMA Command Address register for Channel 9	5-35
0x4000_02A0	DDADR10	DMA Descriptor Address register for Channel 10	5-32
0x4000_02A4	DSADR10	DMA Source Address register for Channel 10	5-33
0x4000_02A8	DTADR10	DMA Target Address register for Channel 10	5-34
0x4000_02AC	DCMD10	DMA Command Address register for Channel 10	5-35
0x4000_02B0	DDADR11	DMA Descriptor Address register for Channel 11	5-32
0x4000_02B4	DSADR11	DMA Source Address register for Channel 11	5-33
0x4000_02B8	DTADR11	DMA Target Address register for Channel 11	5-34
0x4000_02BC	DCMD11	DMA Command Address register for Channel 11	5-35
0x4000_02C0	DDADR12	DMA Descriptor Address register for Channel 12	5-32
0x4000_02C4	DSADR12	DMA Source Address register for Channel 12	5-33
0x4000_02C8	DTADR12	DMA Target Address register for Channel 12	5-34
0x4000_02CC	DCMD12	DMA Command Address register for Channel 12	5-35
0x4000_02D0	DDADR13	DMA Descriptor Address register for Channel 13	5-32
0x4000_02D4	DSADR13	DMA Source Address register for Channel 13	5-33
0x4000_02D8	DTADR13	DMA Target Address register for Channel 13	5-34
0x4000_02DC	DCMD13	DMA Command Address register for Channel 13	5-35
0x4000_02E0	DDADR14	DMA Descriptor Address register for Channel 14	5-32
0x4000_02E4	DSADR14	DMA Source Address register for Channel 14	5-33

Table 5-21. DMA Controller Register Summary (Sheet 6 of 8)

Address	Name	Description	Page
0x4000_02E8	DTADR14	DMA Target Address register for Channel 14	5-34
0x4000_02EC	DCMD14	DMA Command Address register for Channel 14	5-35
0x4000_02F0	DDADR15	DMA Descriptor Address register for Channel 15	5-32
0x4000_02F4	DSADR15	DMA Source Address register for Channel 15	5-33
0x4000_02F8	DTADR15	DMA Target Address register for Channel 15	5-34
0x4000_02FC	DCMD15	DMA Command Address register for Channel 15	5-35
0x4000_0300	DDADR16	DMA Descriptor Address register for Channel 16	5-32
0x4000_0304	DSADR16	DMA Source Address register for Channel 16	5-33
0x4000_0308	DTADR16	DMA Target Address register for Channel 16	5-34
0x4000_030C	DCMD16	DMA Command Address register for Channel 16	5-35
0x4000_0310	DDADR17	DMA Descriptor Address register for Channel 17	5-32
0x4000_0314	DSADR17	DMA Source Address register for Channel 17	5-33
0x4000_0318	DTADR17	DMA Target Address register for Channel 17	5-34
0x4000_031C	DCMD17	DMA Command Address register for Channel 17	5-35
0x4000_0320	DDADR18	DMA Descriptor Address register for Channel 18	5-32
0x4000_0324	DSADR18	DMA Source Address register for Channel 18	5-33
0x4000_0328	DTADR18	DMA Target Address register for Channel 18	5-34
0x4000_032C	DCMD18	DMA Command Address register for Channel 18	5-35
0x4000_0330	DDADR19	DMA Descriptor Address register for Channel 19	5-32
0x4000_0334	DSADR19	DMA Source Address register for Channel 19	5-33
0x4000_0338	DTADR19	DMA Target Address register for Channel 19	5-34
0x4000_033C	DCMD19	DMA Command Address register for Channel 19	5-35
0x4000_0340	DDADR20	DMA Descriptor Address register for Channel 20	5-32
0x4000_0344	DSADR20	DMA Source Address register for Channel 20	5-33
0x4000_0348	DTADR20	DMA Target Address register for Channel 20	5-34
0x4000_034C	DCMD20	DMA Command Address register for Channel 20	5-35
0x4000_0350	DDADR21	DMA Descriptor Address register for Channel 21	5-32
0x4000_0354	DSADR21	DMA Source Address register for Channel 21	5-33
0x4000_0358	DTADR21	DMA Target Address register for Channel 21	5-34
0x4000_035C	DCMD21	DMA Command Address register for Channel 21	5-35
0x4000_0360	DDADR22	DMA Descriptor Address register for Channel 22	5-32
0x4000_0364	DSADR22	DMA Source Address register for Channel 22	5-33
0x4000_0368	DTADR22	DMA Target Address register for Channel 22	5-34
0x4000_036C	DCMD22	DMA Command Address register for Channel 22	5-35
0x4000_0370	DDADR23	DMA Descriptor Address register for Channel 23	5-32
0x4000_0374	DSADR23	DMA Source Address register for Channel 23	5-33
0x4000_0378	DTADR23	DMA Target Address register for Channel 23	5-34
0x4000_037C	DCMD23	DMA Command Address register for Channel 23	5-35

Table 5-21. DMA Controller Register Summary (Sheet 7 of 8)

Address	Name	Description	Page
0x4000_0380	DDADR24	DMA Descriptor Address register for Channel 24	5-32
0x4000_0384	DSADR24	DMA Source Address register for Channel 24	5-33
0x4000_0388	DTADR24	DMA Target Address register for Channel 24	5-34
0x4000_038C	DCMD24	DMA Command Address register for Channel 24	5-35
0x4000_0390	DDADR25	DMA Descriptor Address register for Channel 25	5-32
0x4000_0394	DSADR25	DMA Source Address register for Channel 25	5-33
0x4000_0398	DTADR25	DMA Target Address register for Channel 25	5-34
0x4000_039C	DCMD25	DMA Command Address register for Channel 25	5-35
0x4000_03A0	DDADR26	DMA Descriptor Address register for Channel 26	5-32
0x4000_03A4	DSADR26	DMA Source Address register for Channel 26	5-33
0x4000_03A8	DTADR26	DMA Target Address register for Channel 26	5-34
0x4000_03AC	DCMD26	DMA Command Address register for Channel 26	5-35
0x4000_03B0	DDADR27	DMA Descriptor Address register for Channel 27	5-32
0x4000_03B4	DSADR27	DMA Source Address register for Channel 27	5-33
0x4000_03B8	DTADR27	DMA Target Address register for Channel 27	5-34
0x4000_03BC	DCMD27	DMA Command Address register for Channel 27	5-35
0x4000_03C0	DDADR28	DMA Descriptor Address register for Channel 28	5-32
0x4000_03C4	DSADR28	DMA Source Address register for Channel 28	5-33
0x4000_03C8	DTADR28	DMA Target Address register for Channel 28	5-34
0x4000_03CC	DCMD28	DMA Command Address register for Channel 28	5-35
0x4000_03D0	DDADR29	DMA Descriptor Address register for Channel 29	5-32
0x4000_03D4	DSADR29	DMA Source Address register for Channel 29	5-33
0x4000_03D8	DTADR29	DMA Target Address register for Channel 29	5-34
0x4000_03DC	DCMD29	DMA Command Address register for Channel 29	5-35
0x4000_03E0	DDADR30	DMA Descriptor Address register for Channel 30	5-32
0x4000_03E4	DSADR30	DMA Source Address register for Channel 30	5-33
0x4000_03E8	DTADR30	DMA Target Address register for Channel 30	5-34
0x4000_03EC	DCMD30	DMA Command Address register for Channel 30	5-35
0x4000_03F0	DDADR31	DMA Descriptor Address register for Channel 31	5-32
0x4000_03F4	DSADR31	DMA Source Address register for Channel 31	5-33
0x4000_03F8	DTADR31	DMA Target Address register for Channel 31	5-34
0x4000_03FC	DCMD31	DMA Command Address register for Channel 31	5-35
0x4000_0400–0x4000_10FC	—	reserved	—
0x4000_1100	DRCMR64	Request to Channel Map register for Memory Stick receive request	5-31
0x4000_1104	DRCMR65	Request to Channel Map register for Memory Stick transmit request	5-31
0x4000_1108	DRCMR66	Request to Channel Map register for SSP3 receive request	5-31
0x4000_110C	DRCMR67	Request to Channel Map register for SSP3 transmit request	5-31

Table 5-21. DMA Controller Register Summary (Sheet 8 of 8)

Address	Name	Description	Page
0x4000_1110	DRCMR68	Request to Channel Map register for Quick Capture Interface Receive Request 0	5-31
0x4000_1114	DRCMR69	Request to Channel Map register for Quick Capture Interface Receive Request 1	5-31
0x4000_1118	DRCMR70	Request to Channel Map register for Quick Capture Interface Receive Request 2	5-31
0x4000_111C	DRCMR71	Request to Channel Map register for TPM Receive Request	5-31
0x4000_1120	DRCMR72	Request to Channel Map register for TPM Transmit Request 1	5-31
0x4000_1124	DRCMR73	Request to Channel Map register for TPM Transmit Request 2	5-31
0x4000_1128	DRCMR74	Request to Channel Map register for DREQ<2> (companion chip request 2)	5-31
0x4000_112C–0x400F_FFFC	—	reserved	—
0x4800_0020	FLYCNFG	Fly-by DMA DVAL<1:0> polarities	5-39

Memory Controller 6

This chapter describes the internal and external memory-interface structures for the PXA27x processor. Memory-related registers that configure the memory controller for data transfers to and from static and dynamic memory devices are also described.

6.1 Overview

The external memory-bus interface for the PXA27x processor supports SDRAM, synchronous, and asynchronous burst-mode and page-mode flash memory, page-mode ROM, SRAM, variable-latency I/O (VLIO) memory, PC Card, and CompactFlash expansion memory. Memory types are programmable through the memory-interface configuration registers (see Table 6-44).

Memory requests are placed in a four-deep processing queue and processed in the order they are received.

6.2 Features

The memory controller provides the following features:

- Interfaces to internal synchronous flash and SDRAM devices
- Interfaces to four partitions of SDRAM
- Interfaces to up to 1.0 Gbytes of SDRAM
- Supports 1.8-V JEDEC LP-SDRAM operation at 104 MHz
- Interfaces to six partitions of static memory. Four of these six partitions can be synchronous flash memory.
- Interfaces to up to 384 Mbytes of flash memory
- Interfaces to two sockets of PC Card memory
- Allows an alternate bus master to take control of the bus
- Places the SDRAMs into self-refresh mode before entering sleep, standby, deep-sleep, and frequency-change modes
- Provides signals and controls for fly-by DMA transfers
- Supports non-volatile memory configured as bank 0 from either 16- or 32-bit devices
- Provides three independent output clocks (SDCLK<2:0>) that can be turned on/off separately and can be programmed to be free-running. The clocks can be the same frequency or half the frequency of the input clock, CLK_MEM. One clock (SDCLK<0>) can also be programmed as one quarter of the input-clock frequency. A fourth output clock (SDCLK<3>) depends on configuration bits used to control SDCLK<0>.
- Provides a programmable power-down mode for saving power

6.3 Signal Descriptions

The signals shown in Table 6-1 are inputs or outputs from the external memory controller block. In general, do not change a signal direction unless the change is required. See the *Intel® PXA270 Processor Electrical, Mechanical, and Thermal Specification* and *Intel® PXA27x Processor Family Electrical, Mechanical, and Thermal Specification* (*Intel® PXA27x Processor Family EMTS*) for signal timing.

Table 6-1. Memory Controller I/O Signal Descriptions (Sheet 1 of 2)

Signal Name	Type	Description
Shared Memory Controller I/O Signals		
MD<31:0>	Input/Output	Bidirectional data for all memory types.
MA<25:0>	Output	Output address to all memory types.
DQM<3:0>	Output	Data byte mask control. DQM<0> corresponds to MD<7:0> DQM<1> corresponds to MD<15:8> DQM<2> corresponds to MD<23:16> DQM<3> corresponds to MD<31:24> 0 = Do not mask out corresponding byte 1 = Mask out corresponding byte
SDRAM and Static Memory I/O Signals		
SDCLK<3:0>	Output	Output clocks to external memory. SDCLK<0> is for all static memory partitions. SDCLK<1> is for SDRAM partitions 0 and 1. SDCLK<2> is for SDRAM partitions 2 and 3. SDCLK<3> is dedicated to synchronous flash within the PXA271and PXA272 processors. It is not intended to drive synchronous flash external to these processors.
SDCKE	Output	Output-clock enable signals for external memory. SDCKE is for all SDRAM memory partitions.
nSDRAS	Output	Row address strobe for SDRAM.
nSDCAS	Output	Column address strobe for SDRAM. Also nADV (address strobe) for synchronous flash memory.
nSDCS<3:0>	Output	Chips selects for SDRAM.
nCS<5:0>	Output	Chip selects for static memory.
nWE	Output	Write enable for SDRAM and static memory.
nOE	Output	Output enable for static memory.
Miscellaneous I/O Signals		
RDnWR	Output	During active operation, this is the data direction signal to be used by output transceivers. During inactive operation, RDnWR is driven low. 0 = MD<31:0> is being driven by the processor 1 = MD<31:0> is not being driven by the processor
DVAL<1:0>	Output	Data ready signals for fly-by DMA mode

Table 6-1. Memory Controller I/O Signal Descriptions (Sheet 2 of 2)

Signal Name	Type	Description
RDY	Input	VLIO signal for inserting wait states. 0 = Wait 1 = VLIO is ready
BOOT_SEL	Input	Boot select signal that configures the memory controller for the bus width of the boot memory: 0 = 32-bit ROM/flash memory 1 = 16-bit ROM/flash memory
Alternate Bus Master Mode I/O Signals		
MBREQ	Input	Alternate bus master request†
MBGNT	Output	Alternate bus master grant
PC Card Interface I/O Signals		
nPCE<2:1>	Output	Byte lane enables for the PC Card interface. nPCE<1> enables MD<7:0>; nPCE<2> enables MD<15:8>.
nPREG	Output	Serves as the PC Card interface address bit MA<26> and selects register space (I/O or attribute) or memory space.
nPIOR	Output	PC Card interface I/O space output enable
nPIOW	Output	PC Card interface I/O space write enable
nPWE	Output	PC Card interface attribute and common memory space write enable. Also write enable for variable-latency I/O memory.
nPOE	Output	PC Card interface attribute and common memory space output enable.
nIOIS16	Input	PC Card interface input from I/O space indicating the size of the data bus: 0 = 16-bit I/O space 1 = 8-bit I/O space
nPWAIT	Input	PC Card interface input for inserting wait states: 0 = Wait 1 = PC Card is ready
PSKTSEL	Output	This is the active low output enable that can be used as nOE for the data transceivers. In a single socket solution: 0 = Output enable selected 1 = Output enable not selected In a dual socket solution, the socket select: 0 = Socket 0 selected 1 = Socket 1 selected

NOTE:

† The MBREQ alternate function must not be enabled until the PSSR[RDH] bit field is cleared. For more details, see Table 3-15, “PSSR Bit Definitions” on page 3-72.

6.4 Operation

The processor has three different memory spaces: SDRAM, static memory, and PC Card space. SDRAM has four partitions, static memory has six partitions, and PC Card space has two partitions (or sockets). When user software performs a memory burst across the boundary between any two adjacent partitions, the configurations for each partition must be identical. They must have the same external bus width, burst length, and so forth.

In theory, the partitions can be different types of memory sharing the same configuration characteristics. In practice, cross-partition memory bursts are conducted only when the two partitions hold the same memory type. A typical case is a transfer across two SDRAM partitions, 0 and 1, which are mandated to have the same characteristics.

Figure 6-1 is a block diagram of the maximum configuration of the memory controller.

6.4.1 Stacked SDRAM and Flash Memory

This section describes memory types that may be supported in the PXA271 and PXA272 processors.

6.4.1.1 Stacked SDRAM

On the Intel® PXA271 processor, SDRAM is stacked and connected to SDRAM partition 0. On systems using the Intel® PXA271 processor, external SDRAM memory chips must not exist within the same SDRAM partition pair as that of the internal stacked SDRAM. This could cause negative signal reflection to the stacked SDRAM device. See Section 6.4.2 for additional information on SDRAM partition pairs.

For the Intel® PXA271 processor stacked SDRAM, the Intel® PXA27x processor memory controller must be programmed to multiplex the SDRAM address lines out differently because the address lines to the stacked SDRAM are not connected to the usual MA<24:10> lines. The MDCNFG[STACKx] field on a Intel® PXA271 processor must be programmed to 0b01 to send the SDRAM address out on MA<24:23,13:1> for stacked 16-bit SDRAM of this product.

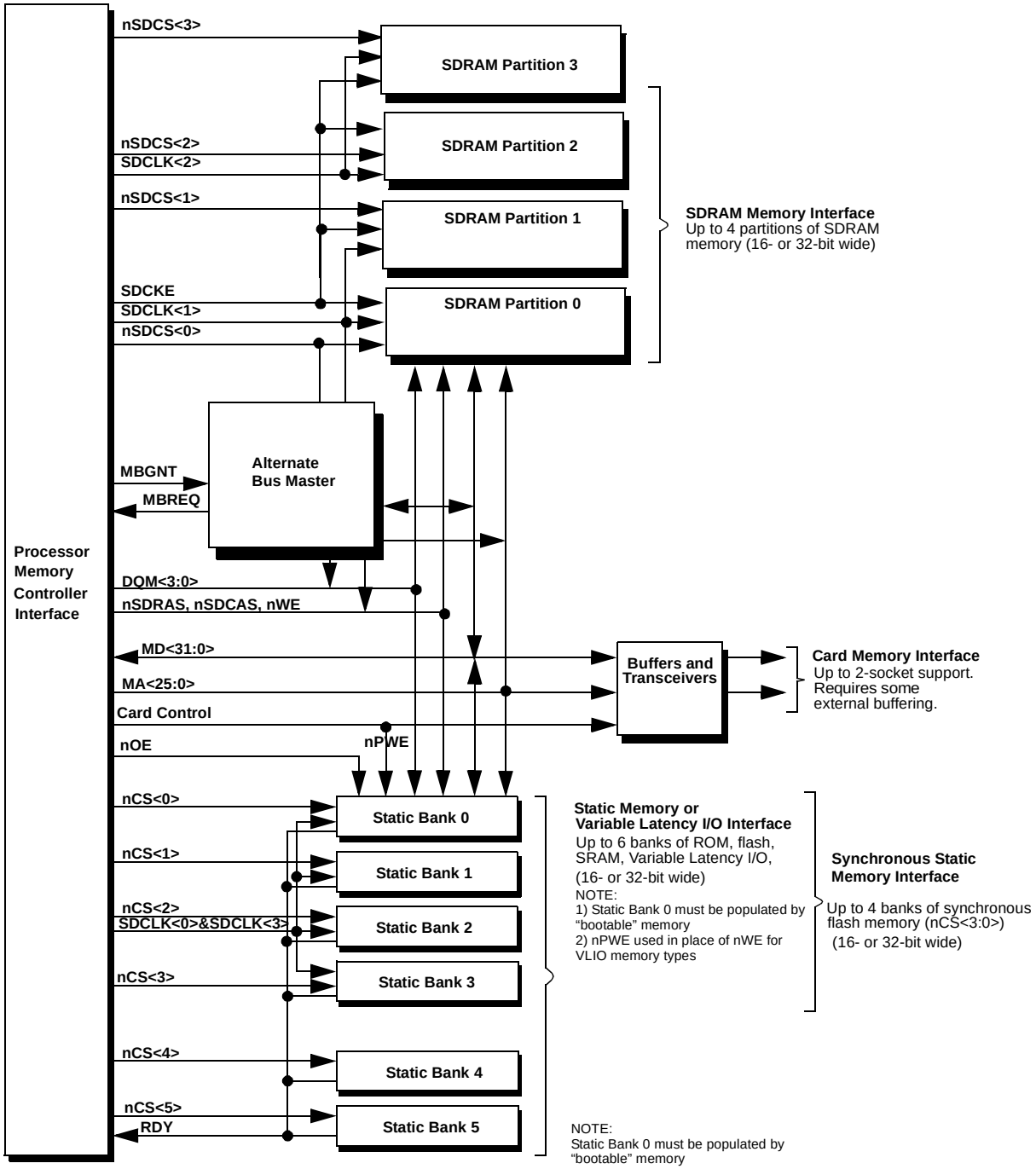
For a non stacked part (Intel® PXA270 processor) or flash only stacked part (Intel® PXA272 processor) the MDCNFG[STACKx] field must be programmed to 0b00 to send the SDRAM address out on MA<24:10>. See Section 6.5.1.1 for information on programming the MDCNFG[STACKx] field.

6.4.1.2 Stacked Flash Memory

A fourth SDCLK, SDCLK<3> is driven by the memory controller, to be used in the PXA271 and PXA272 processors containing stacked flash devices. SDCLK<3> is a buffer duplicate of SDCLK<0> and does not have any control bits of its own to turn it on or off. Use the buffer strength field associated with SDCLK<3> to turn off SDCLK<3> if there is no stacked flash in the system. This buffer strength setting is located in the BSCNTR2 register. See Section 6.5.7.3 for information on programming the SDCLK<3> buffer strength setting.

Static partitions 0 and 1 may contain stacked flash. The memory controller must be aware of which static memory partitions contain stacked flash. This is programmed in the SA1111[SXSTACK] field. See Section 6.5.3.2 for information on programming this field. When a flash device is being written to, the nCS and nWE signals swap functionality from a normal flash write to an off-chip device. This is shown by timing diagrams in the *Intel® PXA27x Processor Family EMTS*.

Figure 6-1. General Memory Interface Configuration



6.4.2 Synchronous Dynamic Memory (SDRAM) Interface

The processor supports the JEDEC synchronous dynamic memory (SDRAM) interface. The SDRAM interface supports four 16-bit or 32-bit wide partitions of SDRAM. Each partition is allocated 64 or 256 Mbytes of the internal memory map. The actual size of each partition depends on the SDRAM configuration used. The four partitions are divided into two partition pairs: the 0/1 pair and the 2/3 pair. Both partitions in a pair must be identical in size and configuration. Pairs 0/1 and 2/3 can be different. For example, the 0/1 pair can be 100-MHz SDRAM on a 32-bit data bus, while the 2/3 pair can be 50-MHz SDRAM on a 16-bit data bus.

The SDRAM interface includes the following:

- Four partition selects, nSDCS<3:0>
- Four byte mask signals, DQM<3:0>
- 15 multiplexed bank/row/column address signals, MA<24:10>, MA<24:23,14:2>, or MA<24:23,13:1>, depending on the MDCNFG[STACKx] setting
- One write enable, nWE
- One column-address strobe (nSDCAS)
- One row-address strobe (nSDRAS)
- One clock enable (SDCKE)
- Two clocks (SDCLK<2:1>)

The processor performs auto-refresh (CBR) during normal operation and supports self-refreshing SDRAM during sleep, deep-sleep, standby, and frequency-change modes. An SDRAM auto-power-down mode bit (MDREFR[APD]) can be set so that the two clocks (SDCLK<2:1>) and the clock-enable signal (SDCKE) to SDRAM are automatically de-asserted whenever none of the corresponding partitions is being accessed.

Each possible SDRAM section of the memory map is referred to as a partition, to distinguish them from banks internal to SDRAM devices.

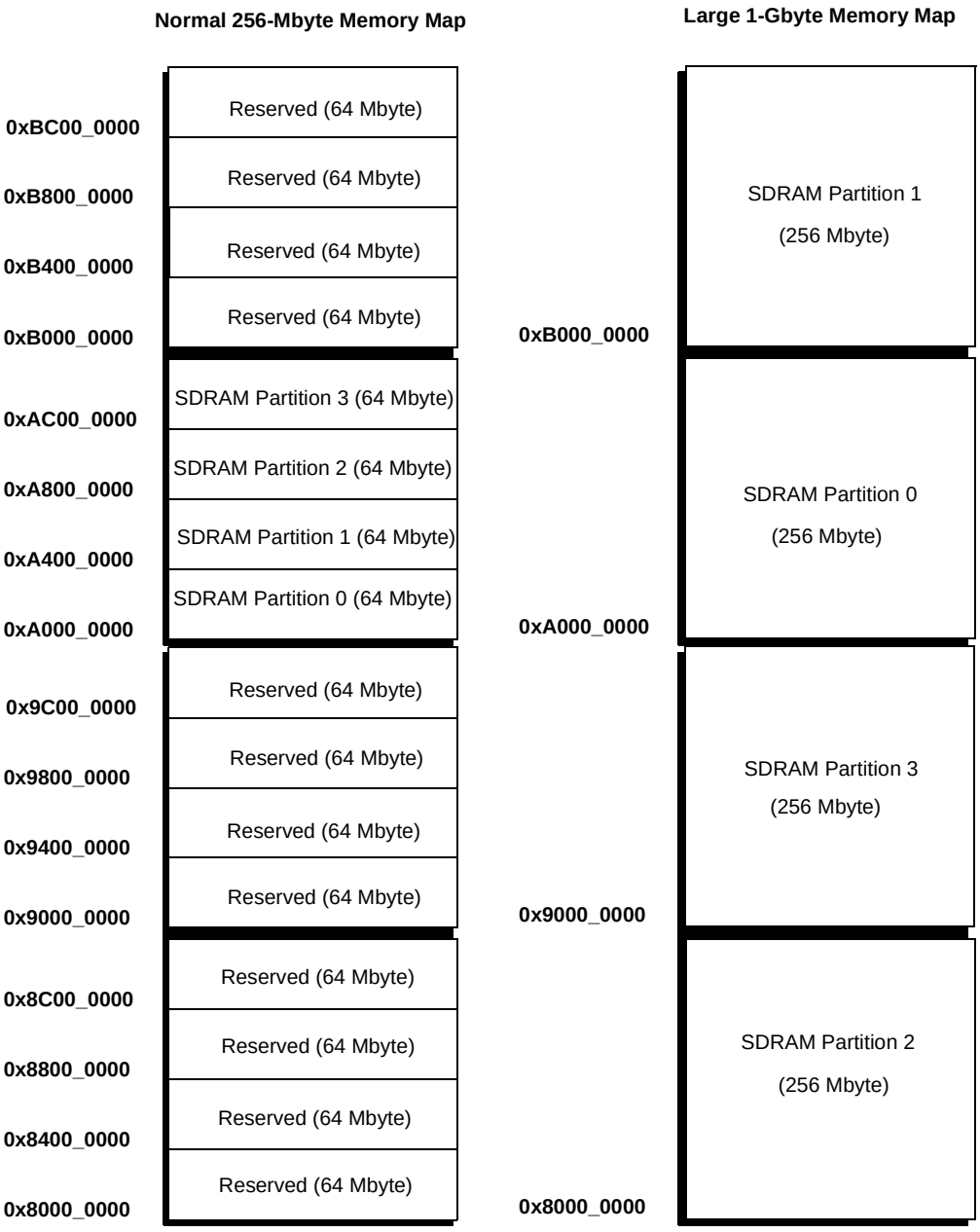
6.4.2.1 Maximum Row Active Time (T_{RAS})

The maximum amount of time that any SDRAM row can be active is defined as $T_{RAS_{MAX}}$. When programming MDREFR[DRI], ensure that the refresh cycle time is less than $T_{RAS_{MAX}}$ because it is not monitored by the memory controller.

6.4.2.2 Programmable Larger SDRAM Memory Space

The read/write MDCNFG register contains control bits for configuring the SDRAM for larger SDRAM configurations than fit in the 64-Mbyte SDRAM partitions. Refer to Table 6-23 for configuration programming. Figure 6-2 shows the programmable option for the SDRAM memory space.

Figure 6-2. Programmable SDRAM Memory Map Options



6.4.2.3 SDRAM Memory Size Options

The SDRAM interface supports up to four partitions, organized as two pairs. Both partitions within a pair must have the same SDRAM size, configuration, timing category, and data-bus width. Initialization software must set up the Memory Interface Configuration register with the SDRAM timing category, data-bus width, number of row, column, and bank-address bits, addressing scheme, and data-latching scheme.

Table 6-2 shows typical SDRAM configurations. Shaded rows are valid only in the programmable 1-GB SDRAM memory-map option (using MDCNFG[MDENX]).

Table 6-2. Example SDRAM Memory Size Options

Partition Size (Mbyte/Partition)		SDRAM Configuration (Words x Bits)	Chip Size	Number Chips/ Partition		Bank Bits x Row Bits x Column Bits	Maximum Memory (4 Partitions)		Total Number of Chips	
16-Bit Bus	32-Bit Bus			16-Bit Bus	32-Bit bus		16-Bit Bus	32-Bit Bus	16-Bit Bus	32-Bit Bus
2 Mbyte	4 Mbyte	1 M x 16	16 Mbit	1	2	1 x 11 x 8	8 Mbyte	16 Mbyte	4	8
4 Mbyte	8 Mbyte	2 M x 8	16 Mbit	2	4	1 x 11 x 9	16 Mbyte	32 Mbyte	8	16
8 Mbyte	16 Mbyte	4 M x 4	16 Mbit	4	8	1 x 11 x 10	32 Mbyte	64 Mbyte	16	32
N/A	8 Mbyte	2 M x 32	64 Mbit	N/A	1	2 x 11 x 8	N/A	32 Mbyte	N/A	4
8 Mbyte	16 Mbyte	4 M x 16	64 Mbit	1	2	1 x 13 x 8 2 x 12 x 8	32 Mbyte	64 Mbyte	4	8
16 Mbyte	32 Mbyte	8 M x 8	64 Mbit	2	4	1 x 13 x 9 2 x 12 x 9	64 Mbyte	128 Mbyte	8	16
32 Mbyte	64 Mbyte	16 M x 4	64 Mbit	4	8	1 x 13 x 10 2 x 12 x 10	128 Mbyte	256 Mbyte	16	32
16 Mbyte	32 Mbyte	8 M x 16	128 Mbit	1	2	2 x 12 x 9	64 Mbyte	128 Mbyte	4	8
32 Mbyte	64 Mbyte	16 M x 8	128 Mbit	2	4	2 x 12 x 10	128 Mbyte	256 Mbyte	8	16
64 Mbyte	N/A	32 M x 4	128 Mbit	4	N/A	2 x 12 x 11	256 Mbyte	N/A	16	N/A
32 Mbyte	64 Mbyte	16 M x 16	256 Mbit	1	2	2 x 13 x 9	128 Mbyte	256 Mbyte	4	8
64 Mbyte	N/A	32 M x 8	256 Mbit	2	N/A	2 x 13 x 10	256 Mbyte	N/A	8	N/A
128 Mbyte	256 Mbyte	64 M x 4	256 M bit	4	8	2 x 13 x 11	512 Mbyte	1 GB	16	32
128 Mbyte	256 Mbyte	64 M x 8	512 M bit	2	4	2 x 13 x 11	512 Mbyte	1 GB	8	16
256 Mbyte	N/A	128 M x 4	512 M bit	4	N/A	2 x 13 x 12	1 GB	N/A	16	N/A
64 Mbyte	128 Mbyte	32 M x 16	512 M bit	1	2	2 x 13 x 10	256 Mbyte	512 Mbyte	4	8

Figure 6-3 shows the bank, row, and column address multiplexing using 2 bank bits x 13 row bits x 9 column bits x 32 data bits as an example for both the normal bank addressing scheme and the alternate bank-addressing scheme. All unused address bits during row-address strobe (RAS) and column-address strobe (CAS) time, including MA<9:0> bits not shown here, retain their existing values.

The addressing modes are summarized in Table 6-3, Table 6-4, Table 6-5, Table 6-6, and Table 6-7. The SDRAM BA<1:0> must be connected to the processor address signals shown in **bold** font in Table 6-3–Table 6-7.

When accessing SDRAM, only MA<19:10> and possibly MA<22:21> (or MA<11:2> and possibly MA<13:12>, or MA<10:1> and possibly MA<12:11>, depending on the MDCNFG[STACKx] setting) are used for column addressing. MA<20> (or MA<12> or MA<11>, depending on the MDCNFG[STACKx] setting) is driven low during column addressing. BA<1:0> indicate to the SDRAM which bank is being read from and remains stable during column addressing. During SDRAM configuration, the address pins transfer the MRS command.

Figure 6-3. External-to-Internal Address Mapping Options

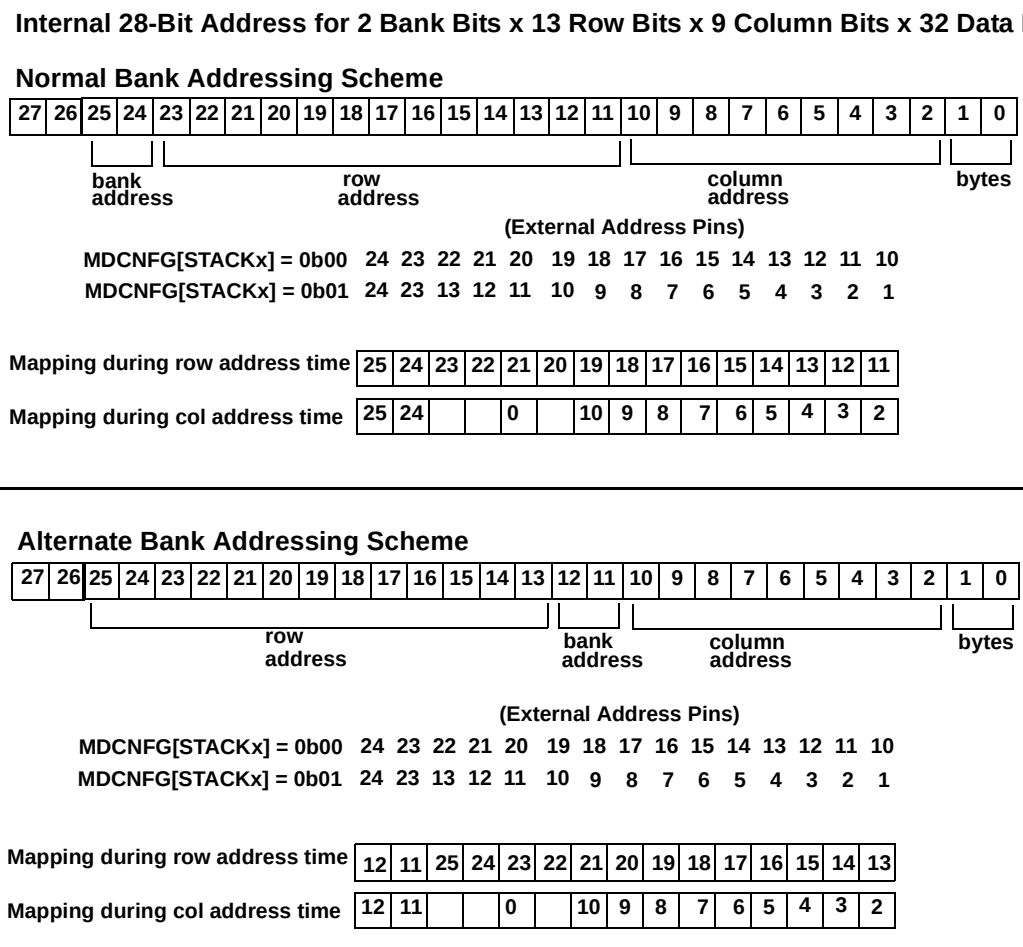


Table 6-3. 64-MB and 256-MB External-to-Internal Address Mapping Options: Normal Bank Addressing without Stacked Flash + SDRAM (Sheet 1 of 2) *(For use with PXA270 and PXA272)*

# Bits Bank x Row x Col x Data	External Address Pins at SDRAM RAS Time															External Address Pins at SDRAM CAS Time													
	MA<24:10> (MDCNFG[STACKx] = 0b00)															MA<24:10> (MDCNFG[STACKx] = 0b00)													
	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
1x11x8x32				**21**	20	19	18	17	16	15	14	13	12	11	10				**21**	0			9	8	7	6	5	4	3	2
1x11x8x16				**20**	19	18	17	16	15	14	13	12	11	10	9				**20**	0			8	7	6	5	4	3	2	1
1x11x9x32				**22**	21	20	19	18	17	16	15	14	13	12	11				**22**	0		10	9	8	7	6	5	4	3	2
1x11x9x16				**21**	20	19	18	17	16	15	14	13	12	11	10				**21**	0		9	8	7	6	5	4	3	2	1
1x11x10x32				**23**	22	21	20	19	18	17	16	15	14	13	12				**23**	0	11	10	9	8	7	6	5	4	3	2
1x11x10x16				**22**	21	20	19	18	17	16	15	14	13	12	11				**22**	0	10	9	8	7	6	5	4	3	2	1
1x11x11x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x11x11x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x11x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x11x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x12x8x32			**22**	21	20	19	18	17	16	15	14	13	12	11	10			**22**		0			9	8	7	6	5	4	3	2
1x12x8x16			**21**	20	19	18	17	16	15	14	13	12	11	10	9			**21**		0			8	7	6	5	4	3	2	1
1x12x9x32			**23**	22	21	20	19	18	17	16	15	14	13	12	11			**23**		0		10	9	8	7	6	5	4	3	2
1x12x9x16			**22**	21	20	19	18	17	16	15	14	13	12	11	10			**22**		0		9	8	7	6	5	4	3	2	1
1x12x10x32			**24**	23	22	21	20	19	18	17	16	15	14	13	12			**24**		0	11	10	9	8	7	6	5	4	3	2
1x12x10x16			**23**	22	21	20	19	18	17	16	15	14	13	12	11			**23**		0	10	9	8	7	6	5	4	3	2	1
1x12x11x32			**25**	24	23	22	21	20	19	18	17	16	15	14	13			**25**	12	0	11	10	9	8	7	6	5	4	3	2
1x12x11x16			**24**	23	22	21	20	19	18	17	16	15	14	13	12			**24**	11	0	10	9	8	7	6	5	4	3	2	1
1x12x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x12x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x13x8x32		**23**	22	21	20	19	18	17	16	15	14	13	12	11	10		**23**			0			9	8	7	6	5	4	3	2
1x13x8x16		**22**	21	20	19	18	17	16	15	14	13	12	11	10	9		**22**			0			8	7	6	5	4	3	2	1
1x13x9x32		**24**	23	22	21	20	19	18	17	16	15	14	13	12	11		**24**			0		10	9	8	7	6	5	4	3	2
1x13x9x16		**23**	22	21	20	19	18	17	16	15	14	13	12	11	10		**23**			0		9	8	7	6	5	4	3	2	1
1x13x10x32		**25**	24	23	22	21	20	19	18	17	16	15	14	13	12		**25**			0	11	10	9	8	7	6	5	4	3	2
1x13x10x16		**24**	23	22	21	20	19	18	17	16	15	14	13	12	11		**24**			0	10	9	8	7	6	5	4	3	2	1
1x13x11x32		**26**	25	24	23	22	21	20	19	18	17	16	15	14	13		**26**		12	0	11	10	9	8	7	6	5	4	3	2
1x13x11x16		**25**	24	23	22	21	20	19	18	17	16	15	14	13	12		**25**		11	0	10	9	8	7	6	5	4	3	2	1
1x13x12x32		**27**	26	25	24	23	22	21	20	19	18	17	16	15	14		**27**	13	12	0	11	10	9	8	7	6	5	4	3	2
1x13x12x16		**26**	25	24	23	22	21	20	19	18	17	16	15	14	13		**26**	12	11	0	10	9	8	7	6	5	4	3	2	1
2x11x8x32			**22**	**21**	20	19	18	17	16	15	14	13	12	11	10			**22**	**21**	0			9	8	7	6	5	4	3	2
2x11x8x16			**21**	**20**	19	18	17	16	15	14	13	12	11	10	9			**21**	**20**	0			8	7	6	5	4	3	2	1
2x11x9x32			**23**	**22**	21	20	19	18	17	16	15	14	13	12	11			**23**	**22**	0		10	9	8	7	6	5	4	3	2

Table 6-3. 64-MB and 256-MB External-to-Internal Address Mapping Options: Normal Bank Addressing without Stacked Flash + SDRAM (Sheet 2 of 2) *(For use with PXA270 and PXA272)*

# Bits	External Address Pins at SDRAM RAS Time															External Address Pins at SDRAM CAS Time													
Bank x Row x Col x Data	MA<24:10> (MDCNFG[STACKx] = 0b00)																											
	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
2x11x9x16			**22**	**21**	20	19	18	17	16	15	14	13	12	11	10			**22**	**21**	0		9	8	7	6	5	4	3	2	1
2x11x10x32			**24**	**23**	22	21	20	19	18	17	16	15	14	13	12			**24**	**23**	0	11	10	9	8	7	6	5	4	3	2
2x11x10x16			**23**	**22**	21	20	19	18	17	16	15	14	13	12	11			**23**	**22**	0	10	9	8	7	6	5	4	3	2	1
2x11x11x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x11x11x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x11x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x11x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x12x8x32		**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10		**23**	**22**		0			9	8	7	6	5	4	3	2
2x12x8x16		**22**	**21**	20	19	18	17	16	15	14	13	12	11	10	9		**22**	**21**		0			8	7	6	5	4	3	2	1
2x12x9x32		**24**	**23**	22	21	20	19	18	17	16	15	14	13	12	11		**24**	**23**		0		10	9	8	7	6	5	4	3	2
2x12x9x16		**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10		**23**	**22**		0		9	8	7	6	5	4	3	2	1
2x12x10x32		**25**	**24**	23	22	21	20	19	18	17	16	15	14	13	12		**25**	**24**		0	11	10	9	8	7	6	5	4	3	2
2x12x10x16		**24**	**23**	22	21	20	19	18	17	16	15	14	13	12	11		**24**	**23**		0	10	9	8	7	6	5	4	3	2	1
2x12x11x32		**26**	**25**	24	23	22	21	20	19	18	17	16	15	14	13		**26**	**25**	12	0	11	10	9	8	7	6	5	4	3	2
2x12x11x16		**25**	**24**	23	22	21	20	19	18	17	16	15	14	13	12		**25**	**24**	11	0	10	9	8	7	6	5	4	3	2	1
2x12x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x12x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x13x8x32	**24**	**23**	22	21	20	19	18	17	16	15	14	13	12	11	10	**24**	**23**			0			9	8	7	6	5	4	3	2
2x13x8x16	**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10	9	**23**	**22**			0			8	7	6	5	4	3	2	1
2x13x9x32	**25**	**24**	23	22	21	20	19	18	17	16	15	14	13	12	11	**25**	**24**			0		10	9	8	7	6	5	4	3	2
2x13x9x16	**24**	**23**	22	21	20	19	18	17	16	15	14	13	12	11	10	**24**	**23**			0		9	8	7	6	5	4	3	2	1
2x13x10x32	**26**	**25**	24	23	22	21	20	19	18	17	16	15	14	13	12	**26**	**25**			0	11	10	9	8	7	6	5	4	3	2
2x13x10x16	**25**	**24**	23	22	21	20	19	18	17	16	15	14	13	12	11	**25**	**24**			0	10	9	8	7	6	5	4	3	2	1
2x13x11x32	**27**	**26**	25	24	23	22	21	20	19	18	17	16	15	14	13	**27**	**26**		12	0	11	10	9	8	7	6	5	4	3	2
2x13x11x16	**26**	**25**	24	23	22	21	20	19	18	17	16	15	14	13	12	**26**	**25**		11	0	10	9	8	7	6	5	4	3	2	1
2x13x12x32	NOT VALID (too big)															NOT VALID (too big)													
2x13x12x16	**27**	**26**	25	24	23	22	21	20	19	18	17	16	15	14	13	**27**	**26**	12	11	0	10	9	8	7	6	5	4	3	2	1

Table 6-4. 64-MB and 256-MB External-to-Internal Address Mapping Options: Normal Bank Addressing with Stacked Flash + SDRAM (Sheet 1 of 2) *(For use with PXA271)*

# Bits	External Address pins at SDRAM RAS Time															External Address pins at SDRAM CAS Time												
Bank x Row x Col x Data	MA<24:23,13:1> (MDCNFG[STACKx] = 0b01)																											
	24	23	13	12	11	10	9	8	7	6	5	4	3	2	1	24	23	13	12	11	10	9	8	7	6	5	4	3	2	1
1x11x8x32				**21**	20	19	18	17	16	15	14	13	12	11	10				**21**	0			9	8	7	6	5	4	3	2
1x11x8x16				**20**	19	18	17	16	15	14	13	12	11	10	9				**20**	0			8	7	6	5	4	3	2	1
1x11x9x32				**22**	21	20	19	18	17	16	15	14	13	12	11				**22**	0		10	9	8	7	6	5	4	3	2
1x11x9x16				**21**	20	19	18	17	16	15	14	13	12	11	10				**21**	0		9	8	7	6	5	4	3	2	1
1x11x10x32				**23**	22	21	20	19	18	17	16	15	14	13	12				**23**	0	11	10	9	8	7	6	5	4	3	2
1x11x10x16				**22**	21	20	19	18	17	16	15	14	13	12	11				**22**	0	10	9	8	7	6	5	4	3	2	1
1x11x11x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x11x11x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x11x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x11x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x12x8x32		**22**		21	20	19	18	17	16	15	14	13	12	11	10		**22**			0			9	8	7	6	5	4	3	2
1x12x8x16		**21**		20	19	18	17	16	15	14	13	12	11	10	9		**21**			0			8	7	6	5	4	3	2	1
1x12x9x32		**23**		22	21	20	19	18	17	16	15	14	13	12	11		**23**			0		10	9	8	7	6	5	4	3	2
1x12x9x16		**22**		21	20	19	18	17	16	15	14	13	12	11	10		**22**			0		9	8	7	6	5	4	3	2	1
1x12x10x32		**24**		23	22	21	20	19	18	17	16	15	14	13	12		**24**			0	11	10	9	8	7	6	5	4	3	2
1x12x10x16		**23**		22	21	20	19	18	17	16	15	14	13	12	11		**23**			0	10	9	8	7	6	5	4	3	2	1
1x12x11x32		**25**		24	23	22	21	20	19	18	17	16	15	14	13		**25**		12	0	11	10	9	8	7	6	5	4	3	2
1x12x11x16		**24**		23	22	21	20	19	18	17	16	15	14	13	12		**24**		11	0	10	9	8	7	6	5	4	3	2	1
1x12x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x12x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x13x8x32		**23**	22	21	20	19	18	17	16	15	14	13	12	11	10		**23**			0			9	8	7	6	5	4	3	2
1x13x8x16		**22**	21	20	19	18	17	16	15	14	13	12	11	10	9		**22**			0			8	7	6	5	4	3	2	1
1x13x9x32		**24**	23	22	21	20	19	18	17	16	15	14	13	12	11		**24**			0		10	9	8	7	6	5	4	3	2
1x13x9x16		**23**	22	21	20	19	18	17	16	15	14	13	12	11	10		**23**			0		9	8	7	6	5	4	3	2	1
1x13x10x32		**25**	24	23	22	21	20	19	18	17	16	15	14	13	12		**25**			0	11	10	9	8	7	6	5	4	3	2
1x13x10x16		**24**	23	22	21	20	19	18	17	16	15	14	13	12	11		**24**			0	10	9	8	7	6	5	4	3	2	1
1x13x11x32		**26**	25	24	23	22	21	20	19	18	17	16	15	14	13		**26**		12	0	11	10	9	8	7	6	5	4	3	2
1x13x11x16		**25**	24	23	22	21	20	19	18	17	16	15	14	13	12		**25**		11	0	10	9	8	7	6	5	4	3	2	1
1x13x12x32		**27**	26	25	24	23	22	21	20	19	18	17	16	15	14		**27**	13	12	0	11	10	9	8	7	6	5	4	3	2
1x13x12x16		**26**	25	24	23	22	21	20	19	18	17	16	15	14	13		**26**	12	11	0	10	9	8	7	6	5	4	3	2	1
2x11x8x32	**22**	**21**			20	19	18	17	16	15	14	13	12	11	10	**22**	**21**			0			9	8	7	6	5	4	3	2
2x11x8x16	**21**	**20**			19	18	17	16	15	14	13	12	11	10	9	**21**	**20**			0			8	7	6	5	4	3	2	1
2x11x9x32	**23**	**22**			21	20	19	18	17	16	15	14	13	12	11	**23**	**22**			0		10	9	8	7	6	5	4	3	2

Table 6-4. 64-MB and 256-MB External-to-Internal Address Mapping Options: Normal Bank Addressing with Stacked Flash + SDRAM (Sheet 2 of 2) *(For use with PXA271)*

# Bits	External Address pins at SDRAM RAS Time															External Address pins at SDRAM CAS Time													
Bank x Row x Col x Data	MA<24:23,13:1> (MDCNFG[STACKx] = 0b01)																												
	24	23	13	12	11	10	9	8	7	6	5	4	3	2	1	24	23	13	12	11	10	9	8	7	6	5	4	3	2	1
2x11x9x16	**22**	**21**			20	19	18	17	16	15	14	13	12	11	10	**22**	**21**			0		9	8	7	6	5	4	3	2	1
2x11x10x32	**24**	**23**			22	21	20	19	18	17	16	15	14	13	12	**24**	**23**			0	11	10	9	8	7	6	5	4	3	2
2x11x10x16	**23**	**22**			21	20	19	18	17	16	15	14	13	12	11	**23**	**22**			0	10	9	8	7	6	5	4	3	2	1
2x11x11x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
2x11x11x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
2x11x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
2x11x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
2x12x8x32	**23**	**22**		21	20	19	18	17	16	15	14	13	12	11	10	**23**	**22**			0			9	8	7	6	5	4	3	2
2x12x8x16	**22**	**21**		20	19	18	17	16	15	14	13	12	11	10	9	**22**	**21**			0			8	7	6	5	4	3	2	1
2x12x9x32	**24**	**23**		22	21	20	19	18	17	16	15	14	13	12	11	**24**	**23**			0		10	9	8	7	6	5	4	3	2
2x12x9x16	**23**	**22**		21	20	19	18	17	16	15	14	13	12	11	10	**23**	**22**			0		9	8	7	6	5	4	3	2	1
2x12x10x32	**25**	**24**		23	22	21	20	19	18	17	16	15	14	13	12	**25**	**24**			0	11	10	9	8	7	6	5	4	3	2
2x12x10x16	**24**	**23**		22	21	20	19	18	17	16	15	14	13	12	11	**24**	**23**			0	10	9	8	7	6	5	4	3	2	1
2x12x11x32	**26**	**25**		24	23	22	21	20	19	18	17	16	15	14	13	**26**	**25**		12	0	11	10	9	8	7	6	5	4	3	2
2x12x11x16	**25**	**24**		23	22	21	20	19	18	17	16	15	14	13	12	**25**	**24**		11	0	10	9	8	7	6	5	4	3	2	1
2x12x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
2x12x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
2x13x8x32	**24**	**23**	22	21	20	19	18	17	16	15	14	13	12	11	10	**24**	**23**			0			9	8	7	6	5	4	3	2
2x13x8x16	**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10	9	**23**	**22**			0			8	7	6	5	4	3	2	1
2x13x9x32	**25**	**24**	23	22	21	20	19	18	17	16	15	14	13	12	11	**25**	**24**			0		10	9	8	7	6	5	4	3	2
2x13x9x16	**24**	**23**	22	21	20	19	18	17	16	15	14	13	12	11	10	**24**	**23**			0		9	8	7	6	5	4	3	2	1
2x13x10x32	**26**	**25**	24	23	22	21	20	19	18	17	16	15	14	13	12	**26**	**25**			0	11	10	9	8	7	6	5	4	3	2
2x13x10x16	**25**	**24**	23	22	21	20	19	18	17	16	15	14	13	12	11	**25**	**24**			0	10	9	8	7	6	5	4	3	2	1
2x13x11x32	**27**	**26**	25	24	23	22	21	20	19	18	17	16	15	14	13	**27**	**26**		12	0	11	10	9	8	7	6	5	4	3	2
2x13x11x16	**26**	**25**	24	23	22	21	20	19	18	17	16	15	14	13	12	**26**	**25**		11	0	10	9	8	7	6	5	4	3	2	1
2x13x12x32	NOT VALID (too big)															NOT VALID (too big)														
2x13x12x16	**27**	**26**	25	24	23	22	21	20	19	18	17	16	15	14	13	**27**	**26**	12	11	0	10	9	8	7	6	5	4	3	2	1

Table 6-5. 64-MB and 256-MB External-to-Internal Address Mapping Options: Alternate Bank Addressing without Stacked Flash (Sheet 1 of 2)

# Bits Bank x Row x Col x Data	External Address Pins at SDRAM RAS Time															External Address Pins at SDRAM CAS Time													
	MA<24:10> (MDCNFG[STACKx] = 0b00)																											
	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
1x11x8x32				**10**	21	20	19	18	17	16	15	14	13	12	11				**10**	0			9	8	7	6	5	4	3	2
1x11x8x16				**9**	20	19	18	17	16	15	14	13	12	11	10				**9**	0			8	7	6	5	4	3	2	1
1x11x9x32				**11**	22	21	20	19	18	17	16	15	14	13	12				**11**	0		10	9	8	7	6	5	4	3	2
1x11x9x16				**10**	21	20	19	18	17	16	15	14	13	12	11				**10**	0		9	8	7	6	5	4	3	2	1
1x11x10x32				**12**	23	22	21	20	19	18	17	16	15	14	13				**12**	0	11	10	9	8	7	6	5	4	3	2
1x11x10x16				**11**	22	21	20	19	18	17	16	15	14	13	12				**11**	0	10	9	8	7	6	5	4	3	2	1
1x11x11x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x11x11x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x11x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x11x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x12x8x32			**10**	22	21	20	19	18	17	16	15	14	13	12	11			**10**		0			9	8	7	6	5	4	3	2
1x12x8x16			**9**	21	20	19	18	17	16	15	14	13	12	11	10			**9**		0			8	7	6	5	4	3	2	1
1x12x9x32			**11**	23	22	21	20	19	18	17	16	15	14	13	12			**11**		0		10	9	8	7	6	5	4	3	2
1x12x9x16			**10**	22	21	20	19	18	17	16	15	14	13	12	11			**10**		0		9	8	7	6	5	4	3	2	1
1x12x10x32			**12**	24	23	22	21	20	19	18	17	16	15	14	13			**12**		0	11	10	9	8	7	6	5	4	3	2
1x12x10x16			**11**	23	22	21	20	19	18	17	16	15	14	13	12			**11**		0	10	9	8	7	6	5	4	3	2	1
1x12x11x32			**13**	25	24	23	22	21	20	19	18	17	16	15	14			**13**	12	0	11	10	9	8	7	6	5	4	3	2
1x12x11x16			**12**	24	23	22	21	20	19	18	17	16	15	14	13			**12**	11	0	10	9	8	7	6	5	4	3	2	1
1x12x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x12x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
1x13x8x32		**10**	23	22	21	20	19	18	17	16	15	14	13	12	11		**10**			0			9	8	7	6	5	4	3	2
1x13x8x16		**9**	22	21	20	19	18	17	16	15	14	13	12	11	10		**9**			0			8	7	6	5	4	3	2	1
1x13x9x32		**11**	24	23	22	21	20	19	18	17	16	15	14	13	12		**11**			0		10	9	8	7	6	5	4	3	2
1x13x9x16		**10**	23	22	21	20	19	18	17	16	15	14	13	12	11		**10**			0		9	8	7	6	5	4	3	2	1
1x13x10x32		**12**	25	24	23	22	21	20	19	18	17	16	15	14	13		**12**			0	11	10	9	8	7	6	5	4	3	2
1x13x10x16		**11**	24	23	22	21	20	19	18	17	16	15	14	13	12		**11**			0	10	9	8	7	6	5	4	3	2	1
1x13x11x32		**13**	26	25	24	23	22	21	20	19	18	17	16	15	14		**13**		12	0	11	10	9	8	7	6	5	4	3	2
1x13x11x16		**12**	25	24	23	22	21	20	19	18	17	16	15	14	13		**12**		11	0	10	9	8	7	6	5	4	3	2	1
1x13x12x32		**14**	27	26	25	24	23	22	21	20	19	18	17	16	15		**14**	13	12	0	11	10	9	8	7	6	5	4	3	2
1x13x12x16		**13**	26	25	24	23	22	21	20	19	18	17	16	15	14		**13**	12	11	0	10	9	8	7	6	5	4	3	2	1
2x11x8x32			**11**	**10**	22	21	20	19	18	17	16	15	14	13	12			**11**	**10**	0			9	8	7	6	5	4	3	2
2x11x8x16			**10**	**9**	21	20	19	18	17	16	15	14	13	12	11			**10**	**9**	0			8	7	6	5	4	3	2	1
2x11x9x32			**12**	**11**	23	22	21	20	19	18	17	16	15	14	13			**12**	**11**	0		10	9	8	7	6	5	4	3	2
2x11x9x16			**11**	**10**	22	21	20	19	18	17	16	15	14	13	12			**11**	**10**	0		9	8	7	6	5	4	3	2	1

Table 6-5. 64-MB and 256-MB External-to-Internal Address Mapping Options: Alternate Bank Addressing without Stacked Flash (Sheet 2 of 2)

# Bits Bank x Row x Col x Data	External Address Pins at SDRAM RAS Time															External Address Pins at SDRAM CAS Time													
	MA<24:10> (MDCNFG[STACKx] = 0b00)																												
	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
2x11x10x32			**13**	**12**	24	23	22	21	20	19	18	17	16	15	14			**13**	**12**	0	11	10	9	8	7	6	5	4	3	2
2x11x10x16			**12**	**11**	23	22	21	20	19	18	17	16	15	14	13			**12**	**11**	0	10	9	8	7	6	5	4	3	2	1
2x11x11x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x11x11x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x11x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x11x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x12x8x32		**11**	**10**	23	22	21	20	19	18	17	16	15	14	13	12		**11**	**10**		0			9	8	7	6	5	4	3	2
2x12x8x16		**10**	**9**	22	21	20	19	18	17	16	15	14	13	12	11		**10**	**9**		0			8	7	6	5	4	3	2	1
2x12x9x32		**12**	**11**	24	23	22	21	20	19	18	17	16	15	14	13		**12**	**11**		0		10	9	8	7	6	5	4	3	2
2x12x9x16		**11**	**10**	23	22	21	20	19	18	17	16	15	14	13	12		**11**	**10**		0		9	8	7	6	5	4	3	2	1
2x12x10x32		**13**	**12**	25	24	23	22	21	20	19	18	17	16	15	14		**13**	**12**		0	11	10	9	8	7	6	5	4	3	2
2x12x10x16		**12**	**11**	24	23	22	21	20	19	18	17	16	15	14	13		**12**	**11**		0	10	9	8	7	6	5	4	3	2	1
2x12x11x32		**14**	**13**	26	25	24	23	22	21	20	19	18	17	16	15		**14**	**13**	12	0	11	10	9	8	7	6	5	4	3	2
2x12x11x16		**13**	**12**	25	24	23	22	21	20	19	18	17	16	15	14		**13**	**12**	11	0	10	9	8	7	6	5	4	3	2	1
2x12x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x12x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x13x8x32	**11**	**10**	24	23	22	21	20	19	18	17	16	15	14	13	12	**11**	**10**			0			9	8	7	6	5	4	3	2
2x13x8x16	**10**	**9**	23	22	21	20	19	18	17	16	15	14	13	12	11	**10**	**9**			0			8	7	6	5	4	3	2	1
2x13x9x32	**12**	**11**	25	24	23	22	21	20	19	18	17	16	15	14	13	**12**	**11**			0		10	9	8	7	6	5	4	3	2
2x13x9x16	**11**	**10**	24	23	22	21	20	19	18	17	16	15	14	13	12	**11**	**10**			0		9	8	7	6	5	4	3	2	1
2x13x10x32	**13**	**12**	26	25	24	23	22	21	20	19	18	17	16	15	14	**13**	**12**			0	11	10	9	8	7	6	5	4	3	2
2x13x10x16	**12**	**11**	25	24	23	22	21	20	19	18	17	16	15	14	13	**12**	**11**			0	10	9	8	7	6	5	4	3	2	1
2x13x11x32	**14**	**13**	27	26	25	24	23	22	21	20	19	18	17	16	15	**14**	**13**		12	0	11	10	9	8	7	6	5	4	3	2
2x13x11x16	**13**	**12**	26	25	24	23	22	21	20	19	18	17	16	15	14	**13**	**12**		11	0	10	9	8	7	6	5	4	3	2	1
2x13x12x32	NOT VALID (too big)															NOT VALID (too big)													
2x13x12x16	**14**	**13**	27	26	25	24	23	22	21	20	19	18	17	16	15	**14**	**13**	12	11	0	10	9	8	7	6	5	4	3	2	1

Table 6-6. 64-MB and 256-MB External-to-Internal Address Mapping Options: Alternate Bank Addressing with Stacked Flash (Sheet 1 of 2)

# Bits	External Address Pins at SDRAM RAS Time															External Address Pins at SDRAM CAS Time													
Bank x Row x Col x Data	MA<24:23,13:1> (MDCNFG[STACKx] = 0b01)																											
	24	23	13	12	11	10	9	8	7	6	5	4	3	2	1	24	23	13	12	11	10	9	8	7	6	5	4	3	2	1
1x11x8x32		**10**			21	20	19	18	17	16	15	14	13	12	11		**10**			0			9	8	7	6	5	4	3	2
1x11x8x16		**9**			20	19	18	17	16	15	14	13	12	11	10		**9**			0			8	7	6	5	4	3	2	1
1x11x9x32		**11**			22	21	20	19	18	17	16	15	14	13	12		**11**			0		10	9	8	7	6	5	4	3	2
1x11x9x16		**10**			21	20	19	18	17	16	15	14	13	12	11		**10**			0		9	8	7	6	5	4	3	2	1
1x11x10x32		**12**			23	22	21	20	19	18	17	16	15	14	13		**12**			0	11	10	9	8	7	6	5	4	3	2
1x11x10x16		**11**			22	21	20	19	18	17	16	15	14	13	12		**11**			0	10	9	8	7	6	5	4	3	2	1
1x11x11x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
1x11x11x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
1x11x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
1x11x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
1x12x8x32		**10**		22	21	20	19	18	17	16	15	14	13	12	11		**10**			0			9	8	7	6	5	4	3	2
1x12x8x16		**9**		21	20	19	18	17	16	15	14	13	12	11	10		**9**			0			8	7	6	5	4	3	2	1
1x12x9x32		**11**		23	22	21	20	19	18	17	16	15	14	13	12		**11**			0		10	9	8	7	6	5	4	3	2
1x12x9x16		**10**		22	21	20	19	18	17	16	15	14	13	12	11		**10**			0		9	8	7	6	5	4	3	2	1
1x12x10x32		**12**		24	23	22	21	20	19	18	17	16	15	14	13		**12**			0	11	10	9	8	7	6	5	4	3	2
1x12x10x16		**11**		23	22	21	20	19	18	17	16	15	14	13	12		**11**			0	10	9	8	7	6	5	4	3	2	1
1x12x11x32		**13**		25	24	23	22	21	20	19	18	17	16	15	14		**13**		12	0	11	10	9	8	7	6	5	4	3	2
1x12x11x16		**12**		24	23	22	21	20	19	18	17	16	15	14	13		**12**		11	0	10	9	8	7	6	5	4	3	2	1
1x12x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
1x12x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
1x13x8x32		**10**	23	22	21	20	19	18	17	16	15	14	13	12	11		**10**			0			9	8	7	6	5	4	3	2
1x13x8x16		**9**	22	21	20	19	18	17	16	15	14	13	12	11	10		**9**			0			8	7	6	5	4	3	2	1
1x13x9x32		**11**	24	23	22	21	20	19	18	17	16	15	14	13	12		**11**			0		10	9	8	7	6	5	4	3	2
1x13x9x16		**10**	23	22	21	20	19	18	17	16	15	14	13	12	11		**10**			0		9	8	7	6	5	4	3	2	1
1x13x10x32		**12**	25	24	23	22	21	20	19	18	17	16	15	14	13		**12**			0	11	10	9	8	7	6	5	4	3	2
1x13x10x16		**11**	24	23	22	21	20	19	18	17	16	15	14	13	12		**11**			0	10	9	8	7	6	5	4	3	2	1
1x13x11x32		**13**	26	25	24	23	22	21	20	19	18	17	16	15	14		**13**		12	0	11	10	9	8	7	6	5	4	3	2
1x13x11x16		**12**	25	24	23	22	21	20	19	18	17	16	15	14	13		**12**		11	0	10	9	8	7	6	5	4	3	2	1
1x13x12x32		**14**	27	26	25	24	23	22	21	20	19	18	17	16	15		**14**	13	12	0	11	10	9	8	7	6	5	4	3	2
1x13x12x16		**13**	26	25	24	23	22	21	20	19	18	17	16	15	14		**13**	12	11	0	10	9	8	7	6	5	4	3	2	1
2x11x8x32	**11**	**10**			22	21	20	19	18	17	16	15	14	13	12	**11**	**10**			0			9	8	7	6	5	4	3	2
2x11x8x16	**10**	**9**			21	20	19	18	17	16	15	14	13	12	11	**10**	**9**			0			8	7	6	5	4	3	2	1
2x11x9x32	**12**	**11**			23	22	21	20	19	18	17	16	15	14	13	**12**	**11**			0		10	9	8	7	6	5	4	3	2
2x11x9x16	**11**	**10**			22	21	20	19	18	17	16	15	14	13	12	**11**	**10**			0		9	8	7	6	5	4	3	2	1

Table 6-6. 64-MB and 256-MB External-to-Internal Address Mapping Options: Alternate Bank Addressing with Stacked Flash (Sheet 2 of 2)

# Bits Bank x Row x Col x Data	External Address Pins at SDRAM RAS Time															External Address Pins at SDRAM CAS Time												
	MA<24:23,13:1> (MDCNFG[STACKx] = 0b01)																											
	24	23	13	12	11	10	9	8	7	6	5	4	3	2	1	24	23	13	12	11	10	9	8	7	6	5	4	3	2	1
2x11x10x32	**13**	**12**			24	23	22	21	20	19	18	17	16	15	14	**13**	**12**			0	11	10	9	8	7	6	5	4	3	2
2x11x10x16	**12**	**11**			23	22	21	20	19	18	17	16	15	14	13	**12**	**11**			0	10	9	8	7	6	5	4	3	2	1
2x11x11x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
2x11x11x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
2x11x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
2x11x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
2x12x8x32	**11**	**10**		23	22	21	20	19	18	17	16	15	14	13	12	**11**	**10**			0			9	8	7	6	5	4	3	2
2x12x8x16	**10**	**9**		22	21	20	19	18	17	16	15	14	13	12	11	**10**	**9**			0			8	7	6	5	4	3	2	1
2x12x9x32	**12**	**11**		24	23	22	21	20	19	18	17	16	15	14	13	**12**	**11**			0		10	9	8	7	6	5	4	3	2
2x12x9x16	**11**	**10**		23	22	21	20	19	18	17	16	15	14	13	12	**11**	**10**			0		9	8	7	6	5	4	3	2	1
2x12x10x32	**13**	**12**		25	24	23	22	21	20	19	18	17	16	15	14	**13**	**12**			0	11	10	9	8	7	6	5	4	3	2
2x12x10x16	**12**	**11**		24	23	22	21	20	19	18	17	16	15	14	13	**12**	**11**			0	10	9	8	7	6	5	4	3	2	1
2x12x11x32	**14**	**13**		26	25	24	23	22	21	20	19	18	17	16	15	**14**	**13**		12	0	11	10	9	8	7	6	5	4	3	2
2x12x11x16	**13**	**12**		25	24	23	22	21	20	19	18	17	16	15	14	**13**	**12**		11	0	10	9	8	7	6	5	4	3	2	1
2x12x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
2x12x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)														
2x13x8x32	**11**	**10**	24	23	22	21	20	19	18	17	16	15	14	13	12	**11**	**10**			0			9	8	7	6	5	4	3	2
2x13x8x16	**10**	**9**	23	22	21	20	19	18	17	16	15	14	13	12	11	**10**	**9**			0			8	7	6	5	4	3	2	1
2x13x9x32	**12**	**11**	25	24	23	22	21	20	19	18	17	16	15	14	13	**12**	**11**			0		10	9	8	7	6	5	4	3	2
2x13x9x16	**11**	**10**	24	23	22	21	20	19	18	17	16	15	14	13	12	**11**	**10**			0		9	8	7	6	5	4	3	2	1
2x13x10x32	**13**	**12**	26	25	24	23	22	21	20	19	18	17	16	15	14	**13**	**12**			0	11	10	9	8	7	6	5	4	3	2
2x13x10x16	**12**	**11**	25	24	23	22	21	20	19	18	17	16	15	14	13	**12**	**11**			0	10	9	8	7	6	5	4	3	2	1
2x13x11x32	**14**	**13**	27	26	25	24	23	22	21	20	19	18	17	16	15	**14**	**13**		12	0	11	10	9	8	7	6	5	4	3	2
2x13x11x16	**13**	**12**	26	25	24	23	22	21	20	19	18	17	16	15	14	**13**	**12**		11	0	10	9	8	7	6	5	4	3	2	1
2x13x12x32	NOT VALID (too big)															NOT VALID (too big)														
2x13x12x16	**14**	**13**	27	26	25	24	23	22	21	20	19	18	17	16	15	**14**	**13**	12	11	0	10	9	8	7	6	5	4	3	2	1

Table 6-7. 64-MB External-to-Internal Address Mapping Options: SA-1110 Addressing (Sheet 1 of 2)

# Bits	External Address Pins at SDRAM RAS Time																External Address Pins at SDRAM CAS Time													
Bank x Row x Col x Data	MA<24:10> (MDCNFG[STACKx] = 0b00) SA-1110 Addressing Not Supported with MDCNFG[STACKx] = 0b01																													
	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10		24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
1x11x8x32				**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0			9	8	7	6	5	4	3	2
1x11x8x16	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x11x9x32				**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0		22	9	8	7	6	5	4	3	2
1x11x9x16				**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0		9	8	7	6	5	4	3	2	1
1x11x10x32				**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0	23	22	9	8	7	6	5	4	3	2
1x11x10x16				**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0	22	9	8	7	6	5	4	3	2	1
1x11x11x32	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x11x11x16	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x11x12x32	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x11x12x16	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x12x8x32			22	**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0			9	8	7	6	5	4	3	2
1x12x8x16	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x12x9x32			22	**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0		23	9	8	7	6	5	4	3	2
1x12x9x16			22	**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0		9	8	7	6	5	4	3	2	1
1x12x10x32			22	**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0	24	23	9	8	7	6	5	4	3	2
1x12x10x16			22	**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0	23	9	8	7	6	5	4	3	2	1
1x12x11x32	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x12x11x16	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x12x12x32	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x12x12x16	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x13x8x32		23	22	**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0			9	8	7	6	5	4	3	2
1x13x8x16	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x13x9x32		23	22	**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0		24	9	8	7	6	5	4	3	2
1x13x9x16		23	22	**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0		9	8	7	6	5	4	3	2	1
1x13x10x32		23	22	**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0	25	24	9	8	7	6	5	4	3	2
1x13x10x16		23	22	**21**	20	19	18	17	16	15	14	13	12	11	10					**21**	0	24	9	8	7	6	5	4	3	2	1
1x13x11x32	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x13x11x16	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x13x12x32	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
1x13x12x16	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
2x11x8x32			**22**	**21**	20	19	18	17	16	15	14	13	12	11	10				**22**	**21**	0			9	8	7	6	5	4	3	2
2x11x8x16	NOT VALID (illegal addressing combination)																NOT VALID (illegal addressing combination)														
2x11x9x32			**22**	**21**	20	19	18	17	16	15	14	13	12	11	10				**22**	**21**	0		23	9	8	7	6	5	4	3	2
2x11x9x16			**22**	**21**	20	19	18	17	16	15	14	13	12	11	10				**22**	**21**	0		9	8	7	6	5	4	3	2	1

Table 6-7. 64-MB External-to-Internal Address Mapping Options: SA-1110 Addressing (Sheet 2 of 2)

# Bits Bank x Row x Col x Data	External Address Pins at SDRAM RAS Time															External Address Pins at SDRAM CAS Time													
	MA<24:10> (MDCNFG[STACKx] = 0b00) SA-1110 Addressing Not Supported with MDCNFG[STACKx] = 0b01																											
	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
2x11x10x32			**22**	**21**	20	19	18	17	16	15	14	13	12	11	10			**22**	**21**	0	24	23	9	8	7	6	5	4	3	2
2x11x10x16			**22**	**21**	20	19	18	17	16	15	14	13	12	11	10			**22**	**21**	0	23	9	8	7	6	5	4	3	2	1
2x11x11x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x11x11x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x11x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x11x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x12x8x32		**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10		**23**	**22**		0			9	8	7	6	5	4	3	2
2x12x8x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x12x9x32		**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10		**23**	**22**		0		24	9	8	7	6	5	4	3	2
2x12x9x16		**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10		**23**	**22**		0		9	8	7	6	5	4	3	2	1
2x12x10x32		**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10		**23**	**22**		0	25	24	9	8	7	6	5	4	3	2
2x12x10x16		**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10		**23**	**22**		0	24	9	8	7	6	5	4	3	2	1
2x12x11x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x12x11x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x12x12x32	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x12x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x13x8x32	24	**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10		**23**	**22**		0			9	8	7	6	5	4	3	2
2x13x8x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													
2x13x9x32	24	**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10		**23**	**22**		0		25	9	8	7	6	5	4	3	2
2x13x9x16	24	**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10		**23**	**22**		0		9	8	7	6	5	4	3	2	1
2x13x10x32	NOT VALID (too big)															NOT VALID (too big)													
2x13x10x16	24	**23**	**22**	21	20	19	18	17	16	15	14	13	12	11	10		**23**	**22**		0	25	9	8	7	6	5	4	3	2	1
2x13x11x32	NOT VALID (too big)															NOT VALID (too big)													
2x13x11x16	NOT VALID (too big)															NOT VALID (too big)													
2x13x12x32	NOT VALID (too big)															NOT VALID (too big)													
2x13x12x16	NOT VALID (illegal addressing combination)															NOT VALID (illegal addressing combination)													

6.4.2.4 SDRAM Command Overview

The processor supports most x4, x8, x16, and x32 SDRAM memory devices and includes the following features:

- 15 multiplexed bank, row, and column address signals, MA<24:10>
- Four chip-select signals, nSDCS<3:0>
- Four data qualifiers for byte mask signals, DQM<3:0>
- 32 data signals, MD<31:0>
- One write-enable signal, nWE
- One row-address strobe, nSDRAS
- One column-address strobe, nSDCAS
- Two memory clocks, SDCLK<2:1>
- One memory clock-enable, SDCKE

After a write to the MDMRS register, a mode-register-set (MRS) command for each enabled SDRAM partition is sent to the SDRAM devices. The MRS command always configures SDRAM internal mode registers for sequential (linear) burst type and a burst length of four (unless entering alternate bus master mode in SA-1110 addressing mode—see Section 6.4.6). The CAS latency is determined by the MDCNFG[DTC0] and MDCNFG[DTC2] bit fields.

The processor accesses SDRAM using the following subset of standard interface commands:

- Mode register set (MRS)
- Bank activate (ACT)
- Read (READ)
- Write (WRITE)
- Precharge all banks (PALL)
- Precharge one bank (PRE)
- Auto-refresh (CBR)
- Power-down (PWRDN)
- Enter self-refresh (SLFRSH)
- Exit power-down (PWRDNX)
- No operation (NOP)

Table 6-8 shows the SDRAM interface commands.

Table 6-8. SDRAM Command Encoding

Command		Processor Pins									
		SDCKE (at clk n-1)	SDCKE (at clk n)	nSDCS <3:0>	nSDRAS	nSDCAS	nWE	DQM <3:0>	MA <24:10>		
									24:21	20	19:10
PWRDN		1	0	1	1	1	1	1	x		
PWRDNX		0	1	1	1	1	1	1	x		
SLFRSH		1	0	0	0	0	1	0	x		
CBR		1	1	0	0	0	1	x	x		
MRS		1	x	0	0	0	0	1	Opcode		
ACT		1	x	0	0	1	1	x	Bank and Row		
READ		1	x	0	1	0	1	0	Bank and Column	0	Column
WRITE		1	x	0	1	0	0	mask	Bank and Column	0	Column
PALL PRE	All	1	x	0	0	1	0	x	x	1	x
	Bank								Bank	0	
NOP		1	1	1	x	x	x	x	x		
			x	0	1	1	1				

The programmable opcode for address bits MA<24:17> (or MA<24:23,14:9> or MA<24:23,13:8>, depending on the MDCNFG[STACKx] setting) used during the mode-register-set command (MRS) is exactly what is programmed in the MDMRS register. For the PXA27x processor, the burst length is always configured as “burst of four.” The only variable field in the MRS is the CAS latency unless the applications processor is operating in SA-1110 legacy mode (MDCNFG[DSA1110x] = 1) and enters alternate bus master mode. In this case, the burst length changes to “burst of one.” See Section 6.4.6.

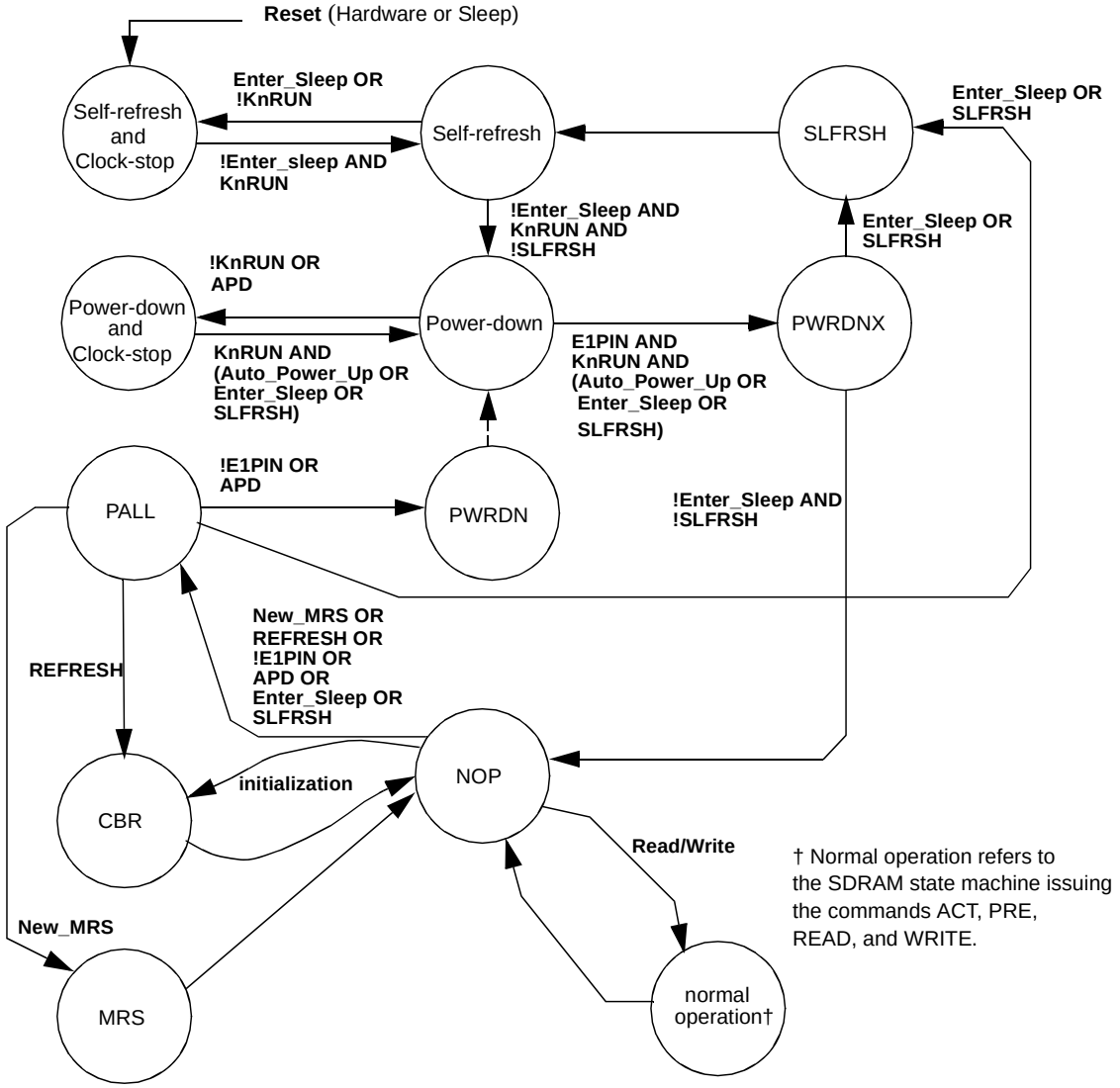
Table 6-9. SDRAM Mode Register Fields

Address Bits	Option	Value
MA<24:17> (or MA<24:23,14:9> or MA<24:23,13,8>, depending on MDCNFG[STACKx] setting)	Reserved	MDMRS[0,2]
MA<16:14> (or MA<8:6> or MA<7:5>, depending on MDCNFG[STACKx] setting)	CAS Latency = 2	0b010
	CAS Latency = 3	0b011
MA<13> (or MA<5> or MA<4>, depending on MDCNFG[STACKx] setting)	Sequential Burst	0b0
MA<12:10> (or MA<4:2> or MA<3:1>, depending on MDCNFG[STACKx] setting)	Burst Length = 4	0b010

6.4.2.5 SDRAM State Machine

Figure 6-4 shows the SDRAM controller states and transitions associated with powering on the PXA27x processor and the SDRAMs properly. Transitions are determined by the overall memory controller state and a few SDRAM power-down, self-refresh status, and control bits. The states that involve multiple SDRAM devices are self-refresh and clock-stop, self-refresh, SLFRSH, PWRDNX, power-down, power-down and clock-stop, PWRDN, PALL, and MRS. The states that involve single SDRAM partitions are ACT, PRE, READ, and WRITE. The MRS command is sent once to configure partition pair 0/1 and a separate MRS command is sent only once to configure partition pair 2/3. The auto-refresh command is issued to memory in the same partition pair at the same time. Therefore, the chip-select signals representing the partition pair are asserted at the same time when the MRS command and auto-refresh is issued from the memory controller to a specific partition pair.

Figure 6-4. SDRAM Power-On State Machine



Sleep, deep-sleep, standby, or frequency-change requests cause the SDRAM state machine to enter the self-refresh and clock-stop state. Software must then complete the appropriate reset procedure (refer to Section 6.4.9). Clearing MDREFR[E1PIN] and MDREFR[KnRUN] provides software control of the SDRAM memory system low-power modes.

Note: (1) Use these modes with extreme caution, because the resulting states prohibit automatic toggles from mode register set, read, write, and refresh commands.

The Auto_Power_Down and Auto_Power_Up transitions (made possible by setting the APD bit in MDREFR) provide a completely automatic alternative for minimizing power consumption in the SDRAM system.

(2) Some companion chips require the clock to be present at all times.

Use the following prioritization scheme for transitions out of the NOP state. If enabled with the APD bit, the Auto_Power_Down transition occurs when none of the higher priority transitions are asserted. The Auto_Power_Up transitions occur when refresh, New_MRS, or read/write is asserted during the Power_Down state.

```
Highest Priority-  "Enter Sleep"
                   "Set_SLFRSH"
                   "Clear_E1PIN"
                   "Refresh"
                   "New_MRS"
                   "Read/Write"
Lowest Priority -  "Auto_Power_Down"
```

```
KnRUN = MDREFR[K1RUN] OR MDREFR[K2RUN]
Auto_Power_Down = MDREFR[APD]
Clear_E1PIN = !(MDREFR[E1PIN])
Set_SLFRSH = MDREFR[SLFRSH]
Enter_Sleep = Sleep/Deep-Sleep/Standby/ Frequency Change Request and no more
              transactions to process
Auto_Power_Up = Read/Write is asserted during the power-down state OR New_MRS
OR Refresh OR !(MDREFR[APD])
```

6.4.3 Synchronous, Static, and Variable-Latency I/O (VLIO) Interfaces

The static memory and VLIO interfaces have six chip selects (nCS<5:0>) and 26 bits of byte address (MA<25:0>) for accesses of up to 64 Mbytes of memory in each of six banks. Alternately, a mode is available to support up to two 128-Mbyte chip selects (nCS<1:0>) with 26 bits of half-word address (MA<0>, MA<25:1>) and two 64-Mbyte chip selects (nCS<5:4>) with 25 bits of byte address (MA<25:0>). This programmable option resides in the Static Memory Configuration register SA1110[SXENX]. Each chip select is individually programmed to select one of the supported static memory types.

- Non-burst ROM (Section 6.4.3.4) or flash memory (Section 6.4.3.2) is supported on each of nCS<5:0>
- Burst ROM (Section 6.4.3.4) or flash memory with non-burst writes (Section 6.4.3.2) is supported on each of nCS<5:0>
- SRAM (Section 6.4.3.5) is supported on each of nCS<5:0>

- Variable-latency I/O (Section 6.4.3.6) is supported on each of nCS<5:0>
- Synchronous flash memory (Section 6.4.3.3) is supported on each of nCS<3:0>

The four synchronous-flash memory partitions (nCS<3:0>) are divided into two partition pairs: the 0/1 pair and the 2/3 pair. Both partitions in a pair must be identical in size and configuration. The two pairs can be different. For example, the 0/1 pair can be 66-MHz synchronous flash memory on a 32-bit data bus while the 2/3 pair is 33-MHz synchronous flash memory on a 16-bit data bus.

The VLIO interface differs from SRAM in that it allows the use of the data-ready input signal, RDY, to insert a variable number of wait states. For all static memory types, each chip select can be configured individually to a 16-bit or 32-bit wide data bus. The nOE signal is asserted on reads, the nPWE signal is asserted on writes to VLIO devices, and the nWE signal is asserted on writes to all other static devices, both synchronous and asynchronous. For SRAM and VLIO, DQM<3:0> are byte enables for both reads and writes. When the processor comes out of reset, it begins to fetch and execute instructions at address 0x00, which corresponds to memory selected by nCS<0>, which is the required location of the boot ROM. The BOOT_SEL pin determines the width of the boot memory (refer to Section 6.5.4).

6.4.3.1 Asynchronous Static Operation

The static-memory interface is comprised of six chip selects, nCS<5:0>. These six chip selects are configurable for the following:

- Non-burst ROM (Section 6.4.3.4) or flash memory (Section 6.4.3.2)
- Burst ROM (Section 6.4.3.4) or flash memory (Section 6.4.3.2)
- SRAM (Section 6.4.3.5)
- VLIO devices (Section 6.4.3.6)

The VLIO interface differs from SRAM in that it allows the use of a data-ready input signal, RDY, to insert a variable number of memory-cycle wait states. The data bus width for each chip-select region can be programmed to be 16- or 32-bit. The nCS<3:0> signals are also configurable for synchronous static memory (refer to Section 6.5.2). The following list describes the use of nOE, nWE, and nPWE:

- nOE is asserted for all reads.
- nWE is asserted for flash memory and SRAM writes.
- nPWE is asserted for VLIO writes.

For SRAM and VLIO implementations, DQM<3:0> are used for the write byte-enables, where DQM<3> corresponds to the MSB. The processor supplies 26 bits of byte address for access of up to 64 Mbytes per chip select. This byte address is sent out on the 26 external address pins. If the byte address is unimportant for an application, the lower bit must be truncated for 16-bit systems and the lower two bits must be truncated for 32-bit systems. For reads, the byte address bits is 0. For writes, the byte address bits are summarized in Table 6-10 and Table 6-11.

Table 6-10. 32-Bit Byte Address Bits MA<1:0> for Writes Based on DQM<3:0>

Transaction	DQM<3:0>	MA<1:0>
Word	0b0000	0b00
Byte 0	0b1110	0b00
Byte 1	0b1101	0b01
Byte 2	0b1011	0b10
Byte 3	0b0111	0b11
Lower half word	0b1100	0b00
Upper half word	0b0011	0b10

Table 6-11. 16-Bit Byte Address Bit MA<0> for Writes Based on DQM<1:0>

Transaction	DQM<1:0>	MA<0>
Half word	0b00	0b0
Byte 0	0b10	0b0
Byte 1	0b01	0b1

The MSCx[RTx] fields specify the type of memory:

- Non-burst ROM or flash memory
- SRAM
- VLIO
- Burst-of-four ROM or flash memory
- Burst-of-eight ROM or flash memory

The MSCx[RBWx] fields specify the bus width for the memory space selected by nCS<5:0>. If a 16-bit bus width is specified, transactions occur across data pins MD<15:0>. Use the BOOT_SEL pin or SXCNFG register to configure nCS<3:0> for synchronous static memory.

6.4.3.2 Asynchronous Flash Memory Interface

The MSCx[RDFx] bit fields define the latency for each read access of non-burst flash memory or the first read access of burst flash memory. The same bit field also controls the nWE de-assertion time during a write cycle to flash memory. The MSCx[RDN] field controls subsequent read access times to burst flash memory. The MSCx[RRR] bit field calculates the minimum period from the nCS signal de-assertion following a read or write and before the start of the read from a different memory.

The following requirements apply to reads from flash memory:

- Because flash memory defaults to read-array mode, burst reads from it are permitted, which allows instruction caching and burst reads (DMA and USB host) from flash memory.
- Some areas of flash memory might not permit burst reads. When attempting to read from these areas, do not attempt burst reads. Consult the flash-memory data sheet for more information.

- Software must partition commands and data, then write the commands to flash memory before a read. The memory controller does not insert any commands before flash-memory reads.

The following requirements apply to writes to flash memory:

- Flash memory space must be uncacheable and unbuffered.
- Burst writes to flash memory do not exist. Writes to flash memory must be exactly the width of the populated flash devices on the data bus and must be a burst length of one write (for instance, no byte writes to a 32-bit bus, no word writes to a 16-bit bus, no writes of 2 bytes to a 32-bit bus, no writes of 1 byte to a 16-bit bus). The allowable writes are 2 bytes to a 16-bit bus and 4 bytes to a 32-bit bus.
- For writes to flash memory, the command and data must be given to the memory controller in separate write instructions. The first instruction carries the command; the next carries the data.
- Software must partition commands and data and write them to flash memory in the appropriate sequence. The memory controller does not insert any commands before flash-memory writes.
- Because burst writes to flash memory cannot occur, the DMA controller and USB host controller must never write to flash memory. Burst writes to flash memory are not performed.

6.4.3.3 Synchronous Flash Memory

This section describes how to interface with synchronous flash memory. Synchronous flash-memory operation resets to asynchronous mode (page mode for reads and asynchronous single-word writes). The only way the system can enter synchronous mode (burst-timing synchronous reads and asynchronous single-word writes) is through the Read Configuration register (RCR). Therefore, at boot time, synchronous flash memory operates the same as asynchronous boot ROM (see Section 6.5.4).

Table 6-12 shows sample programming values for the RCR Synchronous Flash Memory register to ensure proper operation of synchronous flash memory.

Use the values in Table 6-12 as a reference only. Consult the data sheet for the actual part being used. Determine the frequency-configuration code based on the CLK-to-output delay, the CLK period, and the nADV-to-output delay timing parameters for the flash device.

Table 6-12. Sample Read Programming Values for Synchronous Flash Memory (Sheet 1 of 2)

Bits	Field Name	Value to Program†
2:0	BURST LENGTH	0b010 = 8-word burst
5:3	reserved	0b000
6	CLOCK CONFIGURATION	0b1 = Use rising edge of clock
7	BURST SEQUENCE	0b1 = Linear burst order (Intel burst order is not supported)
8	WAIT CONFIGURATION	Not applicable—The processor ignores nWAIT from the flash device.
9	DATA OUTPUT CONFIGURATION	0b0 = Hold data for one clock
10	reserved	0b0
† for configuration register		

Table 6-12. Sample Read Programming Values for Synchronous Flash Memory (Sheet 2 of 2)

Bits	Field Name	Value to Program†
13:11	FREQUENCY CONFIGURATION	0b010 -> Code 2 (CAS latency 3) 0b011 -> Code 3 (CAS latency 4) 0b100 -> Code 4 (CAS latency 5) 0b101 -> Code 5 (CAS latency 6) 0b110 -> Code 6 (CAS latency 7) Choose this value based on the “AC Characteristics—Read-Only Operation” section of the flash-memory device data sheet.
14	reserved	0b0
15	READ MODE	0b0 = Synchronous operation 0b1 = Asynchronous operation
† for configuration register		

6.4.3.4 ROM Interface

The processor provides programmable timing for both burst and non-burst ROMs. The value of MSCx[RDF] defines the latency (in memory clock cycles) for the first and all subsequent data beats from non-burst ROMs and the first data beat from a burst ROM. The value of MSCx[RDN] defines the latency for the burst data beats after the first for burst ROMs. Specifying the MSCx[RRR] value allows a delay on the next access to a different memory space to allow time for the current ROM to three-state the data bus. MSCx[RRR] must be programmed with the maximum T_{OFF} value divided by two, as specified by the ROM manufacturer.

MSC0<15:0> is selected when the address space corresponding to nCS<0> is accessed.

6.4.3.5 SRAM Interface Overview

The processor provides a 16- or 32-bit synchronous SRAM interface that uses the DQM pins for byte enables on writes. Bits nCS<5:0> select the SRAM bank to be used. nOE is asserted on reads and nWE is asserted on writes. Address bits MA<25:0> allow up to 64 Mbytes of SRAM per bank to be addressed.

The RDF fields in the MSCx registers define the latencies for a read access. The MSCx[RDN] field controls the nWE low time during a write cycle. MSCx[RRR] is defined as the minimum time from nCS de-assertion to the beginning of a read or write access of any memory bank

Any DMA mode that does not increment the address is not supported for SRAM reads or writes. DCMDx[INCSRCADDR] and DCMDx[INCTRGADDR] clear cause the address not to be incremented. This DMA mode is not supported for SRAM. The only valid memory types for this mode are VLIO and PC Card/CompactFlash devices. For more information, see Table 5-14, “DCMD0–31 Bit Definitions” on page 5-35.

6.4.3.6 Variable-Latency I/O Interface Overview

When a companion chip is used as a VLIO device, its functionality is similar to that of an SRAM with the additional ability to insert a variable number of wait states through the RDY pin. VLIO can be used in the memory space for any of the six static memory locations (nCS<5:0>) by programming the corresponding MSCx[RTx] to 0b100.

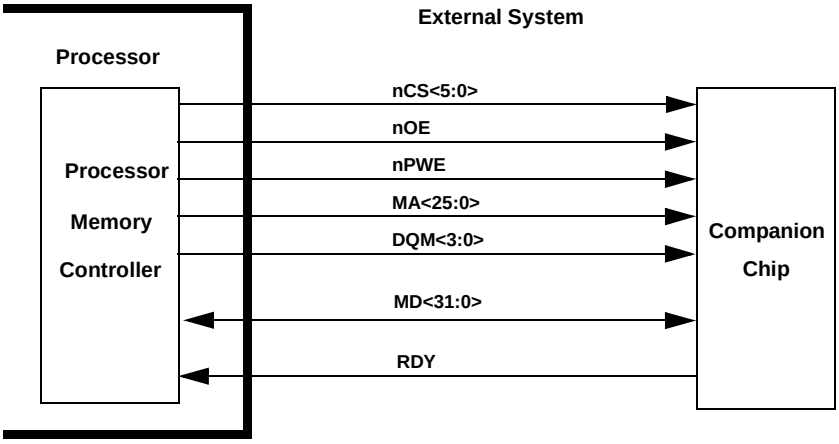
VLIO read accesses differ from SRAM read accesses in that nOE toggles for each beat of a burst. The first nOE assertion occurs two CLK_MEM cycles after the chip select, nCSx, is asserted. For VLIO writes, nPWE is used instead of nWE, which allows SDRAM refreshes to execute while performing the VLIO transfers.

For both reads and writes from and to VLIO, clearing DCMDx[INCSRCADDR] and DCMDx[INCTRGADDR] causes the source and target addresses not to be incremented to the VLIO interface, which allows port-type VLIO chips to interface with the processor. The only valid memory types for this DMA mode are VLIO and PC Card/CompactFlash devices. See Table 5-14, "DCMD0–31 Bit Definitions" on page 5-35 for information.

For writes to VLIO, if all byte enables are turned off (masking out the data, DQM = 0b1111), then the write enable is suppressed (nPWE = 1) for this write-beat to VLIO. This suppression can cause a period when nCS is asserted but neither nOE nor nPWE is asserted, which would happen if there is a write of one beat to VLIO with all byte enables turned off. In this case, the memory controller ignores the RDY signal. The RDY signal must not be asserted late if it is to still be asserted, which could interfere with any following transfers. If the VLIO device does not see an nOE or a nPWE, it must not change the state of RDY, keeping it either asserted or de-asserted.

With the exception of the case above, and when entering a frequency change, the memory controller indefinitely waits for the RDY signal to be asserted, which can hang the system if the external VLIO is not responding. To prevent indefinite hangs, set the watchdog timer when starting a VLIO transfer; a watchdog reset occurs if no response is received from the VLIO device.

Figure 6-5. Variable-Latency I/O Diagram



6.4.4 PC Card and CompactFlash Interface

The PC Card interface conforms to the *PC Card Standard, Volume 2, Electrical Specification, Version 1.4* and *CF+ and CompactFlash Specification Version 1.4.* The PC Card and CompactFlash interfaces provide control signals to support one or two PC or CompactFlash card slots.

The PC Card interface uses the following signals:

- Address line (MA<25:0>)
- Data lines (MD<15:0>)
- nPREG is actually address bit MA<26> and selects register space (I/O or attribute memory) when asserted versus command memory space when de-asserted
- nPOE and nPWE for memory and attribute reads and writes
- nPIOR and nPIOW control I/O reads and writes
- nIOIS16 for 16-bit I/O access—The system generates 8-bit references to the even and odd byte of the 16-bit port being accessed.
- nPWAIT asserted by the PC Card to generate read and write wait states delaying completion of the memory access or I/O access cycle then in progress
- nPCE<2 >and nPCE<1> are byte-select high and low, respectively, of a 16-bit data bus.
- PSKTSEL selects between two PC Card slots.

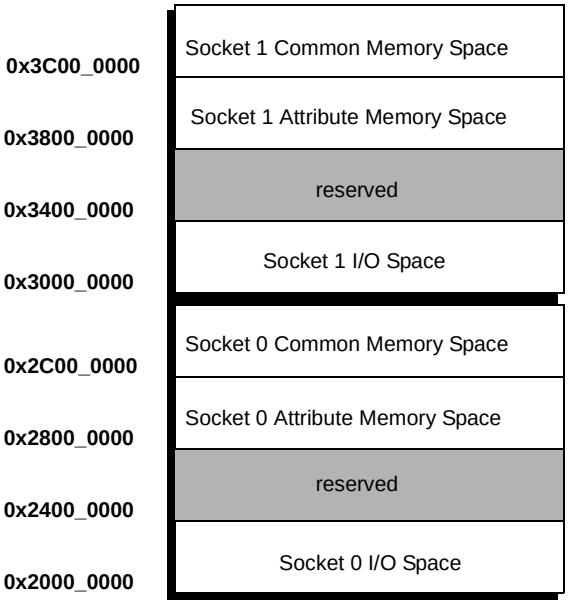
The nPIOIS16 signal is used for I/O transfers from I/O cards and indicates whether the transfer is either 8 or 16 bit. When a 16-bit I/O transfer is attempted from a 16-bit I/O card, the nPIOIS16 signal must be asserted. In all other cases the signal must be negated. If the nPIOIS16 signal is not asserted when a 16-bit I/O transfer is attempted, two separate 8-bit transfers occur to access the 16-bit even and odd bytes.

Any combination of PC Card and CompactFlash can be used for the two PC Card sockets. The PC Card interface supports 8- and 16-bit peripherals and handles common memory, I/O, and attribute memory accesses.

6.4.4.1 Overview

The PC Card interface provides control for two PC Cards with a PSKTSEL pin that differentiates between the two available sockets. The interface supports 8- and 16-bit peripherals and handles common memory, I/O, and attribute memory accesses. The duration of each access is based on values unique to each address space that are programmed by fields in the MCMEMx, MCATTx, and MCIOx registers. Figure 6-6 shows the memory map for the PC Card space.

Figure 6-6. PC Card Memory Map



The PC Card memory map space is divided into eight partitions, four for each card slot. The four partitions for each card slot are common memory, I/O, attribute memory, and a reserved space. Each partition starts on a 64-Mbyte boundary.

For accesses to common, attribute, and I/O memory space, the MA<25:0>, nPREG, and PSKTSEL control signals are driven at the same time. The processor uses nPCE<2> to indicate to the expansion device that the upper half of the data bus, MD<15:8>, is to be used for the transfer. nPCE<1> indicates that the lower half of the data bus, MD<7:0>, is to be used.

For common memory and attribute memory space accesses, the nPCE<1> and nPCE<2> signals share the same timing parameters as the MA<25:0> signals. All even bytes are transferred across the lower byte lane, MD<7:0>, with nPCE<1> asserted. During a read or write transfer, either the nPOE or nPWE control signal is asserted, respectively.

For attribute memory-space accesses, only even bytes are valid data. The card ignores nPCE<2> and the upper byte lane (odd byte), MD<15:8> and looks at nPCE<1> and the lower byte lane (even byte), MD<7:0> only. For this reason, no burst transfers can be performed to card-attribute memory space (DMA, USB host, and LCD).

For I/O space accesses, the value of nPCE<2:1> depends on the value of nIOIS16 and is asserted after the nIOIS16 signal have been sampled. The nIOIS16 input signal determines the bus width of the transfer (8 or 16 bits) for I/O accesses only. After the address is placed on the bus, an I/O device must respond with nIOIS16 to indicate it is performing the transfer in a single 16-bit transfer. If nIOIS16 is not asserted, the address is assumed to be two 8-bit registers and the transfer is completed as two 8-bit transfers on the low byte lane.

- MD<7:0>—nPCE<2> de-asserted and nPCE<1> asserted
- MA<0> = 0b0—For the first 8-bit transfer (even byte)
- MA<0> = 0b1—For the second 8-bit transfer (odd byte)

During I/O read or write accesses, the nPIOR or nPIOW control signal is asserted, respectively.

For common memory and I/O space accesses, the even byte transfers across MD<7:0> and an odd byte transfers across MD<15:8> when nPE<2> and nPE<1> are asserted. When nPCE<2> is de-asserted and nPCE<1> is asserted, MA<0> determines whether the byte being transferred across the lower byte lane is even (MA<0> = 0b0) or odd (MA<0> = 0b1). nPCE<2> is never asserted when nPCE<1> is de-asserted. Therefore, the PC Card controller does not allow odd-byte accesses on the MD<15:8> byte lane as noted in the PC Card standard.

For both reads and writes from and to PC Card/CompactFlash devices, clearing DCMDx[INCSRCADDR] and DCMDx[INCTRGADDR] causes the source and target addresses not to be incremented to the card device interface. The only valid memory types for this DMA mode are VLIO and PC Card/CompactFlash devices. For information, see Table 5-14, “DCMD0–31 Bit Definitions” on page 5-35.

When common memory is accessed, either the MCMEM0 or MCMEM1 register is used, depending on whether PC Card socket 0 or 1 is addressed. Use MCIO0 and MCIO1 for I/O accesses and MCATT0 and MCATT1 to access attribute memory.

The interface waits for a minimum amount of time (ASST_WAIT) before it checks the value of the nPWAIT signal. If nPWAIT is asserted, the interface continues to wait for a variable number of wait states until nPWAIT is de-asserted. After nPWAIT is de-asserted, the command continues to be asserted for a fixed amount of time (ASST_HOLD).

Note: The memory controller waits indefinitely for nPWAIT to be de-asserted, which can lock up the system if the card memory is not responding. To prevent indefinite hangs, set the watchdog timer when starting a PC Card transfer; the watchdog reset occurs if no response is received from the PC Card device.

Table 6-13. Possible Common Memory Space Write Commands

nPCE<2>	nPCE<1>	MA<0>	nPOE	nPWE	MD<15:8>	MD<7:0>
0	0	0	1	0	odd byte	even byte
1	0	0	1	0	ignore	even byte
1	0	1	1	0	ignore	odd byte

Table 6-14. Possible Common Memory Space Read Commands

nPCE2	nPCE1	MA<0>	nPOE	nPWE	MD<15:8>	MD<7:0>
0	0	0	0	1	odd byte	even byte
1	0	0	0	1	ignore	even byte
1	0	1	0	1	ignore	odd byte

Table 6-15. Possible Attribute Memory Space Write Commands

nPCE2	nPCE1	MA<0>	nPOE	nPWE	MD<15:8>	MD<7:0>
0	0	0	1	0	ignore	even byte
1	0	0	1	0	ignore	even byte
1	0	1	1	0	ignore	ignore

Table 6-16. Possible Attribute Memory Space Read Commands

nPCE2	nPCE1	MA<0>	nPOE	nPWE	MD<15:8>	MD<7:0>
0	0	0	0	1	ignore	even byte
1	0	0	0	1	ignore	even byte
1	0	1	0	1	ignore	ignore

Table 6-17. Possible 16-Bit I/O Space Write Commands (nIOIS16 = 0)

nPCE2	nPCE1	MA<0>	nPIOR	nPIOW	MD<15:8>	MD<7:0>
0	0	0	1	0	odd byte	even byte
1	0	0	1	0	ignore	even byte
1	0	1	1	0	ignore	odd byte

Table 6-18. Possible 16-Bit I/O Space Read Commands (nIOIS16 = 0)

nPCE2	nPCE1	MA<0>	nPIOR	nPIOW	MD<15:8>	MD<7:0>
0	0	0	0	1	odd byte	even byte
1	0	0	0	1	ignore	even byte
1	0	1	0	1	ignore	odd byte

Table 6-19. Possible 8-Bit I/O Space Write Commands (nIOIS16 = 1)

nPCE2	nPCE1	MA<0>	nPIOR	nPIOW	MD<15:8>	MD<7:0>
1	0	0	1	0	ignore	even byte
1	0	1	1	0	ignore	odd byte

Table 6-20. Possible 8-Bit I/O Space Read Commands (nIOIS16 = 1)

nPCE2	nPCE1	MA<0>	nPIOR	nPIOW	MD<15:8>	MD<7:0>
1	0	0	0	1	ignore	even byte
1	0	1	0	1	ignore	odd byte

6.4.5 Types and Sizes of Memory Accesses

The PXA27x processor performs memory accesses for the following operations:

- Unbuffered read/write
- Cache line copy-back
- Read-lock-write sequence
- Buffered write
- Internal DMA read/write
- Line fetch
- LCD read
- USB host read/write
- External fly-by DMA read/write transfer (external SDRAM addresses only)
- Quick capture interface
- SRAM read/write

Note: On a 16-bit data bus, each full-word access becomes a two-half-word burst, with address bit 1 always starting at MA<0>. Each write access to flash memory space must occur in a single non-burst operation, regardless of the bus size.

6.4.5.1 Reads and Writes

DQM<3:0> are data-masking bits. When set, each bit masks out the corresponding byte on the MD<31:0> bus. When cleared, the corresponding bit does not mask out the associated byte of data on the MD<31:0> bus.

- DQM<3> corresponds to MD<31:24>
- DQM<2> corresponds to MD<23:16>
- DQM<1> corresponds to MD<15:8>
- DQM<0> corresponds to MD<7:0>

For writes to SDRAM, SRAM, or VLIO memory spaces, the DQM<3:0> lines enable the corresponding byte of the data bus. Flash memory space stores must be exactly the width of the flash data bus, either 16 or 32 bits. See Section 6.4.4 for more information.

For reads from SDRAM, DQM<3:0> are de-asserted to avoid masking out all the data. For reads from SRAM or VLIO, the DQM signals can be configured by enabling the bits in the SA1110 Compatibility register. See Section 6.5.3.2 for more information.

6.4.5.2 Illegal Accesses and Nonexistent Memory

Hardware does not detect accesses to or from nonexistent memory. Reads return indeterminate values if no memory is selected on a read.

If a memory device occupies only a portion of all allocated space in a memory partition, reads and writes of the unoccupied portion are processed as if the memory occupied the entire allocation of the memory partition (applicable to any memory type).

A single 32-bit word (or 16-bit half word if the data bus width is defined as 16 bits) access of a disabled SDRAM partition (MDCNFG[DEx] cleared) causes an auto-refresh (CBR) cycle for all four partitions. The hardware initialization procedure uses this technique. Reads return indeterminate values. Reads and writes are not executed on the external memory bus.

Illegal accesses result from the following:

- Burst access of a disabled SDRAM partition
- Burst writes to flash-memory/ROM space
- Bursts to card configuration space.

Single-beat writes to ROM are allowed, as no distinction is made between ROM and flash memory. A *single beat* is defined as one cycle of data on the external MD bus. Thus, only those write accesses that result in more than one beat on the external memory bus to ROM or flash memory are aborted. Partial-width accesses (for example, the lower two bytes of a 32-bit bus) are not aborted.

Accesses of reserved memory-controller register space cause indeterminate behavior.

6.4.6 Alternate Bus Master Mode

The PXA27x processor supports an alternate master on the SDRAM memory bus. The alternate master gains control of the bus through a hardware handshake performed through MBREQ and MBGNT, which are invoked through the alternate functions on GPIO pins (see Chapter 24, “General-Purpose I/O Controller”) . To take control of the memory bus, the alternate master asserts MBREQ. The processor completes any memory operation that is in progress and any outstanding SDRAM refresh cycle depending on refresh bits (MDREFR[ALTREFA] and MDREFR[ALTREFB]) seen in Table 6-23. If the processor has started a swap operation, it does not begin the alternate bus-master grant sequence until the swap operation is complete. The processor then de-asserts SDCKE and three-states all memory bus pins used with SDRAM bank 0 (nSDCS<0>, MA<25:0>, nOE, nWE, nSDRAS, nSDCAS, SDCLK<1>, MD<31:0>, DQM<3:0>, and RDnWR). All other memory and PC Card pins remain driven. Then, the processor asserts MBGNT, the alternate master must start to drive all pins (including SDCLK<1>), and the processor re-asserts SDCKE.

The alternate master must ensure SDRAM integrity during this period. Proper system design must limit the period of alternate mastership to less than the refresh period or provide the alternate master with a refresh counter that allows it to perform refreshes at the proper intervals.

6.4.6.1 Alternate Bus Master Grant Sequence and Timing

The T_{MEM} unit of time is the CLK_MEM period.

1. The alternate master asserts MBREQ.
2. If MDREFR[ALTREFRB] is clear, the memory controller performs an SDRAM refresh if SDRAM clocks and clock enable are turned on. Otherwise, an SDRAM refresh is performed only if the refresh counter has reached the point where a refresh is due to be performed.
3. If MDCNFG[SA1110x] is turned on, the memory controller sends an MRS command to the SDRAMs to change the SDRAM burst length to 1 instead of 4. The burst length is changed to 1 for SA-1110 compatibility.
4. The processor de-asserts SDCKE at time (t).
5. The processor three-states SDRAM outputs at time ($t + 1*T_{MEM}$).

6. The processor asserts MBGNT at time (t + 2*T_{MEM}).
7. The alternate master drives SDRAM signals prior to time (t + 3*T_{MEM}).
8. The processor asserts SDCKE at time (t + 4*T_{MEM}).

These requirements apply during the three-state period:

- Both MBREQ and MBGNT remain high, and an external device must assume control of the three-stated pins, driving all of them even if some are not used. Floating inputs can cause excessive crossover current and erroneous SDRAM commands.
- The processor cannot perform SDRAM refresh cycles. The alternate master must ensure SDRAM integrity during this period. Proper system design must limit the period of alternate mastership to less than the refresh period or provide the alternate master with a refresh counter that allows it to perform refreshes at the proper intervals.

To give up the bus, the alternate master de-asserts MBREQ. The processor de-asserts SDCKE and MBGNT, the alternate master stops driving the SDRAM pins (including SDCLK<1>), the processor resumes driving the SDRAM pins (including SDCLK<1>), and the processor re-asserts SDCKE. The release sequence and timing are as follows:

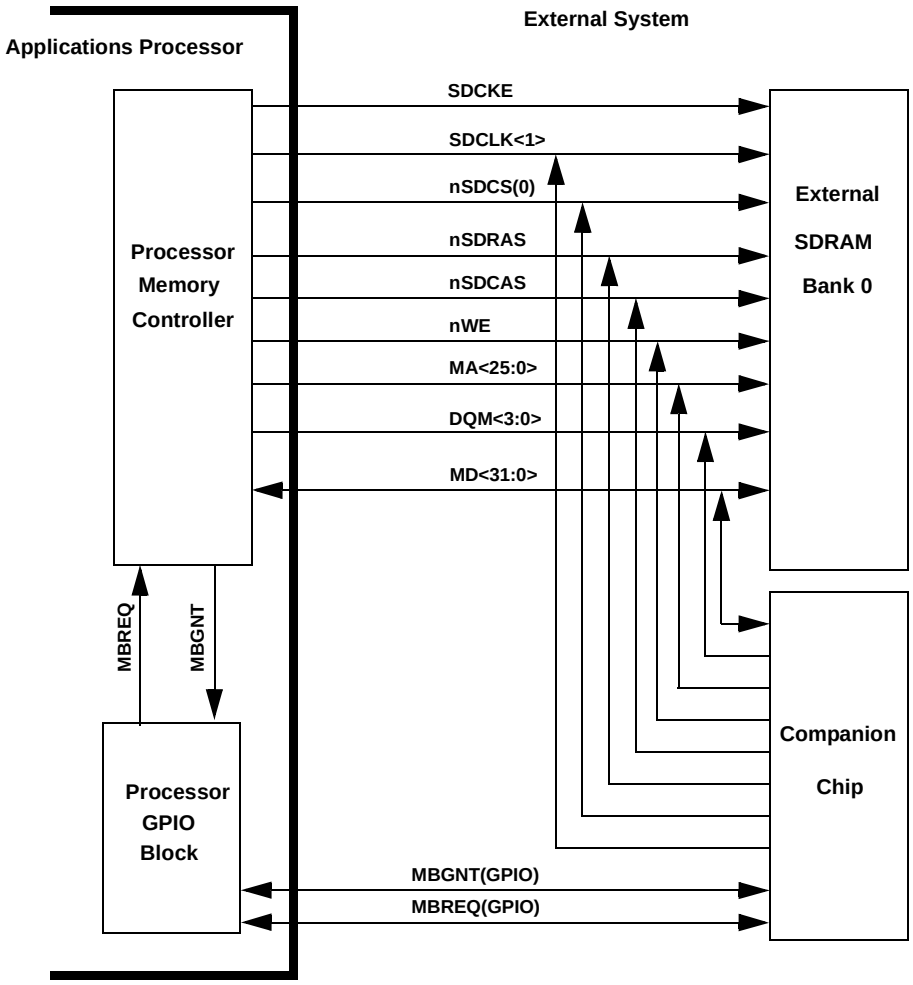
1. The alternate master de-asserts MBREQ.
2. The processor de-asserts SDCKE at time (t).
3. The processor de-asserts MBGNT at time (t + 1*T_{MEM}).
4. The alternate master three-states SDRAM outputs prior to time (t + 2*T_{MEM}).
5. The processor drives SDRAM outputs at time (t + 3*T_{MEM}).
6. The processor asserts SDCKE at time (t + 4*T_{MEM}).
7. If the MDREFR[ALTREFA] bit is clear, the memory controller performs an SDRAM refresh if SDRAM clocks and clock enables are turned on. Otherwise, an SDRAM refresh is performed only if the refresh counter has reached the point where a refresh is due to be performed.
8. The memory controller sends an MRS command to SDRAM if the MDCNFG[SA1110x] bit is turned on. This process changes the SDRAM burst length to four.

To set up alternate bus-master mode, perform the following register writes:

- In the GPIO Pin Direction register (GPDRx), configure the GPIO pin corresponding to MBGNT as an output. Configure the GPIO pin corresponding to MBREG as an input.
- In the GPIO Alternate Function register (GAFRx_x), set the bit that maps the alternate functions on the specified GPIO pins to the alternate functions corresponding to MBREQ and MBGNT.

For details on configuring GPIO, see Chapter 24, “General-Purpose I/O Controller”.

Figure 6-7. Alternate Bus-Master Mode



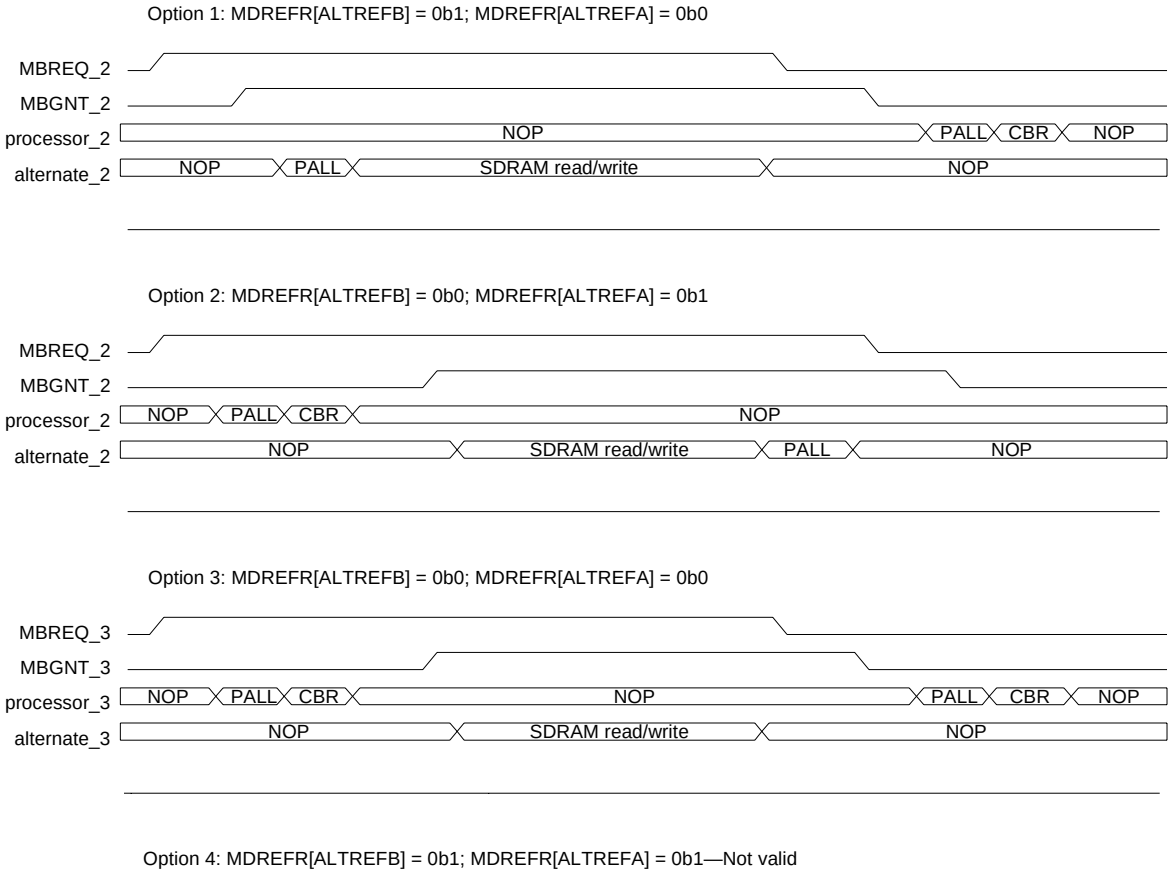
6.4.6.2 Configuring MDREFR[ALTREFA] and MDREFR[ALTREFB]

The MDREFR[ALTREFA] and MDREFR[ALTREFB] register bits must not both be set at the same time. Either bit can be set alone. When a bit is set, the alternate bus master must issue a PALL command to SDRAM partition 0.

When MDREFR[ALTREFB] is set, the memory controller does not perform a PALL and refresh before handing over the bus to the alternate bus master. Thus, the first thing the alternate master must do upon receiving the bus is a PALL command to ensure that all banks are closed in the SDRAM.

When MDREFR[ALTREFA] is set, the memory controller does not perform a PALL and refresh after the alternate bus master has completed using the bus. Therefore, the last thing the alternate master must do before de-asserting the MBREQ signal is a PALL command to the SDRAM, ensuring that all banks are closed before handing off the bus. Figure 6-8 illustrates this behavior.

Figure 6-8. Alternate Bus Master Refresh Options



Option 4: MDREFR[ALTREFB] = 0b1; MDREFR[ALTREFA] = 0b1—Not valid

6.4.7 Alternate Booting

The PXA27x processor allows two configurations for booting that are determined by the pin BOOT_SEL, which is sampled as the system comes out of reset and is described in Table 6-21. The effect of this input pin on the configuration registers at boot time is explained in Section 6.5.4.

Table 6-21. BOOT_SEL Definitions

BOOT_SEL	Type of Boot Memory	Value of MSC0[RBW0]
0	Asynchronous 32-bit ROM/flash memory	0
1	Asynchronous 16-bit ROM/flash memory	1

The MSC0[RBW0] bit field reports the state of the BOOT_SEL pin during reset.

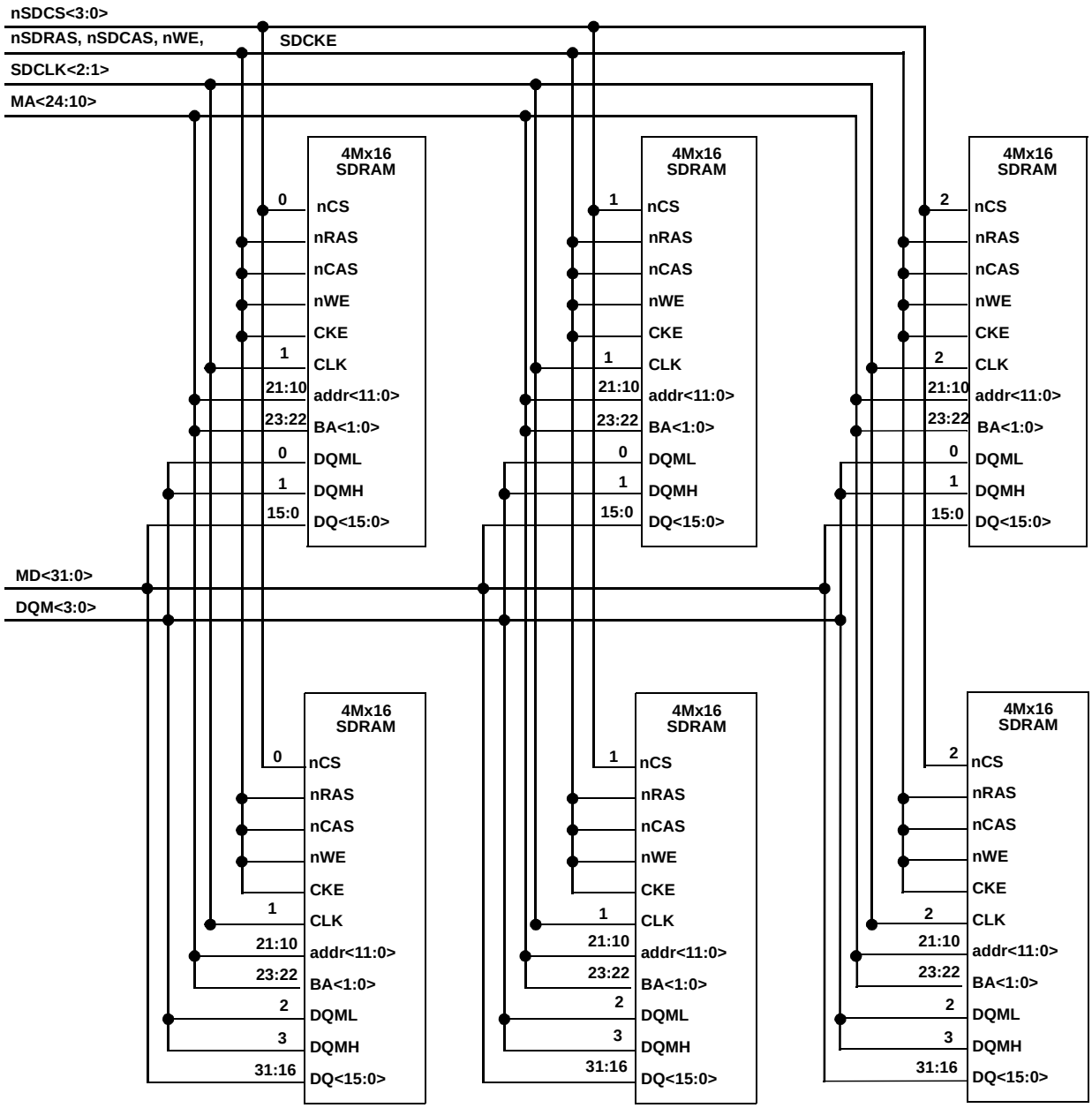
6.4.8 Memory System Examples

This section gives examples of memory configurations supported by the PXA27x processor.

6.4.8.1 SDRAM Memory System Example

Figure 6-9 shows a system using 4 x 16-bit SDRAM devices for a total of 48 Mbytes located in SDRAM partitions 0, 1, and 2.

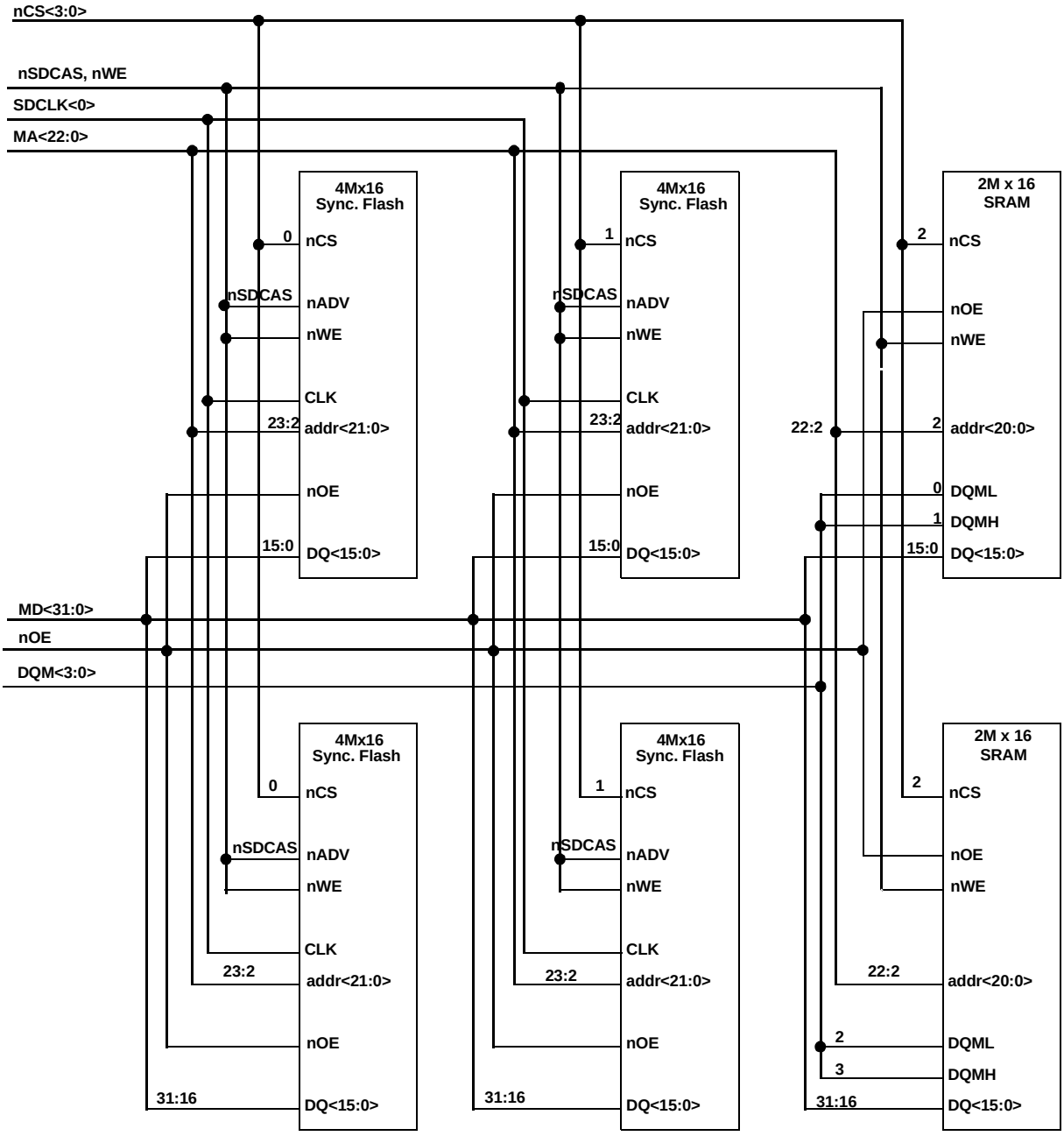
Figure 6-9. SDRAM Memory System Example



6.4.8.2 Static Memory System Example

Figure 6-10 shows an alternate memory configuration that uses 4 x 16-bit synchronous flash devices in static banks 0 and 1 and SRAM x16 devices in static bank 2.

Figure 6-10. Static Memory System Example



6.4.9 Memory Interface Reset and Initialization

On reset, the SDRAM interface is disabled. Reset values for the boot ROM are determined by BOOT_SEL (See Section 6.5.4). Boot ROM is available immediately for reading upon exit from reset, and all memory-interface control registers are available for writing.

On hardware reset, the memory pins and controller are in the state shown in Table 6-22.

Table 6-22. Memory Controller Pin Reset Values

Pin Name	Reset, Sleep, Standby, Deep-Sleep, Frequency Change, and Manual Self-Refresh Mode Values
SDCLK <3:0>	0b000
SDCKE††	0
DQM <3:0>	0b0000
nSDCS <3:2>	GPIO (memory controller drives 0b11)†
nSDCS <1:0>	0b11
nWE	1
nSDRAS	1
nSDCAS	1
nOE	1
MA <25:0>	0x000_0000
RDnWR	0
MD <31:0>	0x0000_0000
nCS <0>	1
nCS <5:1>	GPIO (memory controller drives 0b11111)
nPIOIR	GPIO (memory controller drives high)
nPIOIW	GPIO (memory controller drives high)
nPOE	GPIO (memory controller drives high)
nPWE	GPIO (memory controller drives high)

NOTES:

† This indicates that the GPIO pin, if configured for the alternate function used by the memory controller during reset, drives the represented value. See Chapter 24, “General-Purpose I/O Controller”.

†† If the SDRAMs are in self-refresh mode, they are kept there by driving SDCKE low.

6.4.10 Hardware, Watchdog, or Sleep/Deep-Sleep/Standby Reset Operation

After reset is released, software must follow this startup procedure to ensure proper operation. When MDREFR is written, a valid refresh interval value (MDREFR[DRI]) must be written (not all zeros).

1. On hardware reset, complete a power-on wait period (typically 100–200 µs) to allow the internal clocks (which generate SDCLK) to stabilize. MDREFR[K0RUN] can be enabled at this time for synchronous flash memory. Allowed writes are shown below. Refer to the *Intel® PXA27x Processor Family EMTS* for timing details.
   - MSC<0>, MSC<1>, MSC<2>, SA-1110 (order is not important).
   - MECR, MCMEM<0>, MCMEM<1>, MCATT<0>, MCATT<1>, MCIO<0>, and MCIO<1> (order is not important).
   - FLYCNFG.
   - Reset the system appropriately. Configure, but do not enable, each SDRAM partition pair by clearing the enable bits MDCNFG[DEx] when writing to the MDCNFG register.
   - Set MDREFR[K0RUN]. Properly configure MDREFR[K0DB2] and MDREFR[K0DB4]. Retain the current values of MDREFR[APD] (clear) and MDREFR[SLFRSH] (set). MDREFR[DRI] must contain a valid value (not all 0s). If required, MDREFR[KxFREE] can be de-asserted.
2. In systems that contain synchronous flash memory, write to the SXCNFG to configure all appropriate bits, including the enables. While the synchronous flash banks are being configured, the SDRAM banks must be disabled and MDREFR[APD] must be de-asserted (auto-power-down disabled).
3. In systems that contain SDRAM, toggle the SDRAM controller through the following state sequence: self-refresh and clock-stop to self-refresh to power-down to PWRDNX to NOP. See Figure 6-4. The SDRAM clock run and enable bits, (MDREFR[K1RUN] and MDREFR[K2RUN] and MDREFR[E1PIN]), are described in Section 6.5.1.3. MDREFR[SLFRSH] must not be set.
   a. Set MDREFR[K1RUN], MDREFR[K2RUN] (self-refresh and clock-stop through self-refresh). MDREFR[K1DB2] and MDREFR[K2DB2] must be configured appropriately.
   b. Clear MDREFR[SLFRSH] (self-refresh through power down)
   c. Set MDREFR[E1PIN] (power down through PWRDNX)
   d. No write required for this state transition (PWRDNX through NOP)
4. Appropriately configure, but do not enable, each SDRAM partition pair.

   SDRAM partitions are disabled by keeping the MDCNFG[DEx] bits clear.
5. For systems that contain SDRAM, wait the NOP power-up waiting period required by the SDRAMs (normally 100-200 µsec) to ensure the SDRAMs receive a stable clock with a NOP condition.
6. Ensure the XScale core memory-management data cache (Coprocessor 15, Register 1, bit 2) is disabled. If this bit is enabled, the refreshes triggered by the next step may not be passed properly through to the memory controller. Coprocessor 15, register 1, bit 2 must be re-enabled after the refreshes are performed if data cache is preferred.
7. On hardware reset in systems that contain SDRAM, trigger a number (the number required by the SDRAM manufacturer) of refresh cycles by attempting non-burst read or write accesses to

any disabled SDRAM bank. Each such access causes a simultaneous CBR for all four banks, which in turn causes a pass through the CBR state and a return to NOP. On the first pass, the PALL state is incurred before the CBR state. See Figure 6-4.

8. Set coprocessor 15, register 1, bit 2 if it was cleared in step 6.
9. In systems that contain SDRAM, enable SDRAM partitions by setting MDCNFG[DEx] bits.
10. In systems that contain SDRAM, write the MDMRS register to trigger an MRS command to all enabled banks of SDRAM. For each SDRAM partition pair that has one or both partitions enabled, this forces a pass through the MRS state and a return to NOP. The CAS latency is the only variable option and is derived from what was programmed into the MDCNFG[MDTC0] and MDCNFG[MDTC2] fields. The burst type and length are always programmed to sequential and four, respectively. For more information, see Section 6.4.2.5.
11. In systems that contain SDRAM or synchronous flash, optionally enable auto-power-down by setting MDREFR[APD].

6.4.11 GPIO Reset Procedure

When coming out of GPIO reset, all memory controller registers retain the values they contained before the GPIO reset and any outstanding auto-refresh commands are submitted. No memory configuration programming is required following a GPIO reset. The contents of memory are not guaranteed. Software must determine if the memory contents have been compromised.

No additional logic is required when connecting nRESET_OUT signal to the synchronous flash reset. The additional logic preventing nRESET_OUT during GPIO reset is eliminated as long as the PCFR[GPROD] bit. Setting PCFR[GPROD] prevents RESET_OUT from being asserted, thus allowing the synchronous flash to remain configured for synchronous transfers.

Upon exit from GPIO reset, a series of five refreshes is performed by the memory controller to make up for any lost refreshes during GPIO reset. This number of refreshes accounts for the longest possible GPIO reset.

Note: Software must not allow GPIO resets to occur during flash programming and erase operations. The processor could attempt to access the flash prior to completion of an erase or programming cycle.

6.5 Register Descriptions

6.5.1 Synchronous Dynamic Memory Registers

Each of the possible SDRAM portions of the processor memory map are referred to as *partitions* in this document, to distinguish them from banks internal to SDRAM devices.

6.5.1.1 SDRAM MDCNFG Register (MDCNFG)

MDCNFG, shown in Table 6-23, contains control bits for configuring the SDRAM. Both SDRAM partitions within a pair (0/1 or 2/3) must be implemented with the same type of SDRAM devices, but the two partition pairs may differ.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 6-23. MDCNFG Bit Definitions (Sheet 1 of 5)

Physical Address 0x4800_0000								MDCNFG														Memory Controller							
User Settings†			0		1												0	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1
Bit	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
	MDENX	DCACX2	reserved	DSA1110_2	SETALWAYS	DADDR2	DTC2		DNB2	DRAC2		DCAC2		DWID2	DE3	DE2	STACK1	DCACX0	STACK0	DSA1110_0	SETALWAYS	DADDR0	DTC0		DNB0	DRAC0		DCAC0		DWID0	DE1	DE0
Reset	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	MDENX	SDRAM 1 GB Memory Map Enable 0 = Use normal 256-Mbyte memory map. 1 = Use large 1-Gbyte memory map.
30	R/W	DCACX2	Extra Column Addressing Used with MDCNFG[DCAC2]. See MDCNFG[DCAC2] for usage.
29	—	—	reserved
28	R/W	DSA1110_2	SA-1110 Addressing Mode Compatibility Use SA-1110 addressing multiplexing mode for pair 2/3. Setting this bit overrides the addressing bit programmed in MDCNFG[DADDR2] For an explanation of how the address is driven onto the address bus in SA-1110 addressing mode, see Table 6-7. 0 = Use addressing mode specified in MDCNFG[DADDR2]. 1 = Override the addressing bit programmed in MDCNFG[DADDR2] and use SA-1110 addressing mode.
27	R/W	SETALWAYS	Set Reserved Bit 0 = Undetermined results 1 = Always set this bit
26	R/W	DADDR2	Alternate Addressing Mode Ignored if MDCNFG[DSA1110_2] is set. For an explanation of how the address is driven onto the address bus for the partition pair 2/3 in alternate addressing mode, see Figure 6-3. 0 = Use normal addressing mode 1 = Use alternate addressing mode

NOTE:

† The User Settings row represents the settings required for PXA271 processor configuration only. Other configurations may differ.

Table 6-23. MDCNFG Bit Definitions (Sheet 2 of 5)

Physical Address 0x4800_0000 — **MDCNFG** — **Memory Controller**

	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
User Settings†			0		1												0	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1
Bit	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
	MDENX	DCACX2	reserved	DSA1110_2	SETALWAYS	DADDR2	DTC2		DNB2	DRAC2		DCAC2		DWID2	DE3	DE2	STACK1	DCACX0	STACK0	DSA1110_0	SETALWAYS	DADDR0	DTC0		DNB0	DRAC0		DCAC0		DWID0	DE1	DE0
Reset	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
25:24	R/W	DTC2	The AC timing parameters for SDRAM partition pair 2/3 represent the number of SDCLKs (not memory clocks) for the clocks indicated below. See the SDRAM data sheet to determine the optimal timings. See the *Intel® PXA27x Processor Family EMTS* for SDRAM timing diagrams and the effects on AC timings. For all timing categories, write recovery time (tWR) is hard-coded at 2 clocks. 0b00: tRP = 2 clks, CL = 2, tRCD = 1 clks, $tRAS_{MIN}$ = 3 clks, tRC = 4 clks 0b01: tRP = 2 clks, CL = 2, tRCD = 2 clks, $tRAS_{MIN}$ = 5 clks, tRC = 8 clks 0b10: tRP = 3 clks, CL = 3, tRCD = 3 clks, $tRAS_{MIN}$ =7 clks, tRC = 10 clks 0b11: tRP = 3 clks, CL = 3, tRCD = 3 clks, $tRAS_{MIN}$ = 7 clks, tRC = 11 clks **NOTE:** The CAS latency programmed here is set to 3 if the CAS latency for SDRAM partition pair 0 (MDCNFG[DTC0] CAS latency) is programmed to 3. Otherwise, it is set as programmed here. What this means is that the CAS latency for the two SDRAM partition pairs is the same and equals the greater of the two programmed values. If either partition pair is programmed to CAS latency of 3, then both partition pairs have CAS Latency of 3; otherwise, both partition pairs have CAS latency of 2.
23	R/W	DNB2	Number of Banks in Partition Pair 2/3 0 = 2 internal SDRAM banks 1 = 4 internal SDRAM banks
22:21	R/W	DRAC2	SDRAM Row Address Bit Count for Partition Pair 2/3 0b00 = 11 row address bits 0b01 = 12 row address bits 0b10 = 13 row address bits 0b11 = reserved
20:19	R/W	DCAC2	SDRAM Column Address Bits for Partition Pair 2/3 Use in conjunction with DCACX2 (DCACX2 concatenated with DCAC2). 0b000 = 8 column address bits 0b001 = 9 column address bits 0b010 = 10 column address bits 0b011 = 11 column address bits 0b100 = 12 column address bits

NOTE:

† The User Settings row represents the settings required for PXA271 processor configuration only. Other configurations may differ.

Table 6-23. MDCNFG Bit Definitions (Sheet 3 of 5)

Physical Address 0x4800_0000 — **MDCNFG** — **Memory Controller**

	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
User Settings†			0		1												0	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1
Bit	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
	MDENX	DCACX2	reserved	DSA1110_2	SETALWAYS	DADDR2	DTC2		DNB2	DRAC2		DCAC2		DWID2	DE3	DE2	STACK1	DCACX0	STACK0	DSA1110_0	SETALWAYS	DADDR0	DTC0		DNB0	DRAC0		DCAC0		DWID0	DE1	DE0
Reset	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
18	R/W	DWID2	SDRAM Data Bus Width for Partition Pair 2/3 0 = 32 bits 1 = 16 bits
17	R/W	DE3	SDRAM Enable for Partition 3 A single (non-burst) 32-bit (or 16-bit if MDCNFG[DWIDx] = 1) access (read or write) to a disabled SDRAM partition triggers a CBR refresh cycle to all partitions. When all partitions are disabled, the refresh counter is disabled. 0 = SDRAM partition disabled 1 = SDRAM partition enabled
16	R/W	DE2	SDRAM Enable for Partition 2 A single (non-burst) 32-bit (or 16-bit if MDCNFG[DWIDx] = 1) access (read or write) to a disabled SDRAM partition triggers a CBR refresh cycle to all partitions. When all partitions are disabled, the refresh counter is disabled. 0 = SDRAM partition disabled 1 = SDRAM partition enabled
15	R/W	STACK1	Stack 1 The 2-bit STACK field consists of MDCNFG[STACK1] (MSB) and MDCNFG[STACK0] (LSB) to determine the SDRAM address-multiplexing scheme.This 2-bit setting alters the SDRAM multiplexing scheme for all SDRAM partitions as shown below. 0b00 = SDRAM address is placed on MA<24:10>. See Table 6-3, Table 6-5 and Table 6-7. 0b01 = SDRAM address is placed on MA<24:23,13:1>. Use when flash internal to the PXA271 processor has 16-bit total data bus width on partition 0. See Table 6-4 and Table 6-6. 0b10 = reserved 0b11 = reserved **NOTE:** When MDCNFG[STACKx] equals either 0b01 or 0b10, fly-by DMA transfers are not supported, and SA1110 addressing mode is not supported, DASA1110_0 must be cleared.
14	R/W	DCACX0	Extra Column Addressing Used in conjunction with MDCNFG[DCAC0]. See MDCNFG[DCAC0] for usage.
13	R/W	STACK0	Stack 0 Used in conjunction with MDCNFG[STACK1]—refer to MDCNFG[STACK0].

NOTE:

† The User Settings row represents the settings required for PXA271 processor configuration only. Other configurations may differ.

Table 6-23. MDCNFG Bit Definitions (Sheet 4 of 5)

Physical Address 0x4800_0000 — **MDCNFG** — **Memory Controller**

User Settings†			0		1												0	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1
Bit	**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**
	MDENX	DCACX2	reserved	DSA1110_2	SETALWAYS	DADDR2	DTC2		DNB2	DRAC2		DCAC2		DWID2	DE3	DE2	STACK1	DCACX0	STACK0	DSA1110_0	SETALWAYS	DADDR0	DTC0		DNB0	DRAC0		DCAC0		DWID0	DE1	DE0
Reset	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
12	R/W	DSA1110_0	SA-1110 Addressing Mode Compatibility Use SA-1110 addressing multiplexing mode for pair 0/1. Setting this bit overrides the addressing bit programmed in MDCNFG[DADDR0]. For an explanation of how the address is driven onto the address bus in SA-1110 addressing mode, see Table 6-7. 0 = Use addressing mode specified in MDCNFG[DADDR0] 1 = Use SA-1110 addressing mode
11	R/W	SETALWAYS	Set Reserved Bit 0 = undetermined results 1 = always set this bit
10	R/W	DADDR0	Alternate Addressing Mode Ignored if MDCNFG[DSA1110_0] is set. For an explanation of how the address is driven onto the address bus for the partition pair 0/1 in alternate addressing mode see Figure 6-3. 0 = Use normal addressing mode 1 = Use alternate addressing mode
9:8	R/W	DTC0	The AC timing parameters for SDRAM partition pair 0/1 represent the number of SDCLKs (not memory clocks) for the clocks indicated below. See the SDRAM data sheet to determine the optimal timings. See the *Intel® PXA27x Processor Family EMTS* for SDRAM timing diagrams and the effects on AC timings. For all timing categories, tWR is hard coded at 2 clocks. 0b00—tRP = 2 clks, CL = 2, tRCD = 1 clks, $tRAS_{MIN}$ = 3 clks, tRC = 4 clks 0b01—tRP = 2 clks, CL = 2, tRCD = 2 clks, $tRAS_{MIN}$ = 5 clks, tRC = 8 clks 0b10—tRP = 3 clks, CL = 3, tRCD = 3 clks, $tRAS_{MIN}$ =7 clks, tRC=10 clks 0b11—tRP = 3 clks, CL = 3, tRCD = 3 clks, $tRAS_{MIN}$ = 7 clks, tRC = 11 clks **NOTE:** The CAS latency programmed here is set to 3 if the CAS latency for SDRAM partition pair 2 (MDCNFG[DTC2] CAS latency) is programmed to 3. Otherwise, it is set as programmed here. This means that the CAS latency for the two SDRAM partition pairs is the same and equals the greater of the two programmed values. If either partition pair is programmed to CAS latency of 3, then both partition pairs have CAS latency of 3; otherwise, both partition pairs have CAS latency of 2.

NOTE:

† The User Settings row represents the settings required for PXA271 processor configuration only. Other configurations may differ.

Table 6-23. MDCNFG Bit Definitions (Sheet 5 of 5)

Physical Address 0x4800_0000 — **MDCNFG** — **Memory Controller**

	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
User Settings†			0		1												0	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1
Bit	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
	MDENX	DCACX2	reserved	DSA1110_2	SETALWAYS	DADDR2	DTC2		DNB2	DRAC2		DCAC2		DWID2	DE3	DE2	STACK1	DCACX0	STACK0	DSA1110_0	SETALWAYS	DADDR0	DTC0		DNB0	DRAC0		DCAC0		DWID0	DE1	DE0
Reset	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
7	R/W	DNB0	Number of Banks in Partition Pair 0/1 0 = 2 internal SDRAM banks 1 = 4 internal SDRAM banks
6:5	R/W	DRAC0	SDRAM Row Address Bit Count for Partition Pair 0/1 0b00 = 11 row address bits 0b01 = 12 row address bits 0b10 = 13 row address bits 0b11 = reserved.
4:3	R/W	DCAC0	SDRAM Column Address Bits for Partition Pair 0/1 Use in conjunction with DCACX0 (DCACX0 concatenated with DCAC0) 0b000 = 8 column address bits 0b001 = 9 column address bits 0b010 = 10 column address bits 0b011 = 11 column address bits 0b100 = 12 column address bits
2	R/W	DWID0	SDRAM Data Bus Width for Partition Pair 0/1 0 = 32 bits 1 = 16 bits
1	R/W	DE1	SDRAM Enable for Partition 1 A single (non-burst) 32-bit (or 16-bit if MDCNFG[DWIDx] = 1) access (read or write) to a disabled SDRAM partition that triggers a CBR refresh cycle to all partitions. When all partitions are disabled, the refresh counter is disabled. 0 = SDRAM partition disabled 1 = SDRAM partition enabled
0	R/W	DE0	SDRAM Enable for Partition 0 A single (non-burst) 32-bit (or 16-bit if MDCNFG[DWIDx] = 1) access (read or write) to a disabled SDRAM partition that triggers a CBR refresh cycle to all partitions. When all partitions are disabled, the refresh counter is disabled. 0 = SDRAM partition disabled 1 = SDRAM partition enabled

NOTE:

† The User Settings row represents the settings required for PXA271 processor configuration only. Other configurations may differ.

6.5.1.2 SDRAM Mode Register Set Configuration Register (MDMRS)

MDMRS, shown in Table 6-24, issues MRS commands to SDRAM (see Table 6-8). Writing to this register triggers a two-stage MRS command to external SDRAM. The first stage writes the MRS value to SDRAM partitions 0 and 1. The second stage writes the MRS value to SDRAM partitions 2 and 3. The value written to this register is placed directly on address lines MA<24:17> (or MA<24:23,14:9> or MA<24:23,13:8>, depending on the MDCNFG[STACKx] setting) during the MRS command. For MA<16:10> (or MA<8:2> or MA<7:1>, depending on the MDCNFG[STACKx] setting), values that are derived from the MDCNFG register are placed on the address bus. When setting the values to be written out on the address lines, they must be written out properly (based on the addressing mode being used). Writing to this register triggers an MRS command, but the corresponding chip-select values are asserted if the memory banks are enabled with the MDCNFG register only. To write a new MRS value to SDRAM, enable the memory with the MDCNFG register; then write to MDMRS. Use this register for the value written during the MRS command only. Programmed values in MDCNFG must be correct to ensure proper operation of the processor. Refer to Section 6.4.10 for the proper sequence to use when coming out of reset.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 6-24. MDMRS Bit Definitions (Sheet 1 of 2)

Physical Address 0x4800_0040																MDMRS												Memory Controller			
User Settings†																0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
Bit	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	MDMRS2											MDCL2				MDADD2	MDBL2			reserved	MDMRS0							MDCL0		MDADD0	MDBL0
Reset	?	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	?	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

Bits	Access	Name	Description
31	—	—	reserved
30:23	R/W	MDMRS2	The SDRAM MRS bits represent the value driven onto the SDRAM address bus from MA<24:17> (or MA<24:23,14:9> or MA<24:23,13,8>, depending on MDCNFG[STACKx] setting) during the MRS command for partition pair 2/3.
22:20	R	MDCL2	SDRAM MRS bits representing CAS Latency SDRAM partition pair 2/3 CAS latency is derived from MDCNFG[DTC2]. Writes are ignored. This bit field represents the value driven onto MA<16:14> (or MA<8:6> or MA<7:5>, depending on MDCNFG[STACKx] setting) during the MRS command.
19	R	MDADD2	SDRAM MRS bit representing Burst Type SDRAM partition pair 2/3 burst type. Fix to sequential addressing. This bit field represents the value driven onto MA<13> (or MA<5> or MA<4>, depending on MDCNFG[STACKx] setting) during the MRS command. This field is always zero.

NOTE:

† The User Settings row represents the settings required for PXA271 processor configuration only. Other configurations may differ.

Table 6-24. MDMRS Bit Definitions (Sheet 2 of 2)

Physical Address 0x4800_0040 — **MDMRS** — **Memory Controller**

	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
User Settings†																	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
Bit	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	MDMRS2											MDCL2				MDADD2	MDBL2			reserved	MDMRS0							MDCL0		MDADD0	MDBL0	
Reset	?	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	?	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

Bits	Access	Name	Description
18:16	R	MDBL2	SDRAM MRS bits representing Burst Length SDRAM partition pair 2/3 burst length. Fixed to a burst length of four. This bit field represents the value driven onto the MA<12:10> (or MA<4:2> or MA<3:1>, depending on MDCNFG[STACKx] setting) during the MRS command. This field is read as 0b010.
15	—	—	reserved
14:7	R/W	MDMRS0	The SDRAM MRS bits represent the value driven onto the SDRAM address bus from MA<24:17> (or MA<24:23,14:9> or MA<24:23,13,8>, depending on MDCNFG[STACKx] setting) during the MRS command for partition pair 0/1.
6:4	R	MDCL0	SDRAM MRS bits representing CAS Latency SDRAM partition pair 0/1 CAS latency is derived from MDCNFG[DTC0]. Writes are ignored. This bit field represents the value driven onto MA<16:14> (or MA<8:6> or MA<7:5>, depending on MDCNFG[STACKx] setting) during the MRS command.
3	R	MDADD0	SDRAM MRS bit representing Burst Type SDRAM partition pair 0/1 burst type. Fix to sequential addressing. This bit field represents the value driven onto MA<13> (or MA<5> or MA<4>, depending on MDCNFG[STACKx] setting) during the MRS command. This field is always zero.
2:0	R	MDBL0	SDRAM MRS bits representing Burst Length SDRAM partition pair 0/1 burst length. Fixed to a burst length of 4. This bit field represents the value driven onto MA<12:10> (or MA<4:2> or MA<3:1>, depending on MDCNFG[STACKx] setting) during the MRS command. This field is read as 0b010.

NOTE:

† The User Settings row represents the settings required for PXA271 processor configuration only. Other configurations may differ.

6.5.1.3 Special Low-Power SDRAM Mode Register Set Configuration Register (MDMRSLP)

MDMRSLP, shown in Table 6-25, issues special low-power MRS commands to SDRAM (see Table 6-8). Writing to this register triggers a two-stage MRS command to external SDRAM. The first stage writes the low-power MRS value to SDRAM partitions 0 and 1. The second stage writes the low-power MRS value to SDRAM partitions 2 and 3. The value written in this register is placed directly on address lines MA<24:10> (or MA<24:23,14:2> or MA<24:23,13:1>, depending on the MDCNFG[STACKx] setting) during the MRS command. The values to be written out on the address lines must be written out properly, based on the addressing mode being used. Although writing to this register triggers an MRS command, the corresponding chip-select values are asserted only if the memory banks are enabled with the MDCNFG register and the corresponding MDMRSLP[MDLPENx] bit is set. To write a new low- power MRS value to SDRAM, enable the memory with the MDCNFG register, then write to MDMRSLP with the enable bits set.

Use this register for the value written during the MRS command only. Programmed values in the MDCNFG register must be correct to ensure proper operation of the processor. See Section 6.4.10 for the proper sequence to use when coming out of reset.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 6-25. MDMRSLP Bit Definitions

Physical Address 0x4800_0058 — **MDMRSLP** — **Memory Controller**

	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
User Settings†																	1	0	0	0	0	0	0	0	0	0	Max. Ambient Temp††		Partial Array Refresh††			0
Bit	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
	MDLPEN2	MDMRSLP2															MDLPEN0	MDMRSLP0														
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

Bits	Access	Name	Description
31	R/W	MDLPEN2	Low-Power Enable for Partition Pair 2/3 Enable bit for low-power MRS value for partition pair 2/3.
30:16	R/W	MDMRSLP2	Low-Power MRS Value for Partition Pair 2/3 Low-power MRS value to be written to SDRAM for partition pair 2/3.
15	R/W	MDLPEN0	Low-Power Enable for Partition Pair 0/1 Enable bit for low-power MRS value for partition pair 0/1.
14:0	R/W	MDMRSLP0	Low-Power MRS Value for Partition Pair 0/1 Low-power MRS value to be written to SDRAM for partition pair 0/1.

NOTES:

† The User Settings row represents the settings required for PXA271 processor configuration only. Other configurations may differ.

†† See Intel StrataFlash® Wireless Memory (L18/L30 SCSP) 256-Mbit LX Device with 256-Mbit LPSDRAM Data Sheet for detailed information on Temperature Compensated Self Refresh (TCSR), and Partial Array Self Refresh (PASR).

6.5.1.4 SDRAM Memory Device Refresh Register (MDREFR)

MDREFR, shown in Table 6-26, contains control bits for refresh of both SDRAM partition pairs. MDREFR also contains control/status bits for SDRAM self-refresh, SDRAM/synchronous flash memory clock divisors, SDRAM/synchronous flash memory clocks running and SDRAM clock-enable pin states. Independent control/status is provided for each of the clock pins (SDCLK<2:0>) and clock-enable pin (SDCKE). SDCLK<3> responds identically to the SDCLK<0> control/status bit field.

To Change CLK_MEM from 208 MHz to 104 MHz while SDCLK<2> or SDCLK<1> is currently at 104MHz:

1. Update refresh cycles for a slower CLK_MEM (decrease MDREFR[DRI] to account for CLK_MEM = 104 MHz)
2. Issue frequency change to lower CLK_MEM from 208 MHz to 104 MHz (now SDCLK<1> = 52 MHz and SDCLK<2> = 52 MHz)
3. Change SDCLK<1> or SDCLK<2> to 104 MHz. (Clear MDREFR[K1DB2], which makes SDCLK<1> = 104 MHz, or clear MDREFR[K2DB2], which makes SDCLK<2> = 104 MHz)
4. Update refresh cycles setting and SDRAM timings setting for optimized clock (MDREFR[DRI] and MDCNFG[DTCx]). If the clock frequency is changed, the MDREFR[DRI] register field must be rewritten, even if the value has not changed. This results in a refresh and resets the refresh counter to the refresh interval.

To Change CLK_MEM from 104 MHz to 208 MHz while SDCLK<2> or SDCLK<1> is currently at 104 MHz:

1. Change SDCLK<1> or SDCLK<2> to 52 MHz. (Set MDREFR[K1DB2], which makes SDCLK<1> = 52 MHz, or set MDREFR[K2DB2], which makes SDCLK<2> = 52 MHz)
2. Issue frequency change to raise CLK_MEM from 104 MHz to 208 MHz (now SDCLK<1> = 104 MHz and SDCLK<2> = 104 MHz)
3. Optimize refresh cycles for faster CLK_MEM (increase MDREFR[DRI] to account for CLK_MEM = 208 MHz)

Note: The clock run bits (K0RUN, K1RUN, and K2RUN) and clock-enable bit (E1PIN) provide software control of SDRAM and synchronous flash-memory low-power modes. Use these bits with extreme caution because the corresponding memory is inaccessible when any of these bits are cleared.

Auto-power down, enabled by the APD bit, is an automatic mechanism for minimizing power consumption in the processor SDCLK pin drivers or the SDRAM/synchronous flash devices. APD is typically set. A latency penalty of one memory cycle is incurred when restarting SDCLK and SDCKE between non-consecutive SDRAM/synchronous flash-memory transfers.

Refresh rules:

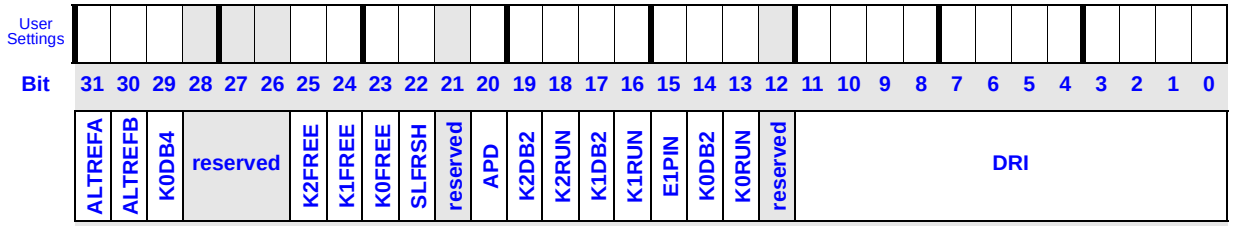
- When the refresh counter is cleared, no refreshes are sent to the SDRAMs.
- If a single transaction to a disabled SDRAM partition is requested, a refresh to all four partitions is performed.
- If all four SDRAM partitions are disabled, the refresh counter is disabled and refreshes are performed when a single transaction to a disabled SDRAM partition is requested only.

- If the clock frequency is changed, the MDREFR[DRI] register field must be rewritten, even if the value has not changed. This rewrite results in a refresh and resets the refresh counter to the refresh interval.
- Upon exit from GPIO reset, a series of five refreshes is performed by the memory controller to make up for any lost refreshes during GPIO reset. This number of refreshes accounts for the longest possible GPIO reset.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 6-26. MDREFR Bit Definitions (Sheet 1 of 5)

Physical Address 0x4800_0004 — **MDREFR** — **Memory Controller**



Bits	Access	Name	Description
31	R/W	ALTREFA	Exiting Alternate Bus Master Mode Refresh Control 0 = SDRAM refresh is always performed after exiting alternate bus master mode. 1 = SDRAM refresh is *not* performed after exiting alternate bus master mode unless the SDRAM refresh counter has timed out. MDREFR[ALTREFB] must be clear. The alternate master must perform a PALL before releasing the external bus. See Section 6.4.6.2 for more information.
30	R/W	ALTREFB	Entering Alternate Bus Master Mode Refresh Control 0 = SDRAM refresh is always performed after entering alternate bus master mode. 1 = SDRAM refresh is *not* performed after entering alternate bus master mode unless the SDRAM refresh counter has timed out. MDREFR[ALTREFA] must be clear. The alternate master must first perform a PALL after receiving the bus grant. See Section 6.4.6 for more information.
29	R/W	K0DB4	Synchronous Static Memory Clock Pin 0 (SDCLK<0> and SDCLK<3>) Divide-by-4 Control/Status When set, SDCLK<0> and SDCLK<3> runs at one-fourth of the memory clock frequency, regardless of the value of MDREFR[K0DB2]. When clear, the SDCLK<0> and SDCLK<3> speed depends on the value programmed in MDREFR[K0DB2]. 0 = Use the value programmed in MDREFR[K0DB2] to generate SDCLK<0> and SDCLK<3>. 1 = Divide CLK_MEM by four to generate SDCLK<0> and SDCLK<3>. K0DB2 is ignored.
28	—	—	reserved
27	—	—	reserved
26	—	—	reserved

Table 6-26. MDREFR Bit Definitions (Sheet 2 of 5)

Physical Address 0x4800_0004 — MDREFR — Memory Controller

Bit	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
User Settings																																
	ALTREFA	ALTREFB	K0DB4	reserved			K2FREE	K1FREE	K0FREE	SLFRSH	reserved	APD	K2DB2	K2RUN	K1DB2	K1RUN	E1PIN	K0DB2	K0RUN	reserved	DRI											
Reset	0	0	1	0	0	0	1	1	1	1	?	0	1	0	1	0	0	1	0	?	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Access	Name	Description
25	R/W	K2FREE	SDCLK<2> Free-Running When set, it forces SDCLK<2> to be free-running, regardless of the value of the MDREFR[APD] or MDREFR[K2RUN] bits. The bit resets to 1 so the clock is driven initially, which provides synchronous memory with a clock for resetting any internal circuitry. To disable free-running, clear this bit. 0 = SDCLK<2> Dependent on MDREFR[K2RUN] 1 = SDCLK<2> Free-running enabled
24	R/W	K1FREE	SDCLK<1> Free-Running When set, it forces SDCLK<1> to be free-running, regardless of the value of the MDREFR[APD] or MDREFR[K1RUN] bits. This bit resets to 1 so the clock is driven initially. This provides synchronous memory with a clock for resetting any internal circuitry. To disable free-running, clear this bit. 0 = SDCLK<1> Dependent on MDREFR[K1RUN] 1 = SDCLK<1> Free-running enabled
23	R/W	K0FREE	SDCLK<0> and SDCLK<3> Free-Running When set, forces SDCLK<0> and SDCLK<3> to be free-running, regardless of the value of the MDREFR[APD] or MDREFR[K0RUN] bits. This bit resets to 1 so the clock is driven initially. This provides synchronous memory with a clock for resetting any internal circuitry. To disable free-running, clear this bit. 0 = SDCLK<0> and SDCLK<3> dependent on MDREFR[K0RUN] 1 = SDCLK<0> and SDCLK<3> free-running enabled
22	R/W	SLFRSH	SDRAM Self-Refresh Control/Status Control/status bit for entering and exiting SDRAM self-refresh. It is automatically set upon a hardware or sleep-exit reset. Setting the SLFRSH bit forces a self-refresh command. E1PIN does not have to be cleared. The appropriate clock run bits (K1RUN and/or K2RUN) must remain set until SDRAM has entered self-refresh and must be set before exiting self-refresh (clearing SLFRSH). **NOTE:** This capability must be used with extreme caution because the resulting state prohibits automatic transitions for any commands. See Section 6.4.2.5. Clearing SLFRSH is a part of the hardware or sleep-exit reset procedure for SDRAM. See Section 6.4.10. 0 = Self-refresh disabled 1 = Self-refresh enabled
21	—	—	reserved

Table 6-26. MDREFR Bit Definitions (Sheet 3 of 5)

Physical Address 0x4800_0004 — **MDREFR** — **Memory Controller**

Bit	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
User Settings																																
	ALTREFA	ALTREFB	K0DB4	reserved			K2FREE	K1FREE	K0FREE	SLFRSH	reserved	APD	K2DB2	K2RUN	K1DB2	K1RUN	E1PIN	K0DB2	K0RUN	reserved	DRI											
Reset	0	0	1	0	0	0	1	1	1	1	?	0	1	0	1	0	0	1	0	?	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Access	Name	Description
20	R/W	APD	SDRAM/Synchronous Static Memory Auto-Power-Down Enable If APD is set and MDEFR[KxFREE] is cleared, the SDCKE signal and SDCLK<2:0> clocks are driven low when none of the corresponding partitions are being accessed. • If no SDRAM partitions are being accessed, the SDRAM is put into power-down mode and the SDCKE signal and SDCLK<2:1> clocks are driven low. • If one SDRAM partition pair is being used and the other pair is not, the SDCLKx clock to the partition pair that is not being used is driven low. • If synchronous flash memory is installed and not being accessed, the SDCLK<0> and SDCLK<3> clocks corresponding to static partitions are driven low and the synchronous flash chips are put into power-down mode. (see part datasheets). See Section 6.4.2.5 and Section 6.5.3 0 = APD disabled 1 = APD enabled
19	R/W	K2DB2	SDRAM Clock Pin 2 (SDCLK<2>) Divide-by-2 Control/Status When set, SDCLK<2> runs at one-half the memory clock frequency. When cleared, SDCLK<2> runs at the memory clock frequency. This bit is automatically set on hardware or sleep-exit reset. 0 = SDCLK<2> equals CLK_MEM 1 = SDCLK<2> equals CLK_MEM divided by 2
18	R/W	K2RUN	SDRAM Clock Pin 2 (SDCLK<2>) Run Control/Status Automatically cleared on hardware or sleep-exit reset. **NOTE:** Use extreme caution when clearing the K2RUN bit because the resulting state prohibits automatic transitions for any commands. See Section 6.4.2.5. Setting K1RUN or K2RUN is a part of the hardware and sleep-exit reset procedure for SDRAM. See Section 6.4.10. 0 = SDCLK<2> Dependent on MDREFR[K2FREE] 1 = SDCLK<2> Enabled
17	R/W	K1DB2	SDRAM Clock Pin 1 (SDCLK<1>) Divide-by-2 Control/Status When set, SDCLK<1> runs at one-half the CLK_MEM frequency. When cleared, SDCLK<1> runs at the memory clock frequency. This bit is automatically set on hardware or sleep-exit reset. 0 = SDCLK<1> equals CLK_MEM 1 = SDCLK<1> equals CLK_MEM divided by 2

Table 6-26. MDREFR Bit Definitions (Sheet 4 of 5)

Physical Address 0x4800_0004 — MDREFR — Memory Controller

Bit	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
User Settings																																
	ALTREFA	ALTREFB	K0DB4	reserved			K2FREE	K1FREE	K0FREE	SLFRSH	reserved	APD	K2DB2	K2RUN	K1DB2	K1RUN	E1PIN	K0DB2	K0RUN	reserved	DRI											
Reset	0	0	1	0	0	0	1	1	1	1	?	0	1	0	1	0	0	1	0	?	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Access	Name	Description
16	R/W	K1RUN	SDRAM Clock Pin 1 (SDCLK<1>) Run Control/Status Automatically cleared on hardware or sleep-exit reset. **NOTE:** Use extreme caution when clearing the K1RUN bit because the resulting state prohibits automatic transitions for any commands. See Section 6.4.2.5. Setting K1RUN or K2RUN is a part of the hardware and sleep-exit reset procedure for SDRAM. See Section 6.4.10. 0 = SDCLK<1> Dependent on MDREFR[K1FREE] 1 = SDCLK<1> Enabled
15	R/W	E1PIN	SDRAM Clock Enable Pin 1 (SDCKE) Level Control/Status Automatically cleared on hardware or sleep-exit reset. **NOTE:** Extreme caution must be taken when clearing the E1PIN bit to cause a power-down command (if K1RUN = 1 and/or K2RUN = 1, and SLFRSH = 0) because the resulting state prohibits automatic transitions for mode register set, read, write, and refresh commands. E1PIN must be set to cause a power-down-exit command (if K1RUN = 1 and/or K2RUN = 1, and SLFRSH = 0). See Section 6.4.10. Setting E1PIN is a part of the hardware reset or sleep-exit reset procedure for SDRAM. See Section 6.4.10. 0 = SDCKE Disabled 1 = SDCKE Enabled
14	R/W	K0DB2	Synchronous Static Memory Clock Pin 0 (SDCLK<0> and SDCLK<3>) Divide-by-2 Control/Status Second control/status bit for clock divisor of SDCLK<0>. The value programmed here is only valid when MDREFR[K0DB4] is cleared. When K0DB2 is set, SDCLK<0> and SDCLK<3> run at one-half the memory clock frequency. When clear, SDCLK<0> and SDCLK<3> run at the memory clock frequency. This bit is automatically set on hardware or sleep-exit reset. 0 = SDCLK<0> and SDCLK<3> equals CLK_MEM 1 = SDCLK<0> and SDCLK<3> equals CLK_MEM divided by two

Table 6-26. MDREFR Bit Definitions (Sheet 5 of 5)

Physical Address 0x4800_0004 — **MDREFR** — **Memory Controller**

Bit	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
User Settings																																
	ALTREFA	ALTREFB	K0DB4	reserved			K2FREE	K1FREE	K0FREE	SLFRSH	reserved	APD	K2DB2	K2RUN	K1DB2	K1RUN	E1PIN	K0DB2	K0RUN	reserved	DRI											
Reset	0	0	1	0	0	0	1	1	1	1	?	0	1	0	1	0	0	1	0	?	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Access	Name	Description
13	R/W	K0RUN	Synchronous Static Memory Clock Pin 0 (SDCLK<0> and SDCLK<3>) Run Control/Status Set on hardware or sleep-exit reset if static memory partition 0 (boot space) is configured for synchronous static memory (see Section 6.5.4). Otherwise, it is cleared on reset. Use extreme caution when clearing the K0RUN bit because the resulting state prohibits automatic transitions for any commands. See Section 6.4.10. 0 = SDCLK<0> and SDCLK<3> dependent on MDREFR[K0FREE] 1 = SDCLK<0> and SDCLK<3> enabled
12	—	—	reserved
11:0	R/W	DRI	SDRAM Refresh Interval for All Partitions The number of memory clock cycles (divided by 32) between auto refresh (CBR) cycles. One row is refreshed in each SDRAM bank during each CBR refresh cycle. This interval applies to all SDRAM in the four partitions. To calculate the refresh interval from this programmed number, multiply it by 32 and add 31. The value that must be loaded into this register is calculated as follows: • DRI = (Number of CLK_MEM cycles – 31) / 32 = (Refresh time / rows x memory clock frequency – 31) / 32. This must be programmed to be shared by both partition pairs. Therefore, the worst case number must be programmed. This number must be less than the $tRAS_{MAX}$ for the SDRAM being accessed. If all four SDRAM partitions are disabled, the refresh counter is disabled and refreshes are only performed when a single transaction to a disabled SDRAM partition is requested. If the clock frequency is changed, this register must be rewritten, even if the value has not changed. This causes a refresh and resets the refresh counter to the refresh interval. 0x000 = No refreshes are sent to the SDRAM. However, refresh cycles can occur if VLIO cycles are performed. **NOTE:** See the refresh rules for programming limitations in Section 6.5.1.4

6.5.2 Synchronous Static Memory Registers

Synchronous static memory is referred to as SX memory in this section.

6.5.2.1 Synchronous Static Memory Configuration Register (SXCNFG)

SXCNFG, shown in Table 6-27 and Table 6-28, controls all synchronous static memory. The MSCx[RBWx] field must be written to define the external data-bus width. For synchronous flash memory, writes are asynchronous and timing must be set up appropriately with the MSCx registers. Refer to Table 6-27 and Table 6-28.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 6-27. SXCNFG Bit Definitions (Sheet 1 of 4)

Physical Address 0x4800_001C — **SXCNFG** — **Memory Controller**

	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
User Settings†																	0	0	1	1	0	0	0	0	0	0	0	1	0	0		1
Bit	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
	SXCLEXT2	reserved	SXTP2		reserved								SXCL2		SXEN3	SXEN2	SXCLEXT0	reserved	SXTP0		reserved							SXCL0			SXEN1	SXEN0
Reset	0	1	0	0	?	?	?	?	?	?	?	0	0	1	0	0	0	1	0	0	?	?	?	?	?	?	?	0	0	1	0	0

Bits	Access	Name	Description
31	R/W	SXCLEXT2	Synchronous Flash Memory CAS Latency Extension for SXCL2 For synchronous flash memory, use this bit in conjunction with SXCL2 to achieve a 4-bit CAS latency.
30	—	—	reserved (set during writes and reads are undefined)
29:28	R/W	SXTP2	SX Memory Type for Partition Pair 2/3 0b00 = reserved 0b01 = reserved 0b10 = Synchronous flash memory in burst-of-8 mode 0b11 = Synchronous flash memory in burst-of-16 mode
27:21	—	—	reserved

NOTE:

† The User Settings row represents the settings required for PXA271, PXA272 processor configuration only. Other configurations may differ.

Table 6-27. SXCNFG Bit Definitions (Sheet 2 of 4)

Physical Address 0x4800_001C — **SXCNFG** — **Memory Controller**

	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
User Settings†																	0	0	1	1	0	0	0	0	0	0	0	1	0	0		1
Bit	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
	SXCLEXT2	reserved	SXTP2	reserved								SXCL2			SXEN3	SXEN2	SXCLEXT0	reserved	SXTP0	reserved								SXCL0			SXEN1	SXEN0
Reset	0	1	0	0	?	?	?	?	?	?	?	0	0	1	0	0	0	1	0	0	?	?	?	?	?	?	?	0	0	1	0	0

Bits	Access	Name	Description
20:18	R/W	SXCL2	CAS Latency for SX Memory Partition Pair 2/3 SXCL2 is the number of SDCLK cycles between the time that the read command is received and the data is latched. The unit size for SXCL2 is the external SDCLK cycle; when SX memory is run at half the memory clock frequency (MDREFR[K0DB2] is set), the delay is two CLK_MEM cycles. Use in conjunction with SXCNFG[SXCLEXT2] to achieve the following CAS latencies. The field SXCLEXT2 is added to the beginning of the SXCL2 field to create the 4 bit values shown below. Any frequency configuration code that inserts wait states in the return of the data from flash memory is not valid. See the appropriate flash datasheets: 0b0000 = reserved 0b0001 = reserved 0b0010 = 3 clocks (frequency configuration code = 2) 0b0011 = 4 clocks (frequency configuration code = 3) 0b0100 = 5 clocks (frequency configuration code = 4) 0b0101 = 6 clocks (frequency configuration code = 5) 0b0110 = 7 clocks (frequency configuration code = 6) 0b0111 = 8 clocks (frequency configuration code = 7) 0b1000 = 9 clocks (frequency configuration code = 8) 0b1001 = 10 clocks (frequency configuration code = 9) 0b1010 = 11 clocks (frequency configuration code = 10) 0b1011 = 12 clocks (frequency configuration code = 11) 0b1100 = 13 clocks (frequency configuration code = 12) 0b1101 = 14 clocks (frequency configuration code = 13) 0b1110 = 15 clocks (frequency configuration code = 14) 0b1111 = 16 clocks (frequency configuration code = 15) The first access latency count configuration tells the device how many clocks must elapse from ADV#-inactive (V_{IH}) before the first data word must be driven onto its data pins. The input clock frequency determines this value.
17	R/W	SXEN3	SX Memory Partition 3 Enable 0 = Partition 3 is not enabled as SX memory. 1 = Partition 3 is enabled as SX memory.
16	R/W	SXEN2	SX Memory Partition 2 Enable 0 = Partition 2 is not enabled as SX memory. 1 = Partition 2 is enabled as SX memory.

NOTE:

† The User Settings row represents the settings required for PXA271, PXA272 processor configuration only. Other configurations may differ.

Table 6-27. SXCNFG Bit Definitions (Sheet 3 of 4)

Physical Address 0x4800_001C — **SXCNFG** — **Memory Controller**

	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
User Settings†																	0	0	1	1	0	0	0	0	0	0	0	1	0	0		1
Bit	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
	SXCLEXT2	reserved	SXTP2		reserved								SXCL2		SXEN3	SXEN2	SXCLEXT0	reserved	SXTP0		reserved								SXCL0		SXEN1	SXEN0
Reset	0	1	0	0	?	?	?	?	?	?	?	0	0	1	0	0	0	1	0	0	?	?	?	?	?	?	?	0	0	1	0	0

Bits	Access	Name	Description
15	R/W	SXCLEXT0	Synchronous Flash Memory CAS Latency Extension for SXCL0 For synchronous flash memory, use this bit in conjunction with SXCL0 to achieve a 4-bit CAS latency.
14	—	—	reserved (set during writes and reads are undefined)
13:12	R/W	SXTP0	SX Memory Type for Partition Pair 0/1 0b00 = reserved 0b01 = reserved 0b10 = Synchronous flash memory in burst-of-eight mode 0b11 = Synchronous flash memory in burst-of-16 mode
11:5	—	—	reserved

NOTE:

† The User Settings row represents the settings required for PXA271, PXA272 processor configuration only. Other configurations may differ.

Table 6-27. SXCNFG Bit Definitions (Sheet 4 of 4)

Physical Address 0x4800_001C — **SXCNFG** — **Memory Controller**

	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
User Settings†																	0	0	1	1	0	0	0	0	0	0	0	1	0	0		1
Bit	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
	SXCLEXT2	reserved	SXTP2		reserved							SXCL2			SXEN3	SXEN2	SXCLEXT0	reserved	SXTP0		reserved							SXCL0			SXEN1	SXEN0
Reset	0	1	0	0	?	?	?	?	?	?	?	0	0	1	0	0	0	1	0	0	?	?	?	?	?	?	?	0	0	1	0	0

Bits	Access	Name	Description
4:2	R/W	SXCL0	CAS Latency for SX Memory Partition Pair 0/1 SXCL0 is the number of SDCLK cycles between the time that the read command is received and the data is latched. The unit size for SXCL0 is the external SDCLK cycle. When SX memory runs at half the memory clock frequency (MDREFR[K0DB0] is set), the delay is two CLK_MEM cycles. Use in conjunction with SXCNFG[SXCLEXT0] to achieve the following CAS latencies. SXCLEXT0 represents bit (3). Any frequency configuration code that inserts wait states in the return of the data from flash memory is not valid. See the appropriate flash data sheet for more details. 0b0000 = reserved 0b0001 = reserved 0b0010 = 3 clocks (frequency configuration code = 2) 0b0011 = 4 clocks (frequency configuration code = 3) 0b0100 = 5 clocks (frequency configuration code = 4) 0b0101 = 6 clocks (frequency configuration code = 5) 0b0110 = 7 clocks (frequency configuration code = 6) 0b0111 = 8 clocks (frequency configuration code = 7) 0b1000 = 9 clocks (frequency configuration code = 8) 0b1001 = 10 clocks (frequency configuration code = 9) 0b1010 = 11 clocks (frequency configuration code = 10) 0b1011 = 12 clocks (frequency configuration code = 11) 0b1100 = 13 clocks (frequency configuration code = 12) 0b1101 = 14 clocks (frequency configuration code = 13) 0b1110 = 15 clocks (frequency configuration code = 14) 0b1111 = 16 clocks (frequency configuration code = 15) The first access latency count configuration tells the device how many clocks must elapse from ADV#-inactive (V_{IH}) before the first data word must be driven onto its data pins. The input clock frequency determines this value.
1	R/W	SXEN1	SX Memory Partition 1 Enable 0 = Partition 1 is not enabled as SX memory. 1 = Partition 1 is enabled as SX memory.
0	R/W	SXEN0	SX Memory Partition 0 Enable 0 = Partition 0 is not enabled as SX memory. 1 = Partition 0 is enabled as SX memory.

NOTE:

† The User Settings row represents the settings required for PXA271, PXA272 processor configuration only. Other configurations may differ.

6.5.3 Asynchronous Static Memory Registers

6.5.3.1 Static Memory Control Registers (MSCx)

MSCx, shown in Table 6-28, contains control bits for configuring static memory or VLIO. These bits correspond to chip-select pairs nCS<1:0>, nCS<3:2>, and nCS<5:4>. Timing fields are specified in number of memory clock cycles. Each register contains two identical CNFG fields: one for each chip select within the pair.

When programming a different memory type in an MSC register, ensure that the new value has been accepted and programmed before issuing a command to that memory. To do this, read the MSC register before accessing the memory.

If any of the nCS<3:0> banks are configured for synchronous flash memory with SXCNFG[SXENx], the corresponding half words of MSC0 or MSC1 are ignored on reads, with the exception of MSCx[RBWx], the data width. This field must be programmed appropriately. Because writes to the synchronous flash devices are asynchronous, all values programmed in the MSCx register apply for writes.

In Table 6-28:

Bit fields in the MSCx register with the suffix of 1 and 0 belong to the MSC0 register.

Bit fields in the MSCx register with the suffix of 3 and 2 belong to the MSC1 register.

Bit fields in the MSCx register with the suffix of 5 and 4 belong to the MSC2 register.

MSC0 maps to physical address 0x4800_0008 and programs bit fields with the suffix of 1 and 0.

MSC1 maps to physical address 0x4800_000C and programs bit fields with the suffix of 3 and 2.

MSC2 maps to physical address 0x4800_0010 and programs bit fields with the suffix of 5 and 4.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 6-28. MSC0/1/2 Bit Definitions (Sheet 1 of 8)

Physical Address
0x4800_0008
0x4800_000C
0x4800_0010

MSC0
MSC1
MSC2

Memory Controller

User Settings†																													††			
Bit	**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**
	RBUFF1/3/5	**RRR1/3/5**			**RDN1/3/5**				**RDF1/3/5**				**RBW1/3/5**	**RT1/3/5**			**RBUFF0/2/4**	**RRR0/2/4**			**RDN0/2/4**				**RDF0/2/4**				**RBW0/2/4**	**RT0/2/4**		
MSC2/1 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	?	0	0	0
MSC0 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	*	0	0	0

Bits	Access	Name	Description
31	R/W	RBUFFx	Return Data Buffer vs. Streaming Behavior When slow memory devices are used in the system (VLIO, slow SRAM/ROM), this bit must be cleared to allow the system to process other information rather than remain idle while all the data is read from the device. When this bit is cleared, the system is allowed to process other information. When the bit is set, the system halts and waits until all data is returned from the device. When synchronous static memory devices have been enabled for a given bank, this bit defaults to streaming behavior. The register bit is read as 0 unless it has specifically been programmed to a 1. This behavior cannot be overridden. 0 = Slower device (return data buffer) 1 = Faster device (streaming behavior)
30:28	R/W	RRRx	ROM/SRAM Recovery Time The value of this bit is half the number of memory clock cycles from the time that chip select is de-asserted after a read or write until the next chip select (of a different static memory bank) or nSDCS is asserted. This field must be programmed with the highest of the three values: t_{OFF} divided by two, write pulse high time (flash memory/SRAM), and write recovery before read (flash memory). t_{OFF} = RRRx * 2 + 1 **NOTE:** The MSCx[RRR] value (recovery time after chip select de-asserted) must be reprogrammed prior to switching the processor to deep-idle mode to avoid long times when the MD bus (MD<31:0>) is in three-state mode after reads.

NOTES:

† The User Settings row represents the settings required for PXA271, PXA272 processor configuration only. Other configurations may differ.

†† This bit must be written as a 1 for the PXA271 processor. This bit must be written as a 0 for the PXA272 processor.

Table 6-28. MSC0/1/2 Bit Definitions (Sheet 2 of 8)

Physical Address
0x4800_0008
0x4800_000C
0x4800_0010

MSC0
MSC1
MSC2

Memory Controller

User Settings†																												††			
Bit	**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**
	RBUFF1/3/5	**RRR1/3/5**			**RDN1/3/5**				**RDF1/3/5**				**RBW1/3/5**	**RT1/3/5**			**RBUFF0/2/4**	**RRR0/2/4**			**RDN0/2/4**				**RDF0/2/4**				**RBW0/2/4**	**RT0/2/4**		
MSC2/1 Reset	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**?**	**0**	**0**	**0**
MSC0 Reset	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	*****	**0**	**0**	**0**

Bits	Access	Name	Description
27:24	R/W	RDNx	ROM Delay Next Access The RDN field is encoded as follows: ENCODED (Programmed) Value -----> DECODED (Actual) Value 0-11 -----> 0-11 12 -----> 15 13 -----> 20 14 -----> 26 15 -----> 30 Use the DECODED value in the equations below for RDNx instead of the actual RDNx value. For burst ROM or flash memory: • RDNx + 1 = number of MEM_CLKs from Address to Data Valid for subsequent access. For flash memory or SRAM: • RDNx + 1 = number of CLK_MEMs nWE is asserted for write accesses For VLIO: • RDNx * 2 = amount of time nOE or nPWE is deasserted to address hold and address setup to nOE or nPWE assertion time. **NOTE:** For VLIO, this number must be greater than or equal to 2. The memory controller substitutes a default value of 2 for values less than 2.

NOTES:

† The User Settings row represents the settings required for PXA271, PXA272 processor configuration only. Other configurations may differ.

†† This bit must be written as a 1 for the PXA271 processor. This bit must be written as a 0 for the PXA272 processor.

Table 6-28. MSC0/1/2 Bit Definitions (Sheet 3 of 8)

Physical Address
0x4800_0008
0x4800_000C
0x4800_0010

MSC0
MSC1
MSC2

Memory Controller

	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
User Settings†																													††				
Bit	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
	RBUFF1/3/5	RRR1/3/5			RDN1/3/5				RDF1/3/5				RBW1/3/5	RT1/3/5			RBUFF0/2/4	RRR0/2/4			RDN0/2/4				RDF0/2/4				RBW0/2/4	RT0/2/4		
MSC2/1 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	?	0	0	0
MSC0 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	*	0	0	0

Bits	Access	Name	Description
23:20	R/W	RDFx	ROM Delay First Access The encoding scheme is: ENCODED (programmed) value ------> DECODED (actual) value: 0-11 ------> 0-11 12 ------> 15 13 ------> 20 14 -----> 26 15 ------> 30 The DECODED value represents: • Number of memory-clock cycles (minus 2) from address to data valid for first read access from all devices except VLIO • Number of memory clock cycles (minus 1) from address to data valid for subsequent read accesses to non-burst devices except VLIO • Number of memory clock cycles (minus 1) of nWE assertion for write accesses to all types of flash memory For variable-latency I/O, this determines the minimum number of memory clock cycles (minus 1) of nOE (nPWE) assert time for each beat of read (write). **NOTE:** For VLIO, this number must be greater than or equal to 3. The memory controller substitutes a default value of 3 for values less than 3.
19	R/W	RBWx	ROM Bus Width For reset value of RBW0, see Section 6.5.4. This value must be programmed, even when using synchronous static memory in banks 0,1,2, or 3. 0 = 32 bits 1 = 16 bits **NOTE:** This value must not be changed during normal operation.

NOTES:

† The User Settings row represents the settings required for PXA271, PXA272 processor configuration only. Other configurations may differ.

†† This bit must be written as a 1 for the PXA271 processor. This bit must be written as a 0 for the PXA272 processor.

Table 6-28. MSC0/1/2 Bit Definitions (Sheet 4 of 8)

Physical Address
0x4800_0008
0x4800_000C
0x4800_0010

MSC0
MSC1
MSC2

Memory Controller

	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
User Settings†																													††				
Bit	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
	RBUFF1/3/5	RRR1/3/5			RDN1/3/5				RDF1/3/5				RBW1/3/5	RT1/3/5			RBUFF0/2/4	RRR0/2/4			RDN0/2/4				RDF0/2/4				RBW0/2/4	RT0/2/4		
MSC2/1 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	?	0	0	0
MSC0 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	*	0	0	0

Bits	Access	Name	Description
18:16	R/W	RTx	ROM Type: 0b000 = Synchronous flash or non-burst ROM or non-burst flash 0b001 = SRAM 0b010 = Burst-of-four ROM or burst-of-four flash (with non-burst writes) 0b011 = Burst-of-eight ROM or burst-of-eight flash (with non-burst writes) 0b100 = Variable-latency I/O (VLIO) 0b101 = reserved 0b110 = reserved 0b111 = reserved
15	R/W	RBUFFx	Return Data Buffer vs. Streaming Behavior When slow memory devices are used in the system (VLIO, slow SRAM/ROM), this bit must be cleared to allow the system to process other information rather than remain idle while all the data is read from the device. When this bit is cleared, the system is allowed to process other information. When the bit is set, the system halts and waits until all data is returned from the device. When synchronous static memory devices have been enabled for a given bank, this bit defaults to streaming behavior. The register bit is read as 0 unless it has specifically been programmed to a 1. This behavior cannot be overridden. 0 = Slower device (return data buffer) 1 = Faster device (streaming behavior)

NOTES:

† The User Settings row represents the settings required for PXA271, PXA272 processor configuration only. Other configurations may differ.

†† This bit must be written as a 1 for the PXA271 processor. This bit must be written as a 0 for the PXA272 processor.

Table 6-28. MSC0/1/2 Bit Definitions (Sheet 5 of 8)

Physical Address
0x4800_0008
0x4800_000C
0x4800_0010

MSC0
MSC1
MSC2

Memory Controller

	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
User Settings†																													††			
Bit	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
	RBUFF1/3/5	RRR1/3/5			RDN1/3/5				RDF1/3/5				RBW1/3/5	RT1/3/5			RBUFF0/2/4	RRR0/2/4			RDN0/2/4				RDF0/2/4				RBW0/2/4	RT0/2/4		
MSC2/1 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	?	0	0	0
MSC0 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	*	0	0	0

Bits	Access	Name	Description
14:12	R/W	RRRx	ROM/SRAM Recovery Time The value of this bit is half the number of memory clock cycles from the time that chip select is deasserted after a read or write until the next chip select (of a different static memory bank) or nSDCS is asserted. This field must be programmed with the highest of the three values: t_{OFF} divided by two, write pulse high time (flash memory/SRAM), and write recovery before read (flash memory). t_{OFF} = RRRx * 2 + 1 **NOTE:** The MSCx[RRR] value (recovery time after chip select deasserted) must be reprogrammed prior to switching the processor to deep-idle mode to avoid long times when the MD bus (MD<31:0>) is in three-state mode after reads.

NOTES:

† The User Settings row represents the settings required for PXA271, PXA272 processor configuration only. Other configurations may differ.

†† This bit must be written as a 1 for the PXA271 processor. This bit must be written as a 0 for the PXA272 processor.

Table 6-28. MSC0/1/2 Bit Definitions (Sheet 6 of 8)

Physical Address
0x4800_0008
0x4800_000C
0x4800_0010

MSC0
MSC1
MSC2

Memory Controller

	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
User Settings†																													††				
Bit	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
	RBUFF1/3/5	RRR1/3/5			RDN1/3/5				RDF1/3/5				RBW1/3/5	RT1/3/5			RBUFF0/2/4	RRR0/2/4			RDN0/2/4				RDF0/2/4				RBW0/2/4	RT0/2/4		
MSC2/1 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	?	0	0	0
MSC0 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	*	0	0	0

Bits	Access	Name	Description
11:8	R/W	RDNx	ROM Delay Next Access The RDN field is encoded as follows: ENCODED (Programmed) Value -----> DECODED (Actual) Value 0–11 -----> 0–11 12 -----> 15 13 -----> 20 14 -----> 26 15 -----> 30 Use the DECODED value in the equations below for RDNx instead of the actual RDNx value. For burst ROM or flash memory: • RDNx + 1 = number of MEM_CLKs from Address to Data Valid for subsequent access. For flash memory or SRAM: • RDNx + 1 = number of CLK_MEMs nWE is asserted for write accesses For VLIO: • RDNx * 2 = amount of time nOE or nPWE is deasserted to address hold and address setup to nOE or nPWE assertion time. **NOTE:** For VLIO, this number must be greater than or equal to 2. The memory controller substitutes a default value of 2 for values less than 2.

NOTES:

† The User Settings row represents the settings required for PXA271, PXA272 processor configuration only. Other configurations may differ.

†† This bit must be written as a 1 for the PXA271 processor. This bit must be written as a 0 for the PXA272 processor.

Table 6-28. MSC0/1/2 Bit Definitions (Sheet 7 of 8)

Physical Address
0x4800_0008
0x4800_000C
0x4800_0010

MSC0
MSC1
MSC2

Memory Controller

	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
User Settings†																													††			
Bit	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
	RBUFF1/3/5	RRR1/3/5			RDN1/3/5				RDF1/3/5				RBW1/3/5	RT1/3/5			RBUFF0/2/4	RRR0/2/4			RDN0/2/4				RDF0/2/4				RBW0/2/4	RT0/2/4		
MSC2/1 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	?	0	0	0
MSC0 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	*	0	0	0

Bits	Access	Name	Description
7:4	R/W	RDFx	The ROM Delay First Access The programmed value to actual value encoding scheme is: ENCODED (programmed) value ------> DECODED (actual) value: 0–11 ------> 0–11 12 ------> 15 13 ------> 20 14 ------> 26 15 ------> 30 The DECODED value represents: • The number of memory clock cycles (minus 2) from address to data valid for first read access from all devices except VLIO • The number of memory clock cycles (minus 1) from address to data valid for subsequent read accesses to non-burst devices except VLIO • The number of memory clock cycles (minus 1) of nWE assertion for write accesses to all types of flash memory For variable-latency I/O, this determines the minimum number of memory clock cycles (minus 1) of nOE (nPWE) assert time for each beat of read (write). **NOTE:** For VLIO, this number must be greater than or equal to 3. The memory controller substitutes a default value of 3 for values less than 3.
3	R/W	RBWx	ROM Bus Width This value must be programmed, even when using synchronous static memory in banks 0,1,2, or 3. 0 = 32 bits 1 = 16 bits **NOTE:** This value must not be changed during normal operation.

NOTES:

† The User Settings row represents the settings required for PXA271, PXA272 processor configuration only. Other configurations may differ.

†† This bit must be written as a 1 for the PXA271 processor. This bit must be written as a 0 for the PXA272 processor.

Table 6-28. MSC0/1/2 Bit Definitions (Sheet 8 of 8)

Physical Address																															
0x4800_0008 0x4800_000C 0x4800_0010													MSC0 MSC1 MSC2								Memory Controller										
User Settings†																											††				
Bit	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
	RBUFF1/3/5	RRR1/3/5			RDN1/3/5				RDF1/3/5				RBW1/3/5	RT1/3/5			RBUFF0/2/4	RRR0/2/4			RDN0/2/4				RDF0/2/4				RBW0/2/4	RT0/2/4		
MSC2/1 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	?	0	0	0
MSC0 Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	*	0	0	0

Bits	Access	Name	Description
2:0	R/W	RTx	ROM Type 0b000 = Synchronous flash or non-burst ROM or non-burst flash 0b001 = SRAM 0b010 = Burst-of-four ROM or burst-of-four flash (with non-burst writes) 0b011 = Burst-of-eight ROM or burst-of-eight flash (with non-burst writes) 0b100 = Variable-latency I/O (VLIO) 0b101 = reserved 0b110 = reserved 0b111 = reserved

NOTES:

† The User Settings row represents the settings required for PXA271, PXA272 processor configuration only. Other configurations may differ.

†† This bit must be written as a 1 for the PXA271 processor. This bit must be written as a 0 for the PXA272 processor.

6.5.3.2 Static Memory SA-1110 Compatibility Configuration Register (SA1110)

SA1110, defined in Table 6-31, facilitates address and data-mask configuration that is compatible with the Intel ® StrongARM* SA-1110 processor and also provides a programmable option for support of larger static memories.

6.5.3.2.1 Large Static Memory Support

When SA1110[SXENX] is set, larger memories are supported in the existing memory space for static memories. There are no restrictions on what type of memories these may be. These memories may be up to 128 Mbytes each, rather than up to 64 Mbytes each and require an extra address bit, as MA<25:2> for 32-bit memories and MA<25:1> for 16-bit memories does not provide a large enough address for the large memories. The extra address bit is MA<0>. Thus, SA-1110 compatibility addressing mode is not valid if SA1110[SXENX] = 0b1. The 128 Mbyte half-word address is MA<0,25:1>. The 64-Mbyte byte address remains MA<25:0>.

Figure 6-11. Programmable Static Memory Map Options

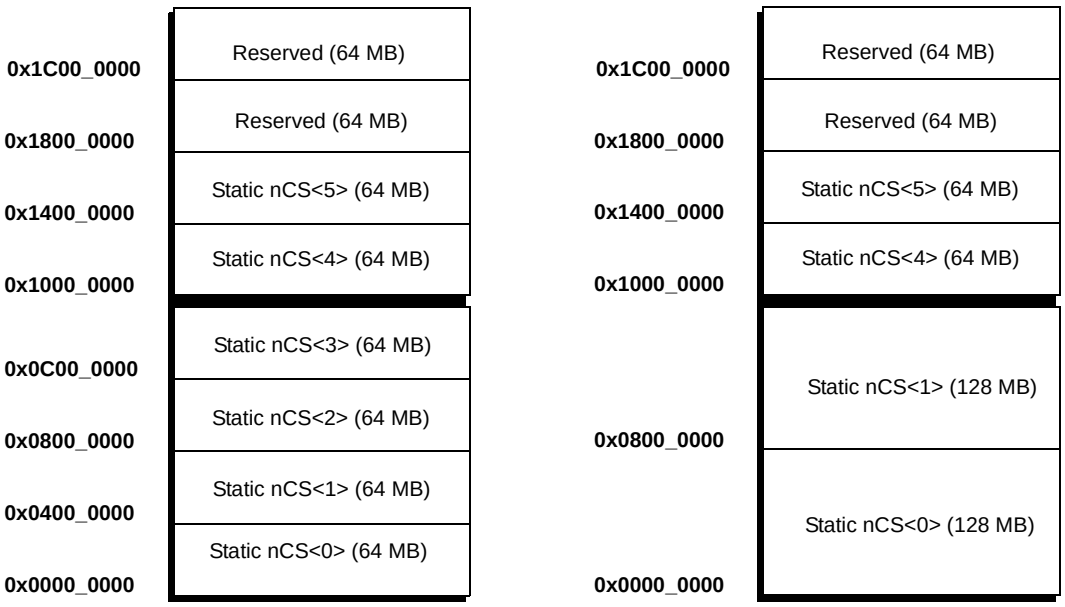
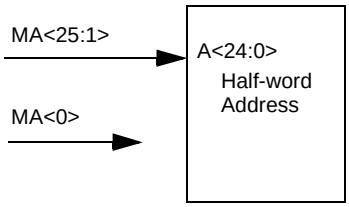


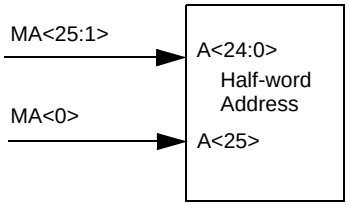
Figure 6-12. Addressing Instructions for Static Memory Chip Selects

64 Mbyte 16-bit Memory



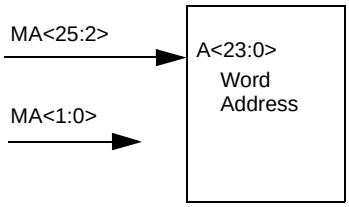
Byte address MA<0> may or may not be used by the memory, depending on the setting of SA1110[SA1110_x].

128 Mbyte 16-bit Memory



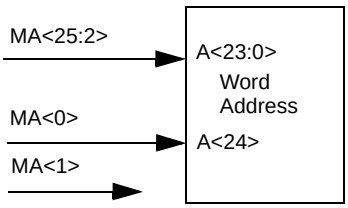
No byte address is given. SA1110[SA1110_x] must be clear.

64 Mbyte 32-bit Memory



Byte address MA<1:0> may or may not be used by the memory, depending on the setting of SA1110[SA1110_x].

128 Mbyte 32-bit Memory



No byte address is given. SA1110[SA1110_x] must be clear.

6.5.3.2.2 SA-1110 Addressing and Data-Masking Compatibility

When SA-1110 compatibility is enabled for a static memory partition (SA1110[SA1110_x] clear), two things occur on reads from static memory:

1. On reads for asynchronous memory, the lower address bits correctly reflect the starting byte address—MA<0> for 16-bit external memory and MA<1:0> for 32-bit external memory, based on the byte enables that may be associated with the read request from the internal bus. See Table 6-29 and Table 6-30 for details of the external address for this mode.
2. For any memory type, the DQM pins correctly reflect the byte enables received for the reads.

Register SA1110 must be programmed with bit setting as shown in the User Settings row in Table 6-31.

Table 6-29. 32-Bit Byte Address Bits MA<1:0> for Reads Based on DQM<3:0>

DQM<3:0>	MA<1:0>
0b0000	0b00
0b0001 0b1101 0b1001 0b0101	0b01
0b1011 0b0011	0b10
0b0111	0b11
Anything Else	0b00

Table 6-30. 16-Bit Byte Address Bit MA<0> for Reads Based on DQM<1:0>

DQM<1:0>	MA<0>]
0b00	0b0
0b10	0b0
0b01	0b1
0b11	0b0

Table 6-31. SA1110 Bit Definitions (Sheet 1 of 2)

Physical Address 0x4800_0064 — SA1110 — Memory Controller

	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
User Settings†																			0	1												
Bit	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																		SXSTACK		reserved			SXENX	reserved		SA1110_5	SA1110_4	SA1110_3	SA1110_2	SA1110_1	SA1110_0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
31:14	—	—	reserved
13:12	R/W	SXSTACK	Stacked Flash Option 0b00 = There is no stacked flash in the system 0b01 = nCS<0> contains stacked flash. 0b10 = nCS<1> contains stacked flash. 0b11 = Both nCS<0> and nCS<1> contain stacked flash. See Section 6.4.2 for information on stacking flash and SDRAM devices.
11:9	—	—	reserved
8	R/W	SXENX	Large Memory Support See Figure 6-11 for a diagram. 0 = Use six 64 Mbyte chip selects (nCS<5:0> 1 = Use two 64 Mbyte chip selects (nCS<5:4>) and two 128 Mbyte chip selects (nCS<1:0>). The corresponding SA1110[3:0] bit fields must be cleared.
7:6	—	—	reserved
5	R/W	SA1110_5††	SA-1110 Compatibility Mode for Static Memory Partition 5 0 = Disable SA-1110 compatibility. 1 = Enable SA-1110 compatibility.
4	R/W	SA1110_4††	SA-1110 Compatibility Mode for Static Memory Partition 4 0 = Disable SA-1110 compatibility. 1 = Enable SA-1110 compatibility.
3	R/W	SA1110_3††	SA-1110 Compatibility Mode for Static Memory Partition 3 0 = Disable SA-1110 compatibility. 1 = Enable SA-1110 compatibility.

NOTE:

† The User Settings row represents the settings required for the PXA271, PXA272 processors. Other configurations may differ.

†† Limitations apply. For a full discussion, see Section 6.5.3.2 and its subsections.

Table 6-31. SA1110 Bit Definitions (Sheet 2 of 2)

	Physical Address 0x4800_0064																SA1110								Memory Controller							
User Settings†																			0	1												
Bit	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																		SXSTACK		reserved			SXENX	reserved		SA1110_5	SA1110_4	SA1110_3	SA1110_2	SA1110_1	SA1110_0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
2	R/W	SA1110_2††	SA-1110 Compatibility Mode for Static Memory Partition 2 0 = Disable SA-1110 compatibility. 1 = Enable SA-1110 compatibility.
1	R/W	SA1110_1††	SA-1110 Compatibility Mode for Static Memory Partition 1 0 = Disable SA-1110 compatibility. 1 = Enable SA-1110 compatibility.
0	R/W	SA1110_0††	SA-1110 Compatibility Mode for Static Memory Partition 0 0 = Disable SA-1110 compatibility. 1 = Enable SA-1110 compatibility.

NOTE:

† The User Settings row represents the settings required for the PXA271, PXA272 processors. Other configurations may differ.

†† Limitations apply. For a full discussion, see Section 6.5.3.2 and its subsections.

6.5.4 Boot Time Default Configuration Register (BOOT_DEF)

This section provides information on default boot parameters. Table 6-32 shows the valid boot configurations. For all processor stack configurations, this signal must be driven to match the flash bus width within the stacked configuration.

Table 6-32. Valid Boot Configurations

BOOT_SEL Signal	Description
Deasserted	Asynchronous 32-bit ROM
Asserted	Asynchronous 16-bit ROM

BOOT_DEF, shown in Table 6-33, contains the boot-up values for the BOOT_SEL pin.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 6-33. BOOT_DEF Bit Definitions

Physical Address 0x4800_0044 — BOOT_DEF — Memory Controller

	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
User Settings																																
Bit	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																												PKG_TYPE	reserved		BOOT_SEL
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	?	?	*

Bits	Access	Name	Description
31:4	—	—	reserved
3	R	PKG_TYPE	Package Type 0 = reserved 1 = 32-bit package
2:1	—	—	reserved
0	R	BOOT_SEL	Boot Select Contains the input pin BOOT_SEL for the memory controller. See Section 6.4.7 for pin configuration definitions. 0 = BOOT_SEL is low (boot from 32-bit memory). 1 = BOOT_SEL is high (boot from 16-bit memory).

6.5.5 Expansion Memory Timing Configuration Registers (MCMEMx, MCATTx, MCIOx)

This section provides information on the processor's PC Card interface, which conforms to the *PC Card Standard, Volume 2, Electrical Specification, Version 1.4* and to the *CF+ and CompactFlash Specification Version 1.4*.

The following registers contain control bits for configuring the timing of the PC Card/CompactFlash interface. Registers 0 apply to socket 0; registers 1 apply to socket 1.

- MCMEM0/1 (memory space)—Table 6-35
- MCATT0/1 (attribute space)—Table 6-36
- MCIO0/1 (I/O space)—Table 6-37

Use the three fields in each of the six registers to individually select the duration of accesses to I/O, common memory, and attribute space for each of the two PC Card/CompactFlash card sockets, as shown in Table 6-34.

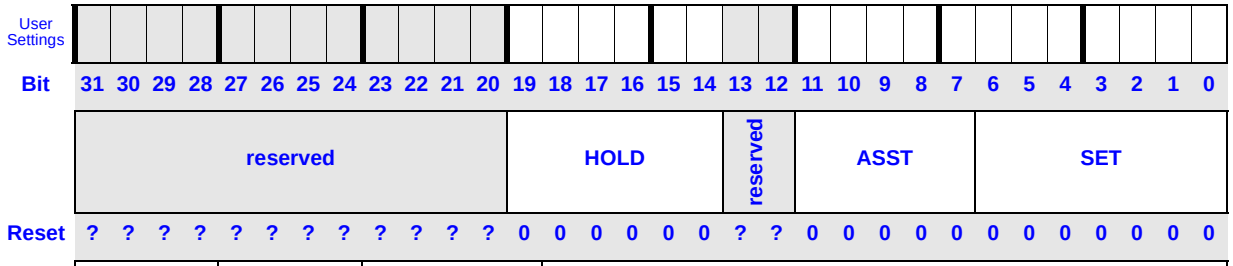
Table 6-34. PC Card/CompactFlash Interface Command Assertion Code

MCMEMx[ASST] MCATTx[ASST] MCIOx[ASST]		*x_ASST_WAIT*	*x_ASST_HOLD*		*x_CMD (x_ASST_WAIT +* (x_ASST_HOLD)	
Programmed Bit Value	Code Decimal value	# CLK_MEMs to Wait Before Checking for nPWAIT = 1	# CLK_MEMs t Assert Command After nPWAIT = 1		# CLK_MEMs for Minimum Command Assertion Time	
(Code)	(Code)	(Code + 2)	(2*Code + 3) (writes)	(2*Code + 4) (reads)	(3*Code + 5) (writes)	(3*Code + 6) (reads)
00000	0	2	3	4	5	6
00001	1	3	5	6	8	9
00010	2	4	7	8	11	12
00011	3	5	9	10	14	15
00100	4	6	11	12	17	18
00101	5	7	13	14	20	21
00110	6	8	15	16	23	24
00111	7	9	17	18	26	27
01000	8	10	19	20	29	30
01001	9	11	21	22	32	33
01010	10	12	23	24	35	36
01011	11	13	25	26	38	39
01100	12	14	27	28	41	42
01101	13	15	29	30	44	45
01110	14	16	31	32	47	48
01111	15	17	33	34	50	51
10000	16	18	35	36	53	54
10001	17	19	37	38	56	57
10010	18	20	39	40	59	60
10011	19	21	41	42	62	63
10100	20	22	43	44	65	66
10101	21	23	45	46	68	69
10110	22	24	47	48	71	72
10111	23	25	49	50	74	75
11000	24	26	51	52	77	78
11001	25	27	53	54	80	81
11010	26	28	55	56	83	84
11011	27	29	57	58	86	87
11100	28	30	59	60	89	90
11101	29	31	61	62	92	93
11110	30	32	63	64	95	96
11111	31	33	65	66	98	99

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 6-35. MCMEMx Bit Definitions

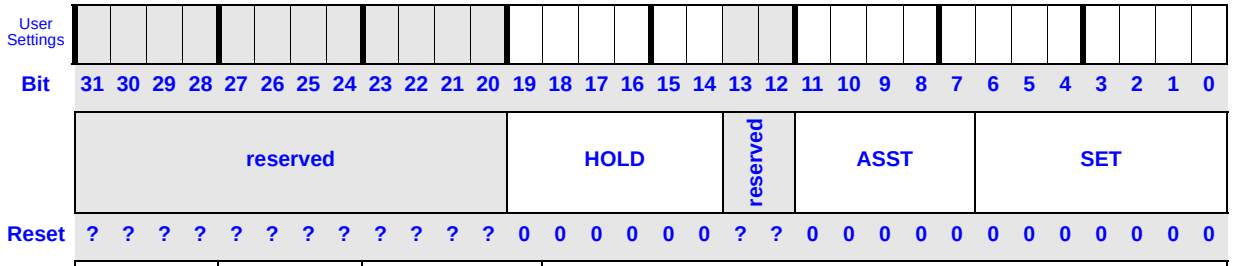
Physical Address		MCMEM0	Memory Controller
0x4800_0028	0x4800_002C	MCMEM1	



Bits	Access	Name	Description
31:20	—	—	reserved
19:14	R/W	HOLD	MCMEMx Address Hold Number of memory clocks to hold address after command deassertion for the MCMEM for socket *x*.
13:12	—	—	reserved
11:7	R/W	ASST	MCMEMx Command Assert Code for the command assertion time. See Table 6-34 for a description of this code and its effects on the command assertion.
6:0	R/W	SET	MCMEMx Address Setup Number of memory clocks (minus 2) to set up address before command assertion for the MCMEM for socket *x*.

Table 6-36. MCATT0/1 Bit Definitions (Sheet 1 of 2)

Physical Address		MCATT0	Memory Controller
0x4800_0030	0x4800_0034	MCATT1	



Bits	Access	Name	Description
31:20	—	—	reserved
19:14	R/W	HOLD	MCATTx Address Hold Number of memory clocks to hold address after command deassertion for the MCATT for socket *x*.

Table 6-36. MCATT0/1 Bit Definitions (Sheet 2 of 2)

Physical Address 0x4800_0030 0x4800_0034 — MCATT0 MCATT1 — Memory Controller

User Settings

Bit	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												HOLD				reserved		ASST					SET								
Reset	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
13:12	—	—	reserved
11:7	R/W	ASST	MCATTx Command Assert Code for the command assertion time. See Table 6-34 for a description of this code and its effects on the command assertion.
6:0	R/W	SET	MCATTx Address Setup Number of memory clocks (minus 2) to set up address before command assertion for the MCATT for socket *x*.

Table 6-37. MCIO0/1 Bit Definitions

Physical Address 0x4800_0038 0x4800_003C — MCIO0 MCIO1 — Memory Controller

User Settings

Bit	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												HOLD				reserved		ASST					SET								
Reset	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:20	—	—	reserved
19:14	R/W	HOLD	MCIOx Address Hold Number of memory clocks to hold address after command deassertion for the MCIO for socket *x*.
13:12	—	—	reserved
11:7	R/W	ASST	MCIOx Command Assertion Code for the command assertion time. See Table 6-34 for a description of this code and its effects on the command assertion.
6:0	R/W	SET	MCIOx Address Setup Number of (memory clocks (minus 2) to set up address before command assertion for the MCIO for socket *x*.

6.5.6 Expansion Memory Configuration Register (MECR)

MECR, shown in Table 6-38, reduces the need for external hardware. Two bits indicate to the memory controller that a PC Card has been inserted in the socket and the number of PC Cards the system has been designed for. The number-of-sockets (NOS) bit is required because the PSKTSEL pin is used as the nOE for the data transceivers. The PC-Card-is-there (CIT) bit reduces external hardware by allowing the system to ignore nIOIS16 and nPWAIT when there is no PC Card inserted in the socket.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 6-38. MECR Bit Definitions

Physical Address 0x4800_0014 — **MECR** — **Memory Controller**

	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
User Settings																																
Bit	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																														CIT	NOS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
31:2	—	—	reserved
1	R/W	CIT	PC Card Is Present This bit must be written by software when either PC Card is inserted into a socket. 0 = No PC Card inserted 1 = PC Card inserted
0	R/W	NOS	PC Card Interface Number of Sockets 0 = 1 socket 1 = 2 sockets

6.5.7 Programmable Output Buffer Strength Registers

The following registers eliminate the requirement for termination on the high-speed address control and data signals used by the memory controller. To clearly understand the impact on programming the buffer-strength registers, use the IBIS models that correspond to the preferred buffer-strength bit configuration in a simulation model. The buffer impedances are listed for a subset of the buffer strength listed in Table 6-39. The AC timings corresponding to the specific buffer-strength configurations are documented in the *Intel® PXA27x Processor Family EMTS*.

Most signals are independently programmable to a specific buffer strength, due to the possibility of unbalanced loading. Signals expected to have the same capacitance loading are grouped together and are programmable using the same bit-field location within the buffer-strength registers.

Table 6-39 indicates values of the buffer impedance for signals based on the programmed bit strength of the individual signals. Values in Table 6-39 are specific to the typical process and a temperature of 25 degrees Celsius. The program values vary across processes and temperatures, though they do maintain the same relation. Larger programmed values in the buffer-strength bit field result in a smaller resistance and a smaller programmed value in the buffer-strength bit field result in a larger resistance.

Note: Use caution when configuring different strength values for signals used by a same memory device, as this could cause AC timing violations due to the added skew between signals of the same memory device.

Table 6-39. Impedance Selection Options

BS Bit 3	BS Bit 2	BS Bit 1	BS Bit 0	Total Resistance (Ohms)		
				1.8 V	2.5 V	3.3 V
0	0	0	0	387.0	290	240
0	0	0	1	129.2	96.7	80
0	0	1	0	77.6	58.3	48
0	0	1	1	55.4	41.6	34.3
0	1	0	0	43.5	32.8	26.7
0	1	0	1	35.5	26.8	21.8
0	1	1	0	30	22.3	18.5
0	1	1	1	26	19.6	16
1	0	0	0	23.5	17.8	14.1
1	0	0	1	20.9	15.9	12.6
1	0	1	0	18.9	14.3	11.4
1	0	1	1	17.2	13	10.4
1	1	0	0	15.9	12	9.6
1	1	0	1	14.7	11.1	8.9
1	1	1	0	13.6	10.3	8.3
1	1	1	1	12.7	9.6	7.7

6.5.7.1 System Memory Buffer Strength Control Register 0 (BSCNTR0)

BSCNTR0 register, shown in Table 6-40, contains control bits for configuring the buffer strengths for the system memory output buffers.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 6-40. BSCNTR0 Bit Definitions

Physical Address 0x4800_004C																BSCNTR0								Memory Controller							
User Settings																															
Bit	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
	CKE1BS				CLK2BS				CLK1BS				CLK0BS				RASBS				CASBS				MDHBS				MDLBS			
Reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Bits	Access	Name	Description
31:28	R/W	CKE1BS	SDCKE Buffer Strength Control register
27:24	R/W	CLK2BS	SDCLK<2> Buffer Strength Control register
23:20	R/W	CLK1BS	SDCLK<1> Buffer Strength Control register
19:16	R/W	CLK0BS	SDCLK<0> Buffer Strength Control register
15:12	R/W	RASBS	SDRAS Buffer Strength Control register
11:8	R/W	CASBS	SDCAS Buffer Strength Control register
7:4	R/W	MDHBS	MD<31:16> Buffer Strength Control register
3:0	R/W	MDLBS	MD<15:0> Buffer Strength Control register

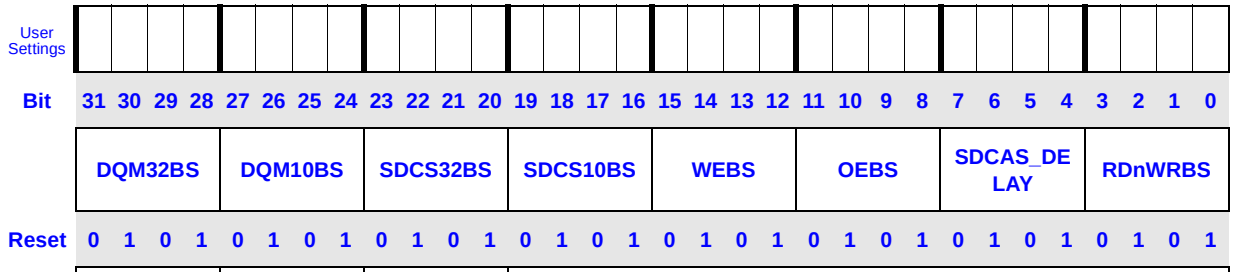
6.5.7.2 System Memory Buffer Strength Control Register 1 (BSCNTR1)

BSCNTR1, shown in Table 6-41, contains control bits for configuring the buffer strengths for the system-memory output buffers.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 6-41. BSCNTR1 Bit Definitions

Physical Address 0x4800_0050	BSCNTR1	Memory Controller



Bits	Access	Name	Description
31:28	R/W	DQM32BS	DQM<3:2> Buffer Strength Control register
27:24	R/W	DQM10BS	DQM<1:0> Buffer Strength Control register
23:20	R/W	SDCS32BS	SDCS<3:2> Buffer Strength Control register
19:16	R/W	SDCS10BS	SDCS<1:0> Buffer Strength Control register
15:12	R/W	WEBS	nWE Buffer Strength Control register
11:8	R/W	OEBS	nOE Buffer Strength Control register
7:4	R/W	SDCAS_ DELAY	SDCAS Return Signal Timing Delay 0x5—average timing delay All other values are not valid. **NOTE:** Do not use this bit field to control the buffer strength for SDCAS. This bit field is for signal return clock timing only.
3:0	R/W	RDnWRBS	RDnWR Buffer Strength Control register

6.5.7.3 System Memory Buffer Strength Control Register 2 (BSCNTR2)

BSCNTR2, shown in Table 6-42, contains control bits for configuring the buffer strengths for the system-memory output buffers.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 6-42. BSCNTR2 Bit Definitions

Physical Address 0x4800_005C																BSCNTR2								Memory Controller							
User Settings																															
Bit	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
	CS5BS				CS4BS				CS3BS				CS2BS				CS1BS				CS0BS				CLK3BS				MA25BS			
Reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	?	?	?	?	0	1	0	1

Bits	Access	Name	Description
31:28	R/W	CS5BS	nCS<5> Buffer Strength Control register
31:24	R/W	CS4BS	nCS<4> Buffer Strength Control register
23:20	R/W	CS3BS	nCS<3> Buffer Strength Control register
19:16	R/W	CS2BS	nCS<2> Buffer Strength Control register
15:12	R/W	CS1BS	nCS<1> Buffer Strength Control register
11:8	R/W	CS0BS	nCS<0> Buffer Strength Control register
7:4	R/W	CLK3BS	SDCLK<3> Buffer Strength Control Bits
3:0	R/W	MA25BS	MA<25> Buffer Strength Control register

6.5.7.4 System Memory Buffer Strength Control Register 3 (BSCNTR3)

BSCNTR3, shown in Table 6-43, contains control bits for configuring the buffer strengths for the system-memory output buffers.

This is a read/write register.

Table 6-43. BSCNTR3 Bit Definitions

Physical Address 0x4800_0060 — BSCNTR3 — Memory Controller

User Settings

Bit	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
	MA24BS				MA23BS				MA22BS				MA21BS				MA2010BS				MA92BS				MA1BS				MA0BS			
Reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Bits	Access	Name	Description
31:28	R/W	MA24BS	MA<24> Buffer Strength Control register
27:24	R/W	MA23BS	MA<23> Buffer Strength Control register
23:20	R/W	MA22BS	MA<22> Buffer Strength Control register
19:16	R/W	MA21BS	MA<21> Buffer Strength Control register
15:12	R/W	MA2010BS	MA<20:10> Buffer Strength Control register
11:8	R/W	MA92BS	MA<9:2> Buffer Strength Control register
7:4	R/W	MA1BS	MA<1> Buffer Strength Control register
3:0	R/W	MA0BS	MA<0> Buffer Strength Control register

6.6 Register Summary

Table 6-44 shows the registers associated with the memory interface and the physical addresses to access them. These registers must be mapped as non-cacheable and non-bufferable and must be accessed only with word accesses. Group the registers together within one page of the MMU table so that they have the same memory protections.

Table 6-44. Memory Controller Register Summary

Address	Name	Description	Page
0x4800_0000	MDCNFG	SDRAM Configuration register	6-43
0x4800_0004	MDREFR	SDRAM Refresh Control register	6-52
0x4800_0008	MSC0	Static Memory Control register 0	6-62
0x4800_000C	MSC1	Static Memory Control register 1	6-62
0x4800_0010	MSC2	Static Memory Control register 2	6-62
0x4800_0014	MECR	Expansion Memory (PC Card/CompactFlash) Bus Configuration register	6-78
0x4800_0018	—	reserved	—
0x4800_001C	SXCNFG	Synchronous Static Memory Configuration register	6-57
0x4800_0024	—	reserved	—
0x4800_0028	MCMEM0	PC Card Interface Common Memory Space Socket 0 Timing Configuration register	6-76
0x4800_002C	MCMEM1	PC Card Interface Common Memory Space Socket 1 Timing Configuration register	6-76
0x4800_0030	MCATT0	PC Card Interface Attribute Space Socket 0 Timing Configuration register	6-76
0x4800_0034	MCATT1	PC Card Interface Attribute Space Socket 1 Timing Configuration register	6-76
0x4800_0038	MCIO0	PC Card Interface I/O Space Socket 0 Timing Configuration register	6-77
0x4800_003C	MCIO1	PC Card Interface I/O Space Socket 1 Timing Configuration register	6-77
0x4800_0040	MDMRS	SDRAM Mode Register Set Configuration register	6-48
0x4800_0044	BOOT_DEF	Boot Time Default Configuration register	6-74
0x4800_0048	ARB_CNTL	Arbiter Control register	29-2
0x4800_004C	BSCNTR0	System Memory Buffer Strength Control register 0	6-80
0x4800_0050	BSCNTR1	System Memory Buffer Strength Control register 1	6-81
0x4800_0054	LCDBSCNTR	LCD Buffer Strength Control register	7-104
0x4800_0058	MDMRSLP	Special Low Power SDRAM Mode Register Set Configuration register	6-50
0x4800_005C	BSCNTR2	System Memory Buffer Strength Control register 2	6-82
0x4800_0060	BSCNTR3	System Memory Buffer Strength Control register 3	6-83
0x4800_0064	SA1110	SA-1110 Compatibility Mode for Static Memory register	6-69
0x4800_0068–0x4800_FFFC	—	reserved	—

LCD Controller 7

The LCD controller provides an interface between the PXA27x processor and a flat-panel display module. The flat-panel display module can be passive (DSTN), active (TFT), or an LCD panel with internal frame buffering.

7.1 Overview

The LCD/flat-panel controller is backward-compatible with Intel® PXA25x and Intel® PXA26x processor LCD controllers. Several additional features are also supported. These additional features are:

- Pixel formats of 18, 19, 24, and 25 bits per pixel (bpp)
  - 18 bpp—RGB 6:6:6
  - 19 bpp—RGBT 6:6:6
  - 24 bpp—RGB 8:8:8, RGBT 8:8:7
  - 25 bpp—RGBT 8:8:8

Note: RGB a:b:c and RGBT a:b:c specify the precision used for the red, green, and blue color components. RGBT uses the most significant bit to indicate transparency for overlay support.

- Base plane with software control of two overlay windows and a hardware cursor
- Color management:
  - Up-scaling for YCbCr 4:2:0 and 4:2:2 to YCbCr 4:4:4
  - Color space conversion CCIR 601—YCbCr 4:4:4 to RGB 8:8:8
  - Conversion from true color, (RGB 8:8:8) to high color (RGB 5:5:5) and the various configurations of RGBT
- Support for LCD panels with an internal frame buffer (smart panels)

7.2 Features

The following list describes features supported by the PXA27x processor LCD controller:

- Display modes
  - Support for single- or dual-scan display modules
  - Passive monochrome mode supports up to 256 gray-scale levels (8 bits)
  - Active color mode supports up to 16777216 colors (24 bits)
  - Passive color mode supports a total of 16777216 colors (24 bits)
  - Support for LCD panels with an internal frame buffer
  - Support for 8-bit (each) passive dual-scan color displays

  - — Support for up to 18-bit per pixel single-scan color displays without an internal frame buffer
  - — Support for up to 24-bit per pixel single-scan color displays with an internal frame buffer
- Support for display sizes from 1x1 to 800 x 600 pixels. See Section 7.4.1.
- 64-entry (by 24 bits) output FIFO
- Three 256-entry by 25-bits internal color-palette RAMs (one for each overlay and Base) programmable to be automatically loaded at the beginning of each frame
- Command data RAM (16 x 9 bits) to hold command data
- Supports pixel depths of 2, 4, 8, 16, 18, and 24 bits per pixel (bpp) in RGB format
- Overlays supported with pixel depths of 16, 19, 24, and 25 bpp in RGBT format
- Provides one base layer plus two overlays for single-scan displays; maximum size of each overlay can equal the display size
- Integrated seven-channel DMA (one channel for base plane, one channel for Overlay 1 and three channels for Overlay 2, one channel for the hardware cursor, and one channel to for the command data)
- Hardware support for color-space conversion from YCbCr to RGB for video streams
- Supports hardware cursor for single-scan display (see Section 7.4.11 for cursor modes and sizes)
- Programmable toggle of AC bias pin output (toggled by line count)
- Programmable pixel clock from 52.0 MHz to 25.4 kHz (104.0 MHz/2 to 13 MHz/512)
- Supports little-endian ordering of pixels in frame buffer
- Programmable wait-state insertion at beginning and end of each line
- Programmable polarity for output enable, frame clock, and line clock
- Programmable interrupts for input and output FIFOs (underrun)
- Six 16 x 64-bit input FIFOs: one for the base channel, one for Overlay 1, three for Overlay 2, and one for the hardware cursor; plus a seventh 4 x 52-bit input FIFO for command data for panels with internal frame buffer
- Backward-compatible with the Intel® PXA25x and Intel® PXA26x processor LCD controllers

7.3 Signal Descriptions

LCD controller signals are listed in Table 7-1. See Section 7.4.16 for detailed information on using the LCD controller data signals LDD<17:0>.

When the LCD controller is disabled, all of its pins can be used for general-purpose input/output (GPIO). Refer to Chapter 24, “General-Purpose I/O Controller” for details.

Table 7-1. LCD Controller I/O Signal Descriptions

Signal Name	Type	Description
LDD<17:0>	Bidirectional	Data lines that transmit 4, 8, 16 or 18 data values at a time to the LCD display module. LDD<7:0> are used as an input data bus during reads from the panel with internal frame buffers.
L_PCLK_WR	Output	Pixel clock used by the LCD display module to clock the pixel data into the Line Shift register. In passive mode, pixel clock toggles only when valid data is available on the data pins. In active mode, pixel clock toggles continuously and the AC bias pin is used as an output to signal when data is valid on the LCD data pins. Write signal for writing to LCD panels with internal frame buffer.
L_LCLK_A0	Output	Line clock used by the LCD display module to signal the end of a line of pixels that transfers the line data from the shift register to the screen and increment the line pointers. Also, it is used by active (TFT) displays as the horizontal synchronization signal. For active displays, this is also referred to as the “horizontal sync signal” or “HSYNC”. It is a control signal that specifies command or data transactions when interfacing to an LCD panel with internal frame buffer.
L_FCLK_RD	Output	Frame clock used by the LCD display module to signal the start of a new frame of pixels that resets the line pointers to the top of the screen. Also, it is used by active (TFT) display module as the vertical synchronization signal. For active displays, this is also referred to as the “vertical sync signal” or “VSYNC”. Read signal during reads from the panel with internal frame buffers.
L_BIAS	Output	AC bias that signals the LCD display module to switch the polarity of the power supplies to the row and column axis of the screen to counteract DC offset. In active (TFT) mode, it is used as the output-enable to signal when data should be latched from the data pins using the pixel clock.
L_CS	Output	For LCD panels with internal frame buffers, this pin is used as chip-select signal.
L_VSYNC	Input	Refresh sync signal from the LCD panel with internal frame buffer.

7.4 Operation

The LCD controller provides a variety of programmable options including display type, resolution, frame buffer, pixel depth, overlays, hardware cursor, and output data formatting. Although all programmable combinations are possible, the selection of displays available dictates which combinations of these programmable options are practical. The type of external memory system used limits the bandwidth of the LCD DMA controller, which, in turn, limits the resolution and type of screen that can be controlled. See Section 7.4.1 to determine the maximum bandwidth of the internal bus that the LCD is allowed to use without negatively affecting all other functions.

Unlike most other peripherals, the LCD controller connects to the PXA27x processor system bus.

7.4.1 Block Diagram

The LCD controller supports a hardware cursor and three image planes, Base, Overlay 1, and Overlay 2. The combination of the three image planes allows multiple images to be displayed simultaneously with software control of window size and position. The simplified top-level block diagram for the LCD controller is illustrated in Figure 7-1. The palette, frame, cursor, and command data utilize a dedicated DMA controller that provides seven channels for fetching the appropriate data from memory into associated input FIFOs. The DMA channel allocation is summarized in Table 7-2.

Figure 7-1. LCD Controller Block Diagram

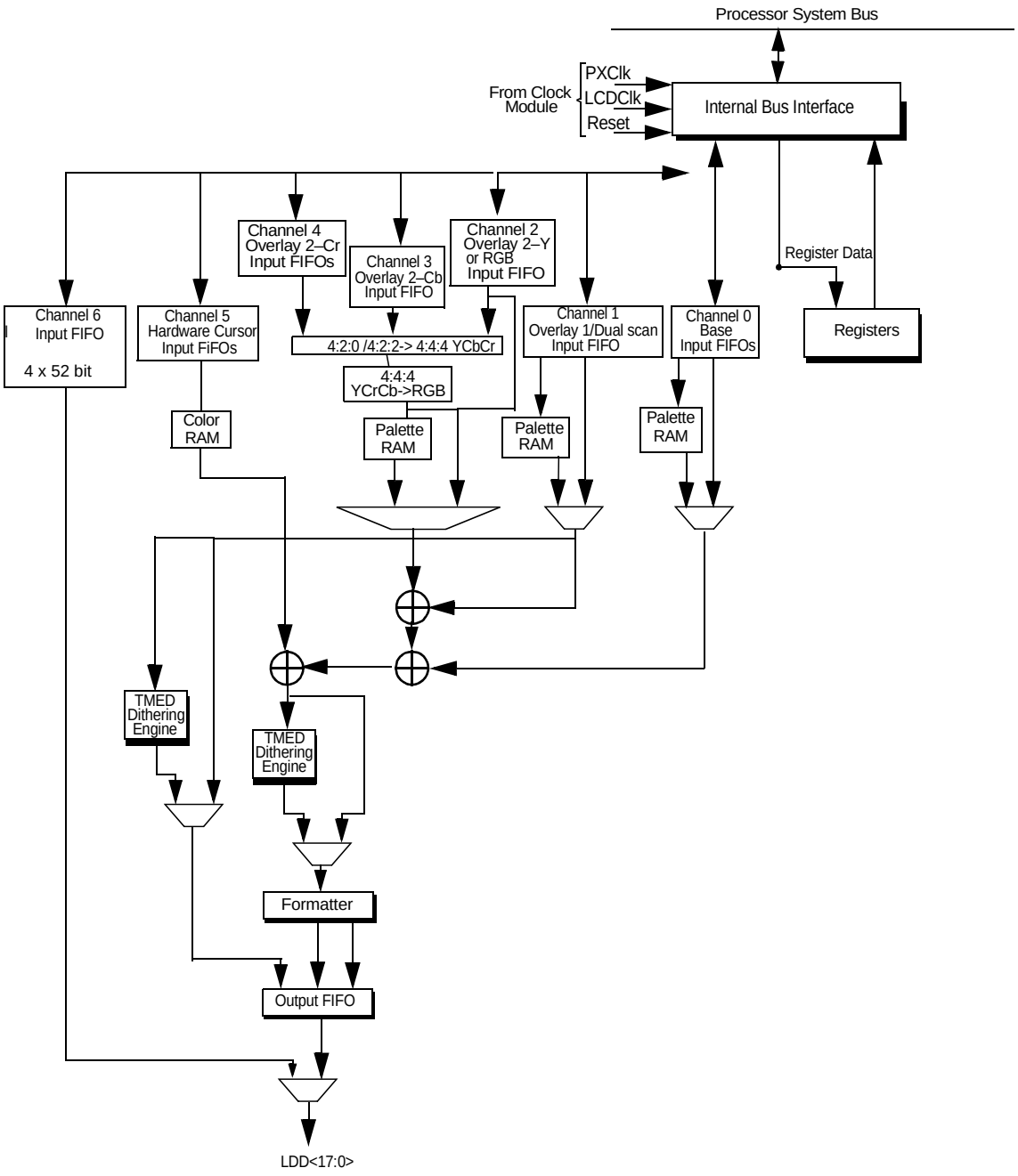


Table 7-2. DMA Channel Use

Channel #	Comments
0	**Single Scan:** Fetches frame and the palette data from memory for the Base frame **Dual Scan:** Fetches upper half of frame data.
1	**Single Scan:** Fetches frame and the palette data from memory for Overlay 1. **Dual Scan:** Fetches lower half frame data
2, 3, 4	**Single Scan:** Fetches frame and palette data from memory for Overlay 2. **Dual Scan:** Not used.
5	**Single Scan:** Fetches pixel and color RAM data from memory for hardware cursor. **Dual Scan:** Not used.
6	**Single Scan:** Fetches command data from the memory for command RAM. **Dual Scan:** Not used.

The frame and the palette data for the base and overlay planes are fetched by DMA channels (0–4) over the internal system bus into separate dedicated 16x64-bit input FIFOs. The Base and Overlay 1 planes each have a single dedicated FIFO and support pseudo-color as well as several standard RGB formats. The Overlay 2 plane uses three input FIFOs and accepts both RGB and YCbCr formats. The frame and palette data can be stored either in internal SRAM or external memory.

The LCD controller supports mapping frame data to palette entries when 2-, 4-, or 8-bpp formats are used. For pixel depths greater than 8 bpp, the palette RAM is bypassed and each of the red, green, and blue color components are combined appropriately when the overlays are enabled. The pixel format of 2 bpp is valid only for the Base plane and is available only when the overlays are disabled. The palette RAM maps the 2-, 4- or 8-bpp formats to 16- or 25-bit values. Three separate 256x25-bit palette RAMs are associated with the Base, Overlay 1, and Overlay 2 planes.

The data for the Base, Overlay 1, Overlay 2, and cursor are combined to go through the dither engine for passive displays, or directly to the output FIFO for active displays. The data output from the dither logic is grouped into the selected format (such as 8-bit color, dual scan, 16-bit color) and placed in the output FIFO. The data from the output FIFO is driven onto the LCD data pins.

There are eight modes of hardware cursor formats described in Section 7.4.11. The pixel data for the cursor is fetched from internal SRAM or external memory by DMA channel 5 over the internal system bus into its associated 16x64-bit input FIFO. The pixel data indexes into a 4x24-bit color RAM (a small palette) to get 24-bit pixel color. The color RAM holds the 24-bit colors used by the cursor and can optionally be loaded for each frame.

The LCD controller supports LCD panels with internal frame buffers. The command data RAM hold the commands to be sent at the beginning of each frame. The 4x52-bit command FIFO is loaded from internal SRAM or external memory by DMA channel 6 over the internal system bus.

The LCD controller supports both single- and dual-scan displays. The LCD controller does not support overlays and hardware cursor for dual-scan displays. The frame data for the upper panel is fetched with DMA channel 0, and channel 1 fetches the frame data for lower panel. The palette data for both the upper and lower panels uses channel 0.

Note: Dual-scan displays are passive displays.

Depending on the type of panel used, the LCD controller is programmed to use 4-, 8-, 16-, or 18-bit-output data pins. Single-scan monochrome displays use either 4 or 8 data pins to output 4 or 8 pixels for each pixel clock; single-scan color displays use 8, 16 or 18 output-data pins. Single-scan passive color displays use 8 pins to output 2-2/3 pixels each pixel clock (8 pins/3 colors/pixel = 2 2/3 pixels per clock). Single-scan active color displays use 8, 16 or 18 output-data pins.

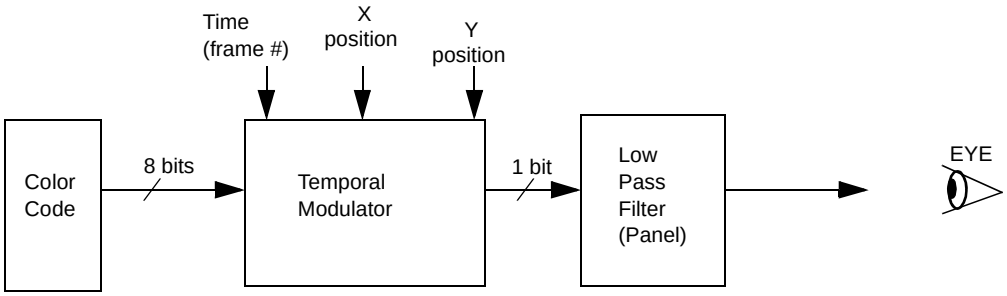
For dual-scan displays, the LCD controller splits the output data lines into two groups: one to drive the top half of the screen, and one to drive the bottom half. For dual-scan displays, the number of pixel data output pins is doubled, allowing twice as many pixels to be output for each pixel clock to both halves of the screen.

Displays with an internal frame buffer are always driven with eight output-data pins, with the data for the color data of one pixel driven each cycle of L_PCLK_WR.

7.4.1.1 Temporal-Modulated Energy-Distribution (TMED) Dithering

Temporal-modulated energy-distribution (TMED) dithering in the LCD controller is shown in Figure 7-1. For passive displays, entries selected from the lookup palette (or directly from memory for all pixel depths greater than 8 bits per pixel) are sent to the TMED dithering engine. TMED is a form of temporal dithering; pixels are run through an algorithm to determine if the pixel should be on or off (modulated over time). See Figure 7-2 for a high-level diagram.

Figure 7-2. Temporal Dithering Concept

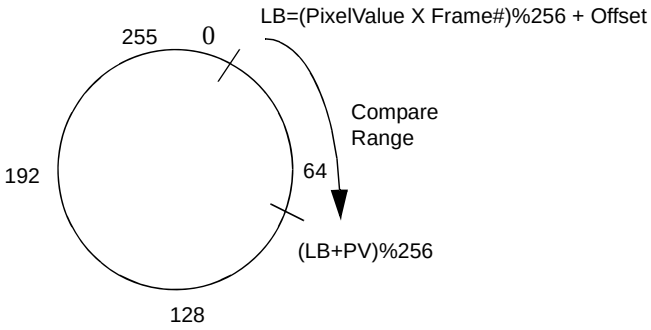


The LCD controller implements the following algorithm, which is used by TMED to determine an upper and lower boundary.

```
LowerBoundary = [(PixelValue X FrameNumber) mod256] + Offset
UpperBoundary = [(PixelValue + LowerBoundary) mod256]
```

A 16-x-16 matrix uses the row (line), column (pixel number), and frame number (which wraps back to 0 from 255) to select a matrix value. When the matrix value is between the lower and upper boundaries from the algorithm, the LCD controller sends a 1 to the flat panel. The boundaries created by the algorithm are circular; they wrap around from 255 to 0 (see Figure 7-3).

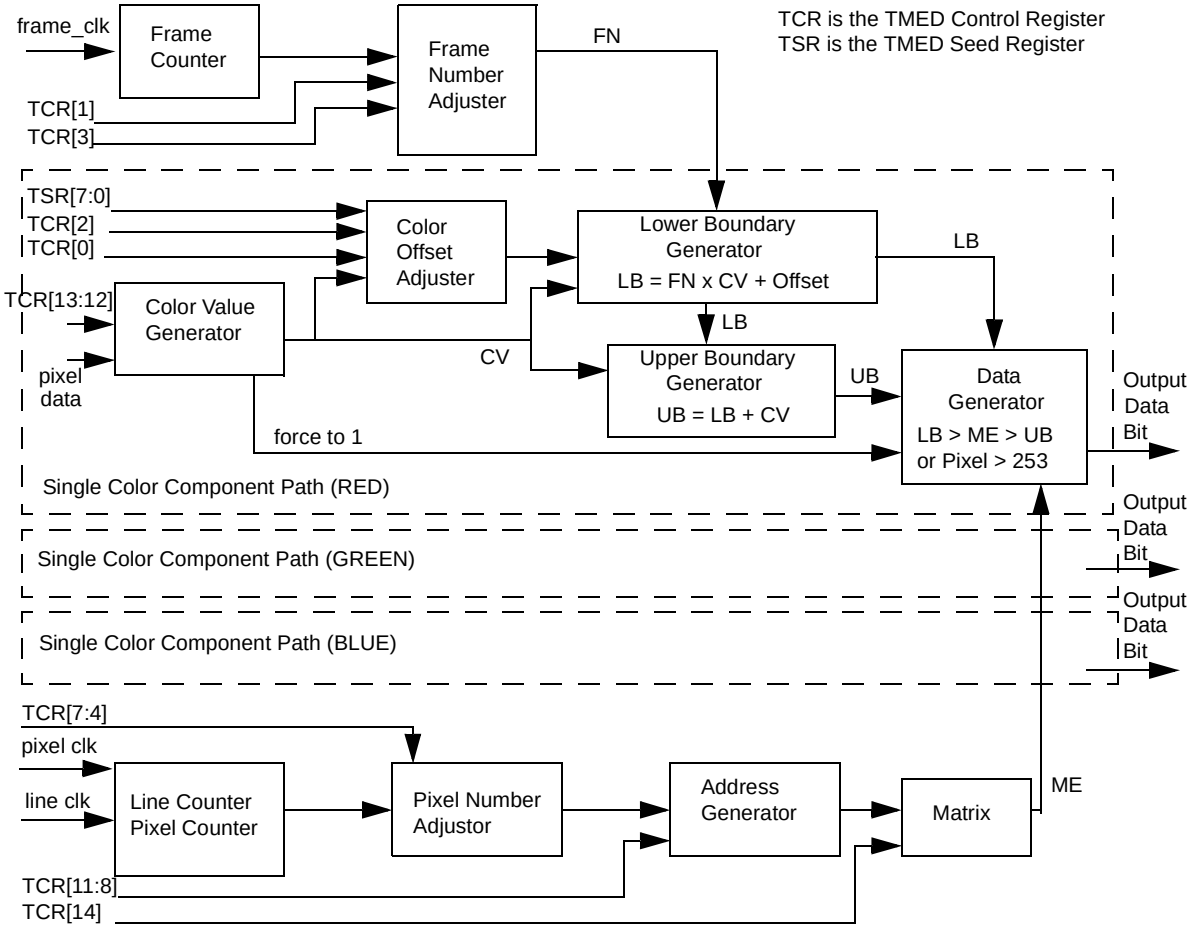
Figure 7-3. Compare Range for TMED



Each color shade (RGB) progresses independently (has its own matrix). Software has a choice of two matrices to use for each color, chosen by TCR[TM2S, TM1S, TED]. Offsets for the shading of each color can be selected by software to avoid gray problems. Although this value may be somewhat panel-dependent, the recommended values are listed in Section 7.5.14, with the blue data path being used for monochrome modes. Software can also choose to use offsets for shifting the row (horizontal), line (vertical) values, and frame number (see Section 7.5.15). The block diagram for TMED is shown in Figure 7-4. Pixel data (up to 8 bit) enters the module and is sent through the color value generator. In the CV generator, it is rounded off (the 1, 2, or 3 lowest bits are not rounded off at all, as chosen by TCR[TSCS]) to create a new color value. If the original pixel value is 254 or 255, the final data out is forced to a 1; otherwise, the following occurs.

The new color value is sent through the color offset adjuster where it is used as a lookup into one of two matrices (selected by TCR[TM1S]). The 8-bit value output of the chosen matrix or simply 00h (selected by TCR[TM2EN]) is added to the appropriate color seed register (TRGBR) value to form an offset. This offset is added to the result of the multiplication of the frame number to the color value to form the algorithm's lower boundary (only lower 8 bits used). The color value is added to the lower boundary to obtain the upper boundary. Row (line) and column (pixel) counters are combined with beat suppression (offset) values in the pixel number adjuster and address generator to form yet another address for a matrix lookup. TCR[TED] chooses which matrix to use. The output of the chosen matrix is then compared to the lower and upper boundaries in the data generator. If the matrix output is between those boundaries or the original pixel value is 254 or 255, then the data output sent to the panel is a 1; otherwise it is a 0.

Figure 7-4. TMED Block Diagram



7.4.1.2 Input FIFOs

As shown in Figure 7-1, there are seven input FIFOs, one 16x64-bit FIFO for base layer, one 16x64-bit FIFO for Overlay 1, three 16x64-bit FIFOs for Overlay 2, one 16x64-bit FIFO for the cursor, and one 4x64-bit FIFO for the command RAM. Frame data is fetched from the frame buffer by the dedicated LCD DMA controller and is placed in one of seven input FIFO buffers in the LCD controller. Palette data is loaded optionally at the beginning of each frame. This data goes through the FIFO and is written to the internal palette RAM at the output of the FIFO. Following the palette fetch, the first FIFO is filled with encoded pixels. A 7-bit counter is loaded each time the DMA begins to fetch palette RAM data and is decremented each time a word is stored to the palette (two palette entries). When the counter wraps around to zero, the palette is loaded and pixel processing begins.

The FIFO signals a service request to the DMA whenever four entries of the FIFO are empty. Pixel data remains packed within individual 64-bit words when it is loaded into the FIFO. The LCD controller port size is 64 bits wide to accommodate the heavy data flow from the frame buffer. If the pixel size is 2, 4, or 8 bits, the FIFO entries are unpacked and used to index the palette RAM to

read the color value. For 16-, 18-, 19-, 24-, or 25-bit pixels, if passive, the entries bypass the palette and go directly to the TMED dither logic; in active (TFT) mode the pixels are output directly to the pins.

7.4.1.3 Lookup Palette

Color palette RAMs for base and overlays are each 25-bit wide x 256 locations. Sixteen, 18, or 24 bits of the data is used, depending on the data width (number of bits per pixel). Monochrome entries are 8 bits wide. Bit 25 defines the transparency of the pixel. The encoded pixel data taken from the bottom entry of the input FIFO is used as an address to index and select individual palette locations. Two-bit pixel encodings address the top four entries, 4-bit pixel encodings address 16 locations, and 8-bit pixel encodings select any of the 256 palette entries. The palette RAM is bypassed for pixel depth greater than 8 bpp.

7.4.1.4 Output FIFO

The LCD controller contains a 64-entry x 24-bit-wide output FIFO that stores pixel pin data before it is driven out to the pins. Each time a modulated pixel value is output from the dither generator, it is placed into a serial shifter. The size of the shifter is controlled by programming the color/monochrome select and single- and dual-scan, double pixel data, and passive/active select bits in the LCD Control registers, and the pixel bit size in the frame descriptor in memory. The shifter can be configured to be 4, 8, 16, 18, or 24 bits wide. Once the correct number of pixels have been placed within the shifter (4-, 8-, 16-, 18-, or 24-bit pixel values), the value is transferred to the top of the output FIFO. The value is then transferred down until it reaches the last empty location within the FIFO. Each time a value is taken from the bottom of the FIFO, the entry is invalidated and all data in the FIFO moves down one position.

7.4.2 Bandwidth Calculations

The LCD bandwidth formulas presented in this section provide a way to determine if any given LCD and system configuration will function without visible video artifacts and the amount of time needed for the LCD refresh operation. The timing values obtained can be used to determine the maximum LCD refresh rate for any given LCD and system timing configuration, and the remaining system task memory-bus bandwidth.

The LCD data rate required for each plane to support the LCD panel selected for the system is calculated using this formula:

$$\text{Data Rate} = \left(\frac{\text{Length} \times \text{Width} \times \text{Refresh Rate} \times \text{Bits per Pixel}}{8}\right) \text{ bytes/sec}$$

The *bits per pixel* is the number of bits used in the memory to store each pixel. For example: 16 bits for a pixel depth of 16 BPP, 32 bits for a pixel depth of 18 BPP unpacked, and 24 bits for a pixel depths of 18 BPP packed. See Section 7.4.13 to derive the bits-per-pixel value.

The number of 4-beat burst operations (8 bytes/beat) that are generated by the LCD DMA controller is as follows:

$$\text{LCD DMA burst Count} = \left(\frac{\text{Data Rate}}{32}\right) \text{ Burst/sec}$$

The time consumed by the LCD refresh operation is then calculated by:

$$\text{LCD refresh time} = (\text{LCD DMA burst count} \times \text{Pdma}) / \text{second}$$

The value of Pdma is the period in microseconds of LCD DMA four-beat burst, including DRAM or SDRAM precharge time. The time remaining within each second after the LCD refresh time is deducted is available for instruction and data fetches, hardware accesses, and memory refresh operations.

Be careful when setting system parameters, such as core frequency, system frequency, memory frequency, and bus arbiter settings, such that LCD FIFOs do not underrun due to bus latencies caused by other internal and external peripherals. This is the case especially for interrupt and polled modes that require a longer time to service.

7.4.3 Pixel Clock Frequency Calculation

For passive and active displays, the pixel clock divider field (LCCR3[PCD]), along with the PCD divisor select (LCCR4[PCDDIV]), configures the frequency of the pixel clock. LCCR3[PCD] can be any value from 0 to 255 and generates a range of pixel clock frequencies from LCLK/2 to LCLK/512 or from LCLK/2, where LCLK is the programmed frequency of the LCD/memory controller. LCLK can vary from 26.0 MHz to 104.0 MHz for normal mode operation. When operating in deep-idle mode or normal mode with the PLLs disabled, the LCLK operates at 13.0 MHz. Additionally, in deep-idle mode only, LCLK can be configured to operate at 26 MHz (instead of 13 MHz) by setting CCCR[26MHz]. Refer to Table 3-31, “CCCR Bit Definitions” on page 3-96 for more information.

The pixel-clock frequency is adjusted to meet the required screen-refresh rate. The refresh rate depends on several factors:

- The number of pixels for the target display
- Whether single- or dual-scan mode is selected
- Whether monochrome or color mode is selected
- The number of pixel-clock wait states programmed at the beginning and end of each line
- The number of line clocks inserted at the beginning and end of each frame
- The width of the VSYNC signal in active mode or VSW line clocks inserted in passive mode
- The width of the frame clock or HSYNC signal.

All of these factors alter the time duration between frame transmissions. Different display manufacturers require different frame-refresh rates, depending on the physical characteristics of the display. Use LCCR3[PCD] and LCCR4[PCDDIV] to alter the pixel-clock frequency to meet these requirements.

If LCCR4[PCDDIV] is cleared, the frequency of the pixel clock for a set pixel-clock divider value or the required LCCR3[PCD] value to yield a target pixel-clock frequency can be calculated using the following two equations.

$$PixelClock = \frac{LCLK}{2(PCD + 1)}$$

$$PCD = \frac{LCLK}{2(PixelClock)} - 1$$

If LCCR4[PCDDIV] is set, the frequency of the pixel clock for a set pixel-clock divider value or the required LCCR3[PCD] value to yield a target pixel clock frequency can be calculated using the two following equations.

$$PixelClock = \frac{LCLK}{(PCD + 1)}$$

$$PCD = \frac{LCLK}{(PixelClock)} - 1$$

Note: If double pixel-clock mode (LCCR3[DPC]) is enabled or if LCCR4[PCDDIV] is set, the minimum allowable value of LCCR3[PCD] is 1.

For LCD panels with an internal frame buffer, this bit field specifies the command-inhibit time between two consecutive write accesses to the LCD panel. The command-inhibit time is equal to:

(LCCR3[PCD]+1)*LCD_CLK_PERIOD

7.4.4 Multiple Panel Considerations

In systems that use more than one LCD panel (for instance a “flip-phone” with a small panel on the exterior cover and a larger panel on the inside of the phone), specific procedures must be followed to ensure proper operation when switching the LCD controller between the two panels.

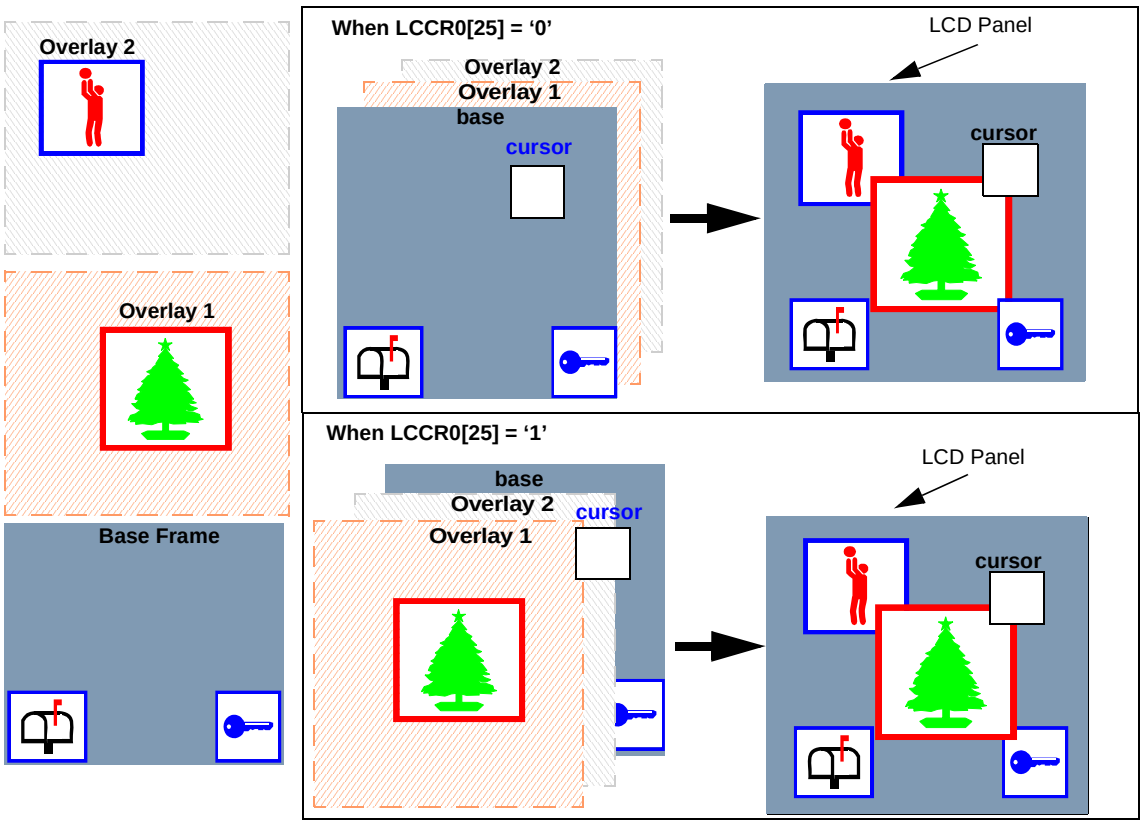
The following procedure is required to switch operation between two LCD panels of different sizes. This procedure is also required to switch operation between two LCD panels where one is a single-scan display and the other is a dual-scan display.

1. Configure the LCD controller for the first display.
2. Enable the LCD controller.
3. When the second display is to be used instead of the first display, disable the LCD controller.
4. Reconfigure the LCD controller for the second display without programming the DMA registers yet.
5. Enable the LCD controller.
6. Perform a quick disable (not normal disable).
7. Program the LCD controller DMA registers for the second display.
8. Enable the LCD controller. The second display is now enabled.

7.4.5 Graphical Overlays

The LCD controller supports a base plane plus two overlays and hardware cursor. Figure 7-5 shows the display of hardware cursor, base frame, and two overlays on the LCD panel.

Figure 7-5. Hardware Cursor, Base Plus 2 Overlays Displayed on LCD Panel



NOTE:

Dashed area of overlays demonstrate the position and size of the base plane. The dashed areas are not defined in the overlay frame buffers in this example.

The display order of the cursor, base frame, and overlays is shown in Table 7-3.

Table 7-3. Display Order of Three Layers and Cursor on LCD Panel

Layer Number	Plane
LCCR0[25] = 0b0:	
TOP or 1st	Hardware Cursor
2nd	Base Plane
3rd	Overlay 1 window
4th	Overlay 2 window
LCCR0[25] = 0b1:	
TOP or 1st	Hardware Cursor
2nd	Overlay 1 Window
3rd	Overlay 2 Window
4th	Base Plane

7.4.5.1 Transparency

A pixel of a particular plane is said to be *fully transparent* if the pixel in the layers below is to be displayed instead. For pixel depths of 4 bpp and 8 bpp, the transparency is defined by the 25th bit of each of the palette RAM location. For pixel depths greater than 8 bpp, the transparency of the pixel is indicated by the most significant bit within each pixel value. If the transparency bit is set and the data field contains all zeros, the pixel is fully transparent. When LCCR0[OUC] is set, the Overlay 2 transparency is defined by the transparency bit for RGB format and the Overlay 2 is not transparent for YCbCr format. If the transparency bit is set and the data field is a non-zero value, the pixel is half transparent.

A pixel of a particular plane is said to be *half transparent* if some logical blend of all the pixels of all the planes at that location are displayed. For pixel depths of 4 bpp and 8 bpp, half transparency is indicated by setting the 25th bit of each palette RAM location, and the data field being a non-zero number. For pixel depths greater than 8bpp, half transparency is indicated by setting the most significant bit of the pixel to logical 1, with the data field being a non-zero number. When LCCR0[OUC] is cleared, the half transparency applies only for the Base layer; Overlay 1 and Overlay 2 do not support half transparency. When LCCR0[OUC] is set, the half transparency applies only for Overlay 1; Overlay 2 and Base do not support half transparency.

When LCCR0[OUC] = 0b0, half transparency is calculated as follows:

$$Output(R) = Base(R) \times K1 + (Overlay1(R) + Overlay2(R)) \times (1 - K1)$$

$$Output(G) = Base(G) \times K2 + (Overlay1(G) + Overlay2(G)) \times (1 - K2)$$

$$Output(B) = Base(B) \times K3 + (Overlay1(B) + Overlay2(B)) \times (1 - K3)$$

When LCCR0[OUC] = 0b1, half transparency is calculated as follows:

$$Output(R) = Overlay1(R) \times K1 + (Base(R) + OVerlay2(R)) \times (1 - K1)$$

$$Output(G) = Overlay1(G) \times K2 + (Base(G) + Overlay2(G)) \times (1 - K2)$$

$$Output(B) = Overlay1(B) \times K3 + (Base(B) + Overlay2(B)) \times (1 - K3)$$

The variables K1, K2, and K3 are constant values between 0 to 1 in increments of 1/8, and are programmed by writing the LCD Controller Control register 4. See Section 7.5.6.

7.4.6 Pixel Formats

The LCD controller supports basic palettized bitmap formats and several formats that contain raw RGB data. Both the palletized and raw RGB formats also support a format to indicate transparency. Each pixel data for pixel depths 2, 4, and 8 bpp indexes into palette RAM to get the 16- or 25-bit value. The most significant bit of this 25-bit value is the transparency bit, and the lower 16 or 24 bits represent the color components. Pixel depths greater then 8 bpp bypass the palette RAM.

Table 7-4 shows which pixel depths support both base and the overlays and which only support a base layer. Table 7-4 also indicates which combinations are compatible with the Intel® PXA25x and Intel® PXA26x processors.

Each plane can be programmed independently to support one of the allowed pixel format combinations, as shown in Table 7-5, when the data format is RGB. Table 7-5 shows the bits/pixel format combinations allowed for all three planes.

Table 7-4. Overlay Support for Bits per Pixel (BPP) Formats

Bits per Pixel Format	Base with Overlays Disabled	Base with Overlays Enabled
2	†	Not Supported
4	†	†
8	†	†
16	†	†
18	†	Not Supported
19	Not Supported	†
24	†	†
25	Not Supported	†

NOTES:

† Indicates that the combination is supported by the PXA27x processor.
Shaded cells in Table 7-4 are compatible with the Intel® PXA25x and PXA26x processor LCD controllers.

Table 7-5. Bit per Pixel Format Combinations Allowed

Base	Overlay 1	Overlay 2	Resultant RGB Value
4 or 8	4, 8, or width of base palette	4, 8, or width of base palette or YCbCr	Width of the palettes defined
16	4, 8, or 16	4, 8, 16 or YCbCr	16 bits
18	4, 8, or 18	4, 8, 18 or YCbCr	18 bits
19	4, 8, or 19	4, 8, 19 or YCbCr	18 bits
24	4, 8, or 24	4, 8, 24 or YCbCr	24 bits
25	4, 8, or 25	4, 8, 25 or YCbCr	24 bits
16 RGBT	24-bit mode†	24-bit YCbCr	16 bits for panels without internal frame buffers.†† 24 bits for panels with internal frame buffers

NOTES:

† Refer to OVL1C1[BPP1] for a description of this mode.

†† Refer to Section 7.4.9.3 for a description of the method of conversion from 24 bits to 16 bits.

The LCD controller also supports 4:2:0, 4:2:2 and 4:4:4 YCbCr video sampling formats for the Overlay 2 plane. The color space conversion from YCbCr to RGB is done based on CCIR 601 standard. The YCbCr formats supported are provided in Table 7-6.

Table 7-6. Video Sampling Formats Supported by Overlay 2

YCbCr Format	Comments
4:4:4	The color space conversion from 4:4:4 YCbCr to RGB 8:8:8
4:2:2	Video data in 4:2:2 format is converted to 4:4:4 prior to color space conversion to RGB 8:8:8. OVL2C1[PPL2] must be > 1
4:2:0	Video data in 4:2:0 format is converted to 4:4:4 prior to color space conversion to RGB 8:8:8. OVL2C1[PLL2] must be > 1 and OVL2C1[LPO2] must be > 1

Table 7-7 shows the valid pixel formats for each base/overlay frame.

Table 7-7. Valid Pixel Formats for Each Frame

Frame	Color/Mono	Valid Bpp: Overlays Enabled	Valid Bpp: Overlays Disabled	Valid Bpp: Dual Scan Passive
Base	Color	4, 8, 16, 19, 24, 25	2, 4, 8, 16, 18, 24	2, 4, 8, 16, 18, 24
	Mono	4, 8	2, 4, 8	2, 4, 8
Overlay1	Color	4, 8, 16, 19, 24, 25, 24-bit mode	NA	NA
	Mono	4, 8	NA	NA
Overlay2	Color	4,8,16,19,24,25 YCbCr4:4:4, YCbCr4:2:2, YCbCr4:2:0	NA	NA
	Mono	4, 8	NA	NA

Use LCCR3[PDFOR] and LCCR4[PAL_FOR] to configure the pixel formats for the base layer. PDFOR configures the pixel data format that the LCD frame buffer represents for all pixel formats of 16bpp or larger. Not all values of PDFOR are valid for all pixel depths. Valid selections are specified in Section 7.4.6.2.1 through Section 7.4.6.2.4.

Pixel depths less than 16BPP require a palette. PAL_FOR configures the palette format for the base plane. Four formats are possible. Refer to Table 7-44 and Table 7.4.12.1 for details on configuring the palette.

Table 7-8 shows the valid PAL_FOR and PD_FOR for various base pixel formats.

Table 7-8. Valid Combinations of PDFOR and PAL_FOR for Various Base BPP

Base BPP	PAL_FOR	PDFOR with Overlays Enabled	PDFOR with Overlays Disabled	Notes
2	0	NA	0	For 16 bpp panels No transparency
2	2	NA	3	For 18 bpp panels No transparency
2	3	NA	3	For smart panels No transparency
4,8	0	NA	0	For 16 bpp panels No transparency
4,8	1	3	NA	For 16 bpp panels. Transparency required
4,8	2	0† LCCR3[BPP] = 0x7 or 0x8	3 LCCR3[BPP] = 0x5 or 0x6	For 18 bpp panels Transparency optional
4,8	3	0† LCCR3[BPP] = 0xA	3 LCCR3[BPP] = 0x9	For smart panels Transparency optional
16	0†	3	0	For 16 bpp panels Transparency optional
18	0†	NA	3	For 18 bpp panels No transparency
19	0†	0†	NA	For 18 bpp panels Transparency required
24	0†	2	3	For smart panels. Transparency optional
25	0†	0†	NA	For smart panels Transparency required

NOTES:

† Indicates that the value of this field is ignored for this mode. The use of the reset value of 0b00 is recommended; however, any value is acceptable.

†† If a palette is used for the base plane, the pixel data in the frame buffer format is that shown in Figure 7-17 through Figure 7-20 (not the format specified by PDFOR).

Note: For pixel depths where only one RGB format is possible (19 bpp and 25 bpp), the value of PDFOR is ignored.

7.4.6.1 Data Format for Pixel Depths of 2, 4 and 8 bpp

The palette RAM (internal to the LCD controller) provides for mapping the 2, 4, or 8 bpp formats to a color value defined by the palette. See Section 7.4.12.1 for detailed information regarding the format of the data within the palette buffer. The palette RAM must be loaded at least once before the pixel data can be displayed on the LCD panel. The palette RAM is loaded from the palette buffer—a separate, small frame buffer used for the palette. Figure 7-9 describes the number of entries and formats for the possible palletized formats.

If transparency is used, the most significant bit determines the transparency and the lower 8- or 24-bits represent the red, green, and blue color components. Load the palette RAM such that the 25th bit of each location defines transparency. Ensure that the 25th bit is clear, if transparency is not intended. Figure 7-18 details the format of the pixel data when overlays are used.

Table 7-9. Palette Entries for 2, 4, and 8 bpp Formats

Palletized Format	Palette Entries	Comments
2 bpp	4	Overlays and hardware cursor disabled, RGB 6:6:6, RGB 5:6:5, RGB 8:8:8 format (16bpp, 18bpp and 24 bpp).
4 bpp	16	Transparency supported if overlays enabled, at 16bpp, 18bpp, 19bpp, 24bpp, and 25bpp.
8 bpp	256	Transparency supported if overlays enabled, at 16bpp, 18bpp, 19bpp, 24bpp, and 25bpp.

7.4.6.2 Data Format for RGB Color Space

The LCD controller supports several standard pixel depths for the RGB color space. *Pixel depth* corresponds to the total number of bits required for representing the red, green, and blue color components with a transparency bit, if applicable. When the overlays are disabled, the pixel depth corresponds directly to the number of bits used to represent the color components. When the overlays are enabled, the most significant bit indicates transparency. The two formats are referred to as *RGB* and *RGBT*, respectively, with the integer fields representing the number of bits used to represent each color channel. The supported RGB and RGBT formats are listed in Table 7-10.

Table 7-10. Standard RGB and RGBT Formats

Memory Depth	RGB Formats: Overlays Disabled	RGB Formats: Overlays Enabled
16-bits	RGB 5:6:5	RGBT 5:5:5
18-bits	RGB 6:6:6	NA
19-bits	NA	RGBT 6:6:6
24-bits	RGB 8:8:8	RGBT 8:8:7, RGB 8:8:8†
25-bits	NA	RGBT 8:8:8

NOTE:
† RGB 8:8:8 is called 24-bit mode and is only possible with Overlay 1.

7.4.6.2.1 Data Format for Pixel Depth of 16 bpp

The 16-bpp format supports two modes. In the first mode, the overlays are disabled and a true 16-bpp format in which 65536 possible colors are provided. The second mode is used when the overlays are enabled. In this mode, the most significant bit indicates transparency, with the remaining 15 bits used for the color components.

The RGB 5:6:5 format is supported when the overlay planes are disabled. This format is illustrated in Table 7-11. The value of the PDFOR bit field must be cleared to 0b00, as described in Table 7-43.

Table 7-11. Pixel Depth of 16 bpp with Overlays Disabled

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDFOR 00 Format 1	Red [4:0]					Green [5:0]						Blue [4:0]				

The RGBT 5:5:5 format is provided when one or both of the overlays are enabled. Each of the color components uses five bits with the most significant bit in the 16-bit word specifying transparency. The RGBT 5:5:5 format is illustrated in Table 7-12.

Table 7-12. Pixel Depth of 16 bpp with Overlays Enabled

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDFOR 11 Format 4	T	Red [4:0]					Green [4:0]					Blue [4:0]				

NOTES:
1. When Bit T (pixel data bit 15) is set and pixel data is non-zero, half transparency mode is enabled.
2. When pixel data = 0x8000, full transparency mode is enabled.

7.4.6.2.2 Data Format for Pixel Depth of 18 bpp

The 18-bpp format is used when both overlays are disabled. The 18 bits allow 262144 possible colors with six bits each of red, green and blue. The RGB 6:6:6 format is shown in Table 7-13.

Table 7-13. Pixel Depth of 18 bpp with Overlays Disabled

Bit	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDFOR 11 Format 4	Red[5:0]						Green[5:0]						Blue[5:0]					

7.4.6.2.3 Data Format for Pixel Depth of 19 bpp

The 19-bpp format is used when one or both of the overlays are enabled. The 19 bits represent six bits for each of the red, green, and blue components with the most significant bit to indicate transparency as shown in Table 7-14.

Table 7-14. Pixel Depth of 19 bpp with Overlays Enabled

Bit	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	T	Red [5:0]						Green [5:0]						Blue [5:0]					

NOTES:
1. When Bit T (pixel data bit 18) is set and pixel data is non-zero, half transparency mode is enabled.
2. When pixel data = 0x4_0000, full transparency mode is enabled.

7.4.6.2.4 Data Format for Pixel Depth of 24 bpp

The 24-bpp format supports three modes. In the first mode, the overlays are disabled and a true 24-bpp format in which 16777216 possible colors are provided. The other two modes are used when overlays are enabled.

The RGB 8:8:8 format is provided when the overlays are disabled and represents eight bits of red data, eight bits of green data, and eight bits of blue data as shown Table 7-15.

Table 7-15. Pixel Depth of 24 bpp with Overlays Disabled

Bit	23 … 16	15 … 8	7 … 0
PDFOR 11 Format 4	Red[7:0]	Green[7:0]	Blue[7:0]

Transparency is enabled by either of two methods. In the first of these modes, the most significant bit indicates transparency with the remaining 23-bits used for the color components. The other mode is unique to Overlay 1 only. When OVL1C1[BPP1] is configured to 0x0, Overlay 1 is configured for 24bpp mode with transparency enabled, but with 8 bits of red, 8 bits of green, and 8 bits of blue. In this mode, the transparency bit (T) does not exist, but transparency is enabled, which allows for greater precision in the overlay color. These formats are shown in Table 7-16.

The RGBT 8:8:7 format is supported when the overlays are enabled and represent eight bits of red data, eight bits of green data, and seven bits of blue data as shown Table 7-16.

Table 7-16. Pixel Depth of 24 bpp with Overlays Enabled

Bit	23	22 … 17	16 … 15	14 … 8	7 … 6	0
PDFOR 10 Format 3[1,2]	T	Red [7:0]		Green [7:0]		Blue [6:0]
24 Bit Mode[3]	Red [7:0]		Green [7:0]		Blue [7:0]	

NOTES:
1. When Bit T (pixel data bit 23) is set and pixel data is non-zero, half transparency mode is enabled.
2. When pixel data = 0x80_0000, full transparency mode is enabled.
3. This format is only available for Overlay 1. The value of PDFOR is ignored for this mode. OVL1C1[BPP1] = 0x0 for this mode.

7.4.6.2.5 Data Format for Pixel Depth of 25 bpp

The 25-bpp format is used when one or both of the overlays are enabled. The 25 bits represent eight bits for each of the red, green, and blue components with the most significant bit to indicate transparency as shown in Table 7-17.

Table 7-17. Pixel Depth of 25 bpp with Overlays Enabled

Bit	24	23 … 16	15 … 8	7 … 0
	T	Red [7:0]	Green [7:0]	Blue [7:0]

NOTE:
1. When Bit T (pixel data bit 24) is set and pixel data is non-zero, half transparency mode is enabled.
2. When pixel data = 0x100_0000, full transparency mode is enabled.

7.4.6.3 Data Format for YCbCr Color Space

The YCbCr video sampling format is used in Overlay 2. In this format, luminance information is stored as a single component (Y), and chrominance information is stored as two color-difference components (Cb and Cr). Cb represents the difference between the blue component and a reference value. Cr represents the difference between the red component and a reference value. Y is defined to have a nominal range of 16 to 235; Cb and Cr are defined to have a range of 16 to 240, with zero signal corresponding to level 128.

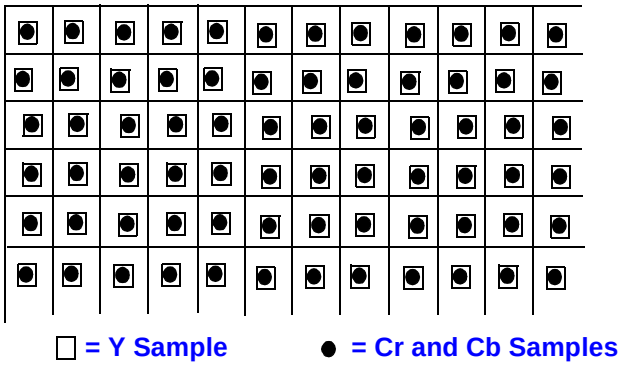
The LCD controller supports the following three YCbCr video formats:

- 4:4:4 YCbCr sampling format
- 4:2:0 YCbCr sampling format
- 4:2:2 YCbCr sampling format

7.4.6.3.1 4:4:4 YCbCr Video Format

In 4:4:4 YCbCr video format, within each video frame, the number of samples of each chrominance component (Cr or Cb) is the same as that of the number of samples of luminance, both horizontally and vertically. In other words, for every sample of luminance, two samples of chrominance, one a Cr sample and the other a Cb sample, exist. In a frame of 4:4:4 YCbCr video, the locations of chrominance samples is the same as that of luminance samples, as shown in Figure 7-6.

Figure 7-6. Luminance and Chrominance Samples in 4:4:4 YCbCr Video Frame

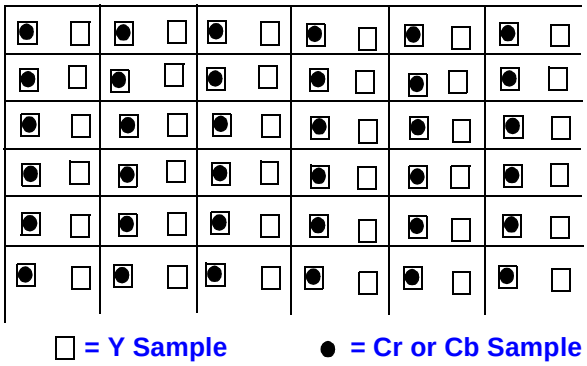


The luminance and chrominance values are 8 bits each. The PXA27x processor supports storage of 4:4:4 YCbCr video format in planar or packed format in the frame buffer. The frame buffer can reside in internal SRAM or external memory. In planar format, the luminance and chrominance data are stored in three different regions in the memory. In packed format, the luminance and chrominance data is stored packed in one memory region.

7.4.6.3.2 4:2:2 YCbCr Format

In the 4:2:2 YCbCr video format, in each frame of video, the number of samples per line of each chrominance component, Cr or Cb is one-half of the number of samples per line of luminance. The chrominance resolution is the same as that of luminance resolution vertically. The exact location of chrominance samples with respect to luminance samples in a frame of 4:2:2 video is shown in Figure 7-7.

Figure 7-7. Luminance and Chrominance Samples in 4:2:2 Video Frame

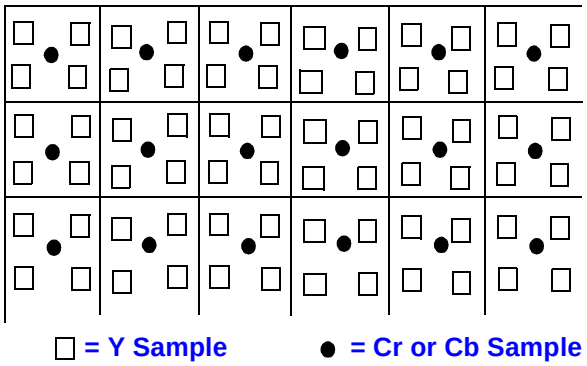


The luminance and chrominance values are 8 bits each. The LCD controller supports the storage of 4:2:2 YCbCr video format in planar format in the frame buffer.

7.4.6.3.3 4:2:0 YCbCr Format

In 4:2:0 YCbCr video format, in each frame of video, the number of samples of each chrominance component (Cr or Cb) is one-half of the number of samples of luminance, both horizontally and vertically. The exact location of chrominance samples with respect to luminance samples in a frame of 4:2:0 video is shown in Table 7-8.The luminance and chrominance values are 8 bits each. The LCD controller supports the storage of 4:2:0 video format in planar format in the frame buffer.

Figure 7-8. Luminance and Chrominance Samples in 4:2:0 YCbCr Video Frame



7.4.7 Base Frame

The number of pixel elements in the base frame is always equal to the resolution of the screen and can be stored in either internal SRAM or external memory in any of the formats described in Section 7.4.6. The screen coordinates are expressed in pixels, and a left-handed coordinate system is used when referencing a spatial location within the screen. The upper left corner of the screen is (0,0), with the X coordinate increasing from left to right and the Y coordinate increasing from top to bottom.

The pixel format is established by writing to the LCD Controller register 3, as described in Section 7-42. For pixel depths of 2 bpp, the overlays and hardware cursor are disabled. For pixel depths of 2, 4, and 8 bpp, the palette data is loaded into the palette RAM by DMA channel 0. If the pixel depth is greater than 8 bpp, the pixel data from the input FIFO bypasses the palette RAM.

The palette RAM provides for mapping the 2-, 4-, or 8-bpp formats to a 16- or 25-bit value and must be loaded at least once before the pixel data can be displayed on the LCD panel. If transparency is used, the most significant bit determines the transparency, and the lower 24 bits (for color) represent the red, green, and blue color components or the lower 8 bits for monochrome. The frame data is loaded into its dedicated input FIFO by DMA channel 0.

7.4.8 Overlay 1 Window

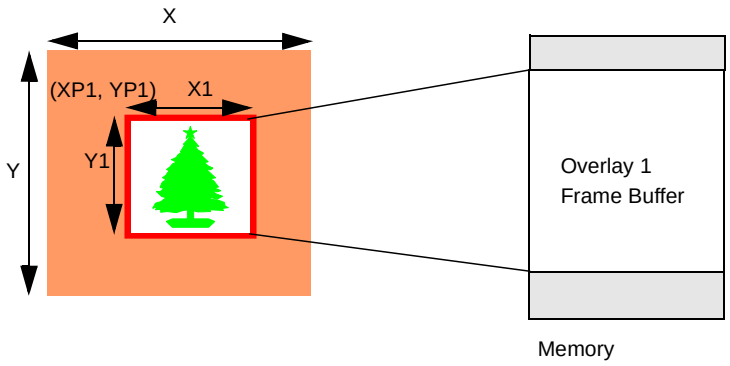
The number of pixel elements and the position of Overlay 1 are programmable. The pixel data for the Overlay 1 plane is stored in either internal SRAM or external memory. The storage of Overlay 1 data in the frame buffers is shown in Figure 7-9. Each pixel stored in memory can be any one of 4-, 8-, 16-, 18-, 19-, 24-, or 25-bpp formats. The size (X1, Y1) and the pixel format is programmed by writing to the Overlay 1 Control register 1. The position (XP1, YP1) of the window is programmed by writing to the Overlay 1 Control register 2. The position (XP1, YP1) represents the upper left corner of the Overlay 1 frame. Results will be unexpected if the position (XP1, YP1) falls outside the base frame window. Use caution such that the position (XP1, YP1) is within the base frame window and the size fits within the base frame window.

It is possible to change the size, pixel format, or position of Overlay 1 while Overlay 1 is enabled by writing the new size or pixel format to the Overlay 1 Control register 1 or writing the new position to the Overlay 1 Control register 2 and then writing the starting location of the descriptor to the DMA Frame Branch registers (FBRx). If the Overlay 1 pixel format is set to 4 bpp or 8 bpp, the palette RAM must be reloaded when the size, pixel format, or position of Overlay 1 is modified. The Overlay 1 display is disabled upon the completion of the write to the Overlay 1 Control register 1 or the Overlay 1 Control register 2 and is re-enabled when the branch specified in the Overlay 1 Frame Branch register has been completed and the LCD has reached the start of the Overlay 1 position on the next frame.

If any of Overlay 1 Frame Branch registers are written when the DMA is in descriptor-fetch mode for the Overlay 1 channel, the branch does not occur until the next frame.Consequently, if the same data is to be used, then the same DMA descriptor address must be written to the Overlay 1 Frame Branch register. Wait for the Overlay 1 branch-status interrupt to occur before updating the size, pixel format, or position of the Overlay 1 again. The same procedure is used for disabling the Overlay 1 window.

If the LCD controller is disabled while processing a frame (branch status interrupt did NOT occur), incorrect operation of the LCD may result. If the LCD is disabled while processing a frame, follow the steps described in Section 7.4.4 to reset the LCD controller and re-initiate correct operation of the LCD.

Figure 7-9. Overlay 1 Frame Buffer Format



X * Y—Size of LCD Screen

X1 * Y1—Size of Overlay 1

Overlay 1 has its own 16x64-bit input FIFO. The frame data is fetched from either internal SRAM or external memory into the input FIFO by DMA channel 1. Overlay 1 has a dedicated 256x25-bit palette RAM. When the pixel depth is 4 or 8 bpp, the palette data is fetched optionally every frame from either internal SRAM or external memory into the palette RAM by DMA channel 1. When the pixel depth is 4 bpp, the 4-bit pixel indexes into the palette RAM to select one of the 16 locations. When the pixel depth is 8 bpp, 8-bit pixel indexes into the palette RAM to select one of the 256 locations. The output of the palette RAM is 25 bits wide. If transparency is used, the most significant bit of the palette RAM output is the transparency bit, which defines the transparency for the pixel and the lower 24 bits represent the pixel data for color (lower 8 bits for monochrome)—refer to Figure 7-18. When the pixel depth is greater than 8 bpp, the pixel data bypasses the palette RAM. The data from the input FIFO is combined with the data from Overlay 2, as shown in Figure 7-1.

Note: The LCD controller does not support overlays and hardware cursor for dual-scan displays.

7.4.9 Overlay 2 Window

The size and the position of Overlay 2 are programmable. The size (X2, Y2) and the pixel format are programmed by writing to the Overlay 2 Control register 1 (see Section 7.5.10) The position (XP2,YP2) of the Overlay 2 window is programmed by writing the Overlay 2 Control register 2 (see Section 7.5.11).

Overlay 2 supports the following formats and pixel depths:

- RGB format, pixel depths of 4, 8, 16, 18, 19, 24, and 25 bpp
- 4:4:4 YCbCr sampling format
- 4:2:2 YCbCr sampling format
- 4:2:0 YCbCr sampling format.

Note: The LCD controller does not support overlays and hardware cursor for dual-scan displays.

For RGB or 4:4:4 YcbCr packed-pixel format, the size, pixel format or position of Overlay 2 can be changed while Overlay 2 is enabled by (1) writing the new size or pixel format to the OVL2C1 register or (2) writing the new position to the OVL2C2 register and then writing the starting location of the descriptor to the FBR2 register. If the Overlay 2 pixel format is set to 4 bpp or 8 bpp, the palette RAM must be reloaded when the size, pixel format, or position of Overlay 2 is modified. The Overlay 2 with new size, pixel format, or position is displayed when the branch specified in the Overlay 2 Frame Branch register has been completed and the LCD has reached the start of the Overlay 2 position on the next frame. If the Channel 2 Frame Branch register is written when the DMA is in descriptor-fetch mode for the Overlay 2 channel (channel 2), the branch does not occur until the next frame of that channel. Consequently, if the same data is to be used, then the same DMA descriptor address must be written to the Overlay 2 Frame Branch register. Wait for the Overlay 2 branch-status interrupt to occur before updating the size, pixel format, or position of the Overlay 2 again. Use the same procedure to disable the Overlay 2 window.

For 4:4:4 unpacked or 4:2:2 or 4:2:0 YCbCr pixel format, the size, pixel format or position of Overlay 2 can be changed while Overlay 2 is enabled by writing the new size or pixel format to the OVL2C1 register or writing the new position to the OVL2C2 register and then writing the starting location of the descriptors to FBR2, FBR2, and FBR4 registers. Overlay 2 with the new size, pixel format or position is displayed when all the three branches specified in the Overlay 2 Frame Branch registers (channel 2, 3 and 4) have been completed and the LCD has reached the start of the Overlay 2 position on the next frame. If any of the Overlay 2 Frame Branch registers are written when the DMA is in descriptor-fetch mode for any of the Overlay 2 channels (channel 2, 3 or 4), the branch does not occur until the next frame. Consequently, if the same data is to be used, then the same DMA descriptor address must be written to the Overlay 2 Frame Branch registers. Wait for the all three (channel 2, 3 and 4) Overlay 2 branch-status interrupts to occur before updating the size, pixel format, or position of Overlay 2 again. The same procedure is used for disabling the Overlay 2 window.

If the LCD controller is disabled while processing a frame (branch status interrupt did NOT occur), incorrect operation of the LCD may result. If the LCD controller is disabled while processing a frame, follow the steps described in Section 7.4.4 to reset the LCD controller and re-initiate correct operation of the LCD.

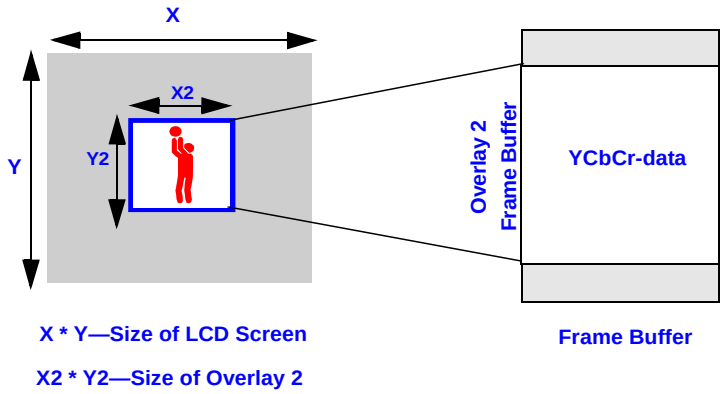
When the pixel data is in RGB format, the frame and the palette data (for 4 or 8 bpp) are fetched from either internal SRAM or external memory by DMA channel 2. The Overlay 2 has its own 256x25-bit palette RAM. The Overlay 2 has three 16x64-bit input FIFOs. For RGB format, the frame data is loaded into one of the input FIFOs. When pixel depth is 4 bpp, the pixel data from the input FIFO indexes into the palette RAM to select one of the top 16 locations. When the pixel depth is 8 bpp, the pixel data from the input FIFO indexes into the palette RAM to select one of 256 locations. The output of the palette RAM is 16- or 25-bits wide. Since Overlay 2 always is considered to reside underneath Overlay 1 (see Figure 7-5), the most significant bit (25th bit) of output of the RAM is ignored, and the lower 24 bits represent the color value, which means that any transparency information (T-bit) for Overlay 2 is ignored. When the pixel depth is greater than 8 bpp, the pixel data bypasses the palette RAM.

When the pixel data is in 4:4:4 YCbCr planar format, the luminance and chrominance data is stored in three different regions in the frame buffer as shown in Figure 7-11. The frame data from the frame buffer is fetched into the LCD input FIFOs by the DMA channels (channels 2, 3, and 4). When the pixel data is in 4:4:4 YCbCr-packed format, the luminance and chrominance data is stored packed in one memory region as shown in Figure 7-10. Each pixel has 8 bits of Y, 8 bits of Cb and 8 bits of Cr data. Table 7-18 shows the format of the pixel data stored in memory. The data from the frame buffer is fetched into the LCD input FIFO by the DMA channel 2. The output of the input FIFO is color converted from 4:4:4 YCbCr format into 16-, 18-, or 24-bit RGB format. The converted data bypasses the palette RAM and is combined with Overlay 1 data.

Table 7-18. YCbCr 4:4:4 Packed Pixel Data Format Stored in Memory

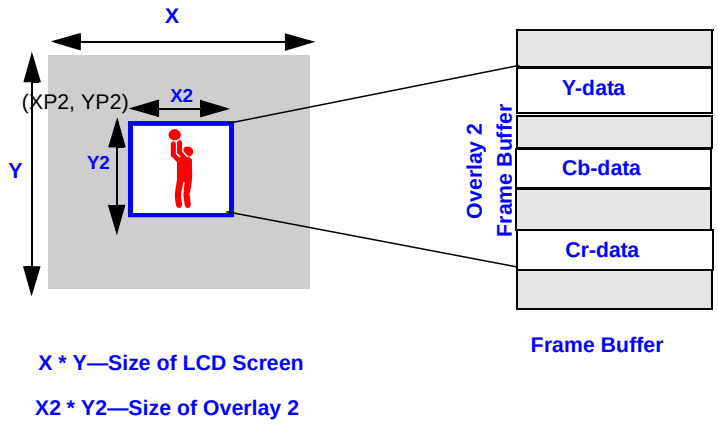
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Cr [7:0]								Cb [7:0]								Y [7:0]							

Figure 7-10. Overlay 2 Frame Buffer Format for 4:4:4 YCbCr Packed Format



When the pixel data is in 4:2:0 or 4:2:2 YCbCr planar format, the data is stored in three different memory regions in the frame buffer, as shown in Figure 7-11. The data from the frame buffer is fetched into the input FIFOs (dedicated for Overlay 2) by the DMA channels (channels 2, 3, and 4). The Y data is fetched by DMA channel 2, Cb data is fetched by DMA channel 3, and Cr data is fetched by DMA channel 4. Video data in 4:2:0 or 4:2:2 is converted into 24-bit YCbCr 4:4:4 using bilinear interpolation (see Section 7.4.9.1). Following the interpolation, the RGB color components are then converted to 16-, 18-, or 24-bit RGB values. The resultant RGB data bypasses the palette RAM and is combined with the Overlay 1 data, as shown in Figure 7-1.

Figure 7-11. Overlay 2 Frame Buffer Format for YCbCr Planar Format

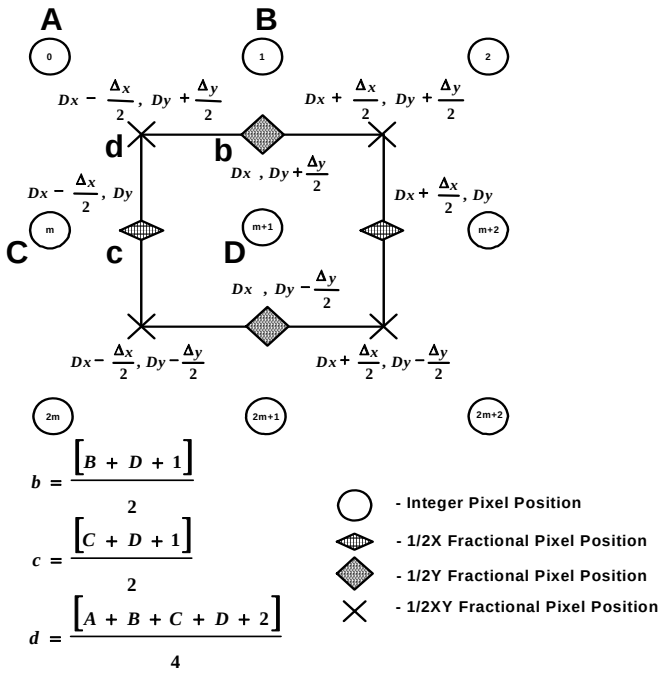


7.4.9.1 Bilinear Interpolation

The pixel data in YCbCr 4:2:0 and YCbCr 4:2:2 sampling formats are converted to YCbCr 4:4:4 sampling format through bilinear interpolation. In this conversion, the value of each pixel in the up-sampled color plane (YCbCr 4:4:4) is interpolated from the values of two or four corresponding

neighboring pixels in the input color plane. The method provides for interleaving the existing integer pixel elements with the 1/2X, 1/2Y, and 1/2XY interpolated values. The bilinear interpolation method is illustrated in Figure 7-12

Figure 7-12. Bilinear Interpolation for 1/2X, 1/2Y, and 1/2XY Locations



To deal with the boundary problem of this up-sampling method, use the following relationship for a MxN array.

$$I(m, N+1) = I(m, N) \quad \text{for} \quad 0 \le m \le M$$

$$I(M+1, n) = I(M, n) \quad \text{for} \quad 0 \le n \le N$$

The last row and column are effectively duplicated using this approach. For example, the 2:1 upsampling of 3x3 input color plane results in a 3x6 array as illustrated in Figure 7-13.

Figure 7-13. 2:1 Upsampling in the Horizontal Dimension

$$I_{in} = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix}$$

$$I_{out2} = \begin{bmatrix} C_{00} & \frac{C_{00}+C_{01}}{2} & C_{01} & \frac{C_{01}+C_{02}}{2} & C_{02} & \alpha \\ C_{10} & \frac{C_{10}+C_{11}}{2} & C_{11} & \frac{C_{11}+C_{12}}{2} & C_{12} & \alpha \\ C_{20} & \frac{C_{20}+C_{21}}{2} & C_{21} & \frac{C_{21}+C_{22}}{2} & C_{22} & \alpha \end{bmatrix}$$

The pixel data in YCbCr 4:2:0 format is similarly converted through 2:1 up-sampling in both the horizontal and vertical dimensions. Figure 7-14 is an example of upscaling a 3x3 block.

Figure 7-14. 2:1 Upsampling in the Horizontal and Vertical Dimensions

$$I_{in} = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix}$$

$$I_{out1} = \begin{bmatrix} C_{00} & \frac{C_{00}+C_{01}}{2} & C_{01} & \frac{C_{01}+C_{02}}{2} & C_{02} & \alpha \\ \frac{C_{00}+C_{10}}{2} & \frac{C_{00}+C_{01}+C_{10}+C_{11}}{4} & \frac{C_{01}+C_{11}}{4} & \frac{C_{01}+C_{02}+C_{11}+C_{12}}{4} & \frac{C_{02}+C_{12}}{2} & \alpha \\ C_{10} & \frac{C_{10}+C_{11}}{2} & C_{11} & \frac{C_{11}+C_{12}}{4} & C_{12} & \alpha \\ \frac{C_{10}+C_{20}}{2} & \frac{C_{10}+C_{11}+C_{20}+C_{21}}{4} & \frac{C_{11}+C_{21}}{4} & \frac{C_{11}+C_{12}+C_{21}+C_{22}}{4} & \frac{C_{12}+C_{22}}{2} & \alpha \\ C_{20} & \frac{C_{20}+C_{21}}{2} & C_{21} & \frac{C_{21}+C_{22}}{4} & C_{22} & \alpha \\ \alpha & \alpha & \alpha & \alpha & \alpha & \alpha \end{bmatrix}$$

7.4.9.2 Color-Space Conversion from YCbCr to RGB Format

The color-space conversion from YCrCb to RGB is done based on the CCIR 601-2 standard. Using the inverse of the encoding equations, the YCrCb 4:4:4 data is converted to gamma-corrected RGB using the following equations.

$$R = 1.164 \times (Y - 16) + 1.596 \times (C_r - 128)$$

$$G = 1.164 \times (Y - 16) - 0.391 \times (C_b - 128) - 0.813 \times (C_r - 128)$$

$$B = 1.164 \cdot (Y - 16) + 2.017 \times (C_b - 128)$$

The matrix representation is as follows:

Equation 7-1. Matrix Representation

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{256} \begin{bmatrix} K_1 & 0 & K_2 \\ K_1 & K_3 & K_4 \\ K_1 & K_5 & 0 \end{bmatrix} \bullet \left(\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \right)$$

where $K_1 = 298.082$, $K_2 = 408.583$, $K_3 = -100.291$, $K_4 = -208.120$, $K_5 = 516.411$

The RGB data must be saturated to 255 on overflow and to 0 on underflow to prevent errors from occurring due to Y and CbCr occasionally going outside the 16–235 and 16–240 ranges, respectively.

7.4.9.3 Color-Space Conversion from 24 bpp to 16, 18, and 19 bpp RGB Formats

The color-space conversion of the Overlay 2 image plane from YCbCr to RGB results in a 24-bit true-color element for each pixel. To combine the image planes arithmetically, convert to a common pixel format being used by the Base, Overlay 1, and Cursor image planes.The conversion algorithm is straightforward and involves a scaling operation. For example, for the format conversion from RGB 8:8:8 to RGB 5:5:5, the five most significant bits of each of the red, green, and blue color are combined into a 16 bpp format as shown in Table 7-19 and Table 7-20. Note that these do not show the transparency bit that must be included in the pixel data in the frame buffer.

This conversion is performed internally by the LCD controller logic and does not require any software interaction. The converted format is determined by LCCR3[BPP] and LCCR3[PDFOR].

Example: Packing and Precision from RGB 8:8:8 to RGB 5:5:5

Table 7-19. Initial Format—RGB 8:8:8

Bit	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGB 8:8:8	Red [7:0]								Green [7:0]								Blue [7:0]							

3 LSBs for Each Color Channel are Truncated to Form Lower Precision RGB 5:5:5 Format.

Table 7-20. Converted Format—RGB 5:5:5

Bit	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGB 5:5:5										Red [4:0]					Green[4:0]					Blue[4:0]				

The supported format conversions also include the RGBT 6:6:6 format for 19 bpp and RGBT 8:8:7 format for 24 bpp. The supported conversions for the Overlay 2 plane following color space conversion are provided in Table 7-21.

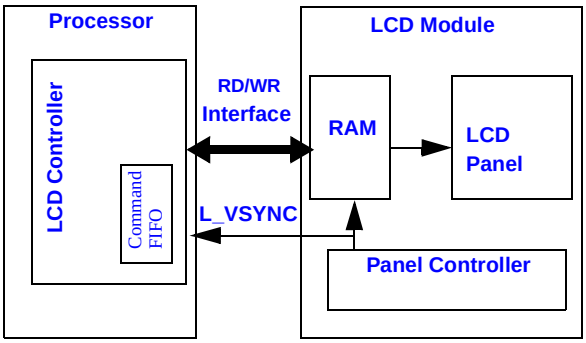
Table 7-21. Supported Color Component Precision Conversions Following Color Space Conversion

Pixel Depth	Supported Format Conversion from RGB 8:8:8
16bpp	RGBT 5:5:5
19bpp	RGBT 6:6:6
24bpp	RGBT 8:8:7

7.4.10 Interfacing with LCD Smart Panels

The LCD controller can interface to LCD smart panels. *Smart panels* are LCD panels that have their own internal memory acting as a local frame buffer. The LCD controller can write to the panel frame buffer using the read/write interface to the smart panel. Figure 7-15 shows the interface between LCD controller and the LCD panel.

Figure 7-15. Interface to LCD Smart Panel with Internal Frame Buffer



The LCD module internal RAM stores the panel frame data. The LCD controller can read and write into the LCD module internal RAM (frame buffer) stores. The LCD controller sends command sequences to configure the LCD module.

The LCD controller has a 4x52-bit FIFO that holds sixteen 13-bit commands. The commands are loaded into the command FIFO by DMA channel 6 from memory. Each command is 13 bits wide as described in Table 7-22.

Table 7-22. Command Data Format

12:9	8	7:0
Command	Command/Data bit (A0)	Data

The various commands are described in Table 7-23.

Table 7-23. Command Description

Command	A0	Command Function	Description
0000	0	Read Status Register	Read the status register within the LCD module and interrupt the processor. Wait until the processor reads the LCD Controller Read register and updates the control bits.
0000	1	Read from Frame Buffer	Same as Read Status register.
0001	0	Command Write	Send a command write to LCD module.
0001	1	Data Write	Send a data write to the LCD module.
0010	x	Frame Data Write	Send the entire frame data before going to next command in the FIFO.
0011	x	Wait for Vsync	Wait for L_VSYNC signal to be asserted by the LCD panel the number of times programmed in CMDCR[SYNC_CNT], then execute the next command.
0100	x	No Operation	Do nothing. Go to the next command
0101	x	Interrupt the Processor	Interrupt the processor by setting the status bit “CMD_INTR” in LCD Status register 0.
Others	x	Reserved	—

7.4.10.1 Read Command

The LCD controller sends Read Status Register commands to the LCD panel with A0 low. Read data from the LCD module is loaded into the LCD Controller Panel Read Status register (PRSR) (see Table 7-57). The LCD controller then clears PRSR[A0] and interrupts the processor by setting LCSR0[RD_ST]. The LCD controller waits until the processor has read PRSR and has written PRSR[CON_NT] and PRSR[ST_OK]. CON_NT and ST_OK indicate to the LCD controller its next action, as listed in Table 7-24.

The LCD controller sends read-from-frame-buffer commands to the LCD panel with A0 high. The read data from the LCD module is loaded into PRSR. The LCD controller then sets PRSR[A0] and interrupts the processor by setting LCSR0[RD_ST]. The LCD controller waits until the processor has read PRSR and has written PRSR[CON_NT] and PRSR[ST_OK].CON_NT and ST_OK indicate to the LCD controller its next action, as listed in Table 7-24.

The Table 7-24 describes the functionality of the PRSR control bits.

Table 7-24. Control Bit Description

[CON_NT, ST_OK]	Description
0x	Wait for the process to intervene
10	Repeat the same read command
11	Continue and do the next command

Table 7-25 shows the format in which the command data is stored in the memory. The command data is fetched into the command RAM by DMA channel 6.

Table 7-25. Command Data Format Stored in Memory

Bits	31		28				16	15		12				0
0x0	Unused		Command Data 1					Unused		Command Data 0				
0x4	Unused		Command Data 3					Unused		Command Data 2				
. . .														
0x1C	Unused		Command Data 15					Unused		Command Data 14				

7.4.11 Hardware Cursor

The LCD controller provides a hardware cursor that can be disabled or configured to one of the possible modes.

- 32x32x2bpp 2-color and transparency mode
- 64x64x2bpp 2-color and transparency mode
- 32x32x2bpp 4-color mode
- 64x64x2bpp 4-color mode
- 32x32x2bpp 3-color and transparency mode
- 64x64x2bpp 3-color and transparency mode

- 128x128x1bpp 2-color mode
- 128x128x1bpp 1-color mode and transparency mode

Note: The LCD controller does not support the hardware cursor for dual-scan displays.

The cursor data is stored in its frame memory (a separate memory space than the main frame buffer), which allows the cursor to be displayed and used without altering the main display image stored. It can have multiple patterns stored in different memory locations, making it possible to change each cursor's appearance simply by switching from one stored image to another. The cursor has a dedicated 16x64-bit input FIFO. Pixel data for the cursor is fetched from the memory by DMA channel 5 into the input FIFO. Users can enable, disable and configure the cursor by programming the Cursor Control register.

7.4.11.1 32x32x2bpp and 64x64x2bpp 2-Color and Transparency Modes

These two modes are follow the Microsoft Windows cursor data-plane structure. Each pixel has 2 bits, which represent four colors: two colors to draw a cursor, a third color for transparency (which allows the main display image behind the cursor to show through) and a fourth color for inverted transparency (which allows the main display image behind to show through, but with its color value inverted) as shown in Table 7-26.

Table 7-26. Pixel Data 32x32x2bpp and 64x64x2bpp 2-Color and Transparency Modes

Bits/Pixel	Color Displayed at Corresponding Pixel Position
00	Cursor Color 0
01	Cursor Color 1
10	Transparent. The pixel of the main display image behind cursor shows through.
11	Transparent, but inverted. The pixel of the main display image behind cursor shows through with inverted color.

7.4.11.2 32x32x2bpp and 64x64x2bpp 4-Color Modes

This mode provides four colors for drawing the cursor. Each pixel contains two bits, which specify four colors as shown in Table 7-27. The four colors reside in the color map, each pixel indexes into the color map to get the color value.

Table 7-27. Pixel Data 32x32x2bpp and 64x64x2bpp 4-Color Modes

Bits/Pixel	Color Displayed at Corresponding Pixel Position
00	Cursor Color 0
01	Cursor Color 1
10	Cursor Color 2
11	Cursor Color 3

7.4.11.3 32x32x2bpp and 64x64x2bpp 3-Color and Transparency Modes

This mode provides three colors for drawing and a fourth color for transparency, which allows the main display image behind the cursor to show through. Each pixel contains two bits, which specifies one of the four colors as shown in Table 7-28.

Table 7-28. Pixel Data 32x32x2bpp and 64x64x2bpp 3-Color and Transparency Modes

Bits/Pixel	Color Displayed at Corresponding Pixel Position
00	Cursor Color 0
01	Cursor Color 1
10	Cursor Color 2
11	Transparent

7.4.11.4 128x128x1bpp 2-Color Mode

This mode provides two colors for drawing the cursor. There is no provision for transparency in this mode. Each pixel contains 1 bit that specifies one of the two colors coded in the color RAM, as shown in Table 7-29.

Table 7-29. Pixel Data 128x128x1bpp 2-Color Mode

Bits/Pixel	Color Displayed at Corresponding Pixel Position
0	Cursor Color 0
1	Cursor Color 1

7.4.11.5 128x128x1bpp 1-Color and Transparency Mode

This mode provides one color for drawing the cursor and a second color for transparency, which allows the image behind the cursor to show through as shown in Table 7-30.

Table 7-30. Pixel Data 128x128x1bpp 1-Color and Transparency Mode

Bits/Pixel	Color Displayed at Corresponding Pixel Position
1	Transparent
0	Cursor Color 2

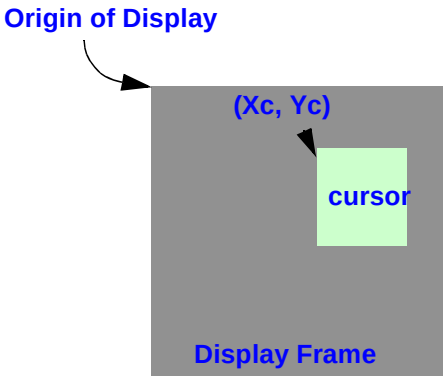
7.4.11.6 Cursor Positioning

The cursor position is defined by the (Xc, Yc) coordinates of the upper left corner pixel. It is specified by writing CCR[CXPOS] and CCR[CYPOS] (see Section 7.5.12). The cursor position (Xc, Yc) value of (0,0) positions the cursor at origin (upper left corner-of the display) of the display frame. The cursor is displayed partly or not displayed at all, depending on the cursor position (Xc, Yc).

It is possible to change the mode or position of the cursor while the cursor is enabled by writing the new position or mode to the Cursor Control register and then writing the start-descriptor address to the DMA Frame Branch register for channel 5 (FBR5). The palette for the cursor must be reloaded whenever the cursor mode or position is changed. The cursor with the new size, pixel format and position is displayed when the branch specified in the DMA Frame Branch registers (FBRx) has been completed and the LCD has reached the start of the cursor position on the next frame. Consequently, even though the palette RAM and palette buffer data remain the same, the same DMA descriptor address must be written to the Cursor Frame Branch register. Wait for the cursor branch-status interrupt to occur before updating the size, pixel format, or position of the cursor again. The same procedure is used for disabling the cursor.

If the LCD controller is disabled while processing a frame (branch status interrupt did NOT occur) incorrect operation of the LCD may result. If the LCD is disabled while processing a frame follow the steps described in Section 7.4.4 to reset the LCD controller and re-initiate correct operation of the LCD.

Figure 7-16. Cursor Position within Display Frame



7.4.11.7 Cursor Color Map

The color map defines the colors used by the cursor. It has four 24-bit-wide locations and is loaded from either internal or external memory through DMA channel 5.

7.4.12 External Palette Buffer

The palette data for each overlay is stored in either internal memory or the memory—this area is referred to as the *palette buffer*. Palette data for the Base, Overlay 1 and Overlay 2 can be up to 256x25 bits. Palette data is 4 entries for pixel depths of 2 bpp, 16 entries for pixel depth of 4 bpp, and 256 entries for pixel depth of 8 bpp. The palette RAM is not used for pixel depths greater then 8 bpp. Users must load the palette RAM at least once if it is to be used; afterward, reloading the palette is optional for every frame.

7.4.12.1 Palette Data Formats

Figure 7-17 through Figure 7-20 show the pixel format for palette data stored in the palette buffer for various pixel depths. These are used only when the pixel data in the frame buffer is less then 16 bits. If transparency is not used and the BPP is less then 18 bits, the palette data must be in the appropriate format specified in Figure 7-17—depending on whether a monochrome or color panel is used. If transparency is used, the palette data must be in the appropriate format specified in Figure 7-18 through Figure 7-20—depending on whether a monochrome or color panel is used. The palette data is programmed to one of the formats by programming LCCR4[PAL_FOR]. In 16-bit and 18-bit palette format with transparency bit, the red, green, and blue components must be expanded to 8 bits (each color) by padding zeros to the right as shown in the corresponding figure.

Figure 7-17. Palette Data Formats—Transparency Disabled

16-Bit Palette Format without Transparency Bit for Color Panel

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Color	Red (R)					Green (G)						Blue (B)				

16-Bit Palette Format without Transparency Bit for Monochrome Panel

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mono	Unused								Monochrome (M)							

Figure 7-18. Palette Data Formats 0b01—Transparency Enabled

16-Bit Palette Format with Transparency Bit for Color Panel

Bit	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
Color	T	Red[4:0]					0	0	0	Green[5:0]						0	0	Blue[4:0]					0	0	0

16-Bit Palette Format with Transparency Bit for Monochrome Panel

Bit	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
Mono	T	Unused																Monochrome							

NOTE: T bit is the transparency bit. It must be zero-value when transparency is not intended.

Figure 7-19. Palette Data Formats 0b10—Transparency Enabled

18-Bit Palette Format with Transparency Bit for Color Panel

Bit	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
Color	T	Red[5:0]						0	0	Green[5:0]						0	0	Blue[5:0]						0	0

18-Bit Palette Format with Transparency Bit for Monochrome Panel

Bit	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
Mono	T	Unused																Monochrome							

NOTE: T bit is the transparency bit. It must be zero-value when transparency is not intended.

Figure 7-20. Palette Data Formats 0b11—Transparency Enabled

25 Bit Palette Format with Transparency Bit for Color Panel

Bit	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
Color	T	Expanded Red[7:0]								Expanded Green[7:0]								Expanded Blue[7:0]							

25 Bit Palette Format with Transparency Bit for Monochrome Panel

Bit	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
Mono	T	Unused																Monochrome							

NOTE: T bit is the transparency bit. It must be zero-value when transparency is not intended.

7.4.12.2 Little-Endian Format

The palette entries must be in little-endian format. Endian does not imply endianness with respect to bytes and half-words within memory. It refers strictly to the ordering of the palette entries; palette entry 0 is located at the LSB of a word boundary. The ordering of RGB values within the entry is fixed. Figure 7-21 shows the format in which the palette data is stored in the palette buffer.

Figure 7-21. Format for Palette Data

Palette Entry Ordering
16- or 256-Entry Palette Buffer

Bit	31 — 16	15 — 0
0x0	Palette entry 1	Palette entry 0
0x4	Palette entry 3	Palette entry 2

Note: Entries 4 through 255 do not exist for 1- and 2-bit/pixel modes

0x1C	Palette entry 15	Palette entry 14
0x20	Palette entry 17	Palette entry 16

Note: Entries 16 through 255 do not exist for 1-, 2- and 4-bit/pixel modes

0x1FC	Palette entry 255	Palette entry 254

Palette Entry Ordering with Overlays Enabled
16- or 256-Entry Palette Buffer

Bit	31 — 24	17 — 16	15 — 0
0x0	Unused		Palette entry 0
0x4	Unused		Palette entry 1
0x8	Unused		Palette entry 2
0xC	Unused		Palette entry 3

Note: Entries 4 through 255 do not exist for 1- and 2-bit/pixel modes

0x38	Unused		Palette entry 14
0x3C	Unused		Palette entry 15

Note: Entries 16 through 255 do not exist for 1-, 2- and 4-bit/pixel modes

0x3FC	Unused		Palette entry 255

7.4.13 Frame Buffer

The frame data for the base, Overlay 1 and Overlay 2, is stored in the frame buffers that reside in either internal or external memory. Each plane used (Base plane, Overlay 1, Overlay 2, cursor and command RAM) requires its own frame buffer to store the pixel data for that plane. Additionally, if a palette is used, a palette buffer (a small memory area that maps the palette colors to the LCD colors) is required. All the planes could use the same palette buffer, or each plane could have a separate palette buffer, depending on the system design preferences.

The amount of the pixel data for each layer depends on the size of the screen (for example, 800 x 600 = 480, 000 encoded pixel values as described in Section 7.4.2). The diagrams in this section show the memory organization within the frame buffer for each pixel size. The pixel entries are ordered starting with the least significant bit and ending with the most significant bit in a 32-bit word.

Each line in the memory must start at a word boundary. For the various pixel sizes, this requires each line of the display to have pixels in multiples of the following:

- 16 pixels for 2-bit pixels
- 8 pixels for 4-bit pixels
- 8 pixels for monochrome
- 8 pixels for pixel depths > 4 bpp passive-color mode
- 4 pixel for 8-bit pixels
- 2 pixels for pixel depth of 16 bpp active
- 8 pixels for packed 18- or 19-bit pixels
- 16 pixels for Overlay 2 frame when the data is in 4:2:0 YCbCr format

If the number of pixels per line of the LCD screen does not meet the requirements listed above, the start address must be adjusted for each line by adding “dummy” pixel values to the end of the previous line. For example, if the screen that is being driven is 107 pixels wide, and 4-bits/pixel mode is used, each line is 107 pixels or nibbles in length (53.5 bytes). The next nearest 8-pixel boundary (for 4-bit pixels) occurs at 112 pixels or nibbles (56 bytes). Thus, each new line must start in the frame buffer at multiples of 56 bytes by adding an extra 5 dummy pixels per line (2.5 bytes). Note that if dummy pixels are to be inserted, the panel being controlled must ignore the extra pixel clocks at the end of each line that correspond to the dummy pixels.

7.4.13.1 Memory Organization for Pixel Depth of 2 bpp

Figure 7-22 shows the format in which the pixel data is stored in memory for a pixel depth of 2 bits per pixel.

Figure 7-22. Memory Organization for Pixel Depth of 2 bpp

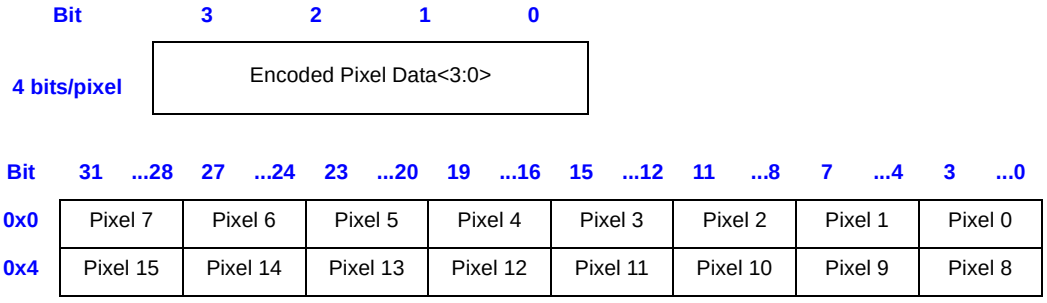
Bit	1	0
2 bits/pixel	Encoded Pixel Data<1:0>	

Bit	31	30	29	28	27	26	...	...	7	6	5	4	3	2	1	0
0x0	Pixel 15		Pixel 14		Pixel 13		...		Pixel 3		Pixel 2		Pixel 1		Pixel 0	
0x4	Pixel 31		Pixel 30		Pixel 29		...		Pixel 19		Pixel18		Pixel 17		Pixel 16	

7.4.13.2 Memory Organization for Pixel Depth of 4 bpp

Figure 7-23 shows the format in which the pixel data is stored in memory for a pixel depth of 4 bits per pixel.

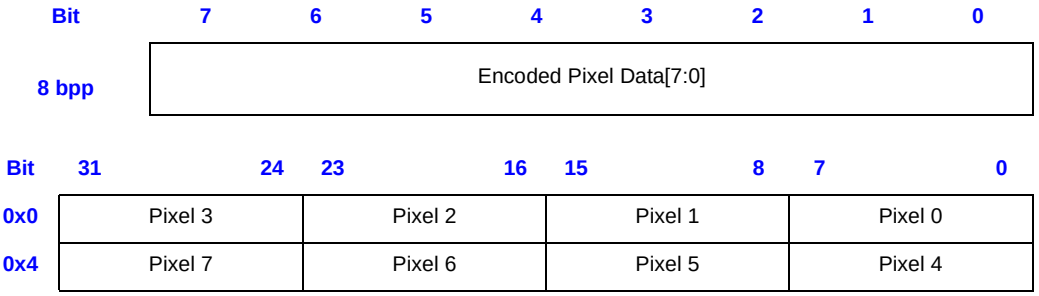
Figure 7-23. Memory Organization for Pixel Depth of 4 bpp



7.4.13.3 Memory Organization for Pixel Depth of 8 bpp

Figure 7-24 shows the format in which the pixel data is stored in memory for a pixel depth of 8 bits per pixel.

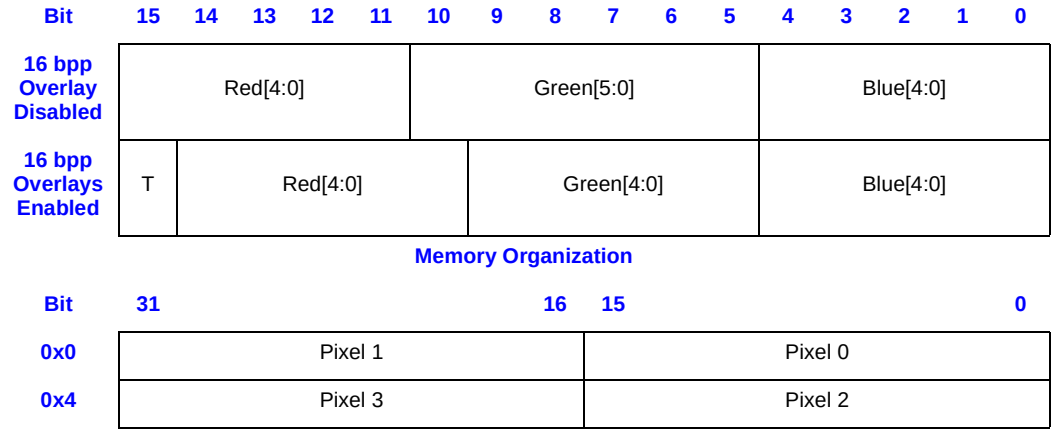
Figure 7-24. Memory Organization for Pixel Depth of 8 bpp



7.4.13.4 Memory Organization for Pixel Depth of 16 bpp

Figure 7-25 shows the format in which the pixel data is stored in memory for a pixel depth of 16 bits per pixel.

Figure 7-25. Memory Organization for Pixel Depth of 16 bpp



7.4.13.5 Memory Organization for Pixel Depth of 18 bpp

Figure 7-26 shows the format in which the pixel data is stored in memory for a unpacked pixel depth of 18 bits per pixel.

Figure 7-26. Memory Organization for Pixel Depth of 18 bpp Unpacked

Bit	17	16	12	11	6	5	0
18 bpp	Red[5:0]			Green[5:0]		Blue[5:0]	

Memory Organization

Bit	31	18	17	0
0x0	Unused		Pixel 0	
0x4	Unused		Pixel 1	

Figure 7-27 shows the format in which the pixel data is stored in memory for a packed pixel depth of 18 bits per pixel.

Figure 7-27. Memory Organization for Pixel Depth of 18 bpp Packed

Bit	17	12	11	6	5	0
18 bpp	Red Data[5:0]		Green Data[5:0]		Blue Data[5:0]	

Memory Organization

Bit	31	24	23	16	15	8	7	0
0x0	pixel1[7:0]		{[000000], pixel0[17:0]}					
0x4	pixel2[15:0]				{[000000], pixel1[17:8]}			
0x8	{[000000], pixel3[17:0]}						{[000000], pixel2[17:16]}	

NOTE: The data is not packed across frame line but only in each individual frame lines.

7.4.13.6 Memory Organization for Pixel Depth of 19 bpp

Figure 7-28 shows the format in which the pixel data is stored in memory for a unpacked pixel depth of 19 bits per pixel.

Figure 7-28. Memory Organization for Pixel Depth of 19 bpp Unpacked

Bit	18	17 … 12	11 … 6	5 … 0
19 bits/ pixel	T	Red [5:0]	Green [5:0]	Blue [5:0]

Memory Organization

Bit	31 … 24	23 … 0
0x0	Unused	{[00000], pixel0[18:0]}
0x4	Unused	{[00000], pixel1[18:0]}

Figure 7-29 shows the format in which the pixel data is stored in memory for a packed pixel depth of 19 bits per pixel.

Figure 7-29. Memory Organization for Pixel Depth of 19 bpp Packed

Bit	18	17 … 12	11 … 6	5 … 0
19 bits/ pixel	T	Red[5:0]	Green[5:0]	Blue[5:0]

Memory Organization

Bit	31 … 24	23 … 7	7 … 0
0x0	pixel1[7:0]	{[00000], Pixel 0[18:0]}	
0x4	pixel 2[15:0]	{[00000], pixel1[18:8]}	
0x8	{[00000], pixel 3[18:0]}		{[00000],pixel2[18:16]}

NOTE: The data is not packed across frame line but only in each individual frame lines.

7.4.13.7 Memory Organization for Pixel Depth of 24 bpp

Figure 7-30 shows the format in which the pixel data is stored in memory for a pixel depth of 24 bits per pixel.

Note: Configuring the LCD controller for 24 bpp output is invalid for panels without an internal frame buffer (LCCR0[LCDT] clear) and results in indeterminate behavior.

Figure 7-30. Memory Organization for Pixel Depth of 24 bpp

Bit	23	22 … 16	15	14 … 8	7	6 … 0
24 bpp (Overlays disabled)	Red[7:0]		Green[7:0]		Blue[7:0]	
24 bpp (Overlays enabled)	T	Red[7:0]		Green[7:0]		Blue[6:0]
24-Bit Mode (Overlays enabled)	Red[7:0]		Green[7:0]		Blue[7:0]	

Memory Organization

Bit	31 … 24	23 … 0
0x0	Unused	Pixel 0
0x4	Unused	Pixel 1

7.4.13.8 Memory Organization for Pixel Depth of 25 bpp

Figure 7-31 shows the format in which the pixel data is stored in memory for a pixel depth of 25 bits per pixel.

Note: Configuring the LCD controller for 25 bpp output is invalid for panels that are not smart panels (LCCR0[LCDT] clear) and results in indeterminate behavior.

Figure 7-31. Memory Organization for Pixel Depth of 25 bpp

Bit	24	23 … 16	15 … 8	7 … 0
25 bpp	T	Red Data[7:0]	Green Data[7:0]	Blue Data[7:0]

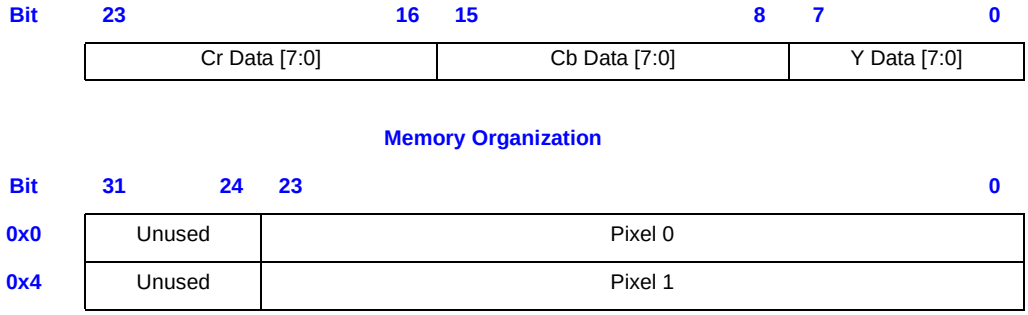
Memory Organization

Bit	31 … 25	24 … 0
0x0	Unused	Pixel 0
0x4	Unused	Pixel 1

7.4.13.9 Memory Organization for 4:4:4 YCbCr Packed Format

Figure 7-32 shows the format in which the pixel data is stored in memory for a 4:4:4 YCbCr packed format.

Figure 7-32. Memory Organization for 4:4:4 YCbCr Packed Format



7.4.14 Dual-Scan Mode

In dual-scan mode, pixels are presented to two halves of the screen at the same time (upper and lower) resulting in two sub-pixels being written every pixel clock. Single-scan panels only write one sub-pixel per pixel clock. Because this allows more pixels to be refreshed in a fixed time interval, lower pixel clock rates can be used, which is especially useful for larger panels and is typically the only market for dual-scan panels.

The hardware cursor and the overlays are disabled in this dual-scan mode. The LCD controller ignores the register values corresponding to the cursor and the overlays in this mode. DMA channel 0 fetches frame data for the upper panel into the Base input FIFO. DMA channel 1 fetches frame data for the lower panel into Overlay 1 input FIFO. For pixel depths less then 16 bpp, the palette data is loaded in the Base palette RAM by DMA channel 0. For monochrome, LDD<3:0> display data on the upper half of the screen and LDD<7:4> display data on the lower half of the screen. For passive color panels, LDD<7:0> display data on the upper half of the screen and LDD<15:8] display data on the lower half of the screen.

7.4.15 Functional Timing

Refer to Figure 7-33 through Figure 7-38 for LCD controller-pin timing diagrams.

Figure 7-33 shows the output pin timing. The LCD controller is enabled when LCCR0[ENB] is set. Bits LCCR3[VSP], LCCR3[HSP], and LCCR3[PCP] define the polarity of the frame clock, line clock, and pixel clock, respectively.

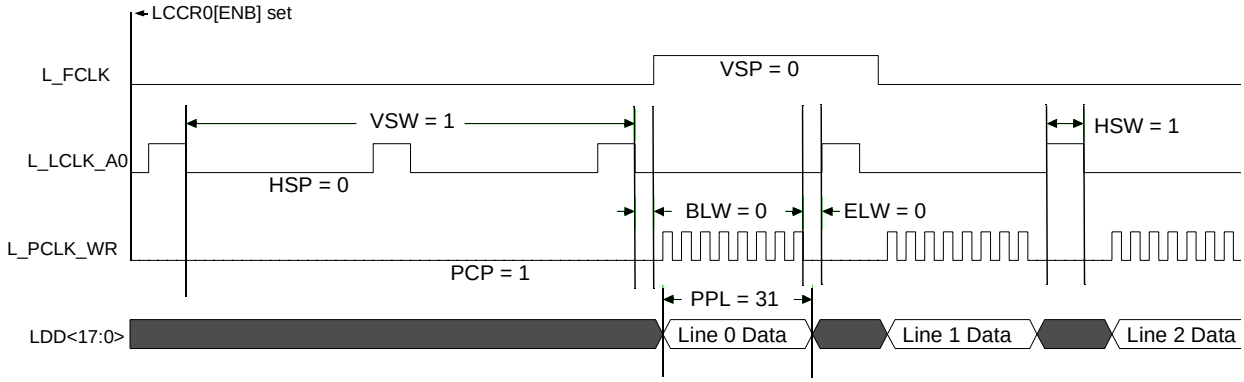
7.4.15.0.1 Passive Mode Timing

For passive (and active) LCD panels, when an entire line of pixels has been output to the LCD controller screen, the line clock pin (L_LCLK) is toggled. Likewise, when an entire frame of pixels has been output to the LCD controller screen, the frame-clock pin (L_FCLK) is toggled.

To prevent a DC charge from building within a passive display, the power and ground supplies must be switched periodically. The LCD controller signals the display to switch the polarity by toggling the AC bias pin (L_BIAS). Users can control the frequency of the bias pin by programming the number of line clock transitions between each toggle.

The programmable timing of the line and frame clock pins supports both passive and active mode. Programming options include: Wait-state insertion both at the beginning and end of each line and frame; pixel clock; line clock; frame clock; output-enable signal polarity; and frame-clock pulse width.

Figure 7-33. LCD Controller Pin Timing



LCCR0[ENB]—LCD Enable
0—LCD is disabled
1—LCD is enabled

VSP—Vertical Sync Polarity
0—Frame clock is active high, inactive low
1—Frame clock is active low, inactive high

HSP—Horizontal Sync Polarity
0—Line clock is active high, inactive low
1—Line clock is active low, inactive high

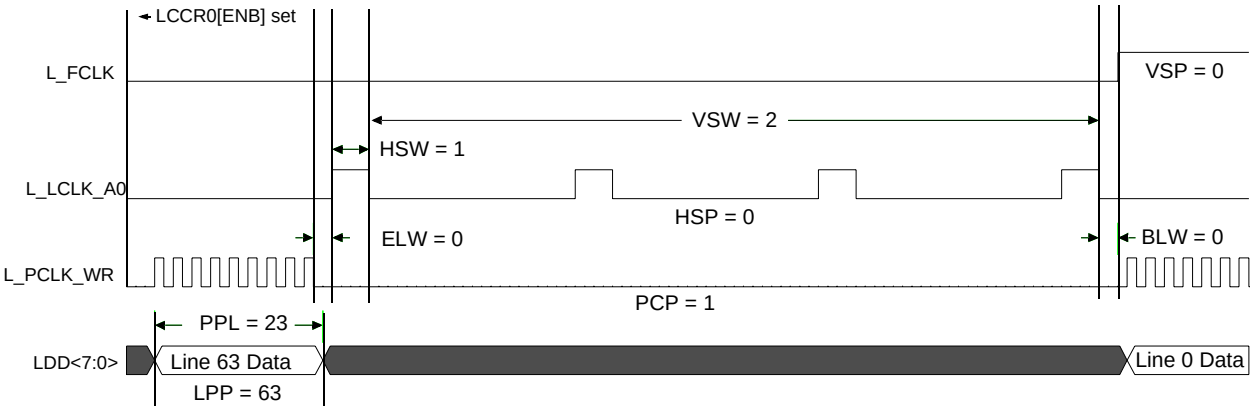
PCP—Pixel Clock Polarity
0—Pixels sampled from data pins on rising edge of clock
1—Pixels sampled from data pins on falling edge of clock

Note that for PCP = 0 data, L_FCLk and L_LCLK stable on the rising edge of the L_PCLK, and they toggle on falling edge of the L_PCLK.

VSW = Vertical Sync Pulse Width—1
HSW = Horizontal Sync (Line Clock) Pulse Width – 1
BLW = Beginning-of-Line Pixel Clock Wait Count – 1
ELW = End-of-Line Pixel Clock Wait Count – 1

Figure 7-34 shows the pin timing at the end of the frame.

Figure 7-34. Passive Mode End-of-Frame Timing



LCCR0[ENB]—LCD Enable
0—LCD is disabled
1—LCD is enabled

VSP—Vertical Sync Polarity
0—Frame clock is active high, inactive low
1—Frame clock is active low, inactive high

HSP—Horizontal Sync Polarity
0—Line clock is active high, inactive low
1—Line clock is active low, inactive high

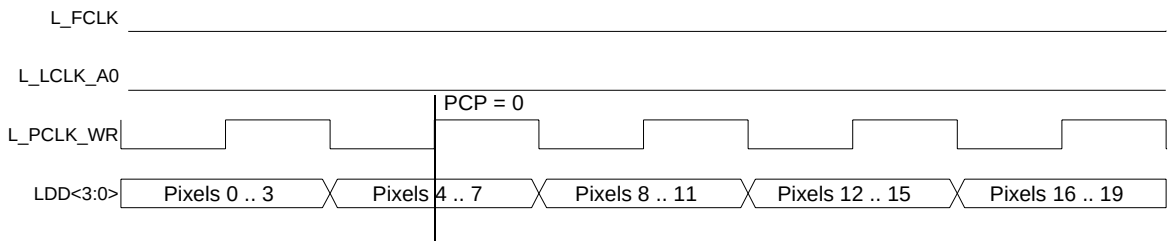
PCP—Pixel Clock Polarity
0—Pixels sampled from data pins on rising edge of clock
1—Pixels sampled from data pins on falling edge of clock

Note that for PCP = 0 data is driven out at the same time and L_PCLK is inverted.

VSW = Vertical Sync Pulse Width – 1
HSW = Horizontal Sync (Line Clock) Pulse Width – 1
BLW = Beginning-of-Line Pixel Clock Wait Count – 1
ELW = End-of-Line Pixel Clock Wait Count – 1
PPL = Pixels Per Line – 1
LPP = Lines Per Panel – 1

Figure 7-35 shows the output data pin timing in monochrome. LCCR3[PCP] defines the edge of the pixel clock on which the data is sampled.

Figure 7-35. Passive Mode Pixel Clock and Data Pin (Monochrome) Timing



PCP—Pixel Clock Polarity
0—Pixels sampled from data pins on rising edge of clock
1—Pixels sampled from data pins on falling edge of clock

Note that for PCP = 1 data is driven out at the same time and L_PCLK is inverted.

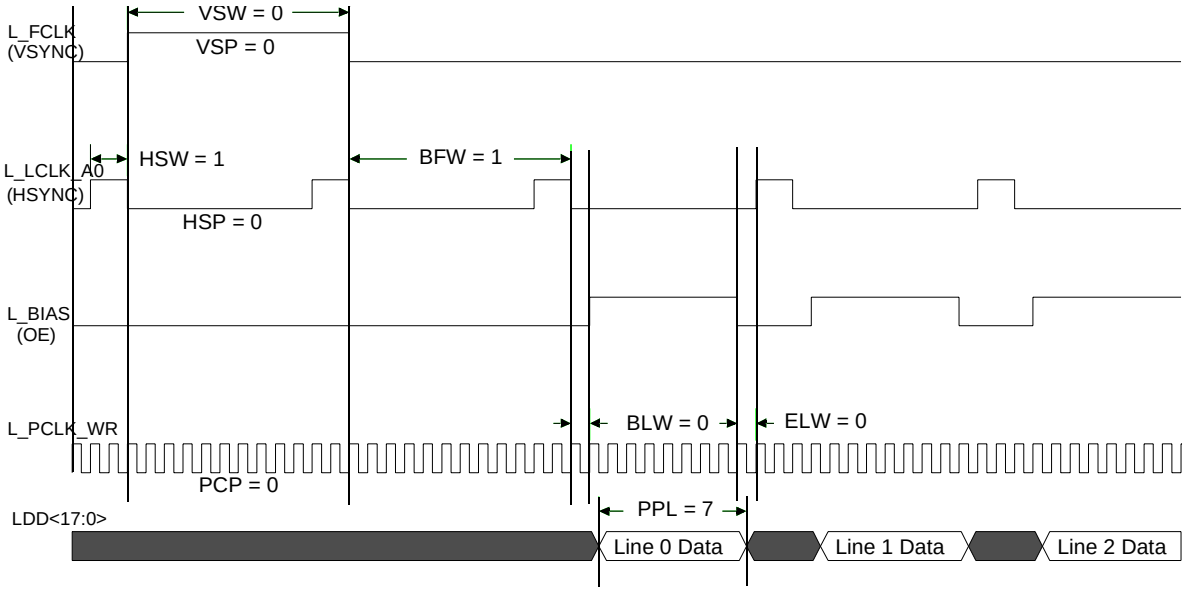
7.4.15.0.2 Active Mode Timing

For active (and passive) LCD panels, when an entire line of pixels has been output to the LCD controller screen, the line-clock pin (L_LCLK) is toggled. Likewise, when an entire frame of pixels has been output to the LCD controller screen, the frame-clock pin (L_FCLK) is toggled.

The pixel clock toggles continuously in this mode as long as the LCD is enabled. The AC bias pin (L_BIAS) functions as an output enable. When L_BIAS is asserted, the display latches data from the LCD pins using the pixel clock. The line-clock pin (L_LCLK_A0) is used as the horizontal synchronization signal, and the frame clock (L_FCLK_RD) as the vertical synchronization signal. The programmable timing of the line and frame-clock pins supports both passive and active mode. Programming options include: Wait-state insertion both at the beginning and end of each line and frame; pixel clock; line clock; frame clock; output-enable signal polarity; and frame-clock pulse width.

Figure 7-36 shows output-pin timing in active mode.

Figure 7-36. Active Mode Timing



ENB—LCD Enable
0—LCD is disabled
1—LCD is enabled

VSP—Vertical Sync Polarity
0—Vertical sync clock is active high, inactive low
1—Vertical sync clock is active low, inactive high

HSP—Horizontal Sync Polarity
0—Horizontal sync clock is active high, inactive low
1—Horizontal sync clock is active low, inactive high

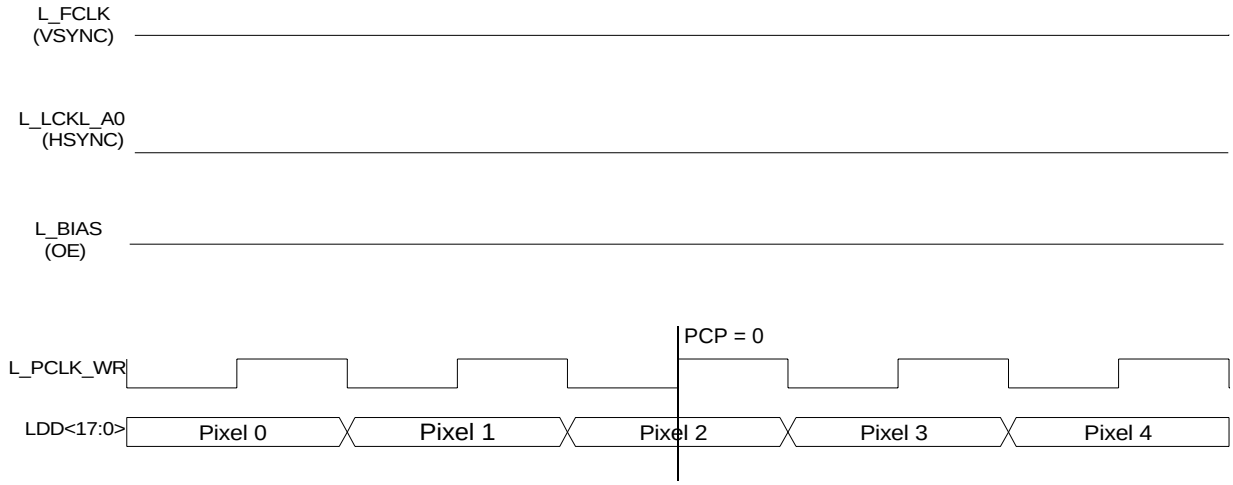
PCP—Pixel Clock Polarity
0—Pixels sampled from data pins on rising edge of clock
1—Pixels sampled from data pins on falling edge of clock

Note that for PCP = 1 data is driven out at the same time and L_PCLK is inverted.

VSW = Vertical Sync Pulse Width – 1
HSW = Horizontal Sync Pulse Width – 1
BFW = Beginning-of-Frame Horizontal Sync Clock Wait Count
BLW = Beginning-of-Line Pixel Clock Wait Count – 1
ELW = End-of-Line Pixel Clock Wait Count – 1
PPL = Pixels Per Line – 1

Figure 7-37 shows the output-data pin timing in active mode.

Figure 7-37. Active Mode Pixel Clock and Data Pin Timing



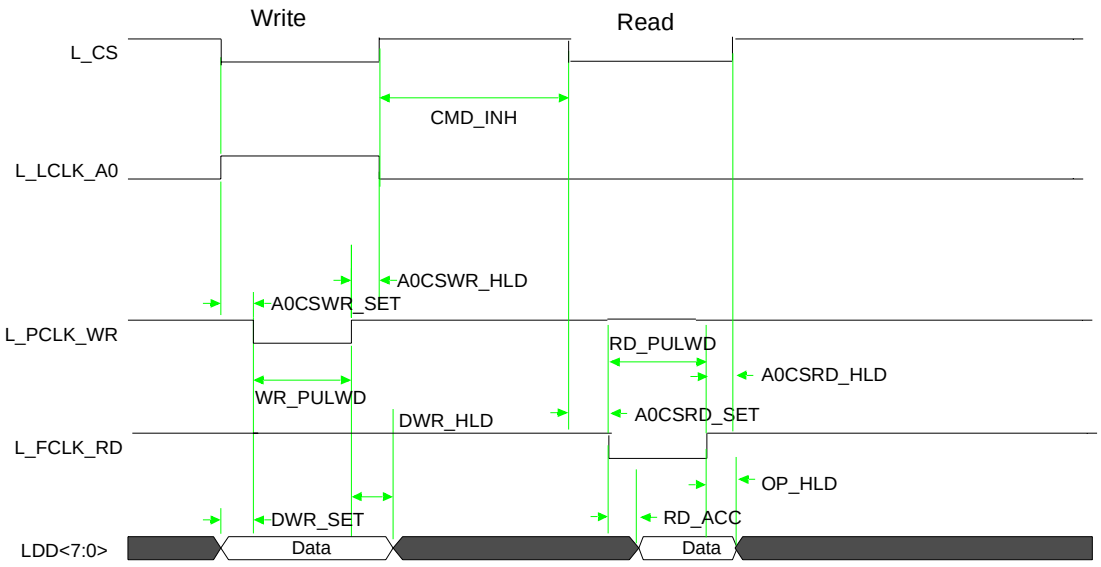
PCP—Pixel Clock Polarity
0—Pixels sampled from data pins on rising edge of clock
1—Pixels sampled from data pins on falling edge of clock

Note that for PCP = 1 data is driven out at the same time and L_PCLK is inverted.

7.4.15.0.3 Smart Panel Mode Timing

For smart panels (LCD panels with internal frame-buffer memory), the data is written into the frame buffer within the smart panel. At the beginning of each frame, the LCD controller sends a command sequence from command RAM, followed by the frame data. The write interface timing is shown in Figure 7-38.

Figure 7-38. Interface with SMART Panels Timing



A0CSRD_SET = A0 and CS Setup Time before L_FCLK_RD is asserted

A0CSRD_HLD = A0 and CS Hold Time after L_FCLK_RD is deasserted

A0CSWR_SET = A0 and CS Setup Time before L_PCLK_WR is asserted

A0CSWR_HLD = A0 and CS Hold Time after L_PCLKWR is deasserted

WR_PULWD = L_PCLK_WR pulse width

RD_PULWD = L_FCLK_RD pulse width

DWR_SET = Data Setup Time before L_PCLK_WR is asserted

DWR_HLD = Data Hold Time after L_PCLK_WR is deasserted

CMD_INH = Command Inhibit time between two writes

RD_ACC = Read Access Time

OP_HLD = Output Hold time from L_FCLK_RD negation

Note that L_CS, L_LCLK_A0 and LDD<7:0> change at the same time. L_PCLK_WR asserts 1 clock after. All the setup widths are all programmable.

7.4.16 Using the LCD Controller Data Pins

Pixel data is removed from the bottom of the output FIFO and is driven in parallel onto the LCD data lines on the edge selected by the pixel-clock polarity bit (LCCR3[PCP]).

- For a 4-bit-wide bus interface, data is driven onto the LCD data lines LDD<3:0>.
- For an 8-bit-wide bus interface, the data is driven onto LDD<7:0>.
- For a 16-bit-wide bus interface, the data is driven onto LDD<15:0>. For 18-bit-wide bus interface, the data is driven onto LDD<17:0>.
- In monochrome dual-scan mode, the pixels for the upper half of the screen are driven onto LDD<3:0> and the lower half to LDD<7:4>.
- In color dual-scan mode, the upper panel pixels are driven onto LDD<7:0> and the lower panel pixels to LDD<15:8>.

See Figure 7-39 for details on how the data is driven onto each LDD pin.

7.4.16.1 Single-Scan/Dual-Scan Select

In passive mode (LCCR0[PAS] cleared), the single-scan/dual-scan select bit (LCCR0[SDS]) selects the type of display control that is implemented by the LCD screen. When SDS = 0, single-scan operation is selected (pixels presented to screen a line at a time), and when SDS = 1, dual-scan operation is selected (pixels presented to screen two lines at a time). In dual-scan mode, the overlays and the hardware cursor are disabled. Single-scan LCD drivers have one line/row shifter and driver for pixels, and one line pointer. Dual-scan LCD controller drivers have two line/row shifters (one for the top half of the screen, one for the bottom), and two line pointers (one for the top half of the screen, one for the bottom). When dual-scan mode is programmed, two of the LCD controller DMA channels are used. DMA channel 0 loads the palette RAM from the frame buffer and drives the upper half of the display; DMA channel 1 drives the lower half. The two channels alternate when fetching data for both halves of the screen, placing encoded pixel values within the two separate input FIFOs. When dual-scan operation is enabled, the LCD controller doubles its pin uses; thus, for monochrome screens, 8 pins are used; for color screens, 16 pins are used.

Table 7-31 shows the LCD data pins used for each mode of operation and the ordering of pixels delivered to a screen for each mode of operation. Figure 7-39 shows the LCD data-pin pixel ordering.

Note: LCCR0[SDS] must be cleared in active mode (PAS = 1).

Table 7-31. LCD Controller Data Pin Utilization

Color/Monochrome Panel	Single Scan/ Dual Scan	Passive/Active Panel	Screen Portion	Pins
Monochrome	Single	Passive	Whole	LDD<3:0>
Monochrome	Single	Passive	Whole	LDD<7:0>†
Monochrome	Dual	Passive	Top	LDD<3:0>
			Bottom	LDD<7:4>
Color	Single	Passive	Whole	LDD<7:0>
Color	Dual	Passive	Top	LDD<7:0>
			Bottom	LDD<15:8>
Color	Single	Active	Whole	LDD<15:0> or LDD<17:0>
LCD Panels with Integrated Frame Buffer	NA	Both	Whole	LDD<7:0>

† Double-pixel data mode (DPD) = 1. Refer to Section 7.4.16.2.

Figure 7-39. LCD Data-Pin Pixel Ordering

Passive Monochrome Single-Scan Display Pixel Ordering

Top Left Corner of Screen

	Column 0	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8
Row 0	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>
Row 1	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>
Row 2	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>
Row 3	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>

Passive Monochrome Single-Scan Double-Pixel Display Pixel Ordering

Top Left Corner of Screen

	Column 0	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8
Row 0	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<0>
Row 1	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<0>
Row 2	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<0>
Row 3	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<0>

Passive Monochrome Dual-Scan Display Pixel Ordering

Top Left Corner of Screen

	Column 0	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8
Row 0	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>
Row 1	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>
⋮									
Row n/2	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<4>
Row n/2+1	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<4>

n = # of rows

Passive Color Single-Scan Display Pixel Ordering

Top Left Corner of Screen

	Column 0 Red	Column 0 Green	Column 0 Blue	Column 1 Red	Column 1 Green	Column 1 Blue	Column 2 Red	Column 2 Green	Column 2 Blue
Row 0	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>	LDD<7>
Row 1	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>	LDD<7>
Row 2	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>	LDD<7>
Row 3	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>	LDD<7>

Passive Color Dual-scan Display Pixel Ordering

Top Left Corner of Screen

	Column 0 Red	Column 0 Green	Column 2 Green	Column 2 Blue	Column 4 Blue	Column 5 Red	Column 5 Green
Row 0	LDD<7>	LDD<6>	LDD<0>	LDD<7>	LDD<1>	LDD<0>	LDD<7>
Row 1	LDD<7>	LDD<6>	LDD<0>	LDD<7>	LDD<1>	LDD<0>	LDD<7>
⋮							
Row n/2	LDD<15>	LDD<14>	LDD<8>	LDD<15>	LDD<9>	LDD<0>	LDD<9>
Row n/2+1	LDD<15>	LDD<14>	LDD<8>	LDD<15>	LDD<9>	LDD<0>	LDD<9>

n = # of rows

7.4.16.2 Output Pin Drive Format for Passive Single Scan

For the monochrome case, the 8-bit pixel value from the palette RAM goes through the dither logic that produces a 1-bit value, which gets stored in the output FIFO. For 4-bit-wide data bus, the output FIFO is 4 bits wide. When the FIFO is loaded with 4 bits, they are driven in parallel onto data bus LDD<3:0> as shown in Table 7-32.

Table 7-32. Monochrome, Passive Single Scan, 4-Bit Bus

Pixel Clock	LDD<3>	LDD<2>	LDD<1>	LDD<0>
0	pixel3	pixel2	pixel1	pixel0
1	pixel7	pixel6	pixel5	pixel4

When double-pixel data (DPD = 1) mode is enabled, the data is driven on to 8-bit-wide bus. When the FIFO is loaded with 8 bits, they are driven in parallel onto the data bus as shown in Table 7-33.

Table 7-33. Monochrome, Passive Single Scan, 8-Bit Bus

Pixel Clock	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>
0	pixel7	pixel6	pixel5	pixel4	pixel3	pixel2	pixel1	pixel0
1	pixel15	pixel14	pixel13	pixel12	pixel11	pixel10	pixel9	pixel8

For color mode, for all pixel depths (2, 4, 8, 16, 18, 19, 24, and 25 bpp) the pixel value from the combination logic goes through the dither logic that produces one bit for each color (red, green and blue), which get loaded into the output FIFO. For an 8-bit-wide data bus, the output FIFO is 8 bits wide. When the FIFO is loaded with 8 bits, they are driven in parallel onto the data bus LDD<7:0>, as shown in Table 7-34.

Table 7-34. Color, Passive Single Scan, 8-Bit Bus

LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>
G_2	R_2	B_1	G_1	R_1	B_0	G_0	R_0
R_5	B_4	G_4	R_4	B_3	G_3	R_3	B_2
B_7	G_7	R_7	B_6	G_6	R_6	B_5	G_5

7.4.16.3 8-Bit Interface for Active Monochrome Single Scan

For monochrome mode, the 8-bits from the combination logic bypass the dithering logic and are directly written into the 8-bit-wide FIFO, which are then driven in parallel on the data bus LDD<7:0>, as shown in Table 7-35.

Table 7-35. Monochrome, Active Single Scan, 8-Bit Bus

LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>
Pixel0							
Pixel1							

7.4.16.4 16-Bit Interface for Active Single Scan

For a pixel depth of 16 bpp, the data is driven in parallel onto the 16-bit data bus LDD<15:0>, as shown in Table 7-36.

Table 7-36. Color, Active Single Scan, 16 bpp, 16-Bit Bus

LDD<15>	LDD<14>	LDD<13>	LDD<112>	LDD<11>	LDD<10>	LDD<9>	LDD<8>	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>
Encoded Pixel0 data[15:0]															
Encoded Pixel1 data[15:0]															

7.4.16.5 18-Bit Interface for Active Single Scan

For pixel depths of 18 bpp and 19 bpp, the data is driven on to 18-bit-wide bus as shown in Table 7-37.

Table 7-37. Color, Active Single Scan, 18 bpp or 19 bpp, 18-Bit Bus

LDD<17>	LDD<16>	LDD<15>	LDD<14>	LDD<13>	LDD<12>	LDD<11>	LDD<10>	LDD<9>	LDD<8>	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>
Encoded Pixel0 data[17:0]																	
Encoded Pixel1 data[17:0]																	

7.4.16.6 Summary of Pin Assignments in Active Mode

Table 7-38 describes the pin assignments for various bus widths in the possible formats.

Table 7-38. Pin Assignments in Active Mode

Base Format	Overlays	PAL_FOR	PD_FOR	LDD<17>	LDD<16>	LDD<15>	LDD<14>	LDD<13>	LDD<12>	LDD<11>	LDD<10>	LDD<9>	LDD<8>	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>
2, 4, 8 bpp	Disabled	0	0	NA		RED<4:0>					GREEN<5:0>						BLUE<4:0>				
4, 8 bpp	Enabled	1	3 LDD_ALT = 0	NA		0	RED<4:0>					GREEN<4:0>					BLUE<4:0>				
4, 8 bpp	Enabled	1	3 LDD_ALT = 1	NA		RED<4:0>					GREEN<4:0>, GREEN<0>						BLUE<4:0>				
2, 4, 8bpp	Disabled	2	3	RED<5:0>							GREEN<5:0>						BLUE<5:0>				
4, 8bpp	Enabled	2	0†	RED<5:0>							GREEN<5:0>						BLUE<5:0>				
16 bpp	Disabled	0†	0	NA		RED<4:0>					GREEN<5:0>						BLUE<4:0>				
16 bpp	Enabled	0†	3 LDD_ALT = 0	NA		0	RED<4:0>					GREEN<4:0>					BLUE<4:0>				
16 bpp	Enabled	0†	3 LDD_ALT = 1	NA		RED<4:0>					GREEN<4:0>, GREEN<0>						BLUE<4:0>				
18 bpp	Disabled	0†	3	RED<5:0>							GREEN<5:0>						BLUE<5:0>				
19 bpp	Enabled	0†	NA	RED<5:0>							GREEN<5:0>						BLUE<5:0>				

NOTE:

† Indicates that the value of this field is ignored in this mode. The use of the reset value of 0 is recommended, however, any value is acceptable.

7.4.16.7 8-Bit Interface for Smart Panels

The data from the output FIFO is driven on to the 8-bit-wide bus in three cycles as shown in Table 7-39. Data for each color (red, green, and blue) can be 5, 6, 7, and 8, depending on the pixel depths and format as discussed in the Section 7.4.6.

Table 7-39. 8-Bit Interface for Smart Panels

LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>
Pixel0 Red Data							
Pixel0 Green Data							
Pixel0 Blue Data							

7.5 Register Descriptions

7.5.1 Using LCD Control Registers

This section summarizes the various types of LCD control registers and describes how to use them. Complete register descriptions for each register follow, beginning with Section 7.5.2.

7.5.1.1 LCD Panel Controller Registers

Write to the LCD Control register to program the following:

- Enable or disable the LCD controller.
- Define the height and width of each overlay.
- Indicate single- versus dual-scan display mode, color versus monochrome mode, passive versus active display.
- Polarity of the control.
- Pulse width of the line and frame clocks, pixel clock and AC bias pin frequency; the number of wait states to insert before and after each line, after each frame, and program various interrupt masks.
- An additional control field exists to tune the DMA performance based on the type of memory system in which the PXA27x processor is used. This field controls the placement of a minimum delay between each LCD DMA request to ensure enough bus bandwidth is given to other system bus masters for accesses.

The Status registers contain bits that signal input and output FIFO underrun errors, DMA bus errors, when the DMA starts and ends a frame, when the last active frame has completed after the LCD is disabled, and each time the AC bias pin has toggled a programmed number of times. Each of these hardware-detected events can signal an interrupt request to the interrupt controller.

7.5.1.2 LCD Controller DMA Registers

The LCD controller has seven fully independent DMA channels to transfer palette data, the frame data, the cursor data, and the command data from the internal or the external memory to the LCD controller. DMA channel 0 is used for the base layer or to display data for the upper data on the lower screen for dual-scan mode. DMA channel 1 is used for Overlay 1 or to display data on the lower screen for dual-scan mode; DMA channels 2, 3, and 4 are used for Overlay 2; channel 5 is used for the hardware cursor. DMA channel 6 transfers data to the command register. The palette RAMs are always loaded through their respective DMA channels. All of the information for the DMA transfers is maintained in registers within the LCD DMA controller. These registers are loaded from frame descriptors located in memory. Typically, one descriptor is used for each frame in memory. Dedicated descriptors load the palette RAMs. Multiple descriptors can be chained together in a list, making it possible for the DMA to transfer data from a (essentially infinite) number of discontinuous locations.

Software programs the DMA descriptor addresses; hardware programs the others. Refer to the Section 7.5.16 on DMA registers for a complete description of how the DMA is programmed.

Note: Dual-scan operation does not permit the use of overlays.

7.5.1.2.1 DMA Frame Descriptors

Although the FDADRx register can be (and is) loaded by software, the FSADRx, FIDRx, and LDCMDx registers can be loaded only indirectly from DMA frame descriptors. A frame descriptor is a four-word (32-bits/word) block, aligned on a 16-byte boundary, in memory.

Word[0] contains the value for the FDADRx register.

Word[1] contains the value for the FSADRx register.

Word[2] contains the value for the FIDRx register.

Word[3] contains the value for the LDCMDx register.

The FDADRx register must be written with the location of the first descriptor by software before enabling the LCD controller. Once the LCD controller is enabled, the first descriptor is read and all four registers are written by the DMA controller. The next frame descriptor pointed to by the FDADRx register is loaded into the registers of the associated DMA channel after all of the data for the current descriptor has been transferred. The FDADRx register is bypassed only when the frame-branch-register (FBRx) branch (BRA) bit is set; in this case, the frame-branch address fetches the descriptor for the next frame. Branches load a new palette or process a regular frame. If only one frame buffer is used in external memory, the FDADRx register must be programmed to point back to itself.

The DMA registers bitmaps are located beginning with Table 7-54.

7.5.1.3 LCD Buffer Strength Control Register

The LCD output buffers drive strengths can be controlled through the use of the LCD Buffer Strength Control register. See Table 7-56.

7.5.2 LCD Controller Control Register 0 (LCCR0)

Table 7-40 shows the location of all bit-fields located in LCD Control register 0 (LCCR0). All bits in the control registers must be programmed before setting LCCR0[ENB], which enables the LCD controller. A word write can be used to configure LCCR0 while setting ENB after all other control registers have been programmed. Also, the LCD controller must be disabled (by clearing LCCR0[ENB]) when changing the state of any control bit within the LCD controller. Reserved bits are unknown at reset, must be written with zeros, and may return zeros or ones when read.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-40. LCCR0 Bit Definitions (Sheet 1 of 8)

Physical Address 0x4400_0000 — **LCCR0** — **LCD Controller**

Bit	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
User Settings																																
	reserved					LDDALT	OUC	CMDIM	RDSTM	LCDT	OUM	BSM0	PDD								QDM	DIS	DPD	reserved	PAS	EOFM0	IUM	SOFM0	LDM	SDS	CMS	ENB
Reset	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:27	—	—	reserved
26	R/W	LDDALT	LDD Alternate Mapping Control Bit LDDALT defines the output format of the output pixel driven on the LDD pins when Base pixel format is RGBT16 (i.e when Base BPP = 4 and PDFOR = 3 or BPP = 0 or 1 or 2 or 3, PAL_FOR = 1 AND PDFOR = 3). 0 = Output is driven as LDD<15:0> = '0' & (5R) & (5G) & (5B) 1 = Output is driven as LDD<15:0> = (5R) & (6G) & (5B)
25	R/W	OUC	Overlay Underlay Control Bit OUC sets Overlay 1 and Overlay 2 to function as overlays or underlays with respect to the Base Frame. When OUC is cleared Overlay 1 and Overlay 2 reside below the Base Frame. When OUC is set, Overlay 1 and Overlay 2 reside above the Base frame. See Table 7-3. 0 = Overlay 1 and Overlay 2 are underlays with respect to the Base 1 = Overlay 1 and Overlay 2 are overlays with respect to the Base
24	R/W	CMDIM	LCD Command Interrupt Mask The Command Interrupt Mask (CMDIM) bit masks interrupt requests that are asserted whenever the LCD controller executes an interrupt command. When CMDIM is cleared, the interrupt is enabled, and whenever the LCD controller executes an interrupt command in the command queue the LCSR0[CMD_INT] status bit is set, an interrupt request is made to the interrupt controller. When CMDIM is set, the interrupt is masked and the state of the CMD_INT status bit is ignored by the interrupt controller. Note that setting CMDIM does not affect the current state of CMD_INT or the LCD controller's ability to set and clear CMD_INT; it only blocks the generation of the interrupt request. 0 = Instruction "command interrupt" generates an interrupt status sent to the interrupt controller 1 = Instruction "command interrupt" does not generate an interrupt (RD_ST status bit is ignored)

Table 7-40. LCCR0 Bit Definitions (Sheet 2 of 8)

Physical Address 0x4400_0000	LCCR0	LCD Controller

	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
User Settings																																
Bit	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					LDDALT	OUC	CMDIM	RDSTM	LCDT	OUM	BSM0	PDD								QDM	DIS	DPD	reserved	PAS	EOFM0	IUM	SOFM0	LDM	SDS	CMS	ENB
Reset	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
23	R/W	RDSTM	LCD Read Status Interrupt Mask The Read Status Mask (RDSTM) bit masks or enables interrupt requests that are asserted whenever the LCD controller executes a read command. When RDSTM is cleared, the interrupt is enabled, and whenever the LCD controller reads from the panel LCSR0[RD_ST] is set, an interrupt request is made to the interrupt controller. When RDSTM is set, the interrupt is masked and the state of the RD_ST status bit is ignored by the interrupt controller. Note that setting RDSTM does not affect the current state of RD_ST or the LCD controller's ability to set and clear RD_ST; it only blocks the generation of the interrupt request. 0 = Read from the panel with frame buffer generate an interrupt status sent to the interrupt controller 1 = Read from the panel with frame buffer does not generate an interrupt (RD_ST status bit is ignored)
22	R/W	LCDT	LCD Panel Type The LCD panel type (LCDT) bit specifies the type of LCD panel. When this bit is set (LCDT = 1), the LCD panel has internal frame buffer. This changes the functionality of the interface pins. When this bit is cleared (LCDT = 0), the LCD panel is regular panel without internal frame buffer. 0 = LCD Panel with no internal frame buffer 1 = LCD Panel with internal frame buffer
21	R/W	OUM	Output FIFO Underrun Mask. 0 = FIFO underrun errors generate an interrupt (state of OU status sent to the interrupt controller). 1 = FIFO underrun errors do not generate an interrupt (OU status bits ignored).
20	R/W	BSM0	Branch Status Mask The branch status mask (BSM0) bit masks or enables the interrupt requests that are asserted after branching to a new frame. When BSM0 = 0, branch status interrupts are enabled, and whenever the branch status BS 0bit within the LCD Status register 0 (LCSR0) is set (one), an interrupt request is made to the interrupt controller. When BSM0 = 1, branch status interrupt is masked; the state of the branch status bit is ignored by the interrupt controller. Note that programming BSM0 = 1 does not affect the current state of BS0 or the LCD controller ability to set and clear BS0; it only blocks the generation of the interrupt request. 0 = Generates an interrupt after branching to a new frame (state of BS0 status sent to the interrupt controller). 1 = BS condition does not generate an interrupt (BS0 status bit ignored).

Table 7-40. LCCR0 Bit Definitions (Sheet 3 of 8)

Physical Address 0x4400_0000 — **LCCR0** — **LCD Controller**

Bit	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
User Settings																																
	reserved					LDDALT	OUC	CMDIM	RDSTM	LCDT	OUM	BSM0	PDD								QDM	DIS	DPD	reserved	PAS	EOFM0	IUM	SOFM0	LDM	SDS	CMS	ENB
Reset	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
19:12	R/W	PDD	Palette DMA Request Delay The 8-bit palette DMA Request Delay (PDD) field selects the minimum number of internal bus clock cycles to wait between the servicing of each DMA request issued while the on-chip palette RAMs are being loaded. Using PDD allows other system bus masters to gain access to shared memory in between palette DMA loads. Use PDD carefully, as it can severely degrade LCD controller performance if not used properly. Leave PDD at zero and only add a delay when it is necessary. Note that PDD does not apply to normal input FIFO DMA requests for Frame buffer information since these DMA requests do not occur back to back. The input FIFO DMA request rate is a function of the rate at which pixels are displayed on the LCD panel. Value (from 0 to 255) that specifies the number of pxclk internal bus clocks to wait before requesting another burst of palette data. The counter starts decrementing when the first word is written to the input FIFO buffer. Programming PDD = 0x0000_0000 disables this function.
11	R/W	QDM	LCD Quick Disable Mask The LCD quick disable interrupt mask (LDM) bit masks or enables interrupt requests that are asserted after the LCD enable (ENB) bit is cleared and the DMA finishes the current burst transfer. The LCD controller immediately stops requesting new data and the current frame is not completed. This shutdown is intended to be used for Sleep shutdown. When QDM = 0, the interrupt is enabled, and whenever the LCD quick disable (QD) status bit within the LCD Status register 0 (LCSR0) is set (one), an interrupt request is made to the interrupt controller. When QDM = 1, the interrupt is masked and the state of the QD status bit is ignored by the interrupt controller. Note that programming QDM = 1 does not affect the current state of QD or the LCD controller ability to set and clear QD; it only blocks the generation of the interrupt request. 0 = Generate an interrupt after quick disable (state of QD status sent to the interrupt controller). 1 = QD status does not generate an interrupt (QD status bit ignored).
10	R/W	DIS	LCD Disable During LCD controller operation, setting DIS = 1 will cause the LCD controller to finish the current frame and cleanly shut down. Completion of the current frame is signaled by the LCD when it sets the LCD disable done flag (LDD) in the LCD Status register. Use a read-modify-write procedure to set this bit because the other bit fields within LCCR0 continue to be used until the current frame is completed. The LCD enable (ENB) bit is also cleared by hardware when the disable is completed. 0 = LCD controller has not been enabled and subsequently disabled. 1 = LCD controller has been disabled, or is in the process of disabling.

Table 7-40. LCCR0 Bit Definitions (Sheet 4 of 8)

Physical Address 0x4400_0000 — **LCCR0** — **LCD Controller**

	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
Bit																																
	reserved					LDDALT	OUC	CMDIM	RDSTM	LCDT	OUM	BSM0	PDD								QDM	DIS	DPD	reserved	PAS	EOFM0	IUM	SOFM0	LDM	SDS	CMS	ENB
Reset	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
9	R/W	DPD	Double-Pixel Data (DPD) Pin Mode The double-pixel data (DPD) pin mode bit selects whether 4 or 8 data pins output pixel data to the LCD screen in single-scan monochrome mode. When DPD = 0, LDD<3:0> pins output 4 pixel values each pixel clock transition; when DPD = 1, LDD<7:0> pins output 8 pixel values each pixel clock. Note that DPD does not affect dual-scan monochrome mode or any of the color modes. 0 = LDD<3:0> pins output 4-pixel values each pixel clock transition. 1 = LDD<7:0> pins output 8-pixel values each pixel clock.
8	—	—	reserved

Table 7-40. LCCR0 Bit Definitions (Sheet 5 of 8)

Physical Address 0x4400_0000 — **LCCR0** — **LCD Controller**

Bit	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
User Settings																																
	reserved					LDDALT	OUC	CMDIM	RDSTM	LCDT	OUM	BSM0	PDD								QDM	DIS	DPD	reserved	PAS	EOFM0	IUM	SOFM0	LDM	SDS	CMS	ENB
Reset	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
7	R/W	PAS	Passive/Active Display Select The passive/active display select (PAS) bit selects whether the LCD controller operates in passive (STN) or active (TFT) display control mode. When PAS = 0, passive mode is selected, all LCD data flow operates normally (including the use of the LCD dither logic), and the LCD controller pin timing operates as described in Section 7.4.15.0.1. When PAS = 1, active (TFT) mode is selected. For 4- and 8-bpp modes, pixel data is transferred using the DMA from the internal or the external memory to the input FIFO, is unpacked and used to select an entry from the palette, just like passive mode. However, the value read from the palette bypasses the LCD dither logic, and is sent directly to the output FIFO to be output on the LCD data pins. For pixel depths greater than 8 bpp, the pixel is transferred using DMA from the internal or the external memory to the input FIFO. The pixel data bypasses both the palette and the dither logic and is placed directly in the output FIFO to be output on the LCD data pins. Note that the pin timing of the LCD changes when active mode is selected. Additionally, the LCD controller can be configured in active color display mode and used with an external DAC and optionally an external palette to drive a video monitor. Note that only monitors that implement the RGB data format can be used; the LCD controller does not support the NTSC standard. However, the 2X pixel clock mode allows the LCD controller to easily interface with an NTSC encoder. If the panel that is being controlled contains more than 16 data pin inputs, users can still use the LCD controller by using one of the modes described in Section 7.5.15. If the panel's full range of colors must be maintained and the granularity of the spectrum increased, the 16 LCD data pins must be interfaced to the panel's most significant R, G, and B pixel data input pins, and the least significant R, G, and B data pins must be tied an appropriate level. If instead, the granularity of the spectrum must be maintained and the overall range of colors limited, the 16 LCD data pins must be interfaced to the panel's least significant R, G, and B pixel data input pins, and the most significant data pins must be tied to an appropriate level. A third option that may yield better results is to replicate the upper bits on the lower bits; this is known as *dumb dithering*. For panels with internal frame buffer, set this bit. The dither logic is bypassed when this bit is set. 0 = Passive display operation enabled. Dither logic is enabled. 1 = Active display operation enable. Dither logic is bypassed,.

Table 7-40. LCCR0 Bit Definitions (Sheet 6 of 8)

Physical Address 0x4400_0000																LCCR0															LCD Controller															
User Settings																																												
Bit	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					LDDALT	OUC	CMDIM	RDSTM	LCDT	OUM	BSM0	PDD								QDM	DIS	DPD	reserved	PAS	EOFM0	IUM	SOFM0	LDM	SDS	CMS	ENB													
Reset	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0												

Bits	Access	Name	Description
6	R/W	EOFM0	End of Frame Mask for Channel 0 and for Channel 1 (Dual Scan) This bit masks or enables interrupt requests that are asserted at the end of each frame (when the DMA length of transfer counter decrements to zero). When EOFM0 = 0, the interrupt is enabled, and whenever LCSR0[EOF0] is set, an interrupt request is made to the interrupt controller. When EOFM0 is set, the interrupt is masked and the state of the EOF status bit is ignored by the interrupt controller. Note that setting EOFM0 does not affect the current state of LCSR0[EOF0] or the ability of the LCD controller to set and clear LCSR0[EOF0]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt at the end of a frame (state of LCSR0[EOF0] sent to the interrupt controller). 1 = EOF condition does not generate an interrupt (LCSR0[EOF0] ignored).
5	R/W	IUM	Input FIFO Underrun Mask This bit masks or enables interrupt requests that are asserted whenever an input FIFO underrun error occurs. When IUM = 0, underrun interrupts are enabled, and whenever an input FIFO underrun (LCSR0[IU0, IU1]) is set, an interrupt request is made to the interrupt controller. When IUM = 1, underrun interrupts are masked; the state of the underrun status bits (LCSR0[IU0, IU1]) is ignored by the interrupt controller. Note that programming IUM = 1 does not affect the current state of these status bits or the LCD controller's ability to set and clear them; it only blocks the generation of the interrupt requests. 0 = FIFO underrun errors generate an interrupt (state of LCSR0[IU0, IU1] (for dual scan) sent to the interrupt controller). 1 = FIFO underrun errors do not generate an interrupt (LCSR0[IU0, IU1] ignored).
4	R/W	SOFM0	Start of Frame Mask for Channel 0 and for Channel 1 (Dual Scan) This bit masks or enables interrupt requests that are asserted at the beginning of each base frame when the LCD frame Descriptor has been loaded into the internal DMA registers. When SOFM0 = 0, the interrupt is enabled, and whenever LCSR0[SOF0] is set, an interrupt request is made to the interrupt controller. When SOFM0 = 1, the interrupt is masked and LCSR0[SOF0] is ignored by the interrupt controller. Note that programming SFM0 = 1 does not affect the current state of LCSR0[SOF0] or the LCD controller's ability to set and clear LCSR0[SOF0]; it only blocks the generation of the interrupt request. 0 = Starting a new frame (after loading frame Descriptor) generates an interrupt (state of LCSR0[SOF0] sent to the interrupt controller). 1 = SOF condition does not generate an interrupt (LCSR0[SOF0] ignored).

Table 7-40. LCCR0 Bit Definitions (Sheet 7 of 8)

Physical Address 0x4400_0000 — **LCCR0** — **LCD Controller**

Bit	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
User Settings																																
	reserved					LDDALT	OUC	CMDIM	RDSTM	LCDT	OUM	BSM0	PDD									QDM	DIS	DPD	reserved	PAS	EOFM0	IUM	SOFM0	LDM	SDS	CMS	ENB
Reset	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
3	R/W	LDM	LCD Disable Done Mask The LCD disable done interrupt mask (LDM) bit masks or enables interrupt requests that are asserted after the LCD is disabled, and the frame currently being output to the pins has completed. When LDM = 0, the interrupt is enabled, and whenever the LCD disable done (LDD) status bit within the LCD status register 0 (LCSR0) is set (one), an interrupt request is made to the interrupt controller. When LDM = 1, the interrupt is masked and the state of the LDD status bit is ignored by the interrupt controller. Note that programming LDM = 1 does not affect the current state of LDD or the LCD controller's ability to set and clear LDD; it only blocks the generation of the interrupt request. This interrupt is particularly useful when users need to ensure the LCD has been disabled and the current frame that is being output to the pins has completed. Note that clearing LCD Enable (ENB) forces a "quick reset", and LDD is not set. This mask bit only applies to regular shutdowns using the LCD disable (DIS) bit. 0 = LCD disable done condition generates an interrupt (state of LDD status sent to the interrupt controller). 1 = LCD disable done condition does not generate an interrupt (LDD status bit ignored).

Table 7-40. LCCR0 Bit Definitions (Sheet 8 of 8)

Physical Address 0x4400_0000	LCCR0	LCD Controller

User Settings																															
Bit	**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					LDDALT	OUC	CMDIM	RDSTM	LCDT	OUM	BSM0	PDD								QDM	DIS	DPD	reserved	PAS	EOFM0	IUM	SOFM0	LDM	SDS	CMS	ENB
Reset	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	R/W	SDS	Single-Scan/dual-scan Display Select 0 = Single-scan display enabled. LDD<3:0> used for monochrome, LDD<7:0> used for passive color and LCD<15:0> used for active color. 1 = Dual-scan display enabled. LDD<7:0> used for monochrome, LDD<15:0> used for color. **NOTE:** SDS must be 0 in active mode (PAS = 1).
1	R/W	CMS	Color/Monochrome Select The color/monochrome select (CMS) bit selects whether the LCD controller operates in color or monochrome mode. When CMS = 0, Color mode is selected, palette entries are 25 bits wide, 8 or 16 data pins are enabled for single-scan mode, 16 data pins are enabled for dual-scan mode, and all three dither blocks are used, one each for the red, green, and blue pixel components. When CMS = 1, monochrome mode is selected; 4 or 8 data pins are enabled for single-scan mode, and 8 data pins are enabled for dual-scan mode. 0 = color operation enabled. 1 = Monochrome operation enabled.
0	R/W	ENB	LCD Controller Enable The LCD controller enable (ENB) bit enables and quickly disables all LCD controller operation. All LCD Control registers must be initialized before setting ENB. Users can program LCCR0 last, and configure all bit fields at the same time using a word write to the register. If users clear ENB while the LCD controller is enabled, the LCD controller will immediately stop requesting data from the LCD DMA controller, and the current frame will not complete. Quick disable is intended to be used for sleep shutdown. Shutdown of the LCD controller at the end of the frame can be accomplished with the LCD disable bit (LCCR0[DIS]). Note that there are separate maskable interrupts for Quick disable and Regular disable. 0 = LCD controller disabled. 1 = LCD controller enabled.

7.5.3 LCD Controller Control Register 1 (LCCR1)

LCD Controller Control register 1 (LCCR1) contains four bit fields that are used as modulus values for a collection of down counters, each of which performs a different function to control the timing of several of the LCD pins. The LCD controller must be disabled (LCCR0[ENB] = 0) when changing the state of any field within this register.

Table 7-41. LCCR1 Bit Definitions (Sheet 1 of 2)

Physical Address 0x4400_0004	LCCR1	LCD Controller

User Settings

Bit	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
	BLW								ELW								HSW						PPL									
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

Bits	Access	Name	Description
31:24	R/W	BLW	Beginning-of-Line Pixel Clock Wait Count For passive and active displays, the 8-bit Beginning-of-Line Pixel Clock Wait Count (BLW) field specifies the number of dummy pixel clocks to insert at the beginning of each line or row of pixels. After the line clock for the previous line has been negated, the value in BLW counts the number of pixel clocks to wait before starting to output the first set of pixels in the next line. BLW generates a wait period ranging from 1 to 256 pixel clock cycles. Note that the pixel clock pin, L_PCLK, does not toggle during these dummy pixel clock cycles in passive display mode (pixel clock toggles continuously in active display mode). For LCD panels with internal Frame buffer, the 8-bit field (BLW) specifies the pulse width of the Write or Read signal (L_PCLK_WR or L_FCLK_RD), which is equal to (BLW+1)*LCD_CLK_PERIOD. Value (from 0 to 255).specifies the number of pixel clock periods to add to the beginning of a line transmission before the first set of pixels is output to the display. BOL wait = (BLW+1).
23:16	R/W	ELW	End-of-Line Pixel Clock Wait Count For passive and active displays, the 8-bit End-of-Line Pixel Clock Wait Count (ELW) field specifies the number of dummy pixel clocks to insert at the end of each line or row of pixels before pulsing the line clock pin. Once a complete line of pixels is transmitted to the LCD panel, the value in ELW counts the number of pixel clocks to wait before pulsing the line clock. ELW generates a wait period ranging from 1 to 256 pixel clock cycles. Note that the pixel clock pin, L_PCLK, does not toggle during the these dummy pixel clock cycles in passive display mode (pixel clock toggles continuously in active display mode). For LCD panels with an internal frame buffer, ELW specifies the setup and hold times for A0 (L_LCLK_A0), Data (LDD<7:0>), and CS (L_CS) with respect to the Write or Read signal (L_PCLK_WR or L_FCLK_RD). The setup and hold time is equal to (ELW+1)*LCD_CLK_PERIOD. EOL = (ELW+1).

Table 7-41. LCCR1 Bit Definitions (Sheet 2 of 2)

Physical Address 0x4400_0004	LCCR1	LCD Controller

	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
User Settings																																
Bit	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
	BLW								ELW								HSW						PPL									
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

Bits	Access	Name	Description
15:10	R/W	HSW	Horizontal Sync Pulse Width The 6-bit Horizontal Sync Pulse Width (HSW) field specifies the pulse width of the line clock in passive mode or horizontal synchronization pulse in active mode. L_LCLK is asserted each time a line or row of pixels is output to the display and a programmable number of pixel clock Wait states have elapsed. When line clock is asserted, the value in HSW is transferred to a 6-bit Down counter, which uses the programmed pixel clock frequency to decrement. When the counter reaches zero, the line clock is negated. HSW can be programmed to generate a line clock pulse width ranging from 1 to 64 pixel clock periods. Note that the pixel clock does not toggle during the line clock pulse in passive display mode, but does toggle in active display mode. Also note that the polarity (active and Inactive state) of the line clock pin is programmed using the horizontal sync polarity (HSP) bit in LCCR3. For LCD Panels with an internal frame buffer, HSW indicates the input Hi-Z hold time after L_FCLK_RD negates during reads from the panel. The count value specifies the number of LCD clocks after L_FCLK_RD negates that the LCD controller will not drive the output data bus. Value (from 0 to 63). HSYNC pulse width = (HSW+1).
9:0	R/W	PPL	Pixels per Line for the Base Frame The pixels per line (PPL) bit-field specifies the number of pixels in each line or row on the LCD panel for the base frame. PPL is a 10-bit value that represents between 1 to 800 pixels per line. PPL counts the correct number of pixel clocks that must occur before the line clock can be asserted. See Section 7.4.13 to see the restrictions on pixels per line. If the display used is not naturally a multiple of the above, “dummy” pixels must be added to each line to keep the frame buffer aligned in memory. For example, if the display being controlled is 250 pixels wide and the pixel-size is 8-bits, the nearest greater multiple of 8 is 256. Users should program PPL to 256 (0b01_0000_0000). In this case, users must also add the six extra dummy pixel values to the frame buffer. **NOTE:** Users must also ensure that the display being controlled will ignore the additional pixel clocks at the end of each line resulting from the dummy pixel values being sent to the screen. Value (from 0 to 799). Actual pixel per line = PPL+1

7.5.4 LCD Controller Control Register 2 (LCCR2)

LCD Controller Control register 2 (LCCR2) contains four bit fields that are used as modulus values for a collection of down counters, each of which performs a different function to control the timing of several LCD pins.

Table 7-42. LCCR2 Bit Definitions (Sheet 1 of 3)

Physical Address 0x4400_0008	LCCR2	LCD Controller

	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
User Settings																																
Bit	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
	BFW								EFW								VSW						LPP									
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

Bits	Access	Name	Description
31:24	R/W	BFW	Beginning-of-Frame Line Clock Wait Count The 8-bit Beginning-of-Frame Line Clock Wait Count (BFW) field is used in active mode (PAS = 1) to specify the number of line clocks to insert at the beginning of each frame before the first set of pixels is output to the display. The BFW count starts just after the VSYNC signal for the previous frame has been negated. After this has occurred, the value in BFW counts the number of line clock periods to insert before starting to output pixels in the next frame. BFW generates a wait period ranging from 0 to 255 extra line clock cycles (BFW = 0x00 disables the BOF wait count). Note that the line clock pin, L_LCLK, does toggle during the generation of the BFW line-clock wait periods. In passive mode, BFW must be set to zero such that no beginning-of-frame Wait states are generated. Use VSW exclusively in passive mode to insert line-clock Wait states to allow the LCD DMA to fill the palette and process a number of pixels before the start of the next frame. **NOTE:** For LCD panels with internal frame buffer, these bits are ignored.
23:16	R/W	EFW	End-of-Frame Line Clock Wait Count The 8-bit End-of-Frame Line Clock Wait Count (EFW) field is used in active mode (PAS = 1) to specify the number of line clocks to insert at the end of each frame. Once a complete frame of pixels is transmitted to the LCD display, the value in EFW counts the number of line clock periods to wait. After the count has elapsed, the VSYNC (L_FCLK) signal is pulsed. EFW generates a wait period ranging from 0 to 255 line clock cycles (setting EFW = 8'h00 disables the EOF wait count). Note that the line clock pin L_LCLK toggles during the generation of the EFW line clock periods. In passive mode, EFW must be set to zero such that no end-of-frame Wait states are generated. Use VSW exclusively in passive mode to insert line-clock wait states to allow the LCD DMA to fill the palette and process a number of pixels before the start of the next frame. **NOTE:** For LCD panels with internal frame buffer, these bits are ignored.

Table 7-42. LCCR2 Bit Definitions (Sheet 2 of 3)

Physical Address 0x4400_0008	LCCR2	LCD Controller

	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
User Settings																																
Bit	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
	BFW								EFW								VSW						LPP									
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

Bits	Access	Name	Description
15:10	R/W	VSW	Vertical Sync Pulse Width VSW specifies the pulse width of the vertical synchronization pulse in active mode, or adds extra dummy line clock wait states between the end and beginning of frame in passive mode. In active mode (PAS = 1), L_FCLK generates the vertical-sync signal and is asserted each time the last line or row of pixels for a frame is output to the display and a programmable number of line-clock wait-states have elapsed as specified by LCCR1[ELW]. VSW can be programmed to generate a vertical-sync pulse width ranging from 1 to 64 line clock periods. VSW must be programmed with the preferred number of line clocks minus one. The polarity of L_FCLK is programmed using LCCR3[FCP]. The line clock does toggle during VSYNC. In passive mode (PAS = 0), VSW does not affect the timing of the L_FCLK pin, but rather adds extra line clock wait states between the end of each frame and the beginning of the next frame. VSW can be programmed to generate from 1 to 64 dummy line clock periods between each frame. Program VSW to ensure that enough wait states occur between frames such that the LCD DMA is able to fully load the on-chip palette, as well as allowing a sufficient number of encoded pixel values to be input from the frame buffer, to be processed by the dither logic, and placed in the output FIFO, ready to be output to the LCD data pins. The number of wait states required is system dependent. The factors that determine the number of wait states include: Palette buffer size (if loaded; 8, 16, 32, 64, 512 or 1024 bytes), memory system speed (number of wait states, burst speed, number of beats), and what value is programmed in LCCR0[PDD]. Note that the line clock pin does toggle during the insertion of the line clock Wait state periods. Passive LCD displays require that the frame clock is active on the rising edge of the first-line clock pulse of each frame, with adequate setup and hold time. To meet this requirement, the LCD controller frame clock pin is asserted on the rising edge of the first pixel clock for each frame. The frame clock remains asserted for the remainder of the first line as pixels are output to the display, and it is then negated on the rising edge of the first pixel clock of the second line of each frame. VSYNC width = (VSW+1). **NOTE:** For LCD panels with internal frame buffer, these bits are ignored.

Table 7-42. LCCR2 Bit Definitions (Sheet 3 of 3)

Physical Address 0x4400_0008																LCCR2												LCD Controller										
User Settings																																				
Bit	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
	BFW								EFW								VSW						LPP									
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

Bits	Access	Name	Description
9:0	R/W	LPP	Lines per Panel for the Base Frame The Lines per Panel (LPP) bit field specifies the number of lines or rows present on the base frame. In single-scan mode, it represents the total number of lines for the entire LCD display. In dual-scan mode, it represents half the number of lines of the entire LCD display because it is split into two panels. LPP is a 10-bit value that represents between 1 and 600 lines per screen. LPP must be programmed with the preferred height of the display minus one. LPP counts the correct number of line clocks that must occur before the frame clock can be pulsed. Value (from 0 to 599) specifies the number of lines per panel. For single-scan mode, this represents the total number of lines on the LCD display; for dual-scan mode, this represents half the number of lines on the whole LCD display. Lines/panel = (LPP+1).

7.5.5 LCD Controller Control Register 3 (LCCR3)

LCD Controller Control register 3 (LCCR3) contains different bit fields to control various functions within the LCD controller. The LCD controller must be disabled (LCCR0[ENB] = 0) when changing the state of any field within this register, with the exception of the LCCR3[PCD] bit field. Software should wait for the end of frame (indicated by LCSR0[EOF0]) before the write to PCCR3[PCD].

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-43. LCCR3 Bit Definitions (Sheet 1 of 5)

Physical Address 0x4400_000C — **LCCR3** — **LCD Controller**

User Settings

Bit	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
	PDFOR		BPP3	reserved	DPC	BPP			OEP	PCP	HSP	VSP	API				ACB								PCD							
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

Bits	Access	Name	Description
31:30	R/W	PDFOR	Pixel Data Format The pixel data format (PDFOR) specifies the number of bits allocated for R, G, and B for various pixel depths. See Section 7.4.6 for more details. This bit field specifies the data format for various pixel depths and determines how the how the data is driven onto the LDD data pins. For pixel depths where only one RGB format is possible (19bpp and 25bpp), the value of PDFOR is ignored. 0b00 = Format 1 0b01 = Format 2 0b10 = Format 3 0b11 = Format 4 See Section 7.4.6 for more definition of the formats.
29	R/W	BPP3	Bits per Pixel Used in conjunction with BPP to configure the number of bits per pixel. Refer to the BPP field for the bit-field settings.
28	—	—	reserved
27	R/W	DPC	Double Pixel Clock Mode DPC instructs the LCD controller to drive out a 2X pixel clock out of the L_PCLK pin. This mode allows a glueless interface to an ADI 7171 NTSC encoder. All of the settings in the LCD controller are still specified in terms of the “original” pixel clock; this mode ONLY affects the pixel clock output pin. Note that if DPC is set, pixel clock divisor (PCD) must be greater than or equal to 2, otherwise the 2X pixel clock would be the same frequency as the LCD control clock, which is not valid. 0 = The L_PCLK pin is driven at the frequency specified by PCD. 1 = The L_PCLK pin is driven at double the frequency specified by PCD.

Table 7-43. LCCR3 Bit Definitions (Sheet 2 of 5)

Physical Address 0x4400_000C — LCCR3 — LCD Controller

Bit	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
	PDFOR	BPP3	reserved	DPC	BPP				OEP	PCP	HSP	VSP	API				ACB								PCD							
User Settings																																
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

Bits	Access	Name	Description
26:24	R/W	BPP	Bits per Pixel Used in conjunction with BPP3 to specify the pixel depth for each pixel stored in the memory for the base frame. Pixel depths of 2, 4, and 8 bpp require the internal palette RAM be loaded before pixels can be displayed on the screen. The following values are a concatenation of BPP3 and BPP: 0b0001 = 2-bits/pixel [4 entry, 8 or 16 byte palette buffer] 0b0010 = 4-bits/pixel [16 entry, 32 or 64 byte palette buffer] 0b0011 = 8-bits/pixel [256 entry, 512 or 1024 byte palette buffer] 0b0100 = 16-bits/pixel [no palette buffer] 0b0101 = 18-bits/pixel[no palette buffer] 0b0110 = 18-bits/pixel packed [no palette buffer] 0b0111 = 19-bits/pixel [no palette buffer] 0b1000 = 19-bits/pixel packed [no palette buffer] 0b1001 = 24-bits/pixel [no palette buffer] 0b1010 = 25-bits/pixel [no palette buffer] others = reserved NOTE: Configuring the LCD controller for 24bpp output (BPP3.BPP set to either 0b1001 or 0b1010) is invalid for panels without an internal frame buffer (LCCR0[LCDT] clear) and will result in indeterminate behavior. Refer to Section 7.5.1.2 for details on programming the DMA to load the palette RAM.
23	R/W	OEP	Output Enable Polarity The output enable polarity (OEP) bit selects the active and Inactive states of the output enable signal in active display mode. In this mode, the AC bias pin is used as an enable that signals the off-chip device when data is actively being driven out using the pixel clock. The pixel clock continuously toggles during operation of active mode (PAS = 1). When OEP = 0, the L_BIAS pin is active high and inactive low. When OEP = 1, the L_BIAS pin is active low and inactive high. In active display mode, data is driven onto the LCD data pins on the programmed edge of the L_PCLK pin when L_BIAS is in its active state. Note that OEP does not affect L_BIAS in passive display mode. 0 = L_BIAS pin is active high and inactive low in active display mode and parallel data input mode. 1 = L_BIAS pin is active low and inactive high in active display mode and parallel data input mode. In active display mode; FIFO data is driven out to the LCD's data pins on programmed pixel clock edge when AC bias pin is active. Note that OEP is ignored in passive display mode.

Table 7-43. LCCR3 Bit Definitions (Sheet 3 of 5)

Physical Address 0x4400_000C — LCCR3 — LCD Controller

Bit	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
	PDFOR		BPP3	reserved	DPC	BPP			OEP	PCP	HSP	VSP	API				ACB								PCD							
User Settings																																
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

Bits	Access	Name	Description
22	R/W	PCP	Pixel Clock Polarity The pixel clock polarity (PCP) bit selects which edge of the pixel clock data is sampled on the LCD data pins. When PCP = 0, data is sampled on the LCD data pins on the rising edge of the L_PCLK_WR pin. When PCP = 1, data is sampled on the LCD data pins on the falling edge of the L_PCLK_A0 pin. 0 = Data is sampled on the LCD data pins on the rising edge of L_PCLK_WR. 1 = Data is sampled on the LCD data pins on the falling edge of L_PCLK_WR.
21	R/W	HSP	Horizontal Sync Polarity The horizontal sync polarity (HSP) bit selects the active and inactive states of the horizontal sync signal in active display mode, and the line clock signal in passive display mode. When HSP = 0, the L_LCLK_A0 pin is active high and inactive low. When HSP = 1, the L_LCLK_A0 pin is active low and inactive high. Both in active and passive display modes, the L_FCLK_RD pin is forced to its inactive state whenever pixels are transmitted After the end of each line and a programmable number of pixel clock periods occur (controlled by ELW), the L_FCLK_RD pin is forced to its active state for a programmable number of line clocks (controlled by HSW), and is then again forced to its inactive state. 0 = L_LCLK_A0 pin is active high and inactive low. 1 = L_LCLK_A0 pin is active low and inactive high.
20	R/W	VSP	Vertical Sync Polarity The vertical sync polarity (VSP) bit selects the active and Inactive states of the vertical sync signal in active display mode (PAS = 1), and the frame clock signal in passive display mode. When VSP = 0, the L_FCLK pin is active high and inactive low. When VSP = 1, the L_FCLK_RD pin is active low and inactive high. In active display mode, the L_FCLK_RD pin is forced to its Inactive state while pixels are transmitted during the frame. After the end of the frame and a programmable number of line clocks periods occur (controlled by EFW), the L_FCLK_RD pin is forced to its active state for a programmable number of line clocks (controlled by VSW), and is then again forced to its Inactive state. In passive display mode, the L_FCLK_RD pin is forced to its Inactive state during the transmission of the second line of each frame through to the end of the frame. Frame clock is then forced to its active state on the rising edge of the first pixel clock of each frame. Frame clock remains active during the transmission of the entire first line of pixels in the frame and is then forced back to its Inactive state on the rising edge of the first pixel clock of the second line of the frame. 0 = L_FCLK_RD pin is active high and inactive low. 1 = L_FCLK_RD pin is active low and inactive high.

Table 7-43. LCCR3 Bit Definitions (Sheet 4 of 5)

Physical Address 0x4400_000C — **LCCR3** — **LCD Controller**

Bit	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
User Settings																																
	PDFOR	BPP3	reserved	DPC	BPP				OEP	PCP	HSP	VSP	API				ACB								PCD							
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

Bits	Access	Name	Description
19-16	R/W	API	AC Bias Pin Transitions per Interrupt In passive mode, the 4-bit AC Bias Pin Transitions per Interrupt (API) field specifies the number of L_BIAS pin transitions to count before setting the *AC* bias count status (ACS) bit in the LCD Controller Status register that signals an interrupt request. After the LCD controller is enabled, the value in API is loaded to a 4-bit Down counter and the counter decrements each time the AC bias pin is inverted. When the counter reaches zero, it stops and the AC bias count (ABC) bit is set in the Status register. Once ABC is set, the 4-bit Down counter is reloaded with the value in API, and is disabled until ABC is cleared. When ABC is cleared by the CPU, the Down counter is enabled and again decrements each time the AC bias pin is inverted. The number of AC bias pin transitions between each interrupt request ranges from 0 to 15. Note that programming API = 0x0 disables the AC bias pin transitions per interrupt function. In active mode, L_BIAS is used as an output-enable signal. However, signaling of the API interrupt can still occur. The ACB bit field counts line clock pulses in active mode. When the programmed number of line clock pulses occurs, an internal signal is transitioned that decrements the 4-bit counter used by the API interrupt logic. Once this internal signal transitions the programmed number of times as specified by API, an interrupt is generated. Program API to zero if the API interrupt function is not required in active mode (PAS = 1). Value (from 0 to 15) specifies the number of AC bias pin transitions to count before setting the line count status (ABC) bit, signaling an interrupt request. Counter frozen when ABC is set and is restarted when ABC is cleared by software. This function is disabled when API = 4'h0. NOTE: This is ignored for LCD panels with internal frame buffers.

Table 7-43. LCCR3 Bit Definitions (Sheet 5 of 5)

Physical Address 0x4400_000C — LCCR3 — LCD Controller

Bit	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
User Settings																																
	PDFOR		BPP3	reserved	DPC	BPP			OEP	PCP	HSP	VSP	API				ACB								PCD							
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

Bits	Access	Name	Description
15-8	R/W	ACB	AC Bias Pin Frequency The 8-bit AC Bias Frequency (ACB) field specifies the number of line clock periods to count between each toggle of the AC bias pin (L_BIAS). In passive mode, after the LCD controller is enabled, the value in ACB is loaded to an 8-bit Down counter and the counter begins to decrement using the line clock. When the counter reaches zero, it stops, the state of L_BIAS is reversed, and the whole procedure restarts. The number of line clocks between each AC bias pin transition ranges from 1 to 256. Program ACB with the preferred number of line clocks minus one. This pin is used by the LCD display to periodically reverse the polarity of the power supplied to the screen to eliminate D.C. offset. If the LCD display being controlled has its own internal means of switching its power supply, set ACB to its maximum value to reduce power consumption (0xFF). Note that the ACB bit field has no effect on L_BIAS in active mode. Because the pixel clock toggles continuously in active mode, the AC bias pin is used as an output-enable signal. It is asserted automatically by the LCD controller in active mode whenever pixel data is driven out to the data pins to signal the display when it may latch pixels using the pixel clock. Value (from 0 to 255) specifies the number of line clocks to count before toggling the AC bias pin in passive mode (PAS = 0). This pin periodically inverts the polarity of the power supply to prevent DC charge buildup within the display. If the passive display that is being controlled does not need to use L_BIAS, program ACB to its maximum value (0xFF) to conserve power. Note that ACB is ignored in active mode (PAS = 1). Number of line clocks/toggle of the L_BIAS pin = (ACB+1)
7-0	R/W	PCD	Pixel Clock Divisor Value (from 0 to 255). Used along with LCCR4[PCDDIV] for passive and active displays to specify the frequency of the pixel clock based on the LCD clock (LCLK) frequency. Pixel clock frequency can range from LCLK/2 to LCLK/512, where LCLK is the programmed frequency of the LCD/Memory controller. LCLK can vary from 13.0 MHz to 104.00 MHz Pixel Clock Frequency = LCLK/2(PCD+1) if LCCR4{PCDDIV] = 0 or Pixel Clock Frequency = LCLK/(PCD+1) if LCCR4{PCDDIV] = 1 Note that PCD must be programmed with a value of 1 or greater if double pixel clock mode is enabled or if LCCR4{PCDDIV] is set. For LCD Panels with an internal frame buffer, this bit field specifies the command inhibit time between any two consecutive reads or writes to the LCD panel (write/write or read/read or write/read or read/write). Command Inhibit TIme = (PCD+1)*LCD_CLK_PERIOD.

7.5.6 LCD Controller Control Register 4 (LCCR4)

LCD Controller Control register 4 (LCCR4) (Table 7-44) contains bit fields to program the constants used to calculate the pixel value for half-transparency, output-drive format and the PCD value for 13M mode operation.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-44. LCCR4 Bit Definitions (Sheet 1 of 5)

Physical Address 0x4400_0010 — **LCCR4** — **LCD Controller**

Bit	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
User Settings																																
	PCDDIV	reserved					13M_PCD_EN	13M_PCD_VAL								PAL_FOR	reserved							K3			K2			K1		
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	PCDDIV	PCD Divisor Selection This selects whether the LCD clock is divided by two in the PCD equation. 0 = Pixel Clock = LCLK/2*(PCD+1) 1 = Pixel Clock = LCLK/(PCD+1) NOTE: if PCDDIV is set, the minimum value of LCCR3[PCD} is one. This bit has is ignored when the LCD controller is configured to control a smart panel.
30:26	—	—	reserved
25	R/W	13M_PCD_EN	13M mode Pixel Clock Divisor Enable This enables the13M PCD value to be used when the processor is in 13M mode. 0 = Use PCD value in LCCR3[PCD] for *all modes* 1 = Use PCD value in LCCR4[13M_PCD_VAL] for *13M mode only* Software can set this bit during LCD configuration. The processor will then use the value in LCCR4[13M_PCD_VAL] for 13M mode only. (**NOTE**: This bit field is available in the C0 stepping and all the subsequent steppings)

Table 7-44. LCCR4 Bit Definitions (Sheet 2 of 5)

Physical Address 0x4400_0010	LCCR4	LCD Controller

User Settings																															
Bit	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
	PCDDIV	reserved					13M_PCD_EN	13M_PCD_VAL								PAL_FOR	reserved							K3			K2			K1		
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
24:17	R/W	13M_PCD_VAL	13M mode Pixel Clock Divisor Value Program the LCCR4[13M_PCD_VAL] with the required PCD value that will be used while the processor is in 13M mode. Value (from 0 to 255). Used along with LCCR4[PCDDIV] for passive and active displays to specify the frequency of the pixel clock while in 13M mode, based on the LCD clock (LCLK) frequency. Pixel clock frequency can range from LCLK/2 to LCLK/512, where LCLK is the programmed frequency of the LCD/Memory controller. LCLK can only be configured to 13MHz or 26MHz using CCCR[LCD_26] during 13M mode operation. Pixel Clock Frequency = LCLK/2(PCD+1) if LCCR4{PCDDIV] = 0 or Pixel Clock Frequency = LCLK/(PCD+1) if LCCR4{PCDDIV] = 1 Note that PCD must be programmed with a value of 1 or greater if double pixel clock mode is enabled or if LCCR4{PCDDIV] is set. For LCD Panels with an internal frame buffer, this bit field specifies the command inhibit time between any two consecutive reads or writes to the LCD panel (write/write or read/read or write/read or read/write). Command Inhibit Time = (PCD+1)*LCD_CLK_PERIOD. **NOTE:** The value programmed into this bit field is only used if LCCR4[13M_PCD_EN] is set. (**NOTE:** This bit field is available in the C0 stepping and all the subsequent steppings)
16:15	R/W	PAL_FOR	Palette Data Format The palette data format (PAL_FOR) specifies the data format for the palette data. When the overlays are disabled and the bits per pixel is less then 18-bits, then PAL_FOR = 0b00 is used. When the overlays are enabled, the 25-bit palette data format with transparency bit is used. See Figure 7-17. The PAL_FOR is set depending on the data formats. For 16-bits per pixel data format set PAL_FOR = 0b01, for 18 bits per pixel data format set PAL_FOR = 0b10 and for 24 bits per pixel data format set PAL_FOR = 0b11. This specifies the palette data format for the palette RAM. 0b00 = 16 bits without Transparency bit. 0b01 = 25 bit format with Transparency for 16 bits per pixel data formats 0b10 = 25 bit format with Transparency for 18 bits per pixel data formats 0b11 = 25 bit format with Transparency for 24 bits per pixel data formats

Table 7-44. LCCR4 Bit Definitions (Sheet 3 of 5)

Physical Address 0x4400_0010 — LCCR4 — LCD Controller

Bit	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
User Settings	PCDDIV	reserved					13M_PCD_EN	13M_PCD_VAL									PAL_FOR	reserved						K3			K2			K1		
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
14:9	—	—	reserved

Table 7-44. LCCR4 Bit Definitions (Sheet 4 of 5)

Physical Address 0x4400_0010 — **LCCR4** — **LCD Controller**

Bit	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
User Settings																																
	PCDDIV	reserved					13M_PCD_EN	13M_PCD_VAL								PAL_FOR	reserved							K3			K2			K1		
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
8:6	R/W	K3	Multiplication Constant for Green for Half Transparency Multiplication constant (K3) is value between 1/8 and 1 in increments of 1/8. Users have to program to the appropriate value to get preferred results for half transparency. K3 calculates the results for the green value. See Section 7.4.5.1 for more detailed equations. 0b000 = 1/8 0b001 = 2/8 0b010 = 3/8 0b011 = 4/8 0b100 = 5/8 0b101 = 6/8 0b110 = 7/8 0b111 = 1
5:3	R/W	K2	Multiplication Constant for Blue for Half Transparency Multiplication constant (K2) is value between 1/8 and 1 in increments of 1/8. Users have to program to the appropriate value to get preferred results for half transparency. K2 calculates the results for the blue value. See Section 7.4.5.1 for more detailed equations. 0b000 = 1/8 0b001 = 2/8 0b010 = 3/8 0b011 = 4/8 0b100 = 5/8 0b101 = 6/8 0b110 = 7/8 0b111 = 1

Table 7-44. LCCR4 Bit Definitions (Sheet 5 of 5)

Physical Address 0x4400_0010																LCCR4															LCD Controller												
User Settings																																									
Bit	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									
	PCDDIV	reserved					13M_PCD_EN	13M_PCD_VAL								PAL_FOR	reserved							K3			K2			K1												
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0									

Bits	Access	Name	Description
2:0	R/W	K1	Multiplication Constant for Red for Half Transparency Multiplication constant (K1) is value between 1/8 and 1 in increments of 1/8. Users have to program to the appropriate value to get preferred results for half transparency. K1 calculates the results for the red value. See Section 7.4.5.1 for more detailed equations. 0b000 = 1/8 0b001 = 2/8 0b010 = 3/8 0b011 = 4/8 0b100 = 5/8 0b101 = 6/8 0b110 = 7/8 0b111 = 1

7.5.7 LCD Controller Control Register 5 (LCCR5)

LCD Controller Control register 5 (LCCR5) (Table 7-45) contains bit fields that mask the various interrupt bits for channel 1 through channel 6.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-45. LCCR5 Bit Definitions (Sheet 1 of 13)

Physical Address 0x4400_0014 — **LCCR5** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
31:30	—	—	reserved
29	R/W	IUM6	Input FIFO Underrun Mask for Command Data The Input FIFO Underrun Mask bit (IUM6) masks an interrupt request that is asserted whenever an input FIFO underrun error occurs. When IUM6 is cleared, underrun interrupts are enabled, and whenever LCSR1[IU6] is set, an interrupt request is made to the interrupt controller. When IUM6 is set, underrun interrupts are masked; LCSR1[IU6] is ignored by the interrupt controller. Note that setting IUM6 does not affect the current state of the status bit or the ability of the LCD controller to set and clear it; it only blocks the generation of the interrupt request. 0 = FIFO underrun errors generate an interrupt (state of LCSR1[IU6] sent to the interrupt controller). 1 = FIFO underrun errors do not generate an interrupt (LCSR1[IU6] ignored).
28	R/W	IUM5	Input FIFO Underrun Mask for Cursor The Input FIFO Underrun Mask bit (IUM5) masks interrupt requests that are asserted whenever an input FIFO underrun error occurs. When IUM5 is cleared, underrun interrupts are enabled, and whenever LCSR1[IU5] is set, an interrupt request is made to the interrupt controller. When IUM5 is set, underrun interrupts are masked; LCSR1[IU5] is ignored by the interrupt controller. Note that setting IUM5 does not affect the current state of the status bit or the ability of the LCD controller to set and clear it; it only blocks the generation of the interrupt request. 0 = FIFO underrun errors generate an interrupt (state of LCSR1[IU5] sent to the interrupt controller. 1 = FIFO underrun errors do not generate an interrupt (LCSR1[IU5] = ignored).

Table 7-45. LCCR5 Bit Definitions (Sheet 2 of 13)

Physical Address 0x4400_0014 — **LCCR5** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
27	R/W	IUM4	Input FIFO Underrun Mask Overlay 2 The Input FIFO Underrun Mask bit (IUM4) masks interrupt requests that are asserted whenever an input FIFO underrun error occurs. When IUM4 is cleared, underrun interrupts are enabled, and whenever LCSR1[IU4] is set, an interrupt request is made to the interrupt controller. When IUM4 is set, underrun interrupts are masked; LCSR1[IU4] is ignored by the interrupt controller. Note that setting IUM4 does not affect the current state of the status bit or the ability of the LCD controller to set and clear it; it only blocks the generation of the interrupt request. 0 = FIFO underrun errors generate an interrupt (state of LCSR1[IU4] sent to the interrupt controller). 1 = FIFO underrun errors do not generate an interrupt (LCSR1[IU4] ignored).
26	R/W	IUM3	Input FIFO Underrun Mask for Overlay 2 The Input FIFO Underrun Mask bit (IUM3) masks interrupt requests that are asserted whenever an input FIFO underrun error occurs. When IUM3 is cleared, underrun interrupts are enabled, and whenever LCSR1[IU3] is set, an interrupt request is made to the interrupt controller. When IUM3 is set, underrun interrupts are masked;LCSR1[IU3] is ignored by the interrupt controller. Note that setting IUM3 does not affect the current state of the status bit or the ability of the LCD controller to set and clear it; it only blocks the generation of the interrupt request. 0 = FIFO underrun errors generate an interrupt (state of LCSR1[IU3] sent to the interrupt controller). 1 = FIFO underrun errors do not generate an interrupt (LCSR1[IU3] ignored).

Table 7-45. LCCR5 Bit Definitions (Sheet 3 of 13)

Physical Address 0x4400_0014 — **LCCR5** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
25	R/W	IUM2	Input FIFO Underrun Mask for Overlay 2 The Input FIFO Underrun Mask bit (IUM2) masks interrupt requests that are asserted whenever an input FIFO underrun error occurs. When IUM2 is cleared, underrun interrupts are enabled, and whenever LCSR1[IU2] is set, an interrupt request is made to the interrupt controller. When IUM2 is set, underrun interrupts are masked; LCSR1[IU2] is ignored by the interrupt controller. Note that setting IUM2 does not affect the current state of the status bit or the ability of the LCD controller to set and clear it; it only blocks the generation of the interrupt request. 0 = FIFO underrun errors generate an interrupt (state of LCSR1[IU2] sent to the interrupt controller). 1 = FIFO underrun errors do not generate an interrupt (LCSR1[IU2] status bits ignored).
24	R/W	IUM1	Input FIFO Underrun Mask for Overlay 1 (When Enabled) The input FIFO Underrun Mask bit (IUM1) masks interrupt requests that are asserted whenever an input FIFO underrun error (Overlay 1 enabled) occurs. When IUM1 is cleared, underrun interrupts are enabled, and whenever LCSR0[IU1] is set, an interrupt request is made to the interrupt controller. When IUM1 is set, underrun interrupts are masked; LCSR0[IU1] is ignored by the interrupt controller. Note that setting IUM6 does not affect the current state of the status bits or the ability of the LCD controller to set and clear it; it only blocks the generation of the interrupt request. 0 = FIFO underrun errors generate an interrupt (state of LCSR0[IU1] status sent to the interrupt controller). 1 = FIFO underrun errors do not generate an interrupt (LCSR0[IU1] status bits ignored).
23:22	—	—	reserved

Table 7-45. LCCR5 Bit Definitions (Sheet 4 of 13)

Physical Address 0x4400_0014 — **LCCR5** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
21	R/W	BSM6	Branch Mask for Command Register (DMA Channel 6) The Branch Status Mask (BSM6) bit masks the interrupt requests that are asserted after branching to a new frame. When BSM6 is cleared, branch status interrupts are enabled, and whenever the branch status bit (LCSR1[BS6]) is set, an interrupt request is made to the interrupt controller. When BSM6 is set, the branch status interrupt is masked; the state of the branch status bit is ignored by the interrupt controller. Note that setting BSM6 does not affect the current state of LCSR1[BS6] or the ability of the LCD controller to set and clear LCSR1[BS6]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt after branching to a new frame (state of LCSR1[BS6] sent to the interrupt controller). 1 = BS condition does not generate an interrupt (LCSR1[BS6] ignored).
20	R/W	BSM5	Branch Mask for Cursor (DMA Channel 5) The Branch Status Mask (BSM5) bit masks the interrupt requests that are asserted after branching to a new frame. When BSM5 is cleared, branch status interrupts are enabled, and whenever the branch status bit (LCSR1[BS5]) is set, an interrupt request is made to the interrupt controller. When BSM5 is set, the branch status interrupt is masked; the state of the branch status bit is ignored by the interrupt controller. Note that setting BSM5 does not affect the current state of LCSR1[BS5] or the ability of the LCD controller to set and clear LCSR1[BS5]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt after branching to a new frame (state of LCSR1[BS5] sent to the interrupt controller). 1 = BS condition does not generate an interrupt (LCSR1[BS5] ignored).

Table 7-45. LCCR5 Bit Definitions (Sheet 5 of 13)

Physical Address 0x4400_0014 — **LCCR5** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
19	R/W	BSM4	Branch Mask for Overlay 2 (DMA Channel 4) The Branch Status Mask (BSM4) bit masks the interrupt requests that are asserted after branching to a new frame. When BSM4 is cleared, branch status interrupts are enabled, and whenever the branch status bit (LCSR1[BS4]) is set, an interrupt request is made to the interrupt controller. When BSM4 is set, the branch status interrupt is masked; the state of the branch status bit is ignored by the interrupt controller. Note that setting BSM4 does not affect the current state of LCSR1[BS4] or the ability of the LCD controller to set and clear LCSR1[BS4]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt after branching to a new frame (state of LCSR1[BS4] sent to the interrupt controller). 1 = BS condition does not generate an interrupt (LCSR1[BS4] ignored).
18	R/W	BSM3	Branch Mask for Overlay 2 (DMA Channel 3) The Branch Status Mask (BSM3) bit masks the interrupt requests that are asserted after branching to a new frame. When BSM3 is cleared, branch status interrupts are enabled, and whenever the branch status bit (LCSR1[BS3]) is set, an interrupt request is made to the interrupt controller. When BSM3 is set, the branch status interrupt is masked; the state of the branch status bit is ignored by the interrupt controller. Note that setting BSM3 does not affect the current state of LCSR1[BS3] or the ability of the LCD controller to set and clear LCSR1[BS3]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt after branching to a new frame (state of LCSR1[BS3] sent to the interrupt controller). 1 = BS condition does not generate an interrupt (LCSR1[BS3] ignored).

Table 7-45. LCCR5 Bit Definitions (Sheet 6 of 13)

Physical Address 0x4400_0014 — **LCCR5** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
17	R/W	BSM2	Branch Mask for Overlay 2 (DMA Channel 2) The Branch Status Mask (BSM2) bit masks the interrupt requests that are asserted after branching to a new frame. When BSM2 is cleared, branch status interrupts are enabled, and whenever the branch status bit (LCSR1[BS2]) is set, an interrupt request is made to the interrupt controller. When BSM2 is set, the branch status interrupt is masked; the state of the branch status bit is ignored by the interrupt controller. Note that setting BSM2 does not affect the current state of LCSR1[BS2] or the ability of the LCD controller to set and clear LCSR1[BS2]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt after branching to a new frame (state of LCSR1[BS2] sent to the interrupt controller). 1 = BS condition does not generate an interrupt (LCSR1[BS2] ignored).
16	R/W	BSM1	Branch Mask for Overlay 1 (DMA Channel 1) The Branch Status Mask (BSM1) bit masks the interrupt requests that are asserted after branching to a new frame. When BSM1 is cleared, branch status interrupts are enabled, and whenever the branch status bit (LCSR1[BS1]) is set, an interrupt request is made to the interrupt controller. When BSM1 is set, the branch status interrupt is masked; the state of the branch status bit is ignored by the interrupt controller. Note that setting BSM1 does not affect the current state of LCSR1[BS1] or the ability of the LCD controller to set and clear LCSR1[BS1]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt after branching to a new frame (state of LCSR1[BS1] sent to the interrupt controller). 1 = BS condition does not generate an interrupt (LCSR1[BS1] ignored).
15:14	—	—	reserved

Table 7-45. LCCR5 Bit Definitions (Sheet 7 of 13)

Physical Address 0x4400_0014 — **LCCR5** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
13	R/W	EOFM6	End Of Frame Mask for Command Data (DMA Channel 6) The End Of Frame Mask (EOFM6) bit masks interrupt requests that are asserted at the end of each frame (when the DMA length of transfer counter decrements to zero). When EOFM6 is cleared, the interrupt is enabled, and whenever the end of the frame status bit (LCSR1[EOF6]) is cleared an interrupt request is made to the interrupt controller. When EOFM6 is set, the interrupt is masked and LCSR1[EOF6] is ignored by the interrupt controller. Note that setting EOFM6 does not affect the current state of LCSR1[EOF6] or the ability of the LCD controller to set and clear LCSR1[EOF6]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt at the end of a frame (state of LCSR1[EOF6] sent to the interrupt controller). 1 = EOF condition does not generate an interrupt (LCSR1[EOF6] ignored).
12	R/W	EOFM5	End of Frame Mask for Cursor (DMA Channel 5) The End Of Frame Mask (EOFM5) bit masks interrupt requests that are asserted at the end of each frame (when the DMA length of transfer counter decrements to zero). When EOFM5 is cleared, the interrupt is enabled, and whenever the end of the frame status bit (LCSR1[EOF5]) is cleared an interrupt request is made to the interrupt controller. When EOFM5 is set, the interrupt is masked and LCSR1[EOF5] is ignored by the interrupt controller. Note that setting EOFM5 does not affect the current state of LCSR1[EOF5] or the ability of the LCD controller to set and clear LCSR1[EOF5]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt at the end of a frame (state of LCSR1[EOF5] sent to the interrupt controller). 1 = EOF condition does not generate an interrupt (LCSR1[EOF5] ignored).

Table 7-45. LCCR5 Bit Definitions (Sheet 8 of 13)

Physical Address 0x4400_0014 — **LCCR5** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
11	R/W	EOFM4	End Of Frame Mask for Overlay 2 (DMA Channel 4) The End Of Frame Mask (EOFM4) bit masks interrupt requests that are asserted at the end of each frame (when the DMA length of transfer counter decrements to zero). When EOFM4 is cleared, the interrupt is enabled, and whenever LCSR1[EOF4] is cleared, an interrupt request is made to the interrupt controller. When EOFM4 is set, the interrupt is masked and LCSR1[EOF4] is ignored by the interrupt controller. Note that setting EOFM4 does not affect the current state of LCSR1[EOF4] or the ability of the LCD controller to set and clear LCSR1[EOF4]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt at the end of a frame (state of LCSR1[EOF4] sent to the interrupt controller). 1 = EOF condition does not generate an interrupt (LCSR1[EOF4] ignored).
10	R/W	EOFM3	End Of Frame Mask for Overlay 2 (DMA Channel 3) The End Of Frame Mask (EOFM3) bit masks interrupt requests that are asserted at the end of each frame (when the DMA length of transfer counter decrements to zero). When EOFM3 is cleared, the interrupt is enabled, and whenever LCSR1[EOF3] is cleared, an interrupt request is made to the interrupt controller. When EOFM3 is set, the interrupt is masked and LCSR1[EOF3] is ignored by the interrupt controller. Note that setting EOFM3 does not affect the current state of LCSR1[EOF3] or the ability of the LCD controller to set and clear LCSR1[EOF3]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt at the end of a frame (state of LCSR1[EOF3] sent to the interrupt controller). 1 = EOF condition does not generate an interrupt (LCSR1[EOF3] ignored).

Table 7-45. LCCR5 Bit Definitions (Sheet 9 of 13)

Physical Address 0x4400_0014 — **LCCR5** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
9	R/W	EOFM2	End Of Frame Mask for Overlay 2 (DMA Channel 2) The End Of Frame Mask bit masks interrupt requests that are asserted at the end of each frame (when the DMA length of transfer counter decrements to zero). When EOFM2 is cleared, the interrupt is enabled, and whenever LCSR1[EOF2] is cleared, an interrupt request is made to the interrupt controller. When EOFM2 is set, the interrupt is masked and the state of LCSR1[EOF2] is ignored by the interrupt controller. Note that setting EOFM2 does not affect the current state of LCSR1[EOF2] or the ability of the LCD controller to set and clear LCSR1[EOF2]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt at the end of a frame (state of LCSR1[EOF2] sent to the interrupt controller). 1 = EOF condition does not generate an interrupt (LCSR1[EOF2] ignored).
8	R/W	EOFM1	End Of Frame Mask for Overlay 1 (DMA Channel 1) The End Of Frame Mask bit masks interrupt requests that are asserted at the end of each frame (when the DMA length of transfer counter decrements to zero). When EOFM1 is cleared, the interrupt is enabled, and whenever LCSR1[EOF1] is cleared, an interrupt request is made to the interrupt controller. When EOFM1 is set, the interrupt is masked and the state of LCSR1[EOF1] is ignored by the interrupt controller. Note that setting EOFM1 does not affect the current state of LCSR1[EOF1] or the ability of the LCD controller to set and clear LCSR1[EOF1]; it only blocks the generation of the interrupt request. 0 = Generates an interrupt at the end of a frame (state of LCSR1[EOF1] sent to the interrupt controller). 1 = EOF condition does not generate an interrupt (LCSR1[EOF1] ignored).
7:6	—	—	reserved

Table 7-45. LCCR5 Bit Definitions (Sheet 10 of 13)

Physical Address 0x4400_0014 — **LCCR5** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
5	R/W	SOFM6	Start Of Frame Mask for Command Data (DMA Channel 6) The start of frame interrupt mask (SOFM6) bit masks interrupt requests that are asserted at the beginning of each frame when the LCD Frame Descriptor has been loaded into the internal DMA registers. When SOFM6 is cleared, the interrupt is enabled, and whenever LCSR1[SOF6] is set, an interrupt request is made to the interrupt controller. When SOFM6 is set, the interrupt is masked and the state of LCSR1[SOF6] is ignored by the interrupt controller. Note that setting SOFM6 does not affect the current state of LCSR1[SOF6] or the ability of the LCD controller to set and clear LCSR1[SOF6]; it only blocks the generation of interrupt requests. 0 = Starting a new frame for command data (after loading frame Descriptor) generates an interrupt (state of LCSR1[SOF6] sent to the interrupt controller). 1 = SOF condition does not generate an interrupt (LCSR1[SOF6] ignored).
4	R/W	SOFM5	Start Of Frame Mask for Cursor (DMA Channel 5) The start of frame interrupt mask (SOFM5) bit masks interrupt requests that are asserted at the beginning of each frame when the LCD Frame Descriptor has been loaded into the internal DMA registers. When SOFM5 is cleared, the interrupt is enabled, and whenever LCSR1[SOF5] is set, an interrupt request is made to the interrupt controller. When SOFM5 is set, the interrupt is masked and the state of LCSR1[SOF5] is ignored by the interrupt controller. Note that setting SOFM5 does not affect the current state of LCSR1[SOF5] or the ability of the LCD controller to set and clear LCSR1[SOF5]; it only blocks the generation of interrupt requests. 0 = Starting a new frame for cursor (after loading frame Descriptor) generates an interrupt (state of LCSR1[SOF5] sent to the interrupt controller). 1 = SOF condition does not generate an interrupt (LCSR1[SOF5] ignored).

Table 7-45. LCCR5 Bit Definitions (Sheet 11 of 13)

Physical Address 0x4400_0014 — **LCCR5** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
3	R/W	SOFM4	Start Of Frame Mask for Overlay 2 (DMA Channel 4) The start of frame interrupt mask (SOFM4) bit masks interrupt requests that are asserted at the beginning of each frame when the LCD Frame Descriptor has been loaded into the internal DMA registers. When SOFM4 is cleared, the interrupt is enabled, and whenever LCSR1[SOF4] is set, an interrupt request is made to the interrupt controller. When SOFM4 is set, the interrupt is masked and the state of LCSR1[SOF4] is ignored by the interrupt controller. Note that setting SOFM4 does not affect the current state of LCSR1[SOF4] or the ability of the LCD controller to set and clear LCSR1[SOF4]; it only blocks the generation of interrupt requests. 0 = Starting a new frame for Overlay 2 (after loading frame Descriptor) generates an interrupt (state of LCSR1[SOF4] sent to the interrupt controller). 1 = SOF condition does not generate an interrupt (LCSR1[SOF4] ignored).

Table 7-45. LCCR5 Bit Definitions (Sheet 12 of 13)

Physical Address 0x4400_0014 — **LCCR5** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
2	R/W	SOFM3	Start Of Frame Mask for Overlay 2 (DMA Channel 3) This bit masks interrupt requests that are asserted at the beginning of each frame when the LCD Frame Descriptor has been loaded into the internal DMA registers. When SOFM3 is cleared, the interrupt is enabled, and whenever LCSR1[SOF3] is set, an interrupt request is made to the interrupt controller. When SOFM3 is set, the interrupt is masked and the state of LCSR1[SOF3] is ignored by the interrupt controller. Note that setting SOFM3 does not affect the current state of LCSR1[SOF3] or the ability of the LCD controller to set and clear LCSR1[SOF3]; it only blocks the generation of interrupt requests. 0 = Starting a new frame for Overlay 2 (after loading frame Descriptor) generates an interrupt (state of LCSR1[SOF3] sent to the interrupt controller). 1 = SOF condition does not generate an interrupt (LCSR1[SOF3] ignored).
1	R/W	SOFM2	Start Of Frame Mask for Overlay 2 (DMA Channel 2) This bit masks interrupt requests that are asserted at the beginning of each frame when the LCD Frame Descriptor has been loaded into the internal DMA registers. When SOFM2 is cleared, the interrupt is enabled, and whenever LCSR1[SOF2] is set, an interrupt request is made to the interrupt controller. When SOFM2 is set, the interrupt is masked and the state of LCSR1[SOF2] is ignored by the interrupt controller. Note that setting SOFM2 does not affect the current state of LCSR1[SOF2] or the ability of the LCD controller to set and clear LCSR1[SOF2]; it only blocks the generation of interrupt requests. 0 = Starting a new frame for Overlay 2 (after loading frame Descriptor) generates an interrupt (state of LCSR1[SOF2] sent to the interrupt controller). 1 = SOF condition does not generate an interrupt (LCSR1[SOF2] ignored).

Table 7-45. LCCR5 Bit Definitions (Sheet 13 of 13)

Physical Address 0x4400_0014	LCCR5	LCD Controller

User Settings																															
Bit	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		IUM6	IUM5	IUM4	IUM3	IUM2	IUM1	reserved		BSM6	BSM5	BSM4	BSM3	BSM2	BSM1	reserved		EOFM6	EOFM5	EOFM4	EOFM3	EOFM2	EOFM1	reserved		SOFM6	SOFM5	SOFM4	SOFM3	SOFM2	SOFM1
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

Bits	Access	Name	Description
0	R/W	SOFM1	Start Of Frame Mask for Overlay 1 (DMA Channel 1) This bit masks interrupt requests that are asserted at the beginning of each frame when the LCD Frame Descriptor has been loaded into the internal DMA registers. When SOFM1 is cleared, the interrupt is enabled, and whenever LCSR1[SOF1] is set, an interrupt request is made to the interrupt controller. When SOFM1 is set, the interrupt is masked and the state of LCSR1[SOF1] is ignored by the interrupt controller. Note that setting SOFM1 does not affect the current state of LCSR1[SOF1] or the ability of the LCD controller to set and clear LCSR1[SOF1]; it only blocks the generation of interrupt requests. 0 = Starting a new frame for Overlay 1 (after loading frame Descriptor) generates an interrupt (state of LCSR1[SOF1] sent to the interrupt controller). 1 = SOF condition does not generate an interrupt (LCSR1[SOF1] ignored).

7.5.8 Overlay 1 Control Register 1 (OVL1C1)

Overlay 1 Control register 1 (OVL1C1) (Table 7-46) contains bit fields that enable and set the size and pixel format for Overlay 1 window.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-46. OVL1C1 Bit Definitions

Physical Address 0x4400_0050																OVL1C1																LCD Controller							
User Settings																																								
Bit	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
	O1EN	reserved							BPP1				LPO1										PPL1									
Reset	0	?	?	?	?	?	?	?	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	O1EN	Enable bit for Overlay 1 Enables or disables the display of the Overlay 1. When it is set, the display of the Overlay 1 frame is enabled. When is clear, the display of Overlay 1 frame is disabled. 0 = Overlay 1 is disabled. 1 = Overlay 1 is enabled.
30:24	—	—	reserved
23:20	R/W	BPP1	Bits per Pixel for Overlay 1 Specifies the pixel depth of each pixel stored in the memory. Pixel depths of 4 and 8 bpp require the internal palette RAM be loaded before pixels can be displayed on the screen. 0b0000 = 24 Bit mode without transparency bit. 0b0010 = 4-bits/pixel [16 entry, 32 or 64 byte palette buffer] 0b0011 = 8-bits/pixel [256 entry, 512 or 1024 byte palette buffer] 0b0100 = 16-bits/pixel [no palette buffer] 0b0101 = 18 bits/pixel unpacked [no palette buffer] 0b0110 = 18-bits/pixel packed [no palette buffer] 0b0111 = 19-bits/pixel unpacked [no palette buffer] 0b1000 = 19 bits/pixel packed [no palette buffer] 0b1001 = 24-bits/pixel [no palette buffer] 0b1010 = 25-bits/pixel [no palette buffer] others = reserved When BPP1 = 0x0, Overlay 1 is configured for 24 bpp mode with transparency enabled and with 8 bits each of red, green and blue. In this mode the transparency bit T does not exist, but transparency is enabled.
19:10	R/W	LPO1	Number of Lines for Overlay 1 Specifies the number line or rows present in the Overlay 1 frame. This also represents the size of the overlay in the vertical direction. LPO1 is a 10-bit value that represents between 1 and 600 lines per screen. Specifies the size of the Overlay 1 window in the vertical direction. This is programmed with the value from 0 to 599. Actual number of line = LPO1 +1
9:0	R/W	PPL1	Pixels per Line for Overlay 1 Frame Specifies the number of pixels in each line or row for the Overlay 1 frame. PPL1 is a 10-bit value that represents between 1 and 800 pixels per line. See Section 7.4.13 for the restrictions on pixels per line. Value (from 0 to 799) specifies the number of pixels contained within each line for the Overlay 1 frame. Actual pixel per line = (PPL1 +1)

7.5.9 Overlay 1 Control Register 2 (OVL1C2)

Overlay 1 Control register 2 (OVL1C2) (Table 7-47) contains bit fields that are programmed to set the position of Overlay 1 on the panel display.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-47. OVL1C2 Bit Definitions

Physical Address 0x4400_0060												OVL1C2												LCD Controller									
User Settings																																	
Bit	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												O1YPOS										O1XPOS									
Reset	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:20	—	—	reserved
19:10	R/W	O1YPOS	Vertical Position of the Upper Left-Most Pixel of Overlay 1 Window The Vertical Position of the Overlay 1 (O1YPOS) bit field specifies the vertical position of the Overlay 1 window. The vertical position of the window is the Y-coordinate of the upper left-most pixel, with respect to the origin of the screen. The origin (0, 0) of the screen is the top left most corner. O1YPOS is 10-bit value that represents a value between 1 to 600. Value (from 0 to 599). Actual vertical position = (O1YPOS+1).
9:0	R/W	O1XPOS	Horizontal Position of the Upper Left-Most Pixel of Overlay 1 Window The Horizontal Position of the Overlay 1 (O1XPOS) bit field specifies the horizontal position of the Overlay 1 window. The horizontal position of the window is the X-coordinate of the upper left-most pixel, with respect to the origin of the screen. The origin (0, 0) of the screen is top left most corner. O1XPOS is 10-bit value that represents a value between 1 to 800. Value (from 0 to 799). Actual horizontal position = (O1XPOS+1).

7.5.10 Overlay 2 Control Register 1 (OVL2C1)

Overlay 2 Control register 1 (OVL2C1) (Table 7-48) contains bit fields that enable and set the size and pixel format for Overlay 2 window.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-48. OVL2C1 Bit Definitions (Sheet 1 of 2)

Physical Address 0x4400_0070	OVL2C1	LCD Controller

	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
User Settings																																
Bit	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
	O2EN	reserved							BPP2				LPO2										PPL2									
Reset	0	?	?	?	?	?	?	?	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	O2EN	Overlay 2 Enable The Overlay 2 enable (O2EN) bit enables or disables the display of the Overlay. When set, the display of the Overlay 2 is enabled. When is clear, the display is disabled. 0 = Overlay 2 is disabled. 1 = Overlay 2 is enabled.
30:24	—	—	reserved
23:20	R/W	BPP2	Bits per Pixel for Overlay 2 BPP2 specifies the pixel depth of each pixel stored in the internal or the external memory for the Overlay 2 frame. Pixel depths of 4 and 8 bpp require the internal palette RAM be loaded before pixels can be displayed on the screen. 0b0010 = 4-bits/pixel [16 entry, 32 or 64 byte palette buffer] 0b0011 = 8-bits/pixel [256 entry, 512 or 1024 byte palette buffer] 0b0100 = 16-bits/pixel [no palette buffer] 0b0101 = 18 bits/pixel, unpacked [no palette buffer] 0b0110 = 18-bits/pixel, packed [no palette buffer] 0b0111 = 19-bits/pixel, unpacked [no palette buffer] 0b1000 = 19 bits/pixel, packed [no palette buffer] 0b1001 = 24-bits/pixel [no palette buffer] 0b1010 = 25-bits/pixel [no palette buffer] others = reserved **NOTE:** This bit field is ignored if OVL2C2[FOR] is configured to be any YCbCr format (0b001–0b100)
19:10	R/W	LPO2	Number of Lines for Overlay 2 Frame The Lines per Overlay 2 (LPO2) bit field specifies the number line or rows present in the Overlay 2 frame. This also represents the size of the Overlay in the vertical direction. LPO2[9:0] represents a value between 1 and 600 lines per screen. This specifies the size of the Overlay 2 frame in the vertical direction. This is programmed with the value from 0 to 599. Actual number of line = LPO2 +1 If YCbCr 4:2:0 planar format is selected, LPO2 must be > 1.

Table 7-48. OVL2C1 Bit Definitions (Sheet 2 of 2)

Physical Address 0x4400_0070 — **OVL2C1** — **LCD Controller**

User Settings

Bit	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
	O2EN	reserved							BPP2				LPO2										PPL2									
Reset	0	?	?	?	?	?	?	?	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
9:0	R/W	PPL2	Pixel per Line for Overlay 2 Frame The Pixels per Line bit field specifies the number of pixels in each line or row for the Overlay 2 frame. PPL2[9:0] represents a value between 1 and 800 pixels per line. See Section 7.4.13 for the restrictions on pixels per line. Value (from 0 to 799). Actual pixel per line = (PPL2+1). If YCbCr 4:2:2 or 4:2:0 Planar formats are selected, PPL2 must be > 1. **NOTE:** 16 pixels for Overlay 2 frame when the data is in 4:2:0 YCbCr format.

7.5.11 Overlay 2 Control Register 2 (OVL2C2)

Overlay 2 Control register 2 (OVL2C2) (Table 7-49) contains bit fields that are programmed to set the format of the pixel data and the position of Overlay 2 on the panel display.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-49. OVL2C2 Bit Definitions

Physical Address 0x4400_0080 — OVL2C2 — LCD Controller

User Settings

Bit	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									FOR			O2YPOS												O2XPOS							
Reset	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

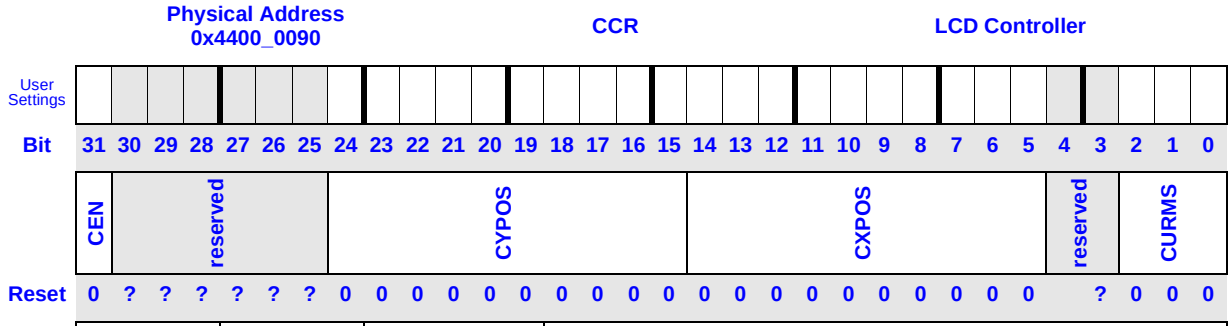
Bits	Access	Name	Description
31:23	—	—	reserved
22:20	R/W	FOR	Format The pixel format (FOR) bit field specifies the format of the pixel data for Overlay 2. The pixel data can be in RGB or YCbCr format as shown below. Specifies the data format stored in the memory 0b000 = RGB 0b001 = YCbCr 4:4:4 Packed 0b010 = YCbCr 4:4:4 Planar 0b011 = YCbCr 4:2:2 Planar 0b100 = YCbCr 4:2:0 Planar Others = reserved **NOTE:** Setting FOR to any YCbCr value (0b001 through 0b100) causes the value of OVL2C1[BPP2] to be ignored.
19:10	R/W	O2YPOS	Vertical Position of Upper Left Most Pixel of Overlay 2 O2YPOS specifies the vertical position of the Overlay 2 frame. The Vertical position of the Overlay 2 frame is the Y-coordinate of the upper left-most pixel, with respect to the origin of the screen. O2YPOS[9:0] represents a value between 1 and 600. Value is between 0 and 599. The origin (0, 0) of the screen is top left-most corner.
9:0	R/W	O2XPOS	Horizontal Position of Upper Left Most Pixel of Overlay 2 O2XPOS specifies the horizontal position of the Overlay 2 frame. The horizontal position of the frame is the X-coordinate of the upper left-most pixel, with respect to the origin of the screen. O2XPOS[9:0] represents a value between 1 to 800. Value is between 0 to 799. The origin (0, 0) of the screen is top left-most corner.

7.5.12 Cursor Control Register (CCR)

The Cursor Control register (CCR) (Table 7-50) contains bit fields that enable, configure, and set the position of the hardware cursor.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-50. CCR Bit Definitions



Bits	Access	Name	Description
31	R/W	CEN	Cursor Enable This bit enables the hardware cursor. When set, the hardware cursor is enabled. When clear, the hardware cursor is disabled. 0 = Hardware Cursor disabled 1 = Hardware Cursor enabled
30:25	—	—	reserved
24:15	R/W	CYPOS	Vertical Position of the Cursor The Vertical Position of the Cursor (CYPOS) bit field specifies the Y coordinates of the upper-left corner pixel of the cursor. CYPOS is 10-bit value and the value can be from 0 to 599. Value(0 to 599)
14:5	R/W	CXPOS	Horizontal Position of the Cursor The Horizontal Position of the Cursor (CXPOS) bit field specifies the X coordinates of the upper-left corner pixel of the cursor. CXPOS is 10-bit value and the value can be from 0 to 799. Value(0 to 799).
4:3	—	—	reserved
2:0	R/W	CURMS	Cursor Mode Select This selects one or the six available cursor modes. 0b000 = 32x32x2bpp 2-color and transparency mode 0b001 = 32x32x2bpp 3-color and transparency mode 0b010 = 32x32x2bpp 4-color mode 0b011 = 64x64x2bpp 2-color and transparency mode 0b100 = 64x64x2bpp 3-color and transparency mode 0b101 = 64x64x2bpp 4-color mode. 0b110 = 128x128x1bpp 2-color mode 0b111 = 128x128x1bpp 1-color and transparency mode

7.5.13 Command Control Register (CMDCR)

The Command Control register (CMDCR) (Table 7-51) contains bit fields that program the counter values used for synchronization when interfacing with the LCD panels with internal frame buffer.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-51. CMDCR Bit Definitions

Physical Address 0x4400_0100																CMDCR												LCD Controller							
User Settings																																			
Bit	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																								SYNC_CNT							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	SYNC_CNT	Synchronous Count The synchronization counter value (SYNC_CNT) specifies the counter value that the LCD controller has to count at the fall of input signal, L_VSYNC whenever the Wait for Vsync command is executed before going to the next command in the command FIFO. When the wait-for-Vsync command is executed, this is the counter value used to count at the fall of L_VSYNC before executing the next command in the command FIFO This applies only when interfacing with LCD panel with an internal frame buffer. This counter value must be programmed according to the LCD panel specifications.

7.5.14 TMED RGB Seed Register (TRGBR)

This register contains the three (red, green, blue) eight-bit seed values used by the TMED algorithm. This value is added into the modified pixel-data value as an offset in creating the lower boundary for the algorithm. These values are used during the dithering process for passive (STN and DSTN) matrix displays. The default recommended setting is 0x00AA_5500. Refer to Section 7.4.1.1.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-52. TRGBR Bit Definitions

Physical Address 0x4400_0040																TRGBR																LCD Controller							
User Settings																																					
Bit	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								TBS								TGS								TRS							
Reset	?	?	?	?	?	?	?	?	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:24	—	—	reserved
23:16	R/W	TBS	TMED Blue Seed Value
15:8	R/W	TGS	TMED Green Seed Value
7:0	R/W	TRS	TMED Red Seed Value

7.5.15 TMED Control Register (TCR)

This register selects the options available to use a patented temporal-modulated energy-distribution (TMED) algorithm, which improves image quality on STN (passive) panels. Two TMED algorithms can be used. The default recommended setting is 0x0000_754F.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-53. TCR Bit Definitions (Sheet 1 of 2)

Physical Address 0x4400_0030 — **TCR** — **LCD Controller**

User Settings

Bit	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																	TED	TSCS		THBS				TVBS				TM1EN	TM2EN	TM1S	TM2S
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1	1	0	1	0	1	0	1	0	0	1	1	1	1

Bits	Access	Name	Description
31:15	—	—	reserved
14	R/W	TED	TMED Energy Distribution Select This bit selects which matrix is used in the final step of TMED algorithm. A 1 selects the (preferred) TMED2 matrix, a 0 selects the older TMED matrix. After the pixel value has gone through the algorithm to determine a lower and upper boundary, the row and column counters are combined and run through one of the matrices to obtain a number that will be compared to the 2 boundaries. If that number is between the 2 boundaries, then the data out for this pixel in this frame is a 1, otherwise it is a 0. 0 = Selects Scheme 1 1 = Selects Scheme 2
13:12	R/W	TSCS	TMED Shades per Color Select This 2-bit field allows software to adjust the value of the incoming pixel by rounding off the 1, 2, or 3 least significant bits. This adjusted value will then be used in both the color-offset adjuster and as the pixel value in the multiplication step of the TMED algorithm. The shades per color (distinct pixel values) can then be reduced to 33, 65, or 129; or remain at 256. 0b00 = Selects 33 shades for red, 65 shades for green, and 33 shades for blue. 0b01 = Selects 65 shades for red, blue, and green. 0b10 = Selects 129 shades for red, blue, and green. 0b11 = Selects 256 shades for red, blue, and green.
11:8	R/W	THBS	TMED Horizontal Beat Suppression This is the column-shift value used as an offset that is combined with the row (line) counter and the pixel counter to create an address to lookup in the matrix. The matrix output is compared to the upper and lower boundaries defined in Section 7.4.1.1. Specifies the column shift value
7:4	R/W	TVBS	TMED Vertical Beat Suppression This is the block-shift value used as an offset that is combined with the pixel counter. Specifies the block shift value

Table 7-53. TCR Bit Definitions (Sheet 2 of 2)

Physical Address 0x4400_0030 — TCR — LCD Controller

User Settings																															
Bit	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																	TED	TSCS		THBS				TVBS				TM1EN	TM2EN	TM1S	TM2S
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1	1	0	1	0	1	0	1	0	0	1	1	1	1

Bits	Access	Name	Description
3	R/W	TM1EN	TMED Method 1 Enable This is the frame shift enable bit that allows the frame-number adjuster to add an offset to the current frame number before the value is sent through the algorithm. Setting this bit enables the addition of the current frame number to a value composed from the row and column counters. This value is derived from going through one of the two look up matrices, which is selected by bit 1 (TMED Method 1 Select). 0 = Disables Scheme 1 1 = Enables Scheme 2
2	R/W	TM2EN	TMED Method 2 Enable This bit enables the color-offset adjuster for each color. The color-offset adjuster creates the offset in the lower boundary in the TMED algorithm (the formula is: LB = PixelValue X FrameNumber + Offset). The Offset is created by adding either the output of the lookup matrix (input was the color value) or 00 to the seed value in the TSR for that color. The color-offset adjuster for each color can be disabled by clearing this bit. When cleared, this bit selects only the Seed register value to go through the algorithm. 0 = Disables Scheme 1 1 = Enables Scheme 2
1	R/W	TM1S	TMED Method 1 Select This bit selects which matrix is used when using the frame-number adjuster. A 1 will select the (preferred) TMED2 matrix, and a 0 will select the older TMED matrix. 0 = Selects Scheme 1 1 = Selects Scheme 2
0	R/W	TM2S	TMED Method 2 Select This bit selects which matrix is used when using the color-offset adjuster. A 1 will select the (preferred) TMED2 matrix, and a 0 will select the older TMED matrix. 0 = Selects Scheme 1 1 = Selects Scheme 2

7.5.16 DMA Frame Descriptor Address Registers (FDADRx)

These registers contain the memory address of the next descriptor for that channel. The DMA controller fetches the descriptor at this location after finishing the current descriptor. The bits in this register are undefined at power on. The descriptor address needs to be aligned to a 128-bit (16-byte) boundary; therefore, bits [3:0] of the address are reserved.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-54. FDADR0/1/2/3/4/5/6 Bit Definitions

Physical Address		LCD Controller
0x4400_0200	FDADR0	
0x4400_0210	FDADR1	
0x4400_0220	FDADR2	
0x4400_0230	FDADR3	
0x4400_0240	FDADR4	
0x4400_0250	FDADR5	
0x4400_0260	FDADR6	

User Settings																															
Bit	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
	DESCRIPTOR ADDRESS																												reserved			
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	?	?	?	?

Bits	Access	Name	Description
31:4	R/W	DESCRIPTOR ADDRESS	Descriptor Address Address of next Descriptor
3:0	—	—	reserved

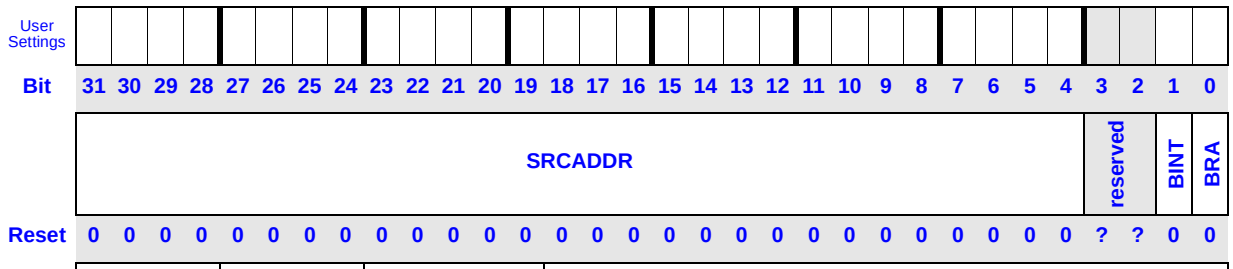
7.5.17 DMA Frame Branch Registers (FBRx)

The Frame Branch registers contain the address of the descriptor to branch to (aligned on a 4-byte boundary). When writing this register and setting BRA = 1, the FDADR (Frame Descriptor Address register) is ignored and the next descriptor is fetched from the address in this register. Setting BINT = 1 tells the DMA to set the branch status interrupt bit (BS) in the LCD Controller Status register after fetching the branched-to descriptor. The BRA bit is automatically cleared by the hardware when the branch is taken.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-55. FBR0/1/2/3/4/5/6 Bit Definitions

Physical Addresses		
0x4400_0020	FBR0	LCD Controller
0x4400_0024	FBR1	
0x4400_0028	FBR2	
0x4400_002C	FBR3	
0x4400_0030	FBR4	
0x4400_0110	FBR5	
0x4400_0114	FBR6	



Bits	Access	Name	Description
31:4	R/W	SRCADDR	Frame Branch Address Address of the descriptor for the branched-to frame.
3:2	—	—	reserved
1	R/W	BINT	Branch Interrupt 0 = Do not set the BS bit. 1 = Set branch status (BS) interrupt bit in LCD Controller Status register after branched-to descriptor is loaded.
0	R/W	BRA	Branch after Finishing the Current Frame 0 = Do not branch. 1 = The next descriptor will be fetched from the frame-branch address. This bit is automatically cleared after loading the new descriptor.

7.5.18 LCD Buffer Strength Control Register (LCDBSCNTR)

LCDBSCNTR is a read/write register that contains control bits for configuring the buffer strengths of the LCD controller output buffers. The output buffer strength configures all of the LCD controller signals (see Table 7-1) to the same output drive strength.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-56. LCDBSCNTR Bit Definitions

Physical Address 0x4800_0054																LCDBSCNTR												LCD Controller			
User Settings																															
Bit	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																												LCDBS			
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	0	1

Bits	Access	Name	Description
31:4	—	—	reserved
3:0	R/W	LCDBS	LCD Buffer Strength This value configures the output drive strength for all of the LCD controller signals. A smaller value configures a weaker drive strength.

7.5.19 Panel Read Status Register (PRSR)

The Panel Read Status register (PRSR) (Table 7-57) contains bit fields that load the read data from the LCD module by the LCD controller, and the status bits. This register is valid only when the LCD controller is configured to operate with smart panels. The Panel Read Status register is described in Table 7-40.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-57. PRSR Bit Definitions

Physical Address 0x4400_0104 — PRSR — LCD Controller

	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
User Settings																																
Bit	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																					CON_NT	ST_OK	A0	DATA							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31-11	—	—	reserved
10	R/W	CON_NT	Continue to Next Command Used by the LCD controller to execute the command. If CONT_NT = 0, the LCD controller waits for processor intervention. If CONT_NT = 1 and ST_OK = 1, the next command in the command FIFO is executed. If CONT_NT = 1 and ST_OK = 0, the same command executed previously is executed again. 0 = Wait for processor intervention. 1 = Continue to the next command or repeat the same command.
9	R/W	ST_OK	Status OK Used by the LCD controller to go to the next command in the FIFO or execute the same command. The LCD controller resets this bit every time it executes a read command. If LCCR0[DIS] is set this bit must be set to go to next command (PRSR[ST_OK] set. This bit must be set to go to next command if LCCR0[DIS] is set. 0 = Status is not OK. 1 = Status is OK.
8	R/W	A0	Read Data Source The status or data bit (A0) bit field specifies the read data from the Status register or from the frame-buffer RAM. This specifies the read data is status read data or the read data from the frame buffer RAM. 0 = Read data from smart panel Status register. 1 = Read data from the smart panel frame-buffer RAM.
7:0	R	DATA	Panel Data The panel-data (DATA) bit field loads the read data from the LCD module. Data from the LCD module.

7.5.20 LCD Controller Status Register 0 (LCSR0)

The LCD Controller Status register 0 (LCSR0) contains bits that signal overrun and underrun errors for both the input and output FIFOs, AC bias pin transition count, LCD disabled, DMA start/end frame and Branch status, and DMA transfer bus error conditions. Unless masked, each of these hardware-detected events signal an interrupt request to the interrupt controller.

Each of the LCD status bits signals an interrupt request as long as the bit is set. Once the bit is cleared, the interrupt request is cleared. Read/write bits are called *status* bits (read-only bits are called *flags*). Status bits are referred to as *sticky*, meaning that once set by hardware, they must be cleared by software. Writing a one to a sticky status bit clears it; writing a zero has no effect. Read-only flags are set and cleared by hardware; writes have no effect. Users have the ability to mask all LCD interrupts. See the Chapter 25, “Interrupt Controller” for more details.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-58. LCSR0 Bit Definitions (Sheet 1 of 5)

Physical Address 0x4400_0038 — **LCSR0** — **LCD Controller**

Bit	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
User Settings																																
	reserved	BER_CH			reserved															CMD_INT	RD_ST	SINT	BS0	EOF0	QD	OU	IU1	IU0	ABC	BER	SOF0	LDD
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

Bits	Access	Name	Description
31	—	—	reserved
30:28	R	BER_CH	Bus Error Channel Number The bus-error-channel (BER_CH) bit field specifies the channel number for which the DMA transfer causes a bus error on the system bus. The channel number in which the bus error has occurred. When BER is set, this bit field specifies on which channel the bus error has occurred. 0b000 = Channel 0 0b001 = Channel 1 0b010 = Channel 2 0b011 = Channel 3 0b100 = Channel 4 0b101 = Channel 5 0b110 = Channel 6 0b111 = Channel 7
27:13	—	—	reserved

Table 7-58. LCSR0 Bit Definitions (Sheet 2 of 5)

Physical Address 0x4400_0038	LCSR0	LCD Controller

Bit	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
User Settings																																
	reserved	BER_CH			reserved															CMD_INT	RD_ST	SINT	BS0	EOF0	QD	OU	IU1	IU0	ABC	BER	SOF0	LDD
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

Bits	Access	Name	Description
12	R/W	CMD_INT	Command Interrupt Status The command-interrupt status (CMD_INT) bit is set whenever the LCD controller executes an interrupt command. This bit applies only to panels with an internal frame buffer. When CMD_INT is set, an interrupt is sent to the interrupt controller if it is unmasked (CMDIM = 0). CMD_INT remains set until cleared. 0 = The LCD controller has not executed the interrupt command. 1 = The LCD controller sets this bit when ever it executes the interrupt command.
11	R/W	RD_ST	Read Status The read status (RD_ST) bit is set whenever the LCD controller executes a read command to the LCD module. When RD_ST is set, an interrupt is sent to the interrupt controller if it is unmasked (RDSTM = 0). RD_ST remains set until cleared by users. 0 = Cleared state of the bit. The user can clear it to zero. 1 = The LCD sets this bit whenever it executes the read command
10	R/W	SINT	Subsequent Interrupt Status The subsequent interrupt status (SINT) is a read/write bit that is set when an unmasked interrupt occurs and there is a pending interrupt. The frame ID of the first interrupt is saved in the interrupt frame ID register. This bit is only set for bus error, start of frame, end of frame, branch status, command interrupt, read status interrupts. Writing a 1 to this bit clears it. 0 = No second unmasked branch, start of frame, end of frame, bus error, command interrupt or read status interrupt has occurred. 1 = Another unmasked branch, start of frame, end of frame, bus error, command interrupt or read status interrupt has occurred before the previous interrupt (the frame in the interrupt Frame ID register) has been cleared.
9	R/W	BS0	Branch Status for Base The branch status for channel 0 (BS0) is set after the DMA controller has branched and loaded the descriptor from FBR0[SRCADDR], and the branch interrupt bit (FBR0[BINT]) is set. When the branch status bit is set, an interrupt request is made to the interrupt controller if the branch-status mask is unmasked (LCCR0[BSM0] cleared). When dual-scan mode is enabled (LCCR0[SDS] set), both DMA channels are enabled, and BS0 is set only after both channel frames have been fetched. BS0 remains set until a zero is written to it. 0 = The DMA has not loaded a branched-to descriptor, or the DMA branched but the branch interrupt (FBR0[BINT]) bit is not set. 1 = The DMA has loaded a branched-to descriptor and the branch-interrupt (FBR0[BINT]) bit is set.

Table 7-58. LCSR0 Bit Definitions (Sheet 3 of 5)

Physical Address 0x4400_0038 — **LCSR0** — **LCD Controller**

	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
User Settings																																
Bit	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	BER_CH			reserved															CMD_INT	RD_ST	SINT	BS0	EOF0	QD	OU	IU1	IU0	ABC	BER	SOF0	LDD
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

Bits	Access	Name	Description
8	R/W	EOF0	End of Frame Status for Base (Channel 0) The end of frame for channel 0 (EOF0) status is set after the DMA controller has finished fetching a frame from memory and the channel 0 Descriptor has the end of frame interrupt (EOFINT) bit set (bit 21 of the fourth word of the DMA descriptor). When EOF0 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR0[EOFM0] cleared). When dual-scan mode is enabled (LCCR0[SDS] set), both DMA channels 0 and 1 are enabled, and EOF0 is set only after both channel frames have been fetched. EOF0 remains set until cleared by software. 0 = A new frame has not been processed or the EOFINT bit is not set in the fourth word of Channel 0 descriptor. 1 = The DMA has finished fetching a frame and the EOFINT bit is set in the fourth word of Channel 0 descriptor.
7	R/W	QD	LCD Quick Disable Status The LCD quick-disable status (QD) bit is set when LCD enable (ENB) is cleared and the DMA finishes its current data burst. When QD is set, an interrupt request is made to the interrupt controller if it is unmasked (QDM = 0). This forces the LCD controller to stop immediately and quit driving the LCD pins. This method of disable is intended for use with sleep shutdown. 0 = LCD has not been quickly disabled by clearing LCD enable (ENB). 1 = LCD has been quickly disabled.
6	R/W	OU	Output FIFO Underrun The output-FIFO-underrun lower-panel status (OU) bit is set when an output FIFO is completely empty and the LCD data pin driver logic attempts to fetch data from the FIFO. It is cleared by writing a one to the bit. This bit is used for single and dual displays. In dual-scan mode (SDS = 1), both FIFOs are filled and read out data at the same time, so underrun occurs at the same time. When OU is set, an interrupt request is made to the interrupt controller if it is unmasked (OUM = 0). 0 = Output FIFO for the lower panel display has not underrun. 1 = LCD dither logic not supplying data to output FIFO for the LCD panel at a sufficient rate. FIFO has completely emptied and data pin driver logic has attempted to take added data from the FIFO.

Table 7-58. LCSR0 Bit Definitions (Sheet 4 of 5)

Physical Address 0x4400_0038 — **LCSR0** — **LCD Controller**

Bit	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
User Settings																																
	reserved	BER_CH			reserved															CMD_INT	RD_ST	SINT	BS0	EOF0	QD	OU	IU1	IU0	ABC	BER	SOF0	LDD
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

Bits	Access	Name	Description
5	R/W	IU1	Input FIFO Underrun for Channel 1 The input FIFO underrun for each channel 1 (IU1) bit is set when the input FIFO is completely empty and the LCD pixel unpacking logic attempts to fetch data from the FIFO. It is cleared by writing a one to the bit. When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (IUM1 = 0). 0 = Input FIFO for channel 1 has not underrun. 1 = DMA not supplying data to the input FIFO for channel 1 at a sufficient rate. FIFO has completely emptied; pixel unpacking logic has attempted to take added data from the FIFO.
4	R/W	IU0	Input FIFO Underrun for Channel 0 The input FIFO underrun for each channel 0 (IU0) bit is set when the input FIFO is completely empty and the LCD pixel unpacking logic attempts to fetch data from the FIFO. It is cleared by writing a one to the bit. When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (IUM0 = 0). 0 = Input FIFO for channel 0 has not underrun. 1 = DMA not supplying data to the input FIFO for channel 0 at a sufficient rate. FIFO has completely emptied, pixel unpacking logic has attempted to take added data from the FIFO.
3	R/W	ABC	AC Bias Count Status The AC bias-count status (ABC) bit it set each time the AC bias pin (L_BIAS) toggles a particular number of times as specified by the AC bias pin transitions per interrupt (API) field in LCCR3. If API is programmed with a non-zero value, a counter is loaded with the value in API and is decremented each time the L_BIAS pin reverses state. When the counter reaches zero, the ABC bit is set, which signals an interrupt request to the interrupt controller. The counter reloads using the value in API, but does not start to decrement again until ABC is cleared by users. 0 = AC bias transition counter has not decremented to zero, or API is programmed to all zeros. 1 = AC bias transition counter has decremented to zero, indicating that the L_BIAS pin has toggled the number of times specified by the API control bit field. Counter is reloaded with the value in API but is disabled until ABC is cleared.

Table 7-58. LCSR0 Bit Definitions (Sheet 5 of 5)

Physical Address 0x4400_0038 — **LCSR0** — **LCD Controller**

Bit	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
User Settings																																
	reserved	BER_CH			reserved															CMD_INT	RD_ST	SINT	BS0	EOF0	QD	OU	IU1	IU0	ABC	BER	SOF0	LDD
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

Bits	Access	Name	Description
2	R/W	BER	Bus Error Status The bus-error status (BER) bit is set when a DMA transfer causes a bus error to occur on the system bus. A bus error is signaled when the DMA controller attempts to access a reserved or nonexistent memory space. The bus error channel (BER_CH) specifies for which channel the bus error has occurred. When this occurs, the DMA controller stops and is halted until users program the FDADR register of channel specified by BER_CH with a valid memory address. BER remains set until cleared by users. 0 = DMA has not attempted an access to reserved/nonexistent memory space. 1 = DMA has attempted an access to a reserved/nonexistent location in external memory. The DMA engine stops and must be re-programmed to resume operation.
1	R/W	SOF0	Start of Frame Status for Base (Channel 0) The start of frame for channel 0 (SOF0) status is set after the DMA controller has loaded a new descriptor for channel 0 from memory and that descriptor has the start of frame interrupt (SOFINT) bit set (bit 22 of the fourth word of the channel 0 DMA descriptor). When SOF0 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR0[SOFM0] cleared). When dual-scan mode is enabled (LCCR0[SDS] set), both DMA channels 0 and 1 are enabled, and SOF0 is set only after both channel descriptors have been loaded. SOF0 remains set until cleared by software. 0 = A new frame descriptor has not been fetched or the SOFINT bit is not set in the fourth word of Channel 0 descriptor. 1 = The DMA has begun fetching a new frame and the SOFINT bit is set in the fourth word of Channel 0 descriptor.
0	R/W	LDD	LCD Disable Done Flag The LCD disable-done flag (LDD) is set after the LCD has been disabled and the frame that is active finishes being output to the LCD data pins. When the LCD is disabled by setting the LCD disable bit (LCCR0[DIS]), the LCD allows the current frame to complete before it is disabled. After the last set of pixels is clocked out onto the LCD data pins by the pixel clock, the LCD is disabled, LDD is set, and an interrupt request is made to the interrupt controller if it is unmasked (LDM = 0). This interrupt is useful to allow an orderly shutdown of the LCD controller before users place the processor into sleep mode. 0 = LCD has not been disabled and the last active frame completed. 1 = LCD has been disabled and the last active frame has completed.

7.5.21 LCD Controller Status Register 1 (LCSR1)

The LCD Controller Status register 1 (LCSR1) contains interrupt status bits for Overlay 1, Overlay 2, cursor, and command data. Each of these hardware-detected events signals an interrupt request to the interrupt controller. See Table 7-59.

Each of the LCD status bits signal an interrupt request as long as the bit is set. Once the bit is cleared, the interrupt request is cleared. Read/write bits are called *status* bits (read-only bits are called *flags*). Status bits are referred to as *sticky* (once set by hardware, they must be cleared by software). Writing a one to a sticky status bit clears it; writing a zero has no effect. Read-only flags are set and cleared by hardware; Writes have no effect. All LCD interrupts can be masked. See Chapter 25, “Interrupt Controller” for more details.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-59. LCSR1 Bit Definitions (Sheet 1 of 7)

Physical Address 0x4400_0034	LCSR1	LCD Controller

	Bit	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
User Settings																																	
		reserved		IU6	IU5	IU4	IU3	IU2	reserved			BS6	BS5	BS4	BS3	BS2	BS1	reserved		EOF6	EOF5	EOF4	EOF3	EOF2	EOF1	reserved		SOF6	SOF5	SOF4	SOF3	SOF2	SOF1
	Reset	?	?	0	0	0	0	0	?	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
31:30	—	—	reserved
29	R/W	IU6	Input FIFO Underrun for Channel 6 The input-FIFO underrun for Channel 6 (IU6) bit is set when the input FIFO is completely empty and the LCD pixel unpacking logic attempts to fetch data from the FIFO. It is cleared by writing a one to the bit. When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (mask bit LCCR5[IUM6] cleared). 0 = Input FIFO for Channel 6 has not underrun. 1 = DMA not supplying data to the input FIFO for Channel 6 at a sufficient rate. FIFO has completely emptied; pixel unpacking logic has attempted to take added data from the FIFO.
28	R/W	IU5	Input FIFO Underrun for Channel 5 The input-FIFO underrun for Channel 5 (IU5) bit is set when the input FIFO is completely empty and the LCD pixel unpacking logic attempts to fetch data from the FIFO. It is cleared by writing a one to the bit. When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (mask bit LCCR5[IUM5] cleared). 0 = Input FIFO for Channel 5 has not underrun. 1 = DMA not supplying data to the input FIFO for Channel 5 at a sufficient rate. FIFO has completely emptied; pixel unpacking logic has attempted to take added data from the FIFO.

Table 7-59. LCSR1 Bit Definitions (Sheet 2 of 7)

Physical Address 0x4400_0034 — **LCSR1** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IU6	IU5	IU4	IU3	IU2	reserved			BS6	BS5	BS4	BS3	BS2	BS1	reserved		EOF6	EOF5	EOF4	EOF3	EOF2	EOF1	reserved		SOF6	SOF5	SOF4	SOF3	SOF2	SOF1
Reset	?	?	0	0	0	0	0	?	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
27	R/W	IU4	Input FIFO Underrun for Channel 4 The input-FIFO underrun for Channel 4 (IU4) bit is set when the input FIFO is completely empty and the LCD pixel unpacking logic attempts to fetch data from the FIFO. It is cleared by writing a one to the bit. When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (mask bit LCCR5[IUM4] cleared). 0 = Input FIFO for Channel 4 has not underrun. 1 = DMA is not supplying data to the input FIFO for Channel 4 at a sufficient rate. FIFO has completely emptied; pixel unpacking logic has attempted to take additional data from the FIFO.
26	R/W	IU3	Input FIFO Underrun for Channel 3 The input-FIFO underrun for Channel 3 (IU3) bit is set when the input FIFO is completely empty and the LCD pixel unpacking logic attempts to fetch data from the FIFO. It is cleared by writing a one to the bit. When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (mask bit LCCR5[IUM3] cleared). 0 = Input FIFO for Channel 3 has not underrun. 1 = DMA not supplying data to the input FIFO for Channel 3 at a sufficient rate. FIFO has completely emptied; pixel unpacking logic has attempted to take added data from the FIFO.
25	R/W	IU2	Input FIFO Input FIFO for Channel 2 The input-FIFO underrun for Channel 2 (IU2) bit is set when the input FIFO is completely empty and the LCD pixel unpacking logic attempts to fetch data from the FIFO. It is cleared by writing a one to the bit. When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (mask bit LCCR5[IUM2] cleared). 0 = Input FIFO for Channel 2 has not underrun. 1 = DMA not supplying data to the input FIFO for Channel 2 at a sufficient rate. FIFO has completely emptied; pixel unpacking logic has attempted to take added data from the FIFO.
24:22	—	—	reserved
21	R/W	BS6	Branch Status for Channel 6 (Command Register). The branch status for Channel 6 (BS6) is set after the DMA controller has branched and loaded the descriptor from FBR6[SRCADDR], and the branch-interrupt bit (FBR6[BINT]) is set. When BS6 is set, an interrupt request is made to the interrupt controller if the branch status mask is unmasked (LCCR5[BSM6] cleared). BS6 remains set until a zero is written to it. 0 = The DMA has not loaded a branched-to descriptor, or the DMA branched but the branch interrupt (FBR6[BINT]) bit is not set. 1 = The DMA has loaded a branched-to descriptor and the branch-interrupt (FBR6[BINT]) bit is set.

Table 7-59. LCSR1 Bit Definitions (Sheet 3 of 7)

Physical Address 0x4400_0034 — **LCSR1** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IU6	IU5	IU4	IU3	IU2	reserved			BS6	BS5	BS4	BS3	BS2	BS1	reserved		EOF6	EOF5	EOF4	EOF3	EOF2	EOF1	reserved		SOF6	SOF5	SOF4	SOF3	SOF2	SOF1
Reset	?	?	0	0	0	0	0	?	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
20	R/W	BS5	Branch Status for Channel 5 (Hardware Cursor) The branch status for Channel 5(BS5) is set after the DMA controller has branched and loaded the descriptor from FBR5[SRCADDR], and the branch interrupt bit (FBR5[BINT]) is set. When BS5 is set, an interrupt request is made to the interrupt controller if the branch status mask is unmasked (LCCR5[BSM5] cleared). BS5 remains set until a zero is written to it. 0 = The DMA has not loaded a branched-to descriptor, or the DMA branched but the branch interrupt (FBR5[BINT]) bit is not set. 1 = The DMA has loaded a branched-to descriptor and the branch-interrupt (FBR5[BINT]) bit is set.
19	R/W	BS4	Branch Status for Channel 4 (Overlay 2) The branch status for Channel 4 (BS4) is set after the DMA controller has branched and loaded the descriptor from FBR4[SRCADDR], and the branch-interrupt bit (FBR4[BINT]) is set. When BS4 is set, an interrupt request is made to the interrupt controller if the branch status mask is unmasked (LCCR5[BSM4] cleared). BS6 remains set until a zero is written to it. 0 = The DMA has not loaded a branched-to descriptor, or the DMA branched but the branch interrupt (FBR4[BINT]) bit is not set. 1 = The DMA has loaded a branched-to descriptor and the branch-interrupt (FBR4[BINT]) bit is set.
18	R/W	BS3	Branch Status for Channel 3 (Overlay 2) The branch status for channel 3 (BS3) is set after the DMA controller has branched and loaded the descriptor from FBR3[SRCADDR], and the branch-interrupt bit (FBR3[BINT]) is set. When BS3 is set, an interrupt request is made to the interrupt controller if the branch status mask is unmasked (LCCR5[BSM3] cleared). BS3 remains set until a zero is written to it. 0 = The DMA has not loaded a branched-to descriptor, or the DMA branched but the branch interrupt (FBR3[BINT]) bit is not set. 1 = The DMA has loaded a branched-to descriptor and the branch-interrupt (FBR3[BINT]) bit is set.
17	R/W	BS2	Branch Status for Channel 2 (Overlay 2) The branch status for channel 2 (BS2) is set after the DMA controller has branched and loaded the descriptor from FBR2[SRCADDR], and the branch interrupt bit (FBR2[BINT]) is set. When BS2 is set, an interrupt request is made to the interrupt controller if the Branch Status Mask is unmasked (LCCR5[BSM2] cleared). BS2 remains set until a zero is written to it. 0 = The DMA has not loaded a branched-to Descriptor, or the DMA branched but the branch interrupt (FBR2[BINT]) bit is not set. 1 = The DMA has loaded a branched-to Descriptor and the branch-interrupt (FBR2[BINT]) bit is set.

Table 7-59. LCSR1 Bit Definitions (Sheet 4 of 7)

Physical Address 0x4400_0034 — **LCSR1** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IU6	IU5	IU4	IU3	IU2	reserved			BS6	BS5	BS4	BS3	BS2	BS1	reserved		EOF6	EOF5	EOF4	EOF3	EOF2	EOF1	reserved		SOF6	SOF5	SOF4	SOF3	SOF2	SOF1
Reset	?	?	0	0	0	0	0	?	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
16	R/W	BS1	Branch Status for Channel 1 (Overlay 1) The branch status for Channel 1 (BS1) is set after the DMA controller has branched and loaded the descriptor from FBR1[SRCADDR], and the branch-interrupt bit (FBR1[BINT]) is set. When BS1 is set, an interrupt request is made to the interrupt controller if the Branch Status Mask is unmasked (LCCR5[BSM1] cleared). BS1 remains set until a zero is written to it. 0 = The DMA has not loaded a branched-to descriptor, or the DMA branched but the branch-interrupt (FBR1[BINT]) bit is not set. 1 = The DMA has loaded a branched-to descriptor and the branch-interrupt (FBR1[BINT]) bit is set.
15:14	—	—	reserved
13	R/W	EOF6	End of Frame Status for Command Register The end-of-frame for Channel 6 (EOF6) status is set after the DMA controller has finished fetching a frame from memory and the Channel 6 descriptor has the end-of-frame interrupt (EOFINT) bit set (bit 21 of the fourth word of the DMA descriptor). When EOF6 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR5[EOFM6] cleared). EOF6 remains set until cleared by software. 0 = A new frame has not been processed or the EOFINT bit is not set in the fourth word of Channel 6 descriptor. 1 = The DMA has finished fetching a frame and the EOFINT bit is set in the fourth word of Channel 6 descriptor.
12	R/W	EOF5	End of Frame Status for Hardware Cursor The end-of-frame for Channel 5 (EOF5) status is set after the DMA controller has finished fetching a frame from memory and the Channel 5 descriptor has the end-of-frame interrupt (EOFINT) bit set (bit 21 of the fourth word of the DMA descriptor). When EOF5 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR5[EOFM5] cleared). EOF5 remains set until cleared by software. 0 = A new frame has not been processed or the EOFINT bit is not set in the fourth word of Channel 5 descriptor. 1 = The DMA has finished fetching a frame and the EOFINT bit is set in the fourth word of Channel 5 descriptor.

Table 7-59. LCSR1 Bit Definitions (Sheet 5 of 7)

Physical Address 0x4400_0034 — **LCSR1** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IU6	IU5	IU4	IU3	IU2	reserved			BS6	BS5	BS4	BS3	BS2	BS1	reserved		EOF6	EOF5	EOF4	EOF3	EOF2	EOF1	reserved		SOF6	SOF5	SOF4	SOF3	SOF2	SOF1
Reset	?	?	0	0	0	0	0	?	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
11	R/W	EOF4	End of Frame Status for Overlay 2 (Channel 4) The end-of-frame for Channel 4 (EOF4) status is set after the DMA controller has finished fetching a frame from memory and the Channel 4 descriptor has the end-of-frame interrupt (EOFINT) bit set (bit 21 of the fourth word of the DMA descriptor). When EOF4 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR5[EOFM4] cleared). EOF4 remains set until cleared by software. 0 = A new frame has not been processed or the EOFINT bit is not set in the fourth word of Channel 4 descriptor. 1 = The DMA has finished fetching a frame and the EOFINT bit is set in the fourth word of Channel 4 descriptor.
10	R/W	EOF3	End of Frame Status for Overlay 2 (Channel 3) The end-of-frame for Channel 3 (EOF3) status is set after the DMA controller has finished fetching a frame from memory and the Channel 3 descriptor has the end-of-frame interrupt (EOFINT) bit set (bit 21 of the fourth word of the DMA descriptor). When EOF3 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR5[EOFM3] cleared). EOF3 remains set until cleared by software. 0 = A new frame has not been processed or the EOFINT bit is not set in the fourth word of Channel 3 descriptor. 1 = The DMA has finished fetching a frame and the EOFINT bit is set in the fourth word of Channel 3 descriptor.
9	R/W	EOF2	End of Frame Status for Overlay 2 (Channel 2) The end-of-frame for Channel 2 (EOF2) status is set after the DMA controller has finished fetching a frame from memory and the Channel 2 descriptor has the end-of-frame interrupt (EOFINT) bit set (bit 21 of the fourth word of the DMA descriptor). When EOF2 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR5[EOFM2] cleared). EOF2 remains set until cleared by software. 0 = A new frame has not been processed or the EOFINT bit is not set in the fourth word of Channel 2 descriptor. 1 = The DMA has finished fetching a frame and the EOFINT bit is set in the fourth word of Channel 2 descriptor.
8	R/W	EOF1	End of Frame Status for Channel 1 The end-of-frame for Channel 1 (EOF1) status is set after the DMA controller has finished fetching a frame from memory and the Channel 1 descriptor has the end-of-frame interrupt (EOFINT) bit set (bit 21 of the fourth word of the DMA descriptor). When EOF1 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR5[EOFM1] cleared). EOF1 remains set until cleared by software. 0 = A new frame has not been processed or the EOFINT bit is not set in the fourth word of Channel 1 descriptor. 1 = The DMA has finished fetching a frame and the EOFINT bit is set in the fourth word of Channel 1 descriptor.

Table 7-59. LCSR1 Bit Definitions (Sheet 6 of 7)

Physical Address 0x4400_0034 — **LCSR1** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IU6	IU5	IU4	IU3	IU2	reserved			BS6	BS5	BS4	BS3	BS2	BS1	reserved		EOF6	EOF5	EOF4	EOF3	EOF2	EOF1	reserved		SOF6	SOF5	SOF4	SOF3	SOF2	SOF1
Reset	?	?	0	0	0	0	0	?	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
7:6	—	—	reserved
5	R/W	SOF6	Start of Frame Status for Command Register The start-of-frame for Channel 6 (SOF6) status is set after the DMA controller has loaded a new descriptor for Channel 6 from memory and that descriptor has the start-of-frame interrupt (SOFINT) bit set (bit 22 of the fourth word of the channel 6 DMA descriptor). When SOF6 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR5[SOFM6] cleared). SOF6 remains set until cleared by software. 0 = A new frame descriptor has not been fetched or the SOFINT bit is not set in the fourth word of Channel 6 descriptor. 1 = The DMA has begun fetching a new frame and the SOFINT bit is set in the fourth word of Channel 6 descriptor.
4	R/W	SOF5	Start of Frame Status for Hardware Cursor The start-of-frame for Channel 5 (SOF5) status is set after the DMA controller has loaded a new descriptor for Channel 5 from memory and that descriptor has the start-of-frame interrupt (SOFINT) bit set (bit 22 of the fourth word of the channel 5 DMA descriptor). When SOF5 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR5[SOFM5] cleared). SOF5 remains set until cleared by software. 0 = A new frame descriptor has not been fetched or the SOFINT bit is not set in the fourth word of Channel 5 descriptor. 1 = The DMA has begun fetching a new frame and the SOFINT bit is set in the fourth word of Channel 5 descriptor.
3	R/W	SOF4	Start of Frame Status for Overlay 2 (Channel 4) The start-of-frame for Channel 4 (SOF4) status is set after the DMA controller has loaded a new descriptor for Channel 4 from memory and that descriptor has the start-of-frame interrupt (SOFINT) bit set (bit 22 of the fourth word of the channel 4 DMA descriptor). When SOF4 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR5[SOFM4] cleared). SOF4 remains set until cleared by software. 0 = A new frame descriptor has not been fetched or the SOFINT bit is not set in the fourth word of Channel 4 descriptor. 1 = The DMA has begun fetching a new frame and the SOFINT bit is set in the fourth word of Channel 4 descriptor.

Table 7-59. LCSR1 Bit Definitions (Sheet 7 of 7)

Physical Address 0x4400_0034 — **LCSR1** — **LCD Controller**

Bit	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
User Settings																																
	reserved		IU6	IU5	IU4	IU3	IU2	reserved			BS6	BS5	BS4	BS3	BS2	BS1	reserved		EOF6	EOF5	EOF4	EOF3	EOF2	EOF1	reserved		SOF6	SOF5	SOF4	SOF3	SOF2	SOF1
Reset	?	?	0	0	0	0	0	?	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
2	R/W	SOF3	Start of Frame Status for Overlay 2 (Channel 3) The start-of-frame for Channel 3 (SOF3) status is set after the DMA controller has loaded a new descriptor for Channel 3 from memory and that descriptor has the start-of-frame interrupt (SOFINT) bit set (bit 22 of the fourth word of the Channel 3 DMA descriptor). When SOF3 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR5[SOFM3] cleared). SOF3 remains set until cleared by software. 0 = A new frame descriptor has not been fetched or the SOFINT bit is not set in the fourth word of Channel 3 descriptor. 1 = The DMA has begun fetching a new frame and the SOFINT bit is set in the fourth word of Channel 3 descriptor.
1	R/W	SOF2	Start of Frame Status for Overlay 2 (Channel 2) The start-of-frame for Channel 2 (SOF2) status is set after the DMA controller has loaded a new descriptor for Channel 2 from memory and that descriptor has the start-of-frame interrupt (SOFINT) bit set (bit 22 of the fourth word of the Channel 2 DMA descriptor). When SOF2 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR5[SOFM2] cleared). SOF2 remains set until cleared by software. 0 = A new frame descriptor has not been fetched or the SOFINT bit is not set in the fourth word of Channel 2 descriptor. 1 = The DMA has begun fetching a new frame and the SOFINT bit is set in the fourth word of Channel 2 descriptor.
0	R/W	SOF1	Start of Frame Status for Overlay 1 (Channel 1) The start of frame for Channel 1(SOF1) status is set after the DMA controller has loaded a new descriptor for Channel 1 from memory and that descriptor has the start-of-frame interrupt (SOFINT) bit set (bit 22 of the fourth word of the Channel 1 DMA descriptor). When SOF1 is set, an interrupt request is made to the interrupt controller if it is unmasked (LCCR5[SOFM1] cleared). SOF1 remains set until cleared by software. 0 = A new frame descriptor has not been fetched or the SOFINT bit is not set in the fourth word of Channel 1 descriptor. 1 = The DMA has begun fetching a new frame and the SOFINT bit is set in the fourth word of Channel 1 descriptor.

7.5.22 LCD Controller Interrupt ID Register (LIIDR)

LIIDR is a read-only register that contains a copy of the Frame ID register (FIDR) when a start-of-frame (SOF), end-of-frame (EOF), branch (BS), or bus-error (BER) interrupt is signaled. LIIDR is written only when an unmasked interrupt of the above type is signaled and there are no other unmasked interrupts in the LCD controller pending. In other words, the register is sticky, and is overwritten only when the signaled interrupt is cleared by writing the LCD Controller Status register.

Table 7-60. LIIDR Bit Definitions

Physical Address 0x4400_003C																LIIDR												LCD Controller			
User Settings																															
Bit	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
	IFRAMEID																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:0	R	IFRAMEID	Interrupt Frame ID

7.5.23 DMA Frame Source Address Registers (FSADRx)

These read-only registers contain the source address of the current descriptor for that channel. The address must be aligned on a 128-bit (16-byte) boundary. If this descriptor is a palette load, this register must point to the memory location at the beginning of the palette data. The size of the palette data must be four entries for 1- and 2-bit pixels, 16 entries for 4-bit pixels, and 256 entries for 8-bit pixels. If this descriptor is for pixel data, this register must point to the beginning of the frame data in memory. This address is incremented as the DMA controller fetches from memory. If preferred, the DMA Frame ID register can hold the initial frame-source address. See Table 7-61.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 7-61. FSADR0/1/2/3/4/5/6 Bit Definitions

Physical Address		
0x4400_0204	FSADR0	LCD Controller
0x4400_0214	FSADR1	
0x4400_0224	FSADR2	
0x4400_0234	FSADR3	
0x4400_0244	FSADR4	
0x4400_0254	FSADR5	
0x4400_0264	FSADR6	

User Settings																															
Bit	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
	SRCADDR																												reserved			
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	?	?	?	?

Bits	Access	Name	Description
31:4	R/W	SRCADDR	Frame Source Address: Address of the palette or pixel frame data in memory.
3:0	—	—	reserved

7.5.24 DMA Frame ID Registers (FIDRx)

These read-only registers contain a 32-bit ID field to describe the current frame. The particular use of this field is user defined. This ID register is copied to the LCD Controller Interrupt ID register when an interrupt occurs. See Table 7-62.

This is a read-only register. Ignore reads from reserved bits.

Table 7-62. FIDR0/1/2/3/4/5/6 Bit Definitions

Physical Address		
0x4400_0208	FIDR0	LCD Controller
0x4400_0218	FIDR1	
0x4400_0228	FIDR2	
0x4400_0238	FIDR3	
0x4400_0248	FIDR4	
0x4400_0258	FIDR5	
0x4400_0268	FIDR6	

User Settings																															
Bit	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
	FRAME ID																													reserved		
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	?

Bits	Access	Name	Description
31:3	R	FRAME ID	Frame ID
2:0	—	—	reserved

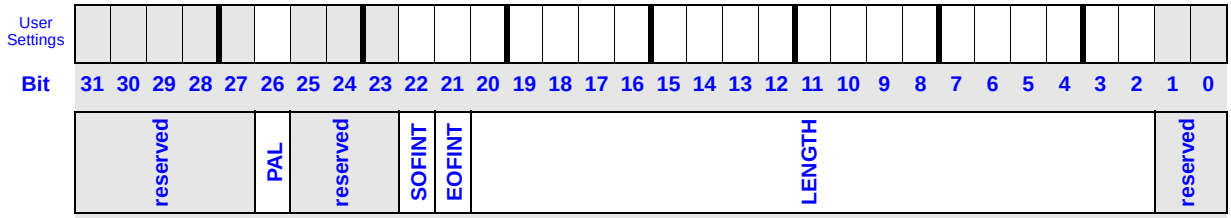
7.5.25 LCD DMA Command Register (LDCMDx)

These read-only registers contain the command and length of the current descriptor for that channel. The bits in this register are initialized to zero at power on. See Table 7-63.

This is a read-only register. Ignore reads from reserved bits.

Table 7-63. LDCMD0/1/2/3/4/5/6 Bit Definitions (Sheet 1 of 2)

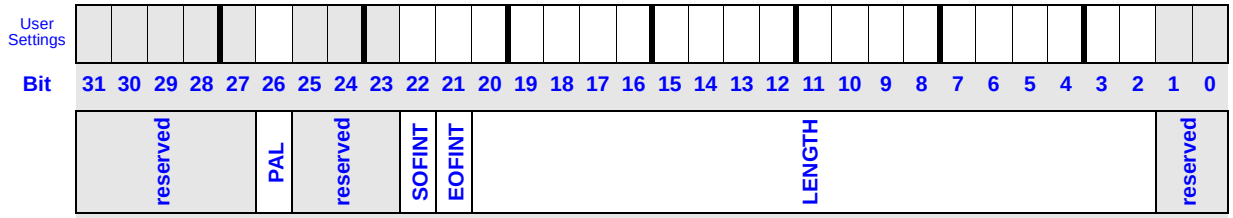
Physical Address		LCD Controller
0x4400_020C	LDCMD0	
0x4400_021C	LDCMD1	
0x4400_022C	LDCMD2	
0x4400_023C	LDCMD3	
0x4400_024C	LDCMD4	
0x4400_025C	LDCMD5	
0x4400_026C	LDCMD6	



Bits	Access	Name	Description
31:27	—	—	reserved
26	R	PAL	Palette Buffer If set, this indicates to the DMA controller that the data being fetched is the palette buffer and it must be loaded into the palette RAM. The palette RAM must be loaded at least once before using the LCD controller (unless more then 8-bit pixels are used). The size of the palette is as follows: 2-bit pixels = 4 Entries of palette data 4-bit pixels = 16 Entries of palette data 8-bit pixels = 256 Entries of palette data 0 = Data described in DMA descriptor is the frame buffer data. 1 = Data described in DMA descriptor is the palette buffer data and is to be loaded into the palette RAM
25:23	—	—	reserved
22	R	SOFINT	Start of Frame Interrupt When SOFINT is set, the DMA controller sets the corresponding start-of-frame bit (LCSR1[SOFx]) when starting a new frame. The bit is set after a new descriptor is loaded from memory, before the palette/frame data is fetched. 0 = Do not set SOF. 1 = Set the start-of-frame (SOF) interrupt-request bit in the LCD Status register when starting a new frame (after loading the frame descriptor).

Table 7-63. LDCMD0/1/2/3/4/5/6 Bit Definitions (Sheet 2 of 2)

Physical Address		LCD Controller
0x4400_020C	LDCMD0	
0x4400_021C	LDCMD1	
0x4400_022C	LDCMD2	
0x4400_023C	LDCMD3	
0x4400_024C	LDCMD4	
0x4400_025C	LDCMD5	
0x4400_026C	LDCMD6	



Bits	Access	Name	Description
21	R	EOFINT	End of the Frame Interrupt Mask When EOFINT is set the DMA controller sets the corresponding end-of-frame bit (LCSR1[EOFx]) after fetching the last word in the frame buffer, that is, when the DMA length counter decrements to zero. 0 = Do not set EOF. 1 = Set the end-of-frame (EOF) interrupt-request bit in the LCD Status register when finished fetching the last word of this frame.
20:2	R	LENGTH	Length The length of transfer in bytes. The value of LENGTH determines how many bytes of data the DMA controller fetches. Setting LENGTH = 0 is invalid. If PAL is set, LENGTH must be programmed with the size of the palette RAM. This corresponds to 8 bytes for 2-bit pixels, 32 or 64 bytes for 4-bit pixels, and 512 or 1024 bytes for 8-bit pixels. A separate descriptor must be used to fetch the frame data. The value of LENGTH for frame data is a function of the screen size and the pixel size. The transfer length must be word (32-bit) aligned. For Overlay 2 data that is in 4:2:2 or 4:2:0 YCbCr formats, the frame length may not naturally align to a 32-bit boundary. When this occurs, the transfer length must be programmed to the next 32-bit word padded with dummy pixels at the end of the frame (which the LCD controller will ignore). The two lowest bits [1:0] must always be zero for proper alignment. Writes to these two lower bits are ignored.
1:0	—	—	reserved

7.6 Register Summary

The LCD controller contains 16 control registers, 35 DMA registers, two status registers, and three 256-entry palette RAMs. Table 7-64 shows the registers associated with the LCD controller and the physical addresses to access them.

Table 7-64. LCD Controller Register Summary (Sheet 1 of 2)

Address	Name	Description	Page
0x4400_0000	LCCR0	LCD Controller Control register 0	7-56
0x4400_0004	LCCR1	LCD Controller Control register 1	7-64
0x4400_0008	LCCR2	LCD Controller Control register 2	7-66
0x4400_000C	LCCR3	LCD Controller Control register 3	7-69
0x4400_0010	LCCR4	LCD Controller Control register 4	7-74
0x4400_0014	LCCR5	LCD Controller Control register 5	7-79
0x4400_0018–0x4400_001C	—	reserved	
0x4400_0020	FBR0	DMA Channel 0 Frame Branch register	7-103
0x4400_0024	FBR1	DMA Channel 1 Frame Branch register	7-103
0x4400_0028	FBR2	DMA Channel 2 Frame Branch register	7-103
0x4400_002C	FBR3	DMA Channel 3 Frame Branch register	7-103
0x4400_0030	FBR4	DMA Channel 4 Frame Branch register	7-103
0x4400_0034	LCSR1	LCD Controller Status register 1	7-111
0x4400_0038	LCSR0	LCD Controller Status register 0	7-106
0x4400_003C	LIIDR	LCD Controller Interrupt ID register	7-118
0x4400_0040	TRGBR	TMED RGB Seed register	7-99
0x4400_0044	TCR	TMED Control register	7-100
0x4400_0048–0x4400_004C	—	reserved	
0x4400_0050	OVL1C1	Overlay 1 Control register 1	7-92
0x4400_0054–0x4400_005C	—	reserved	
0x4400_0060	OVL1C2	Overlay 1 Control register 2	7-93
0x4400_0064–0x4400_006C	—	reserved	
0x4400_0070	OVL2C1	Overlay 2 Control register 1	7-94
0x4400_0074–0x4400_007C	—	reserved	
0x4400_0080	OVL2C2	Overlay 2 Control register 2	7-96
0x4400_0084–0x4400_008C	—	reserved	
0x4400_0090	CCR	Cursor Control register	7-97
0x4400_0094–0x4400_009C	—	reserved	

Table 7-64. LCD Controller Register Summary (Sheet 2 of 2)

Address	Name	Description	Page
0x4400_0100	CMDCR	Command Control register	7-98
0x4400_0104	PRSR	Panel Read Status register	7-105
0x4400_0108–0x4400_010C	—	reserved	
0x4400_0110	FBR5	DMA Channel 5 Frame Branch register	7-103
0x4400_0114	FBR6	DMA Channel 6 Frame Branch register	7-103
0x4400_0118–0x4400_01FF	—	reserved	
0x4400_0200	FDADR0	DMA Channel 0 Frame Descriptor Address register	7-102
0x4400_0204	FSADR0	DMA Channel 0 Frame Source Address register	7-119
0x4400_0208	FIDR0	DMA Channel 0 Frame ID register	7-119
0x4400_020C	LDCMD0	LCD DMA Channel 0 Command register	7-120
0x4400_0210	FDADR1	DMA Channel 1 Frame Descriptor Address register	7-102
0x4400_0214	FSADR1	DMA Channel 1 Frame Source Address register	7-119
0x4400_0218	FIDR1	DMA Channel 1 Frame ID register	7-119
0x4400_021C	LDCMD1	LCD DMA Channel 1 Command register	7-120
0x4400_0220	FDADR2	DMA Channel 2 Frame Descriptor Address register	7-102
0x4400_0224	FSADR2	DMA Channel 2 Frame Source Address register	7-119
0x4400_0228	FIDR2	DMA Channel 2 Frame ID register	7-119
0x4400_022C	LDCMD2	LCD DMA Channel 2 Command register	7-120
0x4400_0230	FDADR3	DMA Channel 3 Frame Descriptor Address register	7-102
0x4400_0234	FSADR3	DMA Channel 3 Frame Source Address register	7-119
0x4400_0238	FIDR3	DMA Channel 3 Frame ID register	7-119
0x4400_023C	LDCMD3	LCD DMA Channel 3 Command register	7-120
0x4400_0240	FDADR4	DMA Channel 4 Frame Descriptor Address register	7-102
0x4400_0244	FSADR4	DMA Channel 4 Frame Source Address register	7-119
0x4400_0248	FIDR4	DMA Channel 4 Frame ID register	7-119
0x4400_024C	LDCMD4	LCD DMA Channel 4 Command register	7-120
0x4400_0250	FDADR5	DMA Channel 5 Frame Descriptor Address register	7-102
0x4400_0254	FSADR5	DMA Channel 5 Frame Source Address register	7-119
0x4400_0258	FIDR5	DMA Channel 5 Frame ID register	7-119
0x4400_025C	LDCMD5	LCD DMA Channel 5 Command register	7-120
0x4400_0260	FDADR6	DMA Channel 6 Frame Descriptor Address register	7-102
0x4400_0264	FSADR6	DMA Channel 6 Frame Source Address register	7-119
0x4400_0268	FIDR6	DMA Channel 6 Frame ID register	7-119
0x4400_026C	LDCMD6	LCD DMA Channel 6 Command register	7-120
0x4400_0270–0x47FF_FFFC	—	reserved	
0x4800_0054	LCDBSCNTR	LCD Buffer Strength Control register	7-104

SSP Serial Ports 8

This chapter describes the signal definitions and operation of the three Synchronous Serial Protocol (SSP) serial ports: SSP1, SSP2, and SSP3 included in the PXA27x processor. All three SSP ports are mostly identical in operation, but differ as follows:

- External port connections
- Memory-map base location
- No external clock and external clock enable for SSP3

8.1 Overview

The SSP ports are a synchronous serial interfaces that connect to a variety of external analog-to-digital (A/D) converters, audio and telecommunication Codecs, and many other devices that use serial protocols for data transfer. The SSP ports provide support for the following protocols:

- Texas Instruments (TI) Synchronous Serial Protocol
- Motorola Serial Peripheral Interface (SPI) protocol
- National Semiconductor Microwire
- Programmable Serial Protocol (PSP)

The SSP ports operate as full-duplex devices for the TI Synchronous Serial Protocol, SPI, and PSP protocols and as a half-duplex device for the Microwire protocol.

The FIFOs can be loaded or emptied by the CPU using programmed I/O or by DMA burst transfers.

8.2 Features

- Supports the TI Synchronous Serial Protocol, the Motorola SPI protocol, National Semiconductor Microwire, and a Programmable Serial Protocol (PSP)
- One transmit FIFO and one receive FIFO, each 16 samples deep by 32-bits wide
- Sample sizes from four to 32-bits
- Bit-rates from 6.3 Kbps (minimum) to 13 Mbps (maximum)
- Master-mode and slave-mode operation
- Receive-without-transmit operation
- Network mode with up to eight time slots and independent transmit/receive in any/all/none of the time slots—available only with TI Synchronous Serial Protocol and Programmable Serial Protocol (PSP) formats
- Audio clock control to provide a 4x output clock and support for selection of most standard audio Codec frequencies

8.3 Signal Descriptions

Table 8-2 lists the external signals between the SSP serial ports and external devices. If any port is unused, its pins are available for GPIO use. See Chapter 24, "General-Purpose I/O Controller" for details on configuring pin direction and Chapter 25, "Interrupt Controller" for interrupt capabilities.

Table 8-1. SSP Serial Port I/O Signal Descriptions (Sheet 1 of 2)

Name	Type	Description
SSPSCLK	I/O	Serial bit-clock to control the timing of a transfer. SSPSCLK is generated internally (master mode) or is supplied externally (slave mode) as indicated by SSCR1_1[SCLKDIR] as defined in Table 8-7.
SSPSYSCLK	Output	SSPSYSCLK is SSPSCLK x 4 when using the audio clock PLL. Select (SSACD_1[ACPS]) and audio clock divider (SSACD_1[ACDS]) as defined in Table 8-16.
SSPSFRM	I/O	Serial frame signal that indicates the beginning and the end of a serialized data word. SSPSFRM is generated internally (master mode) or is supplied externally (slave mode) as indicated by SSCR1_1[SFRMDIR] and defined in Table 8-7.
SSPTXD	Output	Transmit data (serial data out) serialized data line.†
SSPRXD	Input	Receive data (serial data in) serialized data line.†
SSPEXTCLK	Input	External clock that can be selected to replace the internal 13-MHz clock. SSPEXTCLK is used only when SSCR0_1[ECS] is set (Table 8-6) and SSCR0_1[NCS], SSCR0_1[ACS], and SSCR1_1[SCLKDIR] are cleared (Table 8-7). The maximum allowable frequency for the external clock is 13 MHz. SSPEXTCLK is multiplexed with the SSPSCLKEN alternate function (refer to Chapter 24, "General-Purpose I/O Controller").
SSPSCLKEN	Input	Asynchronous external enable for SSPSCLK. SSPSCLKEN is recognized only when SSCR0_1[ECS] is cleared and when the port is the master (SSCR1_1[SCLKDIR] is cleared). When high, SSPSCLK is enabled; when low, SSPSCLK is disabled. SSPSCLKENx is multiplexed with the SSPEXTCLK alternate function (refer to Chapter 24, "General-Purpose I/O Controller").
SSPSCLK2	I/O	Serial bit-clock to control the timing of a transfer. SSPSCLK2 is generated internally (master mode) or is supplied externally (slave mode) as indicated by SSCR1_2[SCLKDIR] as defined in Table 8-7.
SSPSYSCLK2	Output	SSPSYSCLK2 is SSPSCLK2 x 4 when using the audio clock PLL. Select (SSACD_2[ACPS]) and audio clock divider (SSACD_2[ACDS]) as defined in Table 8-16).
SSPSFRM2	I/O	Serial frame signal that indicates the beginning and the end of a serialized data word. SSPSFRM2 is generated internally (master mode) or is supplied externally (slave mode) as indicated by SSCR1_2[SFRMDIR] as defined in Table 8-7.
SSPTXD2	Output	Transmit data (serial data out) serialized data line.†
SSPRXD2	Input	Receive data (serial data in) serialized data line.†
SSPEXTCLK2	Input	External clock that can be selected to replace the internal 13-MHz clock. SSPEXTCLK2 is used only when SSCR0_2[ECS] is set (Table 8-6) and when SSCR0_2[NCS], SSCR0_2[ACS], and SSCR1_2[SCLKDIR] are cleared (Table 8-7). SSPEXTCLK2 is multiplexed with the SSPSCLKEN2 alternate function (refer to Chapter 24, "General-Purpose I/O Controller").

Table 8-1. SSP Serial Port I/O Signal Descriptions (Sheet 2 of 2)

Name	Type	Description
SSPSCLKEN2	Input	Asynchronous external enable for SSPSCLK2. SSPSCLKEN2 is recognized only when SSCR0_2[ECS] is cleared and when the port is the master (SSCR1_2[SCLKDIR] is cleared). When high, SSPSCLK2 is enabled; when low, SSPSCLK2 is disabled. SSPSCLKEN2 is multiplexed with the SSPEXTCLK2 alternate function (refer to Chapter 24, "General-Purpose I/O Controller").
SSPSCLK3	Inout	Serial bit-clock to control the timing of a transfer. SSPSCLK3 is generated internally (master mode) or is supplied externally (slave mode) as indicated by SSPCR1_3[SCLKDIR] as defined in Table 8-7.
SSPSYSCLK3	Output	SSPSYSCLK3 is SSPSCLK3 x 4 when using the audio clock PLL. Select (SSACD_3[ACPS]) and audio clock divider (SSACD_3[ACDS]) as defined in Table 8-16).
SSPSFRM3	Inout	Serial frame signal that indicates the beginning and the end of a serialized data word. SSPSFRM3 is generated internally (master mode) or is supplied externally (slave mode) as indicated by SSPCR1_3[SFRMDIR] as defined in Table 8-7.
SSPTXD3	Output	Transmit data (serial data out) serialized data line.†
SSPRXD3	Input	Receive data (serial data in) serialized data line.†
CLK_EXT	Input	The network clock (CLK_EXT) is an external clock that can replace the internal 13-MHz clock. Use CLK_EXT when SSCR0_x[NCS] is set (Table 8-6) and SSCR1_x[SCLKDIR] is cleared (Table 8-7). CLK_EXT can be used by multiple SSPs.
† Sample length is a function of the selected serial data sample size set by SSCR0_x[EDSS] and the SSCR0_x[DSS], as defined in Table 8-6.		

Note: SSP3 does not have an external clock input or an external clock-enable input.

8.4 Operation

The SSP port controller transfers serial data between the PXA27x processor and an external device through FIFOs in the SSP ports. The CPU initiates the transfers using programmed I/O (PIO), or DMA bursts are used. Separate transmit and receive FIFOs and serial data paths permit simultaneous transfers (in both directions) to and from the external device, depending on the protocols chosen.

Programmed I/O transfers data directly between the CPU and the SSP Data register (SSDR_x). DMA transfers data between memory and the SSP Data register (SSDR_x). Data written to the SSDR_x (by either the CPU or DMA) is automatically transferred to the transmit FIFO. When reading SSDR_x (by either the CPU or DMA), the "oldest" data in the receive FIFO is automatically transferred to the SSDR_x. DMA descriptor programming guidelines are located in Section 5.4.4, "Programming Tips" on page 5-15.

8.4.1 Processor and DMA FIFO Access

The CPU or DMA accesses data through the SSP port transmit and receive FIFOs. Programmed I/O takes the form of a CPU access, transferring one FIFO entry per access. CPU accesses are normally triggered off of an SSSR_x interrupt and are always 32-bits wide. CPU right-justified

writes to the FIFOs ignore bits beyond the programmed FIFO data width (SSCR0_x[EDSS] and SSCR0_x[DSS] values); and CPU reads return zeroes in the MSBs down to the programmed data width.

The FIFOs can also be accessed by DMA bursts (in multiples of one, two or four bytes) depending on the SSCR0_x[EDSS] value. When SSCR0_x[EDSS] is set, DMA bursts must be in multiples of four bytes (the DMA must have the SSP port configured as a 32-bit peripheral).When SSCR0_x[EDSS] is cleared, DMA bursts must be in multiples of one or two bytes (the DMA DCMD[WIDTH] register must be at least the SSP data size programmed into the SSCR0_x[EDSS] and SSCR0_x[DSS]. The FIFO is seen as one 32-bit location by the processor. For writes, the SSP port takes the data from the transmit FIFO, serializes it, and sends it over the serial wire (SSPTXDx) to the external device. Receive data from the external device (on SSPRXDx) is converted to parallel words and stored right-justified with zeroes packed on the left in the receive FIFO.

CPU or DMA data written to the SSP Data register (SSDR_x) is automatically transferred into the transmit FIFO. CPU or DMA reads from SSDR_x contain the “oldest” data sample that is automatically transferred from the receive FIFO. From a memory-map perspective, both reads and writes are at the same address. The FIFOs are 16 samples deep by 32 bits wide. Each FIFO location contains one SSP data sample (four to 32 bits).

When exceeded, a programmable transmit-FIFO trigger threshold generates an interrupt or DMA service request that, if SSCR1_x[TIE] or SSCR1_x[TSRE] are enabled, signal the CPU or DMA, respectively, to move data to the SSDR_x. Similarly, a programmable receive FIFO trigger threshold generates an interrupt or DMA service request that, if SSCR1_x[RIE] or SSCR1_x[RSRE] are enabled, signal the CPU or DMA, respectively, to read data from SSDR_x.

Note: (1) Do not set the value of SSCR1_x[RFT] too high for the system; otherwise, the receive FIFO can overrun because of the bus latencies caused by other internal and external peripherals. This is especially important when using interrupts and polled modes that require a longer time to service.

(2) Do not set the value of SSCR1_x[TFT] too low for the system; otherwise, the transmit FIFO can underrun because of the bus latencies caused by other internal and external peripherals. This is especially important when using interrupts and polled modes that require a longer time to service.

8.4.2 Trailing Bytes in the Receive FIFO

When the number of samples in the receive FIFO is less than the trigger threshold and no additional data is received, the remaining bytes are called *trailing bytes*. Trailing bytes can be handled by either the DMA or the CPU, (see SSCR1_x[TRAIL]). Trailing bytes are identified by a time-out mechanism and the existence of data within the receive FIFO.

8.4.2.1 Time-Out

A time-out condition exists when the receive FIFO is idle for the period of time defined by the Time-Out register (SSTO_x). When a time-out occurs, the receiver time-out interrupt (SSSR_x[TINT]) is set. If the time-out interrupt is enabled (SSCR1_x[TINTE] set), a time-out interrupt signals the CPU that a time-out condition has occurred. The time-out timer is reset after a new sample enters the receive FIFO or after the CPU reads the receive FIFO. Once SSSR_x[TINT] is set, it must be cleared by writing 0b1 to it. If the time-out interrupt is enabled, clearing TINT also causes the time-out interrupt to be de-asserted.

8.4.2.2 Peripheral Trailing Byte Interrupt

It is possible for the DMA to reach the end of its descriptor chain while removing trailing bytes. When this happens, the CPU is forced to take over because the DMA can no longer service the SSP port request until a new chain is linked. When the DMA reaches the end of its descriptor chain, the SSP port performs the following:

- Sets the peripheral trailing-byte interrupt bit (SSSR_x[PINT]) if it is enabled when SSCR1_x[PINTE] is set
- Signals the CPU that a peripheral trailing-byte interrupt condition has occurred
- Sets the SSSR_x[EOC] status bit. If more data is received after EOC is set (and EOC has not been cleared by software), SSSR_x[PINT] is set.

Once SSSR_x[PINT] is set, it must be cleared by writing 0b1 to it. Clearing SSSR_x[PINT] de-asserts the peripheral trailing-byte interrupt.

Always correctly handle the possibility of trailing bytes being present. Remove the remaining bytes using CPU I/O as described in the processor-based method below or by reprogramming a new descriptor chain and restarting the DMA. See Chapter 5, “DMA Controller” for details of descriptor programming and “end-of-chain” events.

8.4.2.3 Removing Trailing Bytes

Processor Based (SSCR1_x[TRAIL] cleared):

This is the default method. In this case, no receive DMA service request is generated. To read out the trailing bytes, software must wait for the time-out interrupt and then read all remaining entries as indicated by SSSR_x[RFL] and SSSR_x[RNE].

Note: The time-out interrupt must be enabled by setting SSCR1_x[TINTE].

DMA Based (SSCR1_x[TRAIL] set):

When the DMA must handle trailing bytes, a DMA service request is automatically issued for the remaining number of samples left in the receive buffer. The DMA empties the contents of the receive buffer unless the DMA reaches the end of its descriptor chain (refer to Section 8.4.2.2). If a time out occurs, the processor is interrupted by a time-out interrupt. Enable the time-out interrupt by setting SSCR1_x[TINTE]. When handling trailing bytes with DMA, if a time out occurs and the receive FIFO is empty, an EOR is sent to the DMA controller.

Note: If an EOC occurs at the time that the last sample is read from the Receive Data register (the DMA descriptor chain was exactly long enough, but the time-out counter is still running—a time out has not occurred and the SSTO register is not zero), then when the time-out does occur, the SSP generates a DMA request, which causes a RAS interrupt from the DMA controller. When the RAS interrupt occurs, software must reprogram the DMA registers and re-enable the channel for the SSP to send its EOR to the DMA controller.

8.4.3 Frame Counter

The SSP sends a start-of-frame signal to the OS timer to increment a frame counter (see Chapter 22, “Operating System Timers” for details). In normal mode (SSCR0_x[MOD] = 0), the start-of-frame signal is asserted on every sample that is received; in network mode (SSCR0_x[MOD] = 1), the start-of-frame signal is asserted once every time that SSPSFRMx is asserted.

8.4.4 Data Formats

Four pins transfer data between the PXA27x processor and external Codecs, modems, or other compatible serial devices. Although four serial-data formats exist, each has the same basic structure and in all cases the pins are used as follows:

- SSPSCLKx—Defines the bit rate at which serial data is driven into, and sampled from, the SSP port.
- SSPSFRMx—Defines the boundaries of a basic data unit, comprised of multiple serial bits.
- SSPTXDx—The serial data path for transmitted data from the processor to the peripheral.
- SSPRXDx—The serial data path for received data from the peripheral to the processor.

A data frame can contain from four to 32-bits, depending on the selected protocol. Serial data is transmitted most significant bit first. Four protocols are supported: Texas Instruments (TI) Synchronous Serial Protocol, Motorola Serial Peripheral Interface (SPI) protocol, National Semiconductor Microwire, and a Programmable Serial Protocol (PSP).

The SSPSFRMx function and use varies between each protocol.

- For the TI Synchronous Serial Protocol, SSPSFRMx is pulsed high for one (serial) data period at the start of each frame. Master and slave modes are supported. TI Synchronous Serial Protocol is a full-duplex protocol.
- For the Motorola SPI protocol, SSPSFRMx functions as a chip select to enable the external device (target of the transfer) and is held active-low during the data transfer (during continuous transfers, the SSPSFRMx signal is held low. Master and slave modes are supported. This is a full-duplex protocol.
- For the National Semiconductor Microwire protocol, SSPSFRMx functions as a chip select to enable the external device (target of the transfer) and is held active-low during the data transfer. Slave mode is not supported. SSPSFRMx is also held low during continuous transfers. This is a half-duplex protocol.
- For the PSP, SSPSFRMx is programmable in direction, delay, polarity, and width. Master and slave modes are supported. PSP can be programmed to be either full or half duplex.

The function and use of SSPSCLKx varies between each protocol.

- For the TI Synchronous Serial Protocol, data sources switch transmit data on the rising edge of SSPSCLKx and sample receive data on the falling edge. Master and slave modes are supported. When SSPSCLKx is provided by the SSP port, it only toggles during active transfers (not continuously) unless the SSPSCLKENx, or SSCR1_x[ECRA] or SSCR1_x[ECRB] bits, are used. When SSPSCLKx is provided by another device, it can be either continuous or driven only during active transfers.
- For the SPI protocol, select which edge of SSPSCLKx to use for switching transmit data and for sampling receive data. In addition, users can move the phase of SSPSCLKx, shifting its

active state one-half cycle earlier or later at the start and end of a frame. Master and slave modes are supported, and SSPSCLKx only toggles during active transfers (not continuously).

- For Microwire, both data sources switch transmit data (change to the next bit) on the falling edge of SSPSCLKx and sample receive data on the rising edge. Slave mode is not supported.
- For the PSP protocol, the configuration of which edge of SSPSCLKx is used for switching transmit data and the edge for sampling receive data is programmable. In addition, the idle state for SSPSCLKx can be programmed and the number of active clocks that precede and follow the data transmission can be programmed. Master and slave modes are supported. When SSPSCLKx is provided by the SSP port, it only toggles during active transfers (not continuously) unless the SSPSCLKENx, or SSCR1_x[ECRA] or SSCR1_x[ECRB] bits, are used. When SSPSCLKx is provided by another device, it can be either continuous or only driven during active transfers, but some restrictions apply (see Section 8.4.4.4).

Microwire uses a half-duplex, master-slave messaging protocol. At the start of a frame, the controller transmits a one-or two-byte control message to the peripheral; no data is sent by the peripheral. The peripheral interprets the message and if the message is a read request, the peripheral, responds with the requested data, one clock after the last bit of the request message. Return data—part of the same frame—can be from four to 16-bits in length. The total frame length is 13 to 33 bits. SSPSCLKx is active during the entire frame.

Note: SSPSCLKx, if provided by the SSP port, toggles only while an active data transfer is underway, unless receive-without-transmit mode is enabled (by setting SSCR1_x[RWOT]) *and* the frame format is not Microwire. If RWOT mode is enabled and the frame format is not Microwire, SSPSCLKx toggles regardless of whether transmit data exists within the transmit FIFO. SSPSCLKx toggles continuously if the SSP port is in network mode, or if <SSPSCLKENx> is active, or the SSCR1_x[ECRA] or SSCR1_x[ECRB] bits, are enabled. At other times, <SSPSCLKx> holds in an inactive (idle) state as defined by the protocol.

8.4.4.1 TI Synchronous Serial Protocol

When outgoing data in the SSP port controller is ready to transmit, SSPSFRMx asserts for one clock period. On the following clock, data to be transmitted is driven on SSPTXDx one bit at a time, the most significant bit first. For receive data, the peripheral similarly drives data on the SSPRXDx pin. The word length can be from four to 32-bits. All output transitions occur on the rising edge of SSPSCLKx while data sampling occurs on the falling edge. At the end of the transfer, the SSPTXDx signal either retains the value of the last bit sent (LSB) (see Figure 8-1 and Figure 8-2) or becomes high impedance (see Figure 8-13 and Figure 8-14).

Figure 8-1 shows the TI Synchronous Serial Protocol for a single transmitted frame. Figure 8-2 shows the TI Synchronous Serial Protocol for when back-to-back frames are transmitted. Once the transmit FIFO contains data, SSPSFRMx is pulsed high for one SSPSCLKx period and the value to be transmitted is transferred from the transmit FIFO to the Transmit Logic Serial Shift register. On the next rising edge of SSPSCLKx, the most significant bit of the four to 32-bit data frame is shifted to the SSPTXDx pin. Likewise, the MSB of the received data is shifted onto the SSPRXDx pin by the off-chip serial slave device. Both the SSP port and the off-chip serial slave device then latch each data bit into the serial shifter on the falling edge of each SSPSCLKx. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SSPSCLKx after the last bit has been latched.

For back-to-back transfers, the start of one frame is the completion of the previous frame. The MSB of one transfer immediately follows the LSB of the preceding with no “dead” time between them. When the SSP port is a master to the SSPSFRMx and a slave to SSPSCLKx, at least three extra clocks are needed at the beginning and end of each block of transfers to synchronize internal control signals (a block of transfers is a group of back-to-back continuous transfers).

Note: When configured as either master or slave to SSPSCLKx or SSPSFRMx, the SSP port continues to drive SSPTXDx until the last bit of data is sent (the LSB) or the SSPTXDx line becomes high impedance (see Figure 8-13 and Figure 8-14). If SSCR0_x[SSE] is cleared, the SSPTXDx line goes low. SSPSP_x[EDTS] has no effect when in SSP mode. SSPRXDx is undefined before the MSB is sent and after the LSB is sent. SSPRXDx must not float.

Figure 8-1. Texas Instruments Synchronous Serial Frame Protocol (Single Transfers)

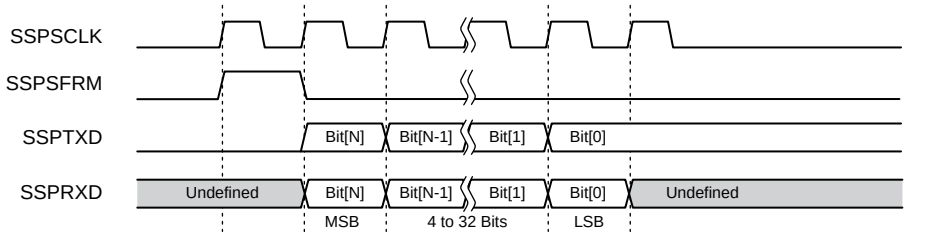
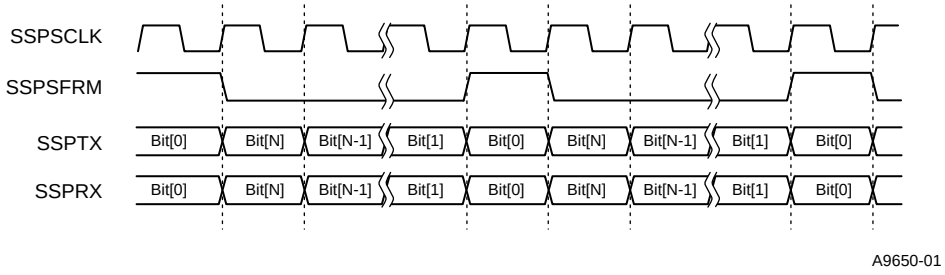
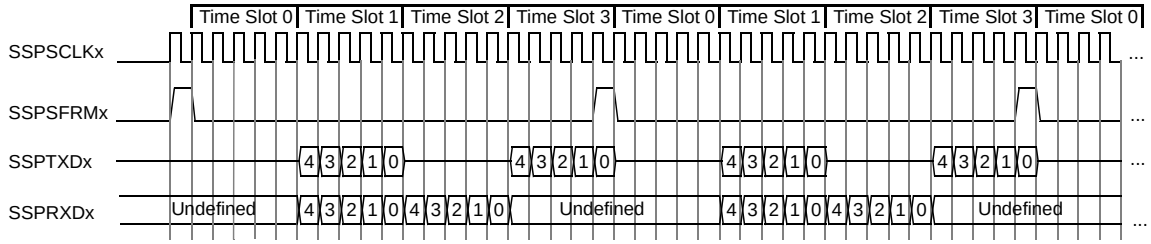


Figure 8-2. Texas Instruments Synchronous Serial Frame Protocol (Multiple Transfers)



The TI SSP supports network mode as described in Table 8-6. Figure 8-3 shows the SSP in network mode with five bits of data per sample. The TXD tristate-enable (SSCR1_x[TTE]) bit and the TXD tristate-enable-on-last-phase (SSCR1_x[TTELP] bit are both set. The SSP is the master of both SSPSCLKx and SSPSFRMx. The SSP has been programmed for four time slots (SSCR0_x[FRDC] = 0b011), the TX Time Slot Active register (SSTSA_x[TTSA]) is programmed to 0x0000 000A, and the RX Time Slot Active register (SSRSA_x[RTSA]) is programmed to 0x0000 0006.

Figure 8-3. TI SSP Network Mode Example (Four Time Slots)



8.4.4.2 Motorola SPI Protocol

The SPI protocol has four possible sub-modes, depending on the SSPSCLKx edges selected for driving data and sampling received data, and on the selection of the phase mode of SSPSCLKx (see Section 8.4.4.2.1 for complete descriptions of each mode).

Note: The following description applies only when SPH = 0 and SPO = 0. Other combinations of SPH and SPO result in different polarities and timings (see Figure 8-5 and Figure 8-6).

When the SSP port is disabled or in idle mode, SSPSCLKx and SSPTXDx are low and SSPSFRMx is high. When transmit data is available to send, SSPSFRMx goes low (one clock period before the first rising edge of SSPSCLKx) and stays low for the remainder of the frame. The most significant bit of the serial data is driven onto SSPTXDx one half-cycle later. Halfway into the first bit period, SSPSCLKx asserts high and continues toggling for the remaining data bits. Data transitions on the falling edge of SSPSCLKx. Four to 32 bits can be transferred per frame.

With the assertion of SSPSFRMx, receive data is simultaneously driven from the peripheral on SSPRXDx, MSB first. Data transitions on the SSPSCLKx falling edge and are sampled by the controller on the SSPSCLKx rising edge. At the end of the frame, SSPSFRMx is de-asserted high one clock period (one half clock cycle after the last falling edge of SSPSCLKx) after the last bit latched at its destination and the completed incoming word is shifted into the incoming FIFO. The peripheral can drive SSPRXDx to a high-impedance state after sending the last bit of the frame. SSPTXDx retains the last value transmitted when the controller goes into idle mode, unless the SSP port is disabled or reset (which forces SSPTXDx low).

For back-to-back transfers, frames start and complete similar to single transfers, except SSPSFRMx does not de-assert between words. Both transmitter and receiver are configured for the word length and internally track the start and end of frames. There are no dead bits; the least significant bit of one frame is followed immediately by the most significant bit of the next.

When using the SPI protocol, the SSP port can be either a master or a slave device. However, the clock and frame direction must be the same. For example, the SSCR1_x[SCLKDIR] and SSCR0_x[SFRMDIR] must both be set or both be cleared.

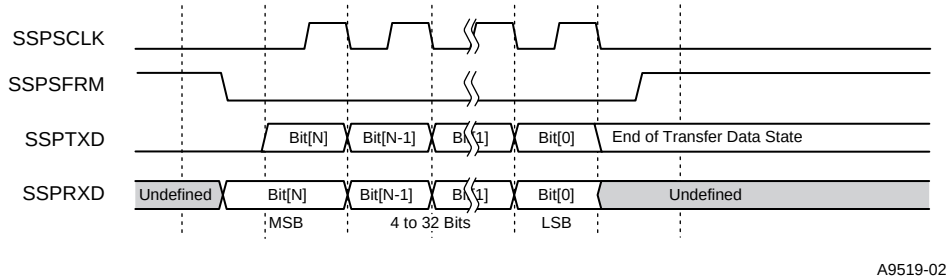
When configured as either master or slave and SSPSP_x[ETDS] is set, the SSP port continues to drive SSPTXDx with the last bit of data sent (the LSB). If SSPSP_x[ETDS] is cleared, SSPTXDx goes low after transmitting the last data bit. The state of SSPRXDx is undefined before the MSB and after the LSB is received. SSPRXDx must not float. When the SSP port is configured as a master and SSCR1_x[TTE] is set, SSPSP_x[ETDS] is ignored and SSPTXDx becomes high impedance between active transfers (see Figure 8-15).

Note: The input clock to the SSP port must not be active when SSPSFRMx is de-asserted.

Note: When the SSP port is slave to clock and frame, SSCR1_x[SCFR] must be set.

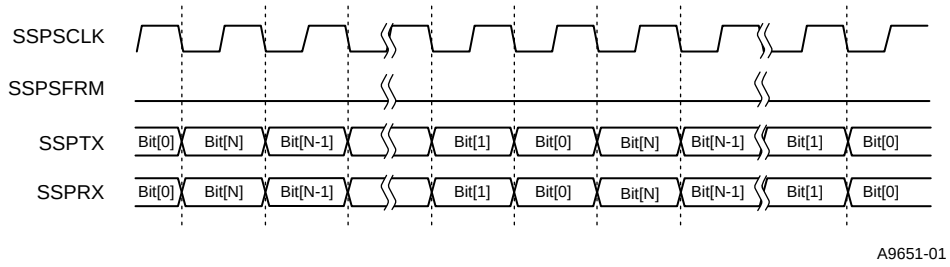
Figure 8-4 shows one of the four possible configurations for the Motorola SPI frame protocol for a single transmitted frame. Figure 8-5 shows when back-to-back frames are transmitted for the Motorola SPI frame protocol.

Figure 8-4. Motorola SPI Frame Protocol (Single Transfers)



Note: The phase and polarity of SSPSCLKx can be configured for four different modes. This example shows just one of those modes (SSCR1_x[SPO] and SSCR1_x[SPH] cleared). Other settings for SPO and SPH result in different polarities and timing.

Figure 8-5. Motorola SPI Frame Protocol (Multiple Transfers)



8.4.4.2.1 Serial Clock Phase (SPH)

The phase relationship between SSPSCLKx and SSPSFRMx when the Motorola SPI protocol is selected is controlled by SSCR1_x[SPH].

The combination of the SSCR1_x[SPO] and SSCR1_x[SPH] settings determine when SSPSCLKx is active during the assertion of SSPSFRMx and which SSPSCLKx edge transmits and receives data on SSPTXDx and SSPRXDx.

When SPH is cleared, SSPSCLKx remains in its inactive (idle) state (as determined by SSCR1_x[SPO]) for one full cycle after SSPSFRMx is asserted low at the beginning of a frame. SSPSCLKx continues to toggle for the rest of the frame. It is then held in its inactive state for one-half of an SSPSCLKx period before SSPSFRMx is de-asserted high at the end of the frame.

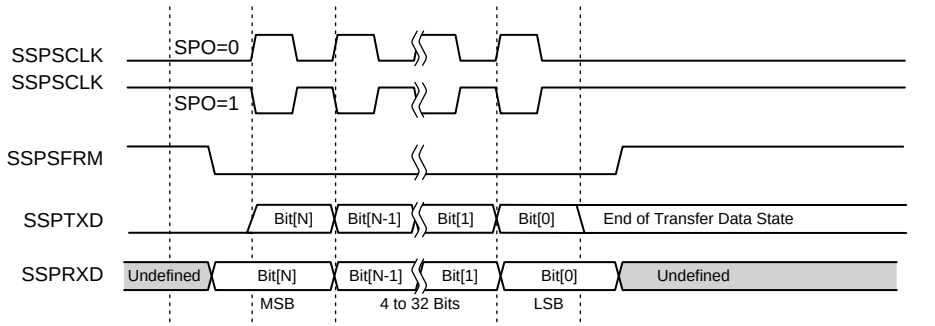
When SPH is set, SSPSCLKx remains in its inactive or idle state (as determined by SSCR1_x[SPO]) for one-half cycle after SSPSFRMx is asserted low at the beginning of a frame. SSPSCLKx continues to toggle for the remainder of the frame and is then held in its inactive state for one full SSPSCLKx period before SSPSFRMx is de-asserted high at the end of the frame.

When programming SSCR1_x[SPO] and SSCR1_x[SPH] to the same value (both set or both cleared), transmit data is driven on the falling edge of SSPSCLKx and receive data is latched on the rising edge of SSPSCLKx. When programming SSCR1_x[SPO] and SSCR1_x[SPH] to opposite values (one set and the other cleared), transmit data is driven on the rising edge of SSPSCLKx and receive data is latched on the falling edge of SSPSCLKx.

Note: SSCR1_x[SPH] is ignored for all data frame formats except for the Motorola SPI protocol.

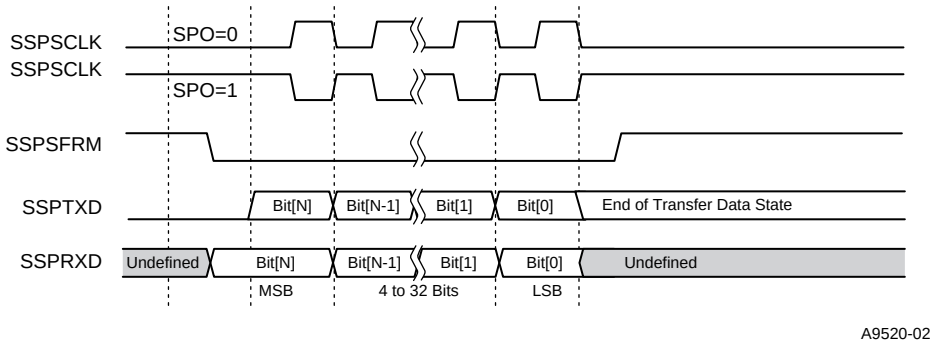
Figure 8-6 and Figure 8-7 show the timing for all four programming combinations of SSCR1_x[SPO] and SSCR1_x[SPH]. SSCR1_x[SPO] inverts the polarity of SSPSCLKx and SSCR1_x[SPH] determines the phase relationship between SSPSCLKx and SSPSFRMx, shifting SSPSCLKx one-half phase to the left or right during the assertion of SSPSFRMx.

Figure 8-6. Motorola SPI Frame Protocols for SPO and SPH Programming (SPH Set)



Note: When in Motorola SPI format, if the SSP port is the master and SSPSP_x[ETDS] is cleared, the end-of-transfer data state for SSPTXDx is low. If the SSP port is the master and SSPSP_x[ETDS] is set, the end-of-transfer data state for SSPTXDx remains at the last bit transmitted (LSB). If the SSP port is the slave, then SSPSP_x[ETDS] is undefined. SSPRXDx is undefined before the frame is active and after the LSB is received. SSPRXDx must not float. When the SSP port is configured as a master and SSCR1_x[TTE] is set, SSPSP_x[ETDS] is ignored and SSPTXDx becomes high impedance between active transfers (see Figure 8-15).

Figure 8-7. Motorola SPI Frame Protocols for SPO and SPH Programming (SPH Cleared)



Note: When in Motorola SPI format, if the SSP port is the master and SSPSP_x[ETDS] is cleared, the end-of-transfer data state for SSPTXDx is low. If the SSP port is the master and SSPSP_x[ETDS] is set, the end-of-transfer data state for SSPTXDx remains at the last bit transmitted (LSB). If the SSP port is the slave, then the SSPSP_x[ETDS] is undefined. SSPRXDx is undefined before the frame is active and after the LSB is received. SSPRXD must not float. When the SSP port is configured as a master and SSCR1_x[TTE] is set, SSPSP_x[ETDS] is ignored and SSPTXDx becomes high impedance between active transfers (see Figure 8-15).

8.4.4.3 Microwire Protocol

The Microwire protocol is similar to SPI, except transmission is half-duplex instead of full-duplex and it uses master-slave message passing. While in the idle state or when the SSP port is disabled, SSPSCLKx and SSPTXDx are low and SSPSFRMx is high.

Each serial transmission begins with SSPSFRMx asserting low, followed by an eight or 16-bit command word sent from the controller to the peripheral on SSPTXDx. The command word data size is selected by the Microwire transmit SSCR1_x[MWDS] data size bit. SSPRXDx is controlled by the peripheral and remains in a high-impedance state. SSPSCLKx asserts high (rising edge) midway into the command's most significant bit and continues toggling at the configured SSPSCLKx bit rate.

One bit period after the last command bit, the peripheral returns the serial-data-requested most significant bit first on SSPRXDx. Data transitions on the falling edge of SSPSCLKx and is sampled on the rising edge. The last falling edge of SSPSCLKx coincides with the end of the last data bit on SSPRXDx and SSPSCLKx remains low after that (if it is the only word or the last word of the transfer). SSPSFRMx de-asserts high one-half clock period later.

The start and end of a series of back-to-back transfers are like those of a single transfer; however, SSPSFRMx remains asserted (low) throughout the transfer. The end of a data word on SSPRXDx is followed immediately by the start of the next command byte on SSPTXDx with no dead time. If SSCR1_x[TTE] is set (selecting SSPTXDx to be high impedance between active transfers), then a three-wire Microwire mode can be supported where SSPTXDx and SSPRXDx are externally tied together (shorted).

When using the Microwire protocol, the SSP port can function only as a master (frame and clock are outputs). Therefore, both SSCR1_x[SCLKDIR] and SSCR0_x[SFRMDIR] must both be cleared. Figure 8-8 shows the National Semiconductor Microwire frame protocol with 8- or 16-bit

command words for a single transmitted frame. Figure 8-9 shows the National Semiconductor Microwire frame protocol with eight-bit command words when back-to-back frames are transmitted.

Figure 8-8. National Semiconductor Microwire Frame Protocol (Single Transfer)

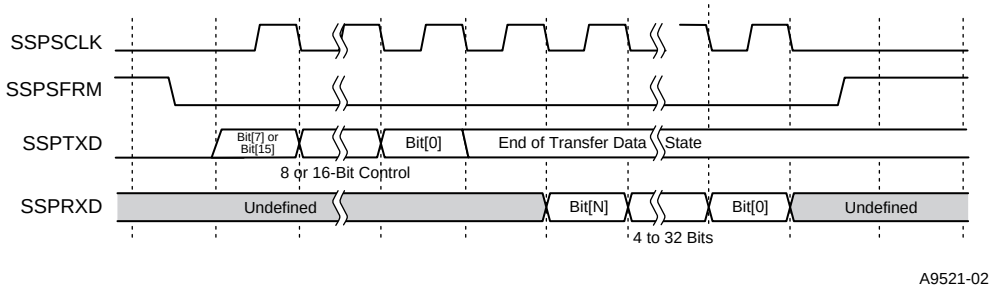
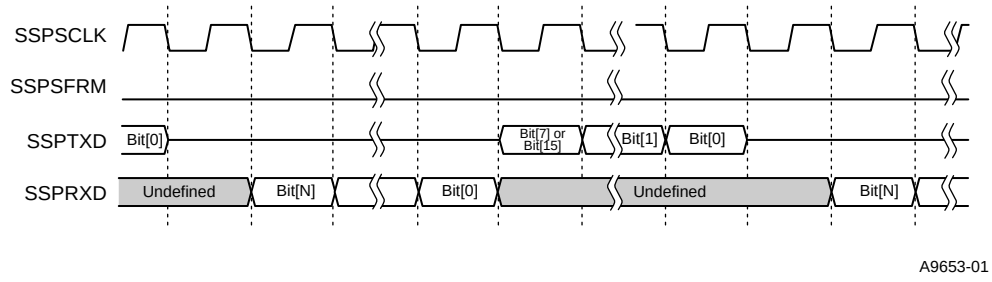


Figure 8-9. National Semiconductor Microwire Frame Protocol (Multiple Transfers)



Note: (1) Microwire format is normally used with SSPSP_x[ETDS] cleared so that the end-of-transfer data state for SSPTXDx is low. If SSPSP_x[ETDS] is set then SSPTXDx remains at the level of the last transmitted bit (LSB). When SSCR1_x[TTE] is set, SSPSPx[ETDS] is ignored and SSPTXDx becomes high impedance between active transfers (see Figure 8-16).

(2) The state of SSPRXD is undefined before the MSB and after the LSB is transmitted. This pin must not float.

8.4.4.4 Programmable Serial Protocol (PSP)

The PSP provides programmability for several parameters that determine the transfer timings between data.

There are four possible serial clock sub-modes, depending on which SSPSCLKx edges are selected for driving data and sampling received data, and the selection of the idle state of SSPSCLKx.

For the PSP, the idle and disable modes of SSPTXDx, SSPSCLKx, and SSPSFRMx are programmable using SSPSP_x[ETDS], SSPSP_x[SCMODE] and SSPSP_x[SFRMP]. When transmit data is ready, SSPSCLKx remains in its idle state for the number of SSPSCLKx periods programmed within the SSPSP_x[STRTDLY] start delay field. SSPSCLKx then starts toggling,

SSPTXDx remains in the idle state for the number of cycles programmed within the SSPSP_x[DMYSTRT] dummy start field. SSPSFRMx asserts after the number of clocks programmed in the SSPSP_x[SFRMDLY] frame delay field. SSPSFRMx remains asserted for the number of half-clocks programmed within the SSPSP_x[SFRMWDTH] frame width field. Four to 32-bits can be transferred per frame. Once the LSB transfers, SSPSCLKx continues toggling based on the SSPSP_x[DMYSTOP] dummy stop field. SSPTXDx either retains the last value transmitted or is forced to zero, depending on the value programmed within the SSPSP_x[ETDS] end-of-transfer data state when the controller goes into idle mode, unless the SSP port is disabled or reset (which forces SSPTXDx low). When SSCR1_x[TTE] is set, SSPSP_x[ETDS] is ignored and SSPTXDx becomes high impedance between active transfers (see Figure 8-17 and Figure 8-18).

With the assertion of SSPSFRMx, receive data is simultaneously driven from the peripheral on SSPRXDx, MSB first. Data transitions on SSPSCLKx are based on the serial-clock mode selected and are sampled by the controller on the opposite edge. When the SSP port is a master to SSPSFRMx and a slave to SSPSCLKx, at least three extra clocks are needed at the beginning and end of each block of transfers to synchronize internal control signals (a block of transfers is a group of back-to-back continuous transfers).

Note: When in PSP mode, if the SSP port is the master of the clock and SSPSP_x[ETDS] is cleared, the end-of-transfer data state for SSPTXD is low. If the SSP port is the master of the clock and SSPSP_x[ETDS] is set, the end-of-transfer data state for SSPTXD remains at the last bit transmitted (LSB). When SSCR1_x[TTE] is set, SSPSP_x[ETDS] is ignored and SSPTXDx becomes high impedance between active transfers (see Figure 8-17 and Figure 8-18).

If the SSP port is a slave to the clock and SSPSP_x[SCMODE] is 0b01 or 0b11, then SSPSP_x[ETDS] can only change from the LSB if more clocks are sent to the SSP port (dummy stop clocks or slave clock is free running).

Figure 8-10. Programmable Serial Protocol (Single Transfer)

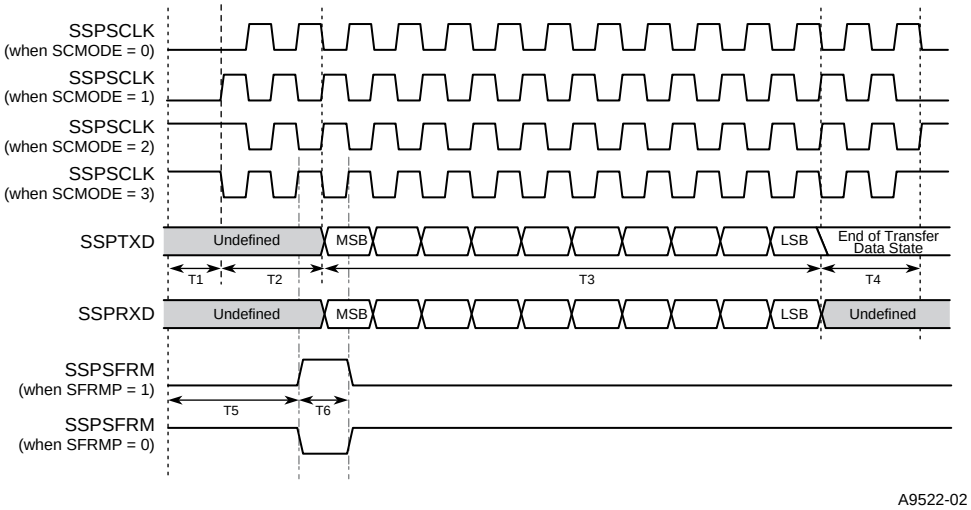


Figure 8-11. Programmable Serial Protocol (Multiple Transfers)

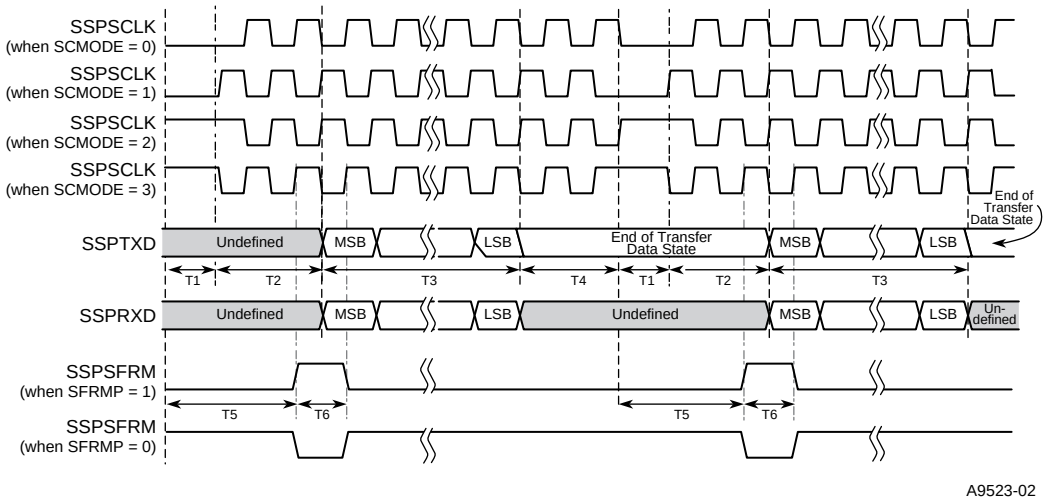
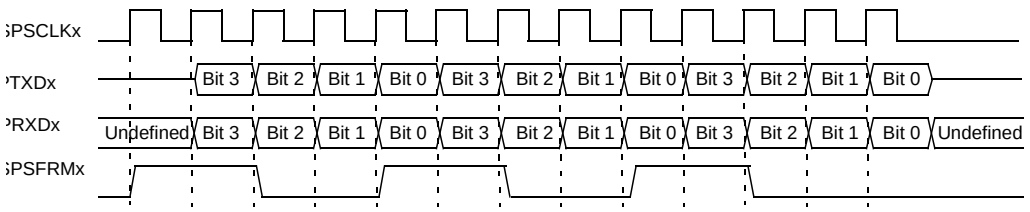


Figure 8-12. Programmable Serial Protocol Format (with Consecutive Transfers and SSPSP[FSRT] Set)



NOTE: This example uses a Frame width of 2, dummy start = 0, dummy stop = 0, start delay = 0, frame delay = 0, scmode = 1, Txd Tristate = 1, FSRT = 1, and frame polarity = 1.

Table 8-2. Programmable Serial Protocol (PSP) Parameters (Sheet 1 of 2)

Symbol	Definition	Range	Units
—	Serial Clock Mode (SSPSP_x[SCMODE])	(Drive, Sample, SSPSCLK Idle) 0—Fall, Rise, Low 1—Rise, Fall, Low 2—Rise, Fall, High 3—Fall, Rise, High	—
—	Serial Frame Polarity (SSPSP_x[SFRMP])	High or Low	—
T1	Start Delay (SSPSP_x[STRTDLY])	0–7	Clock period
T2	Dummy Start (SSPSP_x[DMYSTRT])	0–3	Clock period
T3	Data Size (SSCR0_x[EDSS] and SSCR0_x[DSS])	4–32	Clock period

Table 8-2. Programmable Serial Protocol (PSP) Parameters (Sheet 2 of 2)

Symbol	Definition	Range	Units
T4	Dummy Stop (SSPSP_x[DMYSTOP])	0–3	Clock period
T5	Frame Delay (SSPSP_x[SFRMDLY]	0–88	Half-clock period
T6	Frame Width (SSPSP_x[SFRMWDTH]	1–44	Clock period
	End of Transfer Data State (SSPSP_x[ETDS])	Low or [bit 0]	—

Note: SSPSFRMx delay must not extend beyond the end of T4. SSPSFRMx must be asserted for at least one SSPSCLKx, and must be de-asserted before the end of the T4 cycle (in terms of time, not bit values, (T5 + T6) ≤ (T1 + T2 + T3 + T4), 1≤ T6 < (T2 + T3 + T4), and (T5 + T6) ≥ (T1 + 1) to ensure that SSPSFRMx is asserted for at least two edges of the SSPSCLKx). Additionally, T1+T2 ≥ T5. The T1 start delay value must be programmed to zero when SSPSCLKx is enabled by SSPCLKxEN or either of SSCR1_x[ECRA] or SSCR1_x[ECRB]. While the SSP can be programmed to generate the assertion of SSPSFRMx during the middle of the data transfer (after the MSB was sent), the SSP port is not able to receive data in frame-slave mode (SSCR1_x[SFRMDIR] is set) if the assertion of frame is not before the MSB is sent (for example, T5 ≤ T2 if SSCR1_x[SFRMDIR] is set). Transmit data transitions from the end-of-transfer data state to the next MSB value upon the assertion of SSPSFRMx. The start delay field must be programmed to zero whenever SSPSCLKx or SSPSFRMx is configured as an input.

8.4.5 High Impedance on SSPTXDx

The SSP port supports placing SSPTXDx into high impedance during idle times (instead of driving SSPTXDx) as controlled by SSCR1_x[TTE] and SSCR1_x[TTELP]. The TTE bit enables high impedance on SSPTXDx. The TTELP bit determines which SSPSCLKx phase that SSPTXDx becomes high impedance.

8.4.5.1 TI SSP Format

When SSCR1_x[TTE] is clear, the SSP port behaves as described in Section 8.4.4.1.

When SSCR1_x[TTE] is set, SSPTXDx behaves as shown in Figure 8-13 or Figure 8-14, depending on SSCR1_x[TTELP].

Figure 8-13. TI SSP with SSCR1_x[TTE] = 1 and SSCR1_x[TTELP]] = 0

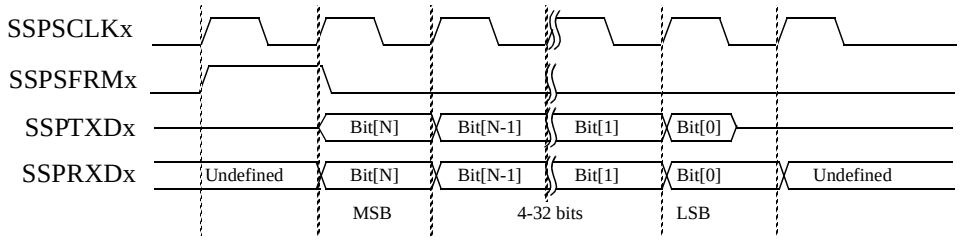
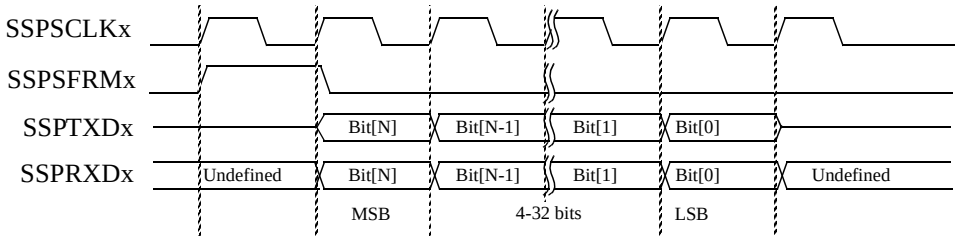


Figure 8-14. TI SSP with SSCR1_x[TTE] = 1 and SSCR1_x[TTELP] = 1

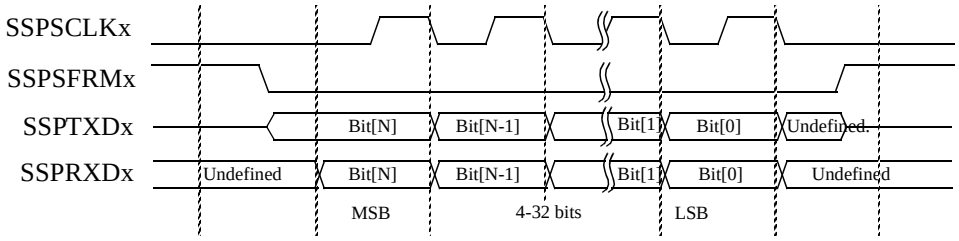


8.4.5.2 Motorola SPI Format

When SSCR1_x[TTE] is cleared, the SSP port behaves as described in Section 8.4.4.2.

When SSCR1_x[TTE] is set, SSPTXDx behaves as in Figure 8-15. SSCR1_x[TTELP] must be cleared for Motorola SPI format, and SSPTXDx becomes high impedance whenever SSPSFRMx is not active (high).

Figure 8-15. Motorola SPI with SSCR1_x[TTE] = 1 and SSCR1_x[TTELP] = 0

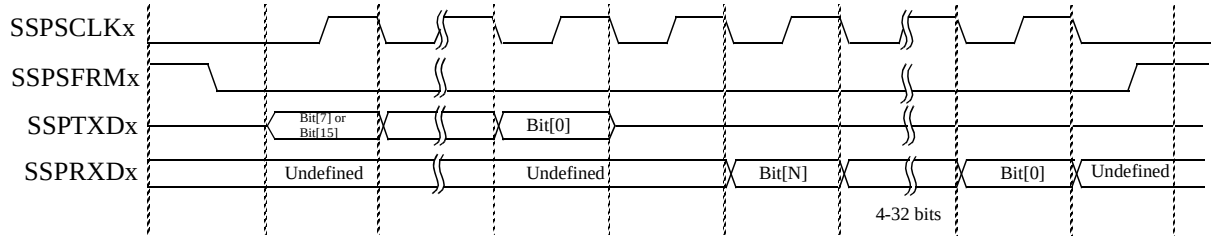


8.4.5.3 National Microwire Format

When SSCR1_x[TTE] is cleared, the SSP port behaves as described in Section 8.4.4.3.

When SSCR1_x[TTE] is set, SSPTXDx behaves as in Figure 8-16. SSCR1_x[TTELP] must be cleared for National Microwire format, and SSPTXDx becomes high impedance whenever SSPSFRMx is inactive (and after the LSB is sent on SSPTXDx).

Figure 8-16. National Microwire with SSCR1_x[TTE] = 1 and SSCR1_x[TTELP] = 0



8.4.5.4 PSP Format

When SSCR1_x[TTE] is cleared, the SSP port behaves as described in Section 8.4.4.4.

When SSCR1_x[TTE] is set, SSPTXDx behaves as shown in Figure 8-17 or Figure 8-18, depending on SSCR1_x[TTELP].

Figure 8-17. PSP Format with SSCR1_x[TTE] = 1 and SSCR1_x[TTELP] = 0

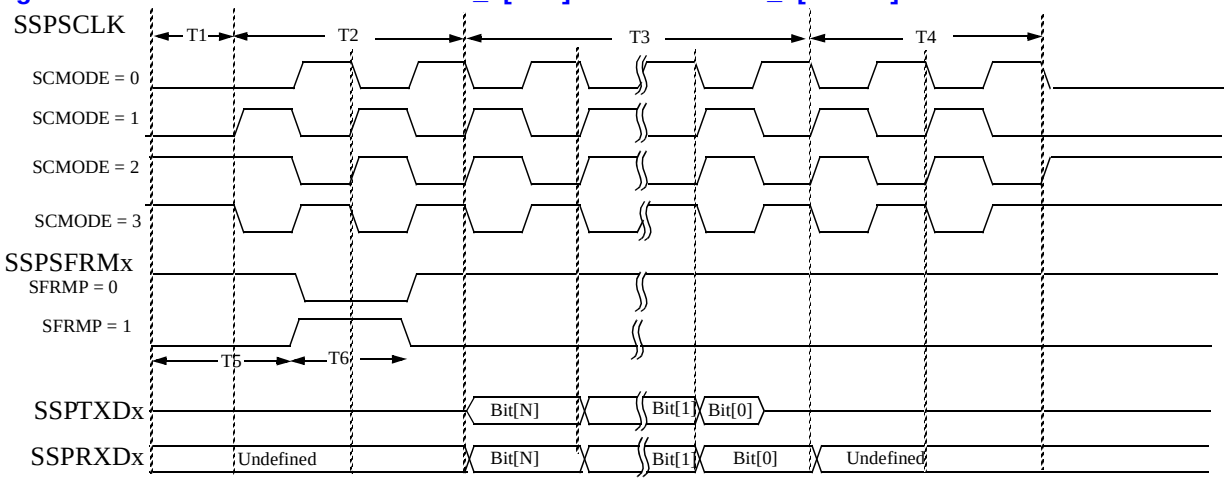
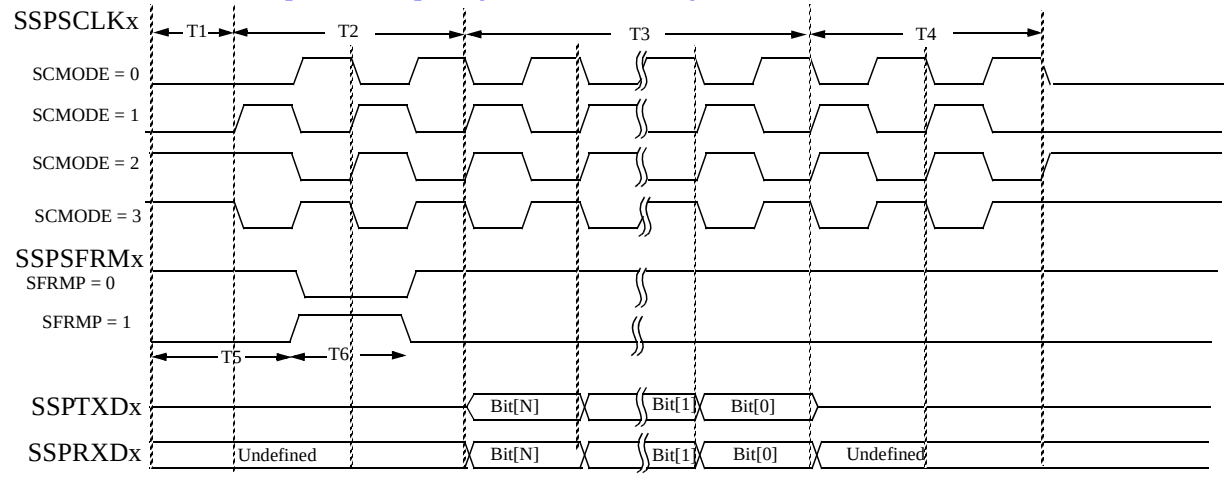


Figure 8-18. PSP Format with SSCR1_x[TTE] = 1 and Either SSCR1_x[TTELP] = 1 or SSCR1_x[SFRMDIR] = 0 (Master to Frame)



8.4.6 Network Mode

Network mode (see Figure 8-3) is used in systems where several devices are connected to the same SSPTXDx, SSPRXDx, SSPSFRMx, and SSPSCLKx lines. In network mode, from 1–8 time slots can be chosen. The SSP port can transmit or receive in any of the slots (see Table 8-13 and Table 8-14). Only TI SSP and PSP formats support network mode, which is enabled by setting SSCR0_x[MOD] (see Table 8-6).

In network mode, SSPSCLKx runs continuously if the SSP port is a master of the clock (SSCR1_x[SCLKDIR] = 0). Only one SSPSFRMx is sent (received) for the number of time slots programmed into the SSCR0_x[FRDC] field.

When beginning in network mode (and the SSP port is a master to the frame signal), the first SSPSFRMx does not occur until after data is in the transmit FIFO. After the first SSPSFRMx, if the SSP port is a master to the frame signal, frame syncs continue to come regardless of whether there is data in the transmit FIFO. Therefore, the transmit-underrun (SSSR_x[TUR]) bit is set if there is no data in the transmit FIFO and the SSP port is scheduled to drive SSPTXDx in the current time slot (even if the SSP port is master to frame).

When shutting down from network mode (and the SSP port is a master to frame), software must clear the MOD bit (SSCR0_x[SSE] does not need to change), then wait until SSTSS_x[NMBSY] is cleared before shutting down the SSP port (clearing SSE). The SSP port continues driving clocks and/or frame (depending upon SSCR1_x[SFMRDIR] and SSCR1_x[SCLKDIR]) until the last valid time slot is over. If the SSP port is a slave to both clock and frame, NMBSY remains asserted until the MOD bit is cleared or until one SSPSCLKx after the last valid time slot is over. When the SSP port is a master to frame, software must ensure that no data remains in the transmit FIFO after the network mode is exited (or else a non-network mode frame is sent). Due to the synchronization between the processor clock domains, one extra frame may be sent out after software clears the MOD bit.

Note: The SSP port transmits data in inactive time slots in network mode. When FRDC is set to > 0 (> 1 slots per frame) and, in SSTSA and SSRSA registers, only slot 0 is enabled and the rest of the slots are disabled, then the SSP repeats the slot 0 data on the rest of the time slots during data transmit. If SSCR1_x[TTELP] and SSCR1_x[TTE] are clear, the SSP continues to ship the last data on disabled slots. For example, if FRDC is set to 3 (four slots per frame), only slot 0 is enabled, and TTELP and TTE are clear, the SSP repeats the slot 0 data on time slot 1, 2, and 3. If TTELP and TTE are set, the SSP three-states the data lines during the disabled slots.

8.4.7 Parallel Data Formats for FIFO Storage

The CPU or the DMA transfers one FIFO entry per access. Data in the FIFOs are stored with one 32-bit value per data sample, regardless of the format data word length. Within each 32-bit field, the stored data sample is right justified, with the LSB of the word in bit 0. In the receive FIFO, unused bits are packed as zeroes above the MSB of the data sample. In the transmit FIFO, unused bits are above the MSB of the data sample. DMA and CPU access do not have to write to the unused bit locations. Logic in the SSP port automatically formats data in the transmit FIFO so that the sample is properly transmitted on SSPTXDx in the selected frame format.

8.4.8 Continuous Serial Clock Operation

In addition to the normal operation, SSPSCLKx can be configured to run continuously even if disabled by clearing SSCR0_x[SSE]. This allows an SSP port (when it has data to transmit) to enable the serial clock of a different SSP port or an external device to enable the SSP port source clocks.

This capability supports applications where the SSP ports interface with multiple peripherals that share a common synchronized bit clock (for instance, a cellular network clock) that is generated by one SSP port but used by other SSP ports. For example, SSP ports can interface with a Bluetooth baseband processor (to carry voice data), a cellular baseband processor, and an audio Codec—all synchronized to a network clock. This clock-gating logic ensures that the common clock is running and consuming power only when an SSP port has data to send or an external device requests the clock using SSPSCLKENx.

When the SSP controller sends a clock request to the serial-clock source SSP port, the serial clock of the serial-clock source SSP port (refer to Table 8-3) is driven depending on SSCR1_x[ECRA] and SSCR1_x[ECRB]. The clock request is a signal internal to the SSP port and is asserted when an SSP port has data to send (for example, the transmit FIFO, Transmit FIFO Holding register, and Shift register are not all empty.) This condition is denoted as the tx_not_empty signal in Figure 8-19.

Note: In the following text, the term SSP_a refers to any of the three SSP ports. The term SSP_b refers to any of the three SSP ports except for those included in SSP_a. For example, if SSP_a includes SSP1, SSP_b cannot include SSP1 but can include SSP2 and SSP3.

The SSPSCLKa of SSP_a continues to run (at the baud rate SSP_a is programmed to) if at least one of the following conditions are true:

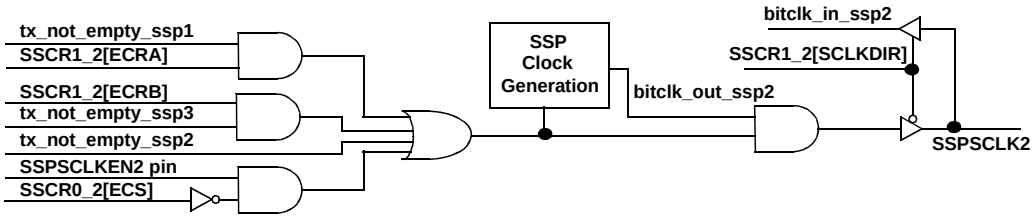
- SSP_b clock request is active (indicated by tx_not_empty in Figure 8-19) and SSCR1_x[ECRA] and SSCR1_x[ECRB] are set so that a clock request from SSP_b into SSP_a is enabled in SSP_a (refer to Table 8-3).
- SSP_a is enabled and in the middle of a transaction (normal operation).
- SSPCLKaEN of SSP_a is high and SSCR0_x[ECS] of SSP_a is cleared.

Table 8-3 shows the clock-request-enable selections possible. By setting either SSCR1_x[ECRA] or SSCR1_x[ECRB], the source clock of an SSP port can be controlled by another SSP port clock request.

Table 8-3. SSP Port Clock Request Enable Selection

		Serial Clock Source SSP Port		
		SSP1	SSP2	SSP3
Clock Request SSP	SSP1		SSCR1_2[ECRA] set	SSCR1_3[ECRA] set
	SSP2	SSCR1_1[ECRA] set		SSCR1_3[ECRB] set
	SSP3	SSCR1_1[ECRB] set	SSCR1_2[ECRB] set	

Figure 8-19. Clock Enabling on SSP2



NOTES: The SSCR0_2 and SSCR1_2 control bits above are in SSP2. SSP3 does not have an external clock source; Because of this, SSPSCLKEN for SSP3 is tied to ground.

8.4.9 FIFO Operation

Two separate and independent FIFOs are present for transmit (to peripheral) and receive (from peripheral) serial data. FIFOs are filled or emptied by CPU programmed I/O or by DMA bursts.

8.4.9.1 Using CPU Programmed I/O Data Transfers

The CPU can perform FIFO filling and emptying in response to an interrupt from the FIFO logic. Each FIFO has a programmable trigger threshold at which an interrupt is triggered (if enabled). When the number of entries in the receive FIFO exceeds the value in SSCR1_x[RFT] + 1, an interrupt is generated (if enabled). This interrupt signals the CPU to empty the receive FIFO. When the number of entries in the transmit FIFO is less than or equal to the value of (SSCR1_x[TFT] + 1), an interrupt is generated (if enabled). This interrupt signals the CPU to refill the transmit FIFO.

Reading the SSP Status register (see Section 8.5.3) shows whether the FIFO is full, empty, or how many samples it contains.

8.4.9.2 Using DMA Data Transfers

The DMA controller can be programmed to transfer data to and from the SSP port FIFOs. The SSP port stores data in its FIFOs one sample per FIFO location (each FIFO has 16 locations). Therefore, the DMA burst size (DCMD_x[SIZE]) must not exceed 16 samples. For example, the maximum DMA burst size is 16 bytes when the DCMD_x[WIDTH] is set to byte wide (0b01). There must be sufficient empty room in the transmit FIFO or a sufficient number of samples in the receive FIFO before a DMA burst is enabled. The SSP port data-sample size, the DMA sample width, and the SSP port TX/RX FIFO threshold settings together determine the allowable DMA burst size. To prevent underruns of the transmit FIFO or overruns of the receive FIFO when using DMA, take care when setting the transmit and receive trigger thresholds. Refer to Table 8-4 for the allowable DMA burst sizes for different combinations of SSP port sample size, DMA sample width and SSP port threshold levels. Refer to Chapter 5, “DMA Controller” for instructions on programming the DMA channels.

The programming model for using the DMA is as follows:

- Program the total number of transmit and receive byte lengths, burst sizes, and peripheral width. Program DCMD_x[WIDTH] to 0b01 for SSP port formats of 8 bits or less; to 0b10 for SSP port formats of 9 to 16 bits; to 0b11 for SSP port formats of more than 16 bits.
- Set the preferred values in the SSP Control registers.

- Set SSCR0_x[SSE] to enable the SSP port (see Section 8.5.1).
- Set DCSR_x[RUN].
- Wait for the DMA transmit- or receive-interrupt requests.
- If the transmit/receive byte length is not an even multiple of the transfer burst size, a trailing-byte condition may occur as described within Section 8.4.2.

In full-duplex formats where the SSP port always receives the same number of data samples as it transmits, the DMA channel must be set up to transmit and receive the same number of bytes.

Table 8-4. TFT and RFT Values with Possible DMA Burst Sizes

SSP Data Size (SSCR0_x[EDSS], SSCR0_x[DSS])	DMA Width (DCMDx[WIDTH])	TX/RX Threshold (SSCR1_x[TFT], SSCR1_x[RFT]) †	Allowed DMA Burst Size (in Bytes)
4–8 bits	1 byte (0b01)	TFT = 0 RFT = 15	8 or 16 8 or 16
		0 < TFT < 8 6 < RFT < 15	8 8
		TFT > 7 RFT < 7	Do not use DMA Do not use DMA
9–16 bits	2 bytes (0b10)	TFT = 0 RFT = 15	8, 16, or 32 8, 16, or 32
		TFT < 8 6 < RFT < 15	8 or 16 8 or 16
		7 < TFT < 12 2 < RFT < 7	8 8
		TFT > 11 RFT < 3	Do not use DMA Do not use DMA
17–32 bits	4 bytes (0b11)	TFT < 8 RFT > 6	8, 16, or 32 8, 16, or 32
		7 < TFT < 12 2 < RFT < 7	8 or 16 8 or 16
		11 < TFT < 14 0 < RFT < 3	8 8
		TFT > 13 RFT = 0	Do not use DMA Do not use DMA

† Valid values for TFT and RFT are 0 through 15. For register details, see Table 8-7.

The DMA transmit burst size is limited because the smallest SSCR1_x[TFT] value is zero, which equates to one sample left in the transmit FIFO. Software must program the transmit FIFO threshold and the DMA burst size (DCMD_x[SIZE]) such that an underflow of the transmit FIFO does not occur.

The receive FIFO threshold and the DMA burst size must be configured such that an overflow of the receive FIFO does not occur.

8.4.10 Baud-Rate Generation

When the SSP port is configured as the master of SSPSCLKx as determined by SSCR1_x[SCLKDIR], the baud rate (serial bit-rate SSPSCLKx) is generated internally by dividing one of the following by a programmable divider (SSCR0_x[SCR]):

- The on-chip 13-MHz clock
- The network clock (CLK_EXT) described in Section 24.4.2, "GPIO Operation as Alternate Function" on page 24-3.
- The SSP port external clock (SSPEXTCLKx)

The division of the baud rate by one of these programmable dividers generates baud rates up to a maximum of 13 Mbits per second. If the audio-clock-select bit is set (SSCR0_x[ACS]), the frequency of the audio clock used by the SSP port is determined by the Audio Clock Divider register (SSACD_x). The audio clock can also be routed through the baud-rate divider, but its phase relationship with SSPSYSCLK will be lost.

Table 8-5 shows the clock selected to source the baud-rate generator by the external clock-select (SSCR0_x[ECS]), network clock-select (SSCR0_x[NCS]), or audio clock-select (SSCR0_x[ACS] (see Section 8.5.2 for register details).

Follow these steps when changing clock sources:

1. Turn off the SSP port internal clock by clearing the appropriate bit in the clock unit's CKEN register—for example, CKEN[23] for SSP1. For full mapping information, see Table 3-33, "Clock Enable Mappings for CKEN Bits" on page 3-99. For CKEN register details, see Section 3.8.2.2, "Clock Enable Register (CKEN)" on page 3-98.
2. Write SSCR0_x[ECS] or SSCR0_x[NCS] or SSCR0_x[ACS].
3. Re-enable the SSP port internal clock by setting the appropriate CKEN register bit.

Table 8-5. SSP Port Clock Selection

SSCR0_x[ACS]	SSCR0_x[ECS]	SSCR0_x[NCS]	Selected Clock
0	0	0	On-Chip Clock (internal 13-MHz PLL)
0	0	1	Network Clock (CLK_EXT)
0	1	0	SSP External Clock (SSPEXTCLK)
0	1	1	Network Clock (CLK_EXT)
1	X	X	On-Chip Audio Clock (determined from internal PLL and SSACD_x)

8.4.11 32-Bit I^2S Emulation using SSP

To emulate 32-bit I^2S mode using the SSP controller, follow the configurations below to support either normal or "MSB-justified" mode.

8.4.11.1 "Normal" Mode

The following bit fields must be configured:

- SSCR0[EDSS] = 0b1 (32-bit data)
- SSCR0[FRF] = 0b11 (PSP format)

- SSCR0[DSS] = 0b1111 (32-bit data)

The example below shows the use of the recommended settings:

- SSCR0 = 0x001000BF (Only SSCR0[NCS] or SSCR0[ECS] bit fields settings are optional)
- SSCR1 = 0x203C3C03 (SSCR1[SCLKDIR] and SSCR1[SFRMDIR] must be cleared, all other bit fields settings are optional.)
- SSPSP = 0x02100000 (all bit fields must be cleared except: FSRT = 1 and SFRMWDTH = 16, DMYSTART=0,1)

8.4.11.2 "MSB-justified" Mode

All configurations in the example for "normal" mode above remain the same for "MSB-justified" mode, with the exception for SSPSP.

SSPSP = 0x00100000 (all bit fields must be cleared except: SFRMWDTH = 16, DMYSTART =0,1)

8.5 Register Descriptions

Each of the three SSP ports contain eleven registers: six control, one data, two status, one time-out, and one test.

- Access all registers using aligned words.
- The SSP Control registers SSCR0_x, and SSCR1_x configure the baud rate, data length, frame format, data-transfer mechanism, and port enabling. They also permit setting the FIFO trigger threshold that triggers an interrupt. For SSCR0_x, the DSS, FRF, EDSS, ECS, NCS, and ACS bit fields must be written before the SSE bit is set. If the DSS, FRF, EDSS, ECS, NCS, or ACS bits need to be modified after the SSE bit is set; clear the SSE bit, make the modifications, and then again set the SSE bit. For SSCR1_x, only the SPO, SPH, and SCFR bits must be written before the SSE bit is set. If the SPO, SPH, or SCFR bits need to be modified after the SSE bit is set; clear the SSE bit, make the modifications, and then again set the SSE bit.
- Any writable bits in the SSSR, SSTO, and SSITR registers can be written at any time.
- All bit fields in the SSPSP and SSACD registers must be written before the SSE bit is set. If these bits need to be modified after the SSE bit is set; clear the SSE bit, make the modifications, and then reset the SSE bit.
- Write all bits in the SSRSA and SSTSA registers before the SSE bit is set. If these bits need to be modified after the SSE bit is set; clear the SSE bit, make the modifications, and then reset the SSE bit.
- The SSP Time-out (SSTO_x) register programs the time-out value to signal a specified period of receive FIFO inactivity.
- While in PSP mode, the SSP Programmable Serial Protocol (SSPSP_x) registers program the parameters used in defining the data transfer.
- The data register is mapped as one 32-bit location, which physically points to either of two 32-bit registers: one register is for writes of data to the transmit FIFO and the other register is for reads that take data from the receive FIFO. A write cycle or burst write loads successive words into the SSP port SSDR_x write register and then into the transmit FIFO. A read cycle or burst

read takes data from the SSP port SSDR_x read register as the receive FIFO reloads it with the "oldest" available FIFO data bits.

Do not increment the address when using read and write DMA bursts. All accesses to the SSP Data register memory address access either the SSDR_x read register or SSDR_x write register.

The FIFOs are independent buffers that allow full duplex operation.

- Besides showing the state of the FIFO buffers, the status register (SSSR_x) shows whether the programmable trigger threshold has been passed and whether a transmit or receive FIFO service request is active. The Status register also shows how full the FIFO is. Flag bits indicate when the SSP port is actively transmitting data, when the transmit FIFO is not full, and when the receive FIFO is not empty. SSSR_x[ROR] bit signals an overrun of the receive FIFO. In this case newly received data is discarded (see Section 8-11).

8.5.1 SSP Control Register 0 (SSCR0_x)

SSCR0_x, shown in Table 8-6, controls various functions within the SSP port. Before enabling the SSP port by setting SSCR0_x[SSE], the preferred values for this register must be programmed.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 8-6. SSCR0_1/2/3 Bit Definitions (Sheet 1 of 5)

Physical Address
0x4100_0000
0x4170_0000
0x4190_0000

SSCR0_1
SSCR0_2
SSCR0_3

SSP Controller

	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
User Settings																																
Bit	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
	MOD	ACS	reserved			FRDC			TIM	RIM	NCS	EDSS	SCR												SSE	ECS	FRF		DSS			
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

Bits	Access	Name	Description
31	R/W	MOD	Mode 0 = Normal SSP port mode 1 = Network mode **NOTE:** When network mode is selected, only use DMA to move data to/from the SSP Data register (SSDR_x). Do not use interrupts or CPU polling. Set this bit only when using PSP and TI SSP formats. Setting this bit causes SSPSCLKx to run continuously (if the SSP port is a master of the clock—SSCR1_x[SCLKDIR] = 0). During network mode, only one SSPSFRMx is sent (received) for the number of time slots programmed into the SSCR0_x[FRDC] field. See Section 8.4.6 for additional information. When using the PSP format in network mode, the parameters SFRMDLY, STRTDLY, DMYSTP, DMYSTRT must be cleared. Other parameters (such as FRMPOL, SCMODE, FSRT, SFRMDWDTH) are programmable.

Table 8-6. SSCR0_1/2/3 Bit Definitions (Sheet 2 of 5)

Physical Address		SSP Controller
0x4100_0000 0x4170_0000 0x4190_0000	SSCR0_1 SSCR0_2 SSCR0_3	

	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
User Settings																																
Bit	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
	MOD	ACS	reserved			FRDC			TIM	RIM	NCS	EDSS	SCR												SSE	ECS	FRF		DSS			
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

Bits	Access	Name	Description
30	R/W	ACS	Audio Clock Select 0 = SSPSCLKx selection is determined by the NCS and ECS bits. 1 = Audio clock (and audio clock divider) creates SSPSCLKx. If the ACS bit is set (and the GPIO are properly configured), SSPSYSCLKx is continually output (even if the SSP port is disabled). SSPSCLKx is output as previously defined (determined by format, SSPSCLKENx, ECRA/ECRB functions).
29:27	—	—	reserved
26:24	R/W	FRDC	Frame Rate Divider Control Value 0-7 indicates the number of time slots per frame when in network mode (the actual number of time slots is FRDC + 1 for 1–8 time slots)
23	R/W	TIM	Transmit FIFO Underrun Interrupt Mask 0 = TUR events generate an SSP port interrupt. 1 = TUR events do not generate an SSP port interrupt. When set, this bit masks the TX FIFO underrun (TUR) event from generating an SSP port interrupt. SSSR_x still indicates that a TUR event has occurred. This bit can be written to at any time (before or after SSP port is enabled).
22	R/W	RIM	Receive FIFO Overrun Interrupt Mask 0 = ROR events generate an SSP port interrupt. 1 = ROR events do not generate an SSP port interrupt. When set, this bit masks the RX FIFO overrun (ROR) event from generating an SSP port interrupt. SSSR_x still indicates that an ROR event has occurred.This bit can be written to at any time (before or after SSP port is enabled).
21	R/W	NCS	Network Clock Select Used with ECS to select the network clock. 0 = The NCS bit determines clock selection. 1 = Network clock creates the SSP port SSPSCLKx. Before setting the NCS bit, first disable the port. The NCS (and ECS) bits must be configured before or at the same time that the SSE bit is set. For more information, see Table 8-5.
20	R/W	EDSS	Extended Data Size Select Used with DSS to select the size of the data transmitted and received by the SSP port. 0 = Zero is pre-appended to the DSS value that sets the DSS range from 4-16- bits. 1 = One is pre-appended to the DSS value that sets the DSS range from 17-32-bits.

Table 8-6. SSCR0_1/2/3 Bit Definitions (Sheet 3 of 5)

Physical Address		SSP Controller
0x4100_0000 0x4170_0000 0x4190_0000	SSCR0_1 SSCR0_2 SSCR0_3	

	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
User Settings																																
Bit	MOD	ACS	reserved			FRDC			TIM	RIM	NCS	EDSS	SCR												SSE	ECS	FRF		DSS				
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

Bits	Access	Name	Description
19:8	R/W	SCR	Serial Clock Rate Selects the bit rate of the SSP port when in master mode with respect to SSPSCLKx (as defined by SSCR1_x[SCLKDIR]). The maximum bit rate is 13 Mbps. The serial-clock generator uses clocks selected by ECS and NCS. The selected clock is divided by the value of SCR plus 1 (a range of 1 to 4096) to generate SSPSCLKx. **NOTE:** This field is ignored when the SSP port is a slave with respect to SSPSCLKx (defined by SSCR1_x[SCLKDIR]) and transmission data rates are determined by an external device. Software must not change SCR when SSPSCLKx is enabled (through use of the SSPSCLKEN pin or SSPCR1_x[ECRA] or SPCR1_x[ECRB]) because doing so causes the SSPSCLKx frequency to immediately change. Values (0 to 4095) generate the clock rate of the SSP port. Serial bit rate = SSP port clock / (SCR + 1), where SCR is a decimal integer
7	R/W	SSE	Synchronous Serial Enable Enables or disables all SSP port operations. When the port is disabled, all of its clocks can be stopped by programmers to minimize power consumption. When cleared during active operation, the SSP port is disabled immediately, terminating the current frame being transmitted or received. Clearing SSE resets the port FIFOs and the status bits; however, the SSP Port Control registers and the receive-FIFO-overrun status bit are not reset. **NOTE:** After reset or after clearing the SSE, ensure that the SSCR1_x, SSITR_x, SSTO_x, and SSPSP_x control registers are properly re-configured and that the SSSR register is reset before re-enabling the SSP port by setting SSE. Also, SSE must be cleared before re-configuring the SSCR0_x, SSCR1_x, or SSPSP_x registers; any or all control bits in SSCR0_x can be written at the same time as the SSE. 0 = SSP port operation disabled. 1 = SSP port operation enabled.

Table 8-6. SSCR0_1/2/3 Bit Definitions (Sheet 4 of 5)

Physical Address		SSP Controller
0x4100_0000 0x4170_0000 0x4190_0000	SSCR0_1 SSCR0_2 SSCR0_3	

	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
User Settings																																
Bit	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
	MOD	ACS	reserved			FRDC			TIM	RIM	NCS	EDSS	SCR												SSE	ECS	FRF		DSS			
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

Bits	Access	Name	Description
6	R/W	ECS	External Clock Select Used with NCS to select the clock source for the SSP port. ECS in conjunction with NCS selects whether the SSP port uses the on-chip 13-MHz clock or one of two off-chip clocks supplied by GPIO: the network clock (CLK_EXT, described in Chapter 24, "General-Purpose I/O Controller") or the SSP port's external clock (SSPEXTCLK) produces serial transmission rates ranging from 6.3 Kbps (minimum recommended bit rate) to a maximum of 13 Mbps. When NCS is cleared, ECS selects between the on-chip 13-MHz clock and external clock (SSPEXTCLK). When NCS is set, the network clock (CLK_EXT) is selected. The frequency of the off-chip clock can be any value up to 13 MHz. When the SSP port is a slave with respect to SSPSCLKx (defined by the SSCR1_x[SCLKDIR]), this field is ignored and transmission data rates are determined by the external device. Before setting the ECS bit, first disable the port. The ECS (and NCS) bit must be configured before or at the same time that SSE is set. For more information see Table 8-5. When ECS is cleared, SSPEXTCLKx is treated as SSPSCLKENx, a clock enable that gates the SSPSCLKx output. See Figure 8-19 for details. When the SSPSCLKENx changes, there is a 1–2 clock lag before SSPSCLKx is started or stopped because of internal synchronization delays. 0 = On-chip clock produces the SSP port SSPSCLKx. 1 = SSPEXTCLK/GPIO pin creates the SSP port SSPSCLKx. **NOTE:** The ECS bit for SSP3, SSCR0_3[ECS], should never be set to one because the SSP3 does not have an associated SSPEXTCLK.
5:4	R/W	FRF	Frame Format Selects which frame format to use. 0b00 = Motorola Serial Peripheral Interface 0b01 = TI Synchronous Serial Protocol 0b10 = Microwire 0b11 = Programmable Serial Protocol

Table 8-6. SSCR0_1/2/3 Bit Definitions (Sheet 5 of 5)

Physical Address		SSP Controller
0x4100_0000 0x4170_0000 0x4190_0000	SSCR0_1 SSCR0_2 SSCR0_3	

Bit	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
User Settings																																
	MOD	ACS	reserved			FRDC			TIM	RIM	NCS	EDSS	SCR												SSE	ECS	FRF		DSS				
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

Bits	Access	Name	Description
3:0	R/W	DSS	Data Size Select Used in conjunction with EDSS to select the size of the data transmitted and received by the SSP port. The concatenated 5-bit value of EDSS and DSS provides a data range from four to 32-bits in length. For the Microwire protocol, DSS and EDSS determine the receive data size. The size of the transmitted data is either eight or 16-bits (determined by SSCR1_x[MWDS]) and the EDSS bit is ignored. For all modes (including Microwire), EDSS and DSS determine the receive data size. When data is programmed to be less than 32 bits, data written to the TX FIFO must be right-justified.

EDSS	DSS	Data Size	EDSS	DSS	Data Size
1	0b0000	17-bit data	0	0b0000	reserved, undefined
1	0b0001	18-bit data	0	0b0001	reserved, undefined
1	0b0010	19-bit data	0	0b0010	reserved, undefined
1	0b0011	20-bit data	0	0b0011	4-bit data
1	0b0100	21-bit data	0	0b0100	5-bit data
1	0b0101	22-bit data	0	0b0101	6-bit data
1	0b0110	23-bit data	0	0b0110	7-bit data
1	0b0111	24-bit data	0	0b0111	8-bit data
1	0b1000	25-bit data	0	0b1000	9-bit data
1	0b1001	26-bit data	0	0b1001	10-bit data
1	0b1010	27-bit data	0	0b1010	11-bit data
1	0b1011	28-bit data	0	0b1011	12-bit data
1	0b1100	29-bit data	0	0b1100	13-bit data
1	0b1101	30-bit data	0	0b1101	14-bit data
1	0b1110	31-bit data	0	0b1110	15-bit data
1	0b1111	32-bit data	0	0b1111	16-bit data

8.5.2 SSP Control Register 1 (SSCR1_x)

SSCR1_x, shown in Table 8-7, controls various SSP port functions. Before enabling the port (using SSCR0_x[SSE]), the preferred values for this register must be set.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 8-7. SSCR1_1/2/3 Bit Definitions (Sheet 1 of 9)

Physical Address
0x4100_0004
0x4170_0004
0x4190_0004

SSCR1_1
SSCR1_2
SSCR1_3

SSP Controller

	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
User Settings																																
Bit	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
	TTELP	TTE	EBCEI	SCFR	ECRA	ECRB	SCLKDIR	SFRMDIR	RWOT	TRAIL	TSRE	RSRE	TINTE	PINTE	Reserved	IFS	STRF	EFWR	RFT				TFT				MWDS	SPH	SPO	LBM	TIE	RIE
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

Bits	Access	Name	Description
31	R/W	TTELP	TXD Tristate Enable on Last Phase TTELP is used in conjunction with TTE. When set, TTELP causes SSPTXDx to become high impedance 1/2 phase later than specified in the TTE bit description under either of the following 2 conditions: 1) the format is SSP; or 2) the format is PSP and the SSP port is a slave to frame (SSCR1_x[SFRMDIR] is set). For example, when in SSP format, with TTELP and TTE set, SSPTXDx goes to high impedance one full clock after the clock edge that starts the LSB. Ensure if SCLKDIR is set (SSP port is a slave to SSPSCLKx) that the device driving SSPSCLKx provides another clock edge. **NOTE:** TTELP must not be set when using Microwire or SPI formats 1 = SSPTXDx becomes high impedance one full clock after the clock edge that starts the LSB. 0 = SSPTXDx becomes high impedance 1/2 clock after the clock edge that starts the LSB.
30	R/W	TTE	TXD Tristate Enable If TTE is cleared, SSPTXDx is always driven. When set, TTE causes SSPTXDx to become high impedance when the SSP port is not transmitting data. The timing for the high impedance enable/disable varies according to the different serial formats and frame direction. For Microwire format, SSPTXDx is driven on the same clock edge that the MSB is driven, and SSPTXDx becomes high impedance one full clock after the clock edge that starts the LSB. See Section 8.4.5.3. For SPI protocol, SSPTXDx becomes high impedance whenever SPSFRMx is de-asserted (driven high). See Section 8.4.5.2. For TI Synchronous Serial Protocol, SSPTXDx is driven at the first rising edge of the SSPSCLKx after SSPSFRMx is asserted (the same time the MSB is driven) and continues to be driven until 1/2 clock (TTELP = 1) or 1 full clock (TTELP = 0) after the clock edge that starts the LSB. See Section 8.4.5.1. For PSP format when the SSP port is a slave to frame, SSPTXDx becomes high impedance 1/2 clock after the clock edge that starts the LSB. For PSP format when the SSP port is a master to frame, SSPTXDx becomes high impedance one full clock after the clock edge that starts the LSB (even if the SSP port is a master of clock and this clock edge does not appear on SSPSCLKx). See Section 8.4.5.4. **NOTE:** TTE must be set when in network mode. 0 = SSPTXDx does not become high impedance. 1 = SSPTXDx becomes high impedance when not transmitting data.

Table 8-7. SSCR1_1/2/3 Bit Definitions (Sheet 2 of 9)

Physical Address		SSCR1_1 SSCR1_2 SSCR1_3	SSP Controller
0x4100_0004 0x4170_0004 0x4190_0004			

User Settings																															
Bit	**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**
	TTELP	TTE	EBCEI	SCFR	ECRA	ECRB	SCLKDIR	SFRMDIR	RWOT	TRAIL	TSRE	RSRE	TINTE	PINTE	Reserved	IFS	STRF	EFWR	RFT				TFT				MWDS	SPH	SPO	LBM	TIE	RIE
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

Bits	Access	Name	Description
29	R/W	EBCEI	Enable Bit Count Error Interrupt When set, EBCEI enables a bit count error interrupt. A bit count error occurs when the SSP port is a slave to SSPSCLKx and/or SSPSFRMx and the SSP port detects a new frame before the internal bit counter has reached 0 (before the LSB was driven). 0 = Interrupt due to a bit count error is disabled. 1 = Interrupt due to a bit count error is enabled.
28	R/W	SCFR	Slave Clock Free Running In slave mode (SSCR1_x[SCLKDIR] is set), SCFR must be cleared if the input clock from the external source is running continuously. In master mode (SSCR1_x[SCLKDIR] is cleared), SCFR is ignored. Slave mode only: 0 = SSPSCLKx is continuously running. 1 = SSPSCLKx is active only during transfers.
27	R/W	ECRA	Enable Clock Request A Refer to Section 8.4.8 for details on the use of this bit. 0 = Clock request from another SSP port is disabled. 1 = Clock request from another SSP port is enabled.
26	R/W	ECRB	Enable Clock Request B Refer to Section 8.4.8 for details on the use of this bit. 0 = Clock request from another SSP port is disabled. 1 = Clock request from another SSP port is enabled.
25	R/W	SCLKDIR	SSPSCLKx Direction SCLKDIR determines whether the port is the master or slave (with respect to driving SSPSCLKx). Depending on the frame format selected, each transmitted bit is driven on either the rising or falling edge of SSPSCLKx, and is sampled on the opposite clock edge. When the GPIO alternate function is selected for the SSP port, this bit has precedence over the GPIO direction bit. SCLKDIR must be written before the GPIO direction bit (to prevent any possible contention on SSPSCLKx). 0 = Master mode, the port generates SSPSCLKx internally, acts as the master, and drives SSPSCLKx. 1 = Slave mode, the port acts as a slave, receives SSPSCLKx from an external device and uses it to determine when to drive transmit data on SSPTXDx and when to sample receive data on SSPRXDx. **NOTE:** When SCLKDIR is set, the SSCR0_x[NCS] and SSCR0_x[ESC] bits must be cleared.

Table 8-7. SSCR1_1/2/3 Bit Definitions (Sheet 3 of 9)

Physical Address
0x4100_0004
0x4170_0004
0x4190_0004

SSCR1_1
SSCR1_2
SSCR1_3

SSP Controller

	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
User Settings																																
Bit	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
	TTELP	TTE	EBCEI	SCFR	ECRA	ECRB	SCLKDIR	SFRMDIR	RWOT	TRAIL	TSRE	RSRE	TINTE	PINTE	Reserved	IFS	STRF	EFWR	RFT				TFT				MWDS	SPH	SPO	LBM	TIE	RIE
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

Bits	Access	Name	Description
24	R/W	SFRMDIR	SSP Frame Direction SFRMDIR determines whether the SSP port is the master or slave (with respect to driving SSPSFRMx). **NOTE:** When the port is configured as a slave to SSPSFRMx, the external device driving SSPSFRMx must wait until SSSR_x[CSS] is cleared after enabling the port and before asserting SSPSFRMx (no external clock cycles are needed). When the GPIO alternate function is selected for the port, SFRMDIR has precedence over the GPIO direction bit. SFRMDIR must be written before the GPIO direction bit (to prevent any possible contention on SSPSFRMx). 0 = Master mode, the port generates SSPSFRMx internally, acts as the master and drives SSPSFRMx. 1 = Slave mode, the port acts as a slave, receives SSPSFRMx from an external device.
23	R/W	RWOT	Receive Without Transmit RWOT puts the SSP port into a mode similar to half duplex. This allows the port to receive data without transmitting data (half-duplex only). When the port is in master mode (SCLKDIR cleared) and RWOT is set, the port continues to clock in receive data, regardless of data existing in the transmit FIFO. Data is sent/received immediately after the port enable bit (SSCR0_x[SSE]) is set. In this mode, if there is no data to send, the DMA service requests and interrupts for the transmit FIFO must be disabled (clear both SSCR1_x[TSRE,TIE]). If the transmit FIFO is empty, SSPTXDx is driven low. The transmit-FIFO underrun condition does not occur when RWOT is set. When RWOT is set, SSSR_x[BSY] remains set until software clears the RWOT bit. After RWOT is cleared, and extra frame cycle may occur due to synchronization delays between the processor clock domains. RWOT must not be used when SSCR0_x[MOD] is set. 0 = Transmit/receive mode. 1 = Receive without transmit mode.
22	R/W	TRAIL	Trailing Byte TRAIL configures how trailing bytes are handled (see Section 8.4.2 for more detail). 0 = Processor based, Trailing bytes are handled by the CPU. 1 = DMA based, Trailing bytes are handled by DMA.

Table 8-7. SSCR1_1/2/3 Bit Definitions (Sheet 4 of 9)

Physical Address		SSCR1_1 SSCR1_2 SSCR1_3	SSP Controller
0x4100_0004 0x4170_0004 0x4190_0004			

User Settings																															
Bit	**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**
	TTELP	TTE	EBCEI	SCFR	ECRA	ECRB	SCLKDIR	SFRMDIR	RWOT	TRAIL	TSRE	RSRE	TINTE	PINTE	Reserved	IFS	STRF	EFWR	RFT				TFT				MWDS	SPH	SPO	LBM	TIE	RIE
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

Bits	Access	Name	Description
21	R/W	TSRE	Transmit Service Request Enable TSRE enables the transmit FIFO DMA Service Request. **NOTE:** Clearing TSRE does not affect the current state of SSSR_x[TFS] or the ability of the transmit-FIFO logic to set and clear SSSR_x[TFS]; it prevents the generation of the DMA service request. The state of TSRE does not effect the generation of the interrupt, which is asserted whenever the SSSR_x[TFS] is set. 0 = DMA service request is disabled and the state of the transmit FIFO DMA service request is ignored. 1 = DMA service request is enabled.
20	R/W	RSRE	Receive Service Request Enable RSRE enables the Receive FIFO DMA Service Request. **NOTE:** Clearing RSRE does not affect the current state of SSSR_x[RFS] or the ability of the receive FIFO logic to set and clear SSSR_x[RFS]; it blocks only the generation of the DMA service request. The state of RFRS does not affect the generation of the interrupt, which is asserted whenever SSSR_x[RFS] is set. 0 = DMA service request is disabled and the state of the SSSR_x[RFS] is ignored. 1 = DMA service request is enabled.
19	R/W	TINTE	Receiver Time-Out Interrupt Enable TINTE enables the receiver time-out interrupt. **NOTE:** Clearing TINTE does not affect the current state of SSSR_x[TINT] or the ability of logic to set and clear SSSR_x[TINT]; it prevents the generation of the interrupt request. 0 = Receiver time-out interrupts are disabled. The interrupt is masked and the state of SSSR_x[TINT] is ignored by the interrupt controller. 1 = Receiver time-out interrupts are enabled.

Table 8-7. SSCR1_1/2/3 Bit Definitions (Sheet 5 of 9)

Physical Address		SSP Controller
0x4100_0004 0x4170_0004 0x4190_0004	SSCR1_1 SSCR1_2 SSCR1_3	

Bit	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
User Settings																																
	TTELP	TTE	EBCEI	SCFR	ECRA	ECRB	SCLKDIR	SFRMDIR	RWOT	TRAIL	TSRE	RSRE	TINTE	PINTE	Reserved	IFS	STRF	EFWR	RFT				TFT				MWDS	SPH	SPO	LBM	TIE	RIE
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

Bits	Access	Name	Description
18	R/W	PINTE	Peripheral Trailing Byte Interrupt Enable PINTE enables the peripheral trailing byte interrupt. **NOTE:** Clearing PINTE does not affect the current state of SSSR_x[PINT] or the ability of logic to set and clear SSSR_x[PINT]; it prevents the generation of the interrupt request. 0 = Peripheral trailing byte interrupts are disabled. The interrupt is masked and the state of SSSRx[PINT] is ignored by the interrupt controller. 1 = Peripheral trailing byte interrupt are enabled.
17	—	—	reserved
16	R/W	IFS	Invert Frame Signal 0 = SSPSFRMx polarity is determined by SSP format and PSP polarity bits. 1 = SSPSFRMx is inverted from the normal SSP frame signal (as defined by the SSP format and PSP polarity bits).
15	R/W	STRF	Select FIFO for EFWR (test mode bit only) Only when SSCR1_x[EFWR] is set, STRF selects whether the transmit or the receive FIFO is enabled for writes and reads. 0 = Transmit FIFO is selected for both writes and reads through SSDR_x 1 = Receive FIFO is selected for both writes and reads through SSDR_x

Table 8-7. SSCR1_1/2/3 Bit Definitions (Sheet 6 of 9)

Physical Address		SSP Controller
0x4100_0004 0x4170_0004 0x4190_0004	SSCR1_1 SSCR1_2 SSCR1_3	

	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
User Settings																																
Bit	TTELP	TTE	EBCEI	SCFR	ECRA	ECRB	SCLKDIR	SFRMDIR	RWOT	TRAIL	TSRE	RSRE	TINTE	PINTE	Reserved	IFS	STRF	EFWR	RFT				TFT				MWDS	SPH	SPO	LBM	TIE	RIE
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

Bits	Access	Name	Description
14	R/W	EFWR	Enable FIFO Write/Read (test mode bit) Enables test mode for the SSP port. When set, the SSP port enters a mode where whenever the CPU reads or writes to the SSP Data register, it reads and writes directly to either the transmit FIFO or the receive FIFO, depending on the programmed state of SSCR1_x[STRF]. In EFWR test mode, data is not transmitted on SSPTXDx, data input on SSPRXDx is not stored, and the busy and ROR bits have no effect. However, the Interrupt Test register is still functional. Using software, this mode can test whether or not the TX FIFO or the RX FIFO operates properly as a FIFO memory stack. Verify that the SSSR_x[CSS] bit has gone from set to clear before reading the TX FIFO. This bit must be cleared for normal operation. When SSCR1_x[STRF] is clear, writes to SSDR_x are performed on the Transmit FIFO, and reads from SSDR_x read back the data written to the TX FIFO in first-in-first-out order. When the STRF is set, writes to SSDR_x are performed on the RX FIFO, and reads from SSDR_x read back the data written to the RX FIFO in first-in-first-out order. 0 = FIFO write/read special function is disabled (normal SSP port operational mode) 1 = FIFO write/read special function is enabled.
13:10	R/W	RFT	Receive FIFO Threshold RFT sets the level at or above which the FIFO controller triggers a DMA service request (if enabled) and a CPU interrupt request (if enabled). This level must be set to the desired trigger threshold value minus 1. **NOTE:** Do not to set the value of RFT too high for the system; otherwise, the receive FIFO can overrun because of the bus latencies caused by other internal and external peripherals. This is especially important when using interrupts and polled modes that require a longer time to service.
9:6	R/W	TFT	Transmit FIFO Threshold TFT sets the level at or below which the FIFO controller triggers a DMA service request (if enabled) and a CPU interrupt request (if enabled). This level must be set to the desired trigger threshold value minus 1. **NOTE:** Do not set the value of TFT too low for the system; otherwise, the transmit FIFO can underrun because of the bus latencies caused by other internal and external peripherals. This is especially important when using interrupts and polled modes that require a longer time to service.

Table 8-7. SSCR1_1/2/3 Bit Definitions (Sheet 7 of 9)

Physical Address		SSP Controller
0x4100_0004 0x4170_0004 0x4190_0004	SSCR1_1 SSCR1_2 SSCR1_3	

Bit	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
User Settings																																
	TTELP	TTE	EBCEI	SCFR	ECRA	ECRB	SCLKDIR	SFRMDIR	RWOT	TRAIL	TSRE	RSRE	TINTE	PINTE	Reserved	IFS	STRF	EFWR	RFT				TFT				MWDS	SPH	SPO	LBM	TIE	RIE
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

Bits	Access	Name	Description
5	R/W	MWDS	Microwire Transmit Data Size MWDS selects between an eight bit or 16-bit size for the command word transmitted using the Microwire protocol. MWDS is ignored for all other frame formats. 0 = 8-bit command word is transmitted. 1 = 16-bit command word is transmitted.
4	R/W	SPH	Motorola SPI SSPSCLKx Phase SPH determines the phase relationship between SSPSCLKx and the SSPSFRMx when the Motorola SPI format is selected. When SPH is clear, SSPSCLKx remains in its Inactive/Idle state (as determined by the SSCR1_x[SPO] setting) for one full cycle after SSPSFRMx is asserted low at the beginning of a frame. SSPSCLKx continues to toggle for the rest of the frame and is then held in its Inactive state for one-half of an SSPSCLKx period before SSPSFRMx is deasserted high at the end of the frame. When SPH is set, SSPSCLKx remains in its inactive/idle state (as determined by the SSCR1_x[SPO] bit value) for one-half cycle after SSPSFRMx is asserted low at the beginning of a frame. SSPSCLKx continues to toggle for the remainder of the frame, and is then held in its inactive state for one full SSPSCLKx period before SSPSFRMx is de-asserted high at the end of the frame. The combination of the SSCR1_x[SPO] bit and SSCR1_x[SPH] bit settings determines when SSPSCLKx is active during the assertion of SSPSFRMx, and which SSPSCLKx edge transmits and receives data on the SSPTXDx and SSPRXDx pins. When SSCR1_x[SPO] and SSCR1_x[SPH] are programmed to the same value (both clear or both set), SSPTXDx is driven on the falling edge of SSPSCLKx, and SSPRXDx is latched on the rising edge of SSPSCLKx. When SSCR1_x[SPO] and SSCR1_x[SPH] are programmed to opposite values (one clear and the other set), SSPTXDx is driven on the rising edge of SSPSCLKx and SSPRXDx is latched on the falling edge of SSPSCLKx. **NOTE:** SPH is ignored for all data frame formats except for the Motorola SPI format. Figure 8-6 and Figure 8-7 show the timing for all four programming combinations of SSCR1_x[SPO] and SSCR1_x[SPH]. SSCR1_x[SPO] inverts the polarity of SSPSCLKx, and SSCR1_x[SPH] determines the phase relationship between SSPSCLKx and SSPSFRMx, shifting the SSPSCLKx one-half phase to the left or right during the assertion of SSPSFRMx. 0 = SSPSCLKx is inactive one cycle at the start of a frame and 1/2 cycle at the end of a frame. 1 = SSPSCLKx is inactive 1/2 cycle at the start of a frame and one cycle at the end of a frame.

Table 8-7. SSCR1_1/2/3 Bit Definitions (Sheet 8 of 9)

Physical Address		SSP Controller
0x4100_0004 0x4170_0004 0x4190_0004	SSCR1_1 SSCR1_2 SSCR1_3	

Bit	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
User Settings																																
	TTELP	TTE	EBCEI	SCFR	ECRA	ECRB	SCLKDIR	SFRMDIR	RWOT	TRAIL	TSRE	RSRE	TINTE	PINTE	Reserved	IFS	STRF	EFWR	RFT				TFT				MWDS	SPH	SPO	LBM	TIE	RIE
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

Bits	Access	Name	Description
3	R/W	SPO	Motorola SPI SSPSCLKx Polarity SPO selects the polarity of the inactive state of SSPSCLKx when the SPI protocol is selected. The programmed setting of SPO alone does not determine which SSPSCLKx edge transmits or receives data; SPO in combination with SSCR1_x[SPH] does. **NOTE:** SSCR1_x[SPO] is ignored for all data frame formats except for SPI protocol (SSCR0_x[FRF] = 0b00). 0 = SSPSCLKx is held low in the inactive or idle state when the SSP port is not transmitting/receiving data. 1 = SSPSCLKx is held high during the inactive or idle state.
2	R/W	LBM	Loop-Back Mode (test-mode bit only) LBM is a test-mode bit that enables and disables the ability of the SSP port transmit and receive logic to communicate. **NOTE:** The loop-back mode cannot be used with the Microwire protocol since this protocol uses half-duplex master-slave message passing. 0 = Normal SSP port operation is enabled. 1 = Output of transmit serial shifter is internally connected to the input of the receive serial shifter. SSPTXDx continues to function normally.
1	R/W	TIE	Transmit FIFO Interrupt Enable TIE enables the TX FIFO service request interrupt. **NOTE:** Clearing TIE does not affect the current state of SSSR_x[TFS] or the ability of the transmit-FIFO logic to set and clear SSSR_x[TFS]—it blocks only the generation of the interrupt request. Also, the state of TIE does not effect the generation of the transmit-FIFO DMA service request, which is asserted whenever SSSR_x[TFS] is set. 0 = TX FIFO level interrupt is disabled. The interrupt is masked and the state of SSSR_x[TFS] is ignored. 1 = TX FIFO level interrupt is enabled. Whenever SSSR_x[TFS] is set, an interrupt request is made to the interrupt controller.

Table 8-7. SSCR1_1/2/3 Bit Definitions (Sheet 9 of 9)

Physical Address		SSP Controller
0x4100_0004 0x4170_0004 0x4190_0004	SSCR1_1 SSCR1_2 SSCR1_3	

Bit	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
User Settings																																
	TTELP	TTE	EBCEI	SCFR	ECRA	ECRB	SCLKDIR	SFRMDIR	RWOT	TRAIL	TSRE	RSRE	TINTE	PINTE	Reserved	IFS	STRF	EFWR	RFT				TFT				MWDS	SPH	SPO	LBM	TIE	RIE
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

Bits	Access	Name	Description
0	R/W	RIE	Receive FIFO Interrupt Enable RIE enables the RX FIFO service request interrupt. **NOTE:** Clearing RIE does not affect the current state of SSSR_x[RFS] or the ability of the RX-FIFO logic to set and clear SSSR_x[RFS]—it blocks only the generation of the interrupt request. The state of RIE does not affect the generation of the RX-FIFO DMA service request, which is asserted whenever SSSR_x[RFS] is set. 0 = RX FIFO level interrupt is disabled. The interrupt is masked and SSSR_x[RFS] is ignored. 1 = RX FIFO level interrupt is enabled. Whenever SSSR_x[RFS] is set, an interrupt request is made to the interrupt controller.

8.5.3 SSP Programmable Serial Protocol Register (SSPSP_x)

SSPSP_x, shown in Table 8-8, contains bit fields to program the various programmable serial-protocol parameters.

The contents of SSPSP_x are ignored if PSP mode is not selected.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 8-8. SSPSP1/2/3 Bit Definitions (Sheet 1 of 2)

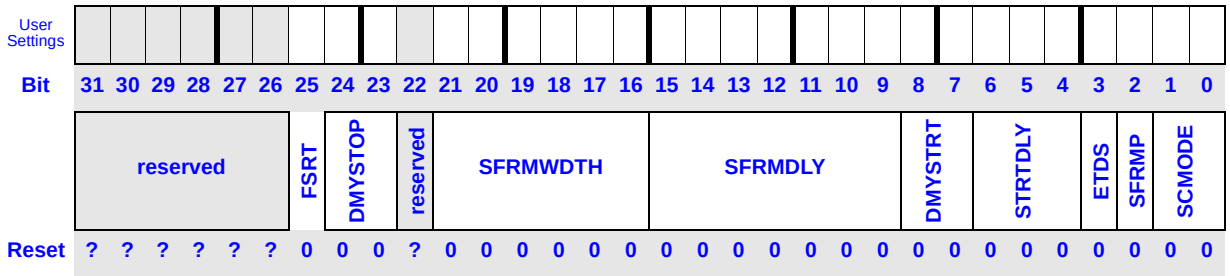
Physical Address			SSP Controller
0x4100_002C	SSPSP_1		
0x4170_002C	SSPSP_2		
0x4190_002C	SSPSP_3		

User Settings	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
Bit	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						FSRT	DMYSTOP		reserved	SFRMWDTH						SFRMDLY							DMYSTRT			STRTDLY			ETDS	SFRMP	SCMODE	
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

Bits	Access	Name	Description
31:26	—	—	reserved
25	R/W	FSRT	Frame Sync Relative Timing 0 = Next frame is asserted after the end of the T4 timing 1 = Next frame is asserted with the LSB of the previous frame **NOTE:** When FSRT is set, SSPSFRMx corresponding to the next sample is asserted during the transmission of the LSB from the current sample (see Figure 8-12).
24:23	R/W	DMYSTOP	Dummy Stop DMYSTOP determines the number of cycles that SSPSCLKx is active following the last bit (bit 0) of transmitted data (SSPTXDx) or received data (SSPRXDx). The value must be from 0 to 3. DMYSTOP must be cleared when PSP format is used in network mode and/or when FSRT is set.
22	—	—	reserved
21:16	R/W	SFRMWDTH	Serial Frame Width SFRMWDTH determines the number of SSPSCLKx cycles that SSPSFRMx is active. The programmed value must not be asserted past the end of DMYSTOP (T4 in Figure 8-10). The value must be from 1 to 44. In PSP slave mode (SSCR1_x[SFRMDIR] is set), SFRMWDTH is ignored. The incoming SSPSFRMx must be asserted for a duration of at least one SSPSCLKx cycle for each sample. The incoming SSPSFRMx and first data bit of the sample can be asserted at the same time. Between samples, the incoming SSPSFRMx must be de-asserted for a duration of at least one SSPSCLKx cycle.
15:9	R/W	SFRMDLY	Serial Frame Delay SFRMDLY determines the number of half SSPSCLKx cycles that SSPSFRMx is delayed from the start of the transfer to the time SSPSFRMx is asserted. The value must be from 0 to 88.
8:7	R/W	DMYSTRT	Dummy Start DMYSTRT determines the number of SSPSCLKx cycles after STRTDLY and before transmitted data (SSPTXDx) or received data (SSPRXDx).

Table 8-8. SSPSP1/2/3 Bit Definitions (Sheet 2 of 2)

Physical Address		SSP Controller
0x4100_002C 0x4170_002C 0x4190_002C	SSPSP_1 SSPSP_2 SSPSP_3	



Bits	Access	Name	Description
6:4	R/W	STRTDLY	Start Delay STRTDLY determines the number of cycles that SSPSCLKx remains in its Idle state between data transfers. The STRTDLY field must be cleared if the SSPSCLKENx, SSCR1_x[ECRA], or SSCR1_x[ECRB] clock enables are used. The STRTDLY field must be cleared whenever SSPSCLKx or SSPSFRMx is configured as an input (SSCR1_x[SCLKDIR] or SSCR1_x[SFRMDIR] are set). The value must be from 0 to 7.
3	R/W	ETDS	End-of-Transfer Data State ETDS determines the state of SSPTXDx at the end of a transfer. When cleared, the state of SSPTXDx is forced low after the LSB of the frame is sent and remains low through the next idle period. When set, the state of SSPTXDx retains the value of the LSB through the next idle period. **NOTE:** ETDS has no effect if SSCR1_x[TTE] is set. ETDS bit has no effect when configured in TI Synchronous Serial Protocol. 0 = Low 1 = Last Value <Bit 0>
2	R/W	SFRMP	Serial Frame Polarity SFRMP determines the active state of SSPSFRMx. In idle mode or when the SSP port is disabled, SSPSFRMx is in its inactive state. In slave mode (SSCR1_x[SFRMDIR] is set), SFRMP indicates the polarity of the incoming SSPSFRMx. 0 = SSPSFRMx is active low. 1 = SSPSFRMx is active high.
1:0	R/W	SCMODE	Serial Bit-Rate Clock Mode SCMODE selects one of four serial clock modes when PSP format is used (SSCR0_x[FRF] = 0b11). Its operation is similar to how SSCR1_x[SPO] and SSCR1_x[SPH] together determine the idle state of SSPSCLKx and on which edges data is driven and sampled. 0b00 = Data Driven (Falling), Data Sampled (Rising), Idle State (Low) 0b01 = Data Driven (Rising), Data Sampled (Falling), Idle State (Low) 0b10 = Data Driven (Rising), Data Sampled (Falling), Idle State (High) 0b11 = Data Driven (Falling), Data Sampled (Rising), Idle State (High) **NOTE:** For all selections of SCMODE, the Idle state is high impedance when SSCR1_x[TIE] is set SSPSP_x[SCMODE] must be 0b00 when operating with TI Format (SSCR0_x[FRF] = 0b01)

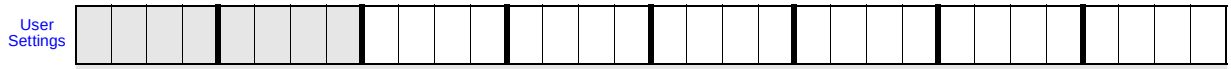
8.5.4 SSP Time-Out Register (SSTO_x)

SSTO_x, shown in Table 8-9, specifies the time-out value to signal a period of inactivity within the receive FIFO.

SSTO_x are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 8-9. SSTO_1/2/3 Bit Definitions

Physical Address		SSTO_1 SSTO_2 SSTO_3	SSP Controller
0x4100_0028 0x4170_0028 0x4190_0028			



Bit	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								TIMEOUT																							
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

Bits	Access	Name	Description
31:24	—	—	reserved
23:0	R/W	TIMEOUT	Time-Out Value Sets the time-out interval. When the TIMEOUT value is cleared, no time-out occurs and SSSR_x[TINT] is not set. The time-out interval is given by the equation: (Time-out Interval) = (TIMEOUT value) / (Peripheral Clock Frequency)

8.5.5 SSP Interrupt Test Register (SSITR_x)

SSITR_x, shown in Table 8-10, contains bit fields used for testing purposes only.

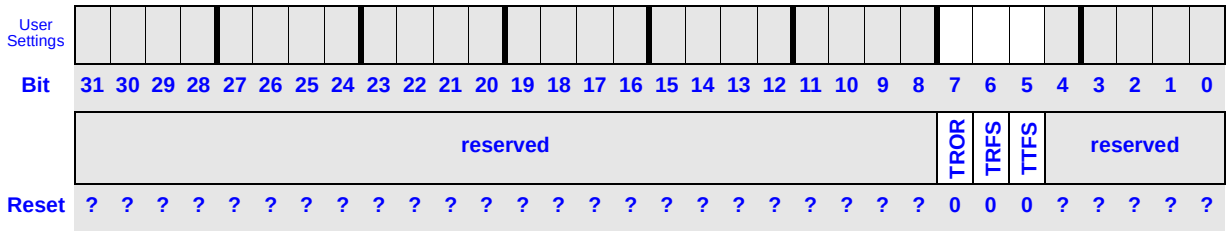
Setting bits in this register causes the SSP port controller to generate interrupts and DMA requests, if they are enabled, which is useful in testing port functionality.

Setting any of these bits also causes corresponding status bits to be set in SSSR_x. The interrupt or DMA service request, caused by the setting of one of these bits, remains active until the bit is cleared. This register must be 0 for normal operation.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 8-10. SSITR1/2/3 Bit Definitions

Physical Address		SSITR_1 SSITR_2 SSITR_3	SSP Controller
0x4100_000C 0x4170_000C 0x4190_000C			



Bits	Access	Name	Description
31:8	—	—	reserved
7	R/W	TROR	Test RX FIFO Overrun 0 = No receive FIFO overrun DMA service request is generated. 1 = Generates a non-maskable interrupt to the CPU. No DMA request is generated. Write 0b0 to clear.
6	R/W	TRFS	Test RX FIFO Service Request 0 = No receive FIFO DMA service request is generated. 1 = Generates a non-maskable interrupt to the CPU and a DMA request for the RX FIFO. Write 0b0 to clear.
5	R/W	TTFS	Test TX FIFO Service Request 0 = No transmit FIFO DMA service request is generated. 1 = Generates a non-maskable interrupt to the CPU and a DMA request for the TX FIFO. Write 0b0 to clear.
4:0	—	—	reserved

8.5.6 SSP Status Register (SSSR_x)

SSSR_x, shown in Table 8-11, contains bit fields that signal overrun errors and the transmit and receive FIFO DMA service requests. Each of these hardware-detected events signal an interrupt request to the interrupt controller. The status register also contains flags that indicate:

- When the SSP port is actively transmitting data
- When the TX FIFO is not full
- When the RX FIFO is not empty

One interrupt signal is sent to the interrupt controller for each SSP port. These events can cause an interrupt:

- End-of-chain
- Receiver time-out
- Peripheral trailing byte
- RX FIFO overrun
- RX FIFO request
- TX FIFO request

An interrupt is signaled as long as the bits are set. The interrupt clears when the bits are cleared. Read and write bits are called *status* bits (status bits are referred to as *sticky* and once set by hardware, they must be cleared by software); read-only bits are called *flags*. Writing 0b1 to a sticky status bit clears it; writing 0b0 has no effect. Read-only flags are set and cleared by hardware; writes have no effect. The reset state of read-write bits is zero and all bits return to their reset state when SSCR0_x[SSE] is cleared. Additionally, some bits that cause interrupts have corresponding mask bits in the Control registers.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 8-11. SSSR1/2/3 Bit Definitions (Sheet 1 of 5)

Physical Address
0x4100_0008
0x4170_0008
0x4190_0008

SSSR_1
SSSR_2
SSSR_3

SSP Controller

User Settings

Bit	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								BCE	CSS	TUR	EOC	TINT	PINT	reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	reserved	
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	?	?	1	1	1	1	0	0	0	0	0	0	0	0	0	1	?	?

Bits	Access	Name	Description
31:24	—	—	reserved
23	R/W†	BCE	Bit Count Error BCE is a read-write status bit that indicates that the SSP port has detected that SSPSFRMx has been asserted at an incorrect time. This causes an interrupt (if enabled by SSCR1_1[BCEI]). When set, BCE indicates that an error occurred and that to get re-synchronized to the master device, the SSP port may disregard both the sample that had SSPDFRMx re-asserted in the middle of it AND the next sample. **NOTE:** BCE does not operate in Motorola SPI mode. **NOTE:** To clear BCE, write 0b1 to it. 0 = The SSP port has not experienced a bit count error 1 = SSPSFRMx has been asserted when the bit counter was not 0
22	R	CSS	Clock Synchronization Status CSS indicates that the SSP port is busy synchronizing the control signals into the SSPSCLKx domain. Software only needs to check CSS when the SSP port is a slave to SSPSFRMx. Software must wait until CSS is cleared before allowing an external device to assert SSPSFRMx. 0 = The SSP port is ready for slave clock operations 1 = The SSP port is currently busy synchronizing slave mode signals

Table 8-11. SSSR1/2/3 Bit Definitions (Sheet 2 of 5)

Physical Address		SSP Controller
0x4100_0008 0x4170_0008 0x4190_0008	SSSR_1 SSSR_2 SSSR_3	

User Settings																															
Bit	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								BCE	CSS	TUR	EOC	TINT	PINT	reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	reserved	
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	?	?	1	1	1	1	0	0	0	0	0	0	0	0	0	1	?	?

Bits	Access	Name	Description
21	R/W†	TUR	Transmit FIFO Underrun TUR indicates that the transmitter tried to send data from the TX FIFO when the TX FIFO was empty. When set, an interrupt is generated to the CPU (that can be locally masked by the SSCR0_x[TIM] bit). Setting TUR does not generate any DMA service request. TUR remains set until cleared by software writing 0b1 to it (which also resets its interrupt request). Writing 0b0 to TUR has no effect. TUR can be set when the SSP port is a slave to SSPSFRMx (SSCR1_x[SFRMDIR] set), or if the SSP port is a master to SSPSFRMx and the SSP port is in network mode. TUR is not set if the SSP port is in receive-without-transmit mode (SSCR1_x[RWOT] set). 0 = TX FIFO has not experienced an underrun 1 = Transmitter tried to send data from the TX FIFO when the FIFO was empty, an interrupt is signaled
20	R/W†	EOC	End Of Chain EOC indicates that the DMA has signaled an end of chain. The end- of-chain event indicates that the DMA descriptor for the RX FIFO is ending. This event requires software intervention if data remains in the RX FIFO. **NOTE:** To clear EOC, write 0b1 to it. 0 = DMA has not signaled an end of chain condition. 1 = DMA has signaled an end-of chain condition. An EOC interrupt is generated only when there are trailing bytes left (the PINT bit is set) or there are no trailing bytes (the TINT bit is set). EOC bit is always set, but does not generate an interrupt if neither of the these conditions are met.
19	R/W†	TINT	Time-Out Interrupt TINT indicates that the RX FIFO has been idle (no samples received) for the period of time defined by the value programmed within SSTO_x. This interrupt can be masked by SSCR1_x[TINTE]. **NOTE:** To clear TINT, write 0b1 to it. 0 = No receiver time-out has occurred 1 = Receiver time-out has occurred
18	R/W†	PINT	Peripheral Trailing Byte Interrupt PINT indicates that a DMA end-of-chain event has occurred and there is data within the RX FIFO. This event requires the CPU or DMA to transfer the remaining bytes from the RX FIFO (see Section 8.4.2.3). This interrupt can be masked by SSCR1_x[PINTE]. **NOTE:** To clear PINT, write 0b1 to it. 0 = No peripheral trailing byte interrupt is pending. 1 = Peripheral trailing byte interrupt is pending

Table 8-11. SSSR1/2/3 Bit Definitions (Sheet 3 of 5)

Physical Address		SSSR_1 SSSR_2 SSSR_3	SSP Controller
0x4100_0008 0x4170_0008 0x4190_0008			

User Settings

Bit	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								BCE	CSS	TUR	EOC	TINT	PINT	reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	reserved	
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	?	?	1	1	1	1	0	0	0	0	0	0	0	0	0	1	?	?

Bits	Access	Name	Description
17:16	—	—	reserved
15:12	R	RFL	RX FIFO Level RFL is the number of valid entries (minus 1) currently in the RX FIFO. **NOTE:** When the value of 0xF is read, the RX FIFO is either empty or full and programmers must refer to the RNE bit.
11:8	R	TFL	TX FIFO Level TFL is the number of valid entries currently in the TX FIFO. **NOTE:** When the value of 0x0 is read, the TX FIFO is either empty or full and programmers must refer to the TNF bit.
7	R/W†	ROR	RX FIFO Overrun ROR indicates that the Receive logic attempted to place data into the RX FIFO after it had been completely filled. When new data is received, ROR is asserted and the newly received data is discarded. This process is repeated for all new data received until at least one empty RX FIFO location exists. When set, an interrupt is generated to the CPU that can be locally masked by the SSCR0_x[RIM} bit. Setting ROR does not generate any DMA service request. Clearing ROR resets its interrupt request. **NOTE:** To clear ROR, write 0b1 to it. 0 = RX FIFO has not experienced an overrun 1 = Attempted data write to a full RX FIFO, request an interrupt
6	R	RFS	Receive FIFO Service A RFS request indicates that the RX FIFO requires service to prevent an overrun. RFS is set when the number of valid entries in the RX FIFO is equal to or greater than the RX FIFO trigger threshold. RFS is cleared when the RX FIFO has fewer entries than the trigger threshold. When RFS is set, an interrupt is generated if SSCR1_x[RIE] is set. When RFS is set, a DMA service request is generated if SSCR1_x[RSRE] is set. After the CPU or DMA reads the RX FIFO such that it has fewer entries than the value of SSCR1_x[RFT], RFS (and the service request and/or interrupt) is automatically cleared. SSCR1_x[RSRE] and SSCR1_x[RIE] must not both be set. 0 = RX FIFO level is less than its trigger threshold or the SSP port is disabled 1 = RX FIFO level is equal to or above its trigger threshold, an interrupt or DMA service request is generated.

Table 8-11. SSSR1/2/3 Bit Definitions (Sheet 4 of 5)

Physical Address		SSP Controller
0x4100_0008 0x4170_0008 0x4190_0008	SSSR_1 SSSR_2 SSSR_3	

	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
User Settings																																
Bit	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								BCE	CSS	TUR	EOC	TINT	PINT	reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	reserved	
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	?	?	1	1	1	1	0	0	0	0	0	0	0	0	0	1	?	?

Bits	Access	Name	Description
5	R	TFS	Transmit FIFO Service A TFS request indicates that the TX FIFO requires service to prevent an underrun. TFS is set when the number of valid entries in the TX FIFO is equal to or less than the TX FIFO trigger threshold. TFS is cleared when the TX FIFO has more entries than the trigger threshold. When TFS is set, an interrupt is generated if SSCR1_x[TIE] is set. When TFS is set, a DMA service request is generated if SSCR1_x[TSRE] is set. After the CPU or DMA fills the TX FIFO such that it has more entries than the value of SSCR1_x[TFT], TFS (and the service request and/or interrupt) is automatically cleared. SSCR1_x[TSRE] and SSCR1_x[TIE] must not both be set. 0 = TX FIFO level exceeds its threshold (TFT + 1) or the SSP port is disabled 1 = TX FIFO level is at or below its trigger threshold (TFT + 1), an interrupt or DMA service request is generated.
4	R	BSY	Busy BSY is automatically set when the SSP port is actively transmitting and/or receiving data and BSY is automatically cleared when the SSP port is idle or disabled. BSY does not generate an interrupt. **NOTE:** When the SSP port is a master of a clock, software determines if the SSP port is active by monitoring SSSR_x[TFL] and SSSR_x[BSY]. If the SSP port is a slave to a clock, software determines if the SSP port is active by monitoring SSSR_x[TFL], SSSRx[RFL] and SSSR_x[BSY] along with the SSTO_x register. Also, using the time-out feature (the SSTO_x register and SSCR1_x[TRAIL]) to handle trailing bytes provides an indication of when the master has completed sending data. 0 = SSP port is idle or disabled 1 = SSP port is actively transmitting or receiving data
3	R	RNE	RX FIFO Not Empty RNE indicates that the RX FIFO contains one or more entries of valid data. RNE is automatically cleared when the RX FIFO no longer contains any valid data. This bit does not generate an interrupt. When using programmed I/O, RNE can be polled to remove remaining bytes of data from the RX FIFO since CPU interrupt requests are made only when the RX FIFO trigger threshold has been met or exceeded. 0 = RX FIFO is empty. 1 = RX FIFO is not empty.

Table 8-11. SSSR1/2/3 Bit Definitions (Sheet 5 of 5)

Physical Address		SSSR_1 SSSR_2 SSSR_3	SSP Controller
0x4100_0008 0x4170_0008 0x4190_0008			

User Settings

Bit	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								BCE	CSS	TUR	EOC	TINT	PINT	reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	reserved	
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	?	?	1	1	1	1	0	0	0	0	0	0	0	0	0	1	?	?

Bits	Access	Name	Description
2	R	TNF	TX FIFO Not Full TNF indicates that the TX FIFO contains one or more entries that do not contain valid data. TNF is automatically cleared when the TX FIFO is completely full. TNF does not generate an interrupt. When using programmed I/O, TNF can be polled to fill the TX FIFO beyond its trigger threshold. 0 = TX FIFO is full 1 = TX FIFO is not full
1:0	—	—	reserved

† Write 0b1 to clear this bit.

8.5.7 SSP Data Register (SSDR_x)

SSDR_x, shown in Table 8-12, is a single-address location that is accessed by both read and write data transfers. Each SSDR register represents two physical registers: the first register provides temporary storage for data on its way to the TX FIFO, while the second register provides temporary storage for data coming from the RX FIFO.

As the CPU or DMA accesses the SSDR_x registers, FIFO control logic transfers data automatically between the registers and FIFOs as fast as the CPU or DMA moves it. Data in the FIFOs shift up or down to accommodate new word(s), unless attempting a write to a full transmit FIFO. Status bits (such as SSSR_x[TFL, RFL, TNF, RNE]) show if the FIFO is full, above the programmable trigger threshold, below the programmable trigger threshold, or empty.

For transmit data, SSDR_x can be loaded (written) by the processor (using programmed I/O or DMA) anytime the TX FIFO falls below its trigger threshold.

When a data size of less than 32 bits is selected, do not left-justify transmit data that is written to SSDR_x. Transmit logic left-justifies the data and ignores any unused bits. Received data of less than 32-bits is automatically right-justified in the RX FIFO.

When the SSP port is programmed for the Microwire protocol and the size of the transmit data is eight bits (SSCR1_x[MWDS] cleared), the most significant 24-bits are ignored. Similarly, if the size for the transmit data is 16-bits (SSCR1_x[MWDS] set), the most significant 16-bits are ignored. SSCR0_x[DSS] controls the receive data size.

Both the TX and RX FIFOs are cleared when the SSP port is reset, or by clearing SSCR0_x[SSE].

Table 8-12. SSDR1/2/3 Bit Definitions

Physical Address
0x4100_0010
0x4170_0010
0x4190_0010

SSDR_1
SSDR_2
SSDR_3

SSP Controller

	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
User Settings																																
Bit	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
	DATA																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:0	R/W	DATA	Data to be written to/read from the TX/RX FIFO, respectively.

8.5.8 SSP TX Time Slot Active Register (SSTSA_x)

SSTSA_x are read-write registers that indicate in which time slot the SSP port transmits data. SSTSA_x are ignored if the SSP port is not in network mode (SSCR0_x[MOD] = 1). See Figure 8-3 for an example of using time slots only when in network mode.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 8-13. SSTSA1/2/3 Bit Definitions

Physical Address
0x4100 0030
0x4170 0030
0x4190 0030

SSTSA_1
SSTSA_2
SSTSA_3

SSP Controller

	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
User Settings																																
Bit	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																								TTSA							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	Reserved
7:0	R/W	TTSA	TX Time Slot Active Only if the SSP port is in network mode, the 8 TTSA bits indicate in which of 8 associated time slots the SSP port transmits. Each TTSA bit selects one time slot, respectively. Time slot bits beyond the SSCR0_x[FRDC] value are ignored (if SSCR0_x[FRDC] = 0b011 to select 4 time slots, then TTSA bits 7:4 are ignored). If SSCR1_x[TTE] is set, then the SSP port causes SSPTXD to be high impedance during time slots where associated TTSA bits are programmed to 0. 0 = SSP port does not transmit data in this time slot 1 = SSP port transmits data in this time slot

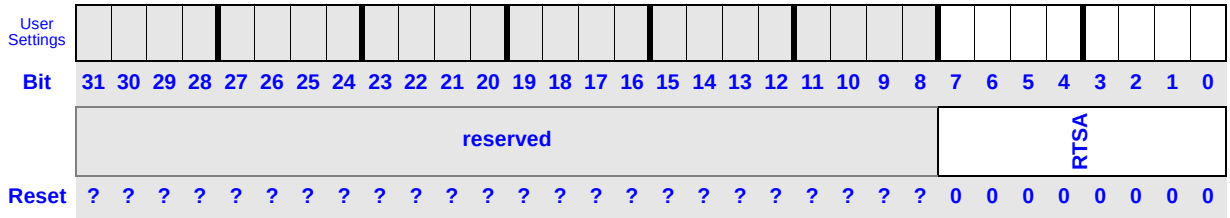
8.5.9 SSP RX Time Slot Active Register (SSRSA_x)

SSRSA_x are read-write registers that indicate in which time slots the SSP port receives data. SSRSA_x are ignored if the SSP port is not in network mode. See Figure 8-3 for an example of using time slots only when in network mode.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 8-14. SSRSA1/2/3 Bit Definitions

Physical Address	SSRSA_1 SSRSA_2 SSRSA_3	SSP Controller
0x4100 0034 0x4170 0034 0x4190 0034)		



Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	RTSA	RX Time Slot Active Only if the SSP port is in network mode, the 8 RTSA bits indicate in which of 8 associated time slots the SSP port receives data. Each RTSA bit selects one time slot, respectively. Time slot bits beyond the SSCR0_x[FRDC] value is ignored (if SSCR0_x[FRDC] = 0b011 to select 4 time slots, then RTSA bits 7:4 are ignored). 0 = SP port does not receive data in this time slot 1 = SSP port receives data in this time slot

8.5.10 SSP Time Slot Status Register (SSTSS_x)

These registers indicate which time slot the SSP port is currently in. SSTSS_x are ignored when the SSP port is not in network mode.

These are read-only registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 8-15. SSTS1/2/3 Bit Definitions

Physical Address	SSTSS_1	SSP Controller
0x4100 0038)	SSTSS_2	
0x4170 0038	SSTSS_3	
0x4190 0038		

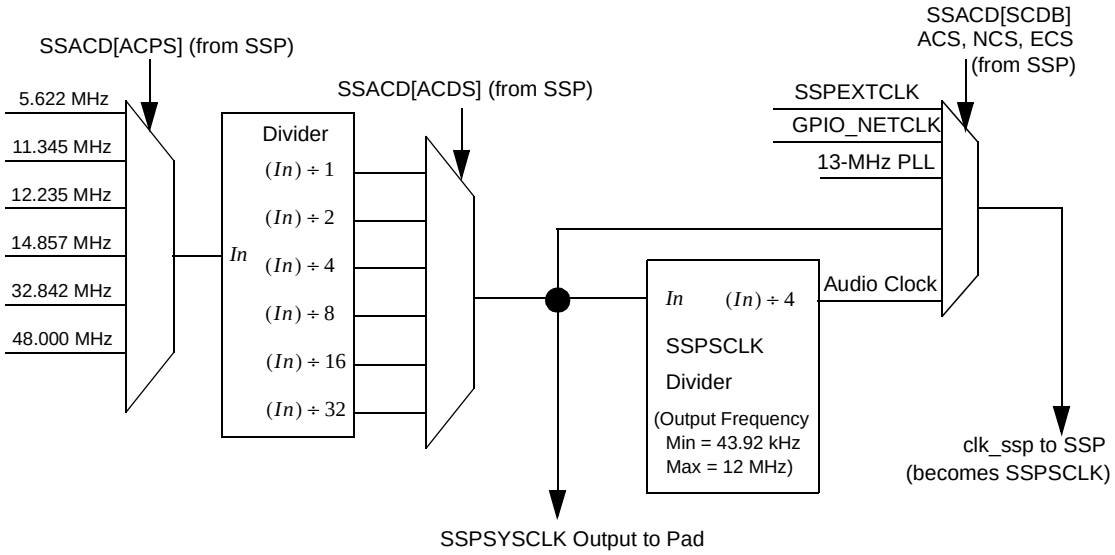
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NMBSY	reserved																												TSS		
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0

Bits	Access	Name	Description
31	R	NMBSY	Network Mode Busy Only if the SSP port is in network mode, NMBSY indicates when the SSP port is in the middle of a frame. NMBSY can be used by software when a clean shutdown of the SSP port is needed. Software must (1) ensure that the TX FIFO is either empty or will be empty at the end of the next frame, (2) deactivate the TX DMA requests, (3) clear the SSCR0_x[MOD] bit; then (4) poll NMBSY until it is 0 before disabling the SSP port (by clearing the SSCR0_x[SSE] bit). When the SSP port is a master of SSPSFRMx, NMBSY is set. If the SSP port is a slave to SSPSFRMx, NMBSY is set only if the current frame (number of bits per sample number of time slots per frame) has not expired since SSPSFRMx was asserted. 0 = No SSPSFRMx is currently asserted (network mode only) 1 = SSPSFRMx is currently asserted (network mode only)
30:3	—	—	reserved
2:0	R	TSS	Time Slot Status Only if the SSP port is in network mode, the 3-bit TSS value indicates which time slot the SSP port is in. Due to synchronization delays between clock domains, the TSS value is a delayed version of the actual time slot.

8.5.11 SSP Audio Clock Divider Register (SSACD_x)

SSACD_x selects which clock frequency is sent to the SSP port and then to SSPSYSCLKx and SSPSCLKx. If SSCR0_x[SCR] is not 0, there is no guaranteed phase relationship between SSPSYSCLKx and SSPSCLKx. The SSPSYSCLKx frequency (see Figure 8-20) is calculated by dividing the chosen PLL output clock frequency (SSACD_x[ACPS]) by the chosen divider (SSACD_x[ACDS]). SSPSYSCLKx is then divided by 4 (or by 1) to get SSPSCLKx (see Figure 8-20). The SSPSFRMx frequency is calculated by dividing SSPSCLKx by the multiply-product of data size (SSCR0_x[EDSS, DSS] values) times the number of time slots being used (SSCR0_x[FRDC] value).

Figure 8-20. Audio Clock Selection



These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 8-16. SSACD1/2/3 Bit Definitions

Physical Address
0x4100 003C
0x4170 003C)
0x4190 003C

SSACD_1
SSACD_2
SSACD_3

SSP Controller

	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
User Settings																																
Bit	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																									ACPS			SCDB	ACDS		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description	
31:7	—	—	Reserved	
6:4	R/W	ACPS	Audio Clock PLL Select The ACPS value indicates which PLL output clock is sent to the clock divider. See Figure 8-20, Table 8-17, and Table 8-18. Some combinations of ACPS and ACDS are not valid (see Table 8-17).	
			ACPS Value	PLL Output Frequency
			0b000	5.622 MHz
			0b001	11.345 MHz
			0b010	12.235 MHz
			0b011	14.857 MHz
			0b100	32.842 MHz
			0b101	48.000 MHz
			0b110	Reserved
			0b111	Reserved
3	R/W	SCDB	SSPSYSCLK Divider Bypass If SCDB is set and SSCR0_x[ACS] is set, SSPSYSCLKx is divided by 1 to become SSPSCLKx. If SCDB is cleared and SSCR0_x[ACS] is set), SSPSYSCLKx is divided by 4 to become SSPSCLKx. If SSCR0_x[ACS] is cleared, SCDB has no effect. 0 = SSPSYSCLKx is divided by 4 to become SSPSCLKx 1 = SSPSYSCLKx is divided by 1 to become SSPSCLKx	
2:0	R/W	ACDS	Audio Clock Divider Select The ACDS value indicates which divider creates SSPSYSCLKx. See Figure 8-20, Table 8-17, and Table 8-18. Some combinations of ACPS and ACDS are not valid (see Table 8-17).	
			ACDS Value	Clock Divider Value
			0b000	1
			0b001	2
			0b010	4
			0b011	8
			0b100	16
			0b101	32
			0b110	Reserved
			0b111	Reserved

Table 8-17. SSPSYSCLKx Frequency Selection

PLL Output Frequency	ACPS Value	ACDS Value					
		1	2	4	8	16	32
5.622MHz	0b000	5.622 MHz	2.811 MHz	1.405 MHz	702.7 kHz	3.514 kHz	175.7 kHz
11.345MHz	0b001	11.345 MHz	5.673 MHz	2.836 MHz	1.418 MHz	709.1 kHz	354.5 kHz
12.235MHz	0b010	12.235 MHz	6.118 MHz	3.059 MHz	1.529 MHz	764.7 kHz	382.4 kHz
14.857MHz	0b011	14.857MHz	7.429 MHz	3.714 MHz	1.857 MHz	928.6 kHz	464.3 kHz
32.842MHz	0b100	not valid	16.421MHz	8.211 MHz	4.105 MHz	2.053 MHz	1.026 MHz
48.00MHz	0b101	not valid	not valid	12.00 MHz	6.000 MHz	3.000 MHz	1.500 MHz

Table 8-18. PLL Output Frequency and Divider Selection (Selected Time Slots and Data Sizes)

PLL Output Frequency	ACPS Value	SCDB	Divider Value (ACDS choice)												Actual SSPSFRMx Frequency	Closest Standard for SSPSFRMx Frequency
			# of Time Slots for 8 bits/sample				# of Time Slots for 16 bits/sample				# of Time Slots for 32 bits/sample					
			1	2	4	8	1	2	4	8	1	2	4	8		
12.235 MHz	0b010	0	8	4	2	1	4	2	1	-	2	1	-	-	47.79 kHz	48.00 kHz
11.345 MHz	0b001	0	8	4	2	1	4	2	1	-	2	1	-	-	44.32 kHz	44.10 kHz
5.622 MHz	0b000	0	8	4	2	1	4	2	1	-	2	1	-	-	21.96 kHz	22.05 kHz
32.842 MHz	0b100	0	-	32	16	8	32	16	8	4	16	8	4	2	16.04 kHz	16.00 kHz
5.622 MHz	0b000	0	16	8	4	2	8	4	2	1	42	2	1	-	10.98 kHz	11.025 kHz
11.345 MHz	0b001	0	-	-	-	-	-	-	-	-	-	-	-	1	11.08 kHz	11.025 kHz
32.842 MHz	0b100	0	-	-	32	16	-	32	16	8	32	16	8	4	8.02 kHz	8.00 kHz
12.235 MHz	0b010	1	-	-	-	-	-	-	-	2	-	-	2	1	47.79 kHz	48.00 kHz
11.345 MHz	0b001	1	-	-	-	-	-	-	-	2	-	-	2	1	44.32 kHz	44.10 kHz
5.622 MHz	0b000	1	-	-	-	-	-	-	-	2	-	-	2	1	21.96 kHz	22.05 kHz

Note: Table 8-18 shows the recommended divider and PLL clock selection to approximate standard frame frequencies for a selected combination of bit sizes and time slots. Use the following formulas to calculate other combinations (use the second equation if SSPSYSCLKx is to be 4x SSPSCLKx):

$$TimeSlots \times BitsPerSample \times StandardFrequency = SSPSCLKFrequency$$
$$SSPSCLKFrequency \times 4 = SSPSYSCLKFrequency$$

Choose divider and PLL clock output to approximate SSPSYSCLK frequency

8.6 Register Summary

Table 8-19 summarizes the SSP port registers and their physical addresses.

Table 8-19. SSP Register Summary (Sheet 1 of 2)

Physical Address	Name	Description	Page
0x4100_0000	SSCR0_1	SSP 1 Control register 0	8-25
0x4100_0004	SSCR1_1	SSP 1 Control register 1	8-30
0x4100_0008	SSSR_1	SSP 1 Status register	8-43

Table 8-19. SSP Register Summary (Sheet 2 of 2)

Physical Address	Name	Description	Page
0x4100_000C	SSITR_1	SSP 1 Interrupt Test register	8-42
0x4100_0010	SSDR_1	SSP 1 Data Write register/Data Read register	8-48
0x4100_0014–0x4100_0024	—	reserved	
0x4100_0028	SSTO_1	SSP 1 Time-Out register	8-41
0x4100_002C	SSPSP_1	SSP 1 Programmable Serial Protocol	8-39
0x4100_0030	SSTSA_1	SSP1 TX Timeslot Active register	8-48
0x4100_0034	SSRSA_1	SSP1 RX Timeslot Active register	8-49
0x4100_0038	SSTSS_1	SSP1 Timeslot Status register	8-50
0x4100_003C	SSACD_1	SSP1 Audio Clock Divider register	8-51
0x4100_0040–0x416F_FFFC	—	reserved	
0x4170_0000	SSCR0_2	SSP2 Control register 0	8-25
0x4170_0004	SSCR1_2	SSP 2 Control register 1	8-30
0x4170_0008	SSSR_2	SSP 2 Status register	8-43
0x4170_000C	SSITR_2	SSP 2 Interrupt Test register	8-42
0x4170_0010	SSDR_2	SSP 2 Data Write register/Data Read register	8-48
0x4170_0014–0x4170_0024	—	reserved	
0x4170_0028	SSTO_2	SSP 2 Time-Out register	8-41
0x4170_002C	SSPSP_2	SSP 2 Programmable Serial Protocol	8-39
0x4170_0030	SSTSA_2	SSP2 TX Timeslot Active register	8-48
0x4170_0034	SSRSA_2	SSP2 RX Timeslot Active register	8-49
0x4170_0038	SSTSS_2	SSP2 Timeslot Status register	8-50
0x4170_003C	SSACD_2	SSP2 Audio Clock Divider register	8-51
0x4170_0040–0x418F_FFFC	—	reserved	
0x4190_0000	SSCR0_3	SSP 3 Control register 0	8-25
0x4190_0004	SSCR1_3	SSP 3 Control register 1	8-30
0x4190_0008	SSSR_3	SSP 3 Status register	8-43
0x4190_000C	SSITR_3	SSP 3 Interrupt Test register	8-42
0x4190_0010	SSDR_3	SSP 3 Data Write register/Data Read register	8-48
0x4190_0014–0x4190_0024	—	reserved	
0x4190_0028	SSTO_3	SSP 3 Time-Out register	8-41
0x4190_002C	SSPSP_3	SSP 3 Programmable Serial Protocol	8-39
0x4190_0030	SSTSA_3	SSP TX Timeslot Active register	8-48
0x4190_0034	SSRSA_3	SSP RX Timeslot Active register	8-49
0x4190_0038	SSTSS_3	SSP Timeslot Status register	8-50
0x4190_003C	SSACD_3	SSP Audio Clock Divider register	8-51
0x4190_0040–0x419f_FFFC	—	reserved	

I²C Bus Interface Unit 9

This chapter describes the Inter-Integrated Circuit (I^2C) bus interface unit, including its operational modes and setup. The PXA27x processor has two I^2C peripherals: the standard I^2C interface and the power-manager interface (a subset of the standard I^2C interface).

The I^2C bus is a true multi-master bus including collision detection and arbitration. For full details of I^2C bus operation, refer to the *I^2C-Bus Specification*, listed in Table 1-1, "Supplemental Documentation" on page 1-3.

9.1 Overview

The serial I^2C bus has a two-pin interface. The serial data and address (SDA) data pin serves I/O functions, and the serial clock line (SCL) clock pin controls and references the I^2C bus. The I^2C interface allows the PXA27x processor to serve as a master and slave device on the I^2C bus.

The I^2C interface enables the PXA27x processor to communicate with I^2C peripherals and microcontrollers for system-management functions. The I^2C bus requires a minimum of hardware to relay status, reliability, and control information between devices.

The I^2C interface resides on the processor internal bus as a peripheral. A buffered interface provides access to data transmitted and received over the I^2C bus. A set of memory-mapped registers relays control and status information.

Note: The I^2C interface does not support the hardware general call, 10-bit addressing, high-speed mode (HS-mode, 3.4 Mbits/s), or CBUS compatibility.

The main differences between the power manager and the standard I^2C interfaces are the register addresses, summarized in Section 9.6 and the speed of operation of the interfaces (see Section 9.2). The power-manager I^2C interface is optimized for connection to the external voltage regulator only (see Section 3.7.1, "Power Manager I^2C and Restrictions" on page 3-55). Except for the ICCR fixed selections, the power-manager I^2C interface is a full-function I^2C interface capable of all normal operations, including master and slave, receive, and transmit operations. See also Section 3.7.1.1, "Programming Restrictions" on page 3-56.

In this chapter, the operation, register bit fields, and signal names are described in terms of the I^2C controller only. Except where noted, all references to the operation, register bit fields, and signal names of the I^2C controller also apply to the power-I^2C controller. When considering the operation of the power-I^2C controller add the prefix "PWR_" to all signal names and "P" to all register names. For example, the I^2C signal "SDA" becomes "PWR_SDA" for the power I^2C controller and the register "ICR" in the I^2C controller becomes "PICR" in the power-I^2C register. (see Table 9-14).

9.2 Features

The I²C unit includes the following features:

- I²C compliant (see the *I²C-Bus Specification*, Version 2.0)
- Multi-master and arbitration support
- Standard-speed operation at 100 kbps. Power-I²C standard-speed operation is 40 kbps.
- Fast-mode operation at 400 kbps. Power-I²C fast-mode operation is 160 kbps.

9.3 Signal Descriptions

Table 9-1 describes the I²C bus signals, SDA, and SCL.

Table 9-1. I²C Bus Interface Unit I/O Signal Descriptions

Signal Name	Input/Output	Description
SDA	Bidirectional	I²C Serial Data/Address signal
SCL	Bidirectional	I²C Serial Clock Line signal
PWR_SDA	Bidirectional	Power I²C Serial Data/Address signal
PWR_SCL	Bidirectional	Power I²C Serial Clock Line signal

9.4 Operation

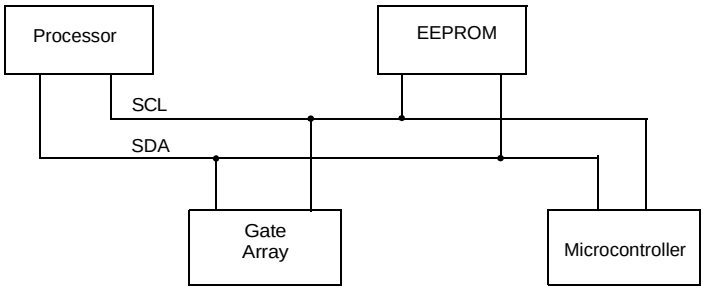
The *I²C-Bus Specification* defines a serial protocol for passing information between agents on the bus, using the two-pin interface shown in Table 9-1: a serial data and address (SDA) line, and a serial clock line (SCL). Each device on the I²C bus is recognized by a unique seven-bit address and can operate as a transmitter or as a receiver in master or slave mode. Table 9-2 defines the I²C-bus terminology.

Table 9-2. I²C Bus Definitions

I²C Device	Definition
Transmitter	Sends data over the I²C bus
Receiver	Receives data over the I²C bus
Master	Initiates transfers, generates clock signals, and terminates transactions
Slave	Device addressed by a master; it responds by transmitting or receiving data over the I²C bus
Multi-master	More than one master can attempt to control the bus at the same time without corrupting the message
Arbitration	Ensures that only one master controls the bus when more than one master simultaneously tries to control the bus. This technique avoids message corruption.

For example, the processor I²C interface can act as a master on the bus to address an EEPROM as the slave to receive data (see Figure 9-1). When the I²C interface addresses the EEPROM, it serves as a master transmitter and the EEPROM as a slave receiver. When the I²C interface reads data, it serves as a master receiver and the EEPROM as a slave transmitter. Whether as a transmitter or receiver, the master generates the clock, initiates the transaction, and terminates the transaction.

Figure 9-1. I²C Bus Configuration Example



The I²C bus uses an open-drain wired-AND structure, which allows multiple devices to drive the bus lines and to communicate status about events such as arbitration, wait states, and error conditions. When a master drives the clock (SCL) line during a data transfer, it transfers a bit on every instance that the clock is high. When the slave is unable to accept or drive data at the rate requested by the master, the slave can hold SCL low between the high states to insert wait intervals. The master clock can be altered only by another master during arbitration or by a slow slave peripheral that keeps the clock line low.

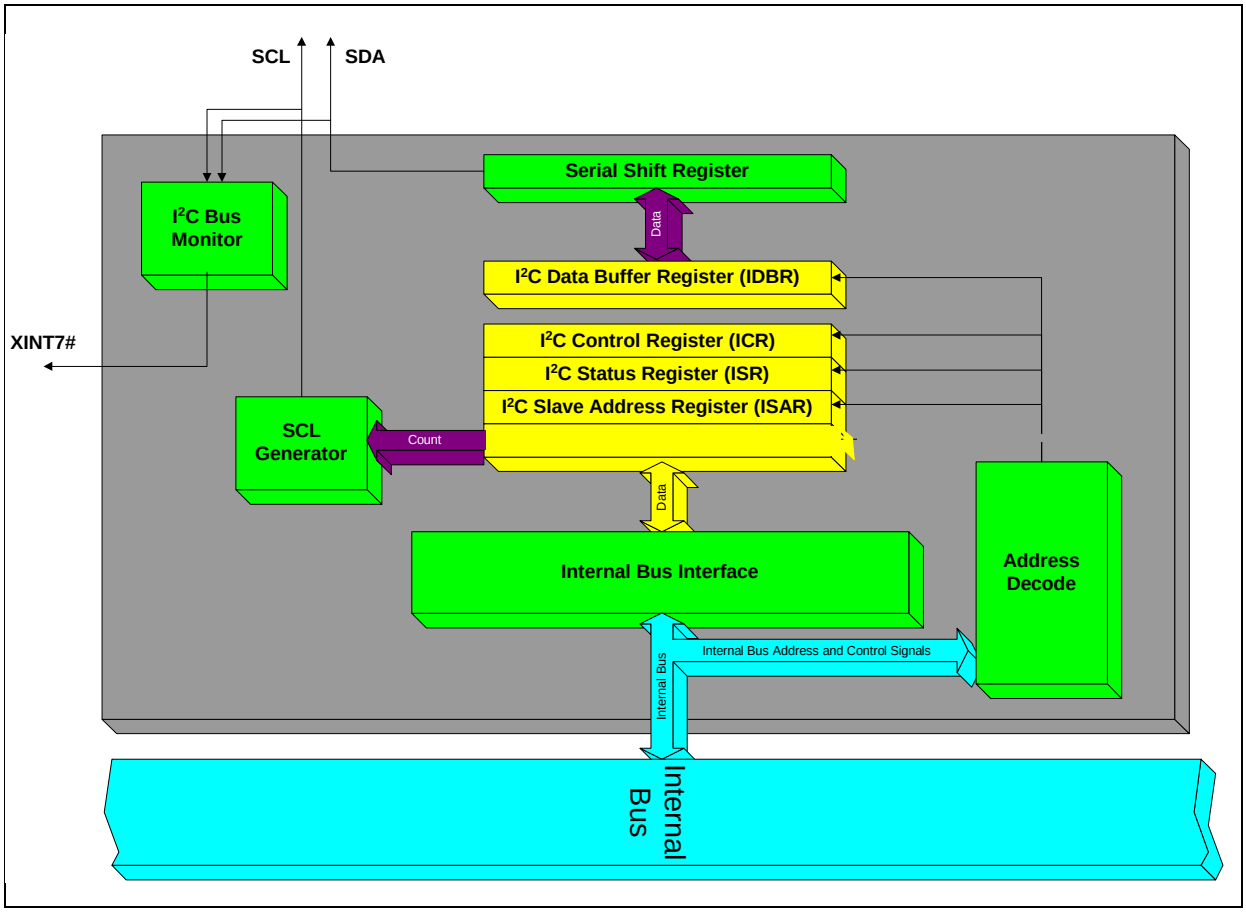
The I²C bus allows multiple masters, which means that more than one device can initiate data transfers at the same time. Bus arbitration resolves conflicts between masters. Two masters can drive the bus simultaneously, provided they drive identical data. A master loses the arbitration if it tries to drive SDA high while another master is driving SDA low. The SCL line is a synchronized combination of clocks generated by the masters using the wired-AND connection to the SCL line.

I²C transactions are initiated by either the I²C interface as a master or received by the I²C interface as a slave. Both conditions can result in reads, writes, or both over the I²C bus.

9.4.1 Operational Blocks

The I²C unit resides on the processor peripheral bus. The processor interrupt mechanism can be used to notify the CPU that there is activity on the I²C bus. Polling can be used instead of interrupts. The I²C interface consists of the two-wire interface to the I²C bus, an eight-bit buffer for passing data to and from the PXA27x processor, a set of Control and Status registers, and a Shift register for parallel/serial conversions (see Figure 9-2).

Figure 9-2. I²C Bus Interface Unit Block Diagram



The I²C interface initiates an interrupt to the PXA27x processor when:

- A buffer is full,
- A buffer is empty
- The I²C interface slave address is detected
- Arbitration is lost
- A bus error condition occurs

All interrupt conditions must be cleared explicitly by software. See Section 9.5.2 for details.

The I²C control, status, and data registers are located in the I²C memory-mapped address space. Section 9.5 defines the registers and their functions. The eight-bit I²C Data Buffer register (IDBR) transmits and receives data to and from the I²C bus using an internal Shift register that is not user-accessible. The I²C interface supports fast-mode operation at 400 kbps and standard-speed operation at 100 kbps. See the *I²C-Bus Specification* for more information.

Note: The Power I²C interface operates at non-standard speeds. Fast-mode operation operates at 160 kbps and standard-speed operation at 40 kbps.

9.4.2 I²C Bus Interface Modes

The I²C unit can accomplish a transfer in different operational modes. Table 9-3 summarizes the different modes.

Table 9-3. I²C Modes of Operation

Mode	Description
Master-transmit	• I²C interface acts as a master • Used for transmit operations • I²C interface sends the data • I²C interface generates the clock • Slave device is in slave-receive mode
Master-receive	• I²C interface acts as a master • Used for receive operations • I²C interface receives the data • I²C interface generates the clock • Slave device is in slave-transmit mode
Slave-transmit	• I²C interface acts as a slave • Responds to a master read operation • I²C interface sends the data • Master device is in master-receive mode
Slave-receive (default)	• I²C interface acts as a slave • Responds to a master write operation • I²C interface receives the data • Master device is in master-transmit mode

While the I²C interface is idle, it defaults to slave-receive mode, which allows the interface to monitor the bus and receive any slave addresses intended for the processor I²C interface.

When the I²C interface receives an address that matches the seven-bit address in the I²C Slave Address register (ISAR) or the general call address (see Section 9.4.12), the interface either remains in slave-receive mode or switches to slave-transmit mode. The read/write bit (R/nW) determines which mode the interface enters. The R/nW bit is the least significant bit of the byte containing the slave address. If R/nW is clear, the master that initiated the transaction intends to write data, and the I²C interface remains in slave-receive mode. If the R/nW bit is set, the master that initiated the transaction intends to read data, and the I²C interface switches to slave-transmit mode. Section 9.4.10 further defines slave operation.

When the PXA27x processor initiates a read or write on the I²C bus, it switches the interface from the default slave-receive mode to the master-transmit mode. If the transaction is a write, the I²C interface remains in master-transmit mode after the address transfer is completed. If the transaction is a read, the I²C interface transmits the slave address, then switches to master-receive mode. Section 9.4.8 further defines master operation.

9.4.3 START and STOP Bus States

The *I²C-Bus Specification* defines a START transaction, used at the beginning of a transfer, and a STOP transaction, used at the end of a transfer. A START condition occurs if a high-to-low transition takes place on the SDA line when SCL is high. A STOP condition occurs if a low-to-high transition takes place on the SDA line when SCL is high.

The I²C unit uses the ICR[START] and ICR[STOP] bits to:

- Initiate an additional byte transfer
- Initiate a START condition on the I²C bus
- Enable data chaining (repeated START)
- Initiate a STOP condition on the I²C bus

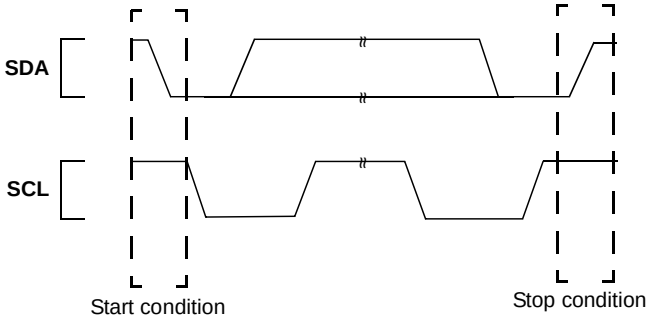
Table 9-4 defines the START and STOP bits in the ICR.

Table 9-4. START and STOP Bit Definitions

STOP bit	START bit	Condition	Notes
0	0	No START or STOP	The I²C interface sends a no START or STOP condition when multiple data bytes are to be transferred.
0	1	START condition and repeated START	The I²C interface sends a START condition and transmits the IDBR's eight-bit contents. The IDBR must contain the seven-bit slave address and the R/nW bit before a START is initiated. For a repeated start, the IDBR contains the target slave address and the R/nW bit. This allows a master to make multiple transfers to different slaves without giving up the bus. The interface stays in master-transmit mode for writes and switches to master-receive mode for reads.
1	X	STOP condition	In master-transmit mode, the I²C interface transmits the IDBR's eight-bit contents and sends a STOP condition on the I²C bus. In master-receive mode, ICR[ACKNAK] must be set, which defines a negative-ACKNOWLEDGE (NAK) pulse (see Section 9.4.6). The I²C interface transmits the NAK pulse, places the received data byte into the IDBR, and sends a STOP condition on the I²C bus.

Figure 9-3 shows the relationship between the SDA and SCL lines for START and STOP.

Figure 9-3. SDA and SCL Signals During START and STOP Conditions



9.4.3.1 START Condition

The START condition (ICR[START] = 1, ICR[STOP] = 0) initiates a master transaction or repeated START. Before it sets ICR[START], software must load the target slave address and the R/nW bit in the IDBR (see Section 9.5.4). The START and the IDBR contents are transmitted on the I²C bus after ICR[TB] is set. The I²C bus stays in master-transmit mode for write requests and enters master-receive mode for read requests. For a repeated START, a change in read or write, or a change in the target slave address, the IDBR contains the updated target slave address and the R/nW bit. A repeated START enables a master to make multiple transfers to different slaves without surrendering the bus.

The START condition is not cleared by the I²C interface. If the I²C interface loses arbitration while initiating a START, it may retry the START when the bus is freed. See Section 9.4.7 for details on how the I²C interface functions in those circumstances.

9.4.3.2 No START or STOP Condition

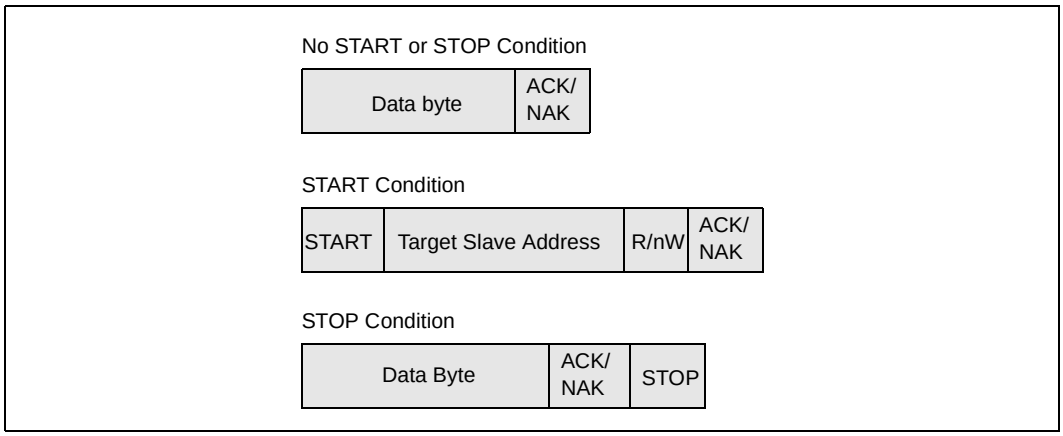
The no START or STOP condition (ICR[START] = 0, ICR[STOP] = 0) is used in master-transmit mode while the I²C interface is transmitting multiple data bytes (see Figure 9-4). Software writes the data byte, and the I²C interface sets ISR[ITE] and clears ICR[TB]. Software then writes a new byte to the IDBR and sets ICR[TB], which initiates the new byte transmission. This process continues until software sets ICR[START] or ICR[STOP]. ICR[START] and ICR[STOP] are not cleared automatically by the I²C interface after the transmission of a START, STOP, or repeated START.

After each byte transfer, including the ACKNOWLEDGE pulse defined by the ICR[ACKNAK] control bit, the I²C interface holds the SCL line low to insert wait states until ICR[TB] is set. This action notifies the I²C interface to release the SCL line and allow the next information transfer to proceed.

9.4.3.3 STOP Condition

The STOP condition (ICR[START] = X, ICR[STOP] = 1) terminates a data transfer. In master-transmit mode, ICR[STOP] and ICR[TB] must be set to initiate the last byte transfer (see Figure 9-4). In master-receive mode, the I²C interface must set ICR[ACKNAK], ICR[STOP], and ICR[TB] to initiate the last transfer. Software must clear ICR[STOP] after the STOP condition is transmitted.

Figure 9-4. START and STOP Conditions



9.4.4 Data Transfer Sequence

The I²C unit transfers data in 1-byte increments and always follows this sequence:

1. START
2. Seven-bit slave address
3. R/nW bit
4. ACKNOWLEDGE pulse
5. Eight bits of data
6. ACKNOWLEDGE pulse
7. Repeat of steps 5 and 6 for the required number of bytes
8. Repeated START (repeat step 1) or STOP

9.4.5 Data and Addressing Management

The I²C Data Buffer register (IDBR) and the I²C Slave Address register (ISAR) manage data and slave addressing. The IDBR (see Section 9.5.4) contains one byte of data or a seven-bit slave address and the R/nW bit. The ISAR contains the processor's programmable slave address. The I²C interface puts received data into the IDBR after a full byte is received and acknowledged. To transmit data, the CPU writes to the IDBR, and the I²C interface passes the information to the serial bus when ICR[TB] is set. See Section 9.5.1.

When the I²C interface is in master- or slave-transmit mode:

1. Software writes data to the IDBR over the internal bus, which initiates a master transaction or sends the next data byte after ISR[ITE] is set.
2. I²C interface transmits data from the IDBR when ICR[TB] is set.
3. When enabled, an IDBR transmit-empty interrupt is signaled when a byte is transferred on the I²C bus and the acknowledge cycle is complete.

4. When the I²C interface is ready to transfer the next byte before the CPU has written the IDBR and a STOP condition is not in place, the I²C interface inserts wait states until the CPU writes a new value to the IDBR and sets ICR[TB].

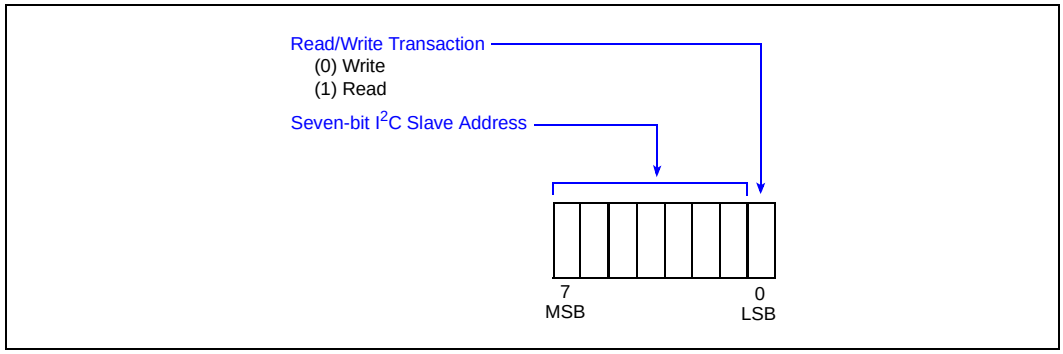
When the I²C interface is in master- or slave-receive mode:

1. The processor reads IDBR data over the internal bus after the IDBR receive-full interrupt is signaled.
2. I²C interface transfers data from the Shift register to the IDBR after the acknowledge cycle completes.
3. I²C interface inserts wait states until the IDBR is read. See Section 9.4.6 for information about the ACKNOWLEDGE pulse in receive mode.
4. After the CPU reads the IDBR, the I²C interface unit updates the ICR[ACKNAK] and ICR[TB] bits, allowing the next byte transfer to proceed.

9.4.5.1 Addressing a Slave Device

As a master device, the I²C interface must compose and send the first byte of a transaction. This byte consists of the slave address for the intended device and a R/nW bit for transaction definition. To address a slave device, write the slave address and the R/nW bit to the IDBR (see Figure 9-5).

Figure 9-5. Data Format of First Byte in Master Transaction



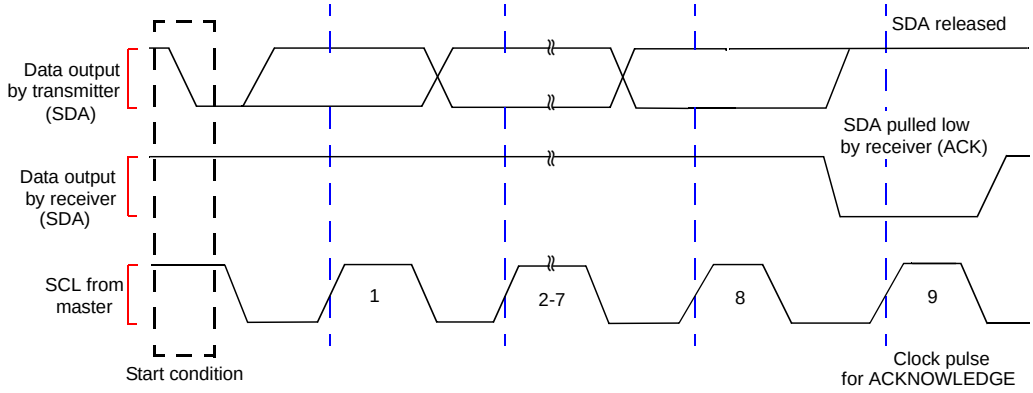
The first byte transmission must be followed by a positive-ACKNOWLEDGE (ACK) pulse from the addressed slave. When the transaction is a write, the I²C interface remains in master-transmit mode and the addressed slave device stays in slave-receive mode. When the transaction is a read, the I²C interface switches to master-receive mode immediately following the ACK, and the addressed slave device switches to slave-transmit mode. When a negative-ACKNOWLEDGE (NAK) is returned, the I²C interface aborts the transaction by automatically sending a STOP and setting ISR[BED].

When the I²C interface is enabled and idle, it remains in slave-receive mode and monitors the I²C bus for a START signal. When it detects a START condition, the I²C interface reads the first seven bits and compares them to those in the ISAR and the general call address (0x00). When the bits match those in the ISAR register, the I²C interface reads the eighth bit (R/nW bit) and transmits an ACK pulse. The I²C interface either remains in slave-receive mode (R/nW = 0) or switches to slave-transmit mode (R/nW = 1). See Section 9.4.12 for actions when a general call address is detected.

9.4.6 I²C ACKNOWLEDGE

Every I²C byte transfer must be accompanied by an ACKNOWLEDGE pulse that the master or slave receiver must generate. The transmitter must release the SDA line for the receiver to transmit the acknowledge pulse (see Figure 9-6).

Figure 9-6. ACKNOWLEDGE Pulse on I²C Bus



In master-transmit mode, if the target slave receiver cannot generate the positive-ACKNOWLEDGE (ACK) pulse, the SDA line remains high. The lack of an ACK causes the I²C interface to set ISR[BED] and generate the associated interrupt when enabled. The I²C interface automatically generates a STOP condition and aborts the transaction.

In master-receive mode, the I²C interface sends a negative-acknowledge (NAK) pulse to signal the slave transmitter to stop sending data. The ICR[ACKNAK] bit controls the ACK/NAK pulse value driven onto the I²C bus. As required by the I²C bus protocol, ISR[BED] is not set for a master-receive mode NAK. The I²C interface automatically transmits the ACK pulse after it receives each byte from the serial bus. Before the unit receives the last byte, software must set ICR[ACKNAK] to generate a NAK. The NAK pulse, sent after the last byte, signals that the last byte has been sent.

In slave mode, the I²C interface automatically acknowledges its own slave address, regardless of the ICR[ACKNAK] setting. In slave-receive mode, an ACK response automatically follows a data byte, regardless of the ICR[ACKNAK] setting. The I²C interface sends the ACKNOWLEDGE value defined by ICR[ACKNAK] after it receives the eighth data bit in a byte. In slave-transmit mode, the I²C interface receives a NAK from the master to indicate that the last byte has been transferred. The master then sends a STOP or repeated START; the processor's unit-busy indication, ISR[UB], remains set until a STOP or repeated START is received.

9.4.7 Arbitration

The I²C bus multi-master capabilities require I²C bus arbitration. Arbitration occurs when two or more masters generate a START condition in the minimum hold time.

Arbitration can take a long time. If the address bit and the R/nW are the same, the arbitration scheme considers the data. Because the I²C bus has a wired-AND nature, a transfer does not lose data if multiple masters signal the same bus states. If the address and the R/nW bit or the data they

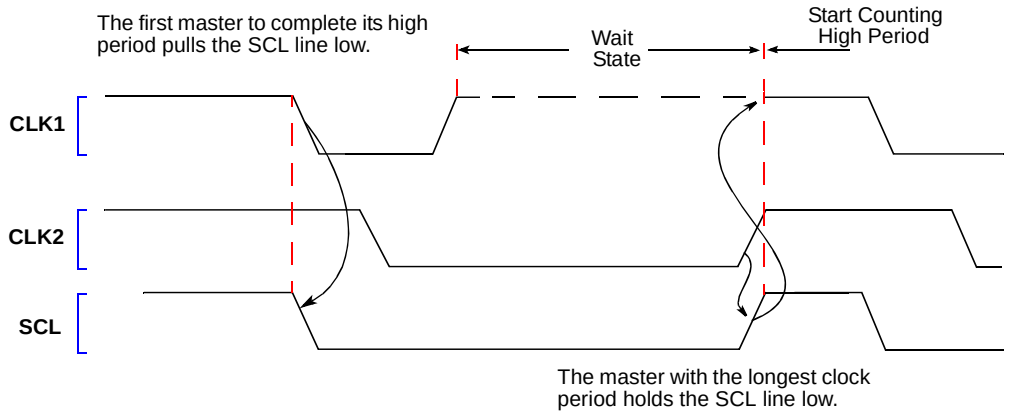
contain are different, the master that signals a high state loses arbitration and shuts off its data drivers. If the I²C unit loses arbitration, it shuts off the SDA or SCL drivers for the rest of the byte transfer, sets ISR[ALD], and returns to slave-receive mode.

9.4.7.1 SCL Arbitration

Each master on the I²C bus generates its own clock on the SCL line for data transfers. As a result, clocks with different frequencies can be connected to the SCL line. Because data is valid when a clock is in the high period, bit-by-bit arbitration requires a defined clock-synchronization procedure.

Clock synchronization is through the wired-AND connection of the I²C interfaces to the SCL line. When a master's clock changes from high to low, the master holds down the SCL line for its associated period (see Figure 9-7). A clock cannot switch from low to high if another master has not completed its period. The master with the longest low period holds down the SCL line. Masters with shorter periods are held in a high wait-state until the master with the longest period completes. After the master with the longest period completes, the SCL line changes to the high state and masters with the shorter periods continue the data cycle.

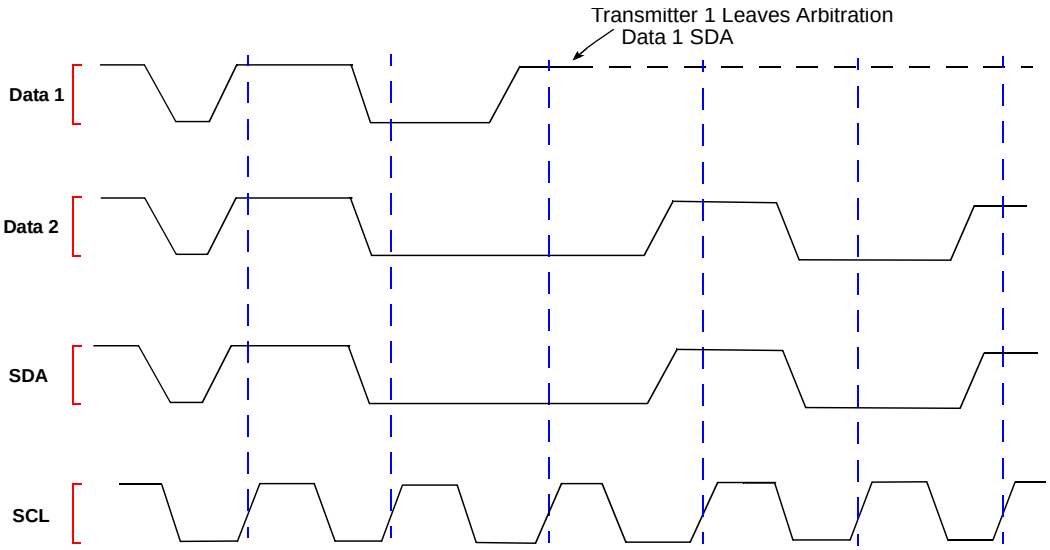
Figure 9-7. Clock Synchronization During Arbitration



9.4.7.2 SDA Arbitration

Arbitration on the SDA line can continue for a long time because it starts with the address and R/nW bits and continues through the data bits. Figure 9-8 shows the arbitration procedure for two masters. More than two masters may be involved if more than two masters are connected to the bus. If the address bit and the R/nW are the same, the arbitration scheme considers the data. Because the I²C bus has a wired-AND nature, a transfer does not lose data if multiple masters signal the same bus states. If the address and the R/nW bit or the data they contain are different, the master that sent the first high data bit loses arbitration and shuts off its data drivers. If the I²C interface loses arbitration, it shuts off the SDA or SCL drivers for the rest of the byte transfer, sets ISR[ALD], and returns to slave-receive mode.

Figure 9-8. Arbitration Procedure for Two Masters



If the I²C interface loses arbitration as the address bits are transferred and it is not addressed by the address bits, the I²C interface re-sends the address when the I²C bus becomes free. A re-send is possible because registers IDBR and ICR are not overwritten when arbitration is lost.

If the I²C interface loses arbitration because another bus master addresses the processor I²C as a slave device, the I²C interface switches to slave-receive mode and overwrites the original data in register IDBR. Software can clear the START and re-initiate the master transaction.

Note: Software must prevent the I²C interface from starting a transaction to its own slave address. Such transactions put the I²C interface into an indeterminate state.

Arbitration has boundary conditions in case an arbitration process is interrupted by a repeated START or STOP condition transmitted on the I²C bus. To prevent errors, the I²C interface acts as a master if no arbitration occurs in the following circumstances:

- Between a repeated START condition and a data bit
- Between a data bit and a STOP condition
- Between a repeated START condition and a STOP condition

These situations occur if different masters write identical data to the same target slave simultaneously and arbitration cannot be resolved after the first data byte transfer.

Note: Software must ensure that arbitration is resolved quickly. For example, software can ensure that masters send unique data by requiring that each master transmit its I²C address as the first data byte of any transaction. When arbitration is resolved, the winning master sends a restart and begins a valid data transfer. The slave discards the master's address and uses the other data.

9.4.8 Master Operations

When software initiates a read or write on the I²C bus, the I²C unit switches from the default slave-receive mode to master-transmit mode. The seven-bit slave address and the R/nW bit follow the START pulse. After the master receives an ACKNOWLEDGE, the I²C interface enters one of two master modes:

- Master-transmit—I²C interface writes data
- Master-receive—I²C interface reads data

The CPU writes to the ICR register to initiate a master transaction. Data is read and written from the I²C unit through the memory-mapped registers. Table 9-5 describes the I²C unit responsibilities as a master device.

Table 9-5. Master Transactions (Sheet 1 of 2)

I²C Master Action	Mode of Operation	Definition
Generate clock output	Master-transmit Master-receive	• The master drives the SCL line. • ICR[SCLE] and ICR[IUE] must be set.
Write target slave address to IDBR	Master-transmit Master-receive	• The CPU writes to IDBR bits [7:1] before enabling a START condition. • The first seven bits are sent on the I²C bus after START. See Section 9.4.3.
Write R/nW bit to IDBR	Master-transmit Master-receive	• CPU writes to least significant IDBR bit with target slave address. • If low, master remains a master-transmitter. If high, master switches to a master receiver. See Section 9.4.5.
Signal START condition	Master-transmit Master-receive	See “Generate clock output” above. After the target slave address and R/nW bit are in the IDBR, • Software sets ICR[START]. • Software sets ICR[TB] to initiate the START condition. See Section 9.4.3.
Initiate first data byte transfer	Master-transmit Master-receive	• The CPU writes a data byte to the IDBR • The I²C interface transmits the byte when ICR[TB] is set. • The I²C interface clears ICR[TB] and sets ISR[ITE] when the transfer is complete.
Arbitrate for I²C bus	Master-transmit Master-receive	If two or more masters signal a START within the same clock period, arbitration must occur. • The I²C interface arbitrates for as long as needed. Arbitration takes place during slave address and R/nW bit or data transmission and continues until all but one master loses the bus. No data is lost. • If the I²C interface loses arbitration, it sets ISR[ALD] after the byte transfer is completed and switches to slave-receive mode. • If the I²C interface loses arbitration as it attempts to send the target address byte, it attempts to resend the byte when the bus becomes free. Software must ensure that the boundary conditions described in Section 9.4 do not occur.

Table 9-5. Master Transactions (Sheet 2 of 2)

I²C Master Action	Mode of Operation	Definition
Write one data byte to the IDBR	Master-transmit only	• Occurs when ISR[ITE] is set and ICR[TB] is clear. If the IDBR transmit-empty interrupt is enabled, the interrupt is generated. • The CPU writes one data byte to the IDBR, sets the appropriate START/STOP bit combination, and sets ICR[TB] to send the data. Eight bits are taken from the shift register and written to the serial bus. The eight bits are followed by a STOP, if requested.
Wait for ACKNOWLEDGE from slave receiver	Master-transmit only	As a master transmitter, the I²C interface generates the clock for the ACKNOWLEDGE pulse. The I²C interface releases the SDA line to allow slave-receiver ACKNOWLEDGE transmission. See Section 9.4.6.
Read one byte of I²C data from the IDBR	Master-receive only	Eight bits are read from the serial bus, collected in the shift register, then transferred to the IDBR after the ICR[ACKNAK bit is read. • The CPU reads the IDBR when ISR[IRF] is set and ICR[TB] is clear. If the IDBR receive-full interrupt is enabled, it is signaled to the CPU. • When the IDBR is read, if ISR[ACKNAK] is clear (indicating ACK), the processor I²C interface writes the ICR[ACKNAK] bit and sets ICR[TB] to initiate the next byte read. • If ISR[ACKNAK] is set (indicating NAK), ICR[TB] is clear, ICR[STOP] is set, and ISR[UB] is set, then the last data byte has been read into the IDBR, and the I²C interface is sending the STOP. • If ISR[ACKNAK] is set (indicating NAK) and ICR[TB] is clear, but ICR[STOP] is clear, then the CPU has two options: 1. Set ICR[START], write a new target address to the IDBR, and set ICR[TB], which sends a repeated START. 2. Set ICR[MA] and leave ICR[TB] clear, which sends a STOP only.
Transmit ACKNOWLEDGE to slave transmitter	Master-receive only	• As a master receiver, the I²C interface generates the clock for the ACKNOWLEDGE pulse and drives the SDA line during the acknowledge cycle. • If the next data byte is to be the last transaction, the CPU sets ICR[ACKNAK] for NAK generation. See Section 9.4.6.
Generate a repeated START to chain I²C transactions	Master-transmit Master-receive	Data chaining takes place by using a repeated START condition instead of a STOP condition. • The repeated START is generated after the last data byte of a transaction has been transmitted on the I²C bus, as described in Section 9.4.4. • The CPU writes the next target slave address and the R/nW bit to the IDBR, sets ICR[START], and sets ICR[TB]. See Section 9.4.3.
Generate a STOP	Master-transmit Master-receive	• A STOP is generated after the last data byte of a transaction has been transmitted on the I²C bus, as described in Section 9.4.4. • ICR[STOP] must be set to generate the STOP condition. See Section 9.4.3.

When the CPU needs to read data, the I²C interface switches from slave-receive mode to master-transmit mode to transmit the slave address, R/nW bit, and the ACK pulse. After it sends the ACK pulse, the I²C interface switches to master-receive mode and waits to receive the read data from the slave device (see Figure 9-9). Multiple transactions can occur during an I²C operation. For example, switching from master-receive to master-transmit through a repeated start or data chaining (see Figure 9-10). Figure 9-11 shows the SDA and SCL wave forms for a complete data transfer.

Figure 9-9. Master Receiver Read from Slave Transmitter

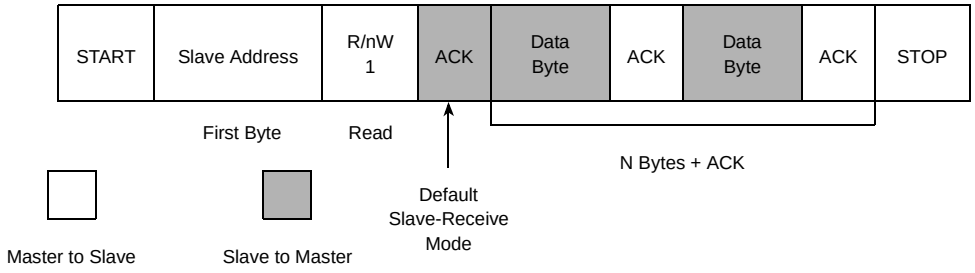


Figure 9-10. Master Receiver Read from Slave Transmitter, Repeated Start, Master Transmitter Write to Slave-Receiver

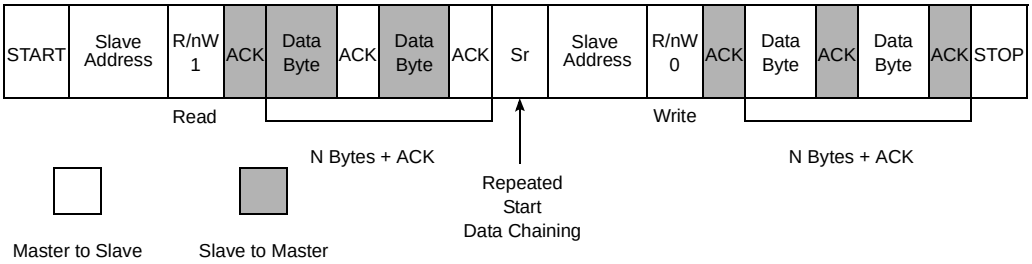
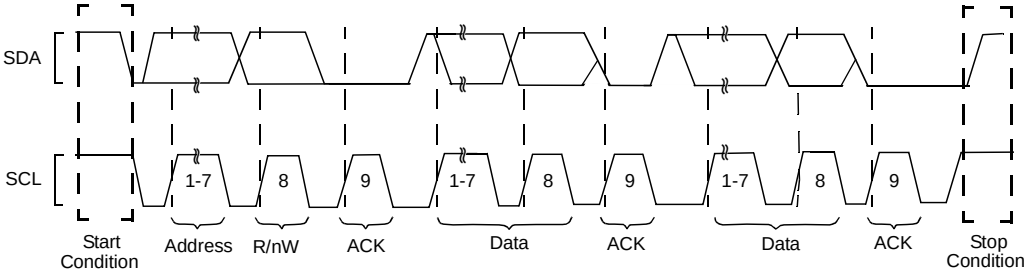


Figure 9-11. A Complete Data Transfer



9.4.9 Master Mode Programming Examples

9.4.9.1 Initialize Unit

1. Set the slave address in the ISAR.
2. Enable the preferred interrupts in the ICR. Do not enable the arbitration-loss-detected interrupt.
3. Set the ICR[IUE] and ICR[SCLE] bits to enable the I^2C interface and SCL.

9.4.9.2 Write 1 Byte as a Master

1. Load target slave address and R/nW bit in the IDBR. R/nW must be 0 for a write.
2. Initiate the write.
   Set ICR[START], clear ICR[STOP], clear ICR[ALDIE], set ICR[TB].
3. When an IDBR transmit-empty interrupt occurs:
   Read ISR: IDBR transmit empty (1), Unit Busy (1), R/nW bit (0).
4. Write 0b1 to the ISR[ITE] bit to clear interrupt.
5. Write 0b1to the ISR[ALD] bit if set.
   If the master loses arbitration, it performs an address retry when the bus becomes free. The arbitration-loss-detected interrupt is disabled to allow the address retry.
6. Load data byte to be transferred in the IDBR.
7. Initiate the write.
   Clear ICR[START], set ICR[STOP], set ICR[ALDIE], set ICR[TB].
8. When an IDBR transmit-empty interrupt occurs (unit is sending STOP):
   Read ISR: IDBR transmit empty (1), Unit busy (x), R/nW bit (0).
9. Write 0b1 to the ISR[ITE] bit to clear the interrupt.
10. Clear ICR[STOP] bit.

9.4.9.3 Read 1 Byte as a Master

1. Load target slave address and R/nW bit in the IDBR. R/nW must be 1 for a read.
2. Initiate the write.
   Set ICR[START], clear ICR[STOP], clear ICR[ALDIE], set ICR[TB].
3. When an IDBR transmit-empty interrupt occurs.
   Read ISR: IDBR transmit empty (1), Unit busy (1), R/nW bit (1).
4. Write 0b1 to the ISR[ITE] bit to clear the interrupt.
5. Initiate the read.
   Clear ICR[START], set ICR[STOP], set ICR[ALDIE], set ICR[ACKNAK], set ICR[TB].
6. When an IDBR receive-full interrupt occurs (unit is sending STOP):
   Read ISR: IDBR receive full (1), Unit Busy (x), R/nW bit (1), ACK/NAK bit (1).
7. Write 0b1 to the ISR[IRF] bit to clear the interrupt.
8. Read IDBR data.
9. Clear ICR[STOP] and ICR[ACKNAK] bits.

9.4.9.4 Write 2 Bytes and Repeated Start Read 1 Byte as a Master

1. Load target slave address and R/nW bit in the IDBR. R/nW must be 0 for a write.
2. Initiate the write.
   Set ICR[START], clear ICR[STOP], clear ICR[ALDIE], set ICR[TB].
3. When an IDBR transmit-empty interrupt occurs.
   Read ISR: IDBR transmit empty (1), Unit Busy (1), R/nW bit (0).
4. Write 0b1 to the ISR[ITE] bit to clear interrupt.

5. Load data byte to be transferred in the IDBR.
6. Initiate the write.
   Clear ICR[START], clear ICR[STOP], set ICR[ALDIE], set ICR[TB].
7. When an IDBR transmit-empty interrupt occurs:
   Read ISR: IDBR transmit empty (1), Unit busy (1), R/nW bit (0).
8. Write 0b1 to the ISR[ITE] bit to clear interrupt.
9. Repeat steps 5-8 one time.
10. Load target slave address and R/nW bit in the IDBR. R/nW must be 1 for a read.
11. Send repeated START as a master.
    Set ICR[START], clear ICR[STOP], clear ICR[ALDIE], set ICR[TB].
12. When an IDBR transmit-empty interrupt occurs.
    Read ISR: IDBR transmit empty (1), Unit busy (1), R/nW bit (1).
13. Write 0b1 to the ISR[ITE] bit to clear interrupt.
14. Initiate the read.
    Clear ICR[START], set ICR[STOP], set ICR[ALDIE], set ICR[ACKNAK], set ICR[TB].
15. When an IDBR receive-full interrupt occurs (unit is sending STOP).
    Read ISR: IDBR receive full (1), Unit Busy (x), R/nW bit (1), ACK/NAK bit (1).
16. Write 0b1 to the ISR[IRF] bit to clear the interrupt.
17. Read IDBR data.
18. Clear ICR[STOP] and ICR[ACKNAK] bits.

9.4.9.5 Read 2 Bytes as a Master—Send STOP Using the Abort

1. Load target slave address and R/nW bit in the IDBR. R/nW must be 1 for a read.
2. Initiate the write.
   Set ICR[START], clear ICR[STOP], clear ICR[ALDIE], set ICR[TB].
3. When an IDBR transmit-empty interrupt occurs:
   Read ISR: IDBR transmit empty (1), Unit busy (1), R/nW bit (1).
4. Write 0b1 to the ISR[ITE] bit to clear interrupt.
5. Initiate the read.
   Clear ICR[START], clear ICR[STOP], set ICR[ALDIE], clear ICR[ACKNAK], set ICR[TB].
6. When an IDBR receive-full interrupt occurs:
   Read ISR: IDBR receive full (1), Unit Busy (1), R/nW bit (1), ACK/NAK bit (0).
7. Write 0b1 to the ISR[IRF] bit to clear the interrupt.
8. Read IDBR data.
9. Clear ICR[STOP] and ICR[ACKNAK] bits.
10. Initiate the read.
    Clear ICR[START], clear ICR[STOP], set ICR[ALDIE], set ICR[ACKNAK], set ICR[TB]
    ICR[STOP] is not set because STOP or repeated START is decided on the byte read.
11. When an IDBR receive-full interrupt occurs.
    Read ISR: IDBR receive full (1), Unit Busy (1), R/nW bit (1), ACK/NAK bit (1).

12. Write 0b1 to the ISR[IRF] bit to clear the interrupt.
13. Read IDBR data.
14. Initiate STOP abort condition (STOP with no data transfer). Set ICR[MA].

Note: If a NAK is not sent in Step 11, the next transaction must involve another data byte read.

9.4.10 Slave Operations

Table 9-6 describes how the I²C unit operates as a slave device.

Table 9-6. Slave Transactions

I²C Slave Action	Mode of Operation	Definition
Slave-receive (default mode)	Slave-receive only	• The I²C interface monitors all slave address transactions. • ICR[IUE] must be set. • The I²C interface monitors bus for START conditions. When a START is detected, the interface reads the first 8 bits and compares the most significant seven bits with the seven-bit ISAR and the general call address (0x00). If there is a match, the I²C interface sends an ACK. • If the first 8 bits are zeros, this is a general call address. If ICR[GCD] is clear, both the ISR[GCAD] and ISR[SAD] are set. See Section 9.4.12. • If the eighth bit of the first byte (R/nW bit) is low, the I²C interface stays in slave-receive mode, and ISR[SAD] is cleared. If R/nW bit is high, the ²C unit switches to slave-transmit mode, and ISR[SAD] is set.
Set the slave-address-detected bit	Slave-receive Slave-transmit	• Indicates that the interface has detected an I²C operation that addresses the PXA27x processor including the general call address. The PXA27x processor can distinguish an ISAR match from a general call by reading ISR[GCAD]. • An interrupt is generated, if enabled, after the matching slave address is received and acknowledged.
Read one byte of I²C data from the IDBR	Slave-receive only	• This operation occurs when ISR[IRF] is set and ICR[TB] is clear. If enabled, the IDBR receive-full interrupt is generated. • Eight bits are read from the serial bus into the shift register. When a full byte is received and the ACK/NAK bit is completed, the byte is transferred from the shift register to the IDBR. • Software reads one data byte from the IDBR. When the IDBR is read, the PXA27x processor writes the desired ICR[ACKNAK] bit and sets ICR[TB]. This causes the I²C interface to stop inserting wait states and let the master transmitter transmit the next piece of information.
Transmit Acknowledge to master transmitter	Slave-receive only	• As a slave receiver, the I²C interface pulls the SDA line low to generate the ACK pulse during the high SCL period. • ICR[ACKNAK] controls the ACKNOWLEDGE pulse that the I²C interface drives. See Section 9.4.6.
Write one byte of I²C data to the IDBR	Slave-transmit only	• This operation occurs when ISR[ITE] is set and ICR[TB] is clear. If enabled, the IDBR transmit-empty interrupt is generated. • The PXA27x processor writes a data byte to IDBR and sets ICR[TB] to start the transfer.
Wait for Acknowledge from master receiver	Slave-transmit only	• As a slave transmitter, the I²C interface releases the SDA line to allow the master receiver to pull the line low for the ACK. See Section 9.4.6.

Figure 9-12 through Figure 9-14 are examples of I²C transactions and show the relationships between master and slave devices.

Figure 9-12. Master Transmitter Write to Slave Receiver

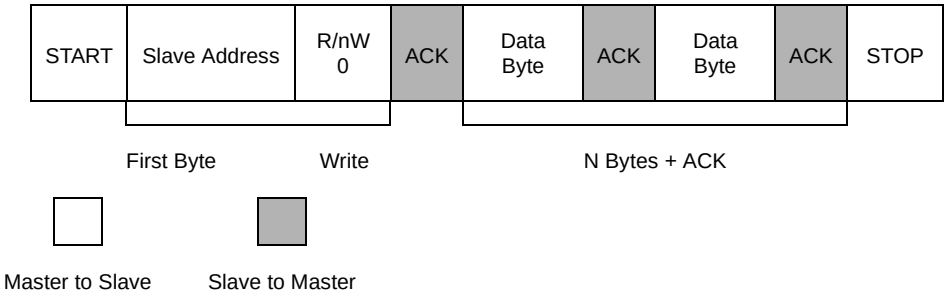


Figure 9-13. Master Receiver Read from Slave-Transmitter

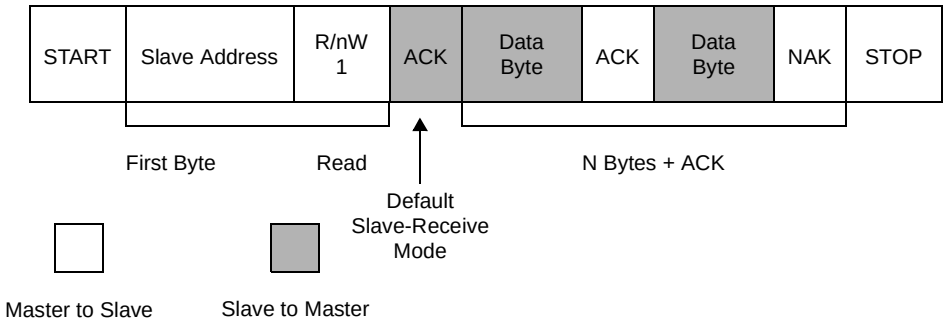
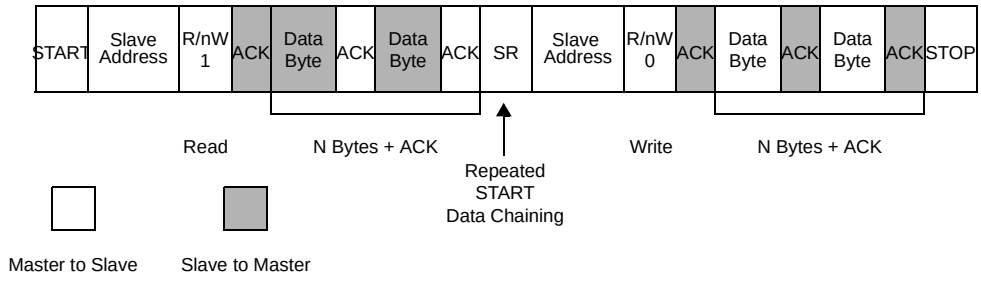


Figure 9-14. Master Receiver Read from Slave Transmitter, Repeated START, Master Transmitter Write to Slave Receiver



9.4.11 Slave Mode Programming Examples

9.4.11.1 Initialize Unit

1. Set the slave address in the ISAR.
2. Enable preferred interrupts in the ICR.

3. Set the ICR[IUE] bit to enable the I²C interface.

9.4.11.2 Write *n* Bytes as a Slave

1. When a slave-address-detected interrupt occurs:
   Read ISR: Slave Address Detected (1), Unit Busy (1), R/nW bit (1), ACK/NAK (0).
2. Write 0b1 to the ISR[SAD] bit to clear the interrupt.
3. Return from interrupt.
4. Load data byte to transfer in the IDBR.
5. Set ICR[TB] bit.
6. When a IDBR transmit-empty interrupt occurs:
   Read ISR: IDBR transmit empty (1), ACK/NAK (0), R/nW bit (0)
7. Load data byte to transfer in the IDBR.
8. Set the ICR[TB] bit.
9. Write 0b1 to the ISR[ITE] bit to clear interrupt.
10. Return from interrupt.
11. Repeat steps 6 to 10 for *n*-1 times. If at any time the slave does not have data, the I²C interface keeps SCL low until data is available.
12. When a IDBR transmit-empty interrupt occurs.
    Read ISR: IDBR transmit empty (1), ACK/NAK (1), R/nW bit (0).
13. Write 0b1 to the ISR[ITE] bit to clear interrupt.
14. Return from interrupt
15. When a slave-STOP-detected interrupt occurs.
    Read ISR: Unit Busy (0), Slave STOP Detected (1).
16. Write 0b1 to the ISR[SSD] bit to clear interrupt.

9.4.11.3 Read *n* Bytes as a Slave

1. When a slave-address-detected interrupt occurs:
   Read ISR: Slave Address Detected (1), Unit busy (1), R/nW bit (0).
2. Write 0b1 to the ISR[SAD] bit to clear the interrupt.
3. Return from interrupt.
4. Set ICR[TB] bit to initiate the transfer.
5. When an IDBR receive-full interrupt occurs:
   Read ISR: IDBR receive full (1), ACK/NAK (0), R/nW bit (0).
6. Read IDBR to get the received byte.
7. Write 0b1 to the ISR[IRF] bit to clear interrupt.
8. Return from interrupt.
9. Repeat steps 4 to 8 for *n*-1 times. Once the IDBR is full, the I²C interface keeps SCL low until the data is read.
10. Set ICR[TB] bit to release I²C bus and allow next transfer.

11. When a slave-STOP-detected interrupt occurs.
    Read ISR: Unit busy (0), Slave STOP Detected (1).
12. Write 0b1 to the ISR[SSD] bit to clear interrupt.

9.4.12 General Call Address

A general call address is a transaction with a slave address of 0x00. When a device requires the data from a general call address, it acknowledges the transaction and remains in slave-receive mode. Otherwise, the device ignores the general call address. The other bytes in a general call transaction are acknowledged by every device that uses it on the bus. Devices that do not use these bytes must not send an ACK. The meaning of a general call address is defined in the second byte sent by the master transmitter. Figure 9-15 shows a general call-address transaction. The least significant bit of the second byte, called B, defines the transaction. Table 9-7 shows the valid values and definitions when B = 0.

The I²C unit supports sending and receiving general call-address transfers on the I²C bus. When software sends a general call message from the I²C interface, it must set the ICR[GCD] bit to prevent the I²C interface from responding as a slave. If the ICR[GCD] is not set, the I²C bus enters an indeterminate state.

If the I²C interface acts as a slave and receives a general call address while the ICR[GCD] bit is clear, it:

- Sets the ISR[GCAD] bit
- Sets the ISR[SAD] bit
- Interrupts the processor (if the interrupt is enabled)

If the I²C interface receives a general call address and the ICR[GCD] bit is set, it ignores the general call address.

Figure 9-15. General Call Address

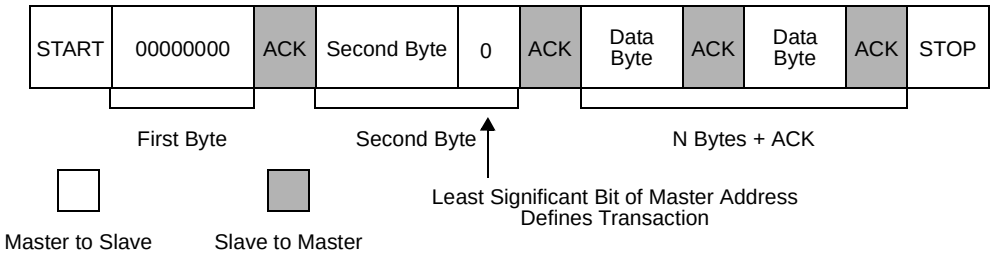


Table 9-7. General Call Address Second Byte Definitions

Least Significant Bit of Second Byte (B)	Second Byte Value	Definition
0	0x06	2-byte transaction in which the second byte tells the slave to reset and store this value in the programmable part of its address.
0	0x04	2-byte transaction in which the second byte tells the slave to store this value in the programmable part of its address. No reset.
0	0x00	Not allowed as a second byte
NOTE: Other values are not fixed and must be ignored.		

Software must ensure that (1) the I²C interface is not busy before it asserts a reset and (2) the I²C bus is idle when the unit is enabled after reset. When directed to reset, the I²C interface, except for ISAR, returns to the default reset condition. ISAR is not affected by a reset.

When B = 1, the sequence is a hardware general call and is not supported by the I²C interface. Refer to the *I²C-Bus Specification* for information on hardware general calls.

I²C 10-bit addresses and CBUS compatibility are not supported.

9.4.13 Reset Conditions

Software must ensure that (1) the I²C unit is not busy before it asserts a reset and (2) the I²C bus is idle when the unit is enabled after reset. When directed to reset, the I²C interface, except for ISAR, returns to the default reset condition. ISAR is not affected by a reset.

When the ICR[UR] bit is set, the I²C interface resets but the associated I²C MMRs remain intact. Use the following guidelines when resetting the I²C interface with the ICR unit reset:

1. Set the reset bit in the ICR register and clear the remainder of the register.
2. Clear the ISR register.
3. Clear reset in the ICR.

9.5 Register Descriptions

For any access to any power-manager I²C registers, the CKEN bit for the power-manager I²C unit must be enabled (CKEN[15] set).

9.5.1 I²C Control Registers (ICR, PICR)

The PXA27x processor uses the bits in the I²C Control register (ICR) to control the I²C unit.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 9-8. ICR, PICR Bit Definitions (Sheet 1 of 4)

	Physical Address 0x4030_1690 0x40F0_0190																ICR PICR												Standard I²C Power I²C															
User Settings																																											
Bit	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																FM	UR	SADIE	ALDIE	SSDIE	BEIE	DRFIE	ITEIE	GCD	IUE	SCLE	MA	TB	ACKNAK	STOP	START											
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											

Bits	Access	Name	Description
31:16	—	—	reserved
15	R/W	FM	Fast Mode ICR (Standard I²C): 0 = 100 kbps operation 1 = 400 kbps operation PICR (Power I²C): 0 = 40 kbps operation 1 = 160 kbps operation
14	R/W	UR	Unit Reset 0 = No reset. 1 = Reset the I²C interface only.
13	R/W	SADIE	Slave Address Detected Interrupt Enable 0 = Disable interrupt. 1 = Enables the I²C interface to interrupt the processor upon detecting a slave address match or a general call address.
12	R/W	ALDIE	Arbitration Loss Detected Interrupt Enable 0 = Disable interrupt. 1 = Enables the I²C interface to interrupt the processor upon losing arbitration while in master mode.
11	R/W	SSDIE	Slave STOP Detected Interrupt Enable 0 = Disable interrupt. 1 = Enables the I²C interface to interrupt the processor when it detects a STOP condition while in slave mode.

Table 9-8. ICR, PICR Bit Definitions (Sheet 2 of 4)

	Physical Address 0x4030_1690 0x40F0_0190												ICR PICR												Standard I²C Power I²C											
User Settings																																				
Bit	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																FM	UR	SADIE	ALDIE	SSDIE	BEIE	DRFIE	ITEIE	GCD	IUE	SCLE	MA	TB	ACKNAK	STOP	START
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
10	R/W	BEIE	Bus Error Interrupt Enable 0 = Disable interrupt. 1 = Enables the I²C interface to interrupt the processor for the following I²C bus errors: • As a master transmitter, no Ack was detected after a byte was sent. • As a slave receiver, the I²C interface generated a NAK pulse. **NOTE:** Software is responsible for guaranteeing that misplaced START and STOP conditions do not occur. See also Section 9.2.
9	R/W	DRFIE	DBR Receive Full Interrupt Enable 0 = Disable interrupt. 1 = Enables the I²C interface to interrupt the processor when the IDBR has received a data byte from the I²C bus.
8	R/W	ITEIE	IDBR Transmit Empty Interrupt Enable 0 = Disable interrupt. 1 = Enables the I²C interface to interrupt the processor after transmitting a byte onto the I²C bus.
7	R/W	GCD	General Call Disable 0 = Enable the I²C interface to respond to general call messages. 1 = Disable I²C interface response to general call messages as a slave. This bit must be set when sending a master mode general call message from the I²C interface.
6	R/W	IUE	I²C Unit Enable 0 = Disables the unit and does not master any transactions or respond to any slave transactions. 1 = Enables the I²C interface (defaults to slave-receive mode). Software must guarantee the I²C bus is idle before setting this bit. Software must guarantee that the internal clock to the Power I²C unit is enabled (CKEN[15] must be set) before setting or clearing this bit.
5	R/W	SCLEA	SCL Enable 0 = Disables the I²C interface from driving the SCL line. 1 = Enables the I²C clock output for master-mode operation.

Table 9-8. ICR, PICR Bit Definitions (Sheet 3 of 4)

Physical Address
0x4030_1690
0x40F0_0190

ICR
PICR

Standard I²C
Power I²C

	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
User Settings																																
Bit	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																FM	UR	SADIE	ALDIE	SSDIE	BEIE	DRFIE	ITEIE	GCD	IUE	SCLE	MA	TB	ACKNAK	STOP	START
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
4	R/W	MA	Master Abort Used by the I²C interface in master mode to generate a STOP without transmitting another data byte: 0 = The I²C interface transmits STOP using the ICR[STOP] bit only. 1 = The I²C interface sends STOP without data transmission. When in master-transmit mode, after transmitting a data byte, the *ICR transfer byte* bit is cleared and *IDBR transmit empty* bit is set. When no more data bytes need to be sent, setting *master abort* bit sends the STOP. The *transfer byte* bit (03) must remain clear. In master-receive mode, when a Nack is sent without a STOP (*stopICR* bit was not set) and the processor does not send a repeated START, setting this bit sends the STOP. Once again, the *transfer byte* bit (03) must remain clear.
3	R/W	TB	Transfer Byte Sends or receives a byte on the I²C bus: 0 = Cleared by I²C interface when the byte is sent/received. 1 = Send/receive a byte. The processor can monitor this bit to determine when the byte transfer has completed. In master or slave mode, after each byte transfer including ACKNOWLEDGE pulse, the I²C interface holds the SCL line low (inserting wait states) until TB is set.

Table 9-8. ICR, PICR Bit Definitions (Sheet 4 of 4)

	Physical Address 0x4030_1690 0x40F0_0190												ICR PICR												Standard I²C Power I²C											
User Settings																																				
Bit	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																FM	UR	SADIE	ALDIE	SSDIE	BEIE	DRFIE	ITEIE	GCD	IUE	SCLE	MA	TB	ACKNAK	STOP	START
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	R/W	ACKNAK	Positive/Negative Acknowledge Defines the type of ACKNOWLEDGE pulse sent by the I²C interface when in master receive mode: 0 = Send a positive acknowledge (ACK) pulse after receiving a data byte. 1 = Send a negative acknowledge (NAK) pulse after receiving a data byte. The I²C interface automatically sends an ACK pulse when responding to its slave address or when responding in slave-receive mode, regardless of the ACKNAK control-bit setting.
1	R/W	STOP	Stop Initiates a STOP condition after transferring the next data byte on the I²C bus when in master mode. In master-receive mode, the ACKNAK control bit must be set in conjunction with the STOP bit. See Section 9.4.3.3 for details of the STOP state. 0 = Do not send a STOP. 1 = Send a STOP.
0	R/W	START	Start Initiates a START condition I²C when in master mode. See Section 9.4.3.1 for details of the START state. 0 = Do not send a START pulse. 1 = Send a START pulse.

9.5.2 I²C Status Registers (ISR, PISR)

The ISR signals I²C interrupts to the PXA27x processor interrupt controller. Software can use the ISR bits to check the status of the I²C unit and bus. ISR bits [9:5] are updated after the ACK/NAK bit is completed on the I²C bus.

The ISR also clears the following interrupts signaled from the I²C interface:

- IDBR receive full
- IDBR transmit empty
- Slave address detected
- Bus error detected
- STOP condition detected
- Arbitration lost

Table 9-9. ISR, PISR Bit Definitions (Sheet 1 of 2)

Physical Address
0x4030_1698
0x40F0_0198

ISR
PISR

Standard I²C
Power I²C

User Settings

Bit	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																					BED	SAD	GCAD	IRF	ITE	ALD	SSD	IBB	UB	ACKNAK	RWM
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:11	—	—	reserved
10	Read Clear	BED	Bus Error Detected 0 = No error detected. 1 = The I²C interface sets this bit when it detects one of the following error conditions: • As a master transmitter, no Ack was detected on the interface after a byte was sent. • As a slave receiver, the I²C interface generates a Nack pulse. **NOTE:** When an error occurs, I²C bus transactions continue. Software must guarantee that misplaced START and STOP conditions do not occur. See Section 9.4.7.
9	Read Clear	SAD	Slave Address Detected 0 = No slave address was detected. 1 = The I²C interface detected a seven-bit address that matches the general call address or ISAR. An interrupt is signaled when enabled in the ICR.
8	Read Clear	GCAD	General Call Address Detected 0 = No general call address received. 1 = I²C interface received a general call address.
7	Read Clear	IRF	IDBR Receive Full 0 = The IDBR has not received a new data byte or the I²C interface is idle. 1 = The IDBR register received a new data byte from the I²C bus. An interrupt is signaled when enabled in the ICR.
6	Read Clear	ITE	IDBR Transmit Empty 0 = The data byte is still being transmitted. 1 = The I²C interface has finished transmitting a data byte on the I²C bus. An interrupt is signaled when enabled in the ICR.
5	Read Clear	ALD	Arbitration Loss Detected Used during multi-master operation: 0 = Cleared when arbitration is won or never took place. 1 = Set when the I²C interface loses arbitration.
4	Read Clear	SSD	Slave STOP Detected 0 = No STOP detected. 1 = Set when the I²C interface detects a STOP while in slave-receive or slave-transmit mode.

Table 9-9. ISR, PISR Bit Definitions (Sheet 2 of 2)

	Physical Address 0x4030_1698 0x40F0_0198												ISR PISR											Standard I²C Power I²C											
User Settings																																			
Bit	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																					BED	SAD	GCAD	IRF	ITE	ALD	SSD	IBB	UB	ACKNAK	RWM
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
3	R	IBB	I²C Bus Busy 0 = I²C bus is idle or the I²C interface is using the bus (unit busy). 1 = Set when the I²C bus is busy but the processor's I²C interface is not involved in the transaction.
2	R	UB	Unit Busy 0 = I²C interface not busy. 1 = Set when the processor's I²C interface is busy. This is defined as the time between the first START and STOP.
1	R	ACKNAK	Ack/Nack Status 0 = The I²C interface received or sent an Ack on the bus. 1 = The I²C interface received or sent a Nack. Used in slave-transmit mode to determine when the byte transferred is the last one. This bit is updated after each byte and Ack/Nack information is received.
0	R	RWM	Read/Write Mode 0 = The I²C interface is in master-transmit or slave-receive mode. 1 = The I²C interface is in master-receive or slave-transmit mode. This is the R/nW bit of the slave address. It is automatically cleared by hardware after a STOP state.

9.5.3 I²C Slave Address Registers (ISAR, PISAR)

The ISAR (see Table 9-10) defines the I²C interface seven-bit slave address. In slave-receive mode, the PXA27x processor responds when the seven-bit address matches the value in this register. The processor writes this register before it enables I²C operations. The ISAR is fully programmable (no address is assigned to the I²C interface) so it can be set to a value other than those of hard-wired I²C slave peripherals in the system. If the processor is reset, the ISAR is not affected. The ISAR register default value is 0000000_2.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 9-10. ISAR, PISAR Bit Definitions

	Physical Address 0x4030_16A0 0x40F0_01A0								ISAR PISAR																Standard I²C Power I²C							
User Settings																																
Bit	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																								Slave Address							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:07	—	—	reserved
6:0	R/W	Slave Address	Slave Address The seven-bit address to which the I²C interface responds when in slave-receive mode.

9.5.4 I²C Data Buffer Register (IDBR, PIDBR)

The PXA27x processor uses the I²C Data Buffer register to transmit and receive data from the I²C bus. The IDBR is accessed by the programmed I/O on one side and by the I²C Shift register on the other. The IDBR receives data coming into the I²C unit after a full byte is received and acknowledged. The processor core writes data going out of the I²C interface to the IDBR and sends it to the serial bus.

When the I²C interface is in transmit mode (master or slave), the processor writes data to the IDBR over the internal bus. The processor writes data to the IDBR when a master transaction is initiated or when the IDBR transmit-empty interrupt is signaled. Data moves from the IDBR to the Shift register when the transfer-byte bit is set. The IDBR transmit-empty interrupt is signaled (if enabled) when a byte is transferred on the I²C bus and the acknowledge cycle is complete. If the IDBR is not written by the processor and a STOP condition is not in place before the I²C bus is ready to transfer the next byte packet, the I²C unit inserts wait states until the processor writes the IDBR and sets the transfer-byte bit.

When the I²C interface is in receive mode (master or slave), the processor reads IDBR data over the internal bus. The processor reads data from the IDBR when the IDBR receive-full interrupt is signaled. The data moves from the Shift register to the IDBR when the ACKNOWLEDGE cycle is complete. The I²C interface inserts wait states until the IDBR is read. See Section 9.4.6 for more information on the ACKNOWLEDGE pulse in receive mode. After the processor reads the IDBR, ICR[ACKNAK] and ICR[ACKNAK] are written, allowing the next byte transfer to proceed to the I²C bus.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 9-11. IDBR, PIDBR Bit Definitions

Physical Address 0x4030_1688 0x40F0_0188 — IDBR PIDBR — Standard I²C Power I²C

	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
User Settings																																
Bit	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																								Data Buffer							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	Data Buffer	Data Buffer Buffer for I²C bus send/receive data.

9.5.5 I²C Bus Monitor Registers (IBMR, PIBMR)

The I²C Bus Monitor register (IBMR) tracks the status of the SCL and SDA pins. The values of these pins are recorded in this read-only IBMR so software can determine when the I²C bus is hung and the I²C unit must be reset.

This a read-only register. Ignore reads from reserved bits.

Table 9-12. IBMR, PIBMR Bit Definitions

Physical Address 0x4030_1680 0x40F0_0180 — IBMR PIBMR — Standard I²C Power I²C

	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
User Settings																																
Bit	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																														SCL	SDA
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1

Bits	Access	Name	Description
31:2	—	—	reserved
1	R	SCL	IBMR[SCL] continuously reflects the value of the SCL pin.
0	R	SDA	IBMR[SDA] continuously reflects the value of the SDA pin.

9.6 Register Summary

Table 9-13 summarizes the standard interface I²C registers, which are located in the peripheral memory-mapped address space. Table 9-14 summarizes the power-manager I²C registers.

Table 9-13. Standard I²C Register Summary

Address	Name	Description	Page
0x4030_1680	IBMR	I²C Bus Monitor register	9-30
0x4030_1684	—	reserved	—
0x4030_1688	IDBR	I²C Data Buffer register	9-29
0x4030_168C	—	reserved	—
0x4030_1690	ICR	I²C Control register	9-23
0x4030_1694	—	reserved	—
0x4030_1698	ISR	I²C Status register	9-26
0x4030_169C	—	reserved	—
0x4030_16A0	ISAR	I²C Slave Address register	9-28
0x4030_16A4–0x403F_FFFC	—	reserved	—

Table 9-14. Power I²C Register Summary

Address	Name	Description	Page
0x40F0_0180	PIBMR	Power Manager I²C Bus Monitor register	9-30
0x40F0_0184	—	reserved	—
0x40F0_0188	PIDBR	Power Manager I²C Data Buffer register	9-29
0x40F0_018C	—	reserved	—
0x40F0_0190	PICR	Power Manager I²C Control register	9-23
0x40F0_0194	—	reserved	—
0x40F0_0198	PISR	Power Manager I²C Status register	9-26
0x40F0_019C	—	reserved	—
0x40F0_01A0	PISAR	Power Manager I²C Slave Address register	9-28
0x40F0_01A4–0x40FF_FFFC	—	reserved	—

UARTs 10

This chapter describes the universal asynchronous receiver/transmitter (UART) serial ports included in the PXA27x processor. The serial ports are controlled using direct-memory access (DMA) or programmed I/O. The PXA27x processor has three UARTs: full-function (FFUART), Bluetooth (BTUART), and standard (STUART). All UARTs use the same programming model.

10.1 Overview

Each serial port contains a UART and a slow infrared-transmit encoder and receive decoder that conform to the IrDA serial-infrared specification.[1]

Each UART performs serial-to-parallel conversion on data characters received from a peripheral device or a modem and parallel-to-serial conversion on data characters received from the processor. The processor can read a UART's complete status during functional operation. Status information includes the type and condition of transfer operations and error conditions (parity, overrun, framing, or break interrupt) associated with the UART.

Each serial port operates in either FIFO or non-FIFO mode. In FIFO mode, a 64-byte transmit FIFO holds data from the processor until it is transmitted on the serial link, and a 64-byte receive FIFO buffers data from the serial link until it is read by the processor. In non-FIFO mode, the transmit and receive FIFOs are bypassed.

Each UART includes a programmable baud-rate generator that can divide the input clock by 1 to $(2^{16} - 1)$. This produces a 16X clock that can be used to drive the internal transmit and receive logic. Software can program interrupts to meet its requirements, which minimizes the number of computations required to handle the communications link. Each UART operates in an environment that is either controlled by software and can be polled or is interrupt-driven.

All three UARTs support the 16550A and 16750[2] functions, but are slightly different in the features supported.

10.1.1 Full-Function UART

The FFUART supports modem-control capability. The maximum baud rate is 921,600 bps.

10.1.2 Bluetooth UART

The BTUART is a high-speed UART that supports baud rates up to 921,600 bps and can be connected to a Bluetooth module. It supports the functions in the feature list, but supports only two modem control pins (nCTS and nRTS).

1. Infrared Data Association, *Serial Infrared Physical Layer Link Specification*, October 17, 1995, Version 1.1
2. The 16550A was originally produced by National Semiconductor Inc. The 16750 is produced as the TL16C750 by Texas Instruments.

10.1.3 Standard UART

The STUART supports all functions in the feature list, but does not support modem-control capability. The maximum baud rate is 921,600 bps.

10.1.4 Compatibility with 16550A and 16750

The UARTs are functionally compatible with the 16550A and 16750 industry standards. Each UART supports most of the 16550A and 16750 functions, as well as the following features:

- DMA requests for transmit and receive data services
- Slow infrared-asynchronous interface
- Non-return-to-zero (NRZ) encoding/decoding function
- 64-byte transmit/receive FIFO buffers
- Programmable receive FIFO trigger threshold
- Auto baud-rate detection
- Auto flow

10.2 Features

The UARTs share the following features:

- Functionally compatible with the 16550A and 16750
- Ability to add or delete standard asynchronous communication bits (start, stop, and parity) in the serial data
- Independently controlled transmit, receive, line status, and data-set interrupts
- Programmable baud-rate generator that allows the internal clock to be divided by 1 to $(2^{16} - 1)$ to generate an internal 16X clock
- Modem control functions (nCTS, nRTS, nDSR, nDTR, nRI, and nDCD)
- Auto-flow capability controls data I/O without generating interrupts:
  - nRTS (output) controlled by UART receive FIFO
  - nCTS (input) from modem controls UART transmitter
- Fully programmable serial interface:
  - 5-, 6-, 7-, or 8-bit characters
  - Even, odd, and no-parity detection
  - 1, 1½, or 2 stop-bit generation
  - Baud-rate generation up to 921 kbps for all UARTs
  - False start-bit detection
- 64-byte transmit FIFO
- 64-byte receive FIFO

- Complete status-reporting capability
- Ability to generate and detect line breaks
- Internal diagnostic capabilities that include:
  - — Loopback controls for communications-link fault isolation
  - — Break, parity, and framing-error simulation
- Fully prioritized interrupt system controls
- Separate DMA requests for transmit and receive data services
- Slow infrared asynchronous interface that conforms to the Infrared Data Association (IrDA) specification

10.3 Signal Descriptions

Table 10-1 lists and describes each external signal that is connected to a UART module. The pins transmit digital CMOS-level signals and are connected to the PXA27x processor through GPIOs. Refer to Section 24, "General-Purpose I/O Controller" for details on the GPIOs.

Table 10-1. UARTs I/O Signal Descriptions (Sheet 1 of 2)

Name	Type	Description
RXD	Input	**Serial Input**—Serial data input to the receive shift register. In infrared mode, it is connected to the infrared receiver input. This signal is present on all three UARTs.
TXD	Output	**Serial Output**—Serial data output to the communications-link peripheral, modem, or data set. The TXD signal is set to the logic 1 state upon a reset operation. It is connected to the output of the infrared transmitter in infrared mode. This signal is present on all three UARTs.
nCTS	Input	**Clear to Send**—When low, indicates that the modem or data set is ready to exchange data. The nCTS signal is a modem status input, and its condition can be tested by reading bit 4 (CTS) of the Modem Status register (MSR). MSR[CTS] is the complement of the nCTS signal. MSR[DCTS] indicates whether the nCTS input has changed state since the last time MSR was read. nCTS has no effect on the transmitter. This signal is present on the FFUART and BTUART. When MSR[CTS] changes state and the modem-status interrupt is enabled, an interrupt is generated. **Non-Auto-Flow Mode**—When not in auto-flow mode, MSR[CTS] indicates the state of nCTS. MSR[DCTS] indicates whether the nCTS input has changed state since the previous reading of MSR. nCTS has no effect on the transmitter. The user can program the UART to interrupt the processor when DCTS changes state. Software can then stall the outgoing data stream by starving the transmit FIFO or disabling the UART with the IER register. **NOTE:** If UART transmission is stalled by disabling the UART, no MSR interrupt is received when nCTS reasserts, because disabling the UART also disables interrupts. To get around this, either use auto-CTS in auto-flow mode or program the nCTS GPIO pin to interrupt. **Auto-Flow Mode**—In this mode, the UART transmit circuity checks the state of nCTS before transmitting each byte. IF nCTS is high, no data is transmitted.
nDSR	Input	**Data Set Ready**—When low, indicates that the modem or data set is ready to establish a communications link with a UART. The nDSR signal is a modem-status input. Its condition can be tested by reading MSR[DSR], which is the complement of nDSR. MSR[DDSR] indicates whether the nDSR input has changed state since MSR was last read. This signal is present only on the FFUART. When MSR[DSR] changes state, an interrupt is generated if the modem-status interrupt is enabled.
nDCD	Input	**Data Carrier Detect**—When low, indicates that the data carrier has been detected by the modem or data set. The nDCD signal is a modem-status input. Its condition can be tested by reading MSR[DCD], which is the complement of the nDCD signal. MSR[DDCD] indicates whether the nDCD input has changed state since the previous reading of MSR. nDCD has no effect on the receiver. This signal is present only on the FFUART. When the DCD bit changes state and the modem-status interrupt is enabled, an interrupt is generated.

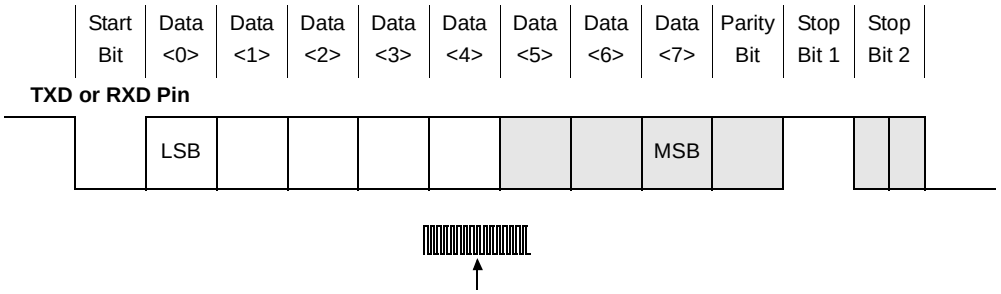
Table 10-1. UARTs I/O Signal Descriptions (Sheet 2 of 2)

Name	Type	Description
nRI	Input	**Ring Indicator**—When low, indicates that the modem or data set has received a telephone ringing signal. The nRI signal is a modem-status input. Its condition can be tested by reading MSR[RI], which is the complement of the nRI signal. MSR[TERI] (trailing-edge-of-ring indicator) indicates whether the nRI input has changed from low to high since MSR was last read. This signal is present only on the FFUART. When the RI bit of the MSR changes from a high to low state and the modem-status interrupt is enabled, an interrupt is generated.
nDTR	Output	**Data Terminal Ready**—When low, signals the modem or the data set that the UART is ready to establish a communications link. To assert the nDTR output (active low), set MSR[DTR], which is the complement of the output signal. A reset operation deasserts this signal (high). Loopback-mode operation holds nDTR deasserted. This signal is present only on the FFUART.
nRTS	Output	**Request to Send**—When low, signals the modem or the data set that the UART is ready to exchange data. To assert the nRTS output (active low), set MSR[RTS], which is the complement of the output signal. A reset operation deasserts this signal (high). Loopback-mode operation holds nRTS deasserted. This signal is used by the FFUART and BTUART. **Non-Auto-Flow Mode**—To assert the nRTS output (active low), set MSR[RTS]. **Auto-Flow Mode**—nRTS is automatically asserted by the auto-flow circuitry when the receive buffer exceeds its programmed trigger threshold. It is deasserted when enough bytes are removed from the buffer to lower the data level back to the trigger threshold.

10.4 Operation

Figure 10-1 shows the format of a UART data frame.

Figure 10-1. Example UART Data Frame



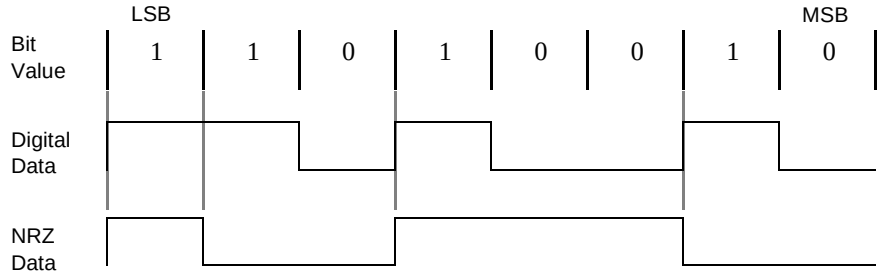
The receive-data sample counter frequency is 16 times the value of the bit frequency. The 16X clock is created by the baud-rate generator. Each bit is sampled three times in the middle. Shaded bits in Figure 10-1 are optional and can be programmed by software.

Each data frame is between 7 and 12 bits long, depending on the size of the data programmed, whether parity is enabled, and the number of stop bits. A data frame begins by transmitting a start bit that is represented by a high-to-low transition. The start bit is followed by from 5 to 8 bits of data that begin with the least significant bit (LSB). The data bits are followed by an optional parity bit. The parity bit is set if even-parity is enabled and the data byte has an odd number of ones, or if odd parity is enabled and the data byte has an even number of ones. The data frame ends with 1, 1½, or 2 stop bits, as programmed by software. The stop bits are represented by 1, 1½, or 2 successive bit periods of logic 1.

Each UART has two FIFOs: one transmit and one receive. The transmit FIFO is 64 bytes deep and 8 bits wide. The receive FIFO is 64 bytes deep and 11 bits wide. Three bits are used for tracking errors.

The UART can use non-return-to-zero (NRZ) coding to represent individual bit values. To enable NRZ coding, set IER[5]. A bit value of 0b1 is represented by a line transition, and 0b0 is represented by no line transition. Figure 10-2 shows the data byte 0b0100_1011 in NRZ coding. The byte's LSB is transmitted first.

Figure 10-2. Example NRZ Bit Encoding—0b0100_1011



10.4.1 Reset

The UARTs are disabled on reset. To enable a UART, software must program the GPIO registers (see Section 24, "General-Purpose I/O Controller"), then set IER[UUE]. When the UART is enabled, the receiver waits for a frame-start bit and the transmitter sends data if it is available in the Transmit Holding register. Transmit data can be written to the Transmit Holding register before the UART unit is enabled. In FIFO mode, data is transmitted from the FIFO to the Transmit Holding register before it goes to the pin.

When the UART unit is disabled, the transmitter or receiver finishes the current byte and stops transmitting or receiving more data. Data in the FIFO is not cleared, and transmission resumes when the UART is enabled.

10.4.2 FIFO Operation

Each UART has a transmit FIFO and a receive FIFO, each FIFO holding 64 characters of data. There are three separate methods for moving data into and out of the FIFOs: interrupts, polling, and DMA.

Note: In polled mode and interrupt mode, the end-of-chain interrupt does not occur. In DMA mode, software must set the DMA-stop interrupt on the last descriptor in the chain to avoid errors.

10.4.2.1 FIFO Interrupt Mode Operation

10.4.2.1.1 Receive Interrupt

For a receive interrupt to occur, the receive FIFO and receive interrupts must be enabled. IIR[IID] changes to show that receive data is available when the FIFO reaches its trigger threshold. IIR[IID] changes to show the next waiting interrupt when the FIFO drops below the trigger threshold. A change in IIR[IID] triggers an interrupt to the core. Software reads IIR[IID] to determine the cause of the interrupt.

The receive-line-status interrupt (IIR = 0xC6) has the highest priority; the received-data-available interrupt (IIR = 0xC4) is lower. The line-status interrupt occurs only when the character at the front of the FIFO has errors.

The data-ready bit (LSR[DR]) is set when a character is transferred from the Shift register to the receive FIFO. LSR[DR] is cleared when the FIFO is empty.

10.4.2.1.2 Character Time-Out Interrupt

A character time-out interrupt occurs when the receive FIFO and receive time-out interrupt are enabled and all of the following conditions exist:

- At least one character is in the FIFO.
- The most recently received character was received more than four continuous character-times ago. If two stop-bits are programmed, the second is included in this interval.
- The most recent FIFO read was performed more than four continuous character-times ago.

After the processor reads one character from the receive FIFO or a new start bit is received, the time-out interrupt is cleared and the time out is reset. If a time-out interrupt has not occurred, the time out is reset when a new character is received or the processor reads the receive FIFO.

10.4.2.1.3 Transmit Interrupt

Transmit interrupts can occur only when the transmit FIFO and transmit interrupt are enabled. The transmit-data-request interrupt occurs when the transmit FIFO is at least half-empty. The interrupt is cleared when the THR is written or the IIR is read.

10.4.2.2 FIFO Polled Mode Operation

When the FIFOs are enabled, clearing both IER[DMAE] and IER[4:0] places the serial port in FIFO-polled operating mode. The receiver and the transmitter are controlled separately. Either one or both can be in polled mode. In polled mode, software checks receiver and transmitter status using the LSR. The processor polls the following bits for receive and transmit data service:

- Receive Data Service—The processor checks the LSR[DR] (data ready) bit, which is set when one or more bytes remain in the receive FIFO or Receive Buffer register (RBR).
- Transmit Data Service—The processor checks the LSR[TDRQ] (transmit data request) bit, which is set when the transmitter needs data.

The processor can also check the LSR[TEMT] (transmitter empty) bit, which is set when the transmit FIFO or holding register is empty.

10.4.2.3 FIFO DMA Mode Operation

The UART has two DMA requests: one for transmit data service and one for receive data service. DMA requests are generated in FIFO mode only. The requests are activated by setting IER[DMAE].

- Data Transmitter Data Service—When IER[DMAE] is set, if the transmit FIFO is absolutely less than half full, the transmit-DMA request is generated. The DMA controller then writes data to the FIFO. For each DMA request, the DMA controller can send 8, 16, or 32 bytes of data to the FIFO. The actual number of bytes to be transmitted is programmed in the DMA controller. The UART accepts partial-word or full-word transfers of one, two, three, or four

consecutive bytes from the DMA controller or processor I/O bridge when in 32-bit peripheral bus mode.

- Data Receiver Data Service—When IER[DMAE] is set, the receive DMA request is generated when the receive FIFO reaches its trigger threshold with no errors in its entries. The DMA controller then reads data from the FIFO. For each DMA request, the DMA controller can read 8, 16, or 32 bytes of data from the FIFO. The actual number of bytes to be read is programmed in the DMA controller along with the bus width. When in 32-bit peripheral-bus mode, the DMA controller always attempts to read four bytes of data per transfer. In the case where fewer than four bytes are being transferred, the valid bytes are indicated by a data-valid bus shared between the UART and the DMA controller. The UART can send 1, 2, 3, or 4 bytes of data per bus transaction.

10.4.2.4 DMA Receive Programming Errors

If the DMA channel stops prematurely due to the end of a descriptor chain or other error, the processor must be notified, since the DMA controller can no longer service the UART FIFOs. If this occurs, the processor must correct the situation by programming another descriptor or by servicing the FIFOs using interrupt or polling mode, as described above. There are two methods for notifying the processor of a stopped DMA channel:

1. Program the DMA controller to interrupt on the event of a stopped channel by setting DCSR[StopIrqEn].
2. For the receive channel, the UART interrupts with an end-of-descriptor chain (EOC) interrupt if FCR[TRAIL] is set, such that the UART makes a DMA request to remove trailing bytes (See Section 10.4.2.6). Using the UART interrupt for the receive channel is preferable to the DMA DCSR interrupt, because extra logic exists to ensure that the UART EOC interrupt asserts only when necessary. For example, a UART EOC interrupt does not assert if the UART has completed the reception of its message (indicated by the character time-out timer) and the receive FIFO is empty. The IIR[EOC] interrupt does not assert if FCR[TRAIL] is cleared.

10.4.2.5 DMA Error Handling

If an error occurs while in DMA mode:

- The receive-DMA requests are disabled.
- The error interrupt IIR[IID] is generated.

The processor must now read out the error bytes through programmed I/O (PIO). When all errors have been removed from the FIFO, the receive DMA requests are once again enabled by the UART.

If an error occurs when the receive FIFO trigger threshold has been reached such that a receive DMA request is set, software must wait for the DMA to finish the transfer before reading out the error bytes through PIO. Otherwise, FIFO underflow could occur.

Note: Ensure that the DMA controller has completed the previous receive DMA requests before the error interrupt handler begins to clear the errors from the FIFO. Otherwise, FIFO underflow could occur.

10.4.2.6 Removing Trailing Bytes in DMA Mode

When the number of entries in the receive FIFO is less than its trigger threshold, and no additional data is received, the remaining bytes are called *trailing bytes*. To program the UART to make a DMA request to remove the trailing bytes, set FCR[TRAIL]. Setting FCR[TRAIL] also enables the IIR[EOC] interrupt, described in Section 10.4.2.4.

When the DMA controller is removing trailing bytes, a request is automatically issued for the remaining number of bytes in the receive buffer. The DMA controller then empties the contents of the receive buffer unless the DMA reaches the end of its descriptor chain. If the DMA reaches the end of the descriptor chain while removing trailing bytes, the processor is forced to take over, since the DMA controller can no longer service the UART request until a new chain is linked. In this situation, the UART sets IIR[EOC] if data exists in the receive FIFO, and if IER[RTOIE] is set, it also sets IIR[TOD].

The remaining bytes must then be removed using the processor, as described in Section 10.4.2.1.

10.4.2.6.1 False EOR Due to Character Time-Out Expiration

It is possible for a false EOR to be asserted by the UART in the middle of receiving a message if a pause in the remote data transmissions is long enough to cause the time-out counter to expire. This pause causes an EOR to be sent to the DMA controller if in DMA mode.

If this pause occurs:

The EOR is applied to the last byte of data in the FIFO when the DMA responds to the EOR request. The EOR is *not* applied to the last byte in the FIFO at the time of the character time out. In other words, if remote transmission resumes before the DMA responds to the EOR request, the EOR flag is applied to the new data that entered the FIFO and not to the last byte in the FIFO at the time of the character time out.

10.4.2.6.2 EOR Must Be Serviced Prior to Transmission of New Message

A caveat to this behavior could be encountered under legitimate EOR situations: for example, if message A ends with 3 bytes in the FIFO, an EOR request is made to the DMA controller to remove these bytes. If transmission of a new message B resumes before the DMA controller responded to the EOR request of message A, the EOR could be applied to the first byte of message B, if this byte is written into the FIFO before the DMA controller responds to message A's EOR request. Although this situation could occur, it would be considered a programming error, since the higher communications protocol must prevent message B transmission until the local receiver acknowledges the receipt of message A. The exception would be if enough new bytes enter the FIFO to push the FIFO level to its watermark. If this push occurs, the request is treated as a normal service request, and no EOR flag is asserted to the DMA controller.

10.4.3 Auto-Flow Control

Auto-flow control uses the clear-to-send (nCTS) and request-to-send (nRTS) signals to control the flow of data between the UART and external modem. When auto-flow is enabled, the remote device is not allowed to send data unless the UART asserts nRTS. If the UART de-asserts nRTS while the remote device is sending data, the remote device is allowed to send one additional byte after nRTS is de-asserted. An overflow could occur if the remote device violates this rule. Likewise, the UART is not allowed to transmit data unless the remote device asserts nCTS. Using this feature increases system efficiency and eliminates the possibility of a receive-FIFO-overflow error due to long interrupt latency.

Auto-flow mode can be used in two ways: full auto-flow, automating both nCTS and nRTS; and half auto-flow, automating only nCTS. To enable full auto-flow, set bits MCR[1] and MCR[5]. To enable auto-nCTS-only mode, set MCR[5] and clear MCR[1].

10.4.3.1 nRTS (UART Output)

When in full auto-flow mode, nRTS is asserted when the UART FIFO is ready to receive data from the remote transmitter. This assertion occurs when the amount of data in the receive FIFO is below the programmable trigger threshold value. When the amount of data in the receive FIFO reaches the programmable trigger threshold, nRTS is de-asserted. It is re-asserted when enough bytes are removed from the FIFO to lower the data level below the trigger threshold.

10.4.3.2 nCTS (UART Input)

When in full- or half-auto-flow mode, nCTS is asserted by the remote receiver when the receiver is ready to receive data from the UART. The UART checks nCTS before sending the next byte of data and does not transmit the byte until nCTS is low. If nCTS goes high while the transfer of a byte is in progress, the transmitter completes this byte.

If UART transmission is stalled by disabling the UART, none of the interrupts in the Modem Status register (MSR) indicates an interrupt when nCTS re-asserts because disabling the UART also disables interrupts. Intel recommends using auto-CTS in auto-flow mode.

Note: Auto-flow mode can be used only in conjunction with FIFO mode.

10.4.4 Auto-Baud-Rate Detection

Each UART supports auto-baud-rate detection. When enabled, the UART counts the number of 14.857-MHz clock cycles within the start-bit pulse. This number is then written into the Auto-Baud-Count register (ACR; see Table 10-14) and is used to calculate the baud rate. When the ACR is written, an auto-baud-lock interrupt is generated (if enabled), and the UART automatically programs the Divisor Latch registers with the appropriate baud rate. If preferred, the processor can read ACR and use this information to program the Divisor-Latch registers with a baud rate calculated by the processor. After the baud rate has been programmed, it is the responsibility of the processor to verify that the predetermined characters (usually **AT** or **at**) are being received correctly.

If the UART is to program the Divisor Latch registers, software can use either of two methods for auto-baud calculation (table based and formula based). The method is selected using the Auto-Baud Control register, bit ABR[ABT].

- **Formula method**—Any baud rate allowed in Section 10.4.7 can be programmed by the UART. This method works well for higher baud rates, but it could fail below 28.8 kbps if the remote transmitter's actual baud rate differs by more than one percent of its target.
- **Table method**—Is more immune to such errors, because the table rejects uncommon baud rates and rounds to the common ones. The table method allows any baud rate defined by the formula in Section 10.4.7 above 28.8 kbps. Below 28.8 kbps, the only baud rates that can be programmed by the UART are 19200, 14400, 9600, 4800, 1200, and 300 baud.

When the baud rate is detected, the auto-baud circuitry disables itself by clearing ABR[ABE]. To re-enable auto-baud detection, reset ABR[ABE]. Changing the baud rate is not permitted when actively transmitting or receiving data.

Note: Auto-baud-rate detection is not supported in IrDA (slow infrared) mode.

See Section 10.5.8 for more information on auto-baud.

10.4.5 32-Bit Peripheral Bus

Each UART supports an 8- (default) or 32-bit peripheral bus. If a 32-bit bus is preferred, set the bus bit in the FIFO Control register, FCR[5]. The bytes are written in little-endian format (7:0) with byte 3 (the most recent byte) starting at bit 31, byte 2 starting at bit 23, and so on.

- **8-Bit Mode—**Only the least significant byte contains valid data on the peripheral bus. The upper 24 bits are ignored.
- **32-Bit Mode—**The UART can read or write partial words of one, two, three, or four continuous bytes from the peripheral bus. The method in which the valid bytes of data are determined differs, depending on whether the transaction is being handled by the DMA controller or PIO.
- **DMA**—The DMA controller can read or write one, two, three, or four continuous bytes per word. The number of valid bytes available per word is determined internally between the DMA controller and the UART.
- **PIO**—The processor is restricted to reading or writing one, two, or four bytes per word. When reading, the processor must read the FIFO Occupancy register (FOR) to retrieve the number of bytes available in the receive buffer. If the number of bytes available is four or greater, the processor can request any number of bytes per word (except three). If the number is less than four, software must request the proper number of bytes. When three bytes are remaining, software must request either two bytes followed by one byte or one byte followed by two bytes. If the processor reads more than the number of bytes available in the receive buffer, the UART retrieves unusable data for the invalid bytes. The receive FIFO counters do not increment.

Note: The receive and transmit FIFOs must be enabled when in 32-bit mode.

10.4.6 Slow Infrared Asynchronous Interface

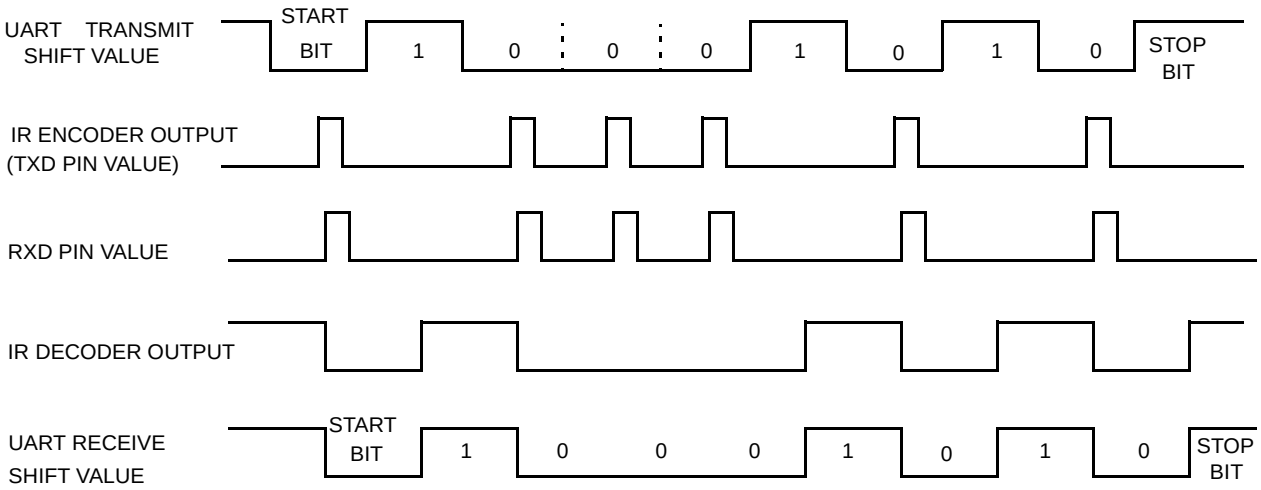
The slow-infrared (SIR) interface is provided in each UART to support two-way wireless communications using infrared transmission. SIR provides a transmit encoder and receive decoder to support a physical link that conforms to the IrDA serial infrared specification.

The SIR interface does not contain the actual IR LED driver or the receive amplifier. The I/O pins attached to the SIR have only digital CMOS-level signals. SIR supports two-way communication, but full-duplex communication is not possible, because reflections from the transmit LED enter the receiver. SIR supports frequencies up to 115.2 kbps. Because the input clock is 14.857 MHz, the baud divisor must be eight or more.

10.4.6.1 Operation

The SIR modulation technique works with 5-, 6-, 7-, or 8-bit characters with an optional parity bit. The data is preceded by a zero-value start bit and ends with one or more stop bits. The encoding scheme sends a pulse 3/16 of a bit wide in the middle of every zero-value bit and sends no pulses for bits with a value of one. The pulse for each zero-value bit must occur, even for consecutive bits with no edge between them. Figure 10-3 shows an example of transmit/receive operation.

Figure 10-3. IR Transmit and Receive Example

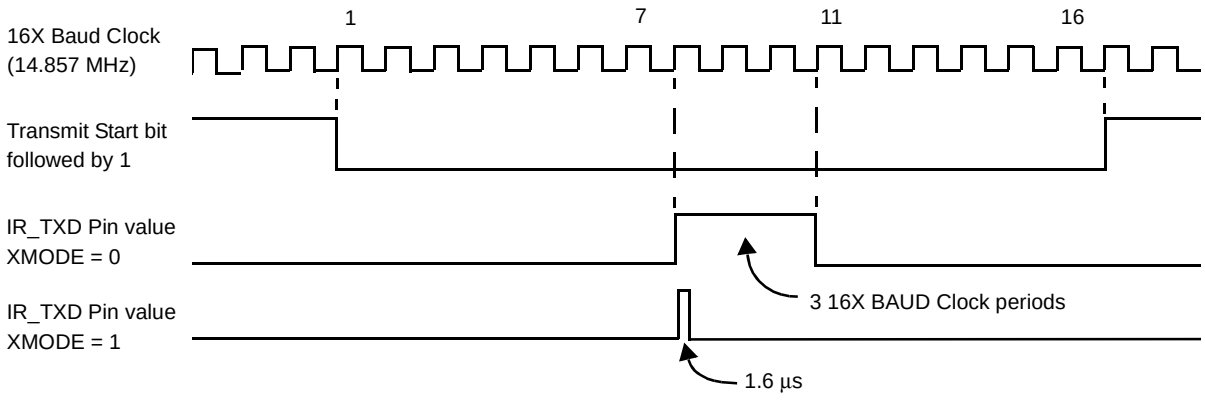


The top line in Figure 10-3 shows an asynchronous transmission as it is sent from the UART. The second line shows the pulses generated by the IR encoder at the TXD pin. A pulse is generated in the middle of the START bit and any data bit that is a zero. The third line shows the values received at the RXD input pin. The fourth line shows the receive decoder's output. The receive decoder drives the receive data line low when it detects a pulse. The bottom line shows how the UART's receiver interprets the decoder's action. This last line is the same as the first, but it is shifted half a bit period.

Intel strongly recommends setting ISR[XMODE] for the transmit pulse width. When ISR[XMODE] is clear, each zero bit has a pulse width of 3/16 of a bit-time. When ISR[XMODE] is set, a pulse of 1.6 µs is generated in the middle of each zero bit. The shorter infrared pulse generated when ISR[XMODE] is set reduces the LED's power consumption. At 2400 bps, the LED is normally on for 78 µs for each zero bit that is transmitted. When ISR[XMODE] is set, the LED is on for only 1.6 µs. Figure 10-4 shows an example of XMODE operation.

Note: ISR[XMODE] toggles only the transmit pulse width. It does not effect the receive pulse width, which is always set to XMODE = 1.

Figure 10-4. XMODE Example



To prevent transmitter LED reflection feedback to the receiver, hardware does not allow the transmit pin to toggle if both TX and RX (ISR[RCVEIR] =1 and ISR[XMITIR] = 1) are enabled. However, software can enable the transmitter permanently by setting ISR[XMITIR] = 1, and then enable and disable the receiver, depending on what transfer is required (transmitting or receiving).

When software enables the receiver by setting ISR[RCVEIR]=1, hardware automatically disables the transmitter (assuming that software permanently enabled the transmitter by setting ISR[XMITIR] = 1). When the receiver is disabled by software (RCVEIR=0), the transmitter is (optionally) already permanently enabled by software.

10.4.7 Programmable Baud-Rate Generator

Each UART contains a programmable baud-rate generator that can take the 14.857-MHz fixed-input clock and divide it by 1 to ($2^{16} - 1$). The baud-rate generator output frequency is 16 times the baud rate. Two 8-bit Divisor Latch registers (DLL and DLH; see Section 10.5.3) store the divisor in a 16-bit binary format. Load these divisor latches during initialization to ensure that the baud-rate generator operates properly. The 16X clock stops if each Divisor Latch register is loaded with 0x0.

The baud rate of the data shifted into or out of a UART is given by the formula:

$$BaudRate = \frac{14.7456\ \text{MHz}}{(16 \times Divisor)}$$

For example: if the divisor is 24, the baud rate is 38400 bps.

Table 10-2. Theoretical Baud Rate vs. Actual Baud Rate

Required Frequency (MHz)	Required Baud Rate	Divisor	Actual Baud Rate	Actual Frequency[1] (MHz)
14.7456	300	3072	302	14.857
14.7456	1200	768	1209	14.857
14.7456	2400	384	2418	14.857
14.7456	4800	192	4836	14.857
14.7456	9600	96	9672	14.857
14.7456	19200	48	19345	14.857
14.7456	38400	24	38690	14.857
14.7456	57600	16	58035	14.857
14.7456	115200	8	116070	14.857

The divisor's reset value is 0x0002. Changing the baud rate (writing to registers DLL and DLH) is not permitted while actively transmitting or receiving data.

Note: 1) The *actual frequency* is derived from the internal dividers.

10.5 Register Descriptions

Each UART has 13 registers: 12 for UART operation and one for slow-infrared configuration. They are all 32-bit registers, but only the lower eight bits have valid data. The 12 UART operation registers share nine address locations in the I/O address space. Table 10-3 shows the registers and their addresses as offsets of a base address. The base address for each UART is 32 bits. The state of the SLCR[DLAB] bit affects the selection of some UART registers. Software must set the SLCR[DLAB] bit to access the Baud Rate Generator Divisor Latch registers.

Table 10-3. UART Register Addresses as Offsets from a Base Address

UART Register Addresses (Base + offset)	DLAB Bit Value	Register Accessed
Base	0	Receive Buffer (read-only)
Base	0	Transmit Buffer (write-only)
Base + 0x04	0	Interrupt Enable (read/write)
Base + 0x08	X	Interrupt Identification (read-only)
Base + 0x08	X	FIFO Control (write-only)
Base + 0x0C	X	Line Control (read/write)
Base + 0x10	X	Modem Control (read/write)
Base + 0x14	X	Line Status (read-only)
Base + 0x18	X	Modem Status (read-only)
Base + 0x1C	X	Scratch Pad (read/write)
Base + 0x20	X	Infrared Selection (read/write)
Base	1	Divisor Latch Low (read/write)
Base + 0x04	1	Divisor Latch High (read/write)

10.5.1 Receive Buffer Register (RBR)

In non-FIFO mode, the Receive Buffer register (RBR) holds the character(s) received by the UART Receive Shift register. If RBR (described in Table 10-4) is configured to use fewer than eight bits, the bits are right-justified, and the most significant bits (MSB) are zeroed. Reading the register empties the register and clears LSR[DR].

In FIFO mode, RBR latches the value of the data byte at the front of the FIFO.

Table 10-4. RBR Bit Definitions

Physical Address 0xBASE_0000 (DLAB = 0)																															
	RBR																								UARTs							
User Settings																															
Bit	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
	Byte 3								Byte 2								Byte 1								Byte 0							
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

Bits	Access	Name	Description
31:24	R	Byte 3	Byte 3 (valid in 32-bit peripheral bus mode only)
24:16	R	Byte 2	Byte 2 (valid in 32-bit peripheral bus mode only)
15:8	R	Byte 1	Byte 1 (valid in 32-bit peripheral bus mode only)
7:0	R	Byte 0	Byte 0

10.5.2 Transmit Holding Register (THR)

In non-FIFO mode, THR holds the data byte(s) to be transmitted next. When the Transmit Shift register (TSR) is emptied, the contents of THR are loaded into the TSR, and LSR[TDRQ] is set.

In FIFO mode, a write to THR puts data into the top of the FIFO. The data at the front of the FIFO is loaded into the TSR when the TSR is empty.

Table 10-5. THR Bit Definitions

Physical Address 0xBASE_0000 (DLAB = 0)																															
	THR																								UARTs							
User Settings																															
Bit	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
	Byte 3								Byte 2								Byte 1								Byte 0							
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

Bits	Access	Name	Description
31:24	W	Byte 3	Byte 3 (valid in 32-bit peripheral bus mode only)
24:16	W	Byte 2	Byte 2 (valid in 32-bit peripheral bus mode only)
15:8	W	Byte 1	Byte 1 (valid in 32-bit peripheral bus mode only)
7:0	W	Byte 0	Byte 0

10.5.3 Divisor Latch Registers, Low and High (DLL, DLH)

The Divisor Latch registers, DLL and DLH and described in Table 10-6 and Table 10-7, contain the divisor for the programmable baud-rate generator, as described in Section 10.4.7. Access these registers with word writes.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 10-6. DLL Bit Definitions

Physical Address 0xBASE_0000 (DLAB = 1)																															
	DLL																				UARTs										
User Settings																															
Bit	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																								DLL							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	1	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	DLL	Low-byte compare value to generate baud rate

Table 10-7. DLH Bit Definitions

Physical Address 0xBASE_0000 (DLAB = 1)																															
	DLH																				UARTs										
User Settings																															
Bit	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																								DLH							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	DLH	High-byte compare value to generate baud rate

10.5.4 Interrupt Enable Register (IER)

The Interrupt Enable register (IER), described in Table 10-8, enables the five types of interrupts that set a value in the Interrupt Identification register (IIR). To disable an interrupt, software must clear the appropriate bit in IER. Software can enable some interrupts by setting the appropriate bit.

The character-time-out-indication interrupt is separated from the received-data-available interrupt to ensure that the processor and the DMA controller do not service the receive FIFO at the same time. When a character-time-out-indication interrupt occurs, the processor must handle the data in the receive FIFO through programmed I/O.

An error interrupt is used when DMA requests are enabled. The interrupt is generated when LSR[7] is set, because a receive DMA request is not generated when the receive FIFO has an error. The error interrupt tells the processor to handle the data in the receive FIFO through programmed I/O. The error interrupt is enabled when DMA requests are enabled, and it cannot be masked. Receiver-line-status interrupts occur when the error is at the front of the FIFO.

Note: (1). When DMA requests are enabled and an interrupt occurs, software must first read LSR to see if an error interrupt exists, then check IIR for the source of the interrupt. If an interrupt occurs and LSR[FIFOE] is clear, software must read ISR to determine the error condition. When the last error byte is read from the FIFO, DMA requests are enabled automatically. Software is not required to check for the error interrupt if DMA requests are disabled, because an error interrupt occurs only when DMA requests are enabled.

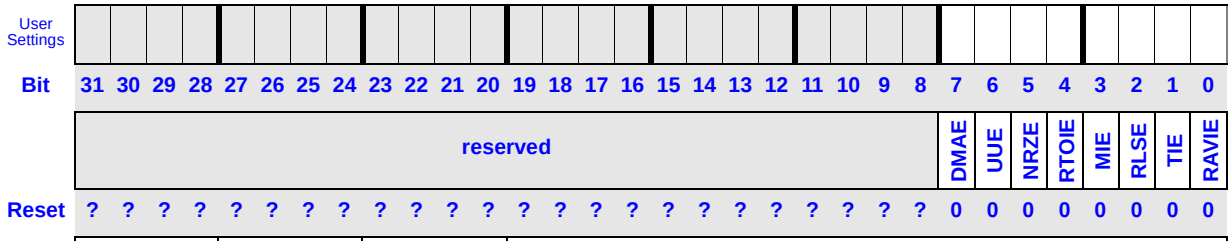
IER[7] enables DMA requests. The IER also contains the unit-enable and NRZ coding-enable control bits. Bits 7 through 4 are used differently from the standard 16550A register definition.

(2). To ensure that the DMA controller and programmed I/O do not access the same FIFO, software must not set DMAE while the TIE or RAVIE bits are set.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 10-8. IER Bit Definitions (Sheet 1 of 2)

Physical Address 0xBASE_0004 — **IER** — **UARTs**



Bits	Access	Name	Description
31:8	—	—	reserved
7	R/W	DMAE	DMA Requests Enable 0 = DMA requests are disabled. 1 = DMA requests are enabled.
6	R/W	UUE	UART Unit Enable 0 = The unit is disabled. 1 = The unit is enabled.
5	R/W	NRZE	NRZ coding Enable NRZ encoding/decoding is only used in UART mode, not in infrared mode. If the slow infrared receiver or transmitter is enabled, NRZ coding is disabled. 0 = NRZ coding disabled. 1 = NRZ coding enabled.
4	R/W	RTOIE	Receiver Time-Out Interrupt Enable (Source IIR[TOD]) 0 = Receiver data time-out interrupt disabled. 1 = Receiver data time-out interrupt enabled.
3	R/W	MIE	Modem Interrupt Enable (Source IIR[IID]) 0 = Modem status interrupt disabled. 1 = Modem status interrupt enabled.

Table 10-8. IER Bit Definitions (Sheet 2 of 2)

Physical Address 0xBASE_0004																IER								UARTs							
User Settings																															
Bit	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																								DMAE	UUE	NRZE	RTOIE	MIE	RLSE	TIE	RAVIE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	R/W	RLSE	Receiver Line Status Interrupt Enable (Source IIR[IID]) 0 = Receiver line status interrupt disabled. 1 = Receiver line status interrupt enabled.
1	R/W	TIE	Transmit Data request Interrupt Enable (Source IIR[IID]) 0 = Transmit FIFO data request interrupt disabled. 1 = Transmit FIFO data request interrupt enabled.
0	R/W	RAVIE	Receiver Data Available Interrupt Enable (Source IIR[IID]) 0 = Receiver data available (trigger threshold reached) interrupt disabled. 1 = Receiver data available (trigger threshold reached) interrupt enabled.

10.5.5 Interrupt Identification Register (IIR)

The UART prioritizes interrupts in four levels (see Table 10-9) and records them in the IIR. The IIR stores information that indicates that a prioritized interrupt is pending and identifies the source of the interrupt.

The error interrupt is reported separately in LSR. In non-DMA mode, IIR is updated immediately to signify the error interrupt. In DMA mode, the generation of an error interrupt (indicated in the IIR) disables DMA requests. Refer to Section 10.4.2.5.

If additional data is received before a receiver time-out interrupt is serviced, the interrupt is de-asserted.

IIR contains an interrupt-pending bit (nIP). When IIR[nIP] is clear, an interrupt is pending from the UART. When IIR[nIP] is set, no interrupt is pending, regardless of the state of the other bits in the IIR register. This condition can occur when the condition that causes an interrupt is removed before the interrupt is cleared.

This is a read-only register. Ignore reads from reserved bits.

Table 10-9. Interrupt Conditions (Sheet 1 of 2)

Priority Level	Interrupt Origin
1 (highest)	Receiver line status: one or more error bits was set.
2	Received data is available. In FIFO mode, the trigger threshold was reached. In non-FIFO mode, RBR has data.

Table 10-9. Interrupt Conditions (Sheet 2 of 2)

Priority Level	Interrupt Origin
2	Receiver time-out occurred. Occurs only in FIFO mode, when data is in the receive FIFO but no data has been sent for a set time period.
3	Transmitter requests data. In FIFO mode, the transmit FIFO is at least half empty. In non-FIFO mode, the THR has been transmitted.
4 (lowest)	Modem status: one or more modem input signals has changed state.

Table 10-10. IIR Bit Definitions

Physical Address 0xBASE_0008 — **IIR** — **UARTs**

User Settings

Bit	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																								FIFOES		EOC	ABL	TOD	IID		nIP
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	1

Bits	Access	Name	Description
31:8	—	—	reserved
7:6	R	FIFOES[1:0]	FIFO Mode Enable Status 0b00 = Non-FIFO mode is selected. 0b01 = reserved 0b10 = reserved 0b11 = FIFO mode is selected (TRFIFOE = 1).
5	R	EOC	DMA End of Descriptor Chain See Section 10.4.5. 0 = DMA has not signaled the end of its programmed descriptor chain. 1 = DMA has signaled the end of its programmed descriptor chain.
4	R	ABL	Auto-Baud Lock See Section 10.4.4. 0 = Auto-baud circuitry has not programmed Divisor Latch registers (DLR). 1 = Divisor Latch registers (DLR) programmed by auto-baud circuitry.
3	R	TOD	Time-Out Detected See Section 10.4.2.1.2. 0 = No time-out interrupt is pending. 1 = Time-out interrupt is pending. (FIFO mode only)
2:1	R	IID[1:0]	Interrupt Source Encoded 0b00 = Modem status (CTS, DSR, RI, DCD modem signals changed state). 0b01 = Transmit FIFO requests data. 0b10 = Received data available 0b11 = Receive error (overrun, parity, framing, break, FIFO error. See Table 10-19.
0	R	nIP	Interrupt Pending 0 = Interrupt is pending (active low). 1 = No interrupt is pending.

Table 10-11 shows the priority, type, and source of the Interrupt Identification register interrupts. It also gives the reset condition used to de-assert the interrupts. Bits[3:0] of the IIR register represent priority encoded interrupts; bits[7:4] do not.

Table 10-11. Interrupt Identification Register (IIR) Decode

Interrupt ID bits					Interrupt SET/RESET Function			
	3	2	1	0	Priority	Type	Source	RESET Control
nIP	0	0	0	1	—	None	No interrupt is pending.	—
IID[11]	0	1	1	0	Highest	Receiver Line Status	Overrun Error, Parity Error, Framing Error, Break Interrupt.	Reading the Line Status register.
IID[10]	0	1	0	0	Second Highest	Received Data Available	Non-FIFO mode: receive buffer is full. FIFO mode: trigger threshold was reached.	Non-FIFO mode: Reading the Receiver Buffer register. FIFO mode: Reading bytes until the receive FIFO drops below trigger threshold or setting.
TOD	1	1	0	0	Second Highest	Character Time-out indication	FIFO mode only: At least one character is left in the receive buffer indicating trailing bytes.	Reading the receive FIFO or setting FCR[RESETRF].
IID[01]	0	0	1	0	Third Highest	Transmit FIFO Data Request	Non-FIFO mode: Transmit Holding register Empty FIFO mode: transmit FIFO has half or less than half data.	Reading the IIR register (if the source of the interrupt) or writing into the Transmit Holding register. Reading the IIR register (if the source of the interrupt) or writing to the transmit FIFO.
IID[00]	0	0	0	0	Fourth Highest	Modem Status	Clear to Send, Data Set Ready, Ring Indicator, Received Line Signal Detect.	Reading the Modem Status register.
Non Prioritized Interrupts:								
ABL	4				None	Auto-Baud Lock Indication	Auto-baud circuitry has locked onto the baud rate.	Reading the IIR register
EOC	5				None	DMA End of Descriptor Chain	The DMA has reached the end of its programmed descriptor chain.	Reading the IIR register

10.5.6 FIFO Control Register (FCR)

FCR, described in Table 10-12, is a write-only register that is located at the same address as the IIR, which is a read-only register. FCR enables/disables the transmit/receive FIFOs, clears the transmit/receive FIFOs, and sets the receive FIFO trigger threshold.

Note: The trigger level must be equal to the DMA burst-length programmed in the DMA registers.

Interrupt Trigger-Level—When the number of bytes in the receive FIFO equals the interrupt trigger-level programmed into this field and the received-data-available interrupt is enabled (using IER), an interrupt is generated and appropriate bits are set in the IIR. The receive-DMA request is generated as well when trigger level is reached. The trigger level must be greater than or equal to the DMA burst size programmed in the DMA registers.

32-Bit Peripheral Bus—When clear, the UART ignores any information in the upper three bytes of the 32-bit bus. A full- or partial-word read or write to the UART with this bit cleared increments the FIFO counters by one byte only. If this bit is set, a full- or partial-word read or write increments the counter by the number of valid bytes within the word.

Trailing Bytes—When clear, the processor handles trailing bytes. When set, the DMA controller handles trailing bytes automatically. See Section 5.4.6, “How DMA Handles Trailing Bytes” on page 5-18 for more information.

Transmit Interrupt Level—Setting TIL causes transmitter interrupts and DMA requests to occur when the transmit FIFO is empty. Clearing TIL causes transmitter interrupts and DMA requests to occur when the transmit FIFO is half-empty.

Reset Transmit FIFO—When RESETTF is set, the transmit FIFO counter is reset to clear all the bytes in the FIFO. The LSR[TDRQ] bit is set, generating a transmitter-requests-data interrupt (IIR[IID]) if IER[TIE] is set. The Transmit Shift register is not reset; it completes the current transmission. Any DMA or transmit-FIFO-service-request interrupts are cleared.

Note: (1) RESETTF is automatically reset to 0 after the FIFO is cleared.

Reset Receive FIFO—When RESETRF is set, the receive FIFO counter is reset to clear all the bytes in the FIFO. LSR[DR] is cleared. All error bits in the FIFO and the LSR[FIFOE] bit are cleared. Any error bits (OE, PE, FE, or BI) that had been set in LSR remain set. The Receive Shift register is not cleared. Any DMA or receive-FIFO-service-request interrupts are cleared.

(2) RESETRF is automatically reset to 0 after the FIFO is cleared.

Transmit and Receive FIFO Enable—TRFIFOE enables and disables the transmit and receive FIFOs. When TRFIFOE is set, both FIFOs are enabled (FIFO mode). When TRFIFOE is clear, the FIFOs are both disabled (non-FIFO mode). Writing 0b0 to this bit clears all bytes in both FIFOs. When changing from FIFO mode to non-FIFO mode and vice versa, data is automatically cleared from the FIFOs. Any DMA or FIFO-service-request interrupts are cleared when TRFIFOE is clear.

(3) This bit must be 1 when other bits in this register are written, or else the other bits are not programmed.

This is a write-only register. Write 0b0 to reserved bits.

Table 10-12. FCR Bit Definitions (Sheet 1 of 2)

Physical Address 0xBASE_0008 — **FCR** — **UARTs**

	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
User Settings																																
Bit	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																								ITL		BUS	TRAIL	TIL	RESETTF	RESETRF	TRFIFOE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:6	W	ITL	Interrupt Trigger-Level (Threshold): When the number of bytes in the receive FIFO equals the interrupt trigger threshold programmed into this field and the received-data-available interrupt is enabled using the IER, an interrupt is generated and appropriate bits are set in the IIR. The receive DMA request is also generated when the trigger threshold is reached. 0b00 = 1 byte or more in FIFO causes interrupt (Not valid in DMA mode). 0b01 = 8 bytes or more in FIFO causes interrupt and DMA request. 0b10 = 16 bytes or more in FIFO causes interrupt and DMA request. 0b11 = 32 bytes or more in FIFO causes interrupt and DMA request.
5	W	BUS	32-Bit Peripheral Bus 0 = 8-bit peripheral bus 1 = 32-bit peripheral bus
4	W	TRAIL	Trailing Bytes 0 = Trailing bytes are removed by the processor. 1 = Trailing bytes are removed by the DMA controller.
3	W	TIL	Transmitter Interrupt Level 0 = Interrupt/DMA request when FIFO is half empty. 1 = Interrupt/DMA request when FIFO is empty.

Table 10-12. FCR Bit Definitions (Sheet 2 of 2)

Physical Address 0xBASE_0008																FCR								UARTs							
User Settings																															
Bit	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																								ITL		BUS	TRAIL	TIL	RESETTF	RESETRF	TRFIFOE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	W	RESETTF	Reset Transmit FIFO When RESETTF is set, all the bytes in the transmit FIFO are cleared. The TDRQ bit in the LSR is set and the IIR shows a transmitter requests data interrupt, if the TIE bit in the IER register is set. The transmit shift register is not cleared, and it completes the current transmission. 0 = Writing 0 has no effect. 1 = The transmit FIFO is cleared.
1	W	RESETRF	Reset Receive FIFO When RESETRF is set, all the bytes in the receive FIFO are cleared. The DR bit in the LSR is reset to 0. All the error bits in the FIFO and the FIFOE bit in the LSR are cleared. Any error bits, OE, PE, FE or BI, that had been set in LSR are still set. The receive shift register is not cleared. If the IIR had been set to Received Data Available, it is cleared. 0 = Writing 0 has no effect. 1 = The receive FIFO is cleared.
0	W	TRFIFOE	Transmit and Receive FIFO Enable TRFIFOE enables/disables the transmit and receive FIFOs. When TRFIFOE is set, both FIFOs are enabled (FIFO mode). When TRFIFOE is clear, the FIFOs are both disabled (non-FIFO mode). Writing 0b0 to this bit clears all bytes in both FIFOs. When changing from FIFO mode to non-FIFO mode and vice versa, data is automatically cleared from the FIFOs. This bit must be set when other bits in this register are written or the other bits are not programmed. 0 = FIFOs are disabled. 1 = FIFOs are enabled.

10.5.7 Receive FIFO Occupancy Register (FOR)

The Receive FIFO Occupancy register (FOR), described in Table 10-13, shows the number of bytes currently remaining the receive FIFO. FOR determines the number of trailing bytes to remove in the case when the DMA reaches the end of its descriptor chain or when FCR[TRAIL] is clear, which indicates that the processor will remove trailing bytes as opposed to the DMA (see Section 10.4.2.6). FOR is incremented once for each byte of data written to the receive FIFO and decremented once for each byte read.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 10-13. FOR Bit Definitions

Physical Address 0xBASE_0024 FOR UARTs

	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
User Settings																																
Bit	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																										Byte Count					
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
31:6	—	—	reserved
5:0	R/W	Byte Count	Number of bytes (0-63) remaining in the receive FIFO

10.5.8 Auto-Baud Control Register (ABR)

The Auto-Baud Control register (ABR), described in Table 10-14, controls the functionality and options for auto-baud-rate detection within the UART. Through this register, software can enable/disable the auto-baud-lock interrupt, direct either the processor or the UART to program the final baud rate in the Divisor Latch registers, and choose between two methods to calculate the final baud rate.

The auto-baud circuitry counts the number of clocks in the start bit and writes this count into the Auto-Baud Count register (ACR). It then interrupts the processor if ABR[ABLIE] is set. It also programs the Divisor Latch registers (DLL and DLH) automatically if ABR[ABUP] is set.

See Section 10.4.4 for more information on auto-baud rate.

Note: Auto-baud-rate detection is not supported in IrDA slow-infrared mode.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 10-14. ABR Bit Definitions

Physical Address 0xBASE_0028 — ABR — UARTs

	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
User Settings																																
Bit	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																												ABT	ABUP	ABLIE	ABE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0

Bits	Access	Name	Description
31:4	—	—	reserved
3	R/W	ABT	Auto-Baud Rate Calculation 0 = Use a formula to calculate baud rates, allowing all possible baud rates to be chosen by UART as shown in Section 10.4.2.1.2. 1 = Use a table to calculate baud rates, which limits UART to choosing common baud rates.
2	R/W	ABUP	Auto-Baud Programmer 0 = Processor Programs Divisor Latch registers. 1 = UART Programs Divisor Latch registers.
1	R/W	ABLIE	Auto-Baud Lock Interrupt 0 = Auto-baud-lock interrupt disabled (Source IIR[ABL]). 1 = Auto-baud-lock interrupt enabled (Source IIR[ABL]).
0	R/W	ABE	Auto-Baud Enable 0 = Auto-baud disabled. 1 = Auto-baud enabled.

10.5.9 Auto-Baud Count Register (ACR)

Auto-Baud Count register (ACR), defined in Table 10-15, stores the number of 14.857-MHz clock cycles within a start-bit pulse. This value is then used by the processor or the UART to calculate the baud rate. If auto-baud mode (ABR[ABE]) and auto-baud interrupts (ABR[ABLIE]) are enabled, the UART interrupts the processor with the auto-baud-lock interrupt (IIR[ABL]) after it has written the count value into ACR. The value is written regardless of the state of the auto-baud UART program bit, (ABR[ABUP]).

This is a read-only register. Ignore reads from reserved bits.

Table 10-15. ACR Bit Definitions

Physical Address 0xBASE_002C — ACR — UARTs

User Settings

Bit	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																Count Value															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R	Count Value	Number of 14.857-MHz clock cycles within a start-bit pulse.

10.5.10 Line Control Register (LCR)

The Line Control register (LCR), defined in Table 10-16, specifies the format for the asynchronous data-communication exchange. The serial-data format consists of a start bit, five to eight data bits, an optional parity bit, and one, one and a half, or two stop bits. LCR has bits that allow access to the Divisor Latch registers and bits that can cause a break condition.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 10-16. LCR Bit Definitions (Sheet 1 of 2)

Physical Address 0xBASE_000C — LCR — UARTs

User Settings

Bit	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																								DLAB	SB	STKYP	EPS	PEN	STB	WLS	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7	R/W	DLAB	Divisor Latch Access Must be set to access the Divisor Latch registers of the baud-rate generator during a read or write operation. Must be clear to access the Receive Buffer, the Transmit Holding register, or the IER. 0 = Access Transmit Holding register (THR), Receive Buffer register (RBR), and IER. 1 = Access Divisor Latch registers (DLL and DLH).
6	R/W	SB	Set Break Causes a break condition to be transmitted to the receiving UART. Acts only on the TXD pin and has no effect on the transmit logic. In FIFO mode, wait until the transmitter is idle (LSR[TEMT] = 1] to set and clear SB. 0 = No effect on TXD output. 1 = Forces TXD output to 0 (space).

Table 10-16. LCR Bit Definitions (Sheet 2 of 2)

Physical Address 0xBASE_000C																LCR								UARTs							
User Settings																															
Bit	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																								DLAB	SB	STKYP	EPS	PEN	STB	WLS	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
5	R/W	STKYP	Sticky Parity Forces the bit value at the parity bit location to be the opposite of the EPS bit rather than the parity value. This stops parity generation. If PEN = 0, STKYP is ignored. 0 = No effect on parity bit. 1 = Forces parity bit to be opposite of EPS bit value.
4	R/W	EPS	Even Parity Select If PEN = 0, EPS is ignored. 0 = Sends or checks for odd parity. 1 = Sends or checks for even parity.
3	R/W	PEN	Parity Enable Enables a parity bit to be generated on transmission or checked on reception. 0 = No parity 1 = Parity
2	R/W	STB	Stop Bits Specifies the number of stop bits transmitted and received in each character. When receiving, the receiver checks only the first stop bit. 0 = 1 stop bit 1 = 2 stop bits, except for 5-bit character then 1-1/2 bits
1:0	R/W	WLS[1:0]	Word Length Select Specifies the number of data bits in each transmitted or received character. 0b00 = 5-bit character 0b01 = 6-bit character 0b10 = 7-bit character 0b11 = 8-bit character

10.5.11 Line Status Register (LSR)

The Line Status register (LSR), defined in Table 10-17, provides data-transfer status information to the processor. In non-FIFO mode, LSR[4:2] show the error status of the character that has just been received. In FIFO mode, LSR[4:2] show the status bits of the character that is currently at the front of the FIFO.

LSR[4:1] produce a receiver-line-status interrupt when the corresponding conditions are detected and the interrupt is enabled. In FIFO mode, the receiver-line-status interrupt occurs only when the erroneous character reaches the front of the FIFO. If the erroneous character is not at the front of the FIFO, a line-status interrupt is generated after the other characters are read, and the erroneous character becomes the character at the front of the FIFO.

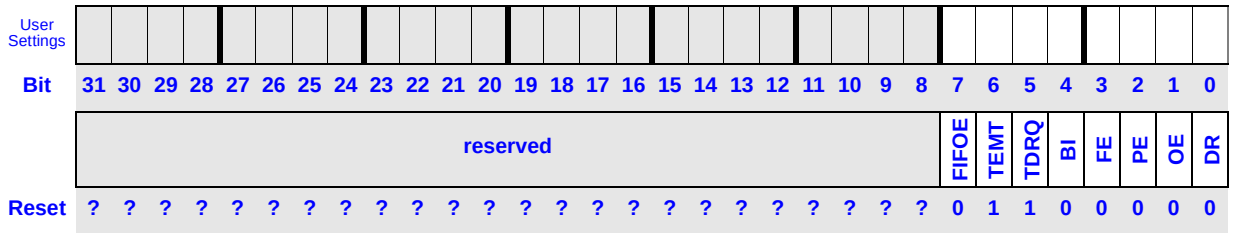
LSR must be read before the erroneous character is read. LSR[4:1] remain set until software reads LSR.

See Section 10.4.2.3 for details on using the DMA controller to receive data.

This is a read-only register. Ignore reads from reserved bits.

Table 10-17. LSR Bit Definitions (Sheet 1 of 3)

Physical Address 0xBASE_0014 — **LSR** — **UARTs**



Bits	Access	Name	Description
31:8	—	—	reserved
7	R	FIFOE	FIFO Error Status In non-FIFO mode, this bit is clear. In FIFO mode, FIFOE is set when there is at least one parity error, framing error, or break indication for any of the characters in the FIFO. A processor read of the LSR does not reset this bit. FIFOE is reset when all erroneous characters have been read from the FIFO. If DMA requests are enabled (IER bit 7 set) and FIFOE is set, the error interrupt is generated, and no receive DMA request is generated even when the receive FIFO reaches the trigger threshold. Once the errors have been cleared by reading the FIFO, DMA requests are re-enabled automatically. If DMA requests are not enabled (IER bit7 clear), FIFOE set does not generate an error interrupt. 0 = No FIFO or no errors in receive FIFO. 1 = At least one character in receive FIFO has errors.
6	R	TEMT	Transmitter Empty Set when the Transmit Holding register and the transmit shift register are both empty. It is cleared when either the Transmit Holding register or the transmit shift register contains a data character. In FIFO mode, TEMT is set when the transmit FIFO and the transmit shift register are both empty. 0 = There is data in the transmit shift register, the Transmit Holding register, or the FIFO. 1 = All the data in the transmitter has been shifted out.

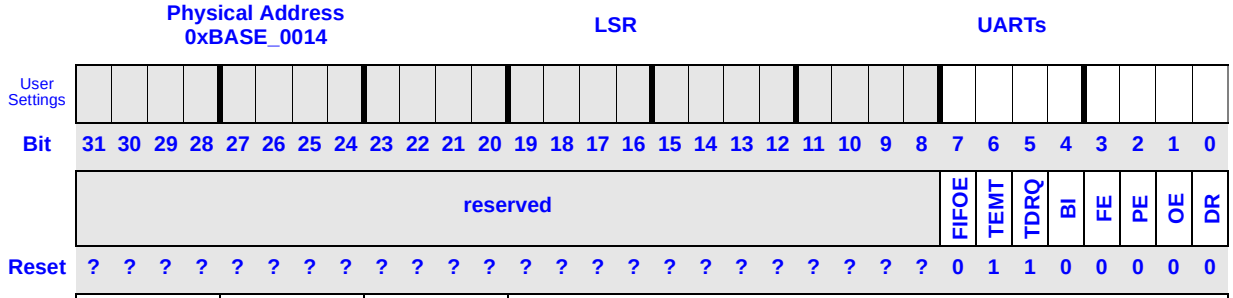
Table 10-17. LSR Bit Definitions (Sheet 2 of 3)

Physical Address 0xBASE_0014 — LSR — UARTs

	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
User Settings																																
Bit	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																								FIFOE	TEMT	TDRQ	BI	FE	PE	OE	DR
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	1	0	0	0	0	0

Bits	Access	Name	Description
5	R	TDRQ	Transmit Data Request Indicates that the UART is ready to accept a new character for transmission. In addition, this bit causes the UART to issue an interrupt to the processor when the transmit data request interrupt enable is set and generates the DMA request to the DMA controller if DMA requests and FIFO mode are enabled. The TDRQ bit is set when a character is transferred from the Transmit Holding register into the transmit shift register. The bit is cleared with the loading of the Transmit Holding register. In FIFO mode, TDRQ is set when half of the characters in the FIFO have been loaded into the shift register or the RESETTF bit in FCR has been set. It is cleared when the FIFO has more than half data. If more than 64 characters are loaded into the FIFO, the excess characters are lost. 0 = There is data in the holding register or FIFO waiting to be shifted out. 1 = The transmit FIFO has half or less than half data.
4	R	BI	Break Interrupt BI is set when the received data input is held low for longer than a full-word transmission time (the total time of start bit + data bits + parity bit + stop bits). BI is cleared when the processor reads the LSR. In FIFO mode, only one character equal to 0x00, is loaded into the FIFO regardless of the length of the break condition. BI shows the break condition for the character at the front of the FIFO, not the most recently received character. 0 = No break signal has been received. 1 = Break signal received.
3	R	FE	Framing Error Indicates that the received character did not have a valid stop bit. FE is set when the bit following the last data bit or parity bit is detected to be 0. If the LCR had been set for two stop bits, the receiver does not check for a valid second stop bit. FE is cleared when the processor reads the LSR. The UART re-synchronizes after a framing error. To do this, it assumes that the framing error was due to the next start bit, so it samples this start bit twice and then reads in the data. In FIFO mode, FE shows a framing error for the character at the front of the FIFO, not for the most recently received character. 0 = No framing error. 1 = Invalid stop bit has been detected.

Table 10-17. LSR Bit Definitions (Sheet 3 of 3)



Bits	Access	Name	Description
2	R	PE	Parity Error Indicates that the received data character does not have the correct even or odd parity, as selected by the even parity select bit. PE is set upon detection of a parity error and is cleared when the processor reads the LSR. In FIFO mode, PE shows a parity error for the character at the front of the FIFO, not the most recently received character. 0 = No parity error. 1 = Parity error has occurred.
1	R	OE	Overrun Error In non-FIFO mode, indicates that data in the Receive Buffer register was not read by the processor before the next character was received. The new character is lost. In FIFO mode, OE indicates that all 64 bytes of the FIFO are full and the most recently received byte has been discarded. OE is set upon detection of an overrun condition and cleared when the processor reads the LSR. 0 = No data has been lost. 1 = Receive data has been lost.
0	R	DR	Data Ready Set when a complete incoming character has been received and transferred into the Receive Buffer register or the FIFO. In non-FIFO mode, DR is cleared when the receive buffer is read. In FIFO mode, DR is cleared if the FIFO is empty (last character has been read from RBR) or the FIFO is reset with FCR[RESETRF]. 0 = No data has been received. 1 = Data is available in RBR or the FIFO.

10.5.12 Modem Control Register (MCR)

MCR, defined in Table 10-18, uses the modem-control pins nRTS and nDTR to control the interface with a modem or data set. MCR also controls the loopback mode, which must be enabled before the UART is enabled. The differences between UARTs specific to this register are described in Table 10-25.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 10-18. MCR Bit Definitions (Sheet 1 of 2)

Physical Address 0xBASE_0010 — **MCR** — **UARTs**

	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
User Settings																																
Bit	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																										AFE	LOOP	OUT2	OUT1	RTS	DTR
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
5	R/W	AFE	Auto-Flow Control Enable 0 = Auto-RTS and auto-CTS are disabled. 1 = Auto-CTS is enabled. If MCR[RTS] is also set, both auto-CTS and auto-RTS are enabled.
4	R/W	LOOP	Loopback Mode Provides a local loopback feature for diagnostic testing of the UART. When LOOP is set, the following occurs: The transmitter serial output is set to a logic 1 state. The receiver serial input is disconnected from the pin. The output of the transmit shift register is “looped back” into the receive shift register input. The four modem control inputs (nCTS, nDSR, nDCD, and nRI) are disconnected from the pins and the modem control output pins (nRTS and nDTR) are forced to their inactive state. Coming out of the loopback mode may result in unpredictable activation of the delta bits (bits 3:0) in the Modem Status register. Read MSR once to clear the MSR delta bits. Loopback mode must be configured before the UART is enabled. The lower four bits of the MCR are connected to the upper four Modem Status register bits: • DTR = 1 forces DSR to a 1 • RTS = 1 forces CTS to a 1 • OUT1 = 1 forces RI to a 1 • OUT2 = 1 forces DCD to a 1 In loopback mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit and receive data paths of the UART. The transmit, receive, and modem-control interrupts are operational, except that the modem control interrupts are activated by MCR bits, not by the modem-control pins. A break signal can also be transferred from the transmitter section to the receiver section in loopback mode. 0 = Normal UART operation 1 = Loopback-mode UART operation
3	R/W	OUT2	OUT2 Signal Control OUT2 connects the UART’s interrupt output to the interrupt controller unit. When LOOP is clear: 0 = UART interrupt is disabled. 1 = UART interrupt is enabled. When LOOP is set, interrupts always go to the processor: 0 = MSR[DCD] forced to 0b0. 1 = MSR[DCD] forced to 0b1.

Table 10-18. MCR Bit Definitions (Sheet 2 of 2)

Physical Address 0xBASE_0010																MCR								UARTs							
User Settings																															
Bit	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																										AFE	LOOP	OUT2	OUT1	RTS	DTR
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
2	R/W	OUT1	Test Bit Used only in loopback mode. It is ignored otherwise. 0 = Force MSR[RI] to 0b0. 1 = Force MSR[RI] to 0b1.
1	R/W	RTS	Request to Send 0 = Non-auto-flow mode. nRTS pin is 1. Auto-RTS is disabled. Auto-flow works only with auto-CTS. 1 = Auto-flow mode. nRTS pin is 0. Auto-RTS is enabled. Auto-flow works with both auto-CTS and auto-RTS.
0	R/W	DTR	Data Terminal Ready 0 = nDTR pin is 1. 1 = nDTR pin is 0.

10.5.13 Modem Status Register (MSR)

The Modem Status register (MSR) provides the current state of the control lines from the modem or data set (or a peripheral device emulating a modem) to the processor. In addition to this current state information, four MSR bits provide change information. MSR[3:0] are set when a control input from the modem changes state. They are cleared when the processor reads MSR. Differences between UARTs specific to this register are listed in Table 10-25.

The status of the modem control lines does not affect the FIFOs. IER[MIE] must be set to use these lines for flow control. When an interrupt occurs on one of the flow-control pins, the interrupt service routine must disable the UART. The UART continues transmission/reception of the current character and then stops. The contents of the FIFOs are preserved. If the UART is re-enabled, transmission continues where it stopped.

Note: When bit 0, 1, 2, or 3 is set, a modem-status interrupt is generated if IER[MIE] is set.

This is a read-only register. Ignore reads from reserved bits.

Table 10-19. MSR Bit Definitions

Physical Address 0xBASE_0018 — MSR — UARTs

Bit	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
User Settings																																
	reserved																								DCD	RI	DSR	CTS	DDCD	TERI	DDSR	DCTS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1	1	1	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7	R	DCD	Data Carrier Detect Complement of the Data Carrier Detect (nDCD) input. Equivalent to MCR[OUT2] if MCR[LOOP] is set. 0 = nDCD pin is 1. 1 = nDCD pin is 0.
6	R	RI	Ring Indicator Complement of the Ring Indicator (nRI) input. Equivalent to MCR[OUT1] if MCR[LOOP] is set. 0 = nRI pin is 1. 1 = nRI pin is 0.
5	R	DSR	Data Set Ready Complement of the Data Set Ready (nDSR) input. Equivalent to MCR[DTR] if MCR[LOOP] is set. 0 = nDSR pin is 1. 1 = nDSR pin is 0.
4	R	CTS	Clear To Send Complement of the Clear to Send (nCTS) input. Equivalent to MCR[RTS] if MCR[LOOP] is set. 0 = nCTS pin is 1. 1 = nCTS pin is 0.
3	R	DDCD	Delta Data Carrier Detect 0 = No change in nDCD pin since last read of MSR. 1 = nDCD pin has changed state.
2	R	TERI	Trailing Edge Ring Indicator 0 = nRI pin has not changed from 0 to 1 since last read of MSR. 1 = nRI pin has changed state.
1	R	DDSR	Delta Data Set Ready 0 = No change in nDSR pin since last read of MSR. 1 = nDSR pin has changed state.
0	R	DCTS	Delta Clear To Send 0 = No change in nCTS pin since last read of MSR. 1 = nCTS pin has changed state.

10.5.14 Scratchpad Register (SCR)

The Scratchpad register (SCR), defined in Table 10-20, has no effect on the UART. It is intended as a scratchpad register for use by programmers. It is included for 16550A compatibility.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 10-20. SCR Bit Definitions

Physical Address 0xBASE_001C																SCR								UARTs							
User Settings																															
Bit	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																								Scratcpad							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	Scratchpad	No effect on UART functions

10.5.15 Infrared Selection Register (ISR)

Each UART can manage an IrDA module associated with it. The Infrared Selection register (ISR) controls the IrDA functions (see Section 10.4.6).

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 10-21. ISR Bit Definitions (Sheet 1 of 2)

Physical Address 0x5800_0020																ISR								UARTs							
User Settings																															
Bit	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																											RXPL	TXPL	XMODE	RCVEIR	XMITIR
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0

Bits	Access	Name	Description
31:5	—	—	reserved
4	R/W	RXPL	Receive Data Polarity 0 = Slow-infrared (SIR) interface decoder takes positive pulses as zeros. 1 = SIR decoder takes negative pulses as zeros.
3	R/W	TXPL	Transmit Data Polarity 0 = SIR encoder generates a positive pulse for a data bit of zero. 1 = SIR encoder generates a negative pulse for a data bit of zero.

Table 10-21. ISR Bit Definitions (Sheet 2 of 2)

Physical Address 0x5800_0020 — **ISR** — **UARTs**

	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
User Settings																																
Bit	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																											RXPL	TXPL	XMODE	RCVEIR	XMITIR
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0

Bits	Access	Name	Description
2	R/W	XMODE	Transmit Pulse Width Select When XMODE is clear, the UART 16x clock clocks the IrDA transmit and receive logic. When XMODE is set, receive decoder operation does not change, and the transmit encoder generates 1.6 µs pulses (that are 3/16 of a bit-time at 115.2 kbps) instead of pulses 3/16 of a bit-time wide. Intel recommends setting XMODE. 0 = Transmit pulse width is 3/16 of a bit-time wide. 1 = Transmit pulse width is 1.6 µs.
1	R/W	RCVEIR	Receiver SIR Enable When RCVEIR is set, the signal from the RXD pin is processed by the IrDA decoder before it is fed to the UART. If RCVEIR is clear, then all clocking to the IrDA decoder is blocked and the RXD pin is fed directly to the UART. 0 = Receiver is in UART mode. 1 = Receiver is in slow infrared mode.
0	R/W	XMITIR	Transmitter SIR Enable When XMITIR is set, the normal TXD output from the UART is processed by the IrDA encoder before it is fed to the device pin. If XMITIR is clear, all clocking to the IrDA encoder is blocked and the UART's TXD signal is connected directly to the device pin. When Transmitter SIR Enable is set, the TXD output pin, which is in a normally high default state, switches to a normally low default state. This can cause a false start bit unless the infrared LED is disabled before XMITIR is set. 0 = Transmitter is in UART mode. 1 = Transmitter is in slow infrared mode.

10.6 Register Summary

Table 10-22, Table 10-23, and Table 10-24 list the register addresses for the FFUART, BTUART, and STUART.

Table 10-25 summarizes the differences between the Bluetooth and standard UARTs.

Table 10-22. FFUART Register Summary

Address	DLAB Bit Value	Name	Description	Page
0x4010_0000	0	FFRBR	Receive Buffer register	10-13
0x4010_0000	0	FFTHR	Transmit Holding register	10-14
0x4010_0000	1	FFDLL	Divisor Latch register, low byte	10-14
0x4010_0004	0	FFIER	Interrupt Enable register	10-15
0x4010_0004	1	FFDLH	Divisor Latch register, high byte	10-14
0x4010_0008	X	FFIIR	Interrupt ID register	10-17
0x4010_0008	X	FFFCR	FIFO Control register	10-19
0x4010_000C	X	FFLCR	Line Control register	10-25
0x4010_0010	X	FFMCR	Modem Control register	10-29
0x4010_0014	X	FFLSR	Line Status register	10-26
0x4010_0018	X	FFMSR	Modem Status register	10-31
0x4010_001C	X	FFSPR	Scratch Pad register	10-33
0x4010_0020	X	FFISR	Infrared Select register	10-33
0x4010_0024	X	FFFOR	Receive FIFO Occupancy register	10-22
0x4010_0028	X	FFABR	Auto-baud Control register	10-23
0x4010_002C	X	FFACR	Auto-baud Count register	10-24
0x4010_0030– 0x401F_FFFC	—	—	reserved	—

Table 10-23. BTUART Register Summary (Sheet 1 of 2)

Address	DLAB Bit Value	Name	Description	Page
0x4020_0000	0	BTRBR	Receive Buffer register	10-13
0x4020_0000	0	BTTHR	Transmit Holding register	10-14
0x4020_0000	1	BTDLL	Divisor Latch register, low byte	10-14
0x4020_0004	0	BTIER	Interrupt Enable register	10-15
0x4020_0004	1	BTDLH	Divisor Latch register, high byte	10-14
0x4020_0008	X	BTIIR	Interrupt ID register	10-17
0x4020_0008	X	BTFCR	FIFO Control register	10-19
0x4020_000C	X	BTLCR	Line Control register	10-25
0x4020_0010	X	BTMCR	Modem Control register	10-29
0x4020_0014	X	BTLSR	Line Status register	10-26

Table 10-23. BTUART Register Summary (Sheet 2 of 2)

Address	DLAB Bit Value	Name	Description	Page
0x4020_0018	X	BTMSR	Modem Status register	10-31
0x4020_001C	X	BTSPR	Scratch Pad register	10-33
0x4020_0020	X	BTISR	Infrared Select register	10-33
0x4020_0024	X	BTFOR	Receive FIFO Occupancy register	10-22
0x4020_0028	X	BTABR	Auto-Baud Control register	10-23
0x4020_002C	X	BTACR	Auto-Baud Count register	10-24
0x4020_0030– 0x402F_FFFC	—	—	reserved	—

Table 10-24. STUART Register Summary

Address	DLAB Bit Value	Name	Description	Page
0x4070_0000	0	STRBR	Receive Buffer register	10-13
0x4070_0000	0	STTHR	Transmit Holding register	10-14
0x4070_0000	1	STDLL	Divisor Latch register, low byte	10-14
0x4070_0004	0	STIER	Interrupt Enable register	10-15
0x4070_0004	1	STDLH	Divisor Latch register, high byte	10-14
0x4070_0008	X	STIIR	Interrupt ID register	10-17
0x4070_0008	X	STFCR	FIFO Control register	10-19
0x4070_000C	X	STLCR	Line Control register	10-25
0x4070_0010	X	STMCR	Modem Control register	10-29
0x4070_0014	X	STLSR	Line Status register	10-26
0x4070_0018	X	STMSR	Modem Status register	10-31
0x4070_001C	X	STSPR	Scratch Pad register	10-33
0x4070_0020	X	STISR	Infrared Select register	10-33
0x4070_0024	X	STFOR	Receive FIFO Occupancy register	10-22
0x4070_0028	X	STABR	Auto-Baud Control register	10-23
0x4070_002C	X	STACR	Auto-Baud Count register	10-24
0x4070_0030– 0x407F_FFFC	—	—	reserved	—

The default descriptions for BTMCR, BTMSR, STMCR, and STMSR are modified as shown in Table 10-25.

Table 10-25. Flow-Control Registers in BTUART and STUART

	Bits 7:6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BTMCR	reserved	AFE	LOOP	OUT2	reserved	RTS	reserved
BTMSR	reserved	reserved	CTS	reserved	reserved	reserved	DCTS
STMCR	reserved	reserved	LOOP	OUT2	reserved	reserved	reserved
STMSR	reserved	reserved	reserved	reserved	reserved	reserved	reserved

Fast Infrared Communications Port 11

This chapter describes the fast infrared communications port included in the PXA27x processor.

11.1 Overview

The fast infrared communications port (FICP) operates at half-duplex and provides direct connection to commercially available Infrared Data Association (IrDA) compliant LED transceivers. The fast infrared communications port is based on the 4-Mbps IrDA standard[1] and uses four-position pulse modulation (4PPM) and a specialized serial packet protocol developed for IrDA transmission. To support the standard, the fast infrared communications port has:

- A bit encoder/decoder
- A serial-to-parallel data engine
- A transmit FIFO 64 entries deep and 8 bits wide
- A receive FIFO 64 entries deep and 11 bits wide

11.2 Signal Descriptions

The fast infrared communications port signals are ICP_RXD and ICP_TXD. Table 11-1 describes each signal's function. Most IrDA transceivers also have enable and speed pins. Use GPIOs to enable the transceiver and select the speed. See Section 24, "General-Purpose I/O Controller" for more information.

Table 11-1. Fast Infrared Communications Port I/O Signal Descriptions

Signal Name	Type	Description
ICP_RXD	Input	Receive pin for fast infrared port
ICP_TXD	Output	Transmit pin for fast infrared port

11.3 Operation

The fast infrared communications port is disabled and does not have control of the port's pins after a reset. Before software enables the FICP for high-speed operation, it must set the control registers to reflect the desired operating mode. After the control registers are set, software can either preload the FICP's transmit FIFO with up to 64 bytes, or leave the FIFO empty and use the DMA to service it after the FICP is enabled. Once the FICP is enabled, transmit/receive data can be sent on the transmit and receive pins.

To support a variety of IrDA transceivers, both the transmit and receive data pins can be individually configured to communicate using normal or active-low data.

1. See *Infrared Data Association Serial Infrared Physical Layer Specification Version 1.3*, October 15, 1998, available from http://www.irda.org

The transmit/receive data is modulated according to the 4PPM IrDA standard and converted to serial or parallel data. The modulation technique and the frame format are discussed in the sections that follow.

11.3.1 4PPM Modulation

Four-position pulse modulation (4PPM) transmits data at the high-speed rate, 4.0 Mbps. Data bits are encoded two at a time by placing a single 125-ns light pulse in one of four timeslots. The four timeslots are collectively called a *chip*. Bytes are encoded one at a time. They are divided into four individual two-bit pairings called *nibbles*. The least significant nibble is transmitted first. Figure 11-1 shows the 4PPM encoding for the possible two-bit combinations, and Figure 11-2 shows an example of 4PPM modulation for the byte 0b10110001, which is constructed with four chips. Bits within each nibble are not reordered, but nibble 0 (least significant) is transmitted first, and nibble 3 (most significant) is transmitted last.

Figure 11-1. 4PPM Modulation Encodings

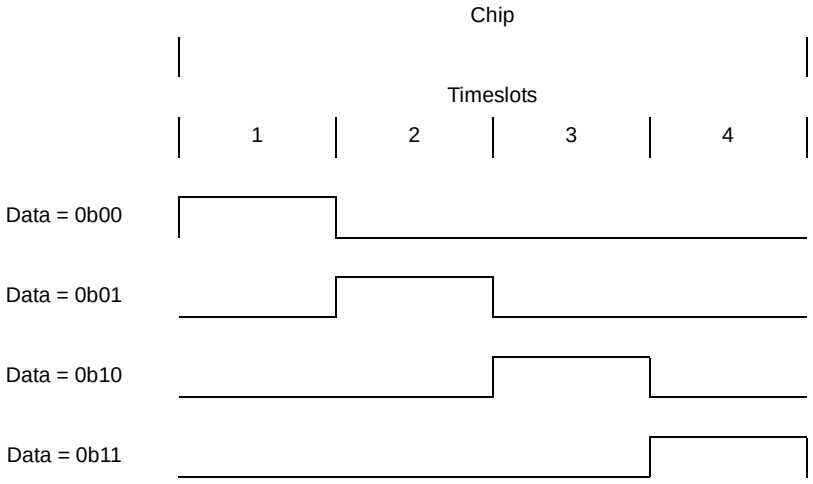
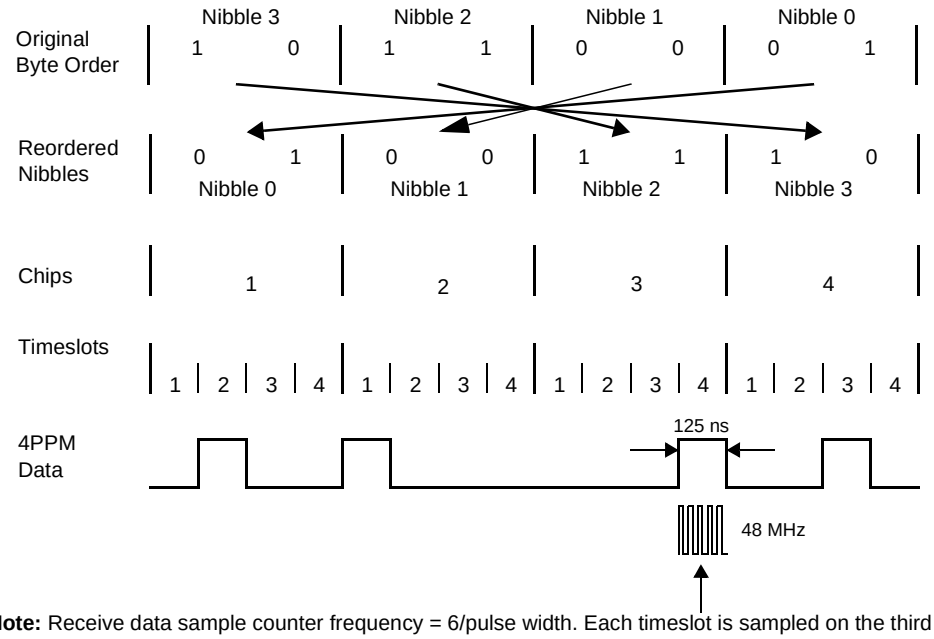


Figure 11-2. 4PPM Modulation Example



Note: Receive data sample counter frequency = 6/pulse width. Each timeslot is sampled on the third clock.

11.3.2 Frame Format

The frame format used with 4-Mbps transmission is shown in Figure 11-3.

Figure 11-3. Frame Format for IrDA Transmission (4.0 Mbps)

64 Chips	8 Chips	4 Chips (8 bits)	4 Chips (8 bits)	8180 Chips max (2045 bytes)	16 Chips (32 bits)	8 Chips
Preamble	Start Flag	Address (optional)	Control (optional)	Data	CRC-32	Stop Flag

Preamble	| 1000 | 0000 | 1010 | 1000 |... repeated at least 16 times
Start flag	| 0000 | 1100 | 0000 | 1100 | 0110 | 0000 | 0110 | 0000 |
Stop Flag	| 0000 |1100 | 0000 | 1100 | 0000 | 0110 | 0000 | 0110 |

The preamble, start, and stop flags are a mixture of chips that contain 0, 1, or 2 pulses in their timeslots. Chips with 0 and 2 pulses are used to construct flags because the chips represent invalid data bit pairings.

- Preamble contains 16 repeated transmissions of the chips: 1000 0000 1010 1000
- Start flag contains 1 transmission of 8 chips: 0000 1100 0000 1100 0110 0000 0110 0000
- Stop flag contains 1 transmission of 8 chips: 0000 1100 0000 1100 0000 0110 0000 0110

The address, control, data, and CRC-32 use the standard 4PPM chip encoding to represent two bits per chip.

11.3.3 Address Field

A transmitter uses the 8-bit address field to target a receiver when multiple stations are connected to the same set of serial lines. The address allows up to 255 stations to be uniquely addressed (0x00–0xFE). The broadcast address 0xFF sends messages to all of the connected stations.

For reception, FICP Control register 1 (ICCR1) programs a unique receive address. The AME bit in the FICP Control register 0 (ICCR0) determines the address match function. The addresses of the received frames are stored in the receive FIFO with normal data.

11.3.4 Control Field

The control field is an optional 8-bit field that is defined by software. The fast infrared communications port does not provide hardware decode support for the control byte. It treats all bytes between the address and the CRC as data.

11.3.5 Data Field

The data field can have a length from 0 to 2045 bytes. Application requirements and the target system's transmission characteristics affect the data field's length. Software must determine the length of the data to maximize the amount that can be transmitted in each frame, while allowing the CRC to detect all errors during transmission. The serial port does not contain hardware that restricts the maximum amount of data that can be transmitted or received. If a data field that is not a multiple of eight bits is received, an abort is signaled.

11.3.6 CRC Field

The fast infrared communications port uses a 32-bit cyclic redundancy check (CRC) to detect bit errors that occur during transmission. The CRC is generated from the address, control, and data fields and is included in each frame. Transmit and receive logic have separate CRC generators. The CRC computation logic is set to all ones before each frame is transmitted or received, and the result is inverted before it is used for comparison or transmission. The transmitter calculates a CRC as data is transmitted and places the inverse of the resulting 32-bit value at the end of each frame until the stop flag is transmitted. The receiver also calculates a CRC and inverts it for each data frame that it receives. The receiver compares the calculated CRC to the expected CRC value at the end of each frame.

If the calculated value does not match the expected value, the CRC error bit that corresponds to the last data byte received is set. When this byte reaches the trigger threshold range, an interrupt is generated.

Note: Unlike the address, control, and data fields, the 32-bit inverted CRC value is transmitted and received with the most significant nibble first.

The cyclic redundancy checker uses the 32-term polynomial:

$$CRC(x) = (x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1)$$

11.3.7 Baud Rate Generation

The baud rate is derived by dividing a fixed 48-MHz clock by six. Using a digital PLL, the 8-MHz baud (or timeslot) clock for the receive logic is synchronized with the 4PPM data stream each time a transition is detected on the receive data line. To encode a 4-Mbps data stream, the required chip frequency is 2.0 MHz, with four timeslots per chip at a frequency of 8.0 MHz. Receive data is sampled halfway through each timeslot period by counting three of the six 48-MHz clock periods that make up each timeslot (see Figure 11-2). The chips are synchronized during the reception preamble. The pattern of four chips repeated 16 times identifies the first timeslot (or the beginning of a chip) and resets the two-bit timeslot counter logic.

11.3.8 Receive Operation

The IrDA standard specifies that all transmission occurs at half-duplex. This restriction forces software to enable one direction at a given time. Either the transmit or receive logic can be enabled, but not both. The FICP hardware does not impose such a restriction. The software can enable both the transmitter and receiver. This feature is used with the FICP's loopback mode, which internally connects the output of the transmit serial shifter to the input of the receive serial shifter.

After the fast infrared communications port is enabled, the receiver logic begins and selects an arbitrary chip boundary, uses a serial shifter to receive four incoming 4PPM chips from the receive data pin, and latches and decodes the chips one at a time. If the chips do not have the correct preamble, the timeslot counter's clock skips one 8-MHz period and effectively delays the timeslot count by one. This process is called *hunt mode* and is repeated until the chips have the correct preamble, which indicates that the time-slot counter is synchronized. The preamble from the transmitter can be repeated as few as 16 times or can be continuously repeated to indicate an idle transmit line.

After 16 preambles are transmitted, the start flag is received. The start flag is eight chips long. If any portion of the start flag does not match the encoding, the receive logic signals a framing error, and the receive logic returns to hunt mode.

When the correct start flag is recognized, each following group of four chips is decoded into a data byte and placed in a five-byte temporary FIFO that prevents the CRC from being placed in the receive FIFO. When the temporary FIFO is full, data values are transferred to the receive FIFO one at a time. A frame's first data byte is the address. If receiver address matching is enabled, the received address is compared to the address in the Address Match Value field in ICCR1. If the values match or the incoming address contains all ones, all following data bytes, including the address byte, are stored in the receive FIFO. If the values do not match, the receiver logic does not store any data in the receive FIFO, ignores the remainder of the frame, and searches for the next preamble. If receiver address matching is not enabled, the frame's first data byte is stored in the FIFO as normal data. The frame's second data byte can contain an optional control field and must be decoded in software.

The IrDA standard limits frames to any amount of data up to a 2047 bytes (including the address and control bytes). The FICP does not limit frame size. Software must ensure that each incoming frame does not exceed 2047 bytes.

When the receive FIFO reaches its trigger threshold, an interrupt (if enabled) and DMA transfer request (if no errors are detected in the data) are signaled. If the data are not removed quickly enough to prevent the FIFO from completely filling, the receive logic attempts to place additional data into the full FIFO, and an overrun error is signaled. When the FIFO is full, all subsequent data bytes received are lost, and all FIFO contents remain intact.

If the data field contains any invalid chips (such as 0011, 1010, 1110, or 0000), the frame aborts, ICSR0[EIF] is set, and the oldest byte in the temporary FIFO is moved to the receive FIFO, the remaining temporary FIFO entries are discarded, the end-of-frame (EOF) tag is set in the FIFO entry that holds the last valid byte of data, and the receiver logic searches for the preamble.

The receive logic continuously searches for the eight-chip stop flag. When the stop flag is recognized, the last byte that was placed within the receive FIFO is flagged as the frame's last byte and the data in the temporary FIFO is removed and used as the CRC value for the frame. The receive logic compares the frame's CRC value to the CRC-32 value, which is continuously calculated from the incoming data stream. If CRC and CRC-32 values do not match, the last byte that was placed in the receive FIFO is also tagged with a CRC error. The frame's CRC value is not placed in the receive FIFO. If the stop flag is not properly detected, an abort is signaled.

If software disables the FICP's receiver while it is operating, the data byte being received stops immediately, the serial shifter and receive FIFO are cleared, the System Integration Unit (SIU) takes control of the receive data pin, and the clocks used by the receive logic are shut off to conserve power. The receive data input polarity must be reprogrammed if the receive data pin is used as a GPIO input.

11.3.9 Transmit Operation

Before it enables the fast infrared communications port for transmission, the software can either preload the transmit FIFO by filling it with data or allow service requests to cause the CPU or DMA to fill the FIFO after the FICP is enabled. When the FICP is enabled, the transmit logic issues a service request if its FIFO requires more data.

A minimum of 16 preambles are transmitted for each frame. If data is not available after the sixteenth preamble, additional preambles are transmitted until a byte of valid data resides in the bottom of the transmit FIFO. The preambles are followed by the start flag and the data from the transmit FIFO. Groups of four chips (eight bits) are encoded and loaded in a serial shift register. The contents of the serial shift register are sent out on the transmit data pin, which is clocked by the 8-MHz baud clock. The preamble, start and stop flags, and CRC value are transmitted automatically.

When the transmit FIFO has 32 or more empty entries, an interrupt (if enabled) and DMA service request are sent. If new data does not arrive quickly enough to prevent the FIFO from becoming empty, the transmit logic attempts to transfer additional data from the empty FIFO. Software determines whether to interpret the data underrun (a lack of data) as a signal of normal frame completion or as an unexpected frame termination.

When software selects normal frame completion and an underrun occurs, the transmit logic transmits the CRC value that was calculated during data transmission, including the address and control bytes, followed by the stop flag that marks the end of the frame. The transmitter then continuously transmits preambles until data is available in the FIFO. When data is available, the transmitter starts to transmit the next frame.

When software selects unexpected frame termination and an underrun occurs, the transmit logic transmits an abort and interrupts the CPU. The transmitter continues to send the abort until data is available in the transmit FIFO. When data is available, the FICP transmits 16 preambles and a start flag and starts the new frame. The off-chip receiver can choose to ignore the abort and continue to receive data or signal the FICP to attempt to transmit the aborted frame again.

At the end of each transmitted frame, the FICP sends a pulse called the serial infrared interaction pulse (SIP). A SIP must be sent at least every 500 ms to ensure that low-speed devices (115.2 Kbps and slower) do not interfere with devices that transmit at higher speeds. The SIP simulates a start bit that causes low-speed devices to stay off the air for at least another 500 ms. The SIP pulse forces the transmit data pin high for 1.625 µs and low for 7.375 µs (the total SIP period is 9.0 µs). After the SIP period, the preamble is transmitted continuously to indicate to the off-chip receiver that the FICP's transmitter is in the idle state. The preamble is transmitted until new data is available in the transmit FIFO or the FICP's transmitter is disabled. At least one frame must be completed every 500 ms to ensure that an SIP pulse can keep low-speed devices from interrupting the transmission. Because most IrDA-compatible devices produce an SIP after each frame transmitted, software only needs to ensure that a frame is either transmitted or received by the FICP every 500 ms. Frame length does not represent a significant portion of the 500 ms timeframe in which an SIP must be produced. At 4.0 Mbps, the longest frame allowed is 16,568 bits, which takes just over 4 ms to transmit. The FICP also issues an SIP when the transmitter is first enabled. This ensures that low-speed devices do not interfere as the FICP transmits its data.

If software disables the FICP's transmitter during operation, data transmission stops immediately, the serial shifter and transmit FIFO are cleared, and the SIU takes control of the transmit data pin. The transmit data output's polarity must be properly reprogrammed if the pin is used as a GPIO output.

11.3.10 Transmit and Receive FIFOs

The transmit FIFO is 64 entries deep and 8 bits wide. The receive FIFO is 64 entries deep, 11 bits wide. The receive FIFO uses three bits of its entries as status bits. The transmit FIFO and the receive FIFO use two separate, dedicated DMA requests.

When the transmit FIFO has 32 or more empty bytes, the transmit DMA request and an interrupt (if enabled) are generated and tell the processor to send more data to the FIFO. When the transmit FIFO is full, any more data from the processor is lost. When the receive FIFO reaches its trigger threshold (programmed in ICCR2), the receive DMA request (if no errors are found within the entries) and an interrupt (if enabled) are generated and tell the processor to remove the data from the FIFO. If an error is found in the FIFO's trigger threshold range, DMA requests are disabled and an ICSR0[EIF] interrupt is generated to ensure that the DMA controller does not read the error bytes.

The number of bytes being transferred for each DMA request is programmed in the DMA controller and can be 8, 16, or 32 bytes. The receive FIFO's trigger threshold must be set so that the FIFO has enough data for the DMA controller to read. The transmit FIFO does not have programmable trigger thresholds. Its DMA request is generated when the FIFO has 32 or more empty bytes, regardless of the DMA transfer size.

If the DMA reaches the end of its descriptor chain while servicing the receive FIFO, the processor is interrupted with the ICSR0[EOC] interrupt set. The processor must then link a new descriptor or service the FIFO using interrupts. The EOC interrupt is not asserted if the descriptor chain ends when the last byte of a message is transferred to the DMA.

Note: Ensure that the DMA controller is not servicing the same FIFO when the processor is trying to respond to a receive-error interrupt.

11.3.11 Removing Trailing Bytes in Receive FIFO

If the last byte of a frame (due to the reception of an stop flag or error condition) is below the trigger level of the receive FIFO, the remaining bytes in the frame are called *trailing bytes*. Trailing bytes are considered a special case and are removed from the receive FIFO in a different manner than other bytes. The exact method of removal varies depending on whether the receive FIFO is being serviced by processor interrupts or the DMA.

In this chapter, End of Frame (EOF) refers to an internal tag placed on the last byte of a frame. This EOF byte is tracked throughout the receive FIFO. The last byte of a frame is determined by the reception of a stop flag. For two error conditions, CRC and overflow, similar tags are placed on each erroneous byte in the receive FIFO. See Section 11.4.6 for additional information.

11.3.11.1 Processor Interrupt Mode

When the receive FIFO is being serviced by processor interrupts, the programmer must ensure that ICCR2[TRAIL] is clear. Clearing ICCR2[TRAIL] does the following:

1. Disables DMA requests for trailing bytes.
2. Sets the ICSR0[EIF] bit (see Table 11-6) when bytes containing either an End of Frame or error flag fall below the trigger level.

Note: When ICCR2[TRAIL] is set, ICSR0[EIF] is set on error flags only.

In interrupt mode, trailing bytes due to the reception of an stop flag or an error condition must be treated the same by the processor. The processor is notified of the presence of trailing bytes by the ICSR0[EIF] interrupt. If ICSR0[TRAIL] is clear, ICSR0[EIF] signals trailing bytes because of either a receive error or a end of frame (EOF) flag. In either case, the assertion of the EIF interrupt disables receive FIFO interrupts. The processor must now remove the trailing bytes and any associated error conditions as follows:

1. Read the ICSR1 register to determine if the byte at the bottom of the FIFO contains EOF or error flags.
2. Read the ICDR register to remove the byte at the bottom of the FIFO.
3. Continue with steps 1 and 2 until the EOF is found. The EOF flag marks the last byte of the message.
4. Read the ICSR0 register. If the EIF bit is still set, then another frame exists in the FIFO that contains either an error or the EOF flag below the trigger level. If this is the case, steps 1–4 must be repeated.

When the ICSR0[EIF] clears, receive FIFO service interrupts are re-enabled, and the processor continues to service the FIFO normally.

11.3.11.2 DMA Mode

When the receive FIFO is being serviced by DMA, software must set ICCR2[TRAIL]. Doing so results in the following:

1. Enables DMA requests for trailing bytes.
2. Sets ICSR0[EIF] (see Section 11.4.5) for EOF flags only.

Note: When ICCR2[TRAIL] is clear, ICSR0[EIF] is set for both EOF and error flags.

3. Enables End of Descriptor Chain (EOC) interrupt.

In this mode, trailing bytes caused by the reception of a stop flag are treated differently than trailing bytes caused by error conditions. If the FICP detects trailing bytes caused by an EOF flag, the ICSR0[EIF] bit is *not* set, and the processor is *not* interrupted. Instead, a DMA request is made for the exact number of bytes remaining in the frame. When the DMA controller reads the last byte of the frame, the FICP asserts an End of Receive (EOR) signal to the DMA controller. The DMA controller can then be programmed to notify the processor of the EOR using the DCSR[EORINT] interrupt in the DMA controller. The FICP makes new DMA requests if additional frames exist in the receive FIFO.

11.3.11.2.1 Error Handling in DMA Mode

If the FICP detects trailing bytes caused by receive errors, all DMA requests are disabled when the error condition falls below the receive FIFO trigger level. At this point, the error and trailing byte condition must be handled as described in Section 11.3.11.1.

11.3.11.2.2 DMA Programming Errors

If the DMA receive channel stops prematurely because of a DMA programming error such as an exhausted descriptor chain, the FICP interrupts the processor with an EOC interrupt if enabled with the ICCR2[TRAIL] bit. When the processor receives this interrupt, it must restore FIFO service by either programming another descriptor or servicing the receive FIFO using interrupt mode. If the FICP is in DMA mode, ICCR2[TRAIL] must be set. Operating the FICP in DMA mode without this bit set puts the FICP into a legacy mode that the PXA27x processor does not support. The EOC interrupt is not asserted if the descriptor chain ends when the last byte of a message is transferred out of the receive FIFO. For more information, see the description of the EOC bit in Section 11.4.5.

11.3.12 32-Bit Peripheral Bus

The fast infrared communications port supports 32-bit transfers. The bytes must be written/read in little-endian format (7:0) with byte 3 (the most recent byte) starting at bit 31, byte 2 starting at bit 23, and so on. An 8-bit bus can be emulated by programming the DMA controller and processor to read/write only one byte at a time. In this case, the upper bits 31:8 are ignored by the FICP and are not used.

The fast infrared communications port supports an 8- or 32-bit peripheral bus. The default is an 8-bit bus. If 32-bit bus mode is preferred, set the ICCR2[BUS] bit. The bytes are written in little-endian format (7:0) with byte 3 (the most recent byte) starting at bit 31, byte 2 starting at bit 23, and so on.

- **8-Bit Mode—**Only the least significant byte contains valid data on the peripheral bus. The upper 24 bits are ignored.
- **32-Bit Mode—**The FICP can read/write partial words of 1, 2, 3, or 4 continuous bytes from the peripheral bus. The method in which the valid bytes of data are determined differs, depending on whether the transaction is being handled by the DMA controller or programmed I/O (PIO).
- **DMA—**The DMA controller can read/write 1, 2, 3, or 4 bytes per word. The number of valid bytes available per word is determined internally between the DMA controller and the FICP.
- **Programmed I/O—**The processor has the restriction of only being able to read/write 1, 2, or 4 bytes per word. When reading, the processor must read the FIFO Occupancy register (ICFOR) to retrieve the number of bytes available in the Receive buffer. If the number of bytes available

is 4 or greater, the processor can request any number of bytes per word (except 3). If the number is less than 4, users must request the proper number of bytes. When three bytes are remaining, users can read 1 byte at a time or request either 2 bytes followed by 1 byte or 1 byte followed by 2 bytes. All three are allowed. If the processor reads more than the number of bytes available in the receive buffer, the FICP retrieves invalid data for the invalid bytes. The receive FIFO counters do not increment past the valid number of bytes. For example, if the processor tries to read 4 bytes when only 2 remain in the receive FIFO, the head counter in the FIFO increments by 2, not 4.

Note: When using PIO, no FICP interrupt is generated on a DMA end-of-chain condition.

11.4 Register Descriptions

The fast infrared communications port has seven registers: three control registers, one data register, and three status registers. The FICP registers are 32 bits wide, but only the lower 8 bits of control and status registers have valid data. The FICP does not support byte or half-word operations. CPU reads and writes to the FICP registers must be word-wide. The data register can contain valid data on all 32 bits and can be accessed in 1, 2, 3, or 4 byte increments.

The control registers determine: IrDA transmission rate, address match value, how transmit FIFO underruns are handled, normal or active low transmit and receive data, whether transmit and receive operations are enabled, the FIFO interrupt service requests, receive address matching, and loopback mode.

The data register addresses the top of the transmit FIFO and the bottom of the receive FIFO. Reads from the data register access the receive FIFO. Writes to the data register access the transmit FIFO.

The status registers contain: CRC, overrun, underrun, framing, and receiver abort errors; the transmit FIFO service request; the receive FIFO service request; and end-of-frame conditions. Each of these hardware-detected events signals an interrupt request to the interrupt controller. The status registers also contain flags for transmitter busy, receiver synchronized, receive FIFO not empty, and transmit FIFO not full (no interrupts are generated for flag events).

11.4.1 FICP Control Register 0 (ICCR0)

The FICP Control register 0 (ICCR0) contains eight valid bit fields that control various functions for 4-Mbps IrDA transmission. The FICP must be disabled (RXE = TXE = 0) when ICCR0[ITR] and ICCR0[LBM] are changed. To allow various modes to be changed during active operation, ICCR0[7:2] can be written when the FICP is enabled.

This a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 11-2. ICCR0 Bit Definitions (Sheet 1 of 2)

Physical Address 0x4080_0000 — ICCR0 — FICP

Bit	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
User Settings																																
	reserved																								AME	TIE	RIE	RXE	TXE	TUS	LBM	ITR
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7	R/W	AME	Address Match Enable Receive logic compares the address of the incoming frames to the Address Match Value field in ICCR1. 0 = Disables receiver address match function. Stores data in receive FIFO. 1 = Enables receiver address match function. Does not put data in the receive FIFO unless address is recognized or address is the broadcast address.
6	R/W	TIE	Transmit FIFO Interrupt Enable 0 = Transmits FIFO service request ICSR0[TFS] does not generate an interrupt. 1 = Transmits FIFO service request generates an interrupt. Setting TIE does not clear TFS or prevent TFS from being set or cleared by the transmit FIFO. TIE does not affect transmit FIFO DMA requests.
5	R/W	RIE	Receive FIFO Interrupt Enable 0 = Receive FIFO service request ICSR0[RFS] does not generate an interrupt. 1 = Receive FIFO service request generates an interrupt. Setting RIE does not clear RFS or prevent RFS from being set or cleared by the receive FIFO. RIE does not affect receive FIFO DMA requests.
4	R/W	RXE	Receive Enable 0 = FICP receive logic disabled. 1 = FICP receive logic enabled if ICCR0[ITR] is set. All other control bits must be configured before setting RXE. If RXE is cleared while receiving data. Then reception is stopped immediately, all data within the receive FIFO and serial input shifter is cleared, and control of the receive data pin is given to the SIU. While communication is normally half-duplex, it is possible to transmit and receive data at the same time. This is used for testing in loopback mode. If RXE is used to clear the receive FIFO, check ICSR1[RNE] to ensure the receive FIFO is clear before re-enabling the receiver.

Table 11-2. ICCR0 Bit Definitions (Sheet 2 of 2)

Physical Address 0x4080_0000 — **ICCR0** — **FICP**

	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
User Settings																																
Bit	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																								AME	TIE	RIE	RXE	TXE	TUS	LBM	ITR
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
3	R/W	TXE	Transmit Enable 0 = FICP transmit logic disabled. 1 = FICP transmit logic enabled if ICCR0[ITR] is set. All other control bits must be configured before TXE is set. An SIP is transmitted immediately after the transmitter is enabled. If the transmit FIFO is empty, preambles are sent until data is placed in the FIFO. If TXE is cleared while it transmits data, transmission stops immediately, all data in the transmit FIFO and serial output shifter is cleared, and the SIU takes control of the transmit data pin. While communication is normally half-duplex, it is possible to transmit and receive data at the same time. Duplex communication is used for testing in loopback mode. If TXE is used to clear the transmitter, check ICSR1[TBY] to ensure the transmitter is not busy before the transmitter is re-enabled.
2	R/W	TUS	Transmit FIFO Underrun Select A transmit FIFO underrun can either end the current frame normally, or transmit an abort. 0 = Transmit FIFO underrun causes CRC, stop flag, and SIP (if enabled) to be transmitted, and masks transmit underrun interrupt generation. 1 = Transmit FIFO underrun causes abort to be transmitted, and generates an interrupt. Clearing ICCR0[TUS] does not affect the current state of ICSR0[TUR] or prevent TUR from being set or cleared by the transmit FIFO. After an abort, a SIP is transmitted followed by 16 preambles. Preambles continue until data is in the FIFO. To change the state of TUS during operation, users must fill the transmit FIFO to ensure TUS is not written at the same time that the transmit FIFO underflows.
1	R/W	LBM	Loopback Mode Used for testing the FICP. 0 = Normal FICP operation enabled. 1 = Output of transmit serial shifter is connected to input of receive serial shifter.
0	R/W	ITR	IrDA Transmission 0 = FICP unit is not enabled. 1 = FICP unit is enabled.

11.4.2 FICP Control Register 1 (ICCR1)

FICP Control register 1 (ICCR1) contains the eight-bit address match value field that the FICP uses to selectively receive frames. To allow the address match value to be changed during active receive operation, ICCR1 can be written while the FICP is enabled.

This a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 11-3. ICCR1 Bit Definitions

Physical Address 0x4080_0004 — **ICCR1** — **FICP**

	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
User Settings																																
Bit	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																								AMV							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	AMV	Address Match Value The eight-bit value used by receiver logic to compare to address of incoming frames. If ICCR0[AME] = 1 and AMV matches the address of the incoming frame, store the frame address, control, and data in receive FIFO. If the address does not match, ignore the frame and search for the next preamble. The broadcast address 0xFF in the incoming frame always generates a match.

11.4.3 FICP Control Register 2 (ICCR2)

The FICP Control register 2 (ICCR2) contains two bit fields that control the polarity of the transmit and receive data pins and two bits that determine the trigger threshold for the receive FIFO. The FICP must be disabled (RXE = TXE = 0) when these bits are changed.

This a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 11-4. ICCR2 Bit Definitions

Physical Address 0x4080_0008																ICCR2												FICP						
User Settings																																		
Bit	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																										BUS	TRAIL	RXP	TXP	TRIG	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	1	1	0	0

Bits	Access	Name	Description
31:6	—	—	reserved
5	R/W	BUS	32-Bit Peripheral Bus 0 = 8-bit peripheral bus 1 = 32-bit peripheral bus
4	R/W	TRAIL	Trailing Bytes 0 = Trailing bytes are handled by the processor. 1 = Trailing bytes are handled by the DMA when enabled. Setting this bit also enables the EOC interrupt.
3	R/W	RXP	Receive Pin Polarity Select 0 = Data from the receive data pin is inverted before being used by the FICP unit. 1 = Data from the receive data pin to the FICP unit is not inverted. Set on reset.
2	R/W	TXP	Transmit Pin Polarity Select 0 = Data from the FICP is inverted before being sent to the transmit data pin. 1 = Data from the FICP is not inverted before being sent to the transmit data pin. Set on reset.
1:0	R/W	TRIG	Receive FIFO Trigger Threshold The receive FIFO generates service requests when the FIFO has reached the trigger threshold and has no errors in its data. The DMA controller data transfer size must be set to the same size as the receive FIFO trigger threshold. To change the trigger threshold, the receive FIFO must be disabled. 0b00 = Receive FIFO service request is generated when the FIFO has 8 bytes or more. 0b01 = Receive FIFO service request is generated when the FIFO has 16 bytes or more. 0b10 = Receive FIFO service request is generated when the FIFO has 32 bytes or more. 0b11 = reserved

11.4.4 FICP Data Register (ICDR)

The FICP Data register (ICDR) is a 32-bit register. Its lower eight bits are the top entry of the transmit FIFO when the register is written and the bottom entry of the receive FIFO when the register is read.

Reads from ICDR access the lower eight bits of the receive FIFO's bottom entry. As data enters the top of the receive FIFO, bits 8–10 are used as tags to indicate conditions that occur as each piece of data is received. The tag bits are transferred down the FIFO with the data byte that encountered the condition. When data reaches the bottom of the FIFO, bit 8 of the FIFO entry is transferred to the end-of-frame (EOF) flag, bit 9 to the CRC error (CRE) flag, and bit 10 to the receiver overrun (ROR) flag. All these flags are in FICP Status register 1. These flags can be read to determine whether the value at the bottom of the FIFO represents the frame's last byte or an error that was encountered during reception. After the flags are checked, the FIFO value can be read. This causes the data in the next location of the receive FIFO to be transferred to the bottom entry and its EOF, CRE, and ROR bits to be transferred to the status register.

The end/error in FIFO (EIF) flag is set in status register 0 when a tag bit is set in any of the receive FIFO's bottom 8, 16, or 32 entries, as determined by the trigger threshold. The EIF flag is cleared when no error bits are set in the FIFO's bottom entries. When EIF is set, an interrupt is generated and the receive FIFO DMA request is disabled. Software must empty the FIFO and check for the EOF, CRE, and ROR error flags in ICSR1 before it removes each data value from the FIFO. After each entry is removed, the EIF bit must be checked to determine if any set end or error tag remains. The procedure is repeated until all set tags are flushed from the FIFO's bottom entries. When EIF is cleared, DMA service for the receive FIFO is re-enabled.

Both FIFOs are cleared when the PXA27x processor is reset. The transmit FIFO is cleared when TXE is 0. The receive FIFO is cleared when RXE is 0.

This a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 11-5. ICDR Bit Definitions

Physical Address 0x4080_000C																ICDR																FICP							
User Settings																																						
Bit	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
	DATA																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:0	R/W	DATA	Top/Bottom of Transmit/Receive FIFO Read = Read data from front of receive FIFO. Write = Place data at end of transmit FIFO.

11.4.5 FICP Status Register 0 (ICSR0)

FICP Status register 0 (ICSR0) contains bits that signal the transmit FIFO service request, receive FIFO service request, receiver abort, transmit FIFO underrun, framing error, and the end/error in receive FIFO conditions. Each of these hardware-detected events signals an interrupt request to the interrupt controller.

If a bit signals an interrupt request, it signals the interrupt request as long as the bit is set. When the bit is cleared, the interrupt is cleared. Read/write bits are called *status* bits. Read-only bits are called *flags*. Status bits that must be cleared by software after they are set by hardware are called *sticky* status bits. Writing a 1 to a sticky status bit clears it. Writing a 0 to a sticky status bit has no effect. Read-only flags are set and cleared by hardware. Writes to read-only flags have no effect. Some bits that cause interrupts have corresponding mask bits in the control registers. These bits are indicated in the sections that follow.

This a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 11-6. ICSR0 Bit Definitions (Sheet 1 of 2)

Physical Address 0x4080_0014 — **ICSR0** — **FICP**

	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
User Settings																																
Bit	reserved																									EOC	FRE	RFS	TFS	RAB	TUR	EIF
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:7	—	—	reserved
6	R/W	EOC	DMA End-of-Descriptor Chain EOC bit is set (and interrupt asserted if enabled) when the DMA signals to the FICP that the receive DMA channel has exhausted its descriptor chain and stopped prematurely. This bit asserts only when necessary, so EOC does not assert if the last byte of the message has been transferred out of the receive FIFO. If data is still being received, software must either reprogram the DMA chain or service the FIFO using interrupt requests. EOC must be cleared by the processor. This interrupt is enabled when ICCR0[TRAIL] is set. 0 = DMA has not signaled the end of its programmed descriptor chain. 1 = DMA has signaled the end of its programmed descriptor chain.
5	R/W	FRE	Framing Error 0 = No framing errors encountered in the receipt of this data. 1 = Framing error occurred. A preamble was followed by something other than another preamble or start flag. Requests interrupt.
4	R	RFS	Receive FIFO Service Request (read-only) 0 = Receive FIFO has not reached it trigger threshold or receiver disabled. 1 = Receive FIFO has reached its trigger threshold and receiver is enabled. DMA service request signaled. Interrupt request signaled if not masked by ICCR0[RIE].

Table 11-6. ICSR0 Bit Definitions (Sheet 2 of 2)

Physical Address 0x4080_0014 — **ICSR0** — **FICP**

	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
User Settings																																
Bit	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																									EOC	FRE	RFS	TFS	RAB	TUR	EIF
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
3	R	TFS	Transmit FIFO Service Request (read-only) 0 = Transmit FIFO has more than 32 entries of data or transmitter disabled. 1 = Transmit FIFO has 32 or less entries of data and transmitter is enabled. DMA service request signaled. Interrupt request signaled if not masked by ICCR0[TIE].
2	R/W	RAB	Receiver Abort 0 = No abort has been detected for the incoming frame. 1 = Abort detected during receipt of incoming frame. Two or more chips containing no pulses or any invalid chips were detected on the receive pin. EOF bit set on last piece of valid data received before the abort. Requests interrupt.
1	R/W	TUR	Transmit FIFO Underrun 0 = Transmit FIFO has not experienced an underrun. 1 = Transmit logic attempted to fetch data from transmit FIFO while it was empty. Interrupt request signaled if not masked by ICCR0[TUS]. Underruns are not generated when the FICP transmitter is first enabled and is idle.
0	R	EIF	End/Error in FIFO (read-only) 0 = Bits 8–10 are not set within any of the entries below the trigger threshold of the receive FIFO. Receive FIFO DMA service requests are enabled. 1 = One or more tag bits (8–10) are set within the entries below the trigger threshold of the receive FIFO. Requests interrupt, disable receive FIFO DMA service requests. This interrupt is not maskable in the FICP. Once the invalid bytes have been removed from the FIFO and EIF is cleared, DMA requests are automatically enabled.

11.4.6 FICP Status Register 1 (ICSR1)

FICP Status register 1 (ICSR1) contains flags that indicate that the receiver is synchronized, the transmitter is active, the transmit FIFO is not full, the receive FIFO is not empty, and that an EOF, CRE, or underrun error has occurred. All bits in ICSR1 are read-only.

This is a read-only register. Ignore reads from reserved bits.

Table 11-7. ICSR1 Bit Definitions (Sheet 1 of 2)

Physical Address 0x4080_0018																														
ICSR1																											FICP			
User Settings																														
Bit	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																									ROR	CRE	EOF	TNF	RNE	TBY	RSY
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	1	0	0	0

Bits	Access	Name	Description
31:7	—	—	reserved
6	R	ROR	Receive FIFO Overrun (read-only) 0 = Receive FIFO has not experienced an overrun. 1 = Receive logic attempted to place data into receive FIFO while it was full. Data received after the FIFO is full are lost. Each time an 11-bit value reaches the bottom of the receive FIFO, bit 10 from the last FIFO entry is transferred to the ROR bit.
5	R	CRE	CRC Error (read-only) 0 = CRC not encountered yet or no CRC check errors encountered in the receipt of data. 1 = CRC calculated on the incoming data. Does not match CRC value contained within the received frame. When CRE is set, ICSR0[EIF] is also set, causing an interrupt to be generated, and the receive FIFO DMA requests to be disabled. Each time an 11-bit value reaches the bottom of the receive FIFO, bit 9 from the last FIFO entry is transferred to the CRE bit.
4	R	EOF	End of Frame (read-only) 0 = Current frame has not completed. 1 = The value at the bottom of the receive FIFO is the last byte of data within the frame, including aborted frames. When EOF is set, ICSR0[EIF] is also set causing an interrupt to be generated, and the receive FIFO DMA requests to be disabled. Each time an 11-bit value reaches the bottom of the receive FIFO, bit 8 from the last FIFO entry is transferred to the EOF bit.
3	R	TNF	Transmit FIFO Not Full (read-only) 0 = Transmit FIFO is full. 1 = Transmit FIFO is not full. No interrupt generated.

Table 11-7. ICSR1 Bit Definitions (Sheet 2 of 2)

Physical Address 0x4080_0018																ICSR1															FICP							
User Settings																																					
Bit	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																									ROR	CRE	EOF	TNF	RNE	TBY	RSY
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	1	0	0	0

Bits	Access	Name	Description
2	R	RNE	Receive FIFO Not Empty (read-only) 0 = Receive FIFO is empty. 1 = Receive FIFO is not empty. No interrupt generated.
1	R	TBY	Transmitter Busy Flag (read-only) 0 = Transmitter is idle (continuous preambles) or disabled. 1 = Transmit logic is currently transmitting a frame (address, control, data, CRC, or start/stop flag). No interrupt generated.
0	R	RSY	Receiver Synchronized Flag (read-only) 0 = Receiver is in hunt mode or is disabled. 1 = Receiver logic is synchronized with the incoming data (no interrupt generated).

11.4.7 FICP FIFO Occupancy Status Register (ICFOR)

This register shows the number of bytes currently remaining in the receive FIFO. It can be used by the processor to determine the number of trailing bytes to remove in the case when the DMA reaches the end of its Descriptor chain or when ICCR2[TRAIL] = 0, which indicates the processor will remove trailing bytes as opposed to the DMA (see Section 11.3.11). ICFOR can also determine the number of bytes available in the receive FIFO when DMA requests are disabled and the processor is servicing the FIFO. The register organization and the individual bit definitions are shown in Table 11-8.

This a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 11-8. ICFOR Bit Definitions

Physical Address 0x4080_001C																ICFOR															FICP							
User Settings																																					
Bit	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																									Byte Count						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:7	—	—	reserved
6:0	R/W	Byte Count	Number of bytes (0–64) remaining in the receiver FIFO

11.5 Register Summary

Table 11-9 shows the registers associated with the fast infrared communications port and the physical addresses to access them. These registers must be mapped as non-cacheable, non-bufferable, and can only be accessed as word accesses. They are grouped together within one page, and all have the same memory protections.

Table 11-9. Fast Infrared Communications Port Register Summary

Address	Name	Description	Page
0x4080_0000	ICCR0	FICP Control register 0	11-10
0x4080_0004	ICCR1	FICP Control register 1	11-13
0x4080_0008	ICCR2	FICP Control register 2	11-14
0x4080_000C	ICDR	FICP Data register	11-15
0x4080_0010	—	reserved	
0x4080_0014	ICSR0	FICP Status register 0	11-16
0x4080_0018	ICSR1	FICP Status register 1	11-18
0x 4080 001C	ICFOR	FICP FIFO Occupancy Status register	11-19
0x 4080 0020–0x 4080 FFFC	—	reserved	

USB Client Controller 12

This section describes the Universal Serial Bus (USB) client, including information on the PXA27x processor implementation of the Universal Serial Bus (USB) device controller, such as number, type, and function of the endpoints, the interrupts to the processor, and the transmit/receive FIFO interface. A working knowledge of the *Universal Serial Bus Specification*, Revision 1.1[1] is necessary to fully understand the material contained in this chapter. The Universal Serial Bus device controller (UDC) is USB 1.1-compliant and supports all standard device requests issued by any USB host controller. Refer to the *Universal Serial Bus Specification*, Revision 1.1, the *On-The-Go Supplement to Universal Serial Bus Specification*, Revision 2.0[2], and the *Pull-up/Pull-down Resistors Engineering Change Notice to the USB 2.0 Specification*[3] for a full description of the USB protocol and its operation.

12.1 Overview

The UDC supports 24 endpoints (endpoint 0 plus 23 programmable endpoints). The UDC is a USB Revision 1.1-compliant, full-speed device that operates half-duplex at a baud rate of 12 Mbps (as a slave only, not as a host or hub controller).

The serial information transmitted and received by the USB client controller contains layers of communications protocols as defined by *Universal Serial Bus Specification, Revision 1.1*, the most basic of which are fields.

- USB *fields* include: sync, packet identifier, address, endpoint, frame number, data, and CRC. Fields are used to produce packets. Depending on the packet function, a different combination and number of fields can be used.
- *Packet* types include: token, start of frame, data, and handshake. Packets are assembled into groups to produce transfers, transactions, and frames.
- *Transfers* fall into four groups: bulk, control, interrupt, and isochronous.
- *Transactions* fall into four groups: IN, OUT, SOF, and SETUP. Data flow is relative to the USB host controller. IN packets represent data flow from the USB client controller to the host controller. OUT packets represent data flow from the USB host controller to the client controller.

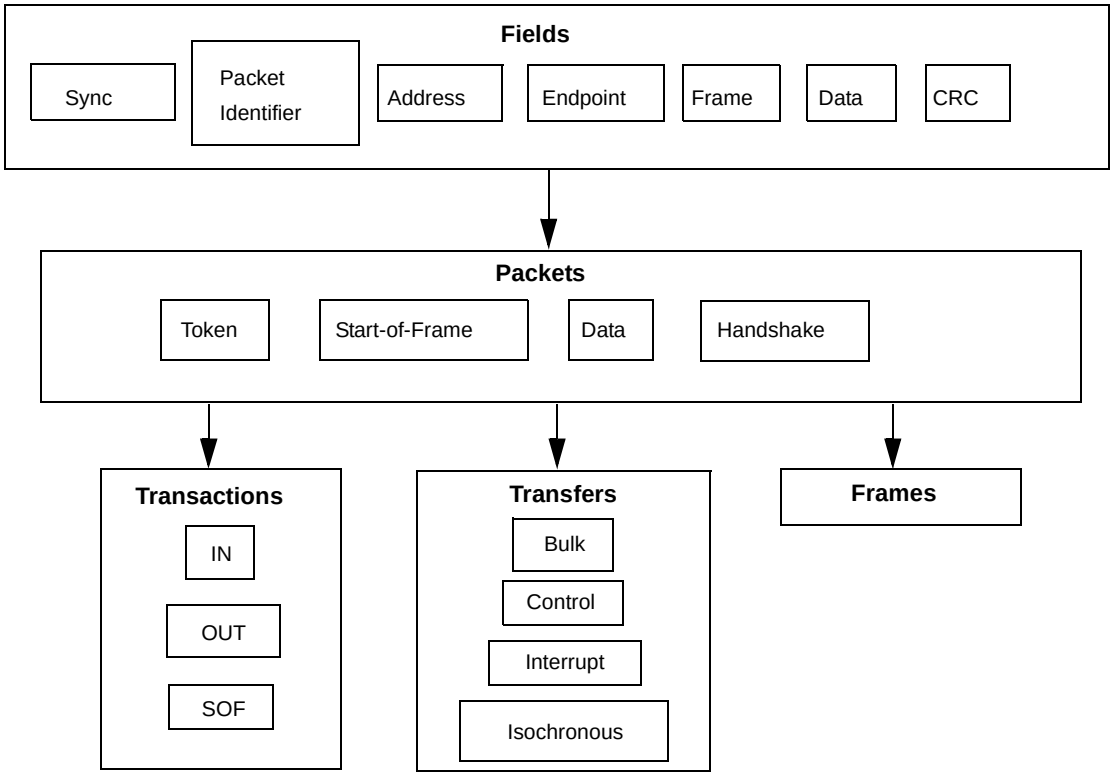
Figure 12-1 graphically represents the communications layers in the protocol. See the *Universal Serial Bus Specification, Revision 1.1* for more details on USB protocol.

The UDC uses single-ported memory to support FIFO operations. Bulk, isochronous, and interrupt endpoint FIFO structures can be double-buffered to enable the endpoint to process one packet while assembling another. Use either DMA or the Intel XScale® core to fill and empty the FIFOs. An interrupt, DMA service request, or polling can be used to detect packet receipt.

1. The latest revision of the *Universal Serial Bus Specification Revision 1.1* can be accessed at: http://www.usb.org/
2. The latest revision of the *On-The-Go Supplement to Universal Serial Bus Specification Revision 2.0* can be accessed at: http://www.usb.org/
3. The latest revision of the *Pull-up/Pull-down Resistors Engineering Change Notice to the Universal Serial Bus 2.0 Specification* can be accessed at: http://www.usb.org/

The USB host controller referenced in this chapter refers to any USB host controller that is compliant to the *Universal Serial Bus Specification, Revision 1.1*, including the PXA27x processor's internal USB host controller.

Figure 12-1. Communications Protocol Layers in the USB Client Controller



12.2 Features

- USB Revision 1.1, full-speed compliant device
- 23 programmable endpoints
  - — Type: bulk, isochronous, or interrupt
  - — Direction: in or out
  - — Maximum packet size
  - — Programmable configuration, interface and alternate interface setting numbers
- Endpoint 0 for control IN and OUT
- Four configurations:
  - — Three programmable configurations with up to seven interfaces with seven alternate interface settings
  - — Default configuration 0 with one interface and control endpoint 0
- Configurable 4-Kbyte memory for endpoint data storage

12.3 Signal Descriptions

This section describes the signals used by the USB client controller (See Table 12-1).

Table 12-1. USB Client Controller Interface I/O Signal Descriptions

Name	Type	Description
USBC_P	Input/Output	USB client port positive pin of differential pair
USBC_N	Input/Output	USB client port negative pin of differential pair

12.3.1 Bidirectional Signals

USBC_P and USBC_N are the differential lines of the USB cable. Using differential signaling allows transmitting multiple states on the serial bus. These states are combined to transmit data as well as various bus conditions, including: idle, resume, start-of-packet (SOP), end-of-packet (EOP), disconnect, connect, and reset. Four distinct states are represented using differential data by decoding the polarity of the USBC_P and USBC_N pins. Two of the four states are used to represent data. A one is represented when USBC_P is high and USBC_N is low; a zero is represented when USBC_P is low and USBC_N is high. The remaining two states and pairings of the four encodings are decoded further to represent the current state of the USB. Table 12-2 shows how seven different bus states as well as one and zero are represented using differential signaling.

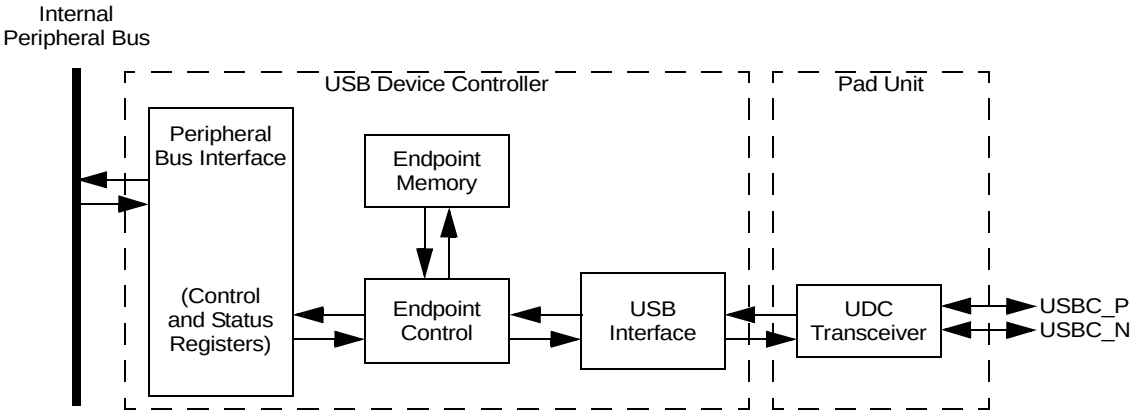
Table 12-2. USB States Using Differential Signaling

Bus State	USBC_P Pin Level	USBC_N Pin Level
1	High	Low
0	Low	High
Idle	High from EOP to SOP (Held by bus termination resistors)	Low from EOP to SOP (Held by bus termination resistors)
Resume	Low	High
Start-of-Packet (SOP)	Transition from idle to resume	
End-of-Packet (EOP)	Low for 2 bit-times followed by an idle for 1 bit-time (Low, Low, High)	Low for 2 bit-times followed by an idle for 1 bit-time (Low, Low, Low)
Disconnect	Below single-ended low trigger threshold for more than 2.5 µs. (Disconnect is the static bus condition that results when no device is plugged into a hub port.)	
Connect	USBC_P OR USBC_N high for more than 2.5 µs.	
Reset	Low for more than 2.5 µs.	Low for more than 2.5 µs.

12.4 Operation

The UDC consists of four major components: the peripheral bus interface, endpoint memory, endpoint control, and USB interface. The peripheral bus interface contains the UDC control and status registers for the endpoint configuration data and provides the interface between the PXA27x processor and the USB data. The endpoint memory is a 4-Kbyte SRAM used for USB endpoint data storage. It has 32 bytes dedicated to endpoint 0, allowing the remainder of its memory to be allocated to any of the 23 programmable endpoints. The endpoint control and USB interface blocks provide the USB functionality. Figure 12-2 is a block diagram of the USB client controller and its dedicated I/O.

Figure 12-2. USB Client Controller Block Diagram



12.4.1 Peripheral Bus Interface and Control/Status Registers

The UDC is a slave peripheral device that is connected to the internal peripheral bus. All user-initiated accesses to the UDC registers and endpoint memory are completed using the internal peripheral bus. The control and status registers include registers for frame-number storage, UDC top-level control and status, interrupt control and status, endpoint control, status, and data transfer.

The UDC Control register (UDCCR) provides control and status of internal UDC functions. The UDCCR status bits indicate the current USB configuration, interface, and alternate interface setting numbers assigned the UDC by the USB host controller. The UDCCR also contains a status bit to indicate if the UDC is actively communicating on the USB, and a status bit to indicate an unusable endpoint memory configuration. The UDCCR also allows selection of UDC enable for USB operation, UDC resume, and endpoint memory configuration control (refer to Section 12.6.1 for more details on the UDCCR).

Either processor interrupts or polling can be used to determine whether USB activity occurs. The Frame Number register (UDCFNR) holds the frame number contained in the last received start-of-frame (SOF) packet.

Although the UDC can generate only a single interrupt to the processor's interrupt controller, there are 53 sources for this interrupt. Each of the 24 endpoints (0 and A–X) has two interrupts: packet complete and FIFO error. In addition, the UDC has five interrupts that can be generated based on USB activity. The interrupt sources are shown in the UDC Interrupt Status registers, which must be read to determine the cause of the interrupt being generated. In addition, USB activity can be determined by reading the UDC Interrupt Status registers. If polling is used, the Endpoint Control/ Status registers can be read to determine activity on the USB.

The UDC Interrupt Control registers contain interrupt-enable bits that enable the generation of the UDC interrupt. When an interruptible event occurs, the appropriate status bit in the Endpoint Control/Status register is set, and if the corresponding interrupt-enable bit in the Interrupt Control register is set, then the appropriate bit in the Interrupt Status register is set and an interrupt is generated. An interrupt is cleared by setting the appropriate bit in the Interrupt Status registers (see Section 12.6.2 and Section 12.6.5 for more details on the UDC Interrupt Control and Status registers).

Endpoint 0 is the only control endpoint and the only bidirectional endpoint in the UDC, and has characteristics different from the programmable endpoints A–X. Key characteristics of endpoint 0 include the following:

- Control/Status, Byte Count, and Data registers
- Configuration is fixed and does not use a configuration register
- Enabled for every USB configuration and interface
- Bidirectional endpoint with 32 bytes of USB data-storage space allocated in the endpoint memory: 16 bytes of FIFO memory are used for IN data and 16 bytes are used for OUT data
- USB data space is not double-buffered.
- Only endpoint configured and available for USB operation after USB reset, and before the USB host controller has enumerated the UDC

The Endpoint 0 Control/Status and Byte Count registers provide the status of the endpoint 0 IN and OUT buffers. The Receive FIFO Not Empty bit (UDCCSR0[RNE]) and OUT Packet Complete bit (UDCCSR0[OPC]) are set when a complete data packet has been received from the USB host controller. If the packet is part of a SETUP transaction, the Setup Active bit (UDCCSR0[SA]) is also set. The endpoint 0 transmit FIFO is flushed by the UDC after receiving an OUT data packet from the USB host controller. The Byte Count register (UDCBCR0) indicates the number of bytes of data that need to be unloaded from the receive buffer. As data is read from the endpoint 0 receive buffer using the Endpoint 0 Data register UDCDR0, the Byte Count register value is decremented to indicate the number of bytes remaining in the buffer. When all of the data has been unloaded from the receive buffer, UDCCSR0[RNE] is cleared by the UDC to indicate the receive buffer is empty. After reading all of the data from the endpoint 0 receive buffer, software must clear UDCCSR0[OPC] to enable the buffer to receive another USB data packet.

Loading a maximum packet size of 16 bytes into the endpoint 0 transmit FIFO automatically sets UDCCSR0[IPR]. If less than 16 bytes are loaded into the endpoint 0 transmit FIFO, UDCCSR0[IPR] must be explicitly set to indicate a complete packet has been loaded. When the data has been transmitted to the USB host controller, the UDC clears UDCCSR0[IPR] to indicate the packet has been sent (see Section 12.6.7, Section 12.6.9, and Section 12.6.10 for more information on Endpoint 0 registers).

Each of the 23 programmable endpoints, referred to as endpoints A–X, has a Configuration register, Control/Status register, Byte Count register, and a Data register. The Configuration registers set the configuration, interface, alternate setting and endpoint numbers, and maximum packet size, as well as enable double-buffering for each endpoint. The Configuration registers can be written only when the UDC is not enabled (UDCCR[UDE] is clear). When UDCCR[UDE] is set, the endpoint configurations are loaded into the USB interface block and are set to read-only access (refer to Section 12.4.2 for more information on configuring the programmable endpoints).

The Control/Status, Byte Count, and Data registers control the operation of each endpoint after enumeration. If an endpoint has double-buffering disabled, the Control/Status and Byte Count registers provide the status of the endpoint buffer. If the endpoint is configured as an OUT endpoint, the FIFO Service (FS) and Packet Complete (PC) bits in the Endpoint Control/Status register are set when a complete data packet has been received from the USB host controller.

The Byte Count register of each endpoint indicates the number of data bytes that need to be unloaded from the buffer. As data is read from the FIFO memory using the Data register, the corresponding Byte Count register value is decremented to indicate the number of bytes remaining in the buffer. When all of the data has been unloaded from the FIFO memory, the FS and Buffer Not Empty/Buffer Not Full (BNE/BNF) bits in the Control/Status register and the Byte Count (BC) in the Byte Count register are cleared by the UDC to indicate the current buffer is empty. After reading all of the data from the endpoint buffer, software must clear the PC bit in the corresponding Control/Status register.

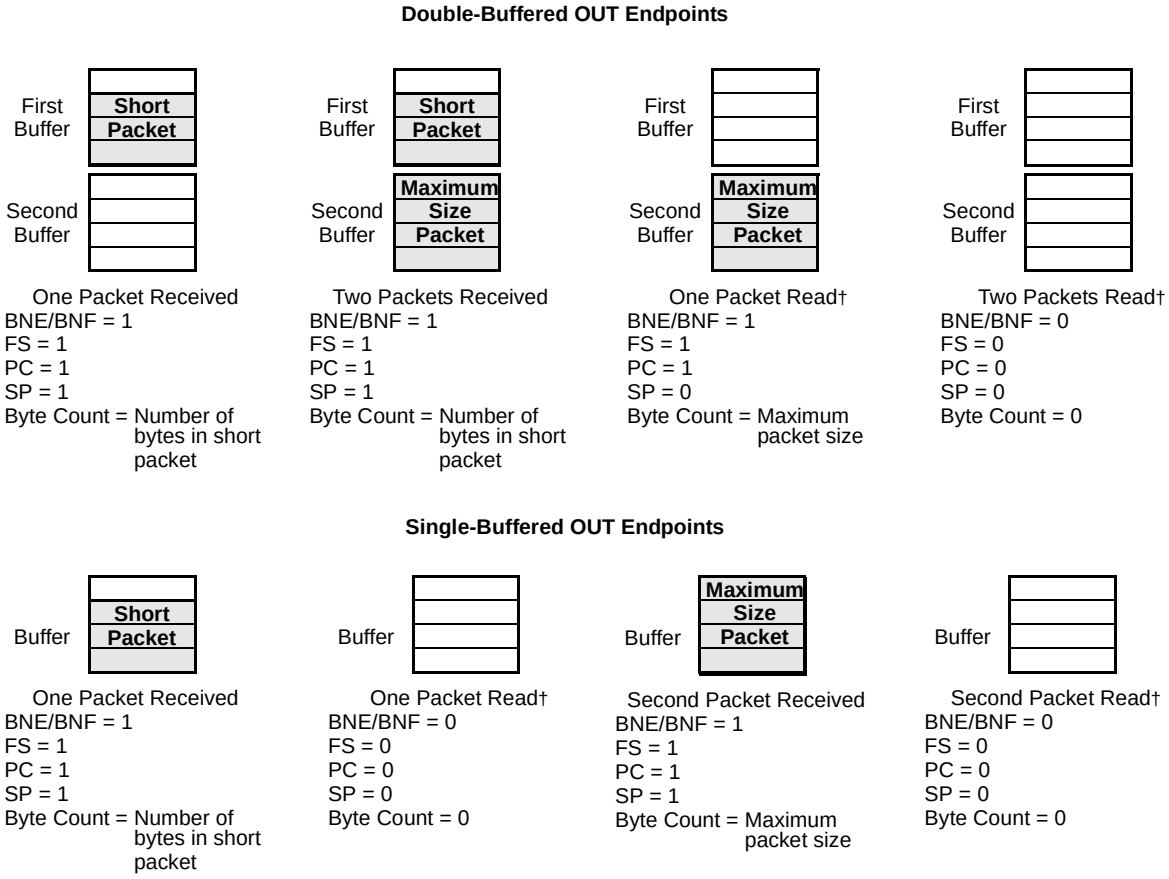
If an OUT endpoint has double-buffering enabled, the Control/Status and Byte Count registers provide the status of the endpoint buffer that is currently active. The FS, PC, and BNE/BNF bits in the Endpoint Control/Status register are set when the first buffer has received a complete data packet. The short-packet (SP) bit indicates a packet smaller than the maximum packet size has been received and is ready for unloading, or has been loaded and is ready for transmission. The Data register unloads data from the first buffer. The Control/Status and Byte Count registers continue to hold the status of the first buffer until software clears the PC bit in the Control/Status register.

As the data is read from the first receive buffer, the value in the Byte Count register is decremented and indicates the number of data bytes that still need to be read from the first buffer in the FIFO memory. When all of the data in the first buffer has been read, the Byte Count register and the BNE/BNF bit in the Control/Status register are cleared. If a second packet is received before all of the data has been read from the first buffer, the FS bit continues to be set after the first buffer is been read, but the second packet of data does not set the PC bit until after software has cleared the PC bit.

The status of the second buffer cannot be determined until the PC bit is cleared. The PC bit must be set to update the Control/Status and Byte Count registers with the status of the second buffer. If the PC bit is set before reading all of the data in the first receive buffer, the data in the first receive buffer is lost.

If the second buffer has received a complete data packet, the PC bit is again set to indicate that the endpoint FIFO has data ready to be unloaded, and the BNE/BNF and Byte Count registers indicate the amount of data present in the second buffer. At this point, the Data register unloads data from the second buffer. The Control/Status and Byte Count registers continue to hold the status of the second buffer until software again clears the PC bit. Only after all of the data has been read from the second buffer must the PC bit be set. Doing so updates the Control/Status and Byte Count registers to reflect the status of the first buffer. (See Figure 12-3 for the relationship between the data in the endpoint buffers and the status bits in the Endpoint Control/Status register.)

Figure 12-3. Status Bits for OUT Endpoints



Notes:

†After PC bit cleared by user

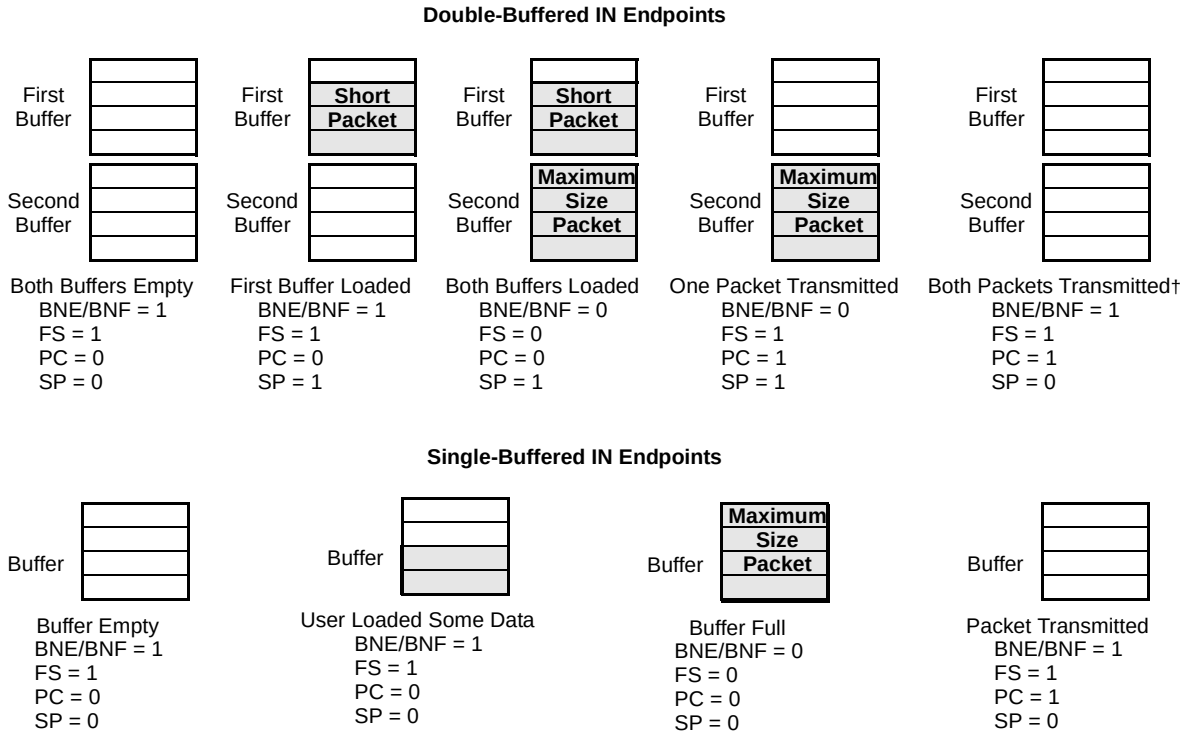
All registers are 32-bits and may not be addressed as partial bytes.

If an endpoint is configured as an IN endpoint and has double-buffering disabled, the Control/Status register provides the status of the endpoint buffer. The Byte Count register is not used for IN endpoints and is reserved. If data has been loaded into the endpoint buffer, but has not been transmitted to the USB host controller, the BNE/BNF and FS bits are cleared by the UDC when a complete packet has been loaded into the FIFO memory and is ready for transmission. To indicate a complete packet has been loaded, either write the maximum packet size to the buffer or set the Short Packet (SP) bit. When the data has been transmitted to the USB host controller, the UDC sets the PC bit to indicate the packet has been sent and the error/status bits in the Control/Status register show the values for the completed transaction. The PC bit must be cleared before loading more data for transmission.

If an IN endpoint has double-buffering enabled, the Control/Status register provides the status of the endpoint buffer that is currently active for loading. If both buffers are empty, the BNE/BNF bit is set and the FS bit is set, indicating that the FIFO is empty and will require service to be able to transmit any data to the USB host controller. Data is loaded into the first buffer by writing to the Endpoint Data register. When the current endpoint buffer has not been transmitted but has a

complete packet loaded, the UDC clears the BNE/BNF bit. The FS bit is cleared when both buffers have been loaded and are waiting to be transmitted to the USB host controller. The buffers can be loaded with a maximum packet, a zero packet, or a short packet. When one buffer of data has been transmitted to the USB host controller, the PC and FS bits are set to indicate a packet has been transmitted and there is room in the FIFO to load another packet. Software must clear the PC bit in the Control/Status register to allow both the transmission of the second loaded buffer of data and loading of the empty buffer. Figure 12-4 illustrates the relationship between data in the endpoint buffers and the status bits in the Endpoint Control/Status register.

Figure 12-4. Status Bits for IN Endpoints



Notes:

†After PC bit cleared by user.

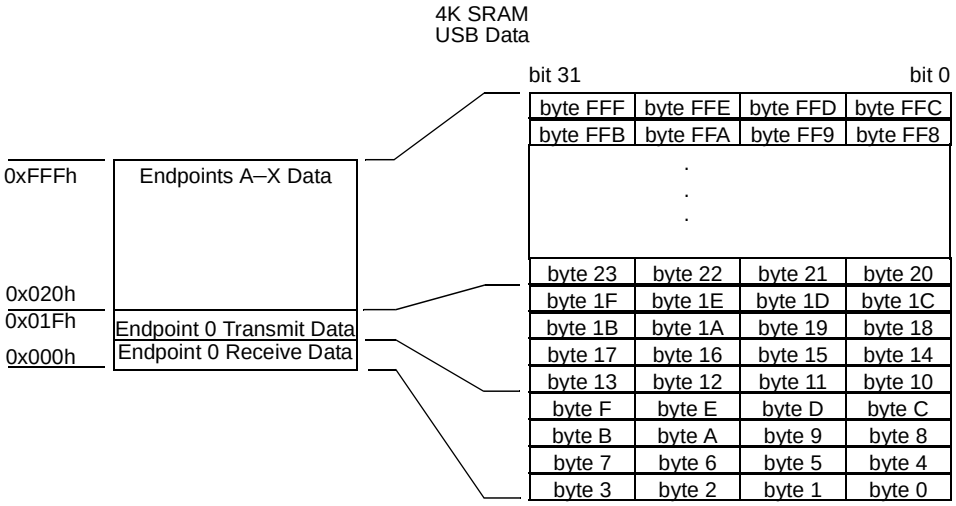
All registers are 32-bits and may not be addressed as partial bytes.

12.4.2 Endpoint Memory Configuration

The endpoint memory consists of 4 Kbytes of SRAM, arranged as a 1K x 32-bits block. The endpoint memory is used for storing USB data that has been received from the USB host controller or has been loaded for transmission to the USB host controller. 32 bytes of the endpoint memory are dedicated for endpoint 0 USB data storage. The remaining 4064 bytes can be allocated by users to endpoints A–X. The FIFO memory space is flushed and the memory reallocated when users set the Switch Endpoint Memory To Active Configuration bit UDCCR[SMAC], allowing each configuration, interface, and alternate interface setting to allocate the entire 4064 bytes of endpoint memory for use when that configuration and interface is active. The USB data stored in the

4-Kbyte SRAM is accessed through the Endpoint Data registers (UDCDR0 and UDCDRA–X) in a FIFO-type fashion and are referred to as the receive FIFO, transmit FIFO, or FIFO memory throughout this chapter. Figure 12-5 shows the unconfigured memory map of the 4-Kbyte SRAM.

Figure 12-5. Map of 4-Kbyte SRAM for USB Endpoint Data



Each programmable endpoint A–X is allocated space in the 4-Kbyte SRAM, up to approximately two times the maximum packet size for the particular endpoint type (if double-buffering is enabled). The maximum packet size and double-buffering are programmed using the Endpoint Configuration registers UDCCRA–X (see Section 12.6.11).

The endpoint memory is 32 bits wide and is allocated only on 32-bit boundaries. Although isochronous endpoints and interrupt endpoints can be programmed for any maximum packet size up to 1023 bytes and 64 bytes, respectively, if an endpoint is programmed for a maximum packet size that is not 32-bit aligned, its FIFO memory space is allocated up to the next valid 4-byte boundary.

For example, if an isochronous endpoint is configured for a maximum packet size of 317 bytes, the endpoint is allocated 320 bytes of FIFO memory space for each buffer. The additional 3 bytes of allocated FIFO memory space are not used and cannot be accessed. If this endpoint was programmed to be double-buffered, it would use a total of 640 bytes of FIFO memory space (320 bytes x 2). In addition, any packet that is less than the programmed maximum packet size is considered as occupying the entire buffer. For example, in a double-buffered OUT endpoint, a short or zero packet completely occupies one of the buffers, leaving the adjacent buffer open for loading.

When loading an endpoint with a packet that is not 32-bit aligned, the user must write all data as 32-bit accesses except for the final, partial-word write. The last word write can be written as 8-, 16- or 24-bit accesses. For example, if a 15-byte packet is to be transmitted, the user must write the first 12 bytes as three 32-bit accesses. The last 3 bytes can be written as either 3 byte writes, 1 byte write and 1 half-word write or as one 24-bit access (if being loaded by the DMA). No other accesses to the endpoint memory may be less than 32-bit accesses. All reads must be 32-bit accesses.

Table 12-3 shows the endpoints that can be used for each type, the direction that can be programmed, and the maximum packet sizes available. The bulk, isochronous, and interrupt endpoints can be programmed to be double-buffered so one packet can be processed while the next is being assembled. If an IN endpoint is double-buffered, users can be loading the endpoint data

into the second transmit buffer while transmitting from the first. Likewise, when unloading an OUT endpoint, the UDC can continue to process the next incoming packet to that endpoint. Endpoint 0 has 32 bytes of FIFO memory space, 16 bytes for IN data and 16 bytes for OUT data, and is not double-buffered.

Table 12-3. Example Endpoint Configuration

Endpoint Type	UDC Endpoint	Endpoint Direction	Maximum Packet Size (bytes)	Allocated Memory if Single Buffer Enabled (bytes)	Allocated Memory if Double Buffer Enabled (bytes)
Control	0	IN/OUT	16	32 (16 for IN, 16 for OUT)	32 (16 for IN, 16 for OUT)
Bulk	A–X	IN or OUT	8, 16, 32, or 64	8, 16, 32, or 64	16, 32, 64, or 128
Isochronous	A–X	IN or OUT	1–1023	4–1024	8–2048
Interrupt	A–X	IN or OUT	1–64	4–64	8–128

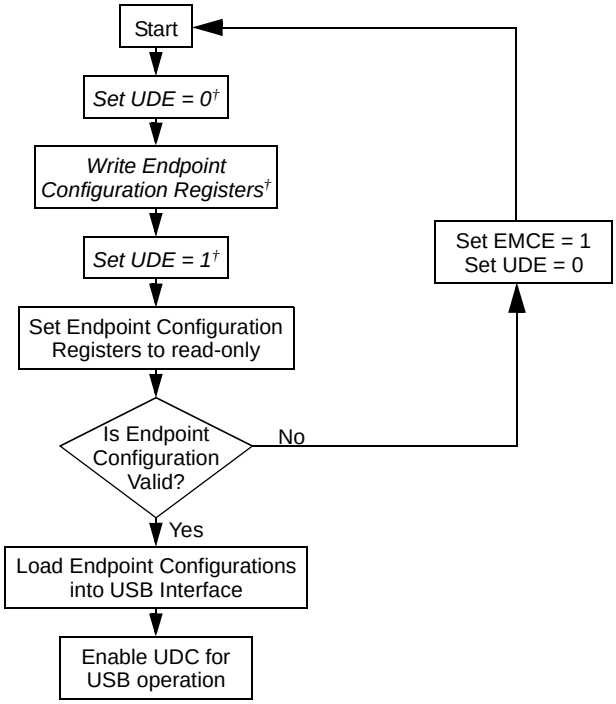
Programmable endpoints A–X can be selected as bulk, isochronous, or interrupt endpoints.

- All *bulk* endpoints have maximum packet sizes of 8, 16, 32, or 64 bytes, and if double-buffered, would result in 16, 32, 64, or 128 bytes of allocated FIFO memory space.
- All *isochronous* endpoints have maximum packet sizes from one to 1023 bytes, and if double-buffered, would result in allocated FIFO memory space of eight to 2048 bytes.
- All *interrupt* endpoints have maximum packet sizes of one to 64 bytes, and if double-buffered, would result in eight to 128 bytes of allocated FIFO memory space.

The total number of bytes of FIFO memory space allocated to all enabled endpoints for each configuration and interface cannot exceed 4064 bytes.

If more than 4064 bytes of FIFO memory space is allocated to a single configuration and interface, when UDCCR[UDE] is set, the endpoint configuration is checked and the error is detected, the endpoint configuration is not loaded, the UDC is not enabled for USB operation, the endpoint memory configuration error bit UDCCR[EMCE] bit is set, and UDCCR[UDE] is cleared by the UDC. Figure 12-6 shows the sequence for allocating the FIFO memory, setting endpoint configuration, and enabling the UDC for USB operation. Actions required by users are *italicized*.

Figure 12-6. FIFO Memory and Endpoint Configuration Sequence



†Actions required by user.

If the DMA loads and unloads the endpoint memory, the DMA channel must be programmed with the length set to the maximum packet size and the width set to word (four bytes). If the PXA27x processor loads and unloads the endpoint memory, it must be accessed as 32-bit read or 32-bit write. When using the PXA27x processor to load data for transmitting, four bytes must be loaded at a time; also, when loading data for transmitting, four bytes at a time must be loaded, unless the packet size is not 4-byte aligned; if so, the final word write can be less than 32-bits in length.

If an IN endpoint maximum packet size is not 32-bit aligned, the memory space in the FIFO that is allocated to an endpoint but outside of the space needed for the maximum packet size is unused. Similarly, when the processor unloads receive data, four bytes at a time must be unloaded. The Endpoint Byte Count register indicates the number of bytes to be read from the FIFO memory space, and decrements by four until the last read occurs.

If an OUT endpoint maximum packet size is not 32-bit aligned, the 32-bit read of the FIFO memory space allocated to the endpoint, but outside of the maximum packet size, reads as unknown and the Byte Count register decrements by the number of valid bytes read. As an example, Figure 12-7 shows the unloading of FIFO memory space by reading the Endpoint Data register, and the decrementing of the Byte Count register for an OUT endpoint with a maximum packet size of 13 bytes. Figure 12-8 shows the loading of FIFO memory space by writing the Endpoint Data register for an IN endpoint with a maximum packet size of 13 bytes.

Figure 12-7. Example Data Ordering and FIFO Organization for OUT Endpoints

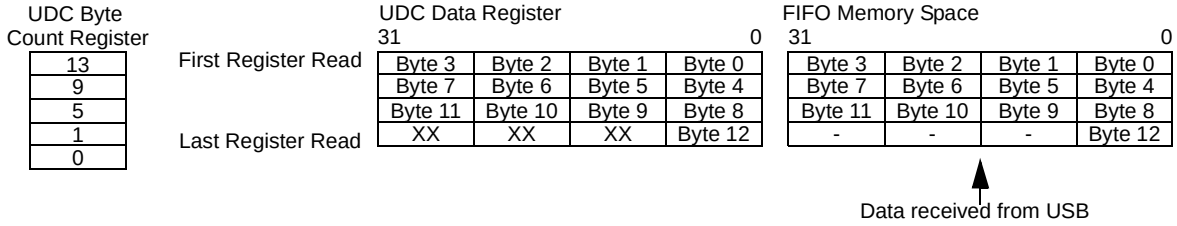
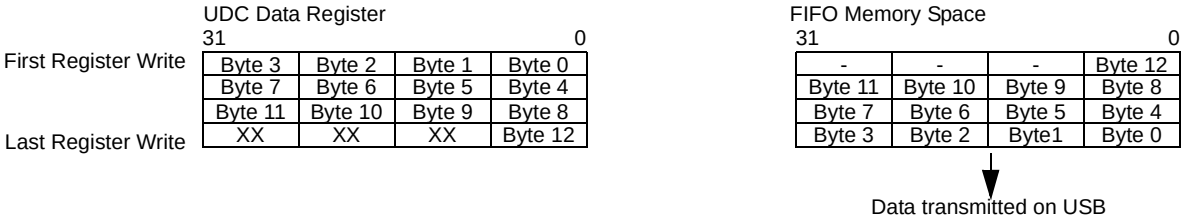
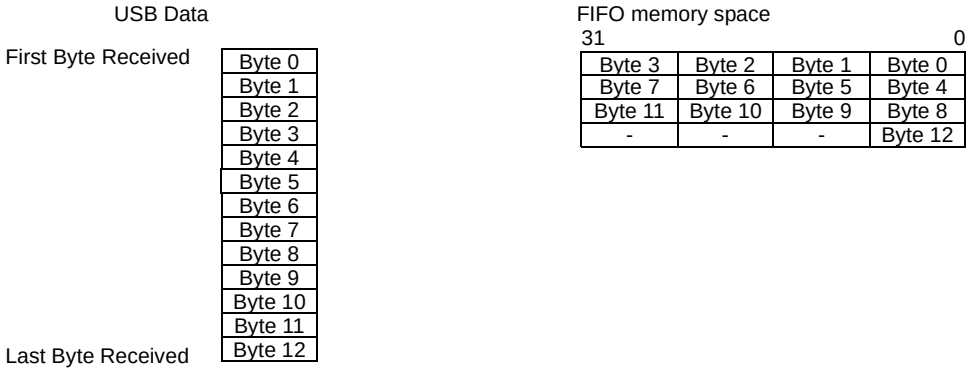


Figure 12-8. Example Data Ordering and FIFO Organization for IN Endpoints



The endpoint memory is little endian with the first byte of data received from the USB being stored in byte 0 (bits 7–0). Figure 12-9 shows an example of how the data received on the USB is organized in the endpoint memory. In the example, 13 bytes of data are received, the receive data is not 32-bit aligned, and 3 bytes of the data in the last register are read as unknown.

Figure 12-9. Example Data Ordering in Endpoint Memory



If the USB host controller requests IN data and no packet is loaded in the endpoint FIFO memory, several things occur:

- The Transmit/Receive NAK (TRN) bit in the Endpoint Control/Status register, UDCCSRx[TRN], is set.
- If the endpoint is configured as a bulk or interrupt endpoint, the UDC issues a NAK handshake to the host controller.

- If the endpoint is configured as an isochronous endpoint, the isochronous data packet is dropped and the FIFO error interrupt for the affected endpoint is generated, if enabled.

If the USB host controller transmits OUT data while the endpoint buffer is full:

- The Transmit/Receive NAK (TRN) bit in the Endpoint Control/Status register, UDCCSRx[TRN] is set.
- If the endpoint is configured as a bulk or interrupt endpoint, the UDC sends a NAK handshake to the host.
- If the endpoint is configured as an isochronous endpoint, the UDC sends a zero-size packet and if enabled, the FIFO error interrupt for the affected endpoint is generated.

If the USB host controller transmits more OUT data than the maximum size packet for a bulk or interrupt endpoint, the UDC does not send any handshake to the host controller causing the host controller to time-out. If the USB host controller transmits more OUT data than the maximum size packet for an isochronous endpoint, the UDC sets the data packet error (DPE) bit in the Endpoint Control/Status register, UDCCSRx[DPE].

12.4.2.1 DMA Access

The processor's DMA controller can be used to load and unload the endpoint memory. When DMA loads or unloads a particular endpoint memory, the DMA and the Endpoint Control and Configuration registers must be set to the correct values to ensure proper operation. The Endpoint Control registers must have the DMA enable (DME) bit set to enable the DMA request for the endpoint. The DMA channel must be programmed with the width set to word (four bytes). The DMA descriptor must be set to access data from the correct address as required by the current endpoint memory configuration. When the UDC and DMA are set for DMA loading and unloading of endpoint memory, it is possible for the UDC work without causing any interrupts.

Note: Disable the DMA-enable (DME) bit only when the DMA channel is stopped.

When an OUT endpoint that has the DME bit set receives a maximum size packet or a short packet (except a zero size packet) from the USB host controller, the UDC asserts a DMA request for that endpoint. The DMA receives the request and responds to the request by unloading the data from the endpoint buffer. When the DMA controller has responded to the request, the UDC negates the DMA request. The DMA controller continues to read the data until either the maximum packet size or the end of descriptor has been reached (see Figure 12-10 for a DMA descriptor for OUT endpoint FIFO servicing). If the DMA controller stops reading data from the endpoint buffer because it reaches the end of the descriptor, more data may still be in the endpoint buffer. In this case, the processor must read any data remaining in the endpoint buffer, or the data might be lost.

If the Short Packet (SP) bit in the Endpoint Control register is not set, the Packet Complete (PC) bit in the Endpoint Control register is cleared when the DMA controller unloads all of the data from the current endpoint buffer. If the SP bit in the Endpoint Control register is set, indicating the packet in the endpoint buffer is a short packet, the UDC asserts a DMA request for the endpoint, and the DMA controller receives the request and responds to the request by unloading the data from the endpoint buffer. When all of the data in the endpoint buffer has been unloaded, the UDC generates an interrupt to the processor, indicating that the SP bit needs to be cleared. The processor must clear the PC bit to enable the endpoint buffer to receive more data. The SP bit is automatically cleared when PC is cleared.

If the SP bit in the Endpoint Control register is set, indicating the packet in the endpoint buffer is a short packet and no data was received with the packet (a zero-size packet), the UDC does not assert a DMA request for the endpoint, but generates an interrupt to the processor indicating that the SP bit needs to be cleared. The processor must clear the SP and PC bits to enable the endpoint buffer to receive more data.

Figure 12-10. DMA Descriptors for OUT Endpoint FIFO Servicing

DSCRx.StopIrqEn = 1

Data Transferring Descriptor
- desc[0].ddadr = Descriptor Address = Second Descriptor
- desc[0].dsadr = UDCDRx
- desc[0].dtadr = Internal or External memory
- desc[0].dcmds = IncTrgAddr = 1, FlowSrc = 1, Size = user selected, Width = 3, Len = transfer length (may be multiple packets)

When an IN endpoint that has the DME bit set has an empty buffer, the UDC asserts a DMA request for that endpoint. The DMA receives the request and responds to the request by loading data into the endpoint memory. When the DMA has responded to the request, the UDC negates the DMA request. The DMA must load either a maximum size packet or load a short packet, and set the SP bit in the Endpoint Control register to enable the packet for transmission. The DMA descriptor must be programmed to write the SP bit when needed. Figure 12-11 illustrates the use of DMA chained descriptors to load multiple packets of USB data for transmission to the USB host controller. The first descriptor loads a maximum size packet, while the second descriptor sets the SP bit. Note that for IN endpoints, the descriptor length must be set to the maximum packet size and multiple descriptors used to send multiple packets except for the case where the maximum packet size is a multiple of the DMA burst size. In the case where the maximum packet size is a multiple of the DMA burst size, the descriptor length can be set to the entire transfer length. For more information on setting the DMA channel registers and the DMA descriptors, refer to Chapter 5, “DMA Controller”.

Figure 12-11. DMA Descriptors for IN Endpoint FIFO Servicing

DSCRx.StopIrqEn = 1

First Descriptor

Data Transferring Descriptor
- desc[0].ddadr = Descriptor Address = Second Descriptor
- desc[0].dsadr = UDCDRx
- desc[0].dtadr = Internal or External memory
- desc[0].dcmds = IncTrgAddr = 1, FlowSrc = 1, Size = user selected, Width = 3, Len = maximum packet size

Second Descriptor

SP Bit Setting Descriptor for the last packet
- desc[0].ddadr = Stop = 1
- desc[0].dsadr = Internal or External Memory location with Data = 0x00000080
- desc[0].dtadr = UDCCSRx

12.4.2.2 USB Configurations and Interfaces

Figure 12-12 shows the configurations, interfaces, and alternate interface settings that are available in the UDC. Configuration 0 has a fixed FIFO memory-space allocation of 32 bytes used for endpoint 0 and is the only configuration available on the USB until the UDC has been enumerated by the USB host controller. The endpoint memory allocations for configurations 1, 2, and 3 are programmable.

Figure 12-12. Configurations, Interfaces, and Alternate Interface Settings for UDC

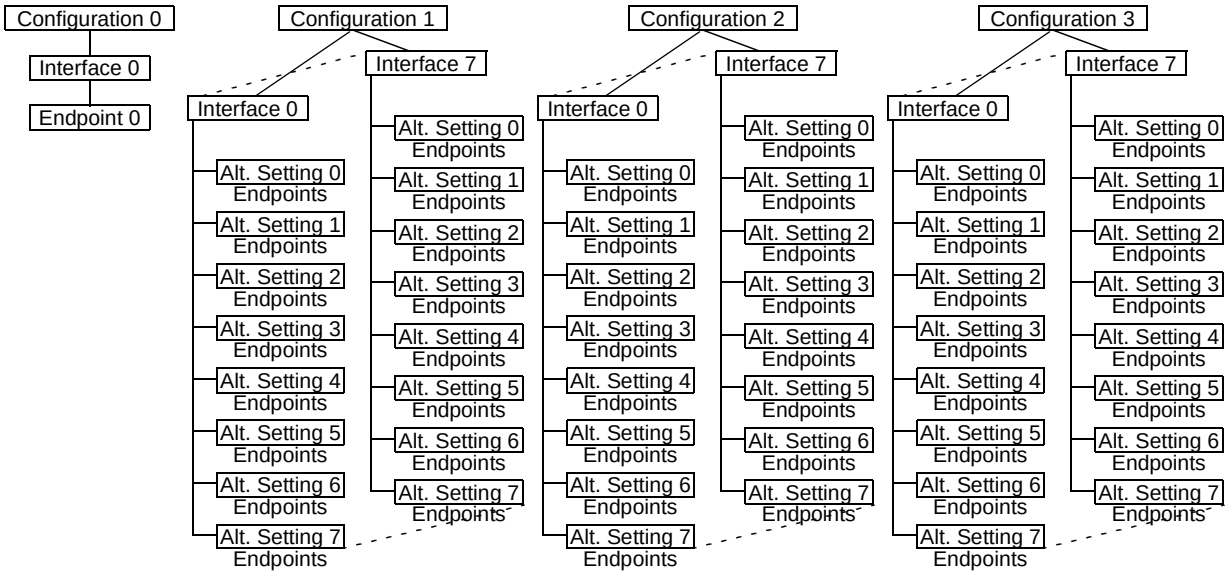
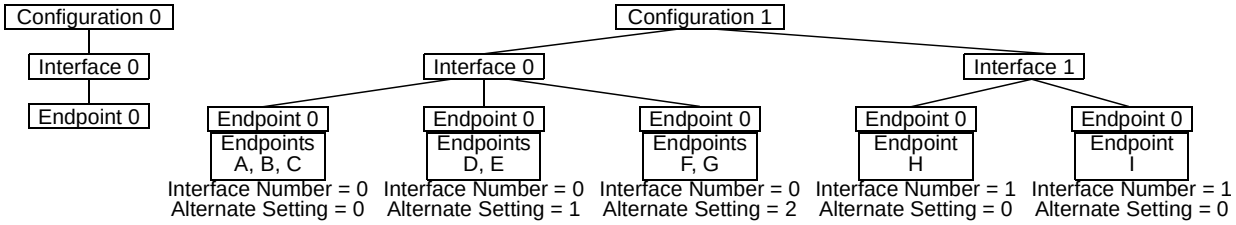


Figure 12-13 shows an example of two configurations (0 and 1), with configuration 1 having two interfaces and a total of five alternate interface settings. Each alternate interface setting is assigned a unique set of endpoints from endpoints A–X. Each programmable endpoint A–X can be assigned only one configuration, interface, and alternate interface setting. These endpoints can be assigned any USB endpoint number, type, direction, and maximum packet size with the total allocated FIFO memory space for each configuration, interface, and alternate interface setting not exceeding 4064 bytes.

Figure 12-13. Example of Two UDC Configurations



When the USB host controller executes a SET_CONFIGURATION or SET_INTERFACE command, and if the configuration change interrupt is enabled, an interrupt is generated. Any data still in the endpoint memory must be unloaded, and UDCCR[SMAC] must be set to flush the

endpoint memory and reallocate the memory according to the new configuration, interface, and alternate interface setting. Any data remaining in the endpoint FIFO is lost when the endpoint memory is flushed. The UDCCR[SMAC] is cleared by the UDC after the endpoint memory reallocation has completed. The UDC completes the status stage of the SET_CONFIGURATION and SET_INTERFACE commands regardless of the UDCCR[SMAC] value.

Table 12-4 shows an example of the memory allocated for endpoints defined by a configuration, interface, and alternate interface setting. In this example, endpoint A is set to an isochronous OUT endpoint, programmed to be endpoint number 1, and double-buffered with a maximum packet size of 1023 bytes. Endpoint B is programmed to be a bulk OUT endpoint without double buffering, assigned endpoint number 2, and a maximum packet size of 64 bytes. Endpoint C is programmed to be a bulk IN endpoint with double-buffering enabled, using endpoint number 3, and a maximum packet size of 32 bytes. Endpoint D is programmed to be an isochronous IN endpoint with an endpoint number 4, a maximum packet size of 387 bytes, and double buffering enabled. Finally, endpoint E is programmed to be an interrupt IN endpoint with an endpoint number 5 and a maximum packet size of 16 bytes. Notice that this configuration has only 6 of the 23 programmable endpoints enabled, but uses 2968 bytes of the 4064 bytes available.

Table 12-4. Maximum Packet Size Example

Endpoint	Endpoint Number	Endpoint Type	Endpoint Direction	Double-Buffering Enabled	MPS (Bytes)	Allocated FIFO Space (Bytes)
A	1	Isochronous	OUT	Yes	1023	2048
B	2	Bulk	OUT	No	64	64
C	3	Bulk	IN	Yes	32	64
D	4	Isochronous	IN	Yes	387	776
E	5	Interrupt	IN	No	16	16
TOTAL						2968

12.4.2.3 Example Endpoint Configuration

The programmable endpoints A–X can be configured to respond as any USB endpoint number 1–15, any type (isochronous, interrupt, or bulk), and either direction (IN or OUT). All programmable endpoints A–X being used must be configured before enabling the UDC for USB operation. Configuring a programmable endpoint defines the configuration and interface where the endpoint is active; the USB endpoint number, type and direction; and the maximum packet size and type of buffering to be used for storing the endpoint data. The UDC endpoint configuration can be changed after the UDC has been enabled for USB operation, but the UDC must be disabled before the Configuration registers can be rewritten. Disabling the UDC resets the USB configuration, interface, alternate interface, and address assigned to the UDC by the USB host controller, and disables all USB features enabled by the USB host controller. The UDC must be reset and re-enumerated by the USB host controller whenever the UDC is disabled and reenabled.

Figure 12-14 is an example configuration setting that could be used for the UDC. In this example, the UDC setup is defined as having two configurations: the default configuration 0 and a user programmed configuration 1. Configuration 1 is defined to have two interfaces (0 and 1), with interface 0 having two alternate interface settings (0 and 1), and interface 1 having three alternate interface settings (0, 1, and 2). Each of the alternate interface settings has a set of endpoints to implement that interface. The Endpoint Configuration registers configure each UDC endpoint A–X for its use within the configuration.

Figure 12-14. Example USB Configurations for UDC

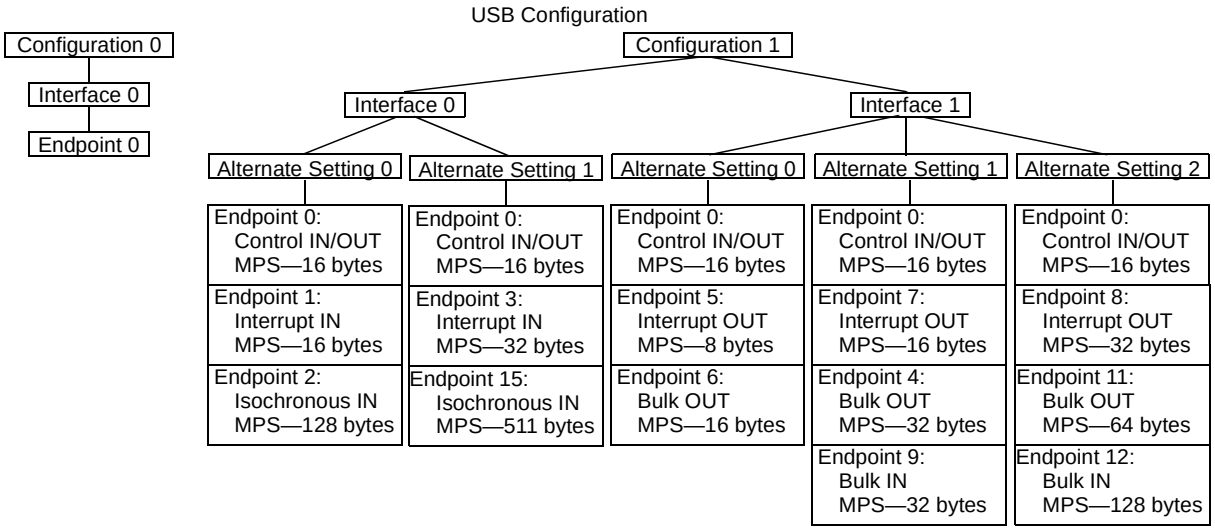


Table 12-5 lists each of the USB physical endpoints and the UDC programmable endpoint assigned to implement it. In Table 12-5, each UDC endpoint has a unique USB endpoint number. Although endpoint numbers can be duplicated across configurations, they may not be duplicated within a single configuration; even if the endpoint direction is different, the endpoint number must be unique for each endpoint within a single configuration. Endpoint 0 is the only endpoint number that is duplicated within a configuration.

Table 12-5. UDC Endpoint Configuration for Example USB Configuration (Sheet 1 of 2)

USB				UDC				
Configuration Number	Interface Number	Alternate Interface Setting	EP Number	EP	Configuration Register	Double-Buffered	Configuration Register Value	EP Memory Allocated
All	All	All	0	0	-	No	-	32
1	0	0	1	A	UDCCRA	No	0x0200_F041	16
1	0	0	2	B	UDCCRB	Yes	0x0201_3203	256
1	0	1	3	C	UDCCRC	No	0x0209_F081	32
1	0	1	15	D	UDCCRD	Yes	0x020A_F7FF	1024
1	1	0	5	E	UDCCRE	No	0x0440_F021	8
1	1	0	6	F	UDCCRF	Yes	0x0441_4043	32
1	1	1	7	G	UDCCRG	No	0x044B_F041	16
1	1	1	4	H	UDCCRH	Yes	0x044C_C083	64
1	1	1	9	I	UDCCRI	Yes	0x0449_3083	64
1	1	2	8	J	UDCCRJ	Yes	0x0650_F083	64
1	1	2	11	K	UDCCRK	Yes	0x0651_4103	128
1	1	2	12	L	UDCCRL	Yes	0x0652_3203	256
-	-	-	-	M	UDCCRM	No	0x0000_0000	0

Table 12-5. UDC Endpoint Configuration for Example USB Configuration (Sheet 2 of 2)

USB				UDC				
Configuration Number	Interface Number	Alternate Interface Setting	EP Number	EP	Configuration Register	Double-Buffered	Configuration Register Value	EP Memory Allocated
-	-	-	-	N	UDCCRN	No	0x0000_0000	0
-	-	-	-	P	UDCCRP	No	0x0000_0000	0
-	-	-	-	Q	UDCCRQ	No	0x0000_0000	0
-	-	-	-	R	UDCCRR	No	0x0000_0000	0
-	-	-	-	S	UDCCRS	No	0x0000_0000	0
-	-	-	-	T	UDCCRT	No	0x0000_0000	0
-	-	-	-	U	UDCCRU	No	0x0000_0000	0
-	-	-	-	V	UDCCRV	No	0x0000_0000	0
-	-	-	-	W	UDCCRW	No	0x0000_0000	0
-	-	-	-	X	UDCCRX	No	0x0000_0000	0

Table 12-6 shows the endpoint memory allocation for each configuration, interface, and alternate interface setting. Configuration 1, interface 0, with alternate interface setting 1 allocates the most endpoint memory space, using a total of 1088 bytes. In Table 12-5, although endpoint D has a maximum packet size of 511, each buffer is allocated 512 bytes with the endpoint using a total of 1024 bytes of memory space.

Table 12-6. Endpoint Memory Allocation by Example USB Configuration

USB			UDC		
Configuration Number	Interface Number	Alternate Interface Setting	Total EP Allocated Memory/Configuration (bytes)†	Active Endpoint	Endpoint Allocated Memory (bytes)
0	0	0	32	0	32
1	0	0	272	A	16
				B	256
1	0	1	1056	C	32
				D	1024
1	1	0	40	E	8
				F	32
1	1	1	144	G	16
				H	64
				I	64
1	1	2	448	J	64
				K	128
				L	256

NOTE:
† Configuration totals do not include the memory allocation required for endpoint 0.

The 23 endpoints can be programmed to support USB endpoints throughout the three possible configurations, eight possible interfaces, and eight possible alternate interface settings. Before loading the Configuration registers, users must clear UDE in the UDCCR to 0.

- When UDE = 0, the UDC Endpoint Configuration registers are set to allow write accesses, the USB I/O signals are three-stated, and the UDC cannot respond to USB host controller commands. Users can program the USB Endpoint Configuration registers and, when finished, can load the Configuration registers into the USB interface block by setting UDE to 1.
- When UDE = 1, the USB Configuration registers are set to read-only; the UDC memory allocation is checked and if the allocated memory space is valid, the endpoint configurations are loaded into the USB interface block, and the UDC is enabled for USB operation.

Section 12.4.3 is a C code listing for programming the UDC endpoints to implement the USB configuration shown in Figure 12-14.

12.4.3 Example Code for Configuring UDC Endpoints

```
/* define the UDC register pointers.
*/
#define udccr ((volatile unsigned long * const) 0x40600000);
#define udccra ((volatile unsigned long * const) 0x40600404);
#define udccrb ((volatile unsigned long * const) 0x40600408);
#define udccrc ((volatile unsigned long * const) 0x4060040C);
#define udccrd ((volatile unsigned long * const) 0x40600410);
#define udccre ((volatile unsigned long * const) 0x40600414);
#define udccrf ((volatile unsigned long * const) 0x40600418);
#define udccrg ((volatile unsigned long * const) 0x4060041C);
#define udccrh ((volatile unsigned long * const) 0x40600420);
#define udccri ((volatile unsigned long * const) 0x40600424);
#define udccrj ((volatile unsigned long * const) 0x40600428);
#define udccrk ((volatile unsigned long * const) 0x4060042C);
#define udccrl ((volatile unsigned long * const) 0x40600430);
#define udccrm ((volatile unsigned long * const) 0x40600434);
#define udccrn ((volatile unsigned long * const) 0x40600438);
#define udccrp ((volatile unsigned long * const) 0x4060043C);
#define udccrq ((volatile unsigned long * const) 0x40600440);
#define udccrr ((volatile unsigned long * const) 0x40600444);
#define udccrs ((volatile unsigned long * const) 0x40600448);
#define udccrt ((volatile unsigned long * const) 0x4060044C);
#define udccru ((volatile unsigned long * const) 0x40600450);
#define udccrv ((volatile unsigned long * const) 0x40600454);
#define udccrw ((volatile unsigned long * const) 0x40600458);
#define udccrx ((volatile unsigned long * const) 0x4060045C);

/* set UDE = 0 disable UDC and make Endpoint configuration registers read-write. */
*udccr = 0x00000000;
/* PROGRAM THE ENDPOINT CONFIGURATION REGISTERS. */
/*EPA: config 1, interface 0, alternate interface setting 0, Endpoint 1, Interrupt
in, MPS = 16, DB = off.  */
*udccra = 0x0200f041;
/*EPB: config 1, interface 0, alternate interface setting 0, endpoint 2,
```

```
isochronous in, MPS = 128, DB = on.*/
*udccrb = 0x02013203;
/*EPC: config 1, interface 0, alternate interface setting 1, endpoint 1, Interrupt
in, MPS = 32, DB = off.  */
*udccrc = 0x0209f081;
/*EPD: config 1, interface 0, alternate interface setting 1, endpoint 2,
isochronous IN, MPS = 511, DB = on.*/
*udccrd = 0x020af7ff;
/*EPE: config 1, interface 1, alternate interface setting 0, endpoint 1, Interrupt
out, MPS = 8, DB = off.  */
*udccre = 0x0440f021;
/*EPF: config 1, interface 1, alternate interface setting 0, endpoint 2, bulk 0,
MPS = 16, DB = on.         */
*udccrf = 0x04414043;
/*EPG: config 1, interface 1, alternate interface setting 1, endpoint 1, Interrupt
out, MPS = 16, DB = off. */
*udccrg = 0x044bf041;
/*EPH: config 1, interface 1, alternate interface setting 1, endpoint 2, bulk out,
MPS = 32, DB = on.       */
*udccrh = 0x044C4083;
/*EPI: config 1, interface 1, alternate interface setting 1, endpoint 3, bulk in,
MPS = 32, DB = on.        */
*udccri = 0x04493083;
/*EPJ: config 1, interface 1, alternate interface setting 2, endpoint 1, Interrupt
out, MPS = 32, DB = on.  */
*udccrj = 0x0650f083;
/*EPK: config 1, interface 1, alternate interface setting 2, endpoint 2, bulk out,
MPS = 64, DB = on.       */
*udccrk = 0x06514103;
/*EPL: config 1, interface 1, alternate interface setting 2, endpoint 3, bulk in,
MPS = 128, DB = on.       */
*udccrl = 0x06523203;
/* set UDE = 1 to load configuration registers and enable UDC for USB operation. */
*udccr = 0x00000001;
```

12.4.4 UDC Device Requests

The UDC Endpoint Control, Status, and Data registers control and monitor the transmit and receive FIFOs for all UDC endpoints. All other UDC configuration and status reporting is controlled by the USB host controller with USB device requests. Device requests are sent as control transfers to endpoint 0. Each control-transfer setup packet to endpoint 0 is eight bytes long and specifies:

- Data transfer direction: USB host controller to UDC, UDC to USB host controller
- Data transfer type: standard, class, vendor
- Data recipient: device, interface, endpoint, other
- Number of bytes to transfer
- Index or offset
- Value: passes a variable-sized data parameter

- Type of request

Table 12-7 is a summary of all standard device requests. Refer to the *Universal Serial Bus Specification Revision 1.1* for a full description of device requests.

Table 12-7. Device Request Summary

Request	Name
SET_FEATURE	Enables a specific feature such as device remote wake-up and endpoint stalls.
CLEAR_FEATURE	Clears or disables a specific feature.
SET_CONFIGURATION	Configures the UDC for operation. Used following a reset of the UDC or after a reset has been signaled by the USB host.
GET_CONFIGURATION	Returns the current UDC configuration to the host.
SET_DESCRIPTOR	Sets existing descriptors or add new descriptors. Existing descriptors include: device, configuration and string
GET_DESCRIPTOR	Returns the specified descriptor if it exists.
SET_INTERFACE	Selects an alternate interface setting for the UDC's interface.
GET_INTERFACE	Returns the selected alternate interface setting for the specified interface.
GET_STATUS	Returns the UDC's status including: remote wake-up, self-powered, data direction, endpoint number, and stall status.
SET_ADDRESS	Sets the UDC's 7-bit address value for all future device accesses.
SYNCH_FRAME	Sets and then report an endpoint's synchronization frame.

The UDC decodes and responds to all but two of the standard-device requests with no intervention required by users. In the case of standard device requests SET_ADDRESS, SET_FEATURE, and CLEAR_FEATURE, the UDC accepts the data and updates the appropriate internal UDC registers. The data for these device requests is not forwarded on to users through the endpoint FIFO memory.

In the case of GET_CONFIGURATION, GET_INTERFACE, and GET_STATUS, the UDC sends the appropriate response the USB host controller with no user intervention and the device request is not forwarded on to the users through the endpoint FIFO memory.

GET_DESCIPTOR, SET_DESCRIPTOR, and SYNCH_FRAME commands are passed through the endpoint FIFO memory to the user. In response to the GET_DESCRIPTOR command, users must return a description of the UDC configuration. In response to the SYNCH_FRAME command, users must return the frame number for the requested isochronous endpoint or STALL endpoint 0 if the SYNCH_FRAME command was sent for an endpoint that is not configured as isochronous. Users must also handle vendor and class-specific requests. The status stage of the control transfer is automatically handled by the UDC and requires no user intervention.

If a SET_CONFIGURATION or SET_INTERFACE transaction is sent by the host, the UDC accepts the data and updates the appropriate internal UDC registers. The data from these device requests is not forwarded on to users through the endpoint FIFO memory. The configuration and interface numbers are stored in the UDCCR. If the ACM bit in the UDCCSR0 is set, the UDC sends a NAK packet in the handshake phase of the SETUP transaction until the user has set AREN. Once AREN is 1, the UDC sends an ACK packet for the handshake from the UDC to the host. If the ACM bit is clear, the UDC sends the appropriate response to the USB host controller with no user intervention.

When the USB host controller sends a SET_CONFIGURATION or SET_INTERFACE command, if the interrupt-enable bit for configuration change is set, the UDC sends an interrupt to the core. Users must unload any data in the endpoint memory and set SMAC to 1 to flush the endpoint memory and enable the UDC to allocate the endpoint memory to the endpoints that are active in the new configuration or interface selected by the USB host controller. After a

SET_CONFIGURATION or SET_INTERFACE command has been sent by the USB host controller, the UDC does not receive or transmit any bulk, interrupt, or isochronous data until the user sets SMAC to 1. Once the UDC has allocated the endpoint memory for the new configuration, the UDC is again ready for USB operation.

When users set UDE to 0, the configuration, interface, address, and features assigned or enabled by the USB host controller to the UDC are reset. The UDC must be reset and re-enumerated by the USB host controller after UDE has been cleared.

When the USB host controller sends a SET_FEATURE command to enable the DEVICE_REMOTE_WAKEUP feature, the Device Remote Wake-up Enable (DRWE) Status bit in the UDCCR is set to indicate the feature has been enabled. When the USB host controller sends a CLEAR_FEATURE command to disable the DEVICE_REMOTE_WAKEUP feature, the DRWE bit is cleared to indicate the feature has been disabled by the host. The user can read the UDCCR at any time to determine if the DEVICE_REMOTE_WAKEUP feature has been enabled by the USB host controller.

If the UDC is connected to an On-The-Go USB host controller and the USB host controller sends a SET_FEATURE command to enable the On-The-Go specific features, the UDC does not decode the command and responds with a STALL unless the On-The-Go Enable (OEN) bit in the UDCCR is set. When the USB host controller sends a SET_FEATURE command to enable On-The-Go specific features and the OEN bit is set, the On-The-Go status bits in the UDCCR are set to indicate the feature has ben enabled. For more information on On-The-Go operation, see Section 12.5.

12.4.5 Configuration

The UDC can physically support more data-channel bandwidth than the USB allows. When responding to the USB host controller, users must specify a legal USB configuration. For example, if users specify a configuration of six isochronous endpoints of 256 bytes each, the USB host controller is not able to schedule the proper bandwidth and does not take the UDC out of configuration 0. Users must determine which endpoints to enable for each configuration and interface to ensure that the total number of endpoints having allocated FIFO memory space does not exceed the memory space available and that the total maximum packet sizes are valid USB configurations.

Note: Enabled isochronous endpoints must not exceed the bandwidth available for isochronous transfers per USB frame.

Following a reset of the PXA27x processor and prior to enabling UDC for USB operation, users must configure and enable all of the endpoints that are to be used for each USB configuration and interface. The USB interface block is held in reset until the UDC is enabled. While the USB interface block is in reset, the UDC I/O signals (USBC_P and USBC_N) are three-stated and unable to receive any USB host controller commands. Users must program the Endpoint Configuration registers, and then set UDE to 1.

After the UDC is enabled, the endpoint control block checks the programmed endpoint memory allocations and, if valid, loads the configuration for all enabled endpoints into the USB interface, and then enables the UDC I/O signals for transmission, preparing the UDC for USB operation. The USB host controller must issue a USB reset to the UDC to reset the device. After the USB host controller issues a USB reset, the UDC enables endpoint 0, and initializes the USB interface block to respond to default address zero. At this point, the UDC is under host control and responds to its commands that are transmitted to endpoint 0 using control transactions. The USB host controller

then enumerates the UDC and assigns the UDC a unique address. Refer to Section 12.4.2 for more details on configuring endpoints, and Section 12.6.11 for details on the format and fields of the Endpoint Configuration registers.

Note: The UDC does not check the programmed endpoint configuration, interface, alternate interface, and endpoint numbers for proper operation before loading. Unpredictable behavior may occur if the endpoint number or configuration, interface, and alternate interface settings are not programmed correctly.

12.4.6 Cable Attach and Detach

To be compliant with *Universal Serial Bus Specification, Revision 1.1*, the USB host controller provides +5 V on the USB cable. Since the PXA27x processor pins are not 5-V tolerant, the power signal must be gated by an external level-shifting device, and the output routed to a GPIO pin, which sends an interrupt to software.

- When the GPIO indicates an attach event, software must enable the UDC by first setting the Control and Configuration registers and then setting UDCCR[UDE].
- When a detach event is detected, unload all data from the endpoint memory first and then disable the UDC by clearing UDCCR[UDE].

12.4.7 Suspend and Resume

If idle persists on the USB for more than 3 ms, the UDC detects the suspend state, and (if the suspend interrupt is enabled) an interrupt is sent to the processor. When the UDC enters the suspend state, the processor stops the 48-MHz clock to the UDC and enables the UDC pins USBC_P and USBC_N to detect the resume state. If the processor does not enter sleep mode, the state of the UDC is preserved and is ready for resume detection.

Note: The presence of SOF packets prevents the UDC from entering suspend mode.

The UDC can exit suspend in three ways:

- Resume initiated by the UDC
- Resume initiated by the USB host controller
- USB reset

If the USB host controller has executed the SET_FEATURE command and enabled the device remote wake-up feature of the UDC, after the UDC has entered the suspend state, sending a wake-up signal to the USB host controller is performed by setting UDCCR[UDR]. Doing so forces the UDC to drive a non-idle state (K state) onto the USB for 3 milliseconds without further user intervention. The UDC hardware then clears UDCCR[UDR]. The UDC waits for the resume signal to be reflected back to it by the USB host controller, and when the resume state is detected on the USB, if the resume interrupt is enabled, an interrupt is sent to the processor.

The USB host controller can wake up the UDC by driving the non-idle state (K state) either by driving a resume or USB reset onto the USB. When the UDC pads detect this non-idle state on the USB, they signal the processor clock manager module to start the 48-MHz clock to the UDC, and (if the resume interrupt is enabled) an interrupt is sent to the processor. Software must take the appropriate action to resume activity.

For information on programming for UDC wake-up events from suspend refer to Chapter 3, “Clocks and Power Manager”.

12.4.7.1 Sleep Mode Operation

If the UDC has entered the suspend state before the PXA27x processor is in sleep mode, the UDC pins USBC_P and USBC_N are used to detect the resume state on the USB and resume operation while the processor is in sleep mode. If the USB host controller tries to access the UDC when the processor is in sleep mode, the USBC_P and USBC_N pins detect the resume state and signal the processor to begin the wake-up sequence. The UDC will have lost all state information and the UDC Configuration registers must be reloaded prior to setting UDCCR[UDE]. The USB host controller must issue a USB reset and re-enumerate the UDC (see Chapter 3, “Clocks and Power Manager” for more information on the wake-up sequence and USB operation during sleep mode).

If the UDC has been disconnected from the USB and the processor is in sleep mode, a GPIO pin must be programmed to detect connection to the USB and to signal the processor to begin the wake-up sequence. The GPIO pin must be connected through a level-shifter to the USB power signal to detect connect/disconnect to the USB. The UDC will have lost all state information and users must load the UDC Configuration register and enable the UDC before the UDC is ready for USB operation.

12.5 USB On-The-Go Operation

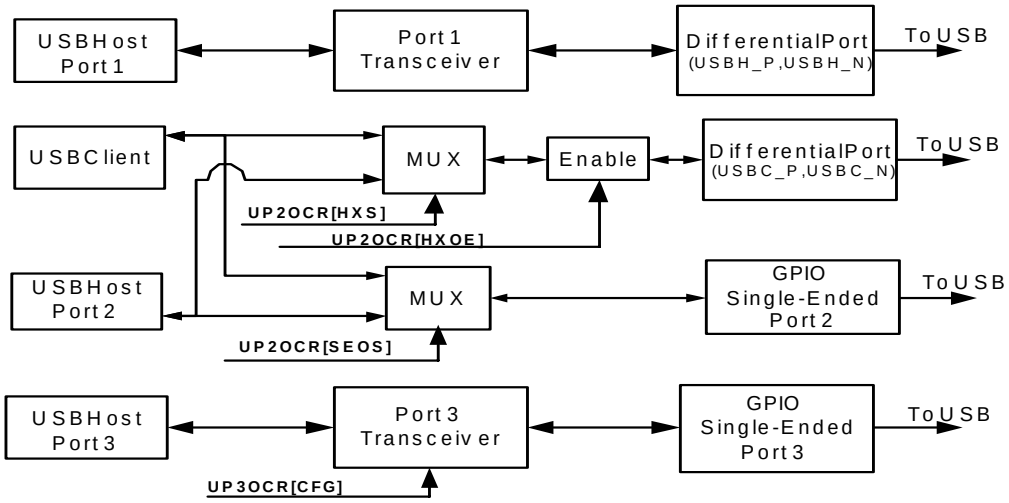
The processor USB device and host controllers can be used to provide A- and B-device On-The-Go (OTG) operation as specified in the *On-The-Go Supplement to the USB 2.0 Specification*. The on-chip OTG transceivers provide on-chip pull-up and pull-down resistors as specified in the *Pull-up/Pull-down Engineering Change Notice to the USB 2.0 Specification*. OTG operation requires user intervention but interrupts are provided to notify the user of OTG activities including Vbus changes, session detection, and OTG ID changes. The user must use these interrupts along with the OTG control and status registers to operate as an OTG device. The UDC OTG support includes the following:

- Decoding of SET_FEATURE commands with OTG specific selector values
- Control for on-chip OTG transceiver with multiplexing between UDC and USB host controller (UHC) port 2
- Control for multiplexing between UDC, UHC port 2 and UHC port 3 data through GPIOs
- Control, status and interrupt registers for interfacing to off-chip OTG transceivers
- Control, status and interrupt registers for interfacing to off-chip charge pump devices
- OTG ID support

Figure 12-15 shows each of the configurations provided to support OTG operation. Each of these is discussed in detail.

Note: The processor does not provide direct connection to or control of the USB Vbus.

Figure 12-15. USB OTG Configurations



12.5.1 UDC OTG SET_FEATURE Commands

To enable the UDC to be used in OTG and non-OTG modes, the UDC can selectively decode SET_FEATURE commands with selector values of 3, 4, or 5. If a USB host controller sends a SET_FEATURE command with a selector value of 3, 4 or 5 to enable OTG features and the On-The-Go Enable (OEN) bit in the UDC Control Register (UDCCR) is set, the UDC decodes the command, responds with an ACK on the USB, and the corresponding OTG status bit in the UDC Control Register (UDCCR) is set to indicate the feature has been enabled. If the USB host controller sends an OTG specific SET_FEATURE command and OEN is clear, the UDC will not decode the command, respond with a STALL on the USB, and will not set any OTG status bits in the UDC Control Register (UDCCR). The reset value for OEN is 0, so the user must set OEN to 1 for OTG SET_FEATURE commands to be decoded. Table 12-8 lists the On-The-Go features and the UDCCR status bits assigned to each. When OEN is set, the user can read the UDC Control Register (UDCCR) at any time to determine if the On-The-Go features have been enabled by the USB host controller.

Table 12-8. On-The-Go Feature Selectors

Feature Selector	Value	UDCCR Status Bit Name	Description
b_hnp_enable	3	BHNP	B-device enabled for host negotiation protocol
a_hnp_support	4	AHNP	B-device is connected to an A-device port that supports host negotiation protocol.
a_alt_hnp_support	5	AALTHNP	B-device is connected to an A-device port that is not capable of host negotiation protocol, but the A-device has an alternate port that is.

12.5.2 On-Chip OTG Transceiver Operation

The USB host port 2 transceiver is designed in accordance with the *Pull-up/Pull-down Resistors Engineering Change Notice to the USB 2.0 Specification* to provide on-chip resistors and OTG-compliant transceiver operation. The USB host controller port 2 multiplexor is a bidirectional I/O multiplexor that connects to the USB host port 2 transceiver and the single-ended I/O through the GPIO. The port 2 multiplexor provides an interface that allows the UDC port or UHC port 2 to connect to the UHC port 2 transceiver for direct bidirectional connection to the USB. The port 2 multiplexor also provides an interface that allows the UDC port, UHC port 2, and the UHC port 3 to connect to single-ended I/O through the GPIOs.

The OTG transceiver contains two pull-up resistors and one pull-down resistor on each D+ and D– that can be enabled using the USB Port 2 Output Control Register (UP2OCR) pull-up/pull-down enable bits (DPPUBE, DMPUBE, DPPUE, DMPUE, DPPDE, DMPDE). Figure 12-16 illustrates the on-chip host port 2 transceiver pad with the pull-up and pull-down resistors.

- Enable SW3 for both D+ and D– when host port 2 is being used for USB host controller data.
- Enable SW1 on the D+ pad and disable SW1 on the D– pad when host port 2 is being used for USB device controller data.
- Disable SW2 on the D+ and D– pads when host port 2 is being used for USB device controller data.
- SW2 is enabled and disabled by hardware when the UDC is idle and receiving data from an upstream device as specified in the *Pull-up/Pull-down Resistors Engineering Change Notice to the USB 2.0 Specification*. Table 12-9 lists the switch settings used for the USB host and USB device controller I/O.

Figure 12-16. Host Port 2 OTG Transceiver

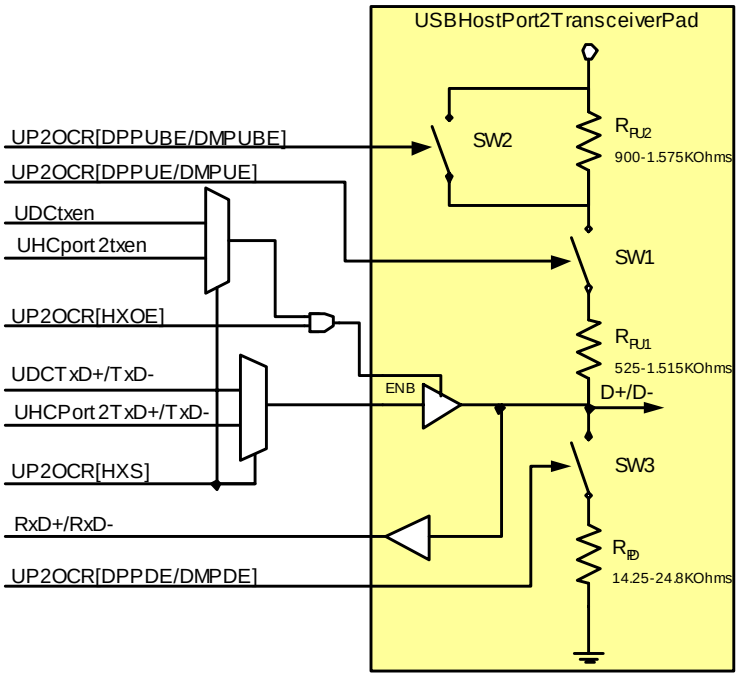


Table 12-9. Host Port 2 OTG Transceiver Switch Control Settings

Controller Selected	D+ Transceiver			D– Transceiver		
	SW1†	SW2†	SW3†	SW1†	SW2†	SW3†
USB Host	Disabled	Disabled	Enabled	Disabled	Disabled	Enabled
USB Device	Enabled	Hardware controlled	Disabled	Disabled	Disabled	Disabled

†SW1, SW2, and SW3 refer to the switches shown in Figure 12-16.

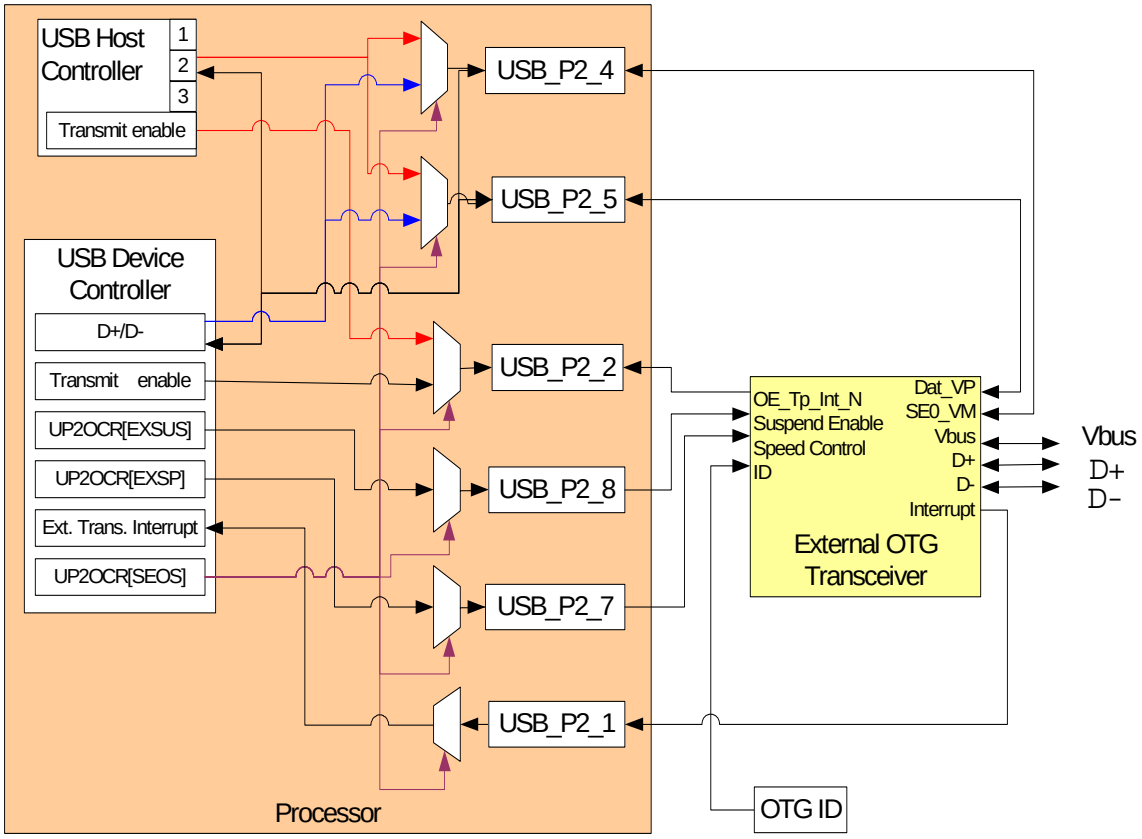
The USB host port 2 transceiver can be disabled and the output selected using the Host Port 2 Transceiver Output Enable (HXOE) and Host Port 2 Transceiver output select (HXS) bits in the USB Port 2 Output Control Register (UP2OCR), respectively. Figure 12-16 shows a schematic of the USB host port 2 transceiver as well as other control signals and their respective logic provided by control bits in the USB Port 2 Output Control Register (UP2OCR). The host port 2 OTG transceiver can be used for non-OTG operation for USB host or USB device controller USB I/O.

After reset, all of the host port 2 transfer pull-up and pull-down resistors are disabled. If the host port 2 transceiver is used in OTG mode, the user must determine the OTG ID setting and then enable the pull-up and pull-down resistors according to OTG ID. If the processor enters standby and the pull-up and/or pull-down resistors are enabled, the resistors continue to be enabled during standby mode and the USB Port 2 Output Control Register (UP2OCR) retains all programmed values during standby mode. If the processor enters sleep mode with the pull-up and/or pull-down resistors enabled, the resistors continue to be enabled during sleep mode, but the USB Port 2 Output Control Register (UP2OCR) is reset during sleep mode exit. The user must re-program all values in the USB Port 2 Output Control Register (UP2OCR) before clearing the USB PH bit in the Power Manager Sleep/Status register to 0. See Chapter 3, “Clocks and Power Manager” for more information on USB operation in sleep and standby modes.

12.5.3 Interface to External OTG Transceiver

In the case where the user does not use the internal OTG transceiver, the UDC contains control, status, and interrupt registers to provide seamless interfacing to external transceivers. External transceivers can be used to provide D+, D–, and Vbus driver to the USB. In this mode, the USB D+ and D– signals are output through GPIO pads with UP2OCR[SEOS] used to control multiplexors to select between UDC and USB host controller D+, D–, and transmit enable signals. In addition, the USB Port 2 Output Control Register (UP2OCR) provides the external transceiver suspend (EXSUS) and external transceiver speed (EXSP) control output bits, and the external transceiver interrupt input to interface to the external transceiver. Figure 12-17 illustrates the OTG connection to an external transceiver.

Figure 12-17. Connection to External OTG Transceiver

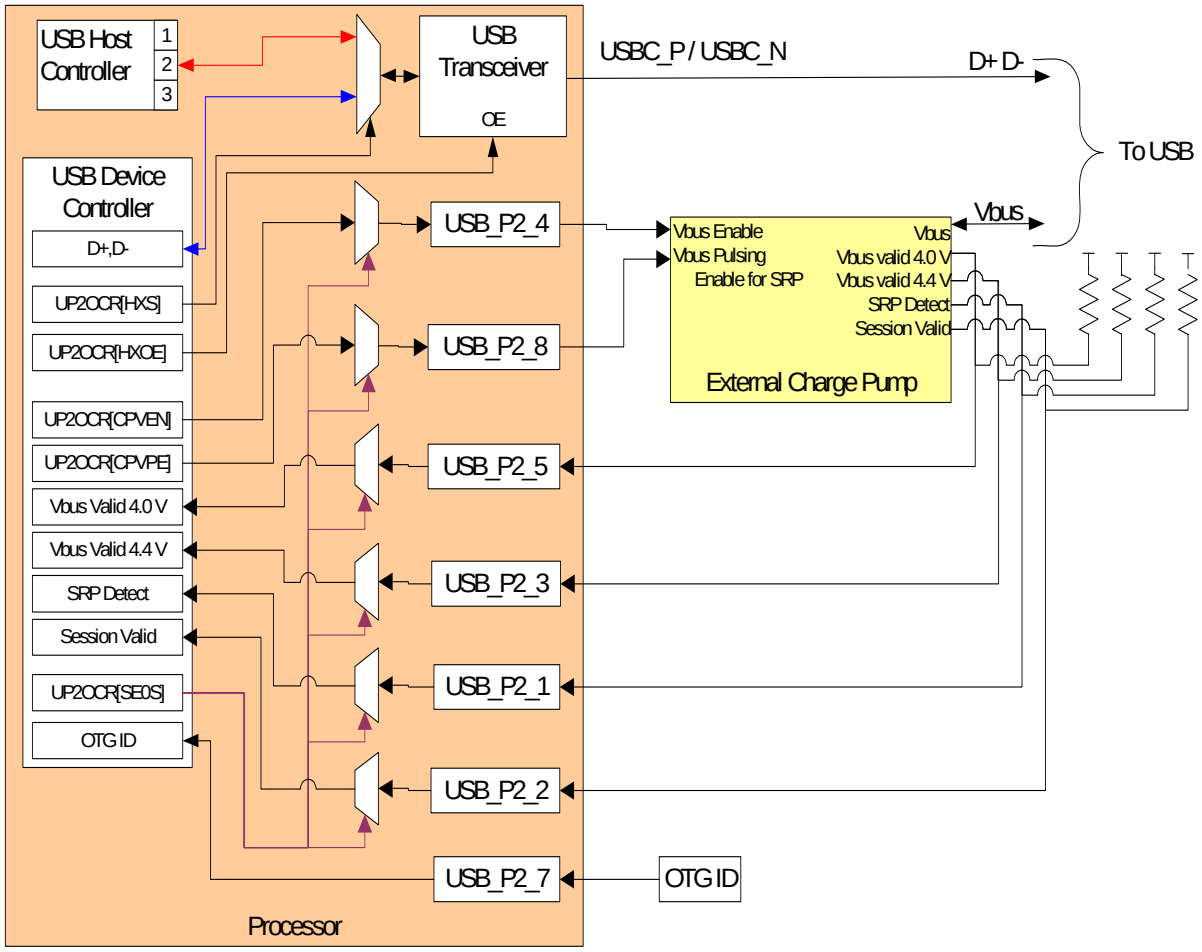


Note: For Figure 12-17, UP2OCR[SEOS] = 4 for USB client and UP2OCR[SEOS] = 5 for USB host.

12.5.4 Interface to External Charge Pump Device

In addition to the interface options described in Section 12.5.4, the UDC provides control outputs and interrupt inputs to drive and monitor an external charge pump device. To do so, the USB D+ and D– signals can be output using the on-chip OTG transceiver and the Vbus interface provided by an external charge pump device. In this mode, UP2OCR[HXS] is used control multiplexors to select between UDC and USB host controller D+, D–, and transmit enable signals to be output through the USB host controller port 2 transceiver. In addition, USB Port 2 Output Control Register (UP2OCR) provides the charge pump Vbus enable (CPVEN) and charge pump Vbus pulse enable (CPVPE) control output bits used to enable the driving of Vbus and to enable the driving of pulses on Vbus, respectively. Additionally, USB Port 2 Output Control Register (UP2OCR) provides the Vbus valid 4.0-V, Vbus valid 4.4-V, session valid, and session request protocol detected interrupt inputs to interface to the external charge pump device. Figure 12-18 illustrates the OTG connections to an external charge pump device.

Figure 12-18. Connection to External OTG Charge Pump

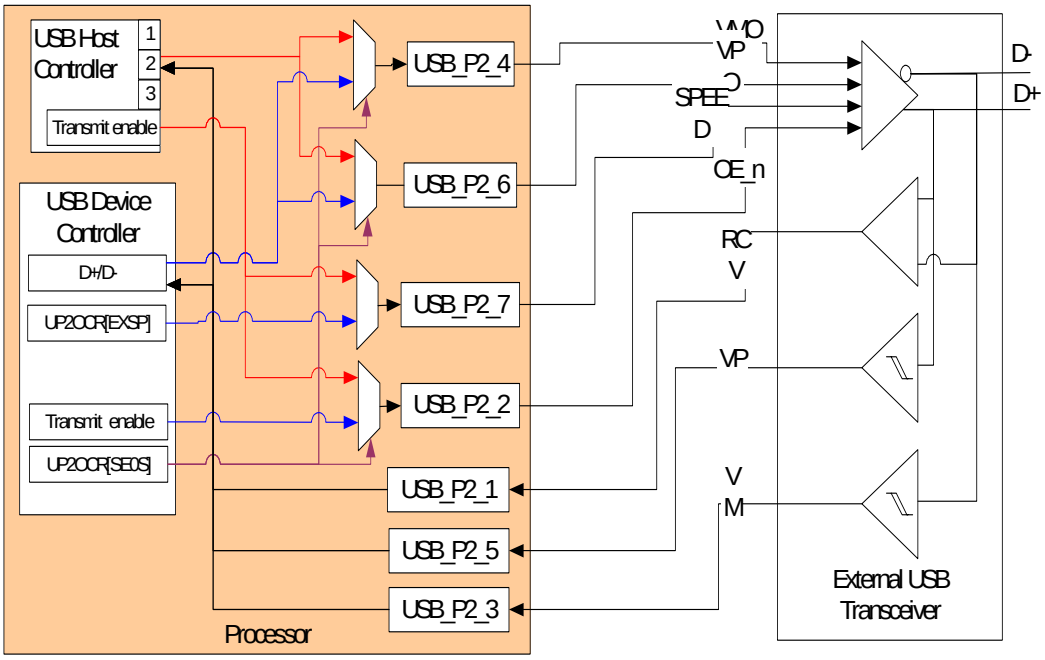


Note: For Figure 12-18, UP2OCR[SEOS] = 6 for USB client and UP2OCR[SEOS] = 7 for USB host.

12.5.5 Interface to External USB Transceiver

In addition to the OTG interfaces to external transceiver and charge pump devices, the UDC and USB host controllers can interface to a non-OTG external USB transceiver through the single-ended interface with the GPIOs. In this mode, the GPIOs provide unidirectional connections to an external transceiver and the external transceiver provides bidirectional connections for D+ and D– to the USB. This mode is selected when SEOS in the USB Port 2 Output Control Register (UP2OCR) is set to 2 or 3. Figure 12-19 illustrates the connection to an external USB transceiver.

Figure 12-19. Connection to External USB Transceiver



Note: For Figure 12-19, UP2OCR[SEOS] = 2 for USB client and UP2OCR[SEOS] = 3 for USB host.

Table 12-10 and Table 12-11 describe the definitions associated with the possible combinations of data while using the external USB transceiver mode.

Table 12-10. Output to External USB Transceiver

P2_6/P3_6	P2_4/P3_4	Result
0	0	Logic 0
0	1	Single-Ended Zero (SE0)
1	0	Logic 1
1	1	SE0

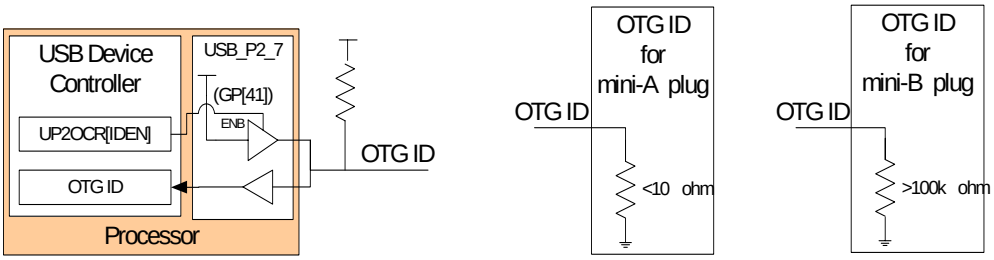
Table 12-11. Inputs from External USB Transceiver

P2_5/P3_5	P2_3/P3_3	Result
0	0	Single-Ended Zero (SE0)
0	1	Low-speed
1	0	Full-speed
1	1	Error

12.5.6 OTG ID

The UDC provides OTG ID interface support through USB_P2_7 (GPIO<41>). The UDC provides USB Port 2 Output Control Register (UP2OCR) ID output enable (IDON) to enable the output for OTG ID reading and the OTG ID interrupt input to detect changes in the OTG ID signal. When IDON is set, the output of USB_P2_7 (GPIO<41>) is enabled and driven high with a weak output driver. When the OTG ID pin on the USB connector is connected to a 100 k ohm resistor, the OTG ID input is a 1. When the OTG ID pin is connected to a 10-ohm resistor to ground, the USB_P2_7 (GPIO<41>) output driver is not able to drive OTG ID to a 1, resulting in the OTG ID input being a 0. Figure 12-20 illustrates the interface to OTG ID.

Figure 12-20. Connection to OTG ID



12.6 Register Descriptions

12.6.1 UDC Control Register (UDCCR)

The UDC Control register (UDCCR) contains control and status bits. All bits in this register are reset after a USB reset is received from the external USB host.

12.6.1.1 ACN, AIN, and AAISN

The active configuration number (ACN), active interface number (AIN), and active alternate interface setting number (AAISN) indicate (respectively) the current configuration, interface, and alternate interface setting selected by the USB host controller to be used by the UDC.

The active configuration number is set when the USB host controller issues a SET_CONFIGURATION command to the UDC. The active interface and active alternate interface setting numbers are set when the USB host controller issues a SET_INTERFACE command to the UDC. See Section 12.4 for more information on the execution of USB device requests.

If the UDCICR1[IECC] is clear, the UDC generates an interrupt to the processor to indicate the SET_CONFIGURATION or SET_INTERFACE command has been completed. Users must set UDCCR[SMAC] to set the endpoint memory allocation to the new configuration or interface setting.

12.6.1.2 Switch Endpoint Memory to Active Configuration

Users can use UDCCR[SMAC] to control the endpoint memory allocation. When the USB host controller sends a SET_CONFIGURATION or SET_INTERFACE command to the UDC (if the configuration-change interrupt is enabled), the UDC sends an interrupt to the processor. After any

data remaining in the endpoint memory is read, setting UDCCR[SMAC] causes the UDC to flush the endpoint memory and changes the endpoint memory allocation to reflect the active configuration and interface set by the USB host controller. The UDC clears UDCCR[SMAC] when the memory allocation has completed. See Section 12.4 for more information on the execution of USB device requests.

Note: The UDC will NAK all bulk and interrupt transactions received after the SET_CONFIGURATION or SET_INTERFACE command was executed, but before the UDC has changed the endpoint memory to the new configuration.

The UDC indicates an overflow condition if an isochronous OUT token is received between the execution of a SET_CONFIGURATION or SET_INTERFACE command and before the changing of the endpoint memory allocation. The UDC sends zero-size packets if an isochronous IN token is received in the same circumstance.

12.6.1.3 Endpoint Memory Configuration Error

The Endpoint Memory Configuration Error (EMCE) bit indicates that the endpoint configuration could not be loaded and that the maximum packet sizes indicated for the configuration are in error. When UDCCR[UDE] is set, the UDC checks the endpoint configurations to verify the endpoint maximum packet sizes are valid and do not allocate more memory per interface than the available 4064 bytes. If the FIFO memory is allocated incorrectly, the UDC does not load the configuration into the USB interface block and sets UDCCR[EMCE] and clears UDCCR[UDE]. UDCCR[EMCE] must be cleared before attempting to re-enable the UDC.

12.6.1.4 UDC Resume

If the USB host controller has executed the SET_FEATURE command and enabled the device remote wake-up feature of the UDC, users can set UDCCR[UDR] when the UDC is in a suspend state to force the UDC into a non-idle state (K state) to perform the remote wake-up operation. When UDCCR[UDR] is set, the UDC drives resume signaling on the USBC_P and USBC_N signals for 3 ms, and then floats the bus. If the USB host controller does not drive resume signaling on the USB within 3 ms, the UDC goes back into suspend. The UDCCR[UDR] bit is cleared by the UDC upon entering the non-idle state.

If the USB host controller has not enabled the device-remote wake-up feature of the UDC, UDCCR[UDR] is ignored.

12.6.1.5 UDC Active

UDCCR[UDA], the UDC active bit, indicates the UDC is currently involved in a USB transaction.

12.6.1.6 UDC Enable

Users can set and clear the UDCCR[UDE] bit to enable and disable the UDC. UDCCR[UDE] is cleared following a reset of the PXA27x processor, which disables the UDC, three-states USBC_P and USBC_N, and disables the monitoring of the USB. Any USB host controller commands or USB reset issued while UDCCR[UDE] is clear are ignored. The Endpoint Configuration registers must be programmed prior to setting UDCCR[UDE]. When UDCCR[UDE] is set, the Endpoint Configuration registers are set to read-only, the configuration is checked and, if valid, the configuration data is loaded into the USB interface block. The UDC is then enabled for USB transmission and reception.

When UDCCR[UDE] is cleared, the entire UDC is disabled and reset. If performed while the UDC is actively transmitting or receiving data, the UDC stops immediately and the remaining bits within the transmit or receive serial shifter are reset, and all data in the transmit and receive FIFOs is lost. While UDCCR[UDE] is clear, the UDC Endpoint Configuration registers are set to read/write access and can be programmed for another configuration. The configuration, interface, address, and features assigned or enabled by the USB host controller to the UDC are reset.

The register organization and individual bit definitions are shown in Table 12-12.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 12-12. UDCCR Bit Definitions (Sheet 1 of 3)

Physical Address 0x40600000 — **UDCCR** — **USB Client Controller**

	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
User Settings																																
Bit	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
	OEN	AALTHNP	AHNP	BHNP	reserved											DWRE	reserved			ACN	AIN				AAISN			SMAC	EMCE	UDR	UDA	UDE
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	OEN	On-The-Go Enable Enables user control of the decoding of OTG SET_FEATURE commands. When OEN is clear, the UDC does not decode SET_FEATURE values of 3, 4, and 5 and responds with a STALL if the USB host controller issues a SET_FEATURE command with those values. When OEN is set, the UDC decodes SET_FEATURE command values of 3, 4, and 5 and responds with an ACK when the USB host controller issues SET_FEATURE command with those values. 0 = On-The-Go features are disabled. 1 = On-The-Go features are enabled.
30	R	AALTHNP	A-Device Alternate Host Negotiation Protocol Port Support Indicates if the A-device alternate host negotiation protocol (HNP) feature has been enabled by the USB host controller. The AALTHNP bit is set when the USB host controller executes a SET_FEATURE command and indicates the a_alt_hnp_support feature is enabled and the *OEN* bit is set. 0 = B-device is connected to an A-device that does not have an alternate port that is capable of HNP. 1 = B-device is connected to an A-device port that is not capable of HNP, but the A-device does have an alternate port that is capable of HNP.
29	R	AHNP	A-Device Host Negotiation Protocol Support Indicates if the A-device HNP feature has been enabled by the USB host controller. The AHNP bit is set when the USB host controller executes a SET_FEATURE command and indicates the a_hnp_support feature is enabled and the *OEN* bit is set. 0 = B-device is connected to an A-device port that does not support HNP. 1 = B-device is connected to an A-device port that supports HNP.

Table 12-12. UDCCR Bit Definitions (Sheet 2 of 3)

Physical Address 0x40600000 — **UDCCR** — **USB Client Controller**

	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
User Settings																																
Bit	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
	OEN	AALTHNP	AHNP	BHNP	reserved											DWRE	reserved			ACN		AIN			AAISN			SMAC	EMCE	UDR	UDA	UDE
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
28	R	BHNP	B-Device Host Negotiation Protocol Enable Indicates if the B-device HNP feature has been enabled by the USB host controller. The BHNP bit is set when the USB host controller executes a SET_FEATURE command and indicates the b_hnp_enable feature is enabled and the OEN bit is set. 0 = B-device HNP enable Feature has not been enabled by the USB host controller. 1 = B-device HNP enable Feature has been enabled by the USB host controller.
27:17	—	—	reserved
16	R	DWRE	Device Remote Wake-Up Feature Indicates if the device remote wake-up feature has been enabled by the USB host controller. This bit is set when the USB host controller executes a SET_FEATURE command and indicates the device remote wake-up feature is to be enabled. 0 = Device remote wake-up feature has not been enabled by the USB host controller. 1 = Device remote wake-up feature has been enabled by the USB host controller.
15:13	—	—	reserved
12:11	R	ACN	0–3—Active UDC Configuration Number Selected by the USB host controller to be used by the UDC. The active configuration number is set when the USB host controller issues a SET_CONFIGURATION command to the UDC.
10:8	R	AIN	0–7—Active UDC Interface Number Selected by the USB host controller to be used by the UDC. The active interface setting number is set when the USB host controller issues a SET_INTERFACE command to the UDC.
7:5	R	AAISN	0–7—Active UDC Alternate Interface Setting Number Selected by the USB host controller to be used by the UDC. The active alternate interface setting number is set when the USB host controller issues a SET_INTERFACE command to the UDC.
4	Read/ Write 1 to Set	SMAC	Switch Endpoint Memory to Active Configuration 0 = No change to active configuration. 1 = Change the endpoint memory allocation to the Active Configuration, Active Interface, and Active Alternate Interface Setting.
3	Read/ Write 1 to Clear	EMCE	Endpoint Memory Configuration Error 0 = No endpoint memory configuration error. 1 = The endpoint memory configuration has an error and cannot be loaded.

Table 12-12. UDCCR Bit Definitions (Sheet 3 of 3)

Physical Address 0x40600000 — UDCCR — USB Client Controller

	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
User Settings																																
Bit	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
	OEN	AALTHNP	AHNP	BHNP	reserved											DWRE	reserved			ACN	AIN				AAISN			SMAC	EMCE	UDR	UDA	UDE
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	Read/ Write 1 to Set	UDR	UDC Resume 0 = Maintain UDC suspend state. 1 = Force UDC out of suspend.
1	R	UDA	UDC Active 0 = UDC currently inactive on the USB. 1 = UDC currently active on the USB.
0	R/W	UDE	UDC Enable 0 = UDC disabled, USBC_P and USBC_N tristated. 1 = UDC enabled, USBC_P and USBC_N used for transmission/ reception.

12.6.2 UDC Interrupt Control Registers (UDCICR0, UDCICR1, and UDCOTGICR)

The UDC Interrupt Control register 0 (UDCICR0) contains 32 read/write control bits to enable interrupt-service requests from data endpoints 0 and A–P. The UDC Interrupt Control register 1 (UDCICR1) contains 21 control bits to enable interrupt-service requests for endpoints Q–X and specific USB events. The UDC On-The-Go Interrupt Control register contains six control bits to enable interrupt-service requests for specific On-The-Go events. Setting any interrupt-enable bit enables the interrupt, while clearing any interrupt-enable bit disables that interrupt. On system reset all of the UDCOTGICR, UDCICR0, and UDCICR1 enable bits are cleared.

Each endpoint (endpoint 0 and programmable endpoints A–X) has the potential of two interrupt sources, FIFO error and packet complete, as specified by the ICx bit field.

- **FIFO Error Interrupt**—This interrupt can be generated when a FIFO underrun or overrun occurs. If an endpoint is programmed to be an IN endpoint, the FIFO error interrupt is generated when a FIFO underrun occurs. A FIFO underrun occurs when the UDC tries to transmit more bytes of data from the transmit buffer than were loaded into the buffer, including when the UDC tries to transmit data when the transmit buffer is empty and the SP bit in the corresponding Endpoint Control register is clear (or IPR bit is not set for endpoint 0). If an endpoint is programmed to be an OUT endpoint, the FIFO error interrupt is generated when a FIFO overrun occurs. A FIFO overrun occurs when the UDC tries to load more bytes of data into the receive buffer than the maximum packet size of the buffer, or if data is received but the receive buffer(s) is full. Since endpoint 0 is a bidirectional endpoint, a FIFO error can occur on either a FIFO-overrun or FIFO-underrun condition. Refer to Section 12.6.8.7 and Section 12.4.2 for more information on FIFO error conditions.

- **Packet Complete Interrupt**—If an endpoint is configured as an OUT endpoint, the packet-complete interrupt indicates that an OUT packet has been received and is ready to be read. If an endpoint is configured as an IN endpoint, the interrupt is generated when an IN packet has been transmitted. The packet-complete interrupt is also generated by the DME bit in the corresponding Control/Status register. (Refer to Section 12.6.8.6 for more details on DME.)

Note: For all endpoints, IEx[1:0] is defined as follows:

0b00 = No interrupts enabled
0b01 = Packet complete interrupt enabled
0b10 = FIFO error interrupt enabled
0b11 = Both packet complete and FIFO error interrupts enabled

Table 12-13 shows the USB event interrupts, the enable and status bits, and the USB event that causes each interrupt to be generated by the UDC. For the interrupt enables for reset (IERS), suspend (IESU), resume (IERU), SOF (IESOF), and configuration-change (IECC) conditions, each bit in the UDCICR1 enables its respective interrupt request. When the interrupt-enable bit is set, the interrupt is enabled and is generated when the USB event occurs. When the interrupt-enable bit clear, the interrupt is disabled and is not generated. The setting of the enable bit does not affect the setting of the UDC and endpoint status bits. If an event occurs, the status bit is set, and if the interrupt-enable bit is set, an interrupt is also generated and the bit in the Interrupt Status register is set.

Table 12-13. USB Event Interrupts

Interrupt Enable Bit (UDCICR1)	Interrupt Status Bit (UDCISR1)	USB Condition That Generates Interrupt
IECC	IRCC	SET_CONFIGURATION or SET_INTERRUPT command received
IESOF	IRSOF	Start-of-frame received
IERU	IRRU	Resume detected
IESU	IRSU	Suspend detected
IERS	IRRS	USB reset detected

Table 12-14 shows the USB On-The-Go events interrupts, the enable and status bits, and the USB event that causes each interrupt to be generated by the UDC. For the On-The-Go interrupt enables, each bit in the UDCOTGICR enables its respective interrupt request. When the interrupt-enable bit is set, the interrupt is enabled and is generated when the USB On-The-Go event occurs. When the interrupt-enable bit is clear, the interrupt is disabled and is not generated. The setting of the enable bit does not affect the setting of the UDC and interrupt status bits. If an event occurs, the status bit is set, and if the interrupt-enable bit is set, an interrupt is also generated and the bit in the Interrupt Status register is set.

Table 12-14. USB On-The-Go Event Interrupts (Sheet 1 of 2)

Interrupt Enable Bit (UDCOTGICR)	Interrupt Status Bit (UDCOTGISR)	USB OTG Condition Generating Interrupt
IESF	IRSF	OTG SET_FEATURE Command
IEXR	IRXR	External OTG Transceiver Interrupt Rising Edge
IEXF	IRXF	External OTG Transceiver Interrupt Falling Edge
IEVV40R	IRVV40R	OTG Vbus Valid 4.0-V Rising Edge
IEVV40F	IRVV40F	OTG Vbus Valid 4.0-V Falling Edge
IEVV44R	IRVV44R	OTG Vbus Valid 4.4-V Rising Edge

Table 12-14. USB On-The-Go Event Interrupts (Sheet 2 of 2)

Interrupt Enable Bit (UDCOTGICR)	Interrupt Status Bit (UDCOTGISR)	USB OTG Condition Generating Interrupt
IEVV44F	IRVV44F	OTG Vbus Valid 4.4-V Falling Edge
IESVR	IRSVR	OTG Session Valid Rising Edge
IESVF	IESVF	OTG Session Valid Falling Edge
IESDR	IRSDR	OTG A-Device SRP Detect Rising Edge
IESDF	IESDF	OTG A-Device SRP Detect Falling Edge
IEIDR	IRIDR	OTG ID Change Rising Edge
IEIDF	IEIDF	OTG ID Change Falling Edge

- **OTG Interrupts**—Intended to notify the user when an OTG SET_FEATURE command has been received and can be used with an external USB transceiver and charge pump to provide OTG operation. The SET_FEATURE command interrupt is set when a valid OTG SET_FEATURE command has been decoded and the OTG feature status bit is set in the UDC Control Register (UDCCR). The external transceiver rising-edge interrupt is set when the interrupt input from an external USB transceiver toggles from 0 to 1. Similarly, the external transceiver falling-edge interrupt is set when the interrupt input from an external USB transceiver toggles from 1 to 0. Select the appropriate transition for the external USB transceiver being used in the system. See Section 12.5.
- **OTG Vbus Valid Interrupts**—Interface to an external charge pump device and detect the Vbus voltage levels. OTG Vbus valid 4.0-V interrupts detect a low-voltage condition and provide notification of Vbus voltage below the value necessary for proper operation of a B-device ($V_{A\_VBUS\_VLD}$ min). OTG Vbus valid 4.4-V interrupts detect the Vbus voltage has exceeded the A-device output voltage ($V_{A\_VBUS\_OUT}$) valid threshold of 4.4 V. The appropriate edge-detection of the interrupts can be used based on the polarity of the OTG charge-pump Vbus valid output signals. For example, if the OTG charge-pump Vbus valid outputs are active high, the OTG Vbus valid 4.0-V falling-edge interrupt indicates the A-device output voltage has fallen below the $V_{A\_VBUS\_VLD}$ minimum specified value, and the OTG Vbus valid 4.4-V rising-edge interrupt indicates the A-device output voltage has reached the valid threshold of 4.4 V.
- **OTG Session Valid Interrupts**—Interface to an external OTG charge pump and interrupt when the external OTG charge pump detects a valid OTG session. The appropriate OTG session valid interrupt can be used based on the polarity of the OTG charge-pump session-valid output signal. For example, if the OTG charge-pump session-valid output signal is active high, the OTG session valid rising-edge interrupt indicates an active OTG session and that the Vbus is not ready for SRP, and the OTG session-valid falling-edge interrupt indicates the OTG session is not valid and the bus is ready for SRP.
- **OTG A-Device SRP-Detect Interrupts**—Interface to an external OTG charge pump and interrupt when the external OTG charge pump detects the start of a valid SRP. The appropriate OTG A-device SRP interrupt can be used based on the polarity of the OTG charge pump A-device SRP-detect output signal. For example, if the OTG charge pump A-device SRP-detect output signal is active high, the OTG A-device SRP rising-edge interrupt indicates a valid SRP has been detected and the OTG A-device SRP falling-edge interrupt indicates the end of the SRP.
- **OTG ID Interrupts**—Interrupt when a change is detected in the OTG ID signal. The OTG ID rising-edge interrupt is set when the OTG ID signal toggles from 0 to 1. Similarly, the OTG ID falling edge interrupt is set when the OTG ID signal toggles from 1 to 0.

Note: Programming any interrupt-enable bit to 0 does not affect the current state of the corresponding interrupt-request bit in the Interrupt Status register or corresponding bit in the Endpoint Control registers; it only blocks future 0-to-1 transitions of the interrupt request and the future setting of the corresponding interrupt-request bit in the Interrupt Status register.

The register organization and individual bit definitions for UDCICR0, UDCICR1, and UDCOTGICR are shown in Table 12-15, Table 12-16, and Table 12-17 respectively.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 12-15. UDCICR0 Bit Definitions

Physical Address 0x4060_0004 — **UDCICR0** — **USB Client Controller**

	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
User Settings																																
Bit	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
	IEP		IEN		IEM		IEL		IEK		IEJ		IEI		IEH		IEG		IEF		IEE		IED		IEC		IEB		IEA		IE0	
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

Bits	Access	Name	Description
31:30	R/W	IEP	Interrupt Enables—Endpoint P 0b00 = No interrupts enabled. 0b01 = Packet-complete interrupt enabled. 0b10 = FIFO error interrupt enabled. 0b11 = Both packet-complete and FIFO error interrupts enabled. **NOTE:** The same definitions also apply to IE bits N–A and IE0 in this register.
29:28	R/W	IEN	Interrupt Enables—Endpoint N
27:26	R/W	IEM	Interrupt Enables—Endpoint M
25:24	R/W	IEL	Interrupt Enables—Endpoint L
23:22	R/W	IEK	Interrupt Enables—Endpoint K
21:20	R/W	IEJ	Interrupt Enables—Endpoint J
19:18	R/W	IEI	Interrupt Enables—Endpoint I
17:16	R/W	IEH	Interrupt Enables—Endpoint H
15:14	R/W	IEG	Interrupt Enables—Endpoint G
13:12	R/W	IEF	Interrupt Enables—Endpoint F
11:10	R/W	IEE	Interrupt Enables—Endpoint E
9:8	R/W	IED	Interrupt Enables—Endpoint D
7:6	R/W	IEC	Interrupt Enables—Endpoint C
5:4	R/W	IEB	Interrupt Enables—Endpoint B
3:2	R/W	IEA	Interrupt Enables—Endpoint A
1:0	R/W	IE0	Interrupt Enables—Endpoint 0

Table 12-16. UDCICR1 Bit Definitions

Physical Address 0x4060_0008 — UDCICR1 — USB Client Controller

	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
User Settings																																
Bit	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
	IECC	IESOF	IERU	IESU	IERS	reserved											IEX		IEW		IEV		IEU		IET		IES		IER		IEQ	
Reset	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	IECC	Interrupt Enable—Configuration Change
30	R/W	IESOF	Interrupt Enable—Start of Frame
29	R/W	IERU	Interrupt Enable—Resume
28	R/W	IESU	Interrupt Enable—Suspend
27	R/W	IERS	Interrupt Enable—Reset
26:16	—	—	reserved
15:14	R/W	IEX	Interrupt Enables—Endpoint X 0b00 = No interrupts enabled. 0b01 = Packet-Complete interrupt enabled. 0b10 = FIFO error interrupt enabled. 0b11 = Both packet-complete and FIFO error interrupts enabled. **NOTE:** The same definitions also apply to IE bits W–Q in this register.
13:12	R/W	IEW	Interrupt Enables—Endpoint W
11:10	R/W	IEV	Interrupt Enables—Endpoint V
9:8	R/W	IEU	Interrupt Enables—Endpoint U
7:6	R/W	IET	Interrupt Enables—Endpoint T
5:4	R/W	IES	Interrupt Enables—Endpoint S
3:2	R/W	IER	Interrupt Enables—Endpoint R
1:0	R/W	IEQ	Interrupt Enables—Endpoint Q

Table 12-17. UDCOTGICR Bit Definitions

Physical Address 0x4060_0018 — **UDCOTGICR** — **USB Client Controller**

User Settings

Bit	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							IESF	reserved						IEXR	IEXF	reserved						IEVV40R	IEVV40F	IEVV44R	IEVV44F	IESVR	IESVF	IESDR	IESDF	IEIDR	IEIDF
Reset	?	?	?	?	?	?	?	0	?	?	?	?	?	?	0	0	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:25	—	—	reserved
24	R/W	IESF	OTG SET_FEATURE Command Received
23:18	—	—	reserved
17	R/W	IEXR	External Transceiver Interrupt Rising-Edge Interrupt Enable
16	R/W	IEXF	External Transceiver Interrupt Falling-Edge Interrupt Enable
15:10	—	—	reserved
9	R/W	IEVV40R	OTG Vbus Valid 4.0-V Rising-Edge Interrupt Enable
8	R/W	IEVV40F	OTG Vbus Valid 4.0-V Falling-Edge Interrupt Enable
7	R/W	IEVV44R	OTG Vbus Valid 4.4-V Rising-Edge Interrupt Enable
6	R/W	IEVV44F	OTG Vbus Valid 4.4-V Falling-Edge Interrupt Enable
5	R/W	IESVR	OTG Session Valid Rising-Edge Interrupt Enable
4	R/W	IESVF	OTG Session Valid Falling-Edge Interrupt Enable
3	R/W	IESDR	OTG A-Device SRP Detect Rising-Edge Interrupt Enable
2	R/W	IESDF	OTG A-Device SRP Detect Falling-Edge Interrupt Enable
1	R/W	IEIDR	OTG ID Change Rising-Edge Interrupt Enable
0	R/W	IEIDF	OTG ID Change Falling-Edge Interrupt Enable

12.6.3 USB Port 2 Output Control Register (UP2OCR)

The USB Port 2 Output Control register contains control bits to select the input and output signals for the host controller port 2 USB transceivers, the USB device controller transceivers, and the GPIOs used for USB OTG operation. With these bits, several interface options can be selected. Only one connection is allowed to each of the ports at a time. Unpredictable behavior will occur if more than one set of I/O is specified for a port. Refer to Section 12.5 for more information on OTG interface options.

Table 12-18 shows the legal combinations of USB port 2 control bit settings.

Table 12-18. Legal Combinations of USB Port 2 Control Bit Settings

UP2OCR Control Bits			Differential Port	Single-Ended Port 2
HXOE	HXS	SEOS		
0	0 or 1	0	Off	Off
0	0 or 1	2	Off	External Non-OTG Transceiver Client
0	0 or 1	3	Off	External Non-OTG Transceiver Host
0	0 or 1	4	Off	External OTG Transceiver Client
0	0 or 1	5	Off	External OTG Transceiver Host
1	0	3	Non-OTG Client	Non-OTG Host
1	0	6	Internal OTG Transceiver Client	External OTG Charge Pump Client
1	1	2	Non-OTG Host	Non-OTG Client
1	1	7	Internal OTG Transceiver Host	External OTG Charge Pump Host
1	0	0	Non-OTG Client	Off
1	1	0	Non-OTG Host	Off

12.6.3.1 USB Host Controller Single-Ended Output Select

The USB host controller single-ended output select bits allow the user to choose the USB data output on the single-ended signals through GPIOs. The GPIO controller must be programmed to select the USB signals. The user can select between USB device controller single-ended signals, USB host controller single-ended signals, the USB device controller signals to interface with an external transceiver, the USB host controller signals to interface to an external transceiver, the USB host controller to interface to an external charge pump, and the USB device controller to interface to an external charge pump. The USB host controller single-ended outputs can provide wake-ups when a connect/disconnect or resume signaling is detected. For information on programming UDC wake-up events refer to Chapter 5, “DMA Controller”. Table 12-18 lists the signals selected for output on the USB alternate function ports for each value of SEOS in USB Port 2 Output Control Register (UP2OCR). A value of SEOS = 0 does not select any outputs. Refer to Section 12.5 for more information on the operation of each selection value.

Table 12-19. Alternate Function Port Signals Selection

GPIO Alternate Function Port	SEOS Value Selection							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
USB_P2_1	-	-	UDC Rx Data (RCV—in)	UHC Rx Data (RCV—in)	External. OTG Transceiver Interrupt		SRP Detect	
USB_P2_2	-	-	UDC OE (OE_n—out)	UHC OE (OE_n—out)	OE_Tp_Int_n		Session Valid	
USB_P2_3	-	-	UDC Rx D– (VM—in)	UHC Rx D– (VM—in)	-		Vbus Valid 4.4-V	
USB_P2_4	-	-	UDC Tx D– (VMO—out)	UHC Tx D– (VMO—out)	UDC D–	UHC D–	Charge Pump Vbus Enable	
USB_P2_5	-	-	UDC Rx D+ (VP—in)	UHC Rx D+ (VP—in)	UDC D+	UHC D+	Vbus Valid 4.0-V	
USB_P2_6	-	-	UDC Tx D+ (VPO—out)	UHC Tx D+ (VPO—out)	-		-	
USB_P2_7	-	-	SPEED		SPEED		OTG ID	
USB_P2_8	-	-	-	USBHPEN2	Suspend		Charge Pump Vbus Pulse Enable	

12.6.3.2 USB Host Port 2 Transceiver Output Enable

The USB Host Port 2 Transceiver Output Enable (HXOE) bit enables and disables the USB host controller port 2 on-chip OTG transceiver. When HXOE is clear, the USB host controller port 2 OTG transceiver is disabled and no USB data can be transmitted or received through this port. When HXOE is set, the USB host controller port 2 OTG transceiver is enabled and USB data can be transmitted and received through this port.

12.6.3.3 USB Host Port 2 Transceiver Output Select

The USB Host Port 2 Transceiver Output Select (HXS) bit selects the on-chip USB controller that is to use the USB host controller port 2 as a transceiver. When HXS is clear, the USB device controller I/O is assigned to the USB host port 2 transceiver and when HXS is set, the USB host controller port 2 I/O is assigned to the USB host port 2 transceiver. If the USB device controller I/O is selected, the transceiver operation is fixed at USB high speed.

12.6.3.4 OTG ID Output Enable

The OTG ID Output Enable (IDON) bit allows the user to read the value selected by the USB OTG cable on the ID pin to select A- or B-device functionality. The ID input signal is valid only when IDON is set. To read the value on the OTG ID pin, set IEIDR and IEIDF in the UDC Interrupt Control Registers (UDCICR0, UDCICR1, and UDCOTGICR) to 1, then set IDON to 1, and then

sample IRIDR and IRIDF in the UDC Interrupt Status Registers (UDCISR0, UDCISR1, and UDCOTGISR). After reading the OTG ID value, set IDON to 0 to reduce power consumption. Refer to Section 12.5.6 for more information.

12.6.3.5 External Transceiver Suspend Enable

The External Transceiver Suspend Enable (EXSUS) bit sets the Suspend Enable bit that can be used by an external transceiver to indicate the USB must enter suspend.

12.6.3.6 External Transceiver Speed Control

The External Transceiver Speed Control (EXSP) bit sets the speed control signal that can be used by an external transceiver to indicate the speed of the USB connection.

12.6.3.7 Host Port 2 D– Pull-Up Bypass Enable

The Host Port 2 D– Pull-Up Bypass Enable (DMPUBE) bit is used in conjunction with Host Port 2 D– Pull-Up Enable to control the pull-up resistance applied to D– pin of the host port 2 transceiver. When DMPUBE is 1, SW2 shown in Figure 12-21 is closed, causing the pull-up R_{PU2} to be bypassed and not included in the pull-up resistance on D–. When DMPUBE is 0, SW2 shown in Figure 12-21 is open, causing the pull-up resistor R_{PU2} to be added to the pull-up resistance on D–. Refer to Table 12-9 for a summary of the SW2 programming values.

In addition to user control of SW2, SW2 is enabled and disabled by the USB device controller when the bus state changes to/from idle and receiving data from the USB host controller. The hardware enabling and disabling of SW2 is not reflected in the value of DMPUBE.

Figure 12-21. D– Pull-Up Resistors

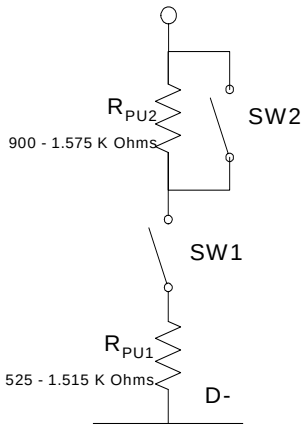
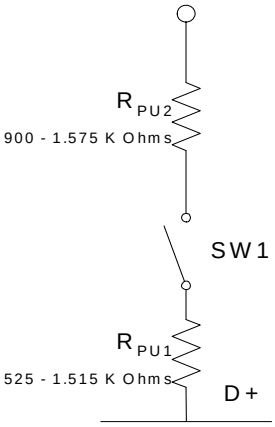


Figure 12-22. D+ Pull-Up Resistors



12.6.3.8 Host Port 2 D– Pull-Up Enable

The Host Port 2 Pull-Up Enable (DMPUE) bit enables and disables the pull-ups on the D– pin on the host controller port 2 transceiver. When DMPUE is 1, SW1 shown in Figure 12-21 is closed, enabling the D– pull-ups. When DMPUE is 0, SW1 is open, disabling the D– pull-ups. Set DMPUE to 0 when the USB device controller is selected as the OTG I/O for host Port 2 output. With this selection, the D– pin pull-ups are disabled when the USB OTG is either an A-device or B-device but is acting as the USB client. Set DMPUE to 0 when OTG is not being used. Refer to Table 12-9 for a summary of the SW1 programming values.

Note: Data-line pulsing is performed using software control of the pull-up enable control bits, Host Port 2 D– Pull Up Enable (DPPUE) and Host Port 2 D+ Pull Up Enable (DMPDE).

12.6.3.9 Host Port 2 D+ Pull-Up Enable

The Host Port 2 Pull-Up Enable (DPPUE) bit enables and disables the pull-ups on D+ pin on the host controller port 2 transceiver. When DPPUE is 1, SW1 shown in Figure 12-22 is closed, enabling the D+ pull-ups. When DPPUE is 0, SW1 is open, disabling the D+ pull-ups. Set DPPUE to 1 when the USB device controller is selected as the OTG I/O for host Port 2 output. With this selection, the D+ pin pull-ups is enabled when the USB OTG is either an A-device or B-device but is acting as the USB client. Set DPPUE to 0 when the USB host controller is selected as the OTG I/O for host Port 2 output. With this selection, the D+ pin pull-ups is disabled when the USB OTG is either an A-device or a B-device but is acting as the USB host controller. Set DPPUE to 0 when OTG is not being used. Refer to Table 12-9 for a summary of the SW1 programming values.

Note: Data-line pulsing is performed using software control of the pull-up enable control bits, Host Port 2 D– Pull Up Enable (DPPUE) and Host Port 2 D+ Pull Up Enable (DMPDE).

12.6.3.10 Host Port 2 Pull D– Down Enable

The Host Port 2 D– Pull-Down Enable (DMPDE) bit enables and disables the pull-down on the D– pin on the host controller port 2 transceiver. When DMPDE is 1, the D– 14.25 K–24.8 K Ohm pull-down resistor is enabled and when PDE is 0, the D– pull-down is disabled.

Set PDE to 1 when the host port 2 implements a USB OTG interface. With this setting, the D– pull-down is enabled when the USB OTG ports are either an A- or B-device. Set PDE to 0 when the host port 2 is being used as the transceiver for the USB device controller in a non-OTG configuration. With this setting, the D– pull-down is disabled as required for UDC transceiver I/O. Refer to Table 12-9 for a summary of the pull-down enable programming values.

12.6.3.11 Host Port 2 Pull D+ Down Enable

The Host Port 2 Pull-Down Enable (DPPDE) bit enables and disables the pull-down on the D+ pins on the host controller port 2 transceiver. When DPPDE is 1, the D+ pull-down is enabled and when DPPDE is 0, the D+ pull-down is disabled.

Set DPPDE to 1 when the host port 2 implements a USB OTG interface and the OTG ID is 0, indicating the USB OTG interface is to act as the A-device on the USB OTG connection. With this setting, the D+ pull-downs are enabled when the USB OTG ports are the A-device and are to pull-down D+. Set PDE to 0 when the host port 2 implements a USB OTG interface and the OTG ID is 1, indicating that the USB OTG interface is to be the B-device on the USB OTG connection. With this setting, the D+ pull-down is disabled when the processor USB OTG ports are the B-device. Also set PDE to 0 when the host port 2 is being used as the transceiver for the USB device controller in a non-OTG configuration. With this setting, the D– pull-down is disabled as required for UDC transceiver I/O. Refer to Table 12-9 for a summary of the pull-down enable programming values.

12.6.3.12 Charge Pump Vbus Pulse Enable

The Charge Pump Vbus Pulse Enable (CPVPE) bit enables the Vbus charge circuitry in an external charge pump device to provide Vbus using a 10-mA current source to provide pulsing required by the OTG session request protocol.

Note: Data-line pulsing is performed using software control of the pull-up enable control bits, Host Port 2 D– Pull Up Enable (DPPUE) and Host Port 2 D+ Pull Up Enable (DMPDE).

12.6.3.13 Charge Pump Vbus Enable

The Charge Pump Vbus Enable (CPVE) bit enables an external charge pump device to provide voltage to the USB Vbus.

The register organization and individual bit definitions are shown in Table 12-22.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 12-20. UP2OCR Bit Definitions (Sheet 1 of 2)

Physical Address 0x4060_0020 — **UP2OCR** — **USB Client Controller**

Bit	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					SEOS			reserved						HXOE	HXS	reserved					IDON	EXSUS	EXSP	DMSTATE	VPMBlockEnbN	DPSTATE	DPPUE	DMPDE	DPPDE	CPVPE	CPVEN
Reset	?	?	?	?	?	0	0	0	?	?	?	?	?	?	1	1	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:27	—	—	reserved
26:24	R/W	SEOS	Single-Ended Output Select 0b000 = Off. No output selected. 0b001 = Not used. 0b010 = Device controller single-ended operation using GPIO pads with an external transceiver. 0b011 = Host controller single-ended operation using GPIO pads with an external transceiver. 0b100 = Device controller with external OTG transceiver. 0b101 = Host controller with external OTG transceiver. 0b110 = Device controller with external charge pump. 0b111 = Host controller with external charge pump.
23:18	—	—	reserved
17	R/W	HXOE	Host Port 2 Transceiver Output Enable 0 = On-chip host controller transceiver disabled. 1 = On-chip host controller transceiver enabled.
16	R/W	HXS	Host Port 2 Transceiver Output Select 0 = Output signals from USB device controller. 1 = Output signals from USB host controller.
15:11	—	—	reserved
10	R/W	IDON	OTG ID Read Enable
9	R/W	EXSUS	External Transceiver Suspend Enable
8	R/W	EXSP	External Transceiver Speed Control
7	R	DMSTATE	This bit reflects the value of the Host Port 2 D- line after two 26MHz clock delay or approximately one 12MHz USB clock. **NOTE:** This bit field is available in the C5 stepping and any subsequent steppings. On C0 processors this bit field is: DMPUBE Host Port 2 Transceiver D– Pull Up Bypass Enable. On C5 and beyond this pullup bypass is always disabled.

Table 12-20. UP2OCR Bit Definitions (Sheet 2 of 2)

Physical Address 0x4060_0020 — UP2OCR — USB Client Controller

Bit	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					SEOS			reserved						HXOE	HXS	reserved					IDON	EXSUS	EXSP	DMSTATE	VPMBlockEnbN	DPSTATE	DPPUE	DMPDE	DPPDE	CPVPE	CPVEN
Reset	?	?	?	?	?	0	0	0	?	?	?	?	?	?	1	1	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
6	R/W	VPMBlockEnbN	Host Port 3 VPM Block Enable 0 = When the USB is in transceiverless mode (UP3OCR[CFG] = 0x2), then USB_P3_6 (VPO) and USB_P3_4 (VMO) are ignored by USB Host Port 3 when USB_P3_2 (OE_n) is deasserted. 1 = When the USB is in transceiverless mode (UP3OCR[CFG] = 0x2), then USB_P3_6 (VPO) and USB_P3_4 (VMO) are not ignored by USB Host Port 3 when USB_P3_2 (OE_n) is deasserted. **NOTE:** This bit field is available in the C5 stepping and any subsequent steppings. On C0 processors this bit field is: DPPUBE Host Port 2 Transceiver D+ Pull Up Bypass Enable. On C5 steppings and beyond this pullup bypass is always disabled.
5	R	DPSTATE	This bit reflects the value of the Host Port 2 D+ line after two 26MHz clock delay or approximately one 12MHz USB clock. **NOTE:** This bit field is available in the C5 stepping and any subsequent steppings. On C0 processors this bit field is: DMPUE Host Port 2 Transceiver D– Pull Up Enable. On C5 and beyond this internal pullup is no longer available for turning on and off. It is always disabled.
4	R/W	DPPUE	Host Port 2 Transceiver D+ Pull Up Enable
3	R/W	DMPDE	Host Port 2 Transceiver D– Pull Down Enable
2	R/W	DPPDE	Host Port 2 Transceiver D+ Pull Down Enable
1	R/W	CPVPE	Charge Pump Vbus Pulse Enable
0	R/W	CPVEN	Charge Pump Vbus Enable

12.6.4 USB Port 3 Output Control Register (UP3OCR)

The USB Port 3 Output Control register (UP3OCR) controls the Port 3 transceivers on the USB host controller to enable using an external transceiver.

12.6.4.1 USB Host Port 3 Configuration

The USB Host Port 3 Configuration (CFG) bit field sets the operating configuration for port 3 of the USB host controller. This port can connect to an external device and/or transceiver in several different configurations. The USB host controller port 3 inputs can provide a wake-up when resume signaling is detected. USB host controller port 3 inputs cannot provide a wake-up when a connect/disconnect is detected. For information on programming UDC wake-up events refer to Chapter 3, “Clocks and Power Manager”.

When CFG is set to 0x0, the USB host controller port 3 is configured to connect the internal USB host controller port 3 outputs to an external USB transceiver.

When CFG is set to 0x2, the USB host controller port 3 is configured to connect the processor USB host controller port 3 outputs to an external USB device controller, with the external USB device controller providing the output enable control to the processor USB host controller port 3 transceiver. In this mode, Port 3 is best thought of as looking like a transceiver to the external device. Table 12-21 lists the USB Host Port 3 Configuration selection values and the signals assigned to each GPIO for each setting.

Table 12-21. Port 3 Configuration Selection

GPIO Alternate Function Port	Host Port 3 Configuration (CFG)			
	0x0	0x1	0x2	0x3
USB_P3_1	UHC Rx Data (RCV—in)	-	UHC Rx Data (RCV—out)	-
USB_P3_2	UHC OE (OE_n—out)	-	UHC OE (OE_n—in)	-
USB_P3_3	UHC Rx D– (VM—in)	-	UHC Rx D– (VM—out)	-
USB_P3_4	UHC Tx D– (VMO—out)	-	UHC Tx D– (VMO—in)	-
USB_P3_5	UHC Rx D+ (VP—in)	-	UHC Rx D+ (VP—out)	-
USB_P3_6	UHC Tx D+ (VPO—out)	-	UHC Tx D+ (VPO—in)	-

The register organization and individual bit definitions are shown in Table 12-22.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 12-22. UP3OCR Bit Definitions

Physical Address 0x4060_0024 — **UP3OCR** — **USB Client Controller**

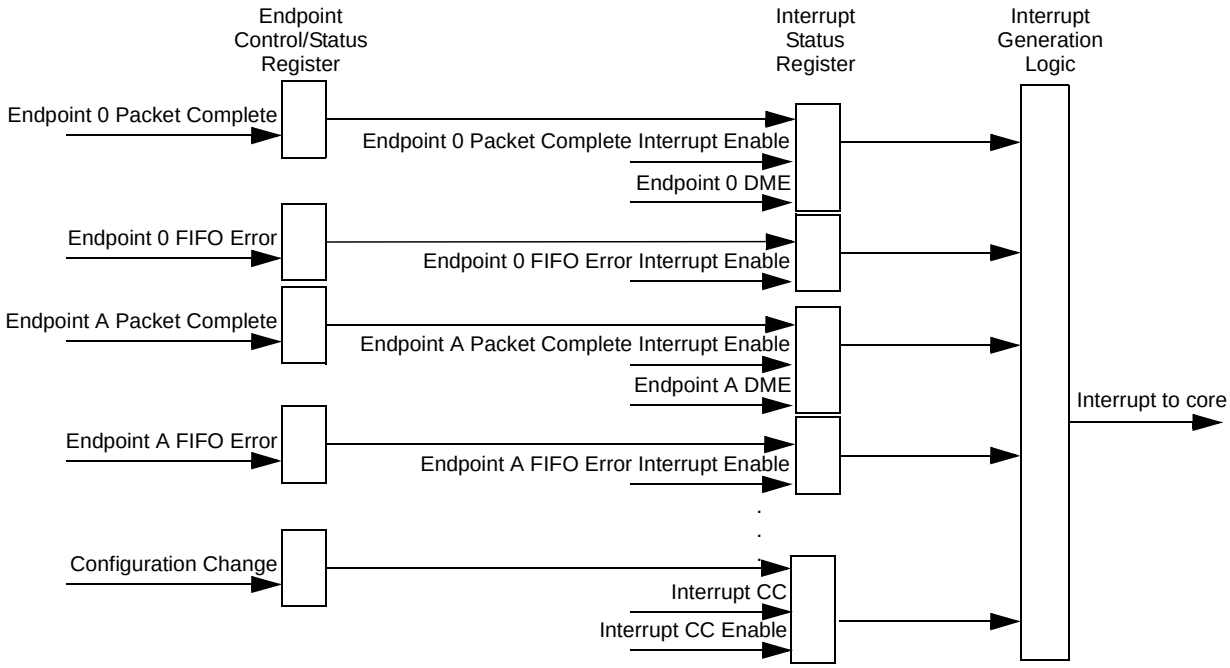
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved																														CFG	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
31:2	—	—	reserved
1:0	R/W	CFG	Host Port 3 Configuration 0x0 = Host controller port 3 transceiver with external host controller 0x1 = Not used 0x2 = Host controller port 3 transceiver with external device controller 0x3 = ForceSE0 mode. This mode modifies USB_P3_3 to be ForceSE0 and USB_P3_5 to be ForceSE0N. The remaining pins keep the same function as CFG=0x2.

12.6.5 UDC Interrupt Status Registers (UDCISR0, UDCISR1, and UDCOTGISR)

The UDC Interrupt Status registers UDCOTGISR, UDCISR0, and UDCISR1 contain bits to generate the UDC interrupt request. Each bit in the UDC Interrupt Status registers is logically ORed together to produce one interrupt request. Figure 12-23 shows the UDC interrupt generation. When the interrupt service routine (ISR) for the UDC is executed, it must read the UDC Interrupt Status registers to determine why the UDC interrupt occurred.

Figure 12-23. UDC Interrupt Generation



Every bit in UDCOTGISR, UDCISR0, and UDCISR1 is controlled by an enable bit in the UDC Interrupt Control registers (UDCOTGICR, UDCICR0, and UDCICR1). When the enable bits are clear, they prevent a status bit in the corresponding UDC Interrupt Status register from generating an interrupt. If the enable bit for a particular status bit is set and an interruptible condition occurs, the interrupt-status bit is set and a UDC interrupt is generated. To clear interrupt-status bits, users must write 0b1 to the bit position to be cleared. The interrupt request for the UDC remains active as long as the value of the UDC interrupt-status register bits ANDed with the enable bit is non-zero.

UDCISR0 contains interrupt status bits for endpoints 0 and A–P. UDCISR1 contains interrupt status bits for endpoints Q–X. Each endpoint has the potential of two interrupt sources: FIFO error and packet complete. UDCOTGISR contains interrupt-status bits for twelve USB On-The-Go events. UDCISR1 also contains interrupt status bits for five USB events (see Section 12.6.2 for additional details on the use of these interrupts).

Note: Setting any interrupt-enable bits does not affect the state of the corresponding interrupt-request bit in the Interrupt Status register; it only blocks future 0-to-1 transitions of the interrupt-request signal. All interrupt-request bits in the Interrupt Status registers are read/write and “write 1 to clear.”

The register organization and individual bit definitions are shown in Table 12-23, Table 12-24, and Table 12-25.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 12-23. UDCISR0 Bit Definitions

Physical Address 0x4060_000C — **UDCISR0** — **USB Client Controller**

	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
User Settings																																
Bit	IRP		IRN		IRM		IRL		IRK		IRJ		IRI		IRH		IRG		IRF		IRE		IRD		IRC		IRB		IRA		IR0	
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

Bits	Access	Name	Description
31:30	R/W	IRP	Interrupt Requests—Endpoint P
29:28	R/W	IRN	Interrupt Requests—Endpoint N
27:26	R/W	IRM	Interrupt Requests—Endpoint M
25:24	R/W	IRL	Interrupt Requests—Endpoint L
23:22	R/W	IRK	Interrupt Requests—Endpoint K
21:20	R/W	IRJ	Interrupt Requests—Endpoint J
19:18	R/W	IRI	Interrupt Requests—Endpoint I
17:16	R/W	IRH	Interrupt Requests—Endpoint H
15:14	R/W	IRG	Interrupt Requests—Endpoint G
13:12	R/W	IMF	Interrupt Requests—Endpoint F
11:10	R/W	IRE	Interrupt Requests—Endpoint E
9:8	R/W	IRD	Interrupt Requests—Endpoint D
7:6	R/W	IRC	Interrupt Requests—Endpoint C
5:4	R/W	IRB	Interrupt Requests—Endpoint B
3:2	R/W	IRA	Interrupt Requests—Endpoint A
1:0	R/W	IR0	Interrupt Requests—Endpoint 0

Table 12-24. UDCISR1 Bit Definitions

Physical Address 0x4060_0010 — **UDCISR1** — **USB Client Controller**

Bit	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
	IRCC	IRSOF	IRRU	IRSU	IRRS	reserved											IRX		IRW		IRV		IRU		IRT		IRS		IRR		IRQ	
Reset	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	IRCC	Interrupt Request—Configuration Change
30	R/W	IRSOF	Interrupt Request—Start of Frame
29	R/W	IRRU	Interrupt Request—Resume
28	R/W	IRSU	Interrupt Request—Suspend
27	R/W	IRRS	Interrupt Request—Reset
26:16	—	—	reserved
15:14	R/W	IRX	Interrupt Requests—Endpoint X
13:12	R/W	IRW	Interrupt Requests—Endpoint W
11:10	R/W	IRV	Interrupt Requests—Endpoint V
9:8	R/W	IRU	Interrupt Requests—Endpoint U
7:6	R/W	IRT	Interrupt Requests—Endpoint T
5:4	R/W	IRS	Interrupt Requests—Endpoint S
3:2	R/W	IRR	Interrupt Requests—Endpoint R
1:0	R/W	IRQ	Interrupt Requests—Endpoint Q

Table 12-25. UDCOTGISR Bit Definitions (Sheet 1 of 2)

Physical Address 0x4060_001C — **UDCOTGISR** — **USB Client Controller**

Bit	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							IRSF	reserved						IRXR	IRXF	reserved						IRVV40R	IRVV40F	IRVV44R	IRVV44F	IRSVR	IRSVF	IRSDR	IRSDF	IRIDR	IRIDF
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:25	—	—	reserved
24	R/WC†	IRSF	OTG SET_FEATURE Command Received
23:18	—	—	reserved
17	R/WC†	IRXR	External Transceiver Interrupt Rising-Edge Interrupt Request
16	R/WC†	IRXF	External Transceiver Interrupt Falling-Edge Interrupt Request
15:10	—	—	reserved

Table 12-25. UDCOTGISR Bit Definitions (Sheet 2 of 2)

Physical Address 0x4060_001C	UDCOTGISR	USB Client Controller

User Settings																															
Bit	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							IRSF	reserved						IRXR	IRXF	reserved						IRVV40R	IRVV40F	IRVV44R	IRVV44F	IRSVR	IRSVF	IRSDR	IRSDF	IRIDR	IRIDF
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
9	R/WC†	IRVV40R	OTG Vbus Valid 4.0-V Rising-Edge Interrupt Request
8	R/WC†	IRVV40F	OTG Vbus Valid 4.0-V Falling-Edge Interrupt Request
7	R/WC†	IRVV44R	OTG Vbus Valid 4.4-V Rising-Edge Interrupt Request
6	R/WC†	IRVV44F	OTG Vbus Valid 4.4-V Falling-Edge Interrupt Request
5	R/WC†	IRSVR	OTG Session Valid Rising-Edge Interrupt Request
4	R/WC†	IRSVF	OTG Session Valid Falling-Edge Interrupt Request
3	R/WC†	IRSDR	OTG A-Device SRP Detect Rising-Edge Interrupt Request
2	R/WC†	IRSDF	OTG A-Device SRP Detect Falling-Edge Interrupt Request
1	R/WC†	IRIDR	OTG ID Change Rising-Edge Interrupt Request
0	R/WC†	IRIDF	OTG ID Change Falling-Edge Interrupt Request

NOTE:
† To clear this bit, write 0b1 to it.

12.6.6 UDC Frame Number Register (UDCFNR)

The UDC Frame Number register (UDCFNR), shown in Table 12-26, holds the 11-bit frame number contained in the last received SOF packet, and is used for isochronous transfers. If the SOF interrupt is enabled, a SOF interrupt is generated when the frame-number bits are updated.

Table 12-26. UDCFNR Bit Definitions

Physical Address 0x4060_0014	UDCFNR	USB Client Controller

User Settings																															
Bit	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																						FN									
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:11	—	—	reserved
10:0	R	FN	Frame number associated with last received SOF.

12.6.7 UDC Endpoint 0 Control/Status Register (UDCCSR0)

The UDC Endpoint 0 Control/Status register, shown in Table 12-27, contains eight bits to operate endpoint 0 (control endpoint).

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 12-27. UDCCSR0 Bit Definitions (Sheet 1 of 3)

Physical Address 0x4060_0100																						UDCCSR0										USB Client Controller									
User Settings																																									
Bit	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																						ACM	AREN	SA	RNE	FST	SST	DME	FTF	IPR	OPC										
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0										

Bits	Access	Name	Description
31:10	—	—	reserved
9	R/W	ACM	ACK Control Mode Enables user control of the ACK response to the status IN requests of SET_CONFIGURATION and SET_INTERFACE commands. When ACM is clear, the UDC automatically responds to the STATUS IN request following a SET_CONFIGURATION and SET_INTERFACE with an ACK. When ACM is set, the UDC responds to the STATUS IN request following a SET_CONFIGURATION and SET_INTERFACE command with a NAK until AREN is set. When the user sets AREN to 1, the UDC responds with an ACK to the next STATUS IN request. 0 = Send ACK response to SET_CONFIGURATION and SET_INTERFACE commands with no user intervention. 1 = Send NAK response to SET_CONFIGURATION and SET_INTERFACE commands until AREN = 1.
8	Read/ Write 1 to Set	AREN	ACK Response Enable When ACM = 1, the AREN bit enables user control of the ACK response to the status IN requests of SET_CONFIGURATION and SET_INTERFACE commands. When ACM is set, the UDC responds to the STATUS IN request following a SET_CONFIGURAION and SET_INTERFACE command with a NAK until AREN is set. When the user sets AREN to 1, the UDC responds with an ACK to the next STATUS IN request. AREN is cleared by the UDC when another SETUP command is received. 0 = Send NAK response to SET_CONFIGURATION and SET_INTERFACE commands. 1 = Send ACK response to SET_CONFIGURATION and SET_INTERFACE commands.
7	Read/ Write 1 to Clear	SA	Setup Active Indicates the current packet in the endpoint 0 receive FIFO is part of a USB SETUP command. This bit is active at the same time as Out-Packet Complete (OPC). Users must clear this bit by writing a 1 to it. SA must be cleared when OPC is cleared. 1 = Setup command is active on the USB.

Table 12-27. UDCCSR0 Bit Definitions (Sheet 2 of 3)

Physical Address 0x4060_0100 — **UDCCSR0** — **USB Client Controller**

Bit	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
User Settings																																
	reserved																						ACM	AREN	SA	RNE	FST	SST	DME	FTF	IPR	OPC
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
6	R	RNE	Receive FIFO Not Empty Indicates there is unread data in the endpoint 0 receive FIFO. This bit must be polled when the OPC bit is set to determine if there is any data in the receive FIFO. The receive FIFO must continue to be read until this bit clears, or data is lost. 0 = Receive FIFO empty. 1 = Receive FIFO not empty.
5	Read/ Write 1 to Set	FST	Force Stall Set this bit to 1 to force the UDC to issue a STALL handshake. After the UDC issues a STALL handshake for the current control transfer, the bit is cleared by the UDC because endpoint 0 cannot remain in a stalled condition. 1 = Force stall handshake.
4	Read/ Write 1 to Clear	SST	Sent Stall This bit is set by the UDC when it must abort the current control transfer by issuing a STALL handshake due to a protocol violation. When SST is set, UDCISR0[IR0] is set if the packet-complete endpoint-0 interrupt is enabled. Write 0b1 to SST to clear it. 1 = UDC sent stall handshake.
3	R/W	DME	DMA Enable Used for endpoint 0 OUT transactions only, to request DMA reading of endpoint 0 FIFO memory. DME is used by the UDC to control the endpoint packet-complete interrupt and DMA request. If the bit is clear, the packet-complete interrupt is asserted when the end-of-packet is received, and all of the received data is still in the receive FIFO. In this case, it is assumed the data is unloaded using the Intel XScale® technology, so an interrupt is generated instead of a DMA request. If DME is set, the packet-complete interrupt is not asserted, but a DMA request is generated to notify the DMA of the received data. DMA Enable can only be disabled when the DMA channel is stopped. 0 = Send data received interrupt after EOP received. 1 = Send data received DMA request after EOP received.

Table 12-27. UDCCSR0 Bit Definitions (Sheet 3 of 3)

Physical Address 0x4060_0100 — **UDCCSR0** — **USB Client Controller**

Bit	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
User Settings																																
	reserved																						ACM	AREN	SA	RNE	FST	SST	DME	FTF	IPR	OPC
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	Read 0/ Write 1 to Set	FTF	Flush Transmit FIFO Set this bit to reset the endpoint 0 transmit FIFO. FTF is reset by the UDC after the FIFO contents have been deleted. The endpoint 0 transmit FIFO is also flushed by the UDC after the UDC has received an OUT packet from the USB host controller. 1 = Flush the contents of transmit FIFO.
1	Read/ Write 1 to Set	IPR	IN Packet Ready Indicates a packet has been loaded and is ready for transmission. Users must set IPR only when a packet smaller than 16 bytes has been written to the endpoint 0 transmit FIFO. There is no need to set IPR when a maximum size packet is loaded for transmission. The UDC automatically clears IPR when the packet has been successfully transmitted or FTF has been set. When IPR is cleared by the UDC, the IR0[0] bit in UDCISR0 is set if the packet-complete endpoint 0 interrupt is enabled. IPR cannot be cleared by software. 1 = IN packet has been loaded and is ready for transmission.
0	Read/ Write 1 to Clear	OPC	OUT Packet Complete This bit is set by the UDC when it receives a valid token to endpoint 0. When OPC is set, the IR0[0] bit in the UDC Interrupt Status register 0 (UDCISR0) is set if the packet-complete endpoint 0 interrupt is enabled. OPC is cleared by writing a 1 to it, and it must not be cleared until all the receive data has been read from the endpoint 0 receive FIFO. If OPC is cleared before all the receive data is read, the receive FIFO is flushed and any data remaining in the FIFO is lost. The UDC does not enter the data phase of an endpoint 0 transaction until the OPC bit is cleared. **NOTE:** OPC is set when DME is set and must be cleared by software to enter the data phase of the endpoint 0 transaction. The UDC does not enter the data phase of an endpoint 0 transaction until the OPC bit is cleared, even when DME is set and the packet complete interrupt is not asserted. 0 = OUT packet not received. 1 = OUT packet received and ready for reading.

12.6.8 UDC Endpoints A–X Control Status Registers (UDCCRSA–UDCCRSX)

Each of the 23 programmable endpoints A–X use their Control/Status registers (UDCCSRA–UDCCSRX) to control the behavior of the endpoint and to report status for that endpoint after USB enumeration. The UDC Control Status registers for each endpoint contain ten bits to operate the respective endpoint.

12.6.8.1 Data Packet Error

The Data-Packet Error (DPE) bit is used for endpoints configured as Isochronous OUT endpoints only. The DPE bit indicates a PID, bit-stuffing, or CRC error has been detected on the active buffer of Isochronous OUT data. DPE is valid while PC is set. If an endpoint is configured as any type other than Isochronous or as an IN endpoint, DPE is not used.

12.6.8.2 Flush Endpoint FIFO

The Flush Endpoint FIFO (FEF) bit resets the endpoint transmit or receive FIFO by setting it. FEF is cleared by the UDC after the FIFO contents have been deleted.

12.6.8.3 Short Packet

The Short Packet (SP) bit indicates a packet smaller than the maximum packet size has been received and is ready for unloading, or has been loaded and is ready for transmission.

If an endpoint is configured as an IN endpoint, the SP bit must be set when the last byte of a transfer that is shorter than the maximum packet size has been loaded into the transmit buffer. Doing so tells the UDC that the packet is ready for transmitting. For maximum size packets, SP does not need to be set. Upon successful transmission of the data packet, the SP bit is cleared by the UDC.

If an endpoint is configured as an OUT endpoint, when the UDC sets the SP bit, it indicates that the last byte of a receive-data transfer that is smaller than the maximum size packet has been loaded into the receive FIFO. If SP is set and BNE/BNF is clear, a zero-length packet has been received. The SP bit is read-only for all OUT endpoints and is valid while the PC bit is set.

12.6.8.4 Buffer Not Empty/Buffer Not Full

The Buffer Not Empty/Buffer Not Full (BNE/BNF) bit indicates there is unread or untransmitted data in the current receive or transmit buffer. If an endpoint is configured as an IN endpoint, the BNE/BNF means that the buffer is not full, and if an endpoint is configured as an OUT endpoint, the BNE/BNF means that the buffer is not empty.

If the endpoint is configured as an IN endpoint, BNE/BNF clears when the endpoint buffer space has been filled with packet data. If the endpoint does not have double-buffering enabled, BNE/BNF clears when one complete packet of data has been loaded into the FIFO memory space. If the endpoint is configured as an IN endpoint and double-buffering is enabled, BNE/BNF indicates the status of the current buffer being loaded. If both buffers are loaded with complete packets of data, both the BNE/BNF and FS bits are 0. A complete packet of data is signified by either loading a maximum size packet or loading a short packet, and setting the SP bit.

If an endpoint is configured as an OUT endpoint, BNE/BNF must be polled when the PC bit is set to determine if there is any data in the FIFO that needs to be unloaded. Users must continue to read the Endpoint Data register until this bit clears, or data is lost. The UDC sets BNE/BNF when the receive FIFO has one complete data packet in it. If the endpoint is a bulk or interrupt endpoint, BNE/BNF is not set until the OUT packet has been error-checked by the UDC and found to be error-free. If the endpoint is an isochronous endpoint, BNE/BNF is not set until the OUT packet data has been error-checked by the UDC and DPE set to indicate any errors that may have been found. A complete packet is defined as a maximum size packet, a short packet, or a zero packet. BNE/BNF does not clear until the read of the current buffer is complete. If the endpoint has double-buffering enabled, BNE/BNF clears when the current buffer is empty and FS bit does not clear until neither buffer has data.

12.6.8.5 Sent STALL and Force STALL

The Sent Stall (SST) and Force Stall (FST) bits are used by endpoints configured as bulk or interrupt endpoints only. SST and FST are considered reserved for all endpoints configured as isochronous endpoints.

The UDC sets SST when it must abort the current transfer by issuing a STALL handshake to the USB host controller. The SST bit is cleared by writing a 1 to it. If the endpoint is configured as an IN endpoint, the transmit FIFO is flushed when SST is cleared. If the endpoint is configured as an OUT endpoint, no action is taken on the receive FIFO when SST is cleared. Any data in the endpoint receive FIFO remains valid and must be unloaded by software.

Note: Users must clear the SST bit before the USB host controller requests more data from IN endpoints or invalid or corrupted data may be sent to the host.

To force the UDC to issue a STALL handshake in response to all IN token requests or all OUT tokens, set the FST bit. The FST bit is cleared and the SST bit is set when the STALL handshake has actually been sent to the USB host controller and the STALL state entered (this may be delayed if the UDC is active when the FST bit is set). The STALL handshakes continue to be sent until the USB host controller clears the STALL feature for the particular endpoint. The actual stall condition in the UDC is cleared by the USB host controller using the CLEAR_FEATURE command.

12.6.8.6 DMA Enable

The DMA Enable (DME) bit is set to request DMA reading or writing of endpoint FIFO memory. For all endpoints configured as OUT endpoints, DME is used by the UDC to control the output of the endpoint packet-complete interrupt. If the DME bit is clear, the packet-complete interrupt is asserted when the end-of-packet is received, and all of the received data is still in the receive FIFO; data must be unloaded using the PXA27x processor. If DME is set, the packet complete interrupt is not asserted, but a DMA request is generated to notify the DMA of the received data.

If an endpoint is configured as an IN endpoint, DME is used by the UDC to enable the output of the DMA request. If the DME bit is clear, the DMA request is not generated, and users must use BNE/BNF and FS to determine when to load data into the endpoint FIFO memory for transmission. If the DME bit is set, the UDC generates a DMA request when there is space in the endpoint FIFO memory for one complete data packet. The UDC continues to issue a DMA request until the endpoint buffer(s) has been filled or the SP bit has been set. When DME is set, the PC bit is set at the end of the transmission of the packet; however, it is not required to clear PC before loading more data into the endpoint FIFO.

Note: Disable the DME bit only when the DMA channel is stopped.

12.6.8.7 Transmit/Receive NAK

The Transmit/Receive NAK (TRN) bit is set by the UDC when the host requests IN data and a complete packet is not ready for transmission or the host sends OUT data and the endpoint buffer(s) is full. The TRN bit is cleared by writing a 1 to it.

For IN endpoints, the TRN bit is set by the UDC when the UDC tries to transmit data but a complete packet of data has not been loaded into the endpoint buffer. If the endpoint is configured as a bulk or interrupt endpoint, the UDC issues NAK handshakes to IN tokens while the endpoint buffer continues to be empty. If the endpoint is configured as an isochronous endpoint, the UDC sends zero-size packets while the endpoint buffer is empty and, if the FIFO error interrupt is enabled, the UDC generates a FIFO error interrupt for the endpoint.

For OUT endpoints, the TRN bit is set by the UDC when the host sends OUT data while the endpoint buffer(s) is full. If the endpoint is configured as a bulk or interrupt endpoint, the UDC issues NAK handshakes to OUT tokens while the endpoint buffer(s) continues to be full. If the OUT endpoint is configured as an isochronous endpoint, isochronous data packets sent from the host while the endpoint buffers are full are dropped and, if the FIFO error interrupt is enabled, generate a FIFO error interrupt for the endpoint.

12.6.8.8 Packet Complete

The UDC sets the Packet Complete (PC) bit when an entire packet has been sent to or received from the USB host controller. PC can be used to validate the other status/error bits in the Endpoint Control/Status register. The PC bit is cleared by writing a 1 to it.

If an endpoint is configured as an OUT endpoint, the PC bit is set by the UDC when an entire packet has been received from the USB host controller, and generates a packet-complete interrupt or DMA request for the endpoint, according to the Packet Complete Interrupt Enable and DME bits. After reading all of the received data from the endpoint receive buffer, the PC bit must be cleared. Any data remaining in the endpoint receive buffer is deleted when the PC bit is cleared. The UDC issues NAK handshakes to all OUT tokens for any bulk or interrupt endpoint with the PC bit set and when the endpoint buffers are full.

If an endpoint is configured as an IN endpoint, the PC bit is set by the UDC after an entire packet has been sent to the USB host controller. Users can load more data into the transmit FIFO regardless of the state of PC. When a packet is received by the UDC, the PC is set by the UDC, and the corresponding PC interrupt bit in the UDCISR0 is set and an interrupt generated, if the interrupt is enabled and DME is clear. If the PC bit was set when a packet is received by the UDC, PC remains set and the corresponding PC interrupt bit in the UDCISR0 is set (if it has not been set already) and an interrupt generated, if the interrupt is enabled and DME is clear. If the PC bit was set when a packet is received by the UDC, the PC remains set and if DME is set, a DMA request is generated.

12.6.8.9 FIFO Service

The FIFO-service (FS) bit indicates there is room in the endpoint FIFO for more data to be loaded or more data in the endpoint FIFO to be unloaded.

If an endpoint is configured as an IN endpoint and has double-buffering enabled, the UDC sets FS when there is space for one or more data packets to be loaded into the transmit FIFO. FS clears when there are two complete packets of data in the transmit FIFO. If an IN endpoint has double-

buffering disabled, FS is active if there is less than one complete data packet in the transmit FIFO. FS clears when one complete data packet has been loaded into the FIFO. A complete packet of data is signified by either loading in a maximum size packet or by setting SP.

If an endpoint is configured as an OUT endpoint and has double-buffering enabled, the FS bit is set by the UDC when the receive FIFO has at least one complete data packet in it. FS does not clear until the read of the FIFO is complete and neither buffer has data. In the case of both buffers being empty, FS is not set until the UDC has checked the OUT packet for errors. If the OUT endpoint has double-buffering disabled, FS is set by the UDC when the receive FIFO has one complete data packet in it. FS does not clear until all of the data has been read from the FIFO. A complete packet can be the maximum size packet, a short packet, or a zero packet.

Table 12-28 shows how each of the control and status bits in the UDCCSRA–UDCCSRX registers are defined by endpoint direction.

Table 12-28. UDCCRSA–UDCCRSX Bit Definition by Endpoint Direction

Register Bit	IN Endpoint	OUT Endpoint
DPE	Not used	Isochronous receive data had PID, bit stuffing or CRC error
FEF	Flush the contents of the transmit FIFO	Flush the contents of the receive FIFO
SP	Short packet has been loaded and is ready for transmission	Short packet has been received and is ready for reading
BNE/BNF	Current transmit buffer full/not full	Current receive buffer empty/not empty
FST	Send STALL handshake to IN tokens	
SST	STALL handshake has been sent	
DME	DMA request asserted when transmit FIFO has room for one complete data packet	Interrupt asserted after EOP received or DMA request asserted after EOP received
EFE	FIFO underrun has occurred	FIFO overflow has occurred
PC	Qualification of other status/error bits. Transmit packet has been sent	Qualification of other status/error bits. Receive packet has been received
FS	Transmit FIFO has room/no room for new data	Receive FIFO has room/no room for new data

The register organization and individual bit definitions are shown in Table 12-29.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 12-29. UDCCRSA–UDCCRSX Bit Definitions (Sheet 1 of 2)

Physical Addresses 0x4060_0104–0x4060_015C — **UDCCSRA–UDCCSRX** — **USB Client Controller**

	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
User Settings																																
Bit	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																						DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:10	—	—	reserved
9	R	DPE	Data Packet Error (isochronous endpoints only) 0 = No error detected on OUT data packet. 1 = PID, bit stuffing, or CRC error was detected on OUT packet.
8	Read/ Write 1 to Set	FEF	Flush Endpoint FIFO For IN endpoints: 1 = Flush the contents of the transmit FIFO. For OUT endpoints: 1 = Flush the contents of the receive FIFO.
7	IN Endpoints: Read/Write 1 to Set OUT Endpoints: R	SP	Short Packet Control/Status For In endpoints: 1 = Short packet ready for transmission. For OUT endpoints: 1 = Short packet received and ready for reading.
6	R	BNE/BNF	Buffer Not Empty/Buffer Not Full For IN endpoints (Buffer Not Full): 0 = Current transmit buffer is full. 1 = Current transmit buffer is not full. For OUT endpoints (buffer not empty): 0 = Current receive buffer is empty. 1 = Current receive buffer is not empty.
5	R/W	FST	Force STALL 1 = Issue STALL handshakes to all IN tokens.
4	Read/ Write 1 to Clear	SST	Sent STALL 1 = STALL handshake was sent.
3	R/W	DME	DMA Enable For IN endpoints: 1 = Send DMA request when transmit FIFO has room for one packet. For OUT endpoints: 0 = Send data received interrupt after EOP received. 1 = Send data received DMA request after EOP received.

Table 12-29. UDCCRSA–UDCCRSX Bit Definitions (Sheet 2 of 2)

Physical Addresses 0x4060_0104–0x4060_015C — **UDCCSRA–UDCCSRX** — **USB Client Controller**

	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
User Settings																																
Bit	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																						DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	Read /Write 1 to Clear	TRN	Tx/Rx NAK For IN endpoints: 1 = Requested data packets cannot be transmitted because FIFO is empty. For OUT endpoints: 1 = Received data packets cannot be stored because FIFO is full.
1	Read /Write 1 to Clear	PC	Packet Complete For IN endpoints: 0 = Error/status bits are invalid. 1 = Transmit packet has been sent and error/status bits are valid. For OUT endpoints: 0 = Error/status bits are invalid. 1 = Receive packet has been received and error/status bits are valid.
0	R	FS	FIFO needs service For IN endpoints: 0 = Transmit FIFO has no room for new data. 1 = Transmit FIFO has room for at least 1 complete data packet. For OUT endpoints: 0 = Receive FIFO has no room for new data. 1 = Receive FIFO has at least 1 complete data packet.

12.6.9 UDC Byte Count Registers (UDCBCR0 and UDCBCRA–UDCBCRX)

The Byte Count registers (UDCBCR0 and UDCBCRA–UDCBCRX) (Table 12-30) maintain the remaining byte count in the active buffer of endpoint 0, and each programmable endpoint A–X is configured as an OUT endpoint. There is one Byte Count register for each endpoint, but the Byte Counter register is considered reserved for all endpoints configured as IN endpoints.

12.6.9.1 Byte Count

The Byte Count (BC) bits are updated by the UDC after each byte is read by users. Upon receiving an interrupt that indicates the endpoint has data or after the PC bit in the corresponding UDCCSR has been set, the Byte Count register can be read to determine the number of bytes that still need to be read from the endpoint receive FIFO. The Byte Count register is decremented by the number of valid data bytes read (usually four) each time data is read from the endpoint FIFO memory. The byte count cannot decrement to less than 0.

If double-buffering is not enabled for an OUT endpoint, the Byte Count register indicates the total number of bytes that need to be read from the endpoint buffer. The Byte Count register is decremented each time data is read from the buffer and is cleared when the buffer is empty. The UDC clears the BNE/BNF bit in the endpoint Control/Status register when the byte count is 0. Continuing to read the endpoint FIFO memory after the byte count is 0 results in reading unknown data.

If double-buffering is enabled for an OUT endpoint, the Byte Count register indicates the number of bytes that need to be read from the active endpoint buffer. The Byte Count register is decremented each time data is read from the buffer and is cleared when the buffer is empty. The BNE/BNF bit in the Endpoint Control/Status register is cleared by the UDC when the byte count of the active endpoint buffer is 0. If the second buffer contains data, the FS bit continues to be set to indicate there is still data in the endpoint FIFO space. The BNE/BNF bit is set when the second buffer is the active endpoint buffer, but remains 0 while the first buffer is the active endpoint buffer. Continuing to read the endpoint FIFO memory after the byte count is 0 results in reading unknown data, and the second buffer is not read. The PC bit in the Endpoint Control/Status register must be cleared to: load the byte count for the second buffer into the Byte-Count register; set the BNE/BNF bit to indicate the status of the second buffer; and enable the second buffer data for reading. See Section 12.4.2 for more information on the Byte Count register.

This is a read-only register. Ignore reads from reserved bits.

Table 12-30. UDCBCR0 and UDCBCRA–UDCBCRX Bit Definitions

Physical Addresses 0x4060_0200–0x4060_025C — **UDCBCR0–UDCBCRX** — **USB Client Controller**

	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
User Settings																																
Bit	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																						BC									
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:10	—	—	reserved
9:0	R	BC	Byte Count

12.6.10 UDC Data Registers (UDCDR0 and UDCDRA-–UDCDRX)

Each UDC Endpoint Data register is a 32-bit by maximum-packet-size-entry bidirectional FIFO. When the USB host controller transmits data to UDC endpoint 0 or endpoint A–X, the appropriate UDC Endpoint register is read to access the data. When the UDC is sending data to the USB host controller, the data to be sent must be placed into the appropriate UDC Endpoint register. Although read and write operations can be performed on a single FIFO during various points in a control sequence, the FIFO can not be read and written at the same time. The UDC controls the direction that the FIFO is flowing. For more details on accessing the endpoint FIFO memory, refer to Section 12.4.2.

For endpoint 0, normally, the UDC is in an idle state, waiting for the USB host controller to send commands. When this happens, the UDC fills the endpoint 0 receive FIFO with the command from the host, and the command is read from the FIFO once it has arrived. The only time software can write the endpoint 0 transmit FIFO is when a GET_DESCRIPTOR, vendor, or class-specific command from the host has been received that requires a transmission in response.

For the programmable endpoints A–X, if an endpoint is configured as a bulk, interrupt, or isochronous IN endpoint, data can be loaded using DMA or direct processor writes. If it is double-buffered, up to two data packets can be loaded for transmission.

If an endpoint is configured as a bulk or interrupt OUT endpoint, the UDC generates either an interrupt or DMA request when the EOP is received and the data has been checked for errors. If the endpoint has double-buffering enabled, up to two data packets may be ready. The data can be removed from the endpoint receive FIFO using DMA or by direct read from the processor.

Note: If the allocated memory space is still occupied with previously received USB data, the UDC issues a NAK to the USB host controller the next time it sends an OUT packet to this endpoint. This NAK condition remains in place until a full packet space is available in the endpoint memory allocated to the endpoint.

If the endpoint is an isochronous OUT endpoint, the UDC generates either an interrupt or DMA request as soon as the EOP is received and the data has been checked for PID, bit stuffing, and CRC errors. If it is double-buffered, up to two packets of data may be ready. The data can be removed from the endpoint receive FIFO using DMA or by direct read from the processor.

Note: If the allocated memory space is still occupied with previously received USB data, the UDC does not issue a NAK to the USB host controller the next time it sends an OUT packet to this endpoint, but the data is lost and an overflow condition is indicated in the Endpoint Control/Status register. The overflow condition remains in place until a full packet space is available in the endpoint memory allocated to this endpoint.

The register organization and individual bit definitions are shown in Table 12-31.

Table 12-31. UDCDR0 and UDCDRA–UDCDRX Bit Definitions

Physical Addresses 0x4060_0300–0x4060_035C																UDCDR0, UDCDRA–UDCDRX																USB Client Controller															
User Settings																																														
Bit	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													
	ED																																													
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?													

Bits	Access	Name	Description
31:0	IN Endpoints: Transmit FIFO has no room for new data OUT Endpoints: R Endpoint 0: R/W	ED	Endpoint Data

12.6.11 UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)

The UDC Configuration registers (UDCCRA–UDCCRX, shown in Table 12-32) define and enable the programmable endpoints that are active for each particular configuration/interface/alternate interface setting combination. The 23 programmable endpoints can each be enabled for one programmable configuration and interface. The maximum packet size, endpoint number, type, direction, and buffering for each endpoint is determined by software, allowing selection of which endpoints are used for each configuration and interface, and allocate the FIFO space as needed.

At power on, the UDC is disabled and all configurations, interfaces, and alternate interface settings are disabled. Users must program the Endpoint Configuration registers before enabling the UDC. When the UDC is enabled, the endpoint configuration information is checked and if valid, loaded into the USB interface and enabled for USB operation. The UDC can then be enumerated and configured by the USB host controller. Once the UDC has been enabled, the endpoint Configuration registers become read-only and cannot be changed until the UDC is disabled. If an error is detected in the memory allocation when the configuration information is checked, UDCCR[EMCE] is set, the endpoint configuration is not loaded, and the UDC is not enabled for USB operation.

The UDC Endpoint Configuration registers can be changed after the UDC has been enabled for USB operation, but the UDC must be disabled before the Configuration registers can be written. Disabling the UDC resets the USB configuration, interface, alternate interface, and address assigned to the UDC by the USB host controller, and disables all USB features enabled by the USB host controller. The UDC must be reset and re-enumerated by the USB host controller whenever the UDC is disabled and re-enabled by users. See Figure 12-6 for more information on the UDC configuration programming sequence.

Table 12-32. UDCCRA–UDCCRX Bit Definitions (Sheet 1 of 2)

Physical Addresses 0x4060_0404–0x4060_045C — **UDCCRA–UDCCRX** — **USB Client Controller**

Bit	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
User Settings																																
	reserved					CN		IN			AISN		EN				ET		ED	MPS										DE	EE	
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

Bits	Access	Name	Description
31:27	—	—	reserved
26:25	If UDE = *0*: R/W If UDE = *1:* R	CN	Configuration Number Must be set before the UDC is enabled. Refer to Figure 12-6 for more details on the Configuration register programming sequence. Each endpoint A–X can be assigned to Configuration 1, 2, or 3. Each endpoint A–X can be assigned to only one configuration/interface/alternate interface setting combination and is active only when the USB host controller configures the UDC for that configuration, interface, and alternate-interface setting. 1–3 = USB Configuration Number
24:22	If UDE = *0*: R/W If UDE = *1:* R	IN	Interface Number Must be set before the UDC is enabled. Refer to Figure 12-6 for more details on the Configuration register programming sequence. Each endpoint A–X can be assigned to Interface 1–7. Each endpoint A–X can be assigned to only one configuration/interface/alternate interface setting combination and is active only when the USB host controller configures the UDC for that configuration, interface, and alternate-interface setting. 1–7 = USB Interface Number
21:19	If UDE = *0*: R/W If UDE = *1:* R	AISN	Alternate Interface Number Alternate interface setting numbers for endpoints A–X must be set before the UCD is enabled. Refer to Figure 12-6 for more details on the Configuration register programming sequence. Each endpoint A–X can be assigned to Alternate Interface setting 1–7. Each endpoint A–X can be assigned to only one configuration/interface/alternate interface setting combination and is active only when the USB host controller configures the UDC for that configuration, interface, and alternate-interface setting. 1–7 = USB Interface Alternate Interface Setting Number
18:15	If UDE = *0*: R/W If UDE = *1:* R	EN	Endpoint Number: Defines the endpoint number. Each endpoint A–X can be programmed to respond to USB endpoint numbers 1–15. More than one endpoint can be programmed to the same endpoint number, but the endpoints cannot be active in the same configuration. For example, if endpoint A and endpoint B can both be assigned endpoint number 1, but with endpoint A in Configuration 1 and endpoint B in configuration 2. 1–15 = USB endpoint number
14:13	If UDE = *0*: R/W If UDE = *1:* R	ET	USB Endpoint Type 0b11 = Interrupt 0b10 = Bulk 0b01 = Isochronous 0b00 = Not used

Table 12-32. UDCCRA–UDCCRX Bit Definitions (Sheet 2 of 2)

Physical Addresses 0x4060_0404–0x4060_045C — **UDCCRA–UDCCRX** — **USB Client Controller**

User Settings

Bit	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					CN	IN		AISN		EN		ET	ED	MPS																DE	EE	
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

Bits	Access	Name	Description
12	If UDE = *0*: R/W If UDE = *1*: R	ED	USB Endpoint Direction Determines whether the endpoint is a an IN endpoint, which sends data from the UDC to the USB host controller, or an OUT endpoint, which receives data from the USB host controller. 0 = OUT 1 = IN
11:2	If UDE = *0*: R/W If UDE = *1* R	MPS	Maximum Packet Size Valid values are defined by endpoint type. For interrupt endpoint: 1–64 For bulk endpoint: 8, 16, 32, 64 For isochronous endpoint: 1–1023
1	If UDE = *0*: R/W If UDE = *1*: R	DE	Double-Buffering Enable Setting this bit allocates two transmit or receive buffers in the endpoint memory. If an endpoint has DE set, the endpoint is allocated at least double the number of bytes of FIFO space as indicated by the maximum packet size. If an IN endpoint is double-buffered, endpoint data can be loaded into the second transmit buffer while the UDC is transmitting from the first. If an IN endpoint is not double-buffered, the data cannot be loaded into the buffer until the transmission is completed. If an OUT endpoint is double-buffered, endpoint data can be unloaded from the first buffer while the UDC is receiving data and loading it into the second. If an OUT endpoint is not double-buffered, the data must be unloaded from the FIFO before more data can be received from the USB host controller. See Section 12.4.2 for more information on allocating endpoint memory. 0 = Double-buffering disabled (endpoint is allocated 1 buffer for endpoint data). 1 = Double-buffering is enabled (endpoint is allocated 2 buffers for endpoint data).
0	If UDE = *0*: R/W If UDE = *1*: R	EE	Endpoint Enable Set this bit to enable the endpoint for USB operation. If an endpoint is enabled, the data in the Configuration register is checked and loaded into the USB interface block to enable the endpoint for USB operation. If an endpoint is not enabled, the data in the Configuration register is not checked and is not loaded into the USB interface block, and will not be used during USB operation. 0 = Endpoint is disabled for USB operation. 1 = Endpoint is enabled for USB operation.

12.7 Register Summary

The USB host controller controls and communicates all configuration, request/service, and status reporting to the UDC via the USB. Several registers are available to users to control the interfacing of the UDC to software. A control register enables the UDC and monitors USB activity. Two control registers enable the various interrupt sources that exist within the UDC. Three status registers indicate the state of the various interrupt sources and the current frame number. Endpoint 0 has one control/status register, one byte-count register, and one register for accessing USB data. Endpoints A–X each have one control/status register, one configuration control register, one byte-count register, and one register for accessing USB data.

Table 12-33 shows the registers associated with the UDC and the memory-mapped addresses to access them.

Table 12-33. USB Client Controller Register Summary (Sheet 1 of 4)

Address	Name	Description	Page
0x4060_0000	UDCCR	UDC Control register	12-31
0x4060_0004	UDCICR0	UDC Interrupt Control register 0	12-35
0x4060_0008	UDCCIR1	UDC Interrupt Control register 1	12-35
0x4060_000C	UDCISR0	UDC Interrupt Status register 0	12-50
0x4060_0010	UDCSIR1	UDC Interrupt Status register 1	12-50
0x4060_0014	UDCFNR	UDC Frame Number register	12-53
0x4060_0018	UDCOTGICR	UDC OTG Interrupt Control register	12-35
0x4060_001C	UDCOTGISR	UDC OTG Interrupt Status register	12-50
0x4060_0020	UP2OCR	USB Port 2 Output Control register	12-41
0x4060_0024	UP3OCR	USB Port 3 Output Control register	12-47
0x4060_0028–0x4060_00FC	—	reserved	
0x4060_0100	UDCCSR0	UDC Control/Status register—Endpoint 0	12-54
0x4060_0104	UDCCSRA	UDC Control/Status register—Endpoint A	12-57
0x4060_0108	UDCCSRB	UDC Control/Status register—Endpoint B	12-57
0x4060_010C	UDCCSRC	UDC Control/Status register—Endpoint C	12-57
0x4060_0110	UDCCSRD	UDC Control/Status register—Endpoint D	12-57
0x4060_0114	UDCCSRE	UDC Control/Status register—Endpoint E	12-57
0x4060_0118	UDCCSRF	UDC Control/Status register—Endpoint F	12-57
0x4060_011C	UDCCSRG	UDC Control/Status register—Endpoint G	12-57
0x4060_0120	UDCCSRH	UDC Control/Status register—Endpoint H	12-57
0x4060_0124	UDCCSRI	UDC Control/Status register—Endpoint I	12-57
0x4060_0128	UDCCSRJ	UDC Control/Status register—Endpoint J	12-57
0x4060_012C	UDCCSRK	UDC Control/Status register—Endpoint K	12-57
0x4060_0130	UDCCSRL	UDC Control/Status register—Endpoint L	12-57
0x4060_0134	UDCCSRM	UDC Control/Status register—Endpoint M	12-57
0x4060_0138	UDCCSRN	UDC Control/Status register—Endpoint N	12-57
0x4060_013C	UDCCSRP	UDC Control/Status register—Endpoint P	12-57

Table 12-33. USB Client Controller Register Summary (Sheet 2 of 4)

Address	Name	Description	Page
0x4060_0140	UDCCSRQ	UDC Control/Status register—Endpoint Q	12-57
0x4060_0144	UDCCSRR	UDC Control/Status register—Endpoint R	12-57
0x4060_0148	UDCCSRS	UDC Control/Status register—Endpoint S	12-57
0x4060_014C	UDCCSRT	UDC Control/Status register—Endpoint T	12-57
0x4060_0150	UDCCSRU	UDC Control/Status register—Endpoint U	12-57
0x4060_0154	UDCCSRV	UDC Control/Status register—Endpoint V	12-57
0x4060_0158	UDCCSRW	UDC Control/Status register—Endpoint W	12-57
0x4060_015C	UDCCSRX	UDC Control/Status register—Endpoint X	12-57
0x4060_0160–0x4060_01FC	—	reserved	
0x4060_0200	UDCBCR0	UDC Byte Count register—Endpoint 0	12-63
0x4060_0204	UDCBCRA	UDC Byte Count register—Endpoint A	12-63
0x4060_0208	UDCBCRB	UDC Byte Count register—Endpoint B	12-63
0x4060_020C	UDCBCRC	UDC Byte Count register—Endpoint C	12-63
0x4060_0210	UDCBCRD	UDC Byte Count register—Endpoint D	12-63
0x4060_0214	UDCBCRE	UDC Byte Count register—Endpoint E	12-63
0x4060_0218	UDCBCRF	UDC Byte Count register—Endpoint F	12-63
0x4060_021C	UDCBCRG	UDC Byte Count register—Endpoint G	12-63
0x4060_0220	UDCBCRH	UDC Byte Count register—Endpoint H	12-63
0x4060_0224	UDCBCRI	UDC Byte Count register—Endpoint I	12-63
0x4060_0228	UDCBCRJ	UDC Byte Count register—Endpoint J	12-63
0x4060_022C	UDCBCRK	UDC Byte Count register—Endpoint K	12-63
0x4060_0230	UDCBCRL	UDC Byte Count register—Endpoint L	12-63
0x4060_0234	UDCBCRM	UDC Byte Count register—Endpoint M	12-63
0x4060_0238	UDCBCRN	UDC Byte Count register—Endpoint N	12-63
0x4060_023C	UDCBCRP	UDC Byte Count register—Endpoint P	12-63
0x4060_0240	UDCBCRQ	UDC Byte Count register—Endpoint Q	12-63
0x4060_0244	UDCBCRR	UDC Byte Count register—Endpoint R	12-63
0x4060_0248	UDCBCRS	UDC Byte Count register—Endpoint S	12-63
0x4060_024C	UDCBCRT	UDC Byte Count register—Endpoint T	12-63
0x4060_0250	UDCBCRU	UDC Byte Count register—Endpoint U	12-63
0x4060_0254	UDCBCRV	UDC Byte Count register—Endpoint V	12-63
0x4060_0258	UDCBCRW	UDC Byte Count register—Endpoint W	12-63
0x4060_025C	UDCBCRX	UDC Byte Count register—Endpoint X	12-63
0x4060_0260–0x4060_02FC	—	reserved	
0x4060_0300	UDCDR0	UDC Data register—Endpoint 0	12-63
0x4060_0304	UDCDRA	UDC Data register—Endpoint A	12-63
0x4060_0308	UDCDRB	UDC Data register—Endpoint B	12-63

Table 12-33. USB Client Controller Register Summary (Sheet 3 of 4)

Address	Name	Description	Page
0x4060_030C	UDCDRC	UDC Data register—Endpoint C	12-63
0x4060_0310	UDCDRD	UDC Data register—Endpoint D	12-63
0x4060_0314	UDCDRE	UDC Data register—Endpoint E	12-63
0x4060_0318	UDCDRF	UDC Data register—Endpoint F	12-63
0x4060_031C	UDCDRG	UDC Data register—Endpoint G	12-63
0x4060_0320	UDCDRH	UDC Data register—Endpoint H	12-63
0x4060_0324	UDCDRI	UDC Data register—Endpoint I	12-63
0x4060_0328	UDCDRJ	UDC Data register—Endpoint J	12-63
0x4060_032C	UDCDRK	UDC Data register—Endpoint K	12-63
0x4060_0330	UDCDRL	UDC Data register—Endpoint L	12-63
0x4060_0334	UDCDRM	UDC Data register—Endpoint M	12-63
0x4060_0338	UDCDRN	UDC Data register—Endpoint N	12-63
0x4060_033C	UDCDRP	UDC Data register—Endpoint P	12-63
0x4060_0340	UDCDRQ	UDC Data register—Endpoint Q	12-63
0x4060_0344	UDCDRR	UDC Data register—Endpoint R	12-63
0x4060_0348	UDCDRS	UDC Data register—Endpoint S	12-63
0x4060_034C	UDCDRT	UDC Data register—Endpoint T	12-63
0x4060_0350	UDCDRU	UDC Data register—Endpoint U	12-63
0x4060_0354	UDCDRV	UDC Data register—Endpoint V	12-63
0x4060_0358	UDCDRW	UDC Data register—Endpoint W	12-63
0x4060_035C	UDCDRX	UDC Data register—Endpoint X	12-63
0x4060_0360–0x4060_03FC	—	reserved	
0x4060_0400	—	reserved	
0x4060_0404	UDCCRA	UDC Configuration register—Endpoint A	12-65
0x4060_0408	UDCCRB	UDC Configuration register—Endpoint B	12-65
0x4060_040C	UDCCRC	UDC Configuration register—Endpoint C	12-65
0x4060_0410	UDCCRD	UDC Configuration register—Endpoint D	12-65
0x4060_0414	UDCCRE	UDC Configuration register—Endpoint E	12-65
0x4060_0418	UDCCRF	UDC Configuration register—Endpoint F	12-65
0x4060_041C	UDCCRG	UDC Configuration register—Endpoint G	12-65
0x4060_0420	UDCCRH	UDC Configuration register—Endpoint H	12-65
0x4060_0424	UDCCRI	UDC Configuration register—Endpoint I	12-65
0x4060_0428	UDCCRJ	UDC Configuration register—Endpoint J	12-65
0x4060_042C	UDCCRK	UDC Configuration register—Endpoint K	12-65
0x4060_0430	UDCCRL	UDC Configuration register—Endpoint L	12-65
0x4060_0434	UDCCRM	UDC Configuration register—Endpoint M	12-65
0x4060_0438	UDCCRN	UDC Configuration register—Endpoint N	12-65
0x4060_043C	UDCCRP	UDC Configuration register—Endpoint P	12-65

Table 12-33. USB Client Controller Register Summary (Sheet 4 of 4)

Address	Name	Description	Page
0x4060_0440	UDCCRQ	UDC Configuration register—Endpoint Q	12-65
0x4060_0444	UDCCRR	UDC Configuration register—Endpoint R	12-65
0x4060_0448	UDCCRS	UDC Configuration register—Endpoint S	12-65
0x4060_044C	UDCCRT	UDC Configuration register—Endpoint T	12-65
0x4060_0450	UDCCRU	UDC Configuration register—Endpoint U	12-65
0x4060_0454	UDCCRV	UDC Configuration register—Endpoint V	12-65
0x4060_0458	UDCCRW	UDC Configuration register—Endpoint W	12-65
0x4060_045C	UDCCRX	UDC Configuration register—Endpoint X	12-65
0x4060_0460–0x406F_FFFC	—	reserved	

AC ‘97 Controller 13

This chapter describes the Audio Codec ‘97 (AC ’97) controller included in the PXA27x processor.

13.1 Overview

The AC ’97 controller supports the *Audio Codec '97 Component Specification*[1], Revision 2.0, features listed in Section 13.2. The AC-link is a synchronous, fixed-rate serial bus interface to the digital AC ’97 controller for transferring digital audio, modem, microphone input (MIC-in), Codec register control, and status information.

The AC ’97 Codec sends the digitized audio samples to the AC ’97 controller, which stores them in memory. For playback or synthesized audio production, the processor retrieves stored audio samples and sends them to the Codec through the AC-link. The external digital-to-analog converter (DAC) in the Codec then converts the audio sample to an analog audio waveform.

This chapter describes the programming model for the AC ’97 controller. The information in this chapter requires an understanding of the AC ’97 specification, Revision 2.0.

Note: The AC ’97 controller and the I^2S controller cannot be used at the same time.

13.2 Features

The PXA27x processor’s AC ’97 controller supports the following AC ’97 features:

- Independent channels for stereo pulse code modulation (PCM) in, stereo PCM out, modem out, modem-in and mono MIC-in

  All of the above channels support only 16-bit samples in hardware. Samples less than 16 bits are supported through software.
- Multiple sample rate AC ’97 2.0 Codecs (48 kHz and below). The AC ’97 controller depends on the Codec to control the varying rate.
- Read/write access to AC ’97 registers
- Secondary Codec support
- Three receive FIFOs (32-bit, 16 entries)
- Two transmit FIFOs (32-bit, 16 entries)
- Optional AC97_SYSCLK output (support for Codecs without oscillators or crystals)

The AC ’97 controller does not support the following optional AC ’97 Revision 2.0 features:

- Double-rate sampling (n+1 sample for PCM L, R and C)
- 18- and 20-bit sample lengths

1. The AC ‘97 specification is available from http://www.intel.com/labs/media/audio.

13.3 Signal Descriptions

The AC ’97 signals form the AC-link, which is a point-to-point synchronous serial interconnect that supports full-duplex data transfers. All digital audio streams, modem line Codec streams, and command/status information are communicated over the AC-link. The AC-link uses general-purpose I/Os (GPIOs). Software must reconfigure the GPIOs to use them as the AC-link. The AC-link pins are listed and described in Table 13-1.

Table 13-1. AC ’97 Controller I/O Signal Descriptions

Name	Type	Description
AC97_RESET_n	Output	Active-low Codec reset. The Codec’s registers are reset when AC97_RESET_n is asserted.
AC97_BITCLK	Input	12.288-MHz bit-rate clock
AC97_SYNC	Output	48-kHz frame indicator and synchronizer
AC97_SDATA_OUT	Output	Serial audio output data to Codec for digital-to-analog conversion
AC97_SDATA_IN_0	Input	Serial audio input data from primary Codec
AC97_SDATA_IN_1	Input	Serial audio input data from secondary Codec
AC97_SYSCLK	Output	Optional 24.576-MHz clock output

13.3.1 Signal Configuration Steps

To configure the AC-link, perform the following steps in order:

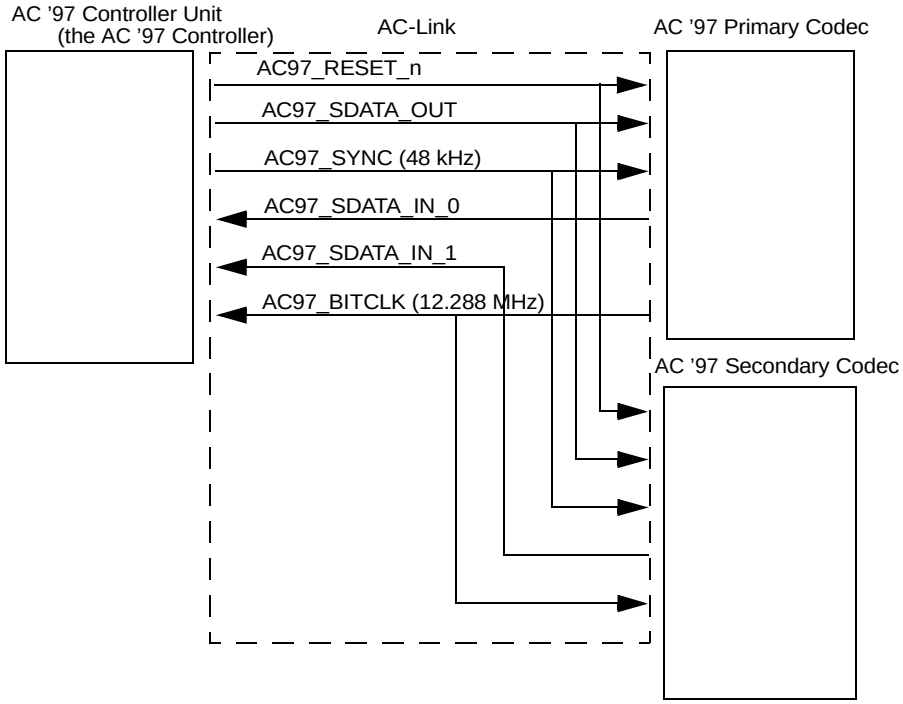
1. Configure AC97_SYSCLK, AC97_SYNC, AC97_RESET_n, and AC97_SDATA_OUT as outputs and configure AC97_BITCLK, AC97_SDATA_IN_0, and AC97_SDATA_IN_1 as inputs.
2. Deassert the AC97_RESET_n signal by setting the GCR[nCRST] bit (see Section 13.7.1).

Note: Refer to Section 24, “General-Purpose I/O Controller” for details on programing the GPDR and GAFR registers for use with the AC ’97 controller.

13.3.2 Example AC-Link

Figure 13-1 shows an example interconnect for an AC-link. The AC ’97 controller supports one or two Codecs on the AC-link. AC97_SDATA_IN_1 is not needed if only a single Codec is connected. The AC ‘97 controller can be optionally programmed to supply the AC97_SYSCLK (not shown).

Figure 13-1. Data Transfer through the AC-Link



13.4 AC-Link Digital Serial Interface Protocol

Each AC '97 Codec incorporates a five-pin digital serial interface that links it to the AC '97 controller. AC-link is a full-duplex, fixed-clock, pulse code modulation (PCM) digital stream. It employs a time-division-multiplexed (TDM) scheme to handle control register accesses and multiple input and output audio streams. The AC-link architecture divides each audio frame into 12 outgoing and 12 incoming data streams. Each stream has 20-bit sample resolution and requires a digital-to-analog converter (DAC) and an analog-to-digital converter (ADC) with a minimum 16-bit resolution. The AC '97 controller supports the data streams shown in Table 13-2.

Table 13-2. Supported Data Stream Formats (Sheet 1 of 2)

Channel	Slots	Comments
PCM Playback	Two output slots	Two-channel composite PCM output stream
PCM Record data	Two input slots	Two-channel composite PCM input stream
Codec control	Two output slots	Control register write port
Codec status	Two input slots	Control register read port
Modem Line Codec Output	One output slot	Modem line Codec DAC input stream
Modem Line Codec Input	One input slot	Modem line Codec ADC output stream

Table 13-2. Supported Data Stream Formats (Sheet 2 of 2)

Channel	Slots	Comments
Dedicated Microphone Input	One input slot	Dedicated microphone input stream in support of stereo acoustic echo canceller (AEC) and other voice applications
I/O Control	One output slot	One slot dedicated to general-purpose outputs (GPOs) on the modem Codec
I/O Status	One input slot	One slot dedicated to status from general-purpose inputs (GPIs) in the modem Codec. Data is returned on every frame.

The AC ’97 controller provides synchronization for all data transactions on the AC-link. A data transaction is made up of 256 bits of information broken up into groups of 13 time slots and is called a *frame*. Time slot 0 is called the *tag phase* and is 16 bits long. The other 12 time slots are called the *data phase*. The tag phase contains one bit that identifies a valid frame and 12 bits that identify the time slots in the data phase that contain valid data. Each time slot in the data phase is 20 bits long (see Figure 13-2).

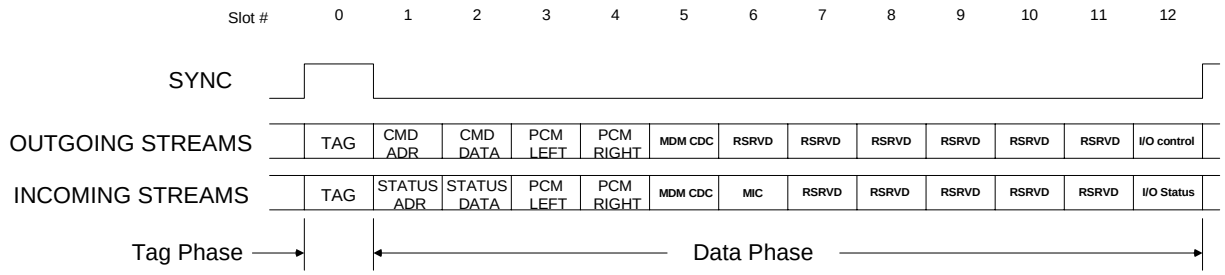
A frame begins when AC97_SYNC goes high. The amount of time that AC97_SYNC is high corresponds to the tag phase. AC ’97 frames occur at fixed 48-kHz intervals and are synchronous to the 12.288-MHz bit rate clock, AC97_BITCLK.

The AC ‘97 controller and Codec use the AC97_SYNC and AC97_BITCLK to determine when to send transmit data and when to sample receive data. A transmitter toggles the serial data stream on each rising edge of AC97_BITCLK and a receiver samples the serial data stream on falling edges of AC97_BITCLK. The transmitter must tag the valid slots in its serial data stream. The valid slots are tagged in slot 0.

Serial data on the AC-link is ordered most significant bit (MSB) to least significant bit (LSB). The tag phase’s first bit is bit 15 and the first bit of each slot in data phase is bit 19. The last bit in any slot is bit 0.

Figure 13-2 shows the organization of the tag phase and the data phase for both the controller and the Codec. Also shown in Figure 13-2 are the slot definitions the AC ’97 controller supports.

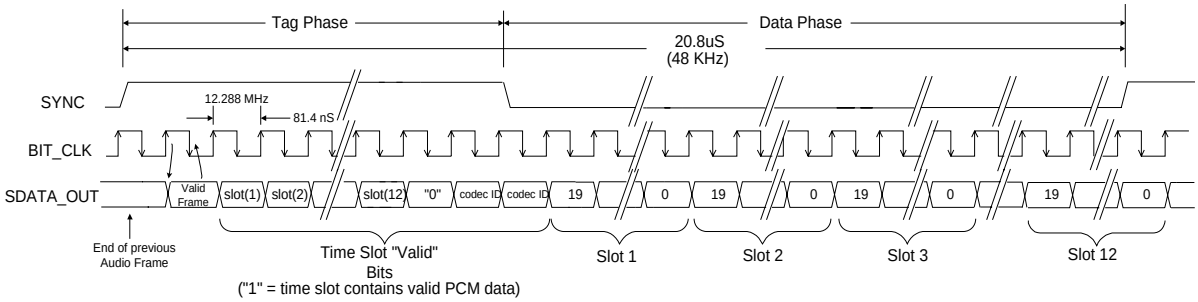
Figure 13-2. AC ’97 Standard Bidirectional Audio Frame



13.4.1 AC-Link Audio Output Frame (AC97_SDATA_OUT)

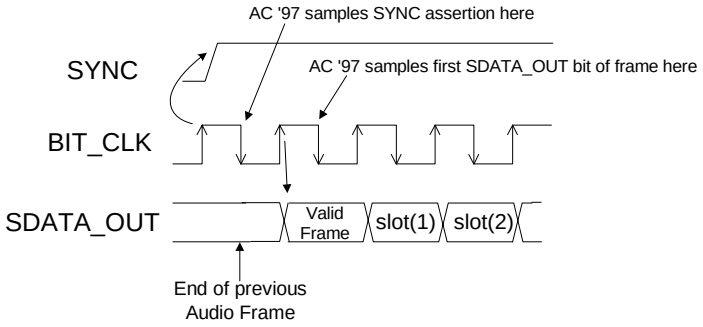
The audio-output-frame data stream corresponds to the multiplexed bundles that make up the digital output data targeting the AC ’97 DAC inputs and control registers. Each audio output frame supports up to twelve 20-bit outgoing data time slots. The AC ’97 controller does not generate samples larger than 16 bits. The four least significant bits are padded with zeroes. Figure 13-3 illustrates the time-slot-based AC-link protocol.

Figure 13-3. AC-Link Audio Output Frame



A new audio output frame begins with a low-to-high AC97_SYNC transition synchronous to AC97_BITCLK's rising edge. AC97_BITCLK's falling edge immediately follows and the AC '97 Codec samples the AC97_SYNC's assertion. AC97_BITCLK's falling edge marks the time when both sides of AC-link are aware of the start of a new audio frame. On the next rising edge of AC97_BITCLK, the AC '97 controller toggles AC97_SDATA_OUT into the first bit position of slot 0 (the valid frame bit). Each new bit position is presented to AC-link on the rising edge of AC97_BITCLK and then sampled by the AC '97 Codec on the following falling edge of AC97_BITCLK. This sequence ensures that data transitions and subsequent sample points for both incoming and outgoing data streams are time-aligned.

Figure 13-4. Start of Audio Output Frame



The AC97_SDATA_OUT composite stream is MSB-justified (MSB first). The AC '97 controller fills all non-valid slot bit positions with zeroes. If fewer than 20 valid bits exist in an assigned valid time slot, then the AC '97 controller fills all trailing non-valid bit positions of the 20-bit slot with zeroes.

For example, if a 16-bit sample stream is being played to an AC '97 DAC, the first 16 bit positions are presented to the DAC MSB-justified. They are followed by the next four bit positions that the the AC '97 controller fills with zeroes. This process ensures that the least significant bits do not introduce any DC biasing, regardless of the implemented DAC's resolution (16-, 18-, or 20-bit).

Note: When the AC '97 controller transmits mono audio sample streams, software must ensure that the left and right sample stream time slots are filled with identical data.

13.4.1.1 Slot 0: Tag Phase

In slot 0, the first bit is a global bit (AC97_SDATA_OUT slot 0, bit 15) that flags the validity for the entire audio frame. If the valid frame bit is a 1, the current audio frame contains at least one slot time of valid data. The next 12 bit positions sampled by the AC '97 controller indicate which of the corresponding 12 time slots contain valid data. Bits 0 and 1 of slot 0 are used as Codec ID bits for I/O reads and writes to the Codec registers, as described in the next section. In this way, data streams of differing sample rates can be transmitted across AC-link at its fixed 48-kHz audio frame rate.

Codec Ready, sent by the Codec on its data out stream in slot 0, bit 15, is not expected to change during normal operation. The AC '97 specification, revision 2.0, requires that a Codec only change its Codec Ready status in response to a power-off (PR) state change issued by the AC '97 controller. The controller hardware by itself does not monitor the Codec Ready for sending or receiving data. The controller stores Codec Ready in GSR[PCRDY] for a primary Codec and GSR[SCRDY] for a secondary Codec only for software to trigger a DMA or a programmed I/O operation. The controller only samples Codec Ready valid once and then ignores it for subsequent frames. Codec Ready is only resampled after a PR state change.

13.4.1.2 Slot 1: Command Address Port

The command port controls features and monitors status for AC '97 functions including, but not limited to, mixer settings and power management (refer to the AC '97 specification, revision 2.0, for more details).

The control-interface architecture supports up to 64 16-bit read/write registers, addressable on even-byte boundaries. Only accesses to even registers (0x00, 0x02, and so forth.) are valid. Accesses to odd registers (0x01, 0x03, ...) are not valid.

Audio output frame slot 1 communicates control register address and write/read command information to the AC '97 controller.

When two Codecs are connected to the single AC97_SDATA_OUT, the AC '97 controller hardware uses the following mechanism to access the primary and secondary Codecs individually:

- When software accesses the primary Codec, the AC '97 controller hardware configures the outgoing frame as follows:
  - In slot 0, the valid bits for slots 1 and 2 are set.
  - In slot 1, bit 19 is set (read) or clear (write). Bits 18–12 (of slot 1) are configured to specify the index to the Codec register.
  - Slot 2 is configured with the write data (in case of a write).
- When software accesses the secondary Codec, the AC '97 controller configures the outgoing frame as follows:
  - In slot 0, the valid bits for slots 1 and 2 are clear. Bits 1 and 0 (of slot 0) are set to 0b01 to select the secondary Codec.
  - In slot 1, bit 19 is set (read) or clear (write). Bits 18–12 (of slot 1) are configured to specify the index to the Codec register.
  - Slot 2 is configured with the write data (in case of a write).

Table 13-3. Slot 1: Command Address Port Bit Definitions

Bit	Name	Description
Bit[19]	RW	1 = read, 0 = write
Bit[18:12]	IDX	Code register index
Bit[11:0]	Reserved	Fill with 0s

Only one I/O cycle can be pending across the AC-link at any time. The AC '97 controller uses write and read posting on I/O accesses across the link. For instance, a read of a Codec register returns immediately, before the access crosses the link. To get the real data, software must monitor the CAR[CAIP] bit. Software must verify that the bit is not set before an access attempt to ensure it is the first access. A set CAR[CAIP] bit indicates that a Codec access is pending. After the CAR[CAIP] bit is cleared, the next Codec access (read or write) can go through.

The exception to posted accesses is reads to the Codec GPIO Pin Status register (at Codec address 0x54). Codec GPIO Pin Status reads are returned immediately with the data from the last slot 12 received. A Codec with a GPIO Pin Status register must constantly send the status of the register in slot 12.

For reads from the Codec, the controller gives the Codec a maximum of four frames to respond, after which if no response is received, it returns a dummy read completion to the CPU (0xFFFF_FFFF) and also sets the read completion status bit, GSR[RDCS].

13.4.1.3 Slot 2: Command Data Port

The command data port delivers 16-bit control register write data in the event that the current command port operation is a write cycle (as indicated by slot 1, bit 19).

Table 13-4. Slot 2: Command Data Port Bit Definitions

Bit	Name	Description
Bit[19:4]	Control register write data	Filled with 0s if current operation is a read
Bit[3:0]	Reserved	Filled with 0s
NOTE: If the current command port operation is a read, the AC '97 controller fills the entire slot-time with zeroes.		

13.4.1.4 Slot 3: PCM Playback Left Channel

Audio output frame slot 3 is the composite digital audio left-playback stream. If a sample stream is transferred with a resolution that is less than 20 bits, the AC '97 controller fills all trailing non-valid bit positions in the slot with zeroes.

13.4.1.5 Slot 4: PCM Playback Right Channel

Audio output frame slot 4 is the composite digital audio right-playback stream. If a sample stream is transferred with a resolution that is less than 20 bits, the AC '97 controller fills all trailing non-valid bit positions in the slot with zeroes.

13.4.1.6 Slot 5: Modem Line Codec

Audio output frame slot 5 contains the MSB-justified modem DAC input data if the line Codec is supported. The optional modem DAC input resolution can be implemented as 16, 18, or 20 bits. If the modem-line Codec is supported, the AC ’97 controller driver determines the DAC resolution at boot time. During normal run-time operation, the AC ’97 controller fills all trailing non-valid bit positions in the slot with zeroes. The modem Codec can be a separate Codec on the secondary line or integrated with the audio Codec.

13.4.1.7 Slots 6-11: Reserved

These audio output frame slots are reserved for future use. The AC ’97 controller fills them with zeroes.

13.4.1.8 Slot 12: I/O Control

Slot 12 has 16 MSB bits for GPIO Control (output) and Status (input). The bits minimize access latency that results from changing conditions. The value of the bits in Slot 12 are the values written to the Codec GPIO Status register at Codec address 0x54 in the modem Codec I/O space. The following rules govern slot 12 use:

1. Slot 12 is initially marked invalid by default.
2. A write to Codec address 0x54 in Codec I/O space transfers the data out of slot 12 in the next frame and slot 12 is marked valid. The data is also sent out on slots 1 and 2.
3. After the first write to Codec address 0x54, slot 12 remains valid for all subsequent frames. The data transmitted on slot 12 is the data last written to Codec address 0x54. Any subsequent write to the register sends the new data out on the next frame.
4. Following a system reset or AC ’97 cold reset, slot 12 is invalidated. Slot 12 remains invalid until the next write to the Codec address 0x54.

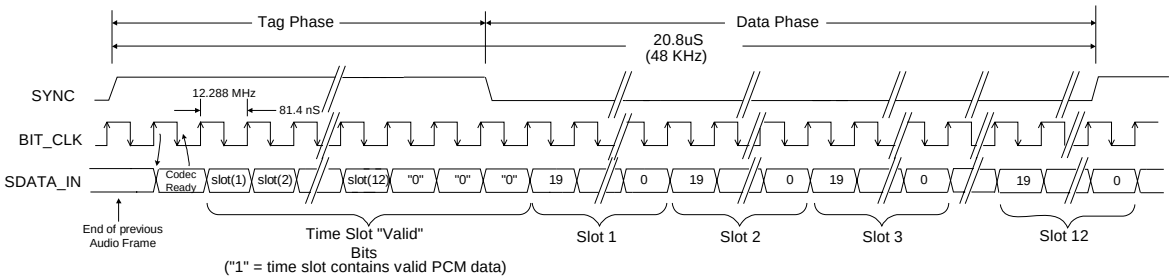
13.4.2 AC-Link Audio Input Frame (AC97_SDATA_IN)

The AC ’97 controller has two AC97_SDATA_IN lines, AC97_SDATA_IN_0 (primary) and AC97_SDATA_IN_1 (secondary). Each line can have Codecs attached. The type of Codec attached determines which slots are valid or invalid. The data slots on the two inputs are completely orthogonal (in other words, no two data slots at the same location will be valid on both lines).

Multiple input data streams are received and multiplexed on slot boundaries, as dictated by the slot valid bits in each stream. Each AC-link audio input frame consists of twelve 20-bit time slots. Slot 0 is reserved and contains 16 bits that are used for AC-link protocol infrastructure.

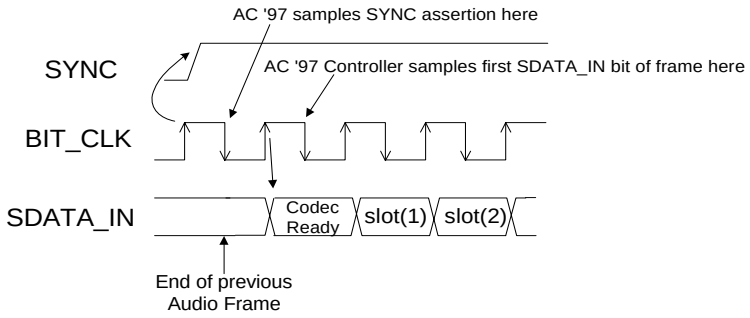
Software must poll the first bit in the audio input frame (AC97_SDATA_IN slot 0, bit 15) for an indication that the controller is in the Codec ready state before it places the AC ’97 controller into operation. When the controller is sampled Codec-ready, the next 12 bit positions sampled indicate which of the 12 time slots are assigned to input data streams and whether they contain valid data. Figure 13-5 illustrates the time-slot-based AC-link protocol.

Figure 13-5. AC '97 Input Frame



A new audio input frame begins when AC97_SYNC toggles from low to high. The low-to-high transition is synchronous to AC97_BITCLK's rising edge. On AC97_BITCLK's next falling edge, AC '97 samples AC97_SYNC's assertion. This falling edge marks the moment that AC-link's sides are each aware that a new audio frame has started. The next time AC97_BITCLK rises, the controller toggles AC97_SDATA_IN to the first bit position in slot 0 (Codec ready bit). Each new bit position is presented to AC-link on a AC97_BITCLK's rising edge and then sampled by the AC '97 controller on the following AC97_BITCLK's falling edge. This sequence ensures data transitions and subsequent sample points are time-aligned for both incoming and outgoing data streams.

Figure 13-6. Start of Audio Input Frame



The AC97_SDATA_IN composite stream is MSB-justified (MSB first), and the AC '97 Codec fills non-valid bit positions with zeroes. AC97_SDATA_IN data is sampled on AC97_BITCLK falling edges.

13.4.2.1 Slot 0: Tag Phase

In slot 0, the first bit is a global bit (AC97_SDATA_IN slot 0, bit 15) which indicates whether the AC '97 controller is in the Codec-ready state. If the Codec Ready bit is a 0b0, the AC '97 controller is not ready for normal operation. This condition is normal after the power is deasserted on reset and the AC '97 controller voltage references are settling. When the AC-link Codec Ready indicator bit is 0b1, the AC-link and AC '97 control and status registers are fully operational. The AC '97 controller must probe the Codec Powerdown Control/Status register to determine which subsections are ready.

13.4.2.2 Slot 1: Status Address Port/SLOTREQ Bits

The status port monitors the status for the AC '97 controller functions including, but not limited to, mixer settings and power management.

The stream for audio input frame slot 1 echoes the control register index for the data to be returned in slot 2, if the controller tags slots 1 and 2 as valid during slot 0.

The controller only accepts status data if the accompanying status address matches the last valid command address issued during the most recent read command.

For multiple sample rate output, the Codec examines its sample-rate control registers, the states of its FIFOs, and the incoming AC97_SDATA_OUT tag bits at the beginning of each audio output frame to determine which SLOTREQ bits to set active (low). SLOTREQ bits asserted during the current audio input frame indicate which output slots require data from the controller in the next audio output frame. For fixed 48-kHz operation, the SLOTREQ bits are set active (low), and a sample is transferred each frame.

For multiple sample-rate input, the *tag* bit for each input slot indicates whether valid data is present.

For slot 1, the audio input frame's Status Address Port delivers Codec control register read address and multiple sample-rate slot-request flags for all output slots. AC '97 defines the ten least significant bits as on-demand data-request flags for output slots 3–12. For two-channel audio, Codec-only data-request flags corresponding to slots 3 and 4 are meaningful.

Table 13-5. Input Slot 1 Bit Definitions

Bit	Description
19	Reserved (Filled with zero)
18:12	Control register Index (Filled with zeroes if AC '97 tags it invalid)
11	Slot 3 request: PCM left channel
10	Slot 4 request: PCM right channel
9	Slot 5 request: Modem line 1
8	Slot 6 request: NA
7	Slot 7 request: NA
6	Slot 8 request: NA
5	Slot 9 request: NA
4	Slot 10 request: NA
3	Slot 11 request: NA
2	Slot 12 request: NA
1:0	Reserved (Filled with zero)

Audio input frame slot 1 tag bit pertains to Status Address Port data. SLOTREQ bits are always valid independent of the slot 1 tag bit.

Note: Slot requests for slots 3 and 4 are always set or cleared in tandem (both set or both cleared).

13.4.2.3 Slot 2: Status Data Port

The Status data port delivers 16-bit control register read data.

Table 13-6. Input Slot 2 Bit Definitions

Bit	Name	Description
Bit[19:4]	Control register read data	Filled with zeroes if AC ’97 tags it invalid
Bit[3:0]	Reserved	Filled with zeroes
NOTE: If slot 2 is tagged invalid, the AC ’97 controller fills the entire slot with zeroes.		

13.4.2.4 Slot 3: PCM Record Left Channel

Audio input frame slot 3 is the AC ’97 controller Codec left-channel output.

The AC ’97 controller transmits its ADC output data (MSB first) and fills any trailing non-valid bit positions with zeroes to fill its 20-bit time slot.

13.4.2.5 Slot 4: PCM Record Right Channel

Audio input frame slot 4 is the AC ’97 controller Codec right-channel output.

The AC ’97 controller transmits its ADC output data (MSB first) and fills any trailing non-valid bit positions with zeroes to fill its 20-bit time slot.

13.4.2.6 Slot 5: Optional Modem Line Codec

Audio input frame slot 5 contains MSB justified modem ADC output data (if the line Codec is supported).

The PXA27x processor supports a 16-bit ADC output resolution for the optional modem.

13.4.2.7 Slot 6: Optional Dedicated Microphone Record Data

Audio input frame slot 6 is an optional third PCM system-input channel available for dedicated use by a microphone. This input channel supplements a true stereo output that would enable a more precise echo-cancellation algorithm for speakerphone applications.

The AC ’97 controller only supports 16-bit resolution for the MIC-in channel.

13.4.2.8 Slots 7-11: Reserved

Audio input frame slots 7–11 are reserved for future use. The AC ‘97 controller ignores them.

13.4.2.9 Slot 12: I/O Status

The GPIOs configured as inputs return their status on this slot every frame. The data returned on the latest frame is accessible to software through the Codec register at Codec address 0x54 in the modem Codec I/O space. Only the 16 MSBs return GPIO status. Bit 0 in the LSBs indicates a GPI input interrupt event. (See the AC ’97 specification, Revision 2.0, for more information.)

Reads from Codec address 0x54 are not transmitted across the link. Data received in slot 12 is stored internally in the controller, and the data from the most recent slot 12 is returned on reads from Codec address 0x54

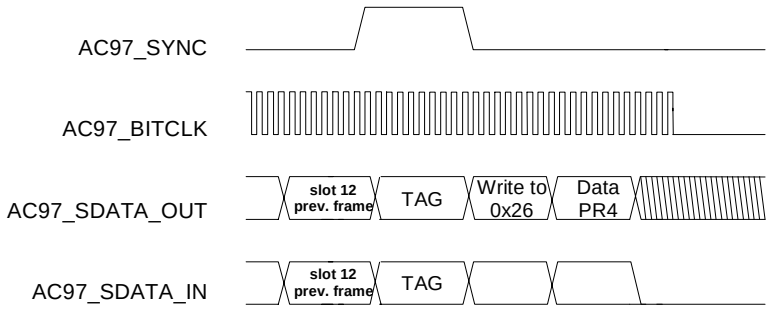
13.5 AC-Link Low-Power Modes

Software must set the GCR[ACLINK_OFF] bit before it enters the PXA27x processor's low-power modes. This ensures that the AC '97 controller does not drive the output pins on the AC-link.

13.5.1 Powering Down the AC-Link

The AC-link signals enter a low-power mode when the AC '97 Codec Powerdown register (Codec address 0x26) PR4 bit is set to a 1 (by writing 0x1000). Then the Primary Codec drives both AC97_BITCLK and AC97_SDATA_IN to a logic-low voltage level. The sequence follows the timing diagram shown in Figure 13-7.

Figure 13-7. AC-Link Power-Down Timing



Note: AC97_BITCLK is not to scale

The AC '97 controller transmits the write to Powerdown register (0x26) over the AC-link. Set up the AC '97 controller so that it does not transmit data to slots 3–12 when it writes to the Powerdown register bit PR4 (data 0x1000). The AC '97 specification, Revision 2.0, does not require the Codec to process other data when it receives a power-off request. When the Codec processes the request, it immediately toggles AC97_BITCLK and AC97_SDATA_IN to a logic-low level.

The AC '97 controller drives AC97_SYNC and AC97_SDATA_OUT to a logic-low level after setting the GCR[ACLINK_OFF] to 0b1. The AC '97 controller maintains AC97_RESET_n high when GCR[ACLINK_OFF] is set.

The following steps initiate a power-down:

1. Write 0x26 and initiate power-down.
2. Wait for GSR[CDONE] indicating that the command has completed.
3. Set GCR[ACOFF] to shut down the AC-link.

4. Set CKEN[31] in the Clock Enable register (see Section 3.8.2.2, "Clock Enable Register (CKEN)" on page 3-98).
5. Wait for GSR[ACOFFD], indicating that the AC-link shutdown is complete.
6. Clear CKEN[31].

Note: If a resume event occurs during this procedure, the Codec must not be awakened until GSR[ACOFFD] is set.

Table 13-7 describes CKEN[2] and CKEN[31] functionality:

Table 13-7. AC '97 Configuration

CKEN[31]	CKEN[2]	Description
0	0	AC97_BITCLK is disabled.
0	1	AC97_BITCLK enabled and is externally provided.
1	0	AC97_RESET_n signal is asserted, and AC97_BITCLK is the 13-MHz clock.
1	1	AC97_BITCLK is a 13-MHz clock.

Note: Software must not set or clear CKEN[31] and CKEN[2] at the same time.

13.5.2 Waking Up the AC-Link

13.5.2.1 Wake-Up Triggered by Codec

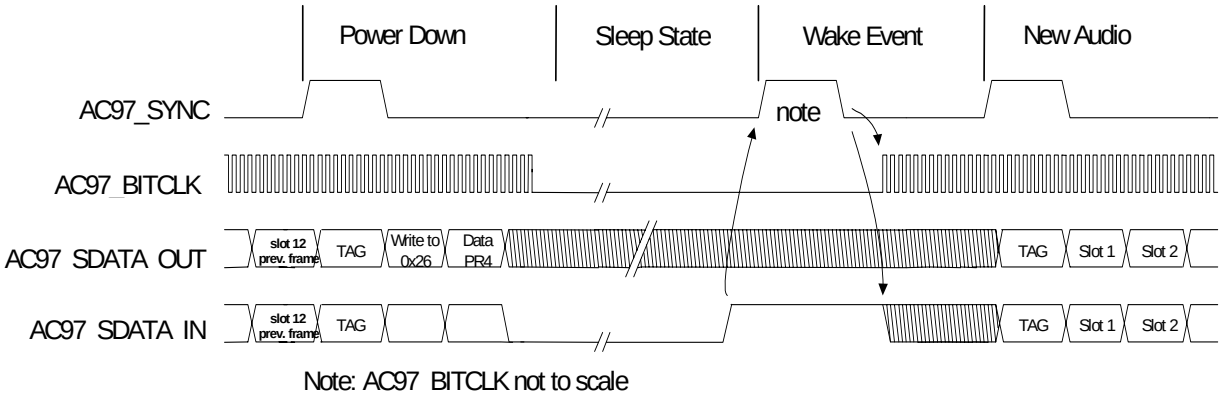
To wake up the AC-link, a Codec drives its AC97_SDATA_IN signal to a logic high level. The rising edge triggers the resume interrupt. The CPU then wakes up the Codec using the cold or warm reset sequence. If a warm reset was issued by the AC '97 controller, the Codec that signaled the wake event must keep its AC97_SDATA_IN high until it detects that a warm reset has been completed. The Codec can then toggle its AC97_SDATA_IN low. Figure 13-8 shows the AC-link timing for a wake-up triggered by a Codec.

In the case of the processor needing to be awakened from the idle low-power mode, the AC '97 controller detects the Codec wake-up event (AC97_SDATA_IN high for more than 1 µs) and signals an interrupt to the CPU.

In the case of the processor needing to be awakened from the standby or sleep low-power modes, the AC97_SDATA_IN_0 signal from the primary Codec must be routed to a valid, properly configured standby/sleep wake-up GPIO signal. For the secondary Codec to wake the processor from the standby or sleep modes, the AC97_SDATA_IN_1 GPIO must be reconfigured to be a keypad wake-up source prior to entering standby or sleep modes.

A modem Codec may require the ability to wake up the AC-link to report events such as Caller-ID and wake-up on ring.

Figure 13-8. AC97_SDATA_IN Wake-Up Signaling



13.5.2.2 Wake-Up Triggered by AC '97 Controller

AC-link protocol provides for a cold AC '97 reset and a warm AC '97 reset. The current power-down state dictates which AC '97 reset is used. Codec registers must stay in the same state during all power-down modes unless a cold AC '97 reset is performed. In a cold ds reset, the AC '97 Codec registers are initialized to their default values.

After a power-off, the AC-link must wait for a minimum of four audio frame times after the frame in which the power-off occurred before it can be reactivated by reasserting the AC97_SYNC signal. When AC-link powers up, it indicates readiness in the Codec ready bit (input slot 0, bit 15).

13.5.2.2.1 Cold AC '97 Reset

A cold reset is generated when the AC97_RESET_n signal is asserted through the GCR[nCRST]. Asserting and de-asserting AC97_RESET_n activates AC97_BITCLK (if supplied the Codec supplies it) and AC97_SDATA_OUT. All AC '97 Codec control registers are initialized to their default power-on reset values. AC97_RESET_n is an asynchronous AC '97 input to the Codec.

Note: Prior to clearing the GCR[nCRST] bit, all AC '97 interrupts must be masked (disabled). The interrupts can be re-enabled after the cold reset has occurred.

The following steps initiate a cold reset:

1. Set GCR[ACOFF] bit to shutdown AC-link
2. Wait for ACOFFD to be set in GSR[3] to ensure the link is cleanly shutdown
3. Mask the AC97 interrupt
4. Clear GCR[nCRST] to cold reset the AC97
5. Read GCR to ensure split transaction is finished
6. Set CKEN[31] to switch AC97_BITCLK to 13 MHz
7. Read CKEN, which guarantees that the CKEN configuration occurred
8. Unmask AC97 interrupt

9. Clear CKEN[31] to disable 13 MHz clock
   - — Read CKEN to guarantee that the CKEN configuration occurred
   - — Set GCR[nCRST] bit to pull AC97 out of reset
10. Read GCR back to ensure the write to GCR happened

13.5.2.2.2 Warm AC '97 Reset

A warm AC '97 reset reactivates the AC-link without altering the current AC '97 Codec register values. A warm reset is generated when AC97_BITCLK is absent and AC97_SYNC is driven high for a minimum of 1 µs.

In normal audio frames, AC97_SYNC is a synchronous AC '97 input. When AC97_BITCLK is absent, AC97_SYNC is treated as an asynchronous input to generate a warm reset to AC '97.

The AC '97 controller must not activate AC97_BITCLK until it samples AC97_SYNC low again, which prevents a new audio frame from being falsely detected.

When the AC '97 controller receives a wake-up from the Codec, it issues an interrupt (if the interrupt on resume is enabled). Software must then issue a warm or cold reset to the Codec by setting the appropriate bit in the GCR.

The following steps initiate a cold reset:

1. Write 0x26 and initiate powerdown
2. Wait for command-done bit to be set in GSR
3. Set GCR[ACOFF] to shut off AC link
4. Set CKEN[31] to switch AC97_BITCLK to 13 MHz
5. Wait for GSR[ACOFFD] to be set
6. Set GPSR3[17]. This is the GPIO set register for reset_n (GPIO 113)
7. Set GPCR0[31]. This is the GPIO clear register for sync (GPIO 31)
8. Write bits GAFR3_U[3:2] = 0b00. This is the alternate function register for GPIO 113 - switch from AC97 to GPIO
9. Write bits GAFR0_U[31:30] = 0b00. This is the alternate function reg for GPIO 31 - switch from AC97 to GPIO
10. Read GAFR0_U. This read guarantees that the GPIO configuration occurred
11. Clear bit 2 of CKEN, which causes RST_AC97 to be asserted
12. Set bit 2 of CKEN, which causes RST_AC97 to be de-asserted
13. Read CKEN, which guarantees that the CKEN configuration occurred
14. Write bits GAFR3_U[3:2] = 0b10. This reverts to AC97 alternate function for GPIO 113
15. Write bits GAFR0_U[31:30] = 0b10. This reverts to AC97 alternate function for GPIO 31
16. Read GAFR0_U. This read guarantees that the GPIO configuration occurred
17. Clear bit 31 of CKEN. This causes the AC97 clock to revert to the real bitclk
18. Perform a warm reset

Note: To prevent the codec from prematurely waking up or cold resetting, the GPIO pins must be configured as regular GPIO during the process. Detailed steps are shown below.

The above procedure assumes GPIO[113] = AC97_nRESET and GPIO[31] = AC97_SYNC. The GPIO registers will change if different GPIOs are used for these signals.

For all register modifications, use a read-modify-write approach and only modify the bits mentioned.

13.6 Operation

The AC '97 controller can be accessed through the processor or the DMA controller. The processor uses programmed I/O instructions to access the AC '97 controller and can access four register types:

1. The AC '97 controller registers: Accessible at 32-bit boundaries. They are listed in Section 13.7.
2. Codec registers: An audio or modem Codec can contain up to sixty-four 16-bit registers. A Codec uses a 16-bit address boundary for registers. The AC '97 controller supplies access to the Codec registers by mapping them to its 32-bit address domain boundary. Section 13.7.17 describes the mapping from the 32-bit to 16-bit boundary. A write or read operation that targets these registers is sent across the AC-link.
3. Modem Codec GPIO register: If the AC '97 controller is connected to a modem Codec, the Codec GPIO register can also be accessed. The Codec GPIO register uses access address 0x0054 in the Codec domain. The GPIO write operation goes across the AC-link, but a read does not. The register contents are continuously updated into a register in the controller domain when a frame is received from the Codec. When the processor tries to read the Codec GPIO register, this shadow register is read instead.
4. The AC '97 controller FIFO data: The AC '97 controller has two transmit FIFOs for audio-out and modem-out and three receive FIFOs for audio-in, modem-in, and MIC-in. The transmit FIFOs are written by writing either the PCM Data register (PCDR) or the Modem Data register (MODR). Receive FIFO entries are read through the PCDR, the MODR, or the Microphone In Data register (MCDR).

The DMA controller accesses are made through the Data registers as explained in the previous paragraph. The DMA controller accesses FIFO data in 32-bit aligned blocks of 32 bytes. DMA responds to the AC '97 controller DMA requests when not disabled.

Programmed I/O (PIO) access requirements of the FIFOs are the same as for DMA. PIO uses the same Data register and requires a 32-bit aligned data block of 32 bytes. PIO responds to AC '97 controller interrupt requests when they are enabled.

DMA requests or PIO interrupts are made for the following conditions. Do not setup a FIFO in the AC '97 controller for both DMA and PIO access.

- PCM FIFO transmit and receive DMA requests made when the PCM transmit and receive FIFOs are half full.
- Modem FIFO transmit and receive DMA requests made when the modem transmit and receive FIFOs are half full.
- MIC-in receive DMA requests made when the MIC-in receive FIFO is half full.

When the DMA signals an EOC (End of Chain) and there is data in a receive FIFO, the AC ‘97 controller signals an end-of-chain interrupt. The interrupt is cleared when software reads remaining data out of the FIFOs.

13.6.1 Initialization

The AC ’97 Codec and AC ’97 controller circuitry are reset at power-on with the AC97_RESET_n signal, which remains asserted (low) until either the audio or modem driver sets the GCR[nCRST] bit.

1. Program the GPIO Direction register (GPDR) and GPIO Alternate Function Select register (GPAR) to assign proper pin directions for the various AC ’97 controller ports. Refer to Section 13.3, for details.
2. Enable either DMA requests in the GCR or PIO interrupts in their respective control registers. The FIFO service requests do not support programmable FIFO thresholds.
3. Deassert AC97_RESET_n by setting GCR[nCRST] to 0b1. Deasserting AC97_RESET_n has the following effects:
   — Places all other registers in their active state allowing them to be programmed. When GCR[nCRST] is 0b0, all other registers are in their reset state.
   — Frames filled with 0s are transmitted because the transmit FIFO is still empty. However, this situation does not cause an error condition (because nothing is tag-valid).
   — The AC ’97 controller does not record any data until it receives a Codec Ready indication from the Codec, and the Codec tags an input frame (and slot) as valid.
4. Enable Primary Ready Interrupt Enable (GCR[PRDY_IE]) and/or Secondary Ready Interrupt Enable (GCR[SCRDY_IE]) in the GCR. Software can also poll these bits.
5. Software responds to primary/secondary ready interrupts by triggering the DMA or PIO operation. The AC ’97 controller triggers a PCM-Out FIFO service request. DMA or PIO responds by filling up the transmit FIFOs.
6. The AC ’97 controller continues to transmit 0s until the transmit FIFO is one-half full. Once one-half full, valid FIFO data is sent across the AC-link.

Note: When AC97_RESET_n is deasserted, a read of the Codec Mixer register 0x00 returns what type of hardware resides in the Codec. If the Codec is not present or if the AC ’97 is not supported, AC ’97 controller does not set the Codec-ready bit, GCR[PCRDY] for the primary Codec or GCR[SCRDY] for secondary Codec.

13.6.2 Trailing Bytes and Clean Shutdown

Trailing bytes in the transmit and receive FIFOs are handled as follows:

If the transmit buffers are not an even multiple of 32-bytes, the trailing bytes in the transmit FIFO are not transmitted. A transmit buffer must be padded with zeroes if it is smaller than a multiple of 32 bytes.

Transmit trailing bytes: Data in the transmit buffers must be a multiple of 32 bytes. If the data is not a multiple of 32 bytes, software must pad the transmit buffer with zeroes to a multiple of 32 bytes. Any data remaining in the transmit FIFO that is not a multiple of 32 bytes is not transmitted.

Receive trailing bytes: When the Codec stops transmitting valid data, as defined by valid tag bits, the AC '97 controller stops recording data for that Codec. If the data is not multiple of 32 bytes, the AC '97 controller holds the trailing bytes in the FIFO until software shuts it down by setting GCR[ACOFF] to initiate a clean shutdown. The AC '97 controller does not make a FIFO service request for these trailing bytes.

Clean Shutdown: Setting GCR[ACOFF] cleanly shuts down the AC '97 controller. The AC '97 controller deasserts any active transmit FIFO service request (DMA or PIO interrupt) and stops transmitting and receiving data. The AC '97 controller discards all remaining data in the transmit FIFO and receive FIFO and drives AC97_SYNC and AC97_SDATA_OUT to a low-logic level. When all FIFOs are empty and AC97_SYNC and AC97_SDATA_OUT are at low-logic level, the AC '97 controller sets GSR[ACOFFD], indicating a complete clean shutdown.

General shutdown issues: Software can determine if the clean shutdown has completed by reading GSR[ACOFFD]. Clearing GCR[nCRST] causes an immediate shutdown of the AC-link and reset of the AC '97 controller circuitry. The GCR[nCRST] bit supersedes the GCR[ACOFF] bit and therefore prevents a clean shutdown if set during or before the shutdown sequence.

13.6.3 Operational Flow for Accessing Codec Registers

Software accesses the Codec registers by translating a 7-bit Codec address into a 32-bit processor physical address. For details regarding the address translation, refer to Section 13.7.17.

Software must read the Codec Access register (CAR) to lock the AC-link. The AC-link is free if the CAR[CAIP] bit is clear. For details about the CAR, refer to Table 13-14.

The read access to the CAR sets the CAR[CAIP] bit. The AC '97 controller clears the CAR[CAIP] bit when the Codec-write or Codec-read operation completes. Software can also clear the CAR[CAIP] bit by writing a 0b0.

After it locks the AC-link, software can write or read a Codec register using the appropriate PXA27x processor physical address.

The AC '97 controller sets the GSR[CDONE] bit after the completion of a Codec write operation. For details, refer to Table 13-9. Software indicates the completion of the Codec write operation bit by setting the GSR[CDONE] bit.

To read a Codec, the software must complete the following steps:

1. Software issues a dummy read to the Codec register. The AC '97 controller responds to this read operation with invalid data. The AC '97 controller then initiates the read access across the AC-link.

2. When the Codec read operation completes, the AC '97 controller sets the GSR[SDONE] bit. For details, refer to Table 13-9. Software clears this bit by writing 0b1 to it.
3. Software repeats the read operation as detailed in Step 1. The AC '97 controller now returns the data sent by the Codec. The second read operation also initiates a read access across the AC-link.

Note: The the AC '97 controller times-out the read operation if the Codec fails to respond in four AC97_SYNC frames. In this case, the second read operation returns a timed-out data value of 0x0000_FFFF.

13.6.4 Clocks and Sampling Frequencies

By default, the AC '97 controller transmits and receives data at a sampling frequency of 48 kHz. It can, however, sample data at frequencies less than 48 kHz if the Codec supports on-demand slot requests. The Codec in this case executes a certain algorithm and informs the controller not to transmit valid data in certain frames. For example, if the AC '97 controller sends out 480 frames, and the Codec instructs the AC '97 controller not to send valid data in 39 of those 480 frames, the Codec would have, in effect, sampled data at 44.1 kHz. When the Codec transmits data (controller-receive mode), it can use the same algorithm to transmit valid frames with some empty ones mixed in.

All data transfers across the AC-link are synchronized to AC97_SYNC's rising edge. The AC '97 controller divides the AC97_BITCLK by 256 to generate the AC97_SYNC signal. This calculation yields a 48 kHz AC97_SYNC signal, and its period defines a frame. Data is toggled on the AC-link on every AC97_BITCLK rising edge and subsequently sampled on AC-link's receiving side on each following AC97_BITCLK falling edge. For a timing diagram, see Figure 13-3.

Note: The AC '97 controller cannot operate when the processor is in deep-idle mode when both PLLs are turned off.

13.6.5 FIFOs

The AC '97 controller has five FIFOs:

- PCM transmit FIFO, with sixteen 32-bit entries
- PCM receive FIFO, with sixteen 32-bit entries
- Modem transmit FIFO, with sixteen 32-bit entries (upper 16 bits must always be 0)
- Modem receive FIFO, with sixteen 32-bit entries (upper 16 bits are always 0)
- MIC-in receive FIFO, with sixteen 32-bit entries (upper 16 bits are always 0)

A receive FIFO triggers a DMA request when the FIFO has eight or more entries. A transmit FIFO triggers a DMA request when it holds less than eight entries. A transmit FIFO must be half-full (filled with eight entries) before any data is transmitted across the AC-link.

13.6.5.1 Transmit FIFO Errors

Channel-specific status bits are updated during transmit-underrun conditions and trigger interrupts if enabled. Refer to Table 13-12 and Table 13-12 for details on the status bits. During transmit-underrun conditions, the last valid sample is continuously sent out across the AC-link. A transmit underrun can occur under the following conditions:

- Valid transmit data is still available in memory, but the programmed I/O or the DMA controller starves the transmit FIFO because it is servicing other higher-priority peripherals.
- Programmed I/O or the DMA controller has transferred all valid data from memory to the transmit FIFO. This prompts the last valid sample to be echoed across the AC-link until AC97_RESET_n is asserted and turns off the AC '97 controller.

13.6.5.2 Receive FIFO Errors

Channel-specific status bits are updated during receive-overrun conditions and trigger interrupts when enabled. Refer to Table 13-13, Table 13-17, and Table 13-22 for details on the status bits. During receive-overrun conditions, data the Codec sends is not recorded.

13.6.6 Interrupts

The following status bits interrupt the processor when the interrupts are enabled:

- MIC-In FIFO Error—MIC-in receive FIFO overrun or underrun error
- Modem-In FIFO Error—Modem receive FIFO overrun or underrun error
- PCM-in FIFO Error—Audio receive FIFO overrun or underrun error
- Modem-Out FIFO Error—Modem transmit FIFO overrun or underrun error
- PCM-Out FIFO Error—Audio transmit FIFO overrun or underrun error
- Mic-In FIFO Service Request—Mic-in receive FIFO contains more than 16 bytes
- Modem-In FIFO Service Request—Modem receive FIFO contains more than 16 bytes
- PCM-In FIFO Service Request—Audio receive FIFO contains more than 32 bytes
- Modem-Out FIFO Service Request—Modem transmit FIFO contains less than 16 bytes
- PCM-Out FIFO Service Request—Audio transmit FIFO contains less than 32 bytes
- Modem Codec GPI Status Change Interrupt—Interrupts the CPU if bit 0 of slot 12 is set. This indicates a change in one of the bits in the modem Codec's GPIO register.
- Primary Codec Resume Interrupt—Sets a status register bit (GSR[PRESINT]) when the Primary Codec resumes from a lower power mode. Software writes a 0b1 to this bit to clear it.
- Secondary Codec Resume Interrupt—Sets a status register bit (GSR[SRESINT]) when the Secondary Codec resumes from a lower power mode. Software writes a 0b1 to this bit to clear it.
- Codec Command Done Interrupt—Interrupts the CPU when a Codec register's command is completed. Software writes a 0b1 to GSR[CDONE] to clear it.
- Codec Status Done Interrupt—Interrupts the CPU when a Codec register's status address and data reception are completed. Software writes a 0b1 to GSR[SDONE] to clear it.
- Primary Codec Ready Interrupt—Sets a status register bit (GSR[PCRDY]) when the primary Codec is ready. The Codec sets bit 0 of slot 0 on the input frame to signal that it is ready. Software clears the GCR[PRIRDY_IEN] bit to clear this interrupt.
- Secondary Codec Ready Interrupt—Sets a status register bit (GSR[SCRDY]) when the secondary Codec is ready. The Codec sets bit 0 of slot 0 on the input frame to signal that it is ready. Software clears the GCR[SECRDY_IEN] bit to clear this interrupt.

- EOC Interrupt—A status bit is set when there is data in a receive FIFO when the DMA signals an end-of-chain. The AC ‘97 controller signals an end-of-chain interrupt when this occurs. The interrupt is cleared when software reads the remaining data out of the FIFOs.

13.7 Register Descriptions

The AC ’97 controller and Codec registers are mapped in addresses 0x4050_0000–0x405F_FFFC. All AC ’97 controller registers are 32-bit-addressable. Although a Codec has up to sixty-four 16-bit registers that are 16-bit addressable, they are accessed using a 32-bit address map and are translated to a 16-bit address for the Codec.

The programmed I/O and DMA bursts can access the following registers:

- Global registers: The AC ’97 controller has three global registers: Status, Control, and Codec access registers that are common to the audio and modem domains.
- Channel-specific audio AC ’97 controller registers refer to PCM-out, PCM-in, and MIC-in channels.
- Channel-specific Modem AC ’97 controller registers refer to modem-out and modem-in channels.
- Audio Codec registers
- Modem Codec registers

Channel-specific data registers are for FIFO accesses, and the PCM, modem, and MIC-in FIFOs each have a register. A write access to one of these registers updates the written data in the corresponding transmit FIFO. A read access to one of these registers flushes out an entry from the corresponding receive FIFO.

Note: Some register bits receive status from Codecs. The Codec status sets the bit, and software clears the bit (write 0b1 to clear). The status can come in at any time, even when the bit is set or during a software clear. If software clears the bit as the Codec status updates the bit, the Codec status event takes higher priority. The term *interruptible* denotes bits that can be affected by this condition.

13.7.1 Global Control Register (GCR)

Table 13-8 describes the AC '97 controller Global Control register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-8. GCR Bit Definitions (Sheet 1 of 2)

Physical Address 0x4050_000C — **GCR** — **AC '97**

Bit	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
User Settings																																
	reserved							nDMAEN	reserved				CDONE_IE	SDONE_IE	reserved								SRDY_IE	PRDY_IE	reserved		SRES_IE	PRES_IE	ACOFF	WRST	nCRST	GPI_IE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	?	?	0	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
31:25	—	—	reserved
24	R/W	nDMAEN	DMA Enable 0 = FIFO Service Requests do cause DMA requests. 1 = FIFO Service Requests do not cause DMA requests. Software must set this bit to 0b1 when using PIO.
23:20	—	—	reserved
19	R/W	CDONE_IE	Command Done Interrupt Enable 1 = The controller triggers an interrupt to the CPU after sending the command address and data to the Codec.
18	R/W	SDONE_IE	Status Done Interrupt Enable 1 = The controller triggers an interrupt to the CPU after receiving the status address and data from the Codec.
17:10	—	—	reserved
9	R/W	SRDY_IE	Secondary Ready Interrupt Enable 1 = Enables an interrupt to occur when the secondary Codec sends the Codec READY bit on the AC97_SDATA_IN_1 pin.
8	R/W	PRDY_IE	Primary Ready Interrupt Enable 1 = Enables an interrupt to occur when the primary Codec sends the Codec READY bit on the AC97_SDATA_IN_0 pin.
7:6	—	—	reserved
5	R/W	SRES_IE	Secondary Resume Interrupt Enable 1 = Enables an interrupt to occur when the secondary Codec causes a resume event on the AC-link.
4	R/W	PRES_IE	Primary Resume Interrupt Enable 1 = Enables an interrupt to occur when the primary Codec causes a resume event on the AC-link.
3	R/W	ACOFF	AC-Link Shut Off 1 = Shuts down the AC '97 controller. The AC '97 controller deasserts any active transmit and receive FIFO service request. The AC '97 controller then discards any remaining data in the transmit and receive FIFOs and drives AC97_SYNC and AC97_SDATA_OUT low.

Table 13-8. GCR Bit Definitions (Sheet 2 of 2)

	Physical Address 0x4050_000C												GCR												AC '97											
User Settings																																				
Bit	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							nDMAEN	reserved				CDONE_IE	SDONE_IE	reserved								SRDY_IE	PRDY_IE	reserved		SRES_IE	PRES_IE	ACOFF	WRST	nCRST	GPI_IE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	?	?	0	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
2	R/W	WRST	AC '97 Warm Reset 1 = Causes a warm reset to occur on the AC-link. The warm reset awakens a suspended Codec without clearing its internal registers. If software attempts to perform a warm reset while AC97_BITCLK is running, the write is ignored, and the bit is not changed. This bit is self-clearing. It remains set until the reset completes and AC97_BITCLK is seen on the AC-link, after which it clears itself.
1	R/W	nCRST	AC '97 Cold Reset 0 = Causes a cold reset to occur throughout the AC '97 circuitry. All data in the controller and the Codec is lost. The value of this bit is retained after suspends. Therefore, if this bit was set to 0b1 prior to suspending, a cold reset is not generated automatically upon resumption. **NOTE:** Prior to asserting the cold reset (GCR[nCRST] = 0b0), all AC '97 controller interrupts must be masked (disabled). The interrupts can be re-enabled after the cold reset has occurred.
0	R/W	GPI_IE	Codec GPI Interrupt Enable: Controls whether the change in status of any modem Codec GPI causes an interrupt. 0 = The Global Status register is set, but an interrupt is not generated. 1 = The change in value of a GPI (as indicated by bit 0 of slot 12) causes an interrupt and sets bit 0 of the Global Status register.

13.7.2 Global Status Register (GSR)

Table 13-9 describes the AC '97 controller Global Status register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-9. GSR Bit Definitions (Sheet 1 of 3)

Physical Address 0x4050_001C — **GSR** — **AC '97**

	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
User Settings																																
Bit	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												CDONE	SDONE	reserved		RCS	B3S12	B2S12	B1S12	SRESINT	PRESINT	SCRDY	PCRDY	MCINT	POINT	PIINT	reserved	ACOFFD	MOINT	MIINT	GSCI
Reset	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0

Bits	Access	Name	Description
31:20	—	—	reserved
19	R/W	CDONE	Command Done This bit is 0b1 after the controller sends command address and data to the Codec. This bit is cleared by software writing 0b1 to this location. (interruptible)
18	R/W	SDONE	Status Done This bit is 0b1 after controller receives status address and data from the Codec. This bit is cleared by software writing 0b1 to this location. (interruptible)
17:16	—	—	reserved
15	R/W	RCS	Read Completion Status: Indicates the status of Codec Read completions. This bit is 0b1 if the Codec Read results in a time-out. Is 0b0 if the Codec Read completes normally. This bit is cleared by software writing 0b1 to this location.
14	R	B3S12	Bit 3 of Slot 12 Display bit 3 of the most recent valid slot 12.
13	R	B2S12	Bit 2 of Slot 12 Display bit 2 of the most recent valid slot 12.
12	R	B1S12	Bit 1 of Slot 12 Display bit 1 of the most recent valid slot 12.
11	R/W	SRESINT	Secondary Resume Interrupt This bit is 0b1 if a resume event occurred on the AC97_SDATA_IN_1 This bit is cleared by software writing 0b1 to this location. (interruptible)
10	R/W	PRESINT	Primary Resume Interrupt This bit is 0b1 if a resume event occurred on the AC97_SDATA_IN_0 This bit is cleared by software writing 0b1 to this location. (interruptible)
9	R	SCRDY	Secondary Codec Ready Reflects the state of the Codec Ready bit in AC97_SDATA_IN_1.
8	R	PCRDY	Primary Codec Ready Reflects the state of the Codec Ready bit in AC97_SDATA_IN_0.

Table 13-9. GSR Bit Definitions (Sheet 2 of 3)

Physical Address 0x4050_001C — GSR — AC '97

	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
User Settings																																
Bit	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												CDONE	SDONE	reserved		RCS	B3S12	B2S12	B1S12	SRESINT	PRESINT	SCRDY	PCRDY	MCINT	POINT	PIINT	reserved	ACOFFD	MOINT	MIINT	GSCI
Reset	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0

Bits	Access	Name	Description
7	R	MCINT	Mic-In Interrupt: Is set if one of MCSR[FIFOE], MCSR[EOC], or MCSR[FSR] is set. This bit is automatically cleared (by AC '97 controller) when MCSR[FIFOE], MCSR[EOC], and MCSR[FSR] are clear. An interrupt is triggered if one or more of the following conditions are true: • MCCR[FEIE] is 0b1 and MCSR[FIFOE] is 0b1 • MCCR[FSRIE] is 0b1 and MCSR[FSR] is 0b1 • MCSR[EOC] is 0b1
6	R	POINT	PCM-Out Interrupt Is set to 0b1 if either POSR[FIFOE] or POSR[FSR] is 0b1. This bit is automatically cleared (by AC '97 controller hardware) when POSR[FIFOE] and POSR[FSR] are 0b0. An interrupt is triggered if one or more of the following conditions are true: • POCR[FEIE] is 0b1 and POSR[FIFOE] is 0b1 • POCR[FSRIE] is 0b1 and POSR[FSR] is 0b1
5	R	PIINT	PCM-In Interrupt Is set to 0b1 if one of PISR[FIFOE], PISR[EOC], or PISR[FSR] is 0b1. This bit is automatically cleared (by AC '97 controller hardware) when PISR[FIFOE], PISR[EOC], and PISR[FSR] are 0b0. An interrupt is triggered if one or more of the following conditions are true: • PICR[FEIE] is 0b1 and PISR[FIFOE] is 0b1 • PICR[FSRIEIE] is 0b1 and PISR[FSR] is 0b1 • PISR[EOC] is 0b1
4	—	—	reserved
3	R	ACOFFD	AC-link Shut Off Done Is 0b1 if the AC-link has been cleanly shutdown. Is 0b0 if the AC '97 controller still has valid data to transfer. It is cleared when GCR[ACOFF] is cleared.This bit only has a meaning when the GCR[ACOFF] bit is 0b1.

Table 13-9. GSR Bit Definitions (Sheet 3 of 3)

Physical Address 0x4050_001C — GSR — AC '97

Bit	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
User Settings																																
	reserved												CDONE	SDONE	reserved		RCS	B3S12	B2S12	B1S12	SRESINT	PRESINT	SCRDY	PCRDY	MCINT	POINT	PIINT	reserved	ACOFFD	MOINT	MIINT	GSCI
Reset	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0	0

Bits	Access	Name	Description
2	R	MOINT	Modem-Out Interrupt Is set to 0b1 if either MOSR[FIFOE] or MOSR[FSR] is 0b1. This bit is automatically cleared (by AC '97 controller hardware) when MOSR[FIFOE] and MOSR[FSR] are 0b0. An interrupt is triggered if one or more of the following conditions are true: • MOCR[FEIE] is 0b1 and MOSR[FIFOE] is 0b1 • MOCR[FSRIE] is 0b1 and MOSR[FSR] is 0b1
1	R	MIINT	Modem-In Interrupt Is set to 0b1 if one of MISR[FIFOE], MISR[EOC], or MISR[FSR] is 0b1. This bit is automatically cleared (by AC '97 controller hardware) when MISR[FIFOE], MISR[EOC], and MISR[FSR] are 0b0. An interrupt is triggered if one or more of the following conditions are true: • MICR[FEIE] is 0b1 and MISR[FIFOE] is 0b1 • MICR[FSRIE] is 0b1 and MISR[FSR] is 0b1 • MISR[EOC] is 0b1
0	R/W	GSCI	Codec GPI Status Change Interrupt Is 0b1 whenever bit 0 of slot 12 is 0b1, which indicates that one of the GPIs changed state, and that the new values are available in slot 12. The bit is cleared by software writing 0b1 to this bit location. (interruptible)

13.7.3 PCM Out Control Register (POCR)

Table 13-10 describes the AC '97 controller Pulse Code Modulation (PCM) Out Control register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-10. POCR Bit Definitions

Physical Address 0x4050_0000																POCR												AC '97							
User Settings																																			
Bit	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																												FEIE	Rsvd	FSRIE	reserved			
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	0	?			

Bits	Access	Name	Description
31:4	—	—	reserved
3	R/W	FEIE	FIFO Error Interrupt Enable 0 = The error causes POSR[FIFOE] to be set, but the interrupt does not occur. 1 = The occurrence of the PCM transmit FIFO error causes an interrupt.
2	—	—	reserved
1	R/W	FSRIE	FIFO Service Request Interrupt Enable 0 = Disables a FIFO service request from generating an interrupt. 1 = Enables a FIFO service request to generate an interrupt.
0	—	—	reserved

13.7.4 PCM In Control Register (PCMICR)

Table 13-11 describes the AC ’97 controller Pulse Code Modulation In Control register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-11. PCMICR Bit Definitions

Physical Address 0x4050_0004 — PCMICR — AC ’97

	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	2
User Settings																																
Bit	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	2
	reserved																												FEIE	reserved	FSRIE	reserved
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	0	?

Bits	Access	Name	Description
31:4	—	—	reserved
3	R/W	FEIE	FIFO Error Interrupt Enable 0 = The error causes PCMISR[FIFOE] to be set, but the interrupt does not occur. 1 = The occurrence of a PCM receive FIFO error causes an interrupt.
2	—	—	reserved
1	R/W	FSRIE	FIFO Service Request Interrupt Enable 0 = Disables a FIFO service request from generating an interrupt. 1 = Enables a FIFO service request to generate an interrupt.
0	—	—	reserved

13.7.5 PCM Out Status Register (POSR)

Table 13-12 describes the AC '97 controller Pulse Code Modulation Out Status register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-12. POSR Bit Definitions

Physical Address 0x4050_0010																POSR								AC '97							
User Settings																															
Bit	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																											FIFOE	reserved	FSR	reserved	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	0	?	?

Bits	Access	Name	Description
31:5	—	—	reserved
4	R/W	FIFOE	FIFO Error Set when a FIFO error occurs. Cleared by writing 0b1 to this bit. Set if: 1) Transmit FIFO underrun occurs. Last valid sample is repetitively sent out. Pointers do not increment. This could happen due to: • No more valid buffer data available for transmits. • The assigned DMA channel does not have sufficient priority to handle bandwidth and latency for AC '97. 2) Transmit FIFO overrun occurs. Data in the transmit FIFO is preserved. Pointers do not increment. This could happen only if programmed I/O tries to update the transmit FIFO when it is already full. This is a critical error. Software must re-align the data buffer to be a multiple of the request size, clear the error, and then wait for a FIFO service request before attempting to write the FIFO again.
3	—	—	reserved
2	R	FSR	FIFO Service Request 0 = FIFO does not need servicing. 0 = FIFO needs servicing. This bit is updated independently of the value its interrupt enable, FSRIE.
1:0	—	—	reserved

13.7.6 PCM In Status Register (PCMISR)

Table 13-13 describes the AC '97 controller Pulse Code Modulation In Status register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-13. PCMISR Bit Definitions

Physical Address 0x4050_0014 — PCMISR — AC '97

	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
User Settings																																
Bit	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																											FIFOE	EOC	FSR	reserved	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?

Bits	Access	Name	Description
31:5	—	—	reserved
4	R/W	FIFOE	FIFO Error Set when a FIFO error occurs. Cleared by writing 0b1 to this bit. Set if: • Receive FIFO overrun occurs. The FIFO pointers do not increment. The incoming data from the AC-link is not written into the FIFO and is lost. This could happen if the assigned DMA channel does not have sufficient priority to handle bandwidth and latency for AC '97. • Receive FIFO underrun occurs. Invalid data is read by the CPU. Pointers do not increment. This could happen only if programmed I/O tries to read the receive FIFO when it is empty.
3	R/W	EOC	DMA End of Chain Interrupt Set to 0b1 by AC '97 controller hardware when DMA signals an end of descriptor chain (EOC) while reading data from the FIFO. DMA channel stops after signaling EOC and is not able to service the FIFO further. Clear this bit by software by writing 0b1 to this bit. Software must do so only after reading out data in the FIFO. **NOTE:** This bit can only be cleared by a dedicated write of 0b1 to this bit location. Concurrent clearing of multiple bits in this register does not clear EOC.
2	R	FSR	FIFO Service Request 0 = FIFO does not need servicing. 1 = FIFO needs servicing. This bit is updated independently of the value of its interrupt enable, FSRIE.
1:0	—	—	reserved

13.7.7 Codec Access Register (CAR)

Table 13-14 describes the AC '97 controller Codec Access register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-14. CAR Bit Definitions

Physical Address 0x4050_0020																CAR												AC '97			
User Settings																															
Bit	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																															CAIP
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0

Bits	Access	Name	Description
31:1	—	—	reserved
0	R/W	CAIP	Codec Access In Progress This bit is read by software to check whether a Codec I/O cycle is currently in progress. If no cycle is in progress, this bit is 0b0, and the act of reading the register sets this bit to 0b1, which reserves the right for that software driver to perform the I/O cycle. Once the cycle is complete, the hardware automatically clears the bit. Software can also clear this bit by writing 0b0 to this bit location, if it decides not to perform a Codec I/O cycle after having read this bit. If the bit is already set when software reads it, it indicates that another driver is performing a Codec I/O cycle across the link and the currently accessing driver must try again later. (This bit applies to all Codec I/O cycles—GPIO or otherwise.) Refer to Section 13.6.3 for more detail.

13.7.8 PCM Data Register (PCDR)

Table 13-15 describes the AC '97 controller Pulse Code Modulation Data register.

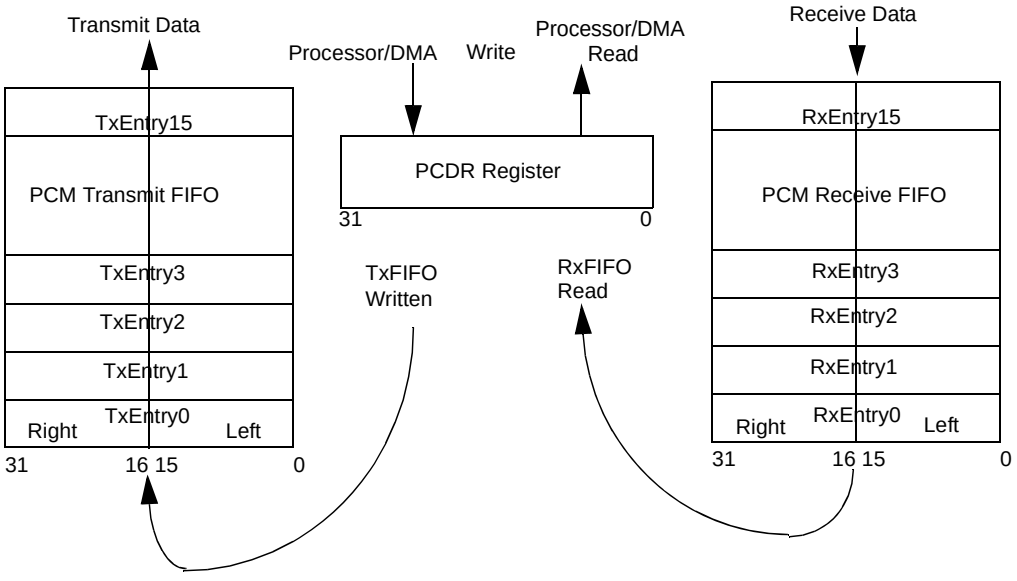
This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-15. PCDR Bit Definitions

	Physical Address 0x4050_0040															PCDR												AC '97															
User Settings																																										
Bit	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
	PCMR																PCML															
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

Bits	Access	Name	Description
31:16	R/W	PCMR	PCM Right Channel Data Writing a sample to this register updates the data into the PCM transmit FIFO. Reading this register flushes a sample from the PCM receive FIFO. Right and left samples are read and written at the same time to form 32 bits of data.
15:0	R/W	PCML	PCM Left Channel Data Writing a sample to this register updates the data into the PCM transmit FIFO. Reading this register flushes a sample from the PCM receive FIFO. Right and left data are read and written at the same time to form 32 bits of data.
NOTE: Writing a 32-bit sample to this register updates the data into the PCM transmit FIFO. Reading this register gets a 32-bit sample from the PCM receive FIFO.			

Figure 13-9. PCM Transmit and Receive Operation



13.7.9 Microphone In Control Register (MCCR)

Table 13-16 describes the AC '97 controller Microphone In Control register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-16. MCCR Bit Definitions

Physical Address 0x4050_0008 — MCCR — AC '97

	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
User Settings																																
Bit	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																												FEIE	reserved	FSRIE	reserved
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	0	?

Bits	Access	Name	Description
31:4	—	—	reserved
3	R/W	FEIE	FIFO Error Interrupt Enable 0 = MCSR[FIFOE] is set, but the interrupt does not occur. 1 = The occurrence of a MIC receive FIFO error causes an interrupt.
2	—	—	reserved
1	R/W	FSR	FIFO Service Request Interrupt Enable 0 = Disables a FIFO service request from generating an interrupt. 1 = Enables a FIFO service request to generate an interrupt.
0	—	—	reserved

13.7.10 Microphone In Status Register (MCSR)

Table 13-17 describes the AC ’97 controller Microphone In Status register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-17. MCSR Bit Definitions

Physical Address 0x4050_0018																MCSR												AC ’97					
User Settings																																	
Bit	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																											FIFOE	EOC	FSR	reserved	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?

Bits	Access	Name	Description
31:5	—	—	reserved
4	R/W	FIFOE	FIFO Error Set when a FIFO error occurs. Cleared by writing 0b1 to this bit position. Set if: • Receive FIFO overrun occurs. The FIFO pointers do not increment, the incoming data from the AC-link is not written into the FIFO and is lost. This could happen if the assigned DMA channel does not have sufficient priority to handle bandwidth and latency for AC ’97. • Receive FIFO underrun occurs. Invalid data is read by the CPU. Pointers do not increment. This could happen only if programmed I/O tries to read the receive FIFO when it is empty.
3	R/W	EOC	DMA End of Chain Interrupt Set to 0b1 by AC ’97 controller hardware when DMA signals an end of descriptor chain (EOC) while reading data from the FIFO. DMA channel stops after signaling EOC and is not able to service the FIFO further. Clear this bit by software by writing 0b1 to this bit. Software must do so only after reading out data in the FIFO. **NOTE:** This bit can only be cleared by a dedicated write of 0b1 to this bit location. Concurrent clearing of multiple bits in this register does not clear EOC.
2	R	FSR	FIFO Service Request 0 = FIFO does not need servicing. 1 = FIFO needs servicing. This bit is updated independently of the value of its interrupt enable, FSRIE.
1:0	—	—	reserved

13.7.11 Microphone In Data Register (MCDR)

Table 13-18 describes the AC '97 controller Microphone In Data register.

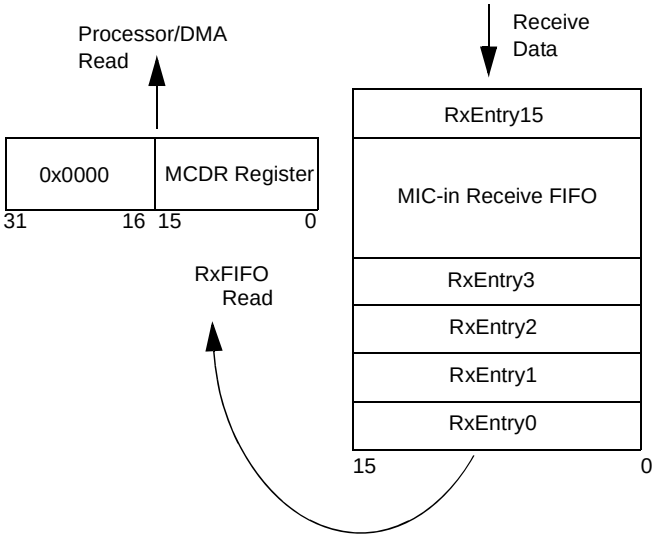
This is a read-only register. Ignore reads from reserved bits.

Table 13-18. MCDR Bit Definitions

Physical Address 0x4050_0060																MCDR																AC '97
User Settings																																
Bit	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																MCDAT															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R	MCDAT	Mic-In Data This is a read-only register. A write to this register has no effect. Reading this register flushes a 32-bit sample from the MIC-In receive FIFO.

Figure 13-10. Microphone-In Receive-Only Operation



13.7.12 Modem Out Control Register (MOCR)

Table 13-19 describes the AC ’97 controller Modem Out Control register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-19. MOCR Bit Definitions

Physical Address 0x4050_0100 — MOCR — AC ’97

User Settings																															
Bit	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																												FEIE	reserved	FSRIE	reserved
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	0	?

Bits	Access	Name	Description
31:4	—	—	reserved
3	R/W	FEIE	FIFO Error Interrupt Enable 0 = The error causes MOSR[FIFOE] to be set, but the interrupt does not occur. 1 = The occurrence of a modem transmit FIFO error causes an interrupt.
2	—	—	reserved
1	R/W	FSRIE	FIFO Service Request Interrupt Enable 0 = Disables a FIFO service request from generating an interrupt. 1 = Enables a FIFO service request to generate an interrupt.
0	—	—	reserved

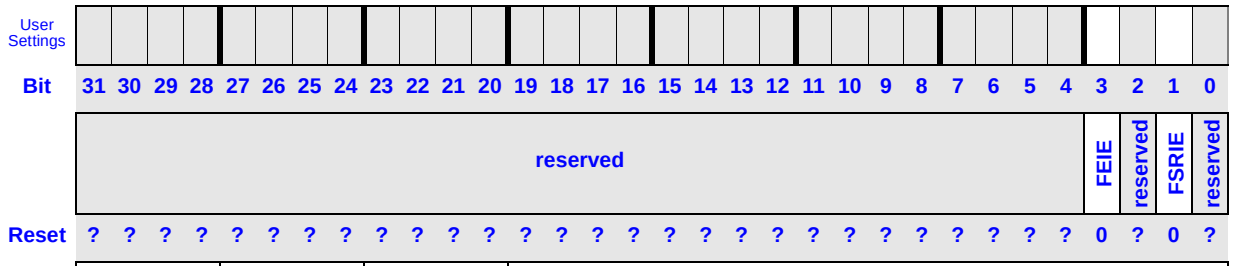
13.7.13 Modem In Control Register (MICR)

Table 13-20 describes the AC ’97 controller Modem In Control register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-20. MICR Bit Definitions

Physical Address 0x4050_0108 — **MICR** — **AC ’97**



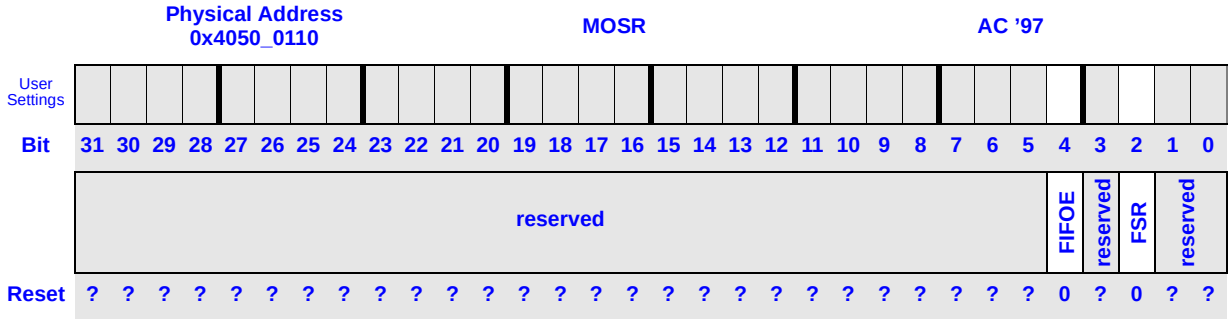
Bits	Access	Name	Description
31:4	—	—	reserved
3	R/W	FEIE	FIFO Error Interrupt Enable 0 = The error causes MISR[FIFOE] to be set, but the interrupt does not occur. 1 = The occurrence of a modem receive FIFO error causes an interrupt.
2	—	—	reserved
1	R/W	FSRIE	FIFO Service Request Interrupt Enable 0 = Disables a FIFO service request from generating an interrupt. 1 = Enables a FIFO service request to generate an interrupt.
0	—	—	reserved

13.7.14 Modem Out Status Register (MOSR)

Table 13-21 describes the AC ’97 controller Modem Out Status register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-21. MOSR Bit Definitions



Bits	Access	Name	Description
31:5	—	—	reserved
4	R/W	FIFOE	FIFO Error Set when a FIFO error occurs. Cleared by writing 0b1 to this bit position. Set if: 1) Transmit FIFO underrun occurs. Last valid sample is repetitively sent out. Pointers do not increment. This could happen due to: • No more valid buffer data available for transmits. • The assigned DMA channel does not have sufficient priority to handle bandwidth and latency for AC ’97. 2) Transmit FIFO overrun occurs. Data in the transmit FIFO is preserved. Pointers do not increment. This could happen only if programmed I/O tries to update the transmit FIFO when it is already full. This is a critical error. Software must re-align the data buffer to be a multiple of the request size, clear the error, and then wait for a FIFO service request before attempting to write the FIFO again.
3	—	—	reserved
2	R	FSR	FIFO Service Request 0 = FIFO does not need servicing. 1 = FIFO needs servicing. This bit is updated independently of the value of its interrupt enable, FSRIE.
1:0	—	—	reserved

13.7.15 Modem In Status Register (MISR)

Table 13-22 describes the AC ’97 controller Modem In Status register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-22. MISR Bit Definitions

Physical Address 0x4050_0118 — MISR — AC ’97

	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
User Settings																																
Bit	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																											FIFOE	EOC	FSR	reserved	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?

Bits	Access	Name	Description
31:5	—	—	reserved
4	R/W	FIFOE	FIFO Error Bit set when a FIFO error occurs. Bit is cleared by writing 0b1 to this bit. Set if: • Receive FIFO overrun occurs. The FIFO pointers do not increment. The incoming data from the AC-link is not written into the FIFO and is lost. This could happen if the assigned DMA channel does not have sufficient priority to handle bandwidth and latency for AC ’97. • Receive FIFO underrun occurs. Invalid data is read by the CPU. Pointers do not increment. This could happen only if programmed I/O tries to read the receive FIFO when it is empty.
3	R/W	EOC	DMA End-of-Chain Interrupt Set to 0b1 by AC ’97 controller hardware when DMA signals an end of descriptor chain (EOC) while reading data from the FIFO. DMA channel stops after signaling EOC and is not able to service the FIFO further. Clear this bit by software by writing 0b1 to this bit. Software must do so only after reading out data in the FIFO. **NOTE:** This bit can only be cleared by a dedicated write of 0b1 to this bit location. Concurrent clearing of multiple bits in this register does not clear EOC.
2	R	FSR	FIFO Service Request 0 = FIFO does not need servicing. 1 = FIFO needs servicing. This bit is updated independently of the value of its interrupt enable, FSRIE.
1:0	—	—	reserved

13.7.16 Modem Data Register (MODR)

Table 13-23 describes the AC '97 controller Modem Data register.

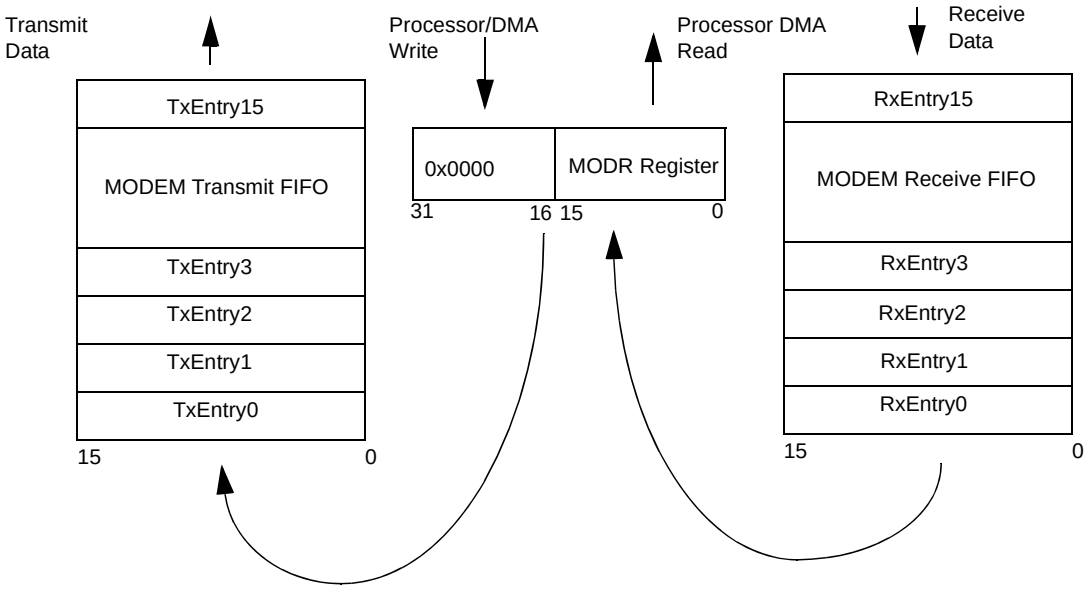
This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 13-23. MODR Bit Definitions

Physical Address 0x4050_0140																MODR																AC '97
User Settings																																
Bit	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																MODAT															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R/W	MODAT	Modem Data Writing a 32-bit sample to this register updates the data into the modem transmit FIFO. Reading this register flushes a 32-bit sample from the modem receive FIFO.

Figure 13-11. MODEM Transmit and Receive Operation



13.7.17 Accessing Codec Registers

Each Codec has up to 64 16-bit registers that are addressable internal to the Codec at half-word boundaries (16-bit boundaries). Because the PXA27x processor only supports internal register accesses at word boundaries (32-bit boundaries), software must select one of the following formulas to translate a 7-bit Codec address into a 32-bit processor address:

- Physical address for a Primary Audio Codec
  = 0x4050_0200 + Shift_Left_Once (Internal 7-bit Codec register address)
- Physical address for a Secondary Audio Codec
  = 0x4050_0300 + Shift_Left_Once (Internal 7-bit Codec register address)
- Physical address for a Primary Modem Codec
  = 0x4050_0400 + Shift_Left_Once (Internal 7-bit Codec register address)
- Physical address for a Secondary Modem Codec
  = 0x4050_0500 + Shift_Left_Once (Internal 7-bit Codec register address)

In the equations, Shift_Left_Once() shifts the 7-bit Codec address left by one bit and shifts a 0 to the LSB. The address translations are shown in Table 13-24.

Data written to a CODEC register is sampled at the beginning of every frame. If software writes data to the CODEC register more than once per frame, the AC '97 controller transmits the last write only.

Table 13-24. Address Mapping for CODEC Registers

7-Bit Codec Address	Physical Address for Primary Audio Codec	Physical Address for Secondary Audio Codec	Physical Address for Primary Modem Codec	Physical Address for Secondary Modem Codec
0x00	0x4050_0200	0x4050_0300	0x4050_0400	0x4050_0500
0x02	0x4050_0204	0x4050_0304	0x4050_0404	0x4050_0504
0x04	0x4050_0208	0x4050_0308	0x4050_0408	0x4050_0508
0x06	0x4050_020C	0x4050_030C	0x4050_040C	0x4050_050C
0x08	0x4050_0210	0x4050_0310	0x4050_0410	0x4050_0510
0x0A	0x4050_0214	0x4050_0314	0x4050_0414	0x4050_0514
0x0C	0x4050_0218	0x4050_0318	0x4050_0418	0x4050_0518
0x0E	0x4050_021C	0x4050_031C	0x4050_041C	0x4050_051C
0x10	0x4050_0220	0x4050_0320	0x4050_0420	0x4050_0520
0x12	0x4050_0224	0x4050_0324	0x4050_0424	0x4050_0524
0x14	0x4050_0228	0x4050_0328	0x4050_0428	0x4050_0528
0x16	0x4050_022C	0x4050_032C	0x4050_042C	0x4050_052C
0x18	0x4050_0230	0x4050_0330	0x4050_0430	0x4050_0530
0x1A	0x4050_0234	0x4050_0334	0x4050_0434	0x4050_0534
0x1C	0x4050_0238	0x4050_0338	0x4050_0438	0x4050_0538
0x1E	0x4050_023C	0x4050_033C	0x4050_043C	0x4050_053C
0x20	0x4050_0240	0x4050_0340	0x4050_0440	0x4050_0540
0x22	0x4050_0244	0x4050_0344	0x4050_0444	0x4050_0544

Table 13-24. Address Mapping for CODEC Registers

7-Bit Codec Address	Physical Address for Primary Audio Codec	Physical Address for Secondary Audio Codec	Physical Address for Primary Modem Codec	Physical Address for Secondary Modem Codec
0x24	0x4050_0248	0x4050_0348	0x4050_0448	0x4050_0548
0x26	0x4050_024C	0x4050_034C	0x4050_044C	0x4050_054C
0x28	0x4050_0250	0x4050_0350	0x4050_0450	0x4050_0550
0x2A	0x4050_0254	0x4050_0354	0x4050_0454	0x4050_0554
0x2C	0x4050_0258	0x4050_0358	0x4050_0458	0x4050_0558
0x2E	0x4050_025C	0x4050_035C	0x4050_045C	0x4050_055C
0x30	0x4050_0260	0x4050_0360	0x4050_0460	0x4050_0560
0x32	0x4050_0264	0x4050_0364	0x4050_0464	0x4050_0564
0x34	0x4050_0268	0x4050_0368	0x4050_0468	0x4050_0568
0x36	0x4050_026C	0x4050_036C	0x4050_046C	0x4050_056C
0x38	0x4050_0270	0x4050_0370	0x4050_0470	0x4050_0570
0x3A	0x4050_0274	0x4050_0374	0x4050_0474	0x4050_0574
0x3C	0x4050_0278	0x4050_0378	0x4050_0478	0x4050_0578
0x3E	0x4050_027C	0x4050_037C	0x4050_047C	0x4050_057C
0x40	0x4050_0280	0x4050_0380	0x4050_0480	0x4050_0580
0x42	0x4050_0284	0x4050_0384	0x4050_0484	0x4050_0584
0x44	0x4050_0288	0x4050_0388	0x4050_0488	0x4050_0588
0x46	0x4050_028C	0x4050_038C	0x4050_048C	0x4050_058C
0x48	0x4050_0290	0x4050_0390	0x4050_0490	0x4050_0590
0x4A	0x4050_0294	0x4050_0394	0x4050_0494	0x4050_0594
0x4C	0x4050_0298	0x4050_0398	0x4050_0498	0x4050_0598
0x4E	0x4050_029C	0x4050_039C	0x4050_049C	0x4050_059C
0x50	0x4050_02A0	0x4050_03A0	0x4050_04A0	0x4050_05A0
0x52	0x4050_02A4	0x4050_03A4	0x4050_04A4	0x4050_05A4
0x54	0x4050_02A8	0x4050_03A8	0x4050_04A8	0x4050_05A8
0x56	0x4050_02AC	0x4050_03AC	0x4050_04AC	0x4050_05AC
0x58	0x4050_02B0	0x4050_03B0	0x4050_04B0	0x4050_05B0
0x5A	0x4050_02B4	0x4050_03B4	0x4050_04B4	0x4050_05B4
0x5C	0x4050_02B8	0x4050_03B8	0x4050_04B8	0x4050_05B8
0x5E	0x4050_02BC	0x4050_03BC	0x4050_04BC	0x4050_05BC
0x60	0x4050_02C0	0x4050_03C0	0x4050_04C0	0x4050_05C0
0x62	0x4050_02C4	0x4050_03C4	0x4050_04C4	0x4050_05C4
0x64	0x4050_02C8	0x4050_03C8	0x4050_04C8	0x4050_05C8
0x66	0x4050_02CC	0x4050_03CC	0x4050_04CC	0x4050_05CC
0x68	0x4050_02D0	0x4050_03D0	0x4050_04D0	0x4050_05D0
0x6A	0x4050_02D4	0x4050_03D4	0x4050_04D4	0x4050_05D4

Table 13-24. Address Mapping for CODEC Registers

7-Bit Codec Address	Physical Address for Primary Audio Codec	Physical Address for Secondary Audio Codec	Physical Address for Primary Modem Codec	Physical Address for Secondary Modem Codec
0x6C	0x4050_02D8	0x4050_03D8	0x4050_04D8	0x4050_05D8
0x6E	0x4050_02DC	0x4050_03DC	0x4050_04DC	0x4050_05DC
0x70	0x4050_02E0	0x4050_03E0	0x4050_04E0	0x4050_05E0
0x72	0x4050_02E4	0x4050_03E4	0x4050_04E4	0x4050_05E4
0x74	0x4050_02E8	0x4050_03E8	0x4050_04E8	0x4050_05E8
0x76	0x4050_02EC	0x4050_03EC	0x4050_04EC	0x4050_05EC
0x78	0x4050_02F0	0x4050_03F0	0x4050_04F0	0x4050_05F0
0x7A	0x4050_02F4	0x4050_03F4	0x4050_04F4	0x4050_05F4
0x7C	0x4050_02F8	0x4050_03F8	0x4050_04F8	0x4050_05F8
0x7E	0x4050_02FC	0x4050_03FC	0x4050_04FC	0x4050_05FC

13.8 Register Summary

All AC '97 controller registers are word-addressable (32 bits wide) and increment in units of 0x00004. The registers in the Codec are half-word-addressable (16 bits wide) and increment in units of 0x00002. These register sets are mapped in the address range of 0x4050_000 through 0x405F_FFFC.

Table 13-25. AC '97 Controller Register Summary

Address	Name	Description	Page
0x4050_0000	POCR	PCM Out Control register	13-27
0x4050_0004	PCMICR	PCM In Control register	13-28
0x4050_0008	MCCR	Microphone In Control register	13-33
0x4050_000C	GCR	Global Control register	13-22
0x4050_0010	POSR	PCM Out Status register	13-29
0x4050_0014	PCMISR	PCM In Status register	13-30
0x4050_0018	MCSR	Microphone In Status register	13-34
0x4050_001C	GSR	Global Status register	13-24
0x4050_0020	CAR	Codec Access register	13-31
0x4050_0024–0x4050_003C	—	reserved	
0x4050_0040	PCDR	PCM Data register	13-32
0x4050_0044–0x4050_005C	—	reserved	
0x4050_0060	MCDR	Microphone In Data register	13-35
0x4050_0064–0x4050_00FC	—	reserved	
0x4050_0100	MOCR	Modem Out Control register	13-36
0x4050_0104	—	reserved	
0x4050_0108	MICR	Modem In Control register	13-37

Table 13-25. AC '97 Controller Register Summary

Address	Name	Description	Page
0x4050_010C	—	reserved	
0x4050_0110	MOSR	Modem Out Status register	13-38
0x4050_0114	—	reserved	
0x4050_0118	MISR	Modem In Status register	13-39
0x4050_011C–0x4050_013C	—	reserved	
0x4050_0140	MODR	Modem Data register	13-40
0x4050_0144–0x4050_01FC	—	reserved	
(0x4050_0200–0x4050_02FC) with all in increments of 0x00004	—	Primary Audio Codec registers	13-41
(0x4050_0300–0x4050_03FC) with all in increments of 0x00004	—	Secondary Audio Codec registers	13-41
(0x4050_0400–0x4050_04FC) with all in increments of 0x0000_0004	—	Primary Modem Codec registers	13-41
(0x4050_0500–0x4050_05FC) with all in increments of 0x00004	—	Secondary Modem Codec registers	13-41
0x4050_0600–0x405F_FFFC	—	reserved	13-41

Inter-IC Sound (I^2S) Controller 14

I^2S, or IIS (for inter-IC sound), is the name of a protocol defined by Philips Semiconductor for transferring two-channel digital audio signals from one IC device to another. I^2S is a protocol for digital stereo audio. The I^2S controller included in the PXA27x processor controls the I^2S link (I2SLINK), which is a low-power four-pin serial interface for stereo audio.

14.1 Overview

The I^2S controller consists of buffers, status registers, control registers, serializers, and counters for transferring digitized audio between the PXA27x processor system memory and an external I^2S Codec.

The I^2S controller can record digitized audio by storing the samples in system memory. For playback of digitized audio or production of synthesized audio, the I^2S controller retrieves digitized audio samples from system memory and sends them to a Codec through the I2SLINK. The external digital-to-analog converter in the Codec then converts the audio samples into an analog audio waveform.

For recording digitized audio, the I^2S controller receives digitized audio samples from a Codec (through the I2SLINK) and stores them in system memory.

The I^2S controller supports the normal I^2S and the MSB-justified I^2S formats (see Section 14.4.9 for details regarding normal I^2S and the MSB-justified I^2S formats). Four, or optionally five, pins connect the controller to an external Codec:

- A bit-rate clock, which can use either an internal or an external source
- A formatting or left/right control signal
- Two serial audio pins, one input and one output
- If the bit-rate clock is supplied by the I^2S controller, an optional system clock is also sent to the Codec by the I^2S controller (see Section 14.4.8 for details).

The I^2S data can be stored to and retrieved from system memory either by the DMA controller or by programmed I/O.

For I^2S systems that support the L3 control bus protocol, additional pins are required to control the external Codec. Codecs that use an L3 control bus require 3 signals: L3_CLK, L3_DATA, and L3_MODE for writing bytes into the L3 bus register. The I^2S controller supports the L3 bus protocol via software control of the general-purpose I/O (GPIO) pins. The I^2S controller does not provide hardware control for the L3 bus protocol.

Two similar protocols exist for transmitting digitized stereo audio over a serial path: normal I^2S and MSB-justified-I^2S. Both work with a variety of clock rates, which can be obtained either by dividing the PLL clock by a programmable divider or from an external clock source. For further details on clock rates, see Table 14-2.

Note: The AC ’97 controller and the I^2S controller cannot be used at the same time.

14.2 Features

The I^2S controller has the following key features:

- Record and playback of 64-bit stereo audio samples
- Each sample has two channels: audio-left and audio-right.
- The audio-left and audio-right channels are each 32 bits wide.
- Each channel has 16 MSB bits of valid data and 16 LSB bits of padded zeros.
- Supports MSB-justified and normal I^2S modes
- Supports sampling frequencies of 48 kHz, 44.1 kHz, 22.05 kHz, 16 kHz, 11.025 kHz, and 8 kHz. The sampling rate variations are restricted to 0.5%.
- The bit-rate clock (I2S_BITCLK) can be configured to be either an input or an output. If configured as output, the processor supplies an I^2S system clock (I2S_SYSCLK) that is four times the I2S_BITCLK.

14.3 Signal Descriptions

All clocks in the I^2S controller are based on the I2S_SYSCLK signal. I2S_SYSCLK generates a frequency between approximately 2 MHz and 12.2 MHz by dividing down the PLL clock with a programmable divisor. This frequency is always 256 times the audio sampling frequency. I2S_SYSCLK is driven out of the PXA27x processor only if I2S_BITCLK is configured as an output.

I2S_BITCLK supplies the serial audio bit rate, which is the basis for the external Codec bit-sampling logic. I2S_BITCLK is one-quarter the frequency of I2S_SYSCLK and is 64 times the audio sampling frequency. One bit of the serial audio data sample is transmitted or received each I2S_BITCLK period. A single serial audio sample comprises a “left” and “right” signal, each containing either 8, 16, or 32 bits.

I2S_SYNC is I2S_BITCLK divided by 64, resulting in an 8–48 kHz signal. The state of I2S_SYNC denotes whether the current serial data samples are left- or right-channel data.

The I2S_SDATA_IN and I2S_SDATA_OUT data pins send/receive the serial audio data to/from the Codec.

Table 14-1 lists the signals between the I^2S and an external Codec device.

Table 14-1. I^2S Controller I/O Signal Descriptions

Name	Type	Description
I2S_SYSCLK	Output	System clock = I2S_BITCLK * 4 used by the Codec only.
I2S_BITCLK	Input or Output	Bit-rate clock = I2S_SYNC * 64
I2S_SYNC	Output	Left/right identifier
I2S_SDATA_OUT	Output	Serial audio output data to Codec
I2S_SDATA_IN	Input	Serial audio input data from Codec

I2S_BITCLK can be configured either as an input or as an output. To program the direction, do the following:

1. Program the GPIO Direction register (GPDR). See Section 24.5.1, “GPIO Pin-Direction Registers (GPDR)” on page 24-11 for details regarding the GPDR.
2. Program the GPIO Alternate Function Select register (GAFR). See Section 24.5.4, “GPIO Alternate Function Register (GAFR)” on page 24-23 for details regarding the GAFR.
3. Program the BCKD bit in the I^2S controller’s Serial Audio Control register. See Section 14.5.1 for details.

Note: Modifying the status of the SACR0[BCKD] bit during normal operation can cause jitter on the I2S_BITCLK and can affect serial activity. To avoid such problems, the clock must be disabled before changing the bit-clock direction and then enabled after changing the bit-clock direction

If I2S_BITCLK is an output, I2S_SYSCLK must be configured as an output. If I2S_BITCLK is supplied by the Codec, the corresponding GPIO pin for I2S_SYSCLK can be used for an alternate function. To configure I2S_SYSCLK as an output, follow these steps:

1. Program the GPIO Direction register (GPDR). See Section 24.5.1, “GPIO Pin-Direction Registers (GPDR)” on page 24-11 for details regarding the GPDR.
2. Program the GPIO Alternate Function Select register (GAFR). See Section 24.5.4, “GPIO Alternate Function Register (GAFR)” on page 24-23 for details regarding the GAFR.

To configure I2S_SYNC and I2S_SDATA_OUT as outputs, follow these steps:

1. Program the GPIO Direction register (GPDR). See Section 24.5.1, “GPIO Pin-Direction Registers (GPDR)” on page 24-11 for details regarding the GPDR.
2. Program the GPIO Alternate Function Select register (GAFR). See Section 24.5.4, “GPIO Alternate Function Register (GAFR)” on page 24-23 for details regarding the GAFR.

To configure I2S_SDATA_IN as an input, follow these steps:

1. Program the GPIO Direction register (GPDR). See Section 24.5.1, “GPIO Pin-Direction Registers (GPDR)” on page 24-11 for details regarding the GPDR.
2. Program the GPIO Alternate Function Select register (GAFR). See Section 24.5.4, “GPIO Alternate Function Register (GAFR)” on page 24-23 for details regarding the GAFR.

14.4 Operation

The I^2S controller can be accessed either by the processor using programmed I/O instructions or by the DMA controller.

The PXA27x processor uses programmed I/O instructions to access the I^2S controller and can access the following types of data:

- I^2S controller register data—All registers are 32 bits wide and are aligned to word boundaries. See Section 14.4.10 for further details.
- I^2S controller FIFO data—An entry is placed into the transmit FIFO by writing to the I^2S controller’s Serial Audio Data register (SADR). Writing to SADR updates a transmit FIFO entry. Reading SADR flushes out a receive FIFO entry.

- I^2S Codec data—The Codec registers can be accessed through the L3 bus. The L3 bus operation is emulated by software controlling three GPIO pins.

The DMA controller can only access the FIFOs. Accesses are made through the data registers, as explained in the previous paragraph. The DMA controller accesses FIFO data in blocks of 8, 16, or 32 bytes. The DMA controller responds to the following DMA requests made by the I^2S controller:

- The transmit FIFO request is based on the transmit trigger-threshold (TFTH) setting and is asserted if the transmit FIFO has less than TFTH + 1 entries. See Table 14-5 for further details regarding TFTH.
- The receive FIFO request is based on the receive trigger-threshold (RFTH) setting and is asserted if the receive FIFO has RFTH + 1 or more entries. See Table 14-5 for further details regarding RFTH.

14.4.1 Initialization

1. Set the I2S_BITCLK direction by programming the GPIO Direction register (GPDR), the GPIO Alternate Function Select register (GAFR), and bit 2 (BCKD) of the I^2S controller's Serial Audio Controller Global Control register (SACR0).
2. Choose between normal I^2S or MSB-justified modes of operation. This can be done by programming bit 0 (AMSL) of Serial Audio Controller I^2S/MSB-Justified Control register (SACR1). For further details, see Section 14.5.2.
3. Optional: Programmed I/O may be used for priming the transmit FIFO with a few samples (ranging from 1 to 16). This allows the I^2S controller to start transmission of data to the Codec immediately after the unit is enabled.
4. The following control bits can be programmed in the I^2S controller's Serial Audio Controller Global Control register (SACR0):
   a. I2SLINK is enabled by setting SACR0[ENB].
   b. Modifying I2S_BITCLK direction by altering SACR0[BCKD] programmed in Step 1 will disrupt the clock and adversely affect I2SLINK activity.
   c. Program transmit and receive trigger thresholds by programming the TFTH and RFTH bits of SACR0[11:8] and SACR0[15:12], respectively. See Section 14.5.1.2, regarding permitted trigger thresholds.

   Valid data is sent across the I2SLINK after filling the transmit FIFO with at least one sample. One sample consists of a 32-bit value with 16 bits each dedicated to a left and a right value.

   Enabling the I2SLINK also causes zeroes to be recorded by the I^2S controller until the Codec sends in valid data.

   Enabling the I2SLINK also enables transmit and receive DMA requests.

14.4.2 Disabling and Enabling Audio Replay

Audio transmission is enabled automatically when the I^2S controller is enabled, but the I^2S controller starts transmitting data to the external Codec only when it has at least one data sample in the transmit FIFO. Transmission or replay can be stopped by setting SACR1[DPRL]. For more details, see Section 14.5.2.

Setting the DRPL bit in SACR1 has the following effects:

1. All I2SLINK replay activity is disabled. The frame or data sample being replayed will have invalid data (some data bits will be over-written with zeroes). To avoid having invalid data, disable replay only after completing transfer of valid data. Doing so causes frames with zeroes to be transmitted.
2. Transmit FIFO pointers are reset to zero.
3. Transmit FIFO fill-level is reset to zero. (Receive logic is not affected.)
4. Zeroes are transmitted across the I2SLINK.
5. Transmit DMA requests are disabled.

14.4.3 Disabling and Enabling Audio Record

Audio recording is enabled automatically when the I^2S controller is enabled. Recording is stopped by setting SACR1[DREC]. For more details, see Section 14.5.2.

Setting the DREC bit in SACR1 has the following effects:

1. I2SLINK recording activity is disabled. The frame or data sample being recorded could have invalid data (some data bits will be over-written with zeroes). To avoid having invalid data, disable recording only after completing the transfer of valid data.
2. Receive FIFO pointers are reset to zero.
3. Receive FIFO fill-level is reset to zero (transmit FIFO fill-level is not affected).
4. Any read operations by the DMA/CPU are returned with zeroes.
5. Receive DMA requests are disabled.

14.4.4 Transmit FIFO Errors

A status bit, SASR0[TUR], is set during transmit underrun conditions. If enabled, this can trigger an interrupt. For further details, see Section 14.5.3, Section 14.5.6, and Section 14.5.5. During transmit underrun conditions, the last valid sample is continuously sent out across the I2SLINK. Transmit underrun can occur under the following conditions:

1. Valid transmit data is still available in memory, but the DMA controller starves the transmit FIFO, as it is busy servicing other higher-priority peripherals.
2. The DMA controller has transferred all valid data from memory to the transmit FIFO.

The second condition prompts for the last valid sample to be echoed across the I2SLINK until the I^2S controller is turned off by clearing the SACR0[ENB] bit.

14.4.5 Receive FIFO Errors

A status bit SASR0[ROR] is set during receive overrun conditions. If enabled, this can trigger an interrupt. For further details, see Section 14.5.3, Section 14.5.6, and Section 14.5.5. During receive overrun conditions, data sent by the Codec is lost and is not recorded.

14.4.6 Trailing Bytes

When the Codec has completed transmitting valid data, zeroes are recorded by the I^2S controller, and this continues until the I^2S controller is turned off by clearing SACR0[ENB].

If the total buffer size of the received data is less than a complete multiple of the receive trigger threshold, zeroes are recorded until the I^2S controller is shut down. A receive DMA request is made when the programmed trigger threshold is reached.

To shutdown the I^2S controller, clear SACR0[ENB], which causes the I^2S controller to deassert any transmit DMA requests, and assert the receive DMA request (if there is any data in the receive FIFO). If the data in the receive FIFO is less than the receive FIFO trigger-threshold (RFTH + 1), the I^2S controller fills the remaining locations with zeroes until the FIFO has RFTH + 1 number of entries, and then generates the receive DMA request. At the same time, SASR0[I2SOFF] is set, indicating a clean shutdown process. When SASR0[I2SOFF] is set, no writes are allowed to the I^2S controller; any attempts to write to I^2S controller registers are ignored.

Only reads are allowed during a clean shutdown process. If the I^2S controller is disabled in the middle of a recording frame, the I^2S controller waits until the end of the frame, and then disables the recording. Software can re-enable the unit only after the unit has completed the clean shutdown process (when SASR0[I2SOFF] has cleared). See Table 14-9 for more information.

14.4.7 Startup and Shutdown

14.4.7.1 Clean Startup

When the I^2S controller is enabled by setting SACR0[ENB], the I^2S controller waits for a valid data sample to be loaded to the transmit FIFO before it starts transmitting data and the I2S_SYNC signal to the external Codec. This happens only at startup and only if replay mode is enabled (SACR1[DRPL] clear) and recording mode is disabled (SACR1[DREC] set). If recording mode is enabled (SACR1[DREC] clear), the I2S_SYNC signal is supplied to the external Codec at startup immediately after setting SACR0[ENB].

14.4.7.2 Clean Shutdown

When the I^2S controller is disabled (by clearing SACR0[ENB]), the I^2S controller waits until all the data in the transmit FIFO is sent to the Codec and data in the receive FIFO is read by DMA, and then shuts down. Also see Section 14.4.6.

If the I^2S controller is not receiving a bit-clock from an external Codec and the I^2S controller goes into the clean shutdown process waiting for data in the transmit FIFO to be transmitted to the external Codec, a 24-bit time-out counter (clocked by peripheral clock) is enabled as soon as the I^2S controller enters the clean shutdown process. The time-out counter is reset whenever there is a change in the transmit FIFO level. At the end of the time-out, if there is no change in the transmit FIFO fill level, the I^2S controller shuts down, even though there is data remaining in the transmit FIFO.

If the I^2S controller is in the middle of a recording frame when SACR0[ENB] is cleared, the I^2S controller waits until the end of the frame and then stops the recording. During this period, SASR0[I2SOFF] is set, indicating a clean shutdown is in progress.

To re-enable the I^2S controller after disabling it, first wait until the SACR0[I2SOFF] has cleared.

14.4.8 Serial Audio Clocks and Sampling Frequencies

I2S_BITCLK is the rate at which audio data bits enter or leave the I2SLINK. I2S_SYSCLK is required by the Codec to run delta sigma ADC operations.

I2S_BITCLK can be supplied either by the Codec or by an internal processor PLL.

- If supplied internally, I2S_BITCLK and I2S_SYSCLK are configured as output pins, and both are supplied to the Codec.
- If I2S_BITCLK is supplied by the Codec, then it is configured as an input pin. In this case, the I2S_SYSCLK's GPIO pin can be used for an alternate function.

I2S_BITCLK varies by sampling frequencies (see Table 14-2). If I2S_BITCLK is chosen as an output, the Audio Clock Divider register divides the PLL clock to generate I2S_SYSCLK. I2S_SYSCLK is further divided by four to generate I2S_BITCLK. The sampling frequency is the frequency of the I2S_SYNC signal, which is generated by dividing the I2S_BITCLK by 64 (for additional details, see Section 14.5.4).

A sampling rate of 48 kHz supports MPEG2 and MPEG4. A rate of 44.1 kHz supports MP3.

Note: The I^2S controller cannot operate when the PXA27x processor is in 13-MHz idle mode when both PLLs are turned off.

Table 14-2. Supported Sampling Frequencies

Audio Clock Divider Register (31:0)	I2S_SYSCLK = PLL Frequency/ (SADIV)	I2S_BITCLK = I2S_SYSCLK / 4	I2S_SYNC or Sampling Frequency = I2S_BITCLK / 64
0x0000_000C	12.235 MHz	3.058 MHz	47.794K (closest std = 48 kHz)
0x0000_000D	11.346 MHz	2.836 MHz	44.318K (closest std = 44.1 kHz)
0x0000_001A	5.622 MHz	1.405 MHz	21.953K (closest std = 22.05 kHz)
0x0000_0024	4.105 MHz	1.026 MHz	16.036K (closest std = 16.00 kHz)
0x0000_0034	2.811 MHz	702.75 kHz	10.980K (closest std = 11.025 kHz)
0x0000_0048	2.053 MHz	513.25 kHz	8.019K (closest std = 8.00 kHz)

14.4.9 Data Formats

14.4.9.1 FIFO and Memory Format

FIFO buffers are 16 words deep and 32 bits wide. This stores 32 samples per channel in each direction.

Audio data is stored with two samples (left + right) per 32-bit word, even if samples are smaller than 16 bits. The left channel data occupies bits [15:0], while the right channel data uses bits [31:16] of the 32-bit word. Within each 16-bit field, the audio sample is left-justified, with unused bits packed as zeroes on the right-hand (LSB) side.

In memory, the mapping of stereo samples is the same as in the FIFO buffers. However, single-channel audio occupies a full 32-bit word per sample, using either the upper or lower half of the word, depending on whether it is considered a left or right sample.

Note: Software must replicate the 16-bit sample to create a 32-bit sample, because the I^2S controller is always expecting a 32-bit sample.

Note: Transmit and receive FIFOs can hold 16 data samples of 32-bit widths. Only four bits are assigned for TFL and RFL to show the transmit and receive FIFO levels, respectively. When FIFOs are completely full or empty, TFL and RFL show 0 entries in the FIFOs. To differentiate when FIFOs are full and when empty, refer to Table 14-3 and Table 14-4.

Table 14-3. Actual TFL Value Calculations

TNF	TFL[3:0] = SASR0[11:8]	Actual_TFL
0	0000	16 samples
1	0000	0 samples

For all the combinations of TNF and TFL other than the values showed in Table 14-3, the TFL values reflect the real TFL values.

Table 14-4. Actual RFL Value Calculations

RNE	RFL[3:0] = SASR0[15:12]	Actual_RFL
0	0000	0 samples
1	0000	16 samples

For all the combinations of RNE and RFL other than the values showed in Table 14-4, the RFL values reflect the real RFL values.

14.4.9.2 I^2S and MSB-Justified Serial Audio Formats

Normal I^2S and MSB-justified are similar protocols for digitized stereo audio transmitted over a serial path.

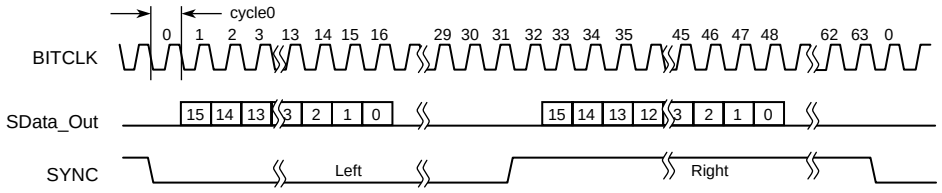
The I2S_BITCLK supplies the serial audio bit rate, the basis for the external Codec bit-sampling logic. Its frequency is 64 times the audio sampling frequency. Divided by 64, the resulting 8 kHz to 48 kHz signal signifies timing for left and right serial data samples passing on the serial data paths. This left/right signal is sent to the Codec on the I2S_SYNC pin. Each phase of the left/right signal is accompanied by one serial audio data sample on the data pins I2S_SDATA_IN and I2S_SDATA_OUT.

Figure 14-1 and Figure 14-2 provide timing diagrams that show formats for the normal I^2S and MSB-justified modes of operations. Data is sampled on the rising edge of the I2S_BITCLK and data is sent out on the falling edge of the I2S_BITCLK.

Data is transmitted and received in frames of 64 I2S_BITCLK cycles. Each frame consists of a left sample and a right sample. Each sample holds 16 bits of valid data. The LSB 16 bits of each sample is padded with zeroes.

- In the normal I^2S mode, the I2S_SYNC is low for the left sample and high for the right sample. Also, the MSB of each data sample lags behind the I2S_SYNC edges by one I2S_BITCLK cycle.
- In the MSB-justified mode, the I2S_SYNC is high for the left sample and low for the right sample. Also, the MSB of each data sample is aligned with the I2S_SYNC edges.

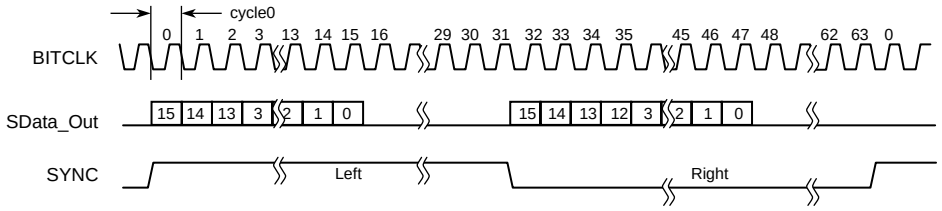
Figure 14-1. I^2S Data Formats (16 Bits)



Note: Timing for **SData_In** is identical to **SData_Out**.

A8842-01

Figure 14-2. MSB-Justified Data Formats (16 Bits)



Note: Timing for **SData_In** is identical to **SData_Out**.

A8843-01

14.4.10 Interrupts

The following SASR0 status bits, if enabled, interrupt the processor:

- Receive FIFO Service DMA Request (RFS)
- Transmit FIFO Service DMA Request (TFS)
- Transmit Underrun (TUR)
- Receive Overrun (ROR)

For further details, see Section 14.5.3

14.5 Register Descriptions

The I^2S controller registers are all 32-bit addressable, ranging from 0x4040_0000 through 0x404F_FFFC.

The I^2S controller has the following types of registers:

- Control registers program common control functions.
- The Data register is used for transmit and receive FIFO accesses.
- The Status register signals the state of the FIFO buffers and the status of the interface that is selected by the common control register.
- The Interrupt registers include the Interrupt Mask register and the Interrupt Clear register.

14.5.1 Serial Audio Controller Global Control Register (SACR0)

SACR0, defined in Table 14-5, controls common I^2S functions. All bits are read/write.

The ENB bit controls the I2SLINK, as follows:

- Clearing ENB to 0b0 does the following:
  - — Asserts SASR0[I2SOFF] bit indicating clean shutdown process
  - — Deasserts any transmit DMA request
  - — Transmits any data in the transmit FIFO
  - — Drives I2S_SDATA_OUT and I2S_SYNC outputs low
  - — Initiates a DMA transfer for the remaining data in the receive FIFO
  - — Resets the counter that controls the I2SLINK
  - — Disables any I2SLINK activity
  - — Deasserts receive DMA requests after all the data in the receive FIFO is read by DMA
  - — The output pin I2S_SYNC does not toggle.
  - — Any read accesses to the Data register (SADR), by the processor or by the DMA controller, after SASR0[I2SOFF] is cleared are returned with zeros.
  - — Disables all interrupts
- Setting ENB to 0b1 does the following:
  - — Enables I2SLINK activity
  - — Enables DMA requests

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 14-5. SACR0 Bit Definitions (Sheet 1 of 2)

Physical Address 0x4040_0000 — **SACR0** — **I^2S Controller**

Bit	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																RFTH				TFTH				reserved		STRF	EFWR	RST	BCKD	reserved	ENB
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15:12	R/W	RFTH	Receive FIFO Interrupt or DMA Trigger Threshold Set to value 0–15. This value must be set to the trigger threshold value minus 1. Receive DMA request asserted whenever the receive FIFO has ≥ (RFTH+1) entries.
11:8	R/W	TFTH	Transmit FIFO Interrupt or DMA Trigger Threshold Set to value 0–15. This value must be set to the trigger threshold value minus 1. Transmit DMA request asserted whenever the transmit FIFO has less than (TFTH + 1) entries.
7:6	—	—	reserved
5	R/W	STRF	Select Transmit or Receive FIFO for EFWR-Based Special-purpose Function 0 = Transmit FIFO is selected. 1 = Receive FIFO is selected. See Table 14-6 for details.
4	R/W	EFWR	Special-Purpose FIFO Write/Read Function 0 = Special-purpose FIFO write/read function is disabled. 1 = Special-purpose FIFO write/read function is enabled. See Table 14-6 for details.
3	R/W	RST(1)	FIFO Reset Resets FIFO logic and all registers, except this register (SACR0). 0 = Not reset. 1 = Reset is active to other registers.
2	R/W	BCKD(3)	Input/Output Direction of I2S_BITCLK 0 = Input. I2S_BITCLK driven by an external source. 1 = Output. I2S_BITCLK generated internally and driven out to the Codec.

Table 14-5. SACR0 Bit Definitions (Sheet 2 of 2)

Physical Address 0x4040_0000 — **SACR0** — **I²S Controller**

	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
User Settings																																
Bit	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																RFTH				TFTH				reserved		STRF	EFWR	RST	BCKD	reserved	ENB
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
1	—	—	reserved
0	R/W	ENB(1,2,4)	Enable I²S Function 0 = I2SLINK is disabled. 1 = I2SLINK is enabled.

NOTES:

1. If SACR0[ENB] is toggled in the middle of a normal operation, the SACR0[RST] bit must also be set and cleared to reset all I²S controller registers.
2. The SACR0[ENB] bit control signal crosses clock domains. It is registered in an internal clock domain that is much faster than the I2S_BITCLK domain. It takes four I2S_BITCLK cycles and four internal clock cycles before SACR0[ENB] is conveyed to the slower I2S_BITCLK domain. If the control setting is modified at a rate faster than (4 I2S_BITCLK + 4 internal clock) cycles, the last updated value in this time frame is stored in a temporary register and is transferred to the I2S_BITCLK domain.
3. When changing the I2S_BITCLK direction, disable the I2S_BITCLK from Clocks Manager's Clock Enable register (CKEN), change the direction and then enable the I2S_BITCLK from Clock Enable register. See Section 3.8.2.2, "Clock Enable Register (CKEN)" on page 3-98 for details.
4. Clearing SACR0[ENB] causes the controller to finish transmitting all the data in the transmit FIFO, drives I2S_SDATA_OUT and I2S_SYNC outputs low, initiates a final DMA transfer for any remaining receive data in the receive FIFO. See Section 14.5.1 for more details on SACR0[ENB].

14.5.1.1 Special-Purpose FIFO Read/Write Function

SACR0[EFWR] and SACR0[STRF] can be programmed for special-purpose FIFO accesses (see Table 14-6). Under normal operating conditions, the processor or the DMA controller can only write to the transmit FIFO and only read the receive FIFO. Programming these bits allows the processor or the DMA controller to read and write both FIFOs.

Table 14-6. I^2S FIFO Write/Read Settings

EFWR	STRF	Description
0	x	Normal CPU/DMA Write/Read Condition • A write access to the Data register writes a transmit FIFO entry. • A read access to the Data register reads out a receive FIFO entry. • I2SLINK reads from the transmit FIFO and writes to the receive FIFO.
1	0	CPU or DMA Only Writes and Reads Transmit FIFO • A write access to the Data register writes a transmit FIFO entry. • A read access to the Data register reads out a transmit FIFO entry. • I2SLINK cannot read the transmit FIFO but can write to the receive FIFO.
1	1	CPU or DMA Only Writes and Reads Receive FIFO • A write access to the Data register writes a receive FIFO entry. • A read access to the Data register reads out a receive FIFO entry. • I2SLINK can read the transmit FIFO but cannot write to the receive FIFO.

14.5.1.2 Suggested TFTH and RFTH for DMA Servicing

The DMA controller can only be programmed to send 8, 16, or 32 bytes of data. This corresponds to 2, 4, or 8 FIFO samples. Table 14-7 shows the recommended TFTH and RFTH values to prevent transmit FIFO overrun errors and receive FIFO underrun errors.

Table 14-7. TFTH and RFTH Values for DMA Servicing

DMA Transfer Size	Number of FIFO Entries	TFTH Value		RFTH Value	
		Min	Max	Min	Max
8 Bytes	2	0	14	1	15
16 Bytes	4	0	12	3	15
32 Bytes	8	0	8	7	15

14.5.2 Serial Audio Controller I^2S/MSB-Justified Control Register (SACR1)

The SACR1 register specifically controls the I^2S and MSB-justified modes. In addition, the SACR1 register controls the audio replay function, record function, and interface loop-back function. Table 14-8 shows the bit layout of SACR1.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 14-8. SACR1 Bit Definitions

Physical Address 0x4040_0004 — SACR1 — I^2S Controller

	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
User Settings																																
Bit	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																										ENLBF	DRPL	DREC	reserved		AMSL
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
31:6	—	—	reserved
5	R/W	ENLBF	Enable I^2S/MSB Interface Loopback Function 0 = I^2S/MSB interface loopback function is disabled. 1 = I^2S/MSB interface loopback function is enabled.
4	R/W	DRPL†	Disable Replaying Function of I^2S or MSB-Justified Interface 0 = Replaying function is enabled. 1 = Replaying function is disabled.
3	R/W	DREC†	Disable Recording Function of I^2S or MSB-Justified Interface 0 = Recording function is enabled. 1 = Recording function is disabled.
2:1	—	—	reserved
0	R/W	AMSL†	Specify Alternate Mode (I^2S or MSB-Justified) Operation 0 = Select normal I^2S operation mode. 1 = Select MSB-justified operation mode.

NOTE:

† SACR1 bits DRPL, DREC, and AMSL cross clock domains. They are registered in an internal clock domain that is much faster than the I2S_BITCLK domain. It takes 4 I2S_BITCLK cycles and 4 internal clock cycles before these controls are conveyed to the slower I2S_BITCLK domain. If the above control settings are modified at a rate faster than (4 I2S_BITCLK + 4 internal clock) cycles, the last updated value in this time frame is stored in a temporary register and is transferred to the I2S_BITCLK domain.

14.5.3 Serial Audio Controller I^2S/MSB-Justified Status Register (SASR0)

The Serial Audio Status register (SASR0) is used for recording the status of the FIFOs and I2SLINK. All bits are read-only. Table 14-9 shows the bit layout of SASR0.

This is a read-only register. Ignore reads from reserved bits.

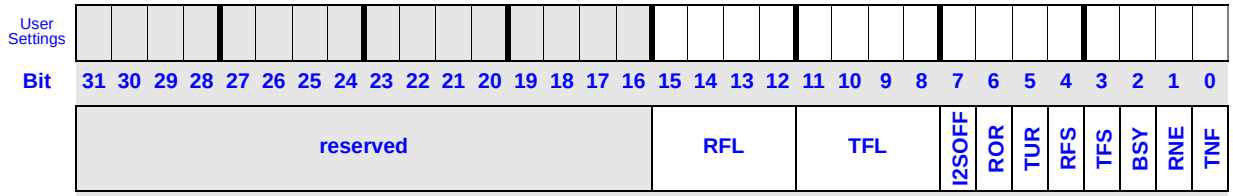
Note: When the interrupts to processor are enabled, then the interrupt to processor and DMA service requests (TFS and RFS) are generated simultaneously. Software must determine which one to service (either interrupt to processor or DMA service request).

Note: For transmitting all the data in the transmit FIFO to the Codec at shutdown, the SACR[DRPL] must be clear (replay must be enabled) before clearing SACR[ENB]. Otherwise, all the data in the transmit FIFO is flushed during clean shutdown. If the I^2S controller is in the middle of a recording frame when SACR0[ENB] is cleared, the I^2S controller waits until the end of the frame, then stops the recording. During this period, SASR0[I2SOFF] is set, indicating a clean shutdown in progress.

If software wants to re-enable the I^2S controller after disabling it, it has to wait until SASR0[I2SOFF] is clear.

Table 14-9. SASR0 Bit Definitions (Sheet 1 of 2)

Physical Address 0x4040_000C — **SASR0** — **I^2S Controller**



Bits	Access	Name	Description
31:16	—	—	reserved
15:12	R	RFL†	Receive FIFO Level Number of Entries in Receive FIFO.
11:8	R	TFL††	Transmit FIFO Level Number of Entries in Transmit FIFO.
7	R	I2SOFF	I^2S Controller Off When I2SOFF is set and SACR0[ENB] is set, it indicates that the I^2S controller is in the process of a clean shutdown. When I2SOFF is set, no write operations to the I^2S controller registers are allowed. When I2SOFF is clear and SACR0[ENB] is clear, then the unit is shut down. When I2SOFF is clear and SACR0[ENB] is set, then the I^2S controller is enabled. When SACR0[ENB] is cleared, the I2SOFF remains set until all data in the transmit FIFO is transmitted to the Codec, and all data in the receive FIFO is read by DMA.
6	R	ROR	Receive FIFO Overrun 0 = Receive FIFO has not experienced an overrun. 1 = I^2S attempted data write to full receive FIFO (interruptible). Can interrupt processor if bit 6 of Serial Audio Interrupt Mask register is set. Cleared by setting bit 6 of Serial Audio Interrupt Clear register.
5	R	TUR	Transmit FIFO Underrun 0 = Transmit FIFO has not experienced an underrun. 1 = I^2S attempted data read from an empty transmit FIFO. Can interrupt processor if bit 5 of Serial Audio Interrupt Mask register is set. Cleared by setting bit 5 of Serial Audio Interrupt Clear register.
4	R	RFS	Receive FIFO Service Request 0 = Receive FIFO level below RFL trigger threshold, or I^2S disabled. 1 = Receive FIFO level is at or above RFL trigger threshold. Can interrupt processor if bit 4 of Serial Audio Interrupt Mask register is set. Cleared automatically when number of receive FIFO entries < (RFTH + 1).
3	R	TFS	Transmit FIFO Service Request 0 = Transmit FIFO level exceeds TFL trigger threshold, or I^2S disabled. 1 = Transmit FIFO level is at or below TFL trigger threshold. Can interrupt processor if bit 3 of Serial Audio Interrupt Mask register is set. Cleared automatically when # of transmit FIFO entries is greater than or equal to (TFTH + 1).

Table 14-9. SASR0 Bit Definitions (Sheet 2 of 2)

Physical Address 0x4040_000C — SASR0 — I^2S Controller

Bit	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
User Settings																																
	reserved																RFL				TFL				I2SOFF	ROR	TUR	RFS	TFS	BSY	RNE	TNF
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bits	Access	Name	Description
2	R	BSY	I^2S Busy 0 = I^2S is idle or disabled. 1 = I^2S currently transmitting or receiving a frame.
1	R	RNE†	Receive FIFO Not Empty 0 = Receive FIFO is empty. 1 = Receive FIFO is not empty.
0	R	TNF††	Transmit FIFO Not Full 0 = Transmit FIFO is full. 1 = Transmit FIFO is not full.

NOTES:
† See Table 14-4 for more details.
†† See Table 14-3 for more details.

14.5.4 Serial Audio Clock Divider Register (SADIV)

This register is used for generating six different I2S_BITCLK frequencies and the resulting six different sampling frequencies. All bits are read/write. Table 14-10 shows the bit layout of SADIV.

The reset value, 0b0011010, defaults to a sampling frequency of 21.953 kHz (closest standard = 22.05 kHz).

Note: Setting this register to values other than those shown in Table 14-2 is not supported and causes unpredictable activity.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 14-10. SADIV Bit Definitions

Physical Address 0x4040_0060																SADIV												I²S Controller							
User Settings																																			
Bit	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																									SADIV								
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	1	1	0	1	0

Bits	Access	Name	Description
31:7	—	—	reserved
6:0	R/W	SADIV	Audio Clock Divider Valid values are: 000 1100 = Selects an I2S_BITCLK of 3.058 MHz 000 1101 = Selects an I2S_BITCLK of 2.836 MHz 001 1010 = Selects an I2S_BITCLKof 1.405 MHz 010 0100 = Selects an I2S_BITCLK of 1.026 MHz 011 0100 = Selects an I2S_BITCLK of 702.75 kHz 100 1000 = Selects an I2S_BITCLK of 513.25 kHz

14.5.5 Serial Audio Interrupt Clear Register (SAICR)

The Serial Audio Interrupt Clear register (SAICR) is the interrupt control register. This is only an addressable location and no data is actually stored. These addressable locations are used only for clearing the SASR0[6:5] status bits. Each bit position corresponds to an interrupt source bit position in the Status register. Table 14-11 shows the bit layout of SAICR.

This is a write-only register. A read operation is treated as a read from a reserved location. The reset value is reserved, since the register cannot be read.

Table 14-11. SAICR Bit Definitions

Physical Address 0x4040_0018																SAICR												I²S Controller							
User Settings																																			
Bit	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																									ROR	TUR	reserved							
Reset	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits	Access	Name	Description
31:7	—	—	reserved
6	W	ROR	Clear receive FIFO overrun interrupt and ROR status bit in SASR0 by setting this bit.
5	W	TUR	Clear transmit FIFO underrun interrupt and TUR status bit in SASR0 by setting this bit.
4:0	—	—	reserved

14.5.6 Serial Audio Interrupt Mask Register (SAIMR)

Writing 0b1 to a bit position in the Interrupt Mask register enables the corresponding interrupt signal. All bits are read/write. Table 14-12 shows the bit layout of SAIMR.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 14-12. SAIMR Bit Definitions

Physical Address 0x4040_0014																SAIMR												I²S Controller			
User Settings																															
Bit	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																									ROR	TUR	RFS	TFS	reserved		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	?	?	?

Bits	Access	Name	Description
31:7	—	—	reserved
6	R/W	ROR	Enable Receive FIFO Overrun Condition Interrupt
5	R/W	TUR	Enable FIFO Underrun Condition Interrupt
4	R/W	RFS	Enable Receive FIFO Service Request Interrupt
3	R/W	TFS	Enable Transmit FIFO Service Request Interrupt
2:0	—	—	reserved

14.5.7 Serial Audio Data Register (SADR)

Writing a 32-bit sample to this register updates the data into the transmit FIFO. Reading this register flushes a 32-bit sample from the receive FIFO.

Table 14-13 shows the bit layout of SADR. Figure 14-3 illustrates data flow through the FIFOs and SADR.

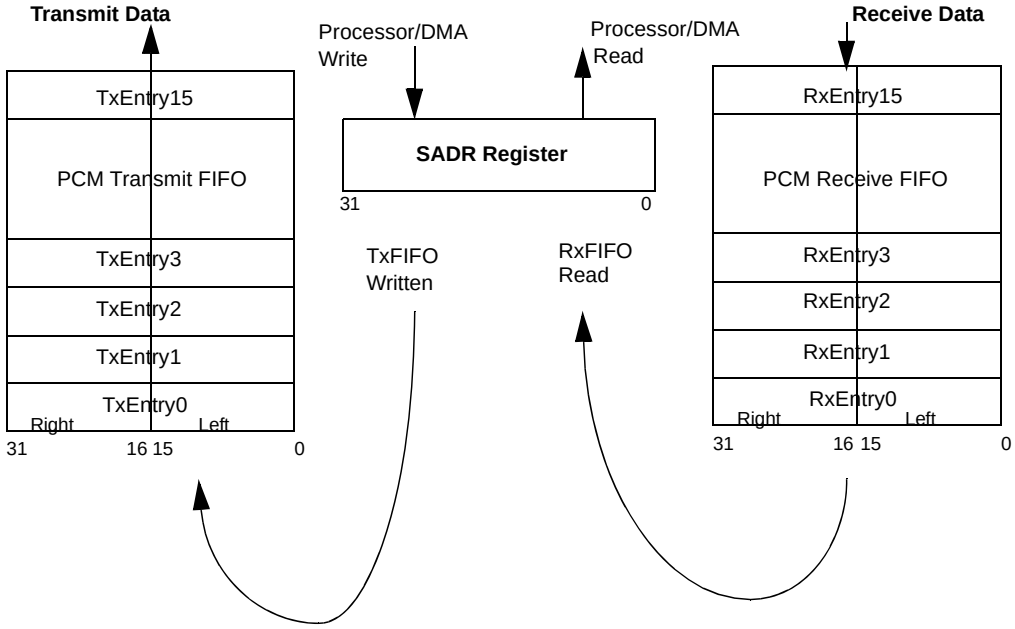
This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 14-13. SADR Bit Definitions

Physical Address 0x4040_0080																Serial Audio Data Register (SADR)												I²S Controller			
User Settings																															
Bit	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
	DTH																DTL															
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

Bits	Access	Name	Description
31:16	R/W	DTH	Right Data Sample
15:0	R/W	DTL	Left Data Sample

Figure 14-3. Transmit and Receive FIFO Accesses through the SADR



14.6 Register Summary

All registers are word addressable (32 bits wide) and increment in units of 0x00004. All I^2S controller registers are mapped in the address range of 0x4040_0000–0x404F_FFFC, as shown in Table 14-14.

Table 14-14. I^2S Register Summary (Sheet 1 of 2)

Address	Name	Description	Page
0x4040_0000	SACR0	Serial Audio Global Control register	14-10
0x4040_0004	SACR1	Serial Audio I^2S/MSB-Justified Control register	14-13
0x4040_0008	—-	reserved	
0x4040_000C	SASR0	Serial Audio I^2S/MSB-Justified Interface and FIFO Status register	14-14
0x4040_0010	—-	reserved	
0x4040_0014	SAIMR	Serial Audio Interrupt Mask register	14-18
0x4040_0018	SAICR	Serial Audio Interrupt Clear register	14-17
0x4040_001C–0x4040_005C	—-	reserved	
0x4040_0060	SADIV	Audio Clock Divider register. (See Section 14.4.8.)	14-16

Table 14-14. I^2S Register Summary (Sheet 2 of 2)

Address	Name	Description	Page
0x4040_0064– 0x4040_007C	—-	reserved	
0x4040_0080	SADR	Serial Audio Data register (TX and RX FIFO access register).	14-18
0x4040_0084– 0x404F_FFFC	—-	reserved	

MultiMediaCard/SD/SDIO Controller 15

This chapter describes the MultiMediaCard (MMC) and Secure Digital (SD/SDIO) controller and related registers supported by the PXA27x processor.

15.1 Overview

The MMC/SD/SDIO controller acts as a link between the software that accesses the PXA27x processor and the MMC stack (a set of memory cards) and supports Multimedia Card, Secure Digital, and Secure Digital I/O communications protocols. The MMC controller supports the MMC system, a low-cost data storage and communications system[1]. The MMC controller in the PXA27x processor is based on the standards outlined in the *MultiMediaCard System Specification Version 3.2*. The SD controller supports one SD or SDIO card based on the standards outlined in the *SD Memory Card Specification Version 1.01* and *SDIO Card Specification Version 1.0 (Draft 4)*.

The MMC/SD/SDIO controller supports the translation protocol from a standard MMC or serial peripheral interface (SPI) bus to the MMC stack. The software that accesses the PXA27x processor must indicate whether to use MMC/SD/SDIO or SPI mode as the protocol to communicate with the MMC/SD/SDIO controller.

15.2 Features

The MMC/SD/SDIO controller features:

- Data-transfer rates up to 19.5 Mbps for MMC, 1-bit SD/SDIO, and SPI mode data transfers
- Data-transfer rates up to 78 Mbps for 4-bit SD/SDIO data transfers
- A response FIFO
- Two transmit FIFOs and two receive FIFOs
- Two modes of operation: MMC/SD/SDIO mode and SPI mode. MMC/SD/SDIO mode supports MMC, SD, and SDIO communications protocols. SPI mode supports the SPI communications protocol.
- 1- and 4-bit data transfers are supported for SD and SDIO communications protocols.
- Controller turns clock on and off, based on status of FIFOs, to prevent overflows and underruns.
- Support for all valid MMC and SD/SDIO protocol data-transfer modes
- Interrupt-based application interface to control software interaction
- For stream writes, only data sizes of 10 bytes or more are allowed.
- Using the MMC communications protocol, multiple MMC cards are supported.
- Using the SD or SDIO communications protocol, one SD or SDIO card is supported.

1. A detailed description of the MMC system is available on the MMC Association's web site at http://www.mmca.org

- Using the SPI communications protocol, up to two MMC or SD/SDIO cards are supported. Mixed card types are supported for the SPI communications protocol only.

15.3 Signal Descriptions

The MMC/SD/SDIO controller signals are described in Table 15-1.

Table 15-1. MultiMediaCard/SD/SDIO Controller I/O Signal Descriptions

Signal	MMC and SD/SDIO	SPI	Description
MMCLK	Output	Output	MMC and SD/SDIO bus clock
MMCMD	Bidirectional	Output	MMC and SD/SDIO: Bidirectional line for command and response tokens SPI: Output for command and write data
MMDAT<0>	Bidirectional	Input	MMC and SD/SDIO: Bidirectional line for read and write data SPI: Input for response token and read data
MMDAT<1>	Bidirectional	Input	MMC and SD/SDIO: Used for SD/SDIO 4 bit data transfers and to signal SDIO interrupts to the controller SPI: Signals SDIO interrupts to the controller
MMDAT<2>/ MMCCS<0>	Bidirectional	Output	SD/SDIO: 4-bit data transfer SPI: CS0 chip select
MMDAT<3>/ MMCCS<1>	Bidirectional	Output	SD/SDIO: 4-bit data transfer SPI: CS1 chip select

The MMCLK, MMCCS<0>, and MMCCS<1> signals are routed through alternate functions within the GPIO pins. Refer to Chapter 24, “General-Purpose I/O Controller” for a description of how to assign these signals to a specific GPIO. Even though there are several GPIOs assigned to each signal, each signal must be programmed to one of the possible GPIOs. Refer to Section 24.4.2, “GPIO Operation as Alternate Function” on page 24-3 for a complete description of the GPIO alternate functions.

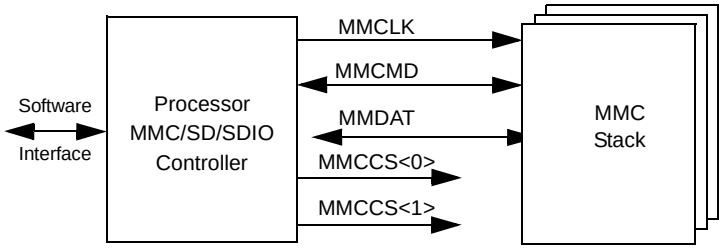
15.4 Operation

The MMC/SD/SDIO controller contains all card-specific functions, serves as the bus master for the MMC system, and implements the standard interface to the card stack. The controller handles card initialization; CRC generation and validation; and command, response, and data transactions.

The MMC/SD/SDIO controller is a slave to the software and consists of command and control registers, a response FIFO, and data FIFOs. The software has access to these registers and FIFOs and generates commands, interprets responses, and controls subsequent actions.

Figure 15-1 shows a block diagram of the interaction of a typical MMC system using the MMC communications protocol.

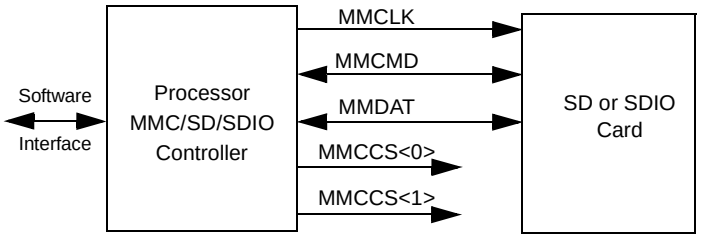
Figure 15-1. MMC (MMC Protocol) System Interaction



Notes:
MMCCS<0> and MMCCS<1> are used in SPI mode only.
MMCCS<0> and MMDAT<2> are multiplexed.
MMCCS<1> and MMDAT<3> are multiplexed.

Figure 15-2 shows a block diagram of the interaction of a typical SD/SDIO system using the SD or SDIO communications protocol.

Figure 15-2. SD/SDIO (SD or SDIO Protocol) System Interaction



Notes:
MMCCS<0> and MMCCS<1> are used in SPI mode only.
MMCCS<0> and MMDAT<2> are multiplexed.
MMCCS<1> and MMDAT<3> are multiplexed.

The MMC bus connects the card stack to the controller. The software and controller can turn the MMCLK on and off. The card stack and the controller communicate serially through the command and data lines and implement a message-based protocol. The messages consist of the following tokens:

- Command—A 6-byte command token starts an operation. The command set includes card initialization, card register reads and writes, and data transfers. The MMC/SD/SDIO controller sends the command token serially on the MMCMD signal. The format for a command token is shown in Table 15-2.

Table 15-2. Command Token Format

Bit Position	47	46	[45:40]	[39:8]	[7:1]	0
Width (bits)	1	1	6	32	7	1
Value	0	1	x	x	x	1
Description	start bit	transmission bit	command index	argument	CRC7	end bit

- Response—A response token is an answer to a command token. Each command has either a specific response type or no response type. The format for a response token varies according to the response expected and the card's mode. Response token formats are detailed in the *MultiMediaCard System Specification, Version 3.1*.
- Data—Is transferred serially between the controller and the card in 8-bit blocks at rates up to 19.5 Mbps for MMC, 1-bit SD/SDIO, and SPI mode data transfers. Data is transferred at rates up to 78 Mbps for 4-bit SD/SDIO data transfers. The format for the data token depends on the card's mode. Table 15-3 shows the data token format for MMC/SD/SDIO mode.

Table 15-3. MMC/SD/SDIO Data Token Format

Stream Data	1	x	no CRC	1
Block Data	0	x	x	1
Description	start bit	data	CRC7	end bit

In MMC/SD/SDIO mode, all operations contain command tokens and most commands have an associated response token. read and write commands also have a data token. Command and response tokens are sent and received on the bidirectional MMCMD signal and data tokens are sent and received on the bidirectional MMDAT signal.

Figure 15-3. MMC/SD/SDIO Mode Operation Without Data Token

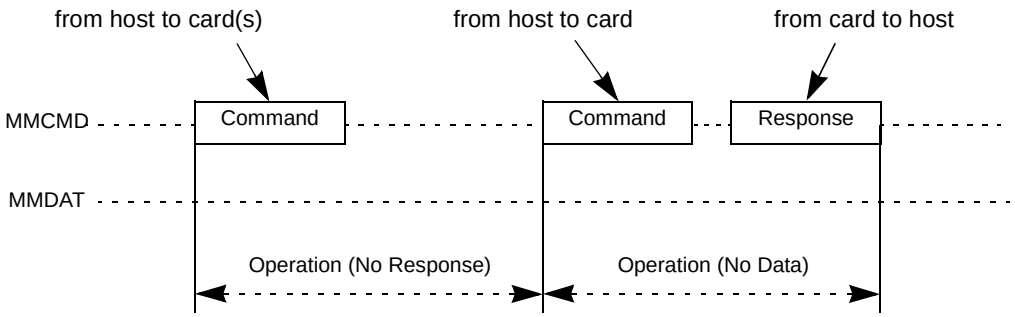
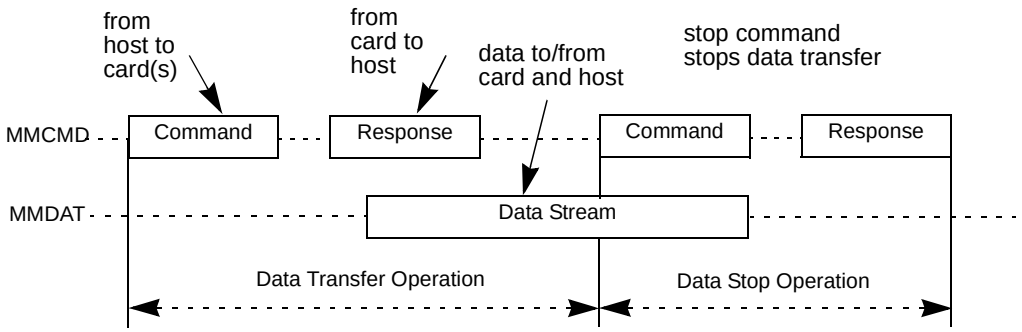


Figure 15-4. MMC/SD/SDIO Mode Operation With Data Token



In SPI mode, not all commands are available. The available commands have both a command and response token. The MMCMD and MMDAT signals are no longer bidirectional in SPI mode. The MMCMD is an output and the MMDAT is an input with respect to the PXA27x processor. The command and data tokens to be written are sent on the MMCMD signal, and the response and read data tokens are received on the MMDAT signal. Figure 15-5 shows a typical SPI mode timing diagram without a data token. Figure 15-6 and Figure 15-7 show SPI mode read and write timing diagrams, respectively.

Figure 15-5. SPI Mode Operation Without Data Token

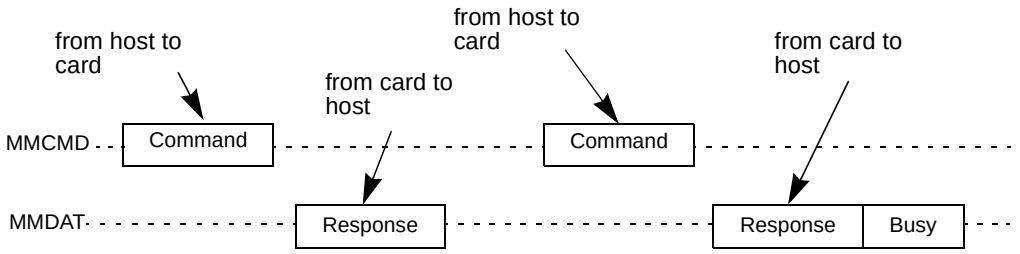


Figure 15-6. SPI Mode Read Operation

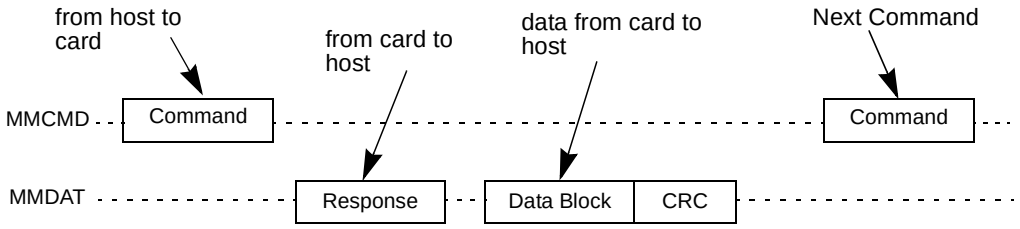
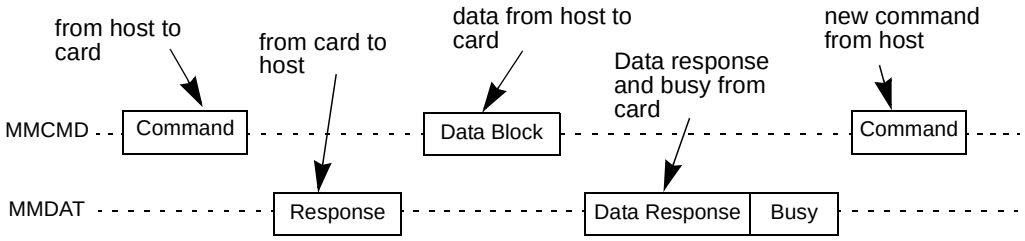


Figure 15-7. SPI Mode Write Operation



Note: Data transfers of 10 or more bytes are supported for stream writes only.

Refer to the *MultiMediaCard System Specification* for detailed information on MMC and SPI modes of operation.

The MMC/SD/SDIO controller can interface to the cards with MMC protocol, SD/SDIO protocol or SPI protocol. All protocols are serial interfaces to the cards, as shown in Table 15-4.

- The MMC protocol supports block, multiple-block, and stream data transfers.
- The SD/SDIO protocol supports block and multiple-block data transfers.
- The SPI protocol supports block and multiple-block data transfers.
- SD/SDIO and SPI protocols do not support stream data transfers. Only MMC cards support stream data transfers.

Table 15-4. MMC/SD/SDIO Data Transfer Types

Data Transfer Types	MMC Protocol	SD and SDIO Protocol	SPI Protocol
Single block	Supported	Supported	Supported
Multiple block—open-ended	Supported	Supported	Supported
Multiple block—predefined	Supported	Supported	Supported
Stream	Supported	Not supported	Not supported
RW_IO Direct, CMD52	Not supported	Supported for SDIO only	Supported for SDIO only
RW_IO Extended, CMD53	Not supported	Supported for SDIO only	Supported for SDIO only

15.4.1 MMC/SD/SDIO Mode

In the MMC/SD/SDIO mode, the MMCMD and MMDAT lines are bidirectional and require external pull-ups. The command and response tokens are sent on the MMCMD line. Data is sent on the MMDAT line(s) and is a multiple of bytes.

In the MMC protocol, card addressing is implemented during the initialization phase with address assignment. A card is then addressed by an address in the command token.

In the SD/SDIO protocol, card addressing is implemented with point-to-point MMCMD and MMDAT lines.

The command token is protected with a 7-bit CRC. The response token has five types of coding schemes, including the “no-response” token. The response token length is 48 or 136 bits, and can be protected with a 7-bit CRC, depending on the response type.

Read and write data transfers are protected with a 16-bit CRC. In write data transfers, after the data and 16-bit CRC has been transmitted, the card sends a 5-bit CRC status token. The CRC status token indicates if the data transmission was erroneous. After the CRC status token, the card can indicate that it is busy programming the data by pulling the MMDAT<0> line low.

15.4.1.1 MMC/SD/SDIO Mode Data Transfers

The MMC/SD/SDIO mode supports the following data transfer modes:

- Single block read/write—Supported by MMC and SD/SDIO protocols
- Multiple block read/write—Supported by MMC and SD/SDIO protocols
  - Open-ended multiple block read/write
  - Multiple block read/write with pre-defined block count
- Stream read/write—Supported by MMC protocol only.
- IO_RW_DIRECT command (CMD52)—Supported by SDIO protocol only.
- IO_RW_EXTENDED command (CMD53)—Supported by SDIO protocol only.

All data transfers can be stopped at any time by the application with a MMC/SD Stop Transmission command, CMD12, or SDIO abort with CMD52 and ASx bits set.

15.4.1.1.1 Single Block

In *single block data transfers*, a single block of data is transmitted. The starting address is specified in the read/write command.

The application must tell the controller the block size. The *block size* is the number of bytes to be transferred. The data block is protected with a 16-bit CRC that is generated by the transmit unit, and checked by the receive unit. The CRC is appended after the transfer of the last data bit.

15.4.1.1.2 Multiple Block

Multiple block data transfers are similar to the single block data transfers, except multiple blocks of data are transferred sequentially. Each block is the same length, and is stored or retrieved from contiguous memory addresses, starting at the address specified in the command.

Two types of multiple block data transfers are defined.

- Open-ended multiple block read/write

  The number of blocks to be transferred is not defined in the card. The card continuously transfers data blocks until a Stop Transmission command, CMD12, is received.

- Multiple block read/write with predefined block count

  The number of blocks to be transferred is defined in the card. The card transfers only the number of data blocks specified. A Stop Transmission command, CMD12, or abort command (CMD52 with ASx bits set) is not required at the end of the data transfer in this case, unless the data transmission terminated with an error.

To start a multiple block data transmission with a pre-defined block count, a SET_BLOCK_COUNT command (CMD23) must immediately precede the multiple block command (CMD18/CMD25).

The application must tell the controller the block size and the number of blocks to transfer. Each data block is protected with a 16 bit CRC.

If the card detects an error during either type of multiple block read operation, the card stops data transmission. The application must then stop the transmission with the Stop Transmission command, CMD12.

If the card detects an error during either type of multiple block write operation, the card ignores any further incoming data. The application must then stop the transmission with the Stop Transmission command, CMD12.

The application can stop a data transmission at any time. An MMC/SD Stop Transmission command, CMD12, or an SDIO abort with CMD52 with ASx bits set terminates multiple block data transfers, regardless of type. Neither CMD12 or CMD52 is needed to stop transmission at the end of a pre-defined multiple block data transfer.

15.4.1.1.3 Stream

In *stream data transfers*, the controller transmits a continuous stream of data. The starting address is specified in the read/write command.

The data stream is terminated with a Stop Transmission command, CMD12. For write transfers, CMD12 must be aligned with the last six bytes of data to ensure that no more and no less data is written into the card. For read transfers, CMD12 can occur after the data has been transmitted. There is no CRC protection on the data with stream data transfers.

For stream write data transfers, only data sizes of 10 bytes or more are allowed.

SPI, SD and SDIO cards do not support stream mode.

15.4.1.1.4 SDIO Data Transfer

In addition to single and multiple block data transmission commands, SDIO supports two additional data transfer commands to support I/O, IO_RW_DIRECT, CMD52, and IO_RW_EXTENDED, CMD53.

The IO_RW_DIRECT command is similar to the MMC “Fast I/O” command and allows direct access to any single register within the 128k of register space in any I/O function. The command reads/writes1 byte to the single I/O register specified.

The IO_RW_EXTENDED command allows the read/write of a a large number of I/O registers. The data transfer may be in byte mode or block mode. The byte mode is similar to the single block data transfer, and the block mode is similar to the multiple block data transfer.

15.4.2 SDIO Mode

This section gives a description of SDIO-specific commands and operations.

15.4.2.1 SDIO Overview

SDIO cards are based on and compatible with SD cards. The intent of the SDIO card is to provide high-speed data I/O with low power consumption for mobile electronic devices.

Some features of SDIO are:

- Plug and play (PnP) support
- Multi-function support, including multiple I/O and combined I/O and memory
- Up to seven I/O functions plus one memory supported on one card
- Allows cards to interrupt application
- Read_Wait operation
- Suspend/resume operation

15.4.2.2 New I/O Read/Write Commands

SDIO includes two new data transfer commands, IO_RW_DIRECT (CMD52) and IO_RW_EXTENDED (CMD53) to support I/O.

15.4.2.2.1 IO_RW_DIRECT command (CMD52)

IO_RW_DIRECT allows the simplest access to a single register within the 128 Kbytes of register space in any I/O function. The command is similar to the MMC “Fast I/O” command (CMD39).

In SDIO, CMD52 is also used to replace the MMC/SD Stop Transmission command, CMD12. CMD52 can be used to write to the SDIO card’s CCCR abort register to abort a data transfer.

Consult *the SDIO Card Specification* for a description of the IO_RW_DIRECT command.

15.4.2.2.2 IO_RW_EXTENDED Command (CMD53)

IO_RW_EXTENDED allows the read/write of multiple I/O registers with a single command. I/O block operations use CMD53, rather than MMC/SD memory block read/write commands. CMD53 supports multi-byte transfer modes and block mode, which are analogous to MMC/SD single and multiple block data transfers.

Multi-byte mode does a read or write of multiple bytes of data to/from a single I/O register. Block mode does a read or write of multiple bytes of data to/from an I/O register address that is incremented by 1 after each operation/block.

In SDIO, CMD53 is similar to the following MMC/SD commands for memory data transfer.

- In multi-byte mode:
    - READ_SINGLE_BLOCK (CMD17)
    - WRITE_SINGLE_BLOCK (CMD24)
- In block mode:
    - READ_MULTIPLE_BLOCK (CMD18)
    - WRITE_MULTIPLE_BLOCK (CMD25)

Multi-byte/block mode is specified in the command argument. In the block mode, the number of blocks to be transferred is specified in the command argument. Therefore, the application does not need to stop the data transmission as in the MMC/SD multiple block data transfers, because the number of blocks of data transferred is known by the card and the controller.

Consult *the SDIO Card Specification* for a description of the IO_RW_EXTENDED command.

15.4.2.3 SDIO Data Transfer Aborts

The application can issue an I/O abort by writing the SDIO card's CCCR register with CMD52 at any time during an I/O extended read or write data transfer. The abort stops the data transmission. On data writes, the abort occurs between data blocks. Also, after an I/O card receives an abort on a data write, the card can respond busy after sending the CMD52 response.

Consult *the SDIO Card Specification* for a description of how to use CMD52 command to abort a data transmission.

15.4.2.4 SDIO Interrupts

An SDIO card is allowed to interrupt the host by asserting the MMDAT<1> line low. The card continues to keep the MMDAT<1> line low until the interrupt is either recognized and acted on by the host or the interrupt is deasserted due to the end of the SDIO interrupt period.

Consult *the SDIO Card Specification* for a description of SDIO interrupts.

15.4.2.5 SDIO Suspend/Resume

In SDIO, the application can temporarily halt (suspend) a data transfer to one function or memory to free the SDIO bus for a higher priority data transfer to a different function or memory. Once the higher priority data transfer has completed, the application can re-start (resume) the suspended data transfer from where it left off.

Note that the application can suspend multiple transactions and resume them in any order desired.

The suspend/resume operation works for SD/SDIO 1 and 4-bit modes. It does not apply to SPI mode.

Consult *the SDIO Card Specification* for a description of SDIO suspend/resume operation.

15.4.2.6 SDIO Read Wait

In MMC/SD cards, the controller can stop the clock to stop a read data transfer in-order to prevent a RXFIFO overrun. The controller restarts the clock and continues the data transfer when the RXFIFO is ready for more data. This stopping of the clock is a limitation for SDIO, because while the clock is stopped, a CMD52 cannot be issued.

SDIO adds a read wait control to enable the host to send a CMD52. With READ-WAIT, the host uses the MMDAT<2> line to signal the card to temporarily halt the sending of read data by a card. This allows a CMD52 to be sent while the data is halted.

When the application requests a READ_WAIT from the controller, the stall of the data transfer may not occur immediately. Therefore, the RD_WAIT request from the controller does not stop the RXFIFOs from turning the clock off/on to prevent overflows. Therefore, the application must continue to read the RXFIFOs during the RD_WAIT operation.

The RD_WAIT operation is only supported for multiple block read data transfers, in SDIO 1-bit and 4-bit modes.

The RD_WAIT operation is not supported in SPI mode.

Consult *the SDIO Card Specification* for a description of SDIO RD_WAIT operation.

15.4.3 MMC/SD/SDIO Controller Functional Description

The software must read and write the MMC/SD/SDIO controller registers and FIFOs to initiate communications to a card.

The MMC/SD/SDIO controller is the interface between the software and the MMC/SD/SDIO bus. It is responsible for the timing and protocol between the software and the MMC/SD/SDIO bus. It consists of control and status registers, a 16-bit response FIFO that is eight entries deep, two 8-bit receive data FIFOs that are 32 entries deep, and two 8-bit transmit FIFOs that are 32 entries deep. The registers and FIFOs are accessible by the software.

The MMC/SD/SDIO controller also enables minimal data latency by buffering data and generating and checking CRCs.

15.4.3.1 MMC/SD/SDIO Controller Reset

The MMC/SD/SDIO controller can only be reset by a hard or soft reset of the PXA27x processor. Resetting PXA27x processor and the MMC/SD/SDIO controller is described in Section 2.7, “Reset” on page 2-8. All registers and FIFO controls are set to their default values after any reset.

15.4.3.2 Card Initialization Sequence

After reset, the MMC card must be initialized by sending 80 clocks to it on the MMCLK signal. To initialize the MMC card, set the MMC_CMDAT[INIT] bit to a 1. This sends 80 clocks before the current command in the MMC_CMD register. This function is useful for acquiring new cards that have been inserted on the bus. Chip selects are not asserted during the initialization sequence while in SPI mode.

After the 80-clock initialization sequence, the software must continuously send CMD1 by loading the appropriate command index into the MMC_CMD register until the card indicates that the power-up sequence is complete. The software can then assign an address to the card or put it into SPI mode.

15.4.3.3 MMC/SD/SDIO and SPI Modes

After reset, the MMC card is in MMC mode. The card can remain in MMC mode or be configured to SPI mode by setting the MMC_SPI register bits. The following sections briefly describe each mode as it pertains to the MMC/SD/SDIO controller.

15.4.3.3.1 MMC Mode

In MMC mode, the MMCMD and MMDAT<0> signals are bidirectional and require external pull-ups. The command and response tokens are sent and received by the MMCMD signal, and data is read and written with the MMDAT<0> signal.

After an MMC card is powered on, it is assigned a default relative card address (RCA) of 0x0001. The software assigns different addresses to each card during the initialization sequence described in Section 15.4.3.2. A card is then addressed by its new relative address in the argument portion of the command token that is protected with a 7-bit CRC (see Table 15-2). For a description of the identification process when multiple cards are connected to a system, refer to the Card Identification Process as described in *The MultiMediaCard System Specification*.

There are five formats for the response token, including a no response token. The response token length is 48 or 136 bits and can be protected with a 7-bit CRC. Details of the response token can be found in the *MultiMediaCard System Specification.*

In write data transfers, the data is suffixed with a 5-bit CRC status token from the card. After the CRC status token, the card can indicate that it is busy by pulling the MMDAT<0> line low.

The start address for a read operation can be any random byte address in the valid address space of the card memory. For a write operation, the start address must be on a sector boundary, and the data length must be an integer multiple of the sector length. A *sector* is the number of blocks that are erased during the write operation and is fixed for each MMC card. A *block* is the number of bytes to be transferred.

The MMC mode supports the following data transfer modes:

- Single block read/write—In single block mode, a single block of data is transferred. The starting address is specified in the command token of the read or write command used. The software must set the block size in the controller by entering the number of bytes to be transferred in the MMC_BLKLEN register. The data block is protected with a 16-bit CRC that is generated by the sending unit and checked by the receiving unit. The CRC is appended to the data after the last data bit is transferred.
- Multiple block read/write—In multiple block mode, multiple blocks of data are transferred. Each block is the same length as specified by the software in the MMC_BLKLEN register. The blocks of data are stored or retrieved from contiguous memory addresses starting at the address specified in the command token. The software specifies the number of blocks to transfer in the MMC_NUMBLK register. Each data block is protected by appending a 16-bit CRC. Multiple block data transfers are terminated with a stop transmission command.
- Stream read/write—In stream mode, a continuous stream of data is transferred. The starting address is specified in the command token of the read or write command used. The data stream is terminated with a stop transmission command. For write transfers, the stop transmission command must be transmitted with the last six bytes of data. This ensures that the correct amount of data is written to the card. For read transfers, the stop transmission command can occur after the data transmission has occurred. There is no CRC protection for data in this mode.

15.4.3.3.2 SPI Mode

SPI mode is an optional secondary communications protocol. In SPI mode, the MMCMD and MMDAT<0> lines are unidirectional. The MMCMD signal is an output from the controller and sends the command token and write data to the MMC/SD/SDIO card. The MMDAT<0> signal is an input to the controller and receives the response token and read data from the MMC/SD/SDIO card.

Note: When the card is in SPI mode, the only way to return to MMC/SD/SDIO mode is by toggling the power to the card.

Card addressing is implemented with hardware chip selects, MMCCS<1> and MMCCS<0>. All command, response, and data tokens are 8-bits long and are transmitted immediately following the assertion of the respective chip select.

The command token is protected with a 7-bit CRC. The card always sends a response to a command token. The response token has four formats, including an 8-bit error response. The length of the response tokens is one, two, or five bytes.

SPI mode offers a non-protected mode. In this mode, CRC bits of the command, and data tokens are still required in the tokens but these bits are ignored by the card and the controller.

In write data transfers, the data is suffixed with an 8-bit CRC status token from the card. As in MMC/SD/SDIO mode, the card can indicate that it is busy by pulling the MMDAT<0> line low after the status token. In read data transfers, the card can respond with the data or a one-byte data error token.

15.4.3.4 Response and Data Error Detection

The MMC/SD/SDIO controller detects response and data errors on the MMC/SD/SDIO bus and reports them in the Status register MMC_STAT. Response errors are also recorded in the RES_ERR interrupt bit. Data errors are also recorded in the DAT_ERR interrupt bit. If a response or data error occurs, the bit is set in the MMC_I_REG and an interrupt is generated to the application, if the appropriate mask bit is cleared. The application can either check for an interrupt or poll the MMC_I_REG.

Table 15-5. Response and Data Errors

Error	MMC	SD/SDIO	SPI	Description
SDIO_INT	no	yes, SDIO only	yes, SDIO only	SDIO interrupt from card occurred
RD_STALLED	no	yes, SDIO only	no	SDIO read data is stalled
FLSH_ERR	no	yes	no	A flash programming error occurred
RES_CRC_ERR	yes	yes	no	A CRC error was calculated on the command response
DAT_ERR_TOKEN	no	no	yes	In SPI mode a read-data error token was detected
CRC_RD_ERR	yes	yes	yes	A CRC error was calculated on read data
CRC_WR_ERR	yes	yes	yes	A CRC error was detected by the card on write data
TIME_OUT_RES	yes	yes	yes	A response time-out occurred
TIME_OUT_READ	yes	yes	no	A read data time-out occurred
SPI_WR_ERR	no	no	yes	An SPI write data rejected error occurred

The SPI_WR_ERR is checked for and recorded in the MMC_STAT register.

In SPI mode write multiple block, the MMC/SD/SDIO controller stops data transmission with the Stop Tran token, if any of the following errors occur:

- Data rejected due to a CRC error—Error reported in MMC_STAT[CRC_WR_ERR].
- Data rejected due to a write error—Error reported in MMC_STAT[SPI_WR_ERR].

In SPI mode, the user must abort a multiple block read with a Stop Transmission command, CMD12, if a read-data error token is sent by the card. The error is reported in MMC_STAT[DAT_ERR_TOKEN].

15.5 Interrupts

The MMC/SD/SDIO controller generates interrupts to signal the status of a command sequence. The software is responsible for masking the interrupts appropriately, verifying the interrupts, and performing the appropriate action as necessary.

Interrupts shown in Table 15-6 and their masking are described in Section 15.9.11 and Section 15.9.12. The CMDAT[DMA_EN] bit also masks the MMC_I_MASK[RXFIFO_RD_REQ] and MMC_I_MASK[TXFIFO_WR_REQ] interrupt bits.

Table 15-6. MMC/SD/SDIO Controller-Generated Interrupts

Interrupt	Description
SDIO_INT	An SDIO card interrupt occurred.
RD_STALLED	Read data transfer has been stalled.
RES_ERR	An error occurred on the command response.
DAT_ERR	A data error occurred during data transmission.
TXFIFO_WR_REQ	For programmed I/O, TXFIFO request to write FIFO. This is never asserted if using the DMA.
RXFIFO_RD_REQ	For programmed I/O, RXFIFO request to read FIFO. This is never asserted if using the DMA.
CLK_IS_OFF	Asserted when controller turns the clock off because the application wrote to the MMC_STRPCL register to turn the clock off.
STOP_CMD	For stream mode writes, indicates that the controller is ready for the Stop-Transmission command.
END_CMD_RES	Asserted when the command and the response transfers have completed.
PRG_DONE	Asserted when a write data transfer has completed and the card is no longer busy programming or when a command has an R1b response and the card is no longer busy.
DATA_TRAN_DONE	Asserted when a data transfer has completed or timed-out.

15.6 Clock Control

Both the MMC/SD/SDIO controller and the software can control the MMC/SD/SDIO bus clock (MMCLK) by turning it on and off. This helps to control the data flow to prevent underruns and overflows and also conserves power. The software can also change the frequency at any time to achieve the maximum data transfer rate specified for a card's identification frequency.

The MMC/SD/SDIO controller has an internal frequency generator that can start, stop, and divide the MMC/SD/SDIO bus clock. The software can start and stop the clock by setting the appropriate bits in the MMC_STRPCL register. The MMCLK frequency is controlled by the value written in the MMC_CLKRT register.

The application is not required to stop the clock before writing any MMC/SD/SDIO controller registers, except the MMC_CLKRT register. Do not stop and start the clock for a CMD52 overlapping a data transfer. Not stopping the clock for each command sequence can save a maximum of 127 cycles for each command sequence, because it may take a maximum of 127 cycles for the clock to stop after setting MMC_STRPCL[STOP_CLK].

Software can specify the clock divisor for the 19.5-MHz clock in the MMC_CLKRT register. The clock rate can be set over a range from 304 kHz to 19.5 MHz. For instructions and full details, see Section 15.9.3.

The controller can also turn the clock off automatically. If both receive FIFOs become full during data reads, or one receive FIFO is being read by the software and the other receive FIFO becomes full, or both transmit FIFOs become empty during data writes, or one transmit FIFO is being written by the software and the other transmit FIFO is empty, the controller automatically turns the clock off to prevent data overflows and underruns. For read data transfers, the controller turns the clock back on after a receive FIFO has been emptied. For write data transfers, the controller turns the clock back on after the transmit FIFO is no longer empty.

If the software stops the clock at any time, it must wait for the MMC_STAT[CLK_EN] status bit to be cleared or the CLK_IS_OFF interrupt before proceeding.

15.7 Data FIFOs

The controller FIFOs for the response tokens, received data, and transmitted data are MMC_RES, MMC_RXFIFO, and MMC_TXFIFO, respectively. These FIFOs are accessible by the software and are described in the following sections.

15.7.1 Response Data FIFO (MMC_RES)

The response FIFO, MMC_RES, contains the response received from an MMC/SD/SDIO card after a command is sent from the controller. MMC_RES is a read-only, 16-bit, and 8-entry deep FIFO.

The FIFO holds all possible response lengths. Responses that are only one byte long are located on the LSB of the 16-bit entry in the FIFO. For reads of odd byte length responses, the last byte is located on the LSB of the 16-bit entry in the FIFO.

The FIFO does not contain the response CRC. The status of the CRC check is in MMC_STAT[RES_CRC_ERR].

15.7.2 Receive Data FIFO (MMC_RXFIFO)

The two receive data FIFOs are read-only by the software and are readable on a one, two, or four byte basis. Each receive data FIFO is 32 entries of 1-byte data. Access to the FIFOs is controlled by the controller and depends on the status of the FIFOs.

Both FIFOs and their controls are cleared to a starting state after a system reset and at the beginning of all command sequences except for the Stop Transmission command, CMD12, or IO_RW_DIRECT, CMD52.

The FIFOs swap between the software and MMC/SD/SDIO bus. At any time, while the software has read access to one of the FIFOs, the MMC/SD/SDIO bus has write access to the other FIFO.

For purposes of an example, the FIFOs are called RXFIFO1 and RXFIFO2. After a reset or at the beginning of a command sequence, both FIFOs are empty and the software has read access to RXFIFO1 and the MMC/SD/SDIO bus has write access to RXFIFO2. When RXFIFO2 becomes full and RXFIFO1 is empty, the FIFOs swap and the software has read access to RXFIFO2 and the MMC/SD/SDIO bus has write access to RXFIFO1. When RXFIFO1 becomes full and RXFIFO2 is empty, the FIFOs swap and the software has read access to RXFIFO1 and the MMC/SD/SDIO bus has write access to RXFIFO2.

This swapping process continues through out the data transfer and is transparent to both the software and the MMCMMC/SD/SDIO controller.

If at any time both FIFOs become full and the data transmission is not complete, the controller turns the MMCLK off to prevent any overflows. When the clock is off, data transmission from the card stops until the clock is turned back on. After the software has emptied the FIFO that it is connected to, the controller turns the clock on to continue data transmission.

Some examples are:

- Receive 96 bytes of data:

Read 32 bytes three times.

For the DMA, use three descriptors of 32 bytes and 32-byte bursts.

- Receive 98 bytes of data:

Read 32 bytes three times, then read two more bytes.

For the DMA, use three descriptors of 32 bytes and 32-byte bursts and one descriptor of two more bytes and 8-, 16-, or 32- byte bursts.

- Receive 105 bytes:

Read 32 bytes three times, then read nine more bytes.

For the DMA, use three descriptors of 32 bytes and 32-byte bursts and one descriptor of nine or more bytes and 16- or 32-byte bursts.

15.7.3 Transmit Data FIFO (MMC_TXFIFO)

The two transmit data FIFOs are written only by the software and are writable on a one, two, or four byte basis. Each transmit data FIFO is 32 entries of one byte data. Access to the FIFOs is controlled by the controller and depends on the status of the FIFOs.

Both FIFOs and their controls are cleared to a starting state after a system reset and at the beginning of all command sequences except for the Stop Transmission command, CMD12, or IO_RW_DIRECT, CMD52.

The FIFOs swap between the software and MMC bus. At any time, while the software has write access to one of the FIFOs, the MMC bus has read access to the other FIFO.

For purposes of an example, the FIFOs are called TXFIFO1 and TXFIFO2. After a reset or at the beginning of a command sequence, both FIFOs are empty and the software has write access to TXFIFO1 and the MMC/SD/SDIO bus has read access to TXFIFO2. When TXFIFO1 becomes full and TXFIFO2 is empty, the FIFOs swap and the software has write access to TXFIFO2 and the MMC/SD/SDIO bus has read access to TXFIFO1. When TXFIFO2 becomes full and TXFIFO1 is empty, the FIFOs swap and the software has write access to TXFIFO1 and the MMC/SD/SDIO bus has read access to TXFIFO2.

This swapping process continues through out the data transfer and is transparent to both the software and the MMC/SD/SDIO controller.

If at any time both FIFOs become empty and the data transmission is not complete, the controller turns the MMCLK off to prevent any underruns. When the clock is off, data transmission to the card stops until the clock is turned back on. When the transmit FIFO is no longer empty, the MMC/SD/SDIO controller automatically restarts the clock.

If the software does not fill the FIFO to which it is connected, the MMC_PRTBUF[BUF_PART_FULL] bit must be set. This enables the FIFOs to swap without filling the FIFO.

Some examples are:

- Transmit 96 bytes of data:

Write 32 bytes three times.

For the DMA, use three descriptors of 32 bytes and 32-byte bursts.

- Transmit 98 bytes of data:

Write 32 bytes three times, then write two more bytes.

For the DMA, use three descriptors of 32 bytes and 32-byte bursts and one descriptor of two more bytes and 8-, 16- or 32-byte bursts and program the descriptor to set an interrupt, for the software to write the MMC_PRTBUF[BUF_PART_FULL] bit.

- Transmit 105 bytes:

Write 32 bytes three times, then write nine more bytes.

For the DMA, use three descriptors of 32 bytes and 32-byte bursts and one descriptor of nine more bytes and 16- or 32-byte bursts and program the descriptor to set an interrupt, for the software to write MMC_PRTBUF[BUF_PART_FULL] bit.

15.7.4 DMA and Programmed I/O

The software may communicate to the MMC/SD/SDIO controller using the DMA or programmed I/O.

To access the FIFOs by DMA, the software must program the DMA to read or write the MMC/SD/SDIO FIFOs with single-byte transfers and 32-byte bursts. For example, to write 64 bytes of data to the MMC_TXFIFO, the software must program the DMA to write 64 bytes with an 8-bit port size to the MMC/SD/SDIO controller and for 32-byte bursts. The MMC/SD/SDIO controller issues a request to read the MMC_RXFIFO and a request to write the MMC_TXFIFO.

With programmed I/O, the software waits for the MMC_I_REG[RXFIFO_RD_REQ] or MMC_I_REG[TXFIFO_WR_REQ] interrupts before reading or writing the respective FIFO by clearing the appropriate mask bits. Or the application may poll the MMC_I_REG register for the FIFO request by disabling the interrupt in the interrupt controller, or setting the appropriate mask bits to a 1. A maximum of 32 bytes can be read/written for each interrupt.

The CMDAT[DMA_EN] bit must be set to enable communication with the DMA and cleared to enable programmed I/O.

15.8 MMC/SD Card Communications Protocol

This section discusses requirements for software and the communications protocols used between the MMC/SD/SDIO controller and the card.

15.8.1 Start and Stop Clock

The software stops the clock, as follows:

1. Write 0x01 in MMC_STRPCL to stop the MMC/SD/SDIO clock.
2. Write 0x0f in MMC_I_MASK to mask all interrupts except the MMC_I_REG[CLK_IS_OFF] interrupt.
3. Wait for the MMC_I_REG[CLK_IS_OFF] interrupt.

To start the clock, the software writes 0x02 in MMC_STRPCL.

15.8.2 Enabling SPI Mode

To communicate with a card in SPI mode, the software must set the MMC_SPI register as follows:

1. MMC_SPI[SPI_EN] must be set.
2. MMC_SPI[SPI_CS_EN] must be set.
3. MMC_SPI[SPI_CS_ADDRESS] must be set to specify the card that the software wants to address. A 0 enables CS0 and a 1 enables CS1.

Note: When the card is in SPI mode, the only way to return to MMC/SD/SDIO mode is by toggling power to the card.

15.8.3 Basic, No Data, and Command-Response Sequence

The MMC/SD/SDIO controller performs the basic MMC/SD or SPI bus transaction. It formats the command from the command registers and generates and appends the 7-bit CRC if applicable. It then serially translates this to the MMCMD bus, collects the response data, and validates the response CRC. It also checks for response time-outs and card busy if applicable. The response data is in the MMC_RES FIFO and the status of the transaction is in the status register, MMC_STAT.

The protocol of events for the software is:

1. Write to the MMC_I_MASK register and wait for and verify the MMC_I_REG[CLK_IS_OFF] interrupt.

2. Write to the following registers, as necessary:
   — MMC_CMD
   — MMC_ARGH
   — MMC_ARGL
   — MMC_CLKRT
   — MMC_SPI
   — MMC_RESTO
   — MMC_CMDAT, this register must be written, even if there is no change to the register.
3. Write to the MMC_I_MASK register and wait for and verify the MMC_I_REG[END_CMD_RES] interrupt.
4. Read the MMC_RES FIFO and MMC_STAT registers.

Some cards can become busy as the result of a command. The software can wait for the card to become not busy by writing the MMC_I_MASK register and waiting for the MMC_I_REG[PRG_DONE] interrupt or the software can start communication to another card. The software may not access the same card again until the card is no longer busy. Refer to the *MultiMediaCard System Specification* for additional information.

15.8.4 Data Transfer

A data transfer is a command and response sequence with the addition of a data transfer to a card.

The software must follow the steps as described in Section 15.8.3. In addition, before starting the clock, the software must write the following registers as necessary:

- MMC_RDTO
- MMC_BLKLEN
- MMC_NUMBLK

After the software writes the registers and starts the clock, the software must read the MMC_RES as described above and read or write the MMC_RXFIFO or MMC_TXFIFO FIFOs.

After completely reading or writing the data FIFOs, the software must wait for the appropriate interrupts. The status register, MMC_STAT, must be read to ensure that the transaction is complete and to check the status of the transaction.

When using DMA request signals, the controller indicates to the DMA when a FIFO is ready for reading or writing. It is expected that all FIFO reads and writes will empty and fill the FIFO to which it is connected. If at any time the MMC_TXFIFO is not filled (32 bytes) by the software, the software must notify the controller by setting the MMC_PRTBUF[BUF_PART_FULL]. The software can write more bytes of data than is needed into the MMC_TXFIFO, but the controller only transmits the number of bytes in the MMC_BLKLEN register.

At the end of any data transfer or busy signal on the MMC/SD/SDIO bus, the MMC/SD/SDIO controller waits eight MMCLKs before asserting the MMC_I_REG[DATA_TRAN_DONE] interrupt to notify the software that the data transfer is complete. This guarantees that the specified minimum of eight MMCLKs occurs between a data transfer and the next command.

On write data transfers, a card can become busy while programming the data. The software can wait for the card to become not busy by writing the MMC_I_MASK register and waiting for the MMC_I_REG[PRG_DONE] interrupt or the software can start communication to another card. Refer to the *MultiMediaCard System Specification* for additional information.

The MMC/SD/SDIO controller performs data transactions in all the basic modes: single block, multiple blocks, and stream modes.

15.8.4.1 Block Data Write

In a single block data write, a block of data is written to a card. In a multiple block write, the controller performs multiple single block write data transfers on the MMC/SD/SDIO bus.

After turning the clock on to start the command sequence, the software must program the DMA to fill the MMC_TXFIFO (write 32 bytes). The software must continue to fill the FIFO until all of the data has been written to the FIFOs. The software must then wait for the transmission to complete by waiting for the MMC_I_REG[DATA_TRAN_DONE] interrupt and MMC_I_REG[PRG_DONE] interrupt. The software can then read the status register, MMC_STAT, to verify the status of the transaction.

For multiple block writes, the *MultiMediaCard System Specification* specifies that the card continue to receive blocks of data until the stop transmission command is received. After the controller has transmitted the number of bytes specified in the MMC_NUMBLK register, the controller stops transmitting data. After the MMC_I_REG[DATA_TRAN_DONE] interrupt is detected, the software must setup the controller to send the stop transmission command, CMD12. Consult the *MultiMediaCard System Specification* for a description of the stop transmission command.

If both transmit FIFOs become empty during data transmission, the MMC/SD/SDIO controller turns the clock off. After a FIFO has been written, the controller turns the clock back on.

In a block data write, the following parameters must be specified:

- The data transfer is a write.
- The block length if the block length is different from the previous block data transfer or this is the first time that the parameter is being specified.
- The number of blocks to be transferred.

15.8.4.2 Block Data Read

In a single block data read, a block of data is read from a card. In a multiple block read, the controller performs multiple single block read data transfers on the MMC/SD/SDIO bus.

After turning the clock on to start the command sequence, the software must program the DMA to empty the MMC_RXFIFO (read 32 bytes). The software continues the process of emptying the FIFO until all of the data has been read from the FIFO. The software must then wait for the transmission to complete by waiting for the MMC_I_REG[DATA_TRAN_DONE] interrupt. The software can then read the register MMC_STAT register to verify the status of the transaction.

For multiple block reads, the *MultiMediaCard System Specification* specifies that the card continue to send blocks of data until the stop transmission command is received. After the controller has received the number of bytes specified in the MMC_NUMBLK register, the controller stops

receiving data. After the MMC_I_REG[DATA_TRAN_DONE] interrupt is detected, the software must set up the controller to send the stop transmission command, CMD12. Consult the *MultiMediaCard System Specification* for a description of the stop transmission command.

If both receive FIFOs become full during the data transmission, the controller turns the clock off. Once the software empties the FIFO to which it is connected, the controller turns the clock back on.

In a block data read, the following parameters must be specified:

- The data transfer is a read.
- The block length, if the block length is different from the previous block data transfer or this is the first time that the parameter is being specified.
- The number of blocks to be transferred.
- The receive data time-out period.

The controller marks the data transaction as timed out if data is not received before the time-out period. The delay for the time-out period is defined as:

$$\text{Time-Out Delay} = \frac{(\text{MMC\_RDTO[READ\_TO]}) \times (13128)}{10^9}\text{sec}$$

The software is required to calculate this value and write the appropriate value into the MMC_RDTO register.

15.8.4.3 Stream Data Write

The stream data write looks like the single block write, except that a stop transmission command is sent in parallel with the last six bytes of data.

After turning the clock on to start the command sequence, the software must start the process of filling the MMC_TXFIFO and starting the clock as describe in Section 15.8.4.1. The software must then wait for the MMC_I_REG[STOP_CMD] interrupt. This interrupt indicates that the MMC/SD/SDIO controller is ready for the stop transmission command. The software must then write the registers for a stop transmission command. At this point, the software must wait for the MMC_I_REG[DATA_TRAN_DONE] and MMC_I_REG[PRG_DONE] interrupts.

In a stream data write, the following parameters must be specified:

- The data transfer is a write.
- The data transfer is in stream mode.
- The block length, if the block length is different from the previous block data transfer or this is the first time that the parameter is being specified.
- The number of blocks to be transferred as 0xFFFF.

15.8.4.4 Stream Data Read

The stream data read looks like the single block read except that a stop transmission command must be sent after the data transfer.

After turning the clock on to start the command sequence, the software must start the process of reading the MMC_RXFIFO, as described in Section 15.8.4.2.

When it uses DMA, the software must also configure the DMA to send an interrupt after all data has been read. After the DMA interrupt or the program has read all of the data, the software must send the stop transmission command.

In a stream data read, the following parameters must be specified:

- The data transfer is a read.
- The data transfer is in stream mode.
- The block length, if the block length is different from the previous block data transfer or this is the first time that the parameter is being specified.
- The number of blocks to be transferred as 0xFFFF.
- The receive data time-out period.

15.8.4.5 Stop Data Transmission, Randomly

Single block, multiple block, and stream data transmissions can be stopped during the data transmission by setting the STOP_CLK_CMD12 in the MMC_STRPCL register. Doing so stops the clock and allows the user to set the control registers for CMD12 after the CLK_IS_OFF interrupt has occurred.

15.8.5 Busy Sequence

The MMC/SD/SDIO controller expects a busy signal automatically from the card after every block of data for single and multiple block write operations. It also expects a busy at the end of a command every time the software specifies that a busy signal is expected. For example, a busy signal is expected after the commands for stop transmission, card select, erase, and program CID. Refer to the *MultiMediaCard System Specification* for more information.

While a busy signal is on the MMC/SD/SDIO bus, the software can send only one of two commands:

- Send status command (CMD13)
- Disconnect command (CMD7)

If the software disconnects a card while it is in a busy state, the busy signal is turned off and the software can connect a different card. The software may not start another command sequence on the same card while the card is busy.

15.8.6 SPI Functionality

The MMC/SD/SDIO controller can address two cards in SPI mode. Once the software specifies the card address and enables the chip select, the chip select signal is driven active low at the negative edge of the MMCLK after two MMCLKs. The software deasserts the chip select when it has performed at least one of the following:

- Turned the chip select enable off
- Selected a different card

The software must turn the chip select off between command sequences even if they are to the same card to maintain the byte-alignment in SPI mode. This is required because the *MultiMediaCard System Specification* specifies that the SPI mode must byte align all commands and data to the chip select falling edge.

The software specifies the card address in the MMC_SPI register. The address can be changed for every command.

In SPI mode, the software has the option of performing a CRC check. The default is no CRC checking.

The command and data are sent on the MMC/SD/SDIO bus aligned to every 8 clocks as described in the SPI section of the *MultiMediaCard System Specification*.

In a read sequence, the card can return data or a data error token. If a data error token is received, the controller stops the transmission and updates the status register.

15.8.7 SDIO Card Communications Protocol

SDIO cards can do data transfers in 1-bit or 4-bit mode. The 1-bit or 4-bit data transfers are enabled in the controller with MMC_CMDAT [SD_4DAT].

15.8.8 Basic, No Data, Command-Response Sequence

The MMC/SD/SDIO controller performs the basic MMC/SD/SDIO bus transaction by formatting the command from the command registers, MMC_CMD, CMD_ARGH and CMD_ARGL, generating and appending the 7-bit CRC, if applicable, and then serially transmits the command and CRC to the MMC/SD/SDIO CMD bus. The MMC/SD/SDIO controller then collects the response data, validates the CRC, and checks for response time-outs and “card busy,” if applicable. The response data is located in the MMC_RES FIFO, and the transaction status is located in the Status register, MMC_STAT.

The protocol of events for the application is as follows:

1. Write the control registers, as necessary.
2. Write the MMC_CMDAT register.
3. Either poll MMC_STAT [END_CMD_RES] or unmask MMC_I_MASK [END_CMD_RES] and wait for the END_CMD_RES interrupt from the controller.
4. Read command response in MMC_RES FIFO.
5. Wait for PRG_DONE by polling or interrupt, if the command is expected to respond busy.
6. Check the status in MMC_STAT register.

The command sequence starts on the MMC/SD/SDIO bus after MMC_CMDAT has been written.

Some cards can become busy as the result of a command. The application can wait for the card to become “not busy” by writing the MMC_I_MASK register, and waiting for the PRG_DONE interrupt or polling the MMC_STATUS register. Or, the application can start communicating with another card. The application cannot access the same card again until the card is no longer busy. Refer to the *SDIO Card Specification* for additional details.

15.8.9 Data Transfer

A data transfer is a command-response sequence with the addition of a data transfer to/from a card.

The application must follow the events as described in Section 15.8.8.

After the application writes the registers, the application must begin the read/write of the MMC_RXFIFO/MMC_TXFIFO, wait for END_CMD_RES, and read the MMC_RES, as described in Section 15.8.3.

After the application has completed reading/writing the data FIFOs, it must wait for the appropriate interrupts and verify the Status register, MMC_STAT, to ensure that the transaction is complete and as a check of the Transaction status.

Using request signals, the controller indicates when a FIFO is ready for reads/writes.

All application FIFO reads/writes empty/fill the FIFO to which it is connected. If at any time the MMC_TXFIFO is not filled (32 bytes) by the application, the application must notify the controller by setting the MMC_PRTBUF register. The application can write more data bytes than necessary into the MMC_TXFIFO; the controller only transmits the number of bytes in the MMC_BLKLEN register.

At the end of any MMC/SD/SDIO bus data transfer, the MMC/SD/SDIO controller notifies the application that the data transfer is complete with the MMC_DATA_TRAN_DONE interrupt. If the card is not busy after a data write, the MMC_PRG_DONE interrupt is asserted.

On write-data transfers, a card can become busy programming the data. The application can wait for the card to become “not busy” by writing the MMC_I_MASK register, and waiting for the PRG_DONE interrupt. Or, the application can start communicating to another card. Refer to the *SDIO Card Specification* for additional details.

The MMC/SD/SDIO controller performs data transactions in all the basic modes: single block, multiple blocks, and stream.

15.8.9.1 Block Data Write

In a single block data write, a block of data is written to a card. In a multiple block write, the controller repeatedly performs the single block write data transfer on the MMC/SD/SDIO bus.

The protocol of events for the application is as follows:

1. Write the control registers, as necessary.
2. Write the MMC_CMDAT register.
3. Begin writing the data TXFIFOs and wait for END_CMD_RES before reading the command response in MMC_RES_FIFO.
4. Wait for DATA_TRAN_DONE and PRG_DONE by polling or interrupt.
5. Read MMC_STAT register.

After starting the command sequence, the application must begin writing the MMC_TXFIFO and continue writing until all the data has been written in the FIFO. The application must then wait for the DATA_TRAN_DONE and PRG_DONE by polling or interrupt, and then read the status register, MMC_STAT, to verify the Transaction status.

For MMC/SD/SDIO open-ended multiple block writes, the Stop Transmission command, CMD12, must be sent to the card after the data transmission is complete. For SPI multiple block writes, the controller terminates the transmission with a “Stop-Tran” token. Therefore, SPI mode multiple block writes do not require a CMD12 after the data transmission is complete. Consult the *SDIO Card Specification* for a description of the Stop Transmission command and SPI mode “Stop-Tran” token.

In a block data write, the following parameters must be defined:

- Specify the data transfer is a write
- Specify the block length if it is different from the previous Block data transfer, or the parameter is being specified for the first time
- Specific the number of blocks to be transferred.

15.8.9.2 Block Data Read

In a single block data read, a block of data is read from a card. In a multiple block read, the controller repeatedly performs the single block-read data transfer on the MMC/SD/SDIO bus.

The protocol of events for the application is as follows:

1. Write the control registers, as necessary.
2. Write the MMC_CMDAT register.
3. Begin reading the data RXFIFOs and wait for END_CMD_RES before reading the command response in MMC_RES_FIFO.
4. Wait for DATA_TRAN_DONE by polling or interrupt.
5. Read MMC_STAT register.

After starting the command sequence, the application must begin reading the MMC_RXFIFO and continue reading until all of the data has been read from the FIFO. The application must then wait for DATA_TRAN_DONE by polling or interrupt, and then read the Status register, MMC_STAT, to verify the Transaction status.

For open-ended multiple block reads, the Stop-Transmission command, CMD12, must be sent to the card after the data transmission is complete. Consult the *SDIO Card Specification* for a description of the Stop Transmission command.

In a block-data read, the following parameters must be defined:

- Specify the data transfer is a read
- Specify the block length if it is different from the previous block-data transfer, or the parameter is being specified for the first time
- Specific the number of blocks to be transferred
- Specify the receive data time-out period.

The controller marks the data transaction as timed out if data is not received before the time-out period. The number of cycles in the time-out period is defined as follows:

Read time-out = (MMC_RDTO * 13128) ns

The application must calculate this value.

15.8.9.3 Stream Data Write

Stream data writes are not supported by SD/SDIO cards.

15.8.9.4 Stream Data Read

Stream data reads are not supported by SD/SDIO cards.

15.8.9.5 Stop Data Transmission Command (CMD12 or I/O ABORT with CMD52)

In MMC and SD protocols, a data transmission is stopped with Stop Transmission command, CMD12. In SDIO data transmission is stopped with CMD52 by setting the SDIO ASx register bits, to abort the data transmission.

The STOP_TRAN bit in MMC_CMDAT register must be set when using CMD12 or CMD52 to stop a data transmission. If using CMD52 to abort a data transmission to a different function other than the current function data transmission, the STOP_TRAN bit must not be set.

Since CMD12 and CMD52 abort are sent in parallel with a data transfer, the data control signals in MMC_CMDAT register, SD_4DAT, DMA_EN, STRM_BLK, WR_RD, DATA_EN, must also be set as for the previous data transfer command. Also, read/write service to RXFIFO/TXFIFO must continue during CMD12 and CMD52.

The application may stop any data transmission at any time with CMD12 or CMD52. However, the application may not send a CMD12 during SPI mode writes.

15.8.10 Overlapping a Command with a Data Transfer

Any command without a data transfer can be issued during a data transfer from a previous command. The control registers for the overlapping command may not be written until after END_CMD_RES has asserted, either by interrupt or polling.

Since the command is sent in parallel with a data transfer, the data control signals in MMC_CMDAT register, SD_4DAT, DMA_EN, STRM_BLK, WR_RD, DATA_EN, must also be set as for the previous data transfer command. Also, read/write service to RXFIFO/TXFIFO must continue during the overlapping command.

Issuing a new command with a data transfer while a previous command’s data transfer is on the MMC/SD/SDIO bus is not supported by this controller.

15.8.11 Busy Sequence

A card can respond busy after any data block for single and multiple block write operations or after any R1b type response. The card responds busy by pulling the MMDAT<0> line low and stops pulling the MMDAT<0> line low when it is no longer busy. Consult the *SDIO Card Specification* for details.

The MMC/SD/SDIO controller checks for busy automatically after every data block for single and multiple block write operations. For commands with R1b type responses, the user must set MMC_CMDAT[BUSY]. If BUSY is set, the controller checks for busy after the command response.

When the card is not busy or stops being busy, the controller asserts MMC_STAT[PRG_DONE] and sets the PRG_DONE interrupt if it is not masked.

A busy signal on the MMC/SD/SDIO bus means the application can send only these two commands:

- Send-Status command (CMD13)
- Disconnect command (CMD7)

If the application disconnects a card while in a Busy state, the busy signal is turned off, and the application can connect a different card. The application cannot start another command sequence on the same card while the card is busy.

15.8.12 SPI Functionality

The MMC/SD/SDIO controller can access two cards in SPI mode. Once the application has specified the card address and enabled the chip select (CS), the CS signal is clear, active low, at the negative edge of the MMCLK after two MMCLKs. The CS is set back to 1 only when the application has performed at least one of the following two options:

- Turn off the CS enable, bit 2 of MMC_SPI register
- Select a different card, bit 3 of MMC_SPI register.

The address can be changed for every command. The application specifies the card address in the MMC_SPI register. In SPI mode, the application can specify whether to perform a CRC check with the CRC_ON bit in the MMC_SPI register. The default is no CRC verification.

A multiple block write data transfer in SPI mode is terminated by the controller with a “Stop Tran” token. A multiple block read data transfer in SPI mode is terminated with a Stop Transmission command, CMD12, generated by the application.

If the card detects an error during a multiple block read operation, either type, the card stops data transmission. The application must then stop the transmission with the Stop Transmission command, CMD12.

If the card detects an error during a multiple block write operation, either type, the card ignores any further incoming data. The controller then stops the transmission with the “Stop Tran” token. The application may not send a Stop Transmission command, CMD12 during an SPI mode write data transmission.

The command and data are sent on the MMC/SD/SDIO bus aligned to every 8 clocks as described in the SPI section of the *SDIO Card Specification*.

15.8.13 SDIO Interrupts

An SDIO card can implement an interrupt to the controller. If the SDIO_INT_EN bit in the MMC_CMDAT register is set, the controller monitors and report any interrupts from the card. If an interrupt occurs the SDIO_INT bit in the MMC_STATUS register is set and an interrupt to the application occurs if the SDIO_INT mask bit is cleared in the MMC_I_MASK register.

15.8.14 SDIO Suspend/Resume

The SDIO suspend/resume operation allows the application to suspend a data transfer to a SDIO function or memory to free the bus for a higher priority data transfer to a different function or memory. When the higher priority data transfer is complete, the original data transfer can be resumed.

The suspend operation suspends any data transfer if it has not even started on the bus or between blocks for multiple block data transfers.

The application can suspend any data transfer by sending CMD52 during the data transfer. The argument of CMD52 must be set to suspend the current function. The SDIO_SUSPEND bit in the MMC_CMDAT register must be set. When the card acknowledges the suspend, the SDIO_SUSPEND_ACK bit in MMC_STAT and MMC_I_REG is asserted. Also the number of blocks that have been suspended is written in the MMC_BLKS_REM. The application is responsible to read the MMC_BLKS_REM register and save the value for when the data transfer is resumed.

The data transfer may complete before the suspend is acknowledged by the card. In this case, DATA_TRAN_DONE asserts.

For multiple block writes or CMD53 block mode write, when the SDIO_SUSPEND bit is set, if the data transfer has not started, the controller does not send any data to the card. If a data transfer is in progress when the SDIO_SUSPEND bit is set, the controller stops sending data to the card after the current data block is finished. The number of blocks that were not transferred is MMC_BLKS_REM.

The application may resume any suspended data transfer by sending CMD52. The command argument must be set to resume the function and the SDIO_RESUME bit in MMC_CMDAT register must be set. Also the MMC_NUMBLK must be written the number of blocks that were in MMC_BLKS_REM when the data transfer is suspended.

If DMA descriptors are being used to read/write the FIFOs, the application must stop the DMA channel when the SDIO_SUSPEND_ACK bit is asserted.

15.8.15 SDIO Read Wait

The SDIO READ_WAIT operation allows the application to temporarily stall a data transfer. This operation is only supported for SDIO multiple block read data transfers.

If READ_WAIT is activated, the controller signals the card to enter the read wait state by driving MMDAT<2> low at the end of a data block. During the read Wait time, the application can communicate with any SDIO card function using CMD52. The application can restart the stalled read data transfer at any time.

The MMC_RDWAIT register controls the READ_WAIT operation. The RD_WAIT_EN bit in MMC_RDWAIT register enables the controller to drive MMDAT<2> low at the end of a data block and therefore stalling the data transfer from the card. The RD_WAIT_START bit in the MMC_RDWAIT register enables the controller to restart the stalled data transfer.

When a data transfer is stalled, the RD_STALLED bit in the MMC_STATUS register is asserted and the RD_STALLED interrupt occurs, if the appropriate mask bit is set.

The application must continue reading of MMC_RXFIFO to prevent the FIFO from turning the clock off. Therefore, the waiting for RD_STALLED interrupt must be done in parallel to reading the MMC_RXFIFO.

15.9 Register Descriptions

15.9.1 MMC Clock Start/Stop Register (MMC_STRPCL)

MMC_STRPCL, defined in Table 15-7, allows the software to start and stop the MMCLK. The register is cleared after the clock is started or stopped.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-7. MMC_STRPCL Bit Definitions

Physical Address 0x4110_0000 — **MMC_STRPCL** — **MMC/SD/SDIO**

	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
User Settings																																
Bit	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																														STRT_CLK	STOP_CLK
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?0	?	?	?	?	?	0	0

Bits	Access	Name	Description
31:2	—	—	reserved
1	R/W	STRT_CLK	Start the MMCLK 1 = Starts the MMCLK and then the bit is automatically cleared.
0	R/W	STOP_CLK	Stop the MMCLK 1 = Stops the MMCLK and then the bit is automatically cleared.

15.9.2 MMC Status Register (MMC_STAT)

MMC_STAT, defined in Table 15-8, is the status register for the MMC/SD/SDIO controller. The register is cleared at the beginning of every command sequence.

This is a read-only register. Ignore reads from reserved bits.

Table 15-8. MMC_STAT Bit Definitions

Physical Address 0x4110_0004 — **MMC_STAT** — **MMC/SD/SDIO**

Bit	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
User Settings																																
	reserved															SDIO_SUSPEND_ACK	SDIO_INT	RD_STALLED	END_CMD_RES	PRG_DONE	DATA_TRAN_DONE	SPI_WR_ERR	FLASH_ERR	CLK_EN	reserved	reserved	RES_CRC_ERR	DAT_ERR_TOKEN	CRC_RD_ERR	CRC_WR_ERR	TIME_OUT_RES	TIME_OUT_READ
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bits	Access	Name	Description
31:17	—	—	reserved
16	R	SDIO_SUSPEND_ACK	1 = SDIO data transfer has been suspended by SDIO card
15	R	SDIO_INT	1 = SDIO interrupt occurred
14	R	RD_STALLED	1 = Read data transfer has been stalled in response to RD_WAIT
13	R	END_CMD_RES	1 = Command and response sequence has completed
12	R	PRG_DONE	1 = Card has finished programming and is not busy
11	R	DATA_TRAN_DONE	1 = Data transmission to card has completed
10	R	SPI_WR_ERR	1 = Write data rejected by card due to a write error
9	R	FLASH_ERR	1 = Flash programming error occurred
8	R	CLK_EN	1 = MMC/SD/SDIO Clock, MMCLK, is on
7:6	—	—	reserved
5	R	RES_CRC_ERR	1 = CRC error occurred on the response
4	R	DAT_ERR_TOKEN	1 = SPI data error token has been received
3	R	CRC_RD_ERR	1 = CRC error occurred on received data
2	R	CRC_WR_ERR	1 = Write data rejected by card due to a CRC error
1	R	TIME_OUT_RES	1 = Card response timed out
0	R	TIME_OUT_READ	1 = Card read data timed out

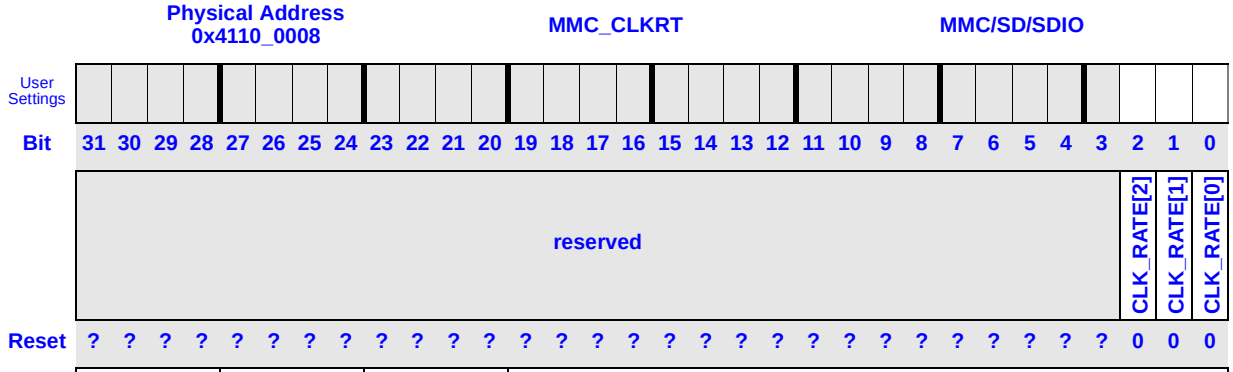
15.9.3 MMC Clock Rate Register (MMC_CLKRT)

MMC_CLKRT, defined in Table 15-9, specifies the frequency division of the MMCLK. The software is responsible for setting this register.

Software can write to this register only after the clock is turned off and software has received an interrupt that indicates that the clock is turned off.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-9. MMC_CLKRT Bit Definitions



Bits	Access	Name	Description
31:3	—	—	reserved
2:0	R/W	CLK_RATE [0:2]	Clock Frequency 0b000 = 19.5 MHz (no division) 0b001 = 9.75 MHz (19.5 MHz / 2) 0b010 = 4.88 MHz (19.5 MHz / 4) 0b011 = 2.44 MHz (19.5 MHz / 8) 0b100 = 1.22 MHz (19.5 MHz / 16) 0b101 = 609 kHz (19.5 MHz / 32) 0b110 = 304 kHz (19.5 MHz / 64) 0b111 = reserved

15.9.4 MMC SPI Mode Register (MMC_SPI)

The MMC_SPI register, defined in Table 15-10, configures the MMC/SD/SDIO controller for SPI mode. The register is for SPI mode only and is set by the software.

Both MMC_SPI[SPI_MODE] and MMC_SPI[SPI_CS_EN] must be set to configure the MMC/SD/SDIO controller for SPI mode. Otherwise, the MMC/SD/SDIO controller remains in MMC/SD/SDIO mode.

For example: If an SPI card is connected to MMCCS<1>, then to communicate to the card in DPI mode:

- SPI_MODE must be set.
- SPI_CS_EN must be set.
- SPI_CS_ADDRESS must be set.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-10. MMC_SPI Bit Definitions

Physical Address 0x4110_000c																MMC_SPI												MMC/SD/SDIO			
User Settings																															
Bit	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																												SPI_CS_ADDRESS	SPI_CS_EN	SPI_CRC_EN	SPI_MODE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0

Bits	Access	Name	Description
31:4	—	—	reserved
3	R/W	SPI_CS_ ADDRESS	Relative Address of the Card to Activate SPI CS 0 = Enables MMCCS<0> 1 = Enables MMCCS<1>
2	R/W	SPI_CS_EN	SPI Chip Select Enable 0 = Disables the SPI chip select 1 = Enables the SPI chip select
1	R/W	SPI_CRC_ EN	CRC Generation Enable 0 = Disables CRC generation and verification 1 = Enables CRC generation and verification
0	R/W	SPI_MODE	SPI Mode Enable 0 = Disables SPI mode 1 = Enables SPI mode

15.9.5 MMC Command/Data Register (MMC_CMDAT)

The MMC_CMDAT register controls the command sequence. Writing to this register starts the command sequence on the MMC/SD/SDIO bus when MMCLK is turned on.

Writing the MMC_CMDAT register clears the MMC_STAT register and clears the FIFOs unless the STOP_TRAN bit is being written to a 1 or the command is SDIO CMD52.

If a command is to be sent in parallel with a data transfer, the data control signals in MMC_CMDAT register, SD_4DAT, DMA_EN, STRM_BLK, WR_RD, DATA_EN, must also be set as for the previous data transfer command. Also, read/write service to RXFIFO/TXFIFO must continue during the overlapping command.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-11. MMC_CMDAT Register

Physical Address 0x4110_0010 — MMC_CMDAT — MMC/SD/SDIO

Bit	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
User Settings																																
	reserved																		SDIO_RESUME	SDIO_SUSPEND	SDIO_INT_EN	STOP_TRAN	reserved	SD_4DAT	DMA_EN	INIT	BUSY	STRM_BLK	WR_RD	DATA_EN	RES_TYPE[1]	RES_TYPE[0]
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:14	—	—	reserved
13	R/W	SDIO_ RESUME	1 = SDIO CMD52, resume a suspended data transfer
12	R/W	SDIO_ SUSPEND	1 = SDIO CMD52, suspend current data transfer
11	R/W	SDIO_ INT_EN	1 = Enables controller to check for an SDIO interrupt from the card
10	R/W	STOP_TRAN	1 = Stop data transmission
9	—	—	reserved
8	R/W	SD_4DAT	0 = Enable 1 bit data transfers 1 = Enable 4 bit data transfers, valid for SD/SDIO protocol only
7	R/W	DMA_EN	0 = Programmed I/O access to FIFOs 1 = DMA access to FIFOs When DMA mode is used, this bit is a mask on RXFIFO_RD_REQ and TXFIFO_WR_REQ interrupts.
6	R/W	INIT	1 = Precede command sequence with 80 clocks, for initialization
5	R/W	BUSY	1 = Specifies whether a busy signal is possible after the current command sequence This bit is for no data, command/response transactions only.
4	R/W	STRM_BLK	1 = Data transfer of the current command sequence is in stream mode
3	R/W	WR_RD	1 = Data transfer of the current command sequence is a write operation
2	R/W	DATA_EN	1 = Current command includes a data transfer
1:0	R/W	RES_TYPE	These bits specify the response format for the current command. Refer to Table 15-12.

Table 15-12. CMD_DAT_CONT RES_TYPE Bit Definitions

RES_TYPE	MMC Mode Response Format	SD/SDIO Mode Response Format	SPI Mode Response Format
00	No response	No response	No response
01	R1, R4, R5	R1, R6	R1
10	R2	R2	R2
11	R3	R3	R3

15.9.6 MMC Response Time-Out Register (MMC_RESTO)

The MMC_RESTO register controls the number of MMCLKs the controller must wait after the command before it can turn on the time-out error if a response has not occurred. The default value of this register is 64.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-13. MMC_RESTO Bit Definitions

Physical Address 0x4110_0014	MMC_RESTO	MMC/SD/SDIO

User Settings																															
Bit	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																									RES_TO						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	0	0	0	0	0	0

Bits	Access	Name	Description
31:7	—	—	reserved
6:0	R/W	RES_TO	Number of MMCLKs before a Response Time-Out

15.9.7 MMC Read Time-Out Register (MMC_RDTO)

The MMC Read Time-Out register specifies the number of clocks following the command after which the controller indicates a received-data time-out error. The units for this register is 13128 ns. For example, if MMC_RDTO[READ_TO] is set to 0b10, the controller waits 26256 ns after the response end bit for the data start; if data is not read, it reports a time-out.

Note: Because it takes a minimum of 2 cycles after the CMD End bit before the data Start bit occurs, software must set the value of MMC_RDTO[READ_TO] to 0b10 or greater.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-14. MMC_RDTO Bit Definitions

Physical Address 0x4110_0018	MMC_RDTO	MMC/SD/SDIO

User Settings																															
Bit	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																READ_TO[15:0]															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R/W	READ_TO	Length of Time before a Data Read Time-Out

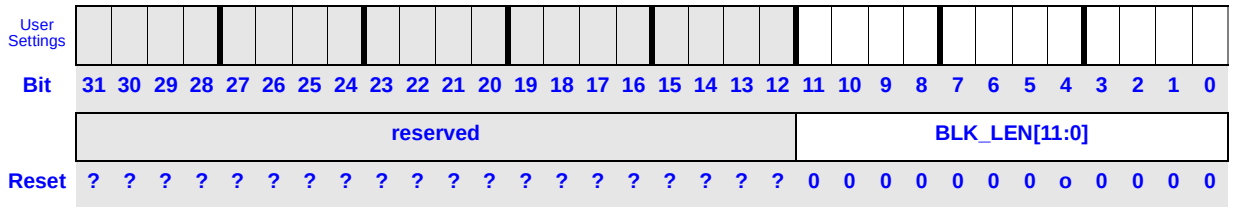
15.9.8 MMC Block Length Register (MMC_BLKLEN)

The MMC_BLKLEN register specifies the number of bytes in a block of data. The number of bytes in a block can be up to 2048 bytes. This register must always be set to a value greater than 0 for data transfers.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-15. MMC_BLKLEN Bit Definitions

Physical Address 0x4110_001c — MMC_BLKLEN — MMC/SD/SDIO



Bits	Access	Name	Description
31:12	—	—	reserved
11:0	R/W	BLK_LEN	Number of Bytes in a Block of Data

15.9.9 MMC Number of Blocks Register (MMC_NUMBLK)

In block mode, this register specifies the number of blocks.

For single block data transfers, MMC_NUMBLK must be set.

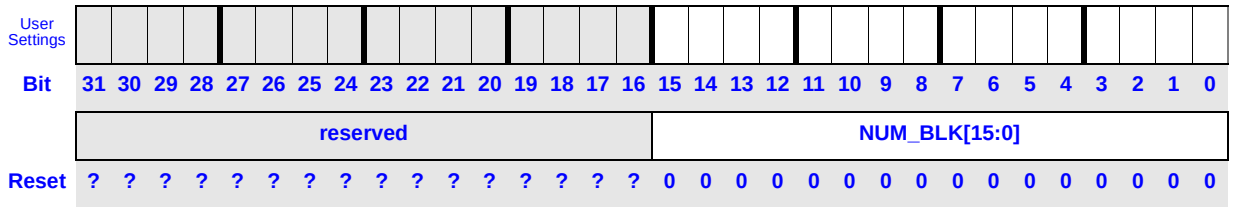
For multiple block data transfers, MMC_NUMBLK must be set to a value greater than 0.

For stream data transfers, MMC_NUMBLK can be any value. The value is ignored by the MMC/SD/SDIO controller.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-16. MMC_NUMBLK Bit Definitions

Physical Address 0x4110_0020 — MMC_NUMBLK — MMC/SD/SDIO



Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R/W	NUM_BLK	Number of Blocks for a Multiple Block Data Transfer

15.9.10 MMC Buffer Partly Full Register (MMC_PRTBUF)

The MMC_PRTBUF register is used when MMC_TXFIFO is partially written. The FIFOs swap when either FIFO is full (32 bytes) or the MMC_PRTBUF register is set to a 1. This register is cleared by the MMC/SD/SDIO controller after the FIFOs have swapped.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-17. MMC_PRTBUF Bit Definitions

Physical Address 0x4110_0024	MMC_PRTBUF	MMC/SD/SDIO

User Settings																															
Bit	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																															PRT_BUF
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0

Bits	Access	Name	Description
31:1	—	—	reserved
0	R/W	PRT_BUF	Buffer Partially Full 0 = Buffer is not partially full. 1 = Buffer is partially full and must be swapped to the other transmit buffer.

15.9.11 MMC Interrupt Mask Register (MMC_I_MASK)

The MMC_I_MASK register masks off the various interrupts. To mask an interrupt, set its bit.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-18. MMC_I_MASK Bit Definitions (Sheet 1 of 2)

Physical Address 0x4110_0028	MMC_I_MASK	MMC/SD/SDIO

User Settings																															
Bit	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																			SDIO_SUSPEND_ACK	SDIO_INT	RD_STALLED	RES_ERR	DAT_ERR	TINT	TXFIFO_WR_REQ	RXFIFO_RD_REQ	CLK_IS_OFF	STOP_CMD	END_CMD_RES	PRG_DONE	DATA_TRAN_DONE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Access	Name	Description
31:13	—	—	reserved

Table 15-18. MMC_I_MASK Bit Definitions (Sheet 2 of 2)

Physical Address 0x4110_0028																MMC_I_MASK														MMC/SD/SDIO													
User Settings																																										
Bit	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																			SDIO_SUSPEND_ACK	SDIO_INT	RD_STALLED	RES_ERR	DAT_ERR	TINT	TXFIFO_WR_REQ	RXFIFO_RD_REQ	CLK_IS_OFF	STOP_CMD	END_CMD_RES	PRG_DONE	DATA_TRAN_DONE										
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1	1	1	1	1	1	1	1	1	1	1	1										

Bits	Access	Name	Description
12	R/W	SDIO_ SUSPEND_ ACK	0 = Enabled 1 = Masked
11	R/W	SDIO_INT	0 = Enabled 1 = Masked
10	R/W	RD_STALLED	0 = Enabled 1 = Masked
9	R/W	RES_ERR	0 = Enabled 1 = Masked
8	R/W	DAT_ERR	0 = Enabled 1 = Masked
7	R/W	TINT	0 = Enabled 1 = Masked
6	R/W	TXFIFO_WR _REQ	0 = Enabled 1 = Masked
5	R/W	RXFIFO_RD _REQ	0 = Enabled 1 = Masked
4	R/W	CLK_IS_ OFF	0 = Enabled 1 = Masked
3	R/W	STOP_CMD	0 = Enabled 1 = Masked
2	R/W	END_CMD_ RES	0 = Enabled 1 = Masked
1	R/W	PRG_DONE	0 = Enabled 1 = Masked
0	R/W	DATA_TRAN _DONE	0 = Enabled 1 = Masked

15.9.12 MMC Interrupt Request Register (MMC_I_REG)

The MMC_I_REG register shows the currently requested interrupt. The FIFO request interrupts, TXFIFO_WR_REQ, and RXFIFO_RD_REQ are masked off with the MMC_DMA_EN bit in the MMC_CMDAT register. The software is responsible for monitoring these bits in programmed I/O mode. The bits are cleared as described in Table 15-19.

If RES_ERR or DAT_ERR occurs, the type of error is in the MMC_STAT register.

This is a read-only register. Ignore reads from reserved bits.

Table 15-19. MMC_I_REG Bit Definitions (Sheet 1 of 3)

Physical Address 0x4110_002C — **MMC_I_REG** — **MMC/SD/SDIO**

Bit	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
User Settings																																
	reserved																			SDIO_SUSPEND_ACK	SDIO_INT	RD_STALLED	RES_ERR	DAT_ERR	TINT	TXFIFO_WR_REQ	RXFIFO_RD_REQ	CLK_IS_OFF	STOP_CMD	END_CMD_RES	PRG_DONE	DATA_TRAN_DONE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:13	—	—	reserved
12	R	SDIO_ SUSPEND_ ACK	1 = SDIO data transfer has been suspended by SDIO card
11	R	SDIO_INT	1 = SDIO interrupt occurred
10	R	RD_STALLED	1 = Card has stalled read, in response to RD_WAIT
9	R	RES_ERR	1 = An error occurred on the response
8	R	DAT_ERR	1 = A data error occurred during data transmission
7	R	TINT	Transmit Interrupt 0 = Receive data transfer did not have a time-out. 1 = Receive data transfer has time-out.
6	R	TXFIFO_WR _REQ	Transmit FIFO Write Request 0 = No Request for data write to MMC_TXFIFO FIFO. 1 = Request for data write to MMC_TXFIFO FIFO. Up to a maximum of 32 bytes

Table 15-19. MMC_I_REG Bit Definitions (Sheet 2 of 3)

Physical Address 0x4110_002C — **MMC_I_REG** — **MMC/SD/SDIO**

Bit	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
User Settings																																
	reserved																			SDIO_SUSPEND_ACK	SDIO_INT	RD_STALLED	RES_ERR	DAT_ERR	TINT	TXFIFO_WR_REQ	RXFIFO_RD_REQ	CLK_IS_OFF	STOP_CMD	END_CMD_RES	PRG_DONE	DATA_TRAN_DONE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
5	R	RXFIFO_RD_REQ	Receive FIFO Read Request 0 = No Request for data read from MMC_RXFIFO FIFO. 1 = Request for data read from MMC_RXFIFO FIFO. When the RXFIFO is full, or all of the last bytes of a transfer are in the RXFIFO, RXFIFO_RD_REQ = 1. Then, after the first read from the RXFIFO, RXFIFO_RD_REQ = 0. Software should always know the number of bytes that are to be received. Software is responsible for keeping track of the number of bytes in the RXFIFO. Examples: If the total number of bytes to be received is 64, then software should wait for the first RXFIFO_RD_REQ = 1 and then read 32 bytes. Then software should wait for the second RXFIFO_RD_REQ = 1 and read the last 32 bytes. If the total number of bytes to be received is 8, software should wait for RXFIFO_RD_REQ = 1 and then read 8 bytes. If the total number of bytes to be received is 36, then software should wait for the first RXFIFO_RD_REQ = 1 and then read 32 bytes. Then software should wait for the second RXFIFO_RD_REQ = 1 and read the last 4 bytes.
4	R	CLK_IS_OFF	Clock is Off 0 = MMCLK has not been turned off. 1 = MMCLK has been turned off, due to stop bit in STRP_CLK register. Cleared by the MMC_STAT[CLK_EN] bit when the clock is started.
3	R	STOP_CMD	For stream mode writes. 0 = MMC is not ready for the stop transmission command. 1 = MMC is ready for the stop transmission command. Cleared when CMD12 is loaded in the MMC_CMD register and the clock is started.

Table 15-19. MMC_I_REG Bit Definitions (Sheet 3 of 3)

Physical Address 0x4110_002C — MMC_I_REG — MMC/SD/SDIO

Bit	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
User Settings																																
	reserved																			SDIO_SUSPEND_ACK	SDIO_INT	RD_STALLED	RES_ERR	DAT_ERR	TINT	TXFIFO_WR_REQ	RXFIFO_RD_REQ	CLK_IS_OFF	STOP_CMD	END_CMD_RES	PRG_DONE	DATA_TRAN_DONE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	R	END_CMD_RES	End Command Response 0 = MMC has not received the response. 1 = MMC has received the response or a response time-out has occurred. Cleared by the MMC_STAT[END_CMD_RES] bit.
1	R	PRG_DONE	Programming Done 0 = Card has not finished programming and is busy. 1 = Card has finished programming and is no longer busy. Cleared by the MMC_STAT[PRG_DONE] bit.
0	R	DATA_TRAN_DONE	Data Transfer Done 0 = Data transfer is not complete. 1 = Data transfer has completed or a read data time-out has occurred. Cleared by the MMC_STAT[DATA_TRAN_DONE] bit.

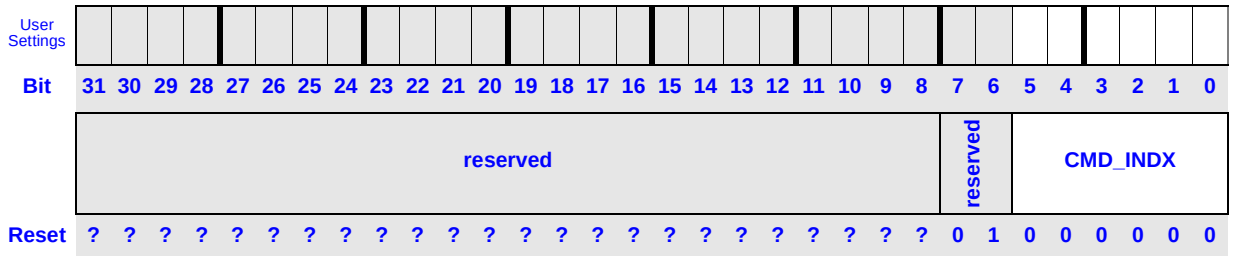
15.9.13 MMC Command Register (MMC_CMD)

The MMC_CMD register specifies the command number.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-20. MMC_CMD Bit Definitions

Physical Address 0x4110_0030	MMC_CMD	MMC/SD/SDIO



Bits	Access	Name	Description
31:8	—	—	reserved
7	R	—	reserved, read-only, always 0. Start bit for command sequence and cannot be changed.
6	R	—	reserved, read-only, always 1. Transmission bit in command sequence and cannot be changed.
5:0	R/W	CMD_INDX	Command Index Refer to *The MultiMediaCard System Specification, SD Memory Card Specification, and SDIO Card Specification* for additional details.

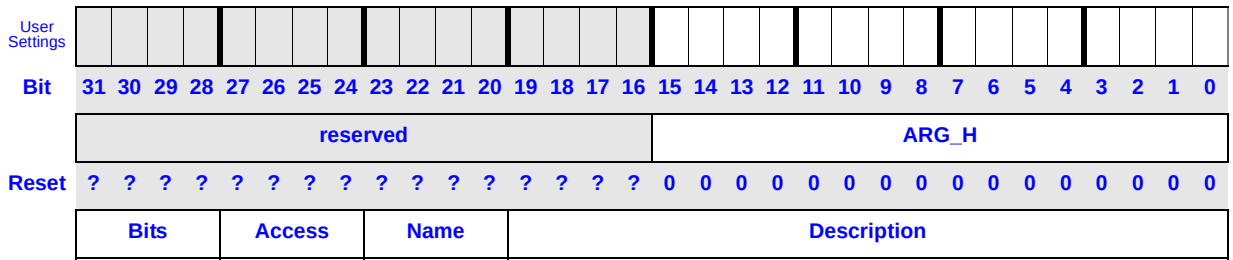
15.9.14 MMC Argument High Register (MMC_ARGH)

The MMC_ARGH register specifies the upper 16 bits of the argument for the current command.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-21. MMC_ARGH Bit Definitions

Physical Address 0x4110_0034	MMC_ARGH	MMC/SD/SDIO



Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R/W	ARG_H	Upper 16 Bits of Command Argument

15.9.15 MMC Argument Low Register (MMC_ARGL)

The MMC_ARGL register specifies the lower 16 bits of the argument in the current command.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-22. MMC_ARGL Bit Definitions

Physical Address 0x4110_0038	MMC_ARGL	MMC/SD/SDIO

Bit	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																ARG_L															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R/W	ARG_L	Lower 16 Bits of Command Argument

15.9.16 MMC Response FIFO (Read-Only) (MMC_RES)

The MMC_RES FIFO contains the response after a command. It is 16 bits wide by eight entries.The RES FIFO does not contain the 7-bit CRC for the response. The status for CRC checking and response time-out status is in the status register, MMC_STAT.

This is a read-only register. Ignore reads from reserved bits.

Table 15-23. MMC_RES Bit Definitions

Physical Address 0x4110_003C	MMC_RES FIFO	MMC/SD/SDIO

Bit	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																Data															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R	Data	Two Bytes of Response Data

15.9.17 MMC Receive FIFO (Read-Only) (MMC_RXFIFO)

The MMC_RXFIFO consists of two FIFOs, where each FIFO is eight bits wide by 32 entries deep. This FIFO holds the data read from a card. It is a read-only FIFO to the software and is read on 8-bit boundaries. The eight bits of data are read on a 32-bit peripheral bus and occupy the least significant byte lane (7:0).

MMC_RXFIFO is readable on 1-, 2-, or 4-byte boundaries. For example, LDB reads 1 byte; LDH reads 2 bytes; and LD reads 4 bytes.

This is a read-only register. Ignore reads from reserved bits.

Table 15-24. MMC_RXFIFO Bit Definitions

Physical Address 0x4110_0040	MMC_RXFIFO	MMC/SD/SDIO

Bit	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																								Data							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R	Data	One, Two, or Four Bytes of Received Data

15.9.18 MMC Transmit FIFO (MMC_TXFIFO)

The MMC_TXFIFO consists of two FIFOs, where each FIFO is eight bits wide by 32 entries deep. This FIFO holds the data to be written to a card. It is a write-only FIFO to the software, and is written on boundaries eight bits wide. The eight bits of data are written on a 32-bit peripheral bus and occupy the least significant byte lane (7:0).

MMC_TXFIFO is writable on 1-, 2-, or 4-byte boundaries. For example, STRB writes 1 byte; STRH writes 2 bytes; and STR writes 4 bytes.

This is a write-only register. Write 0b0 to reserved bits.

Table 15-25. MMC_TXFIFO Bit Definitions

Physical Address 0x4110_0044	MMC_TXFIFO	MMC/SD/SDIO

User Settings																																
Bit	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																								Data							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	W	Data	One, Two, or Four Bytes of Transmitted Data

15.9.19 MMC RD_WAIT Register (MMC_RDWAIT)

This register is used send a RD_WAIT to the card. The RD_WAIT operation is only supported for SDIO cards. It is not supported for SPI transfers.

The RD_WAIT_STRT bit is cleared by the controller after the read data transfer has restarted.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-26. MMC_RDWAIT Bit Definitions

Physical Address 0x4110_0048	MMC_RDWAIT	MMC/SD/SDIO

Bit	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
User Settings																																
	reserved																														RD_WAIT_START	RD_WAIT_EN
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
31:2	—	—	reserved
1	R/W	RD_WAIT_START	1 = Restart the read data transfer.
0	R/W	RD_WAIT_EN	1 = RD_WAIT is enabled. SDIO mode only.

15.9.20 MMC Blocks Remaining Register (MMC_BLKS_REM)

This register contains the number of blocks that were not transferred due to:

- SDIO suspension
- read data time-out
- SPI read data error token
- SPI write data error token
- SPI write CRC error
- CMD12 or CMD52 abort to stop data transmission

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 15-27. MMC_BLKS_REM Bit Definitions

Physical Address 0x4110_004C	MMC_BLKS_REM	MMC/SD/SDIO

Bit	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
User Settings																																
	reserved																BLKS_REM[15:0]															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R/W	BLKS_REM	Number of Data Blocks Not Transferred

15.10 Register Summary

The MMC/SD/SDIO controller is controlled by a set of registers that software configures before every command sequence on the MMC/SD/SDIO bus.

Table 15-28 lists the addresses, names, and descriptions of the MMC/SD/SDIO controller registers.

Table 15-28. MMC Controller Register Summary

Address	Name	Description	Page
0x4110_0000	MMC_STRPCL	MMC Clock Start/Stop register	15-29
0x4110_0004	MMC_STAT	MMC Status register	15-29
0x4110_0008	MMC_CLKRT	MMC Clock Rate register	15-31
0x4110_000C	MMC_SPI	MMC SPI Mode register	15-31
0x4110_0010	MMC_CMDAT	MMC Command/Data register	15-32
0x4110_0014	MMC_RESTO	MMC Response Time-Out register	15-34
0x4110_0018	MMC_RDTO	MMC Read Time-Out register	15-34
0x4110_001C	MMC_BLKLEN	MMC Block Length register	15-35
0x4110_0020	MMC_NUMBLK	MMC Number of Blocks register	15-35
0x4110_0024	MMC_PRTBUF	MMC Buffer Partly Full register	15-36
0x4110_0028	MMC_I_MASK	MMC Interrupt Mask register	15-36
0x4110_002C	MMC_I_REG	MMC Interrupt Request register	15-38
0x4110_0030	MMC_CMD	MMC Command register	15-41
0x4110_0034	MMC_ARGH	MMC Argument High register	15-41
0x4110_0038	MMC_ARGL	MMC Argument Low register	15-42
0x4110_003C	MMC_RES	MMC Response FIFO	15-42
0x4110_0040	MMC_RXFIFO	MMC Receive FIFO	15-42
0x4110_0044	MMC_TXFIFO	MMC Transmit FIFO	15-43
0x4110_0048	MMC_RDWAIT	MMC RD_WAIT register	15-43
0x4110_004C	MMC_BLKS_REM	MMC Blocks Remaining register	15-44

Mobile Scalable Link (MSL) Interface 16

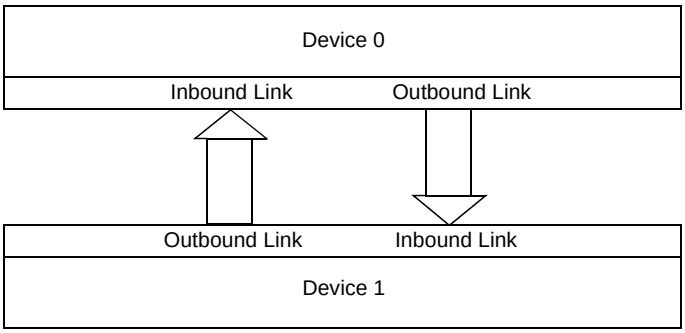
This chapter describes the mobile scalable link (MSL), a low-power, scalable, high-speed, narrow, chip-to-chip physical interface for mobile or wireless platforms. The MSL meets the requirements of the Intel® Personal Internet Client Architecture, which describes the framework for rapidly building and deploying wireless devices.

16.1 Overview

The MSL physical interface consists of a pair of unidirectional, high-speed links for connecting two devices. It has multiple logical channels for sending and receiving both packet-based and streaming transfers such as those for multimedia data and voice communications.

The MSL data-link protocol provides reliable data transfer services for peer-to-peer communication between upper layer protocol entities over the physical link. In addition, it provides connection-oriented data services with acknowledged or unacknowledged transfers. The MSL data link is capable of transporting data streams with different quality-of-service requirements (for example, telephony signaling, real-time voice, or video) and can operate over a variety of lower-layer physical media.

Figure 16-1. Mobile Scalable Link Example



16.2 Features

The mobile scalable link has these key features:

- Two independent, high-speed, unidirectional links
- Scalable links with data-channel width options
- Asynchronous clocking from 0 to over 48 MHz per link
- Transfer rate per link up to 192 Mbps at 48 MHz
- Low-power electrical interface: 1.8 V (+20%/-5%), 2.5 V, 3.0 V and 3.3 V +10%/-10%
- Power management protocol and features

- 14 independent logical data channels for managing multiple simultaneous data streams
- Large 64-byte FIFOs for all data channels
- Round-robin FIFO service with independent enables and configuration options
- Single- or multiple-burst transfers
- Support for DMA-, interrupt-, or poll-driven operation

16.3 Signal Descriptions

Table 16-1 summarizes all external signals connected to the interface. For complete information about reset values, see "Pin Mapping and Usage" in the *Intel® PXA270 Processor Electrical, Mechanical, and Thermal Specification* and *Intel® PXA27x Processor Family Electrical, Mechanical, and Thermal Specification* (*Intel® PXA27x Processor Family EMTS*).

Table 16-1. Mobile Scalable Link I/O Signal Descriptions

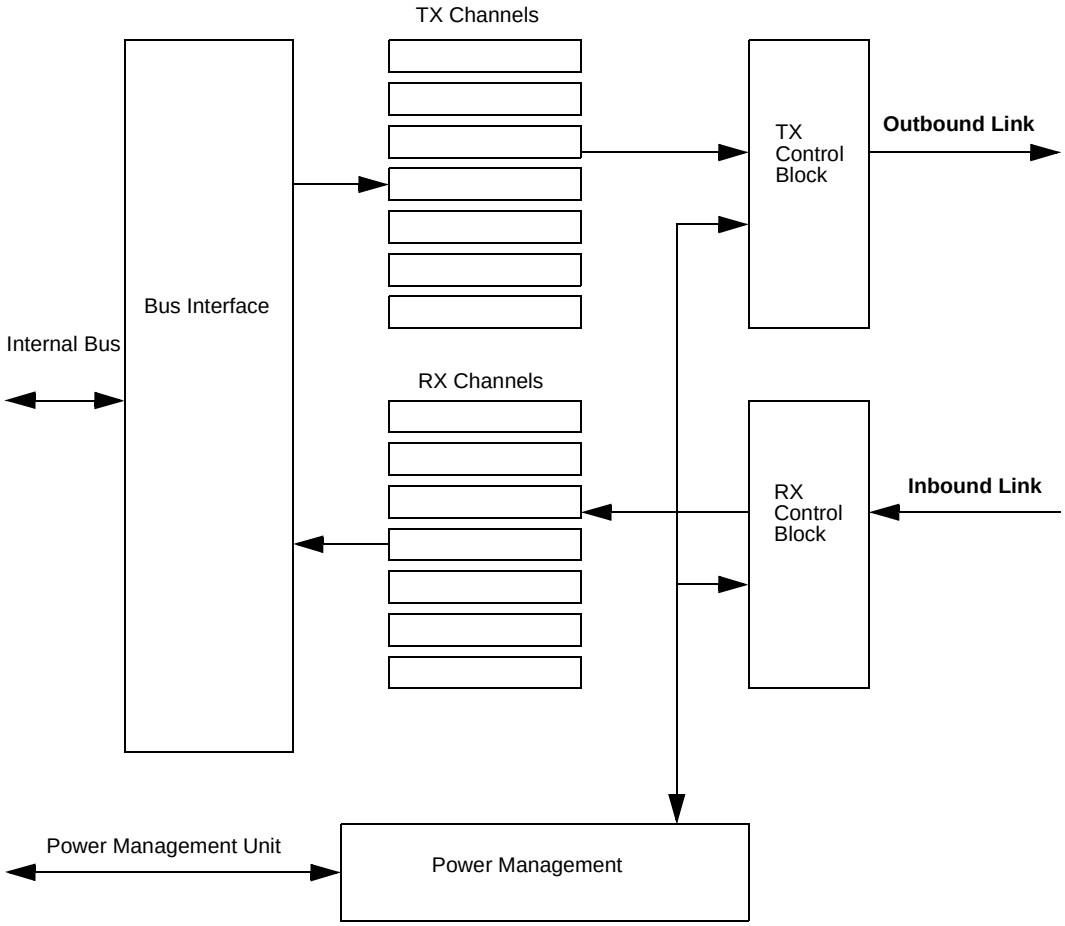
Name	Type	Description
BB_OB_DAT<3:0>	Output	Outbound data bits. These pins transmit data between the source and the target.
BB_OB_CLK	Output	Outbound clock strobe. This strobe indicates valid data and control signals when it toggles from low to high. **NOTE:** The BB_IB_CLK frequency cannot exceed 48 MHz when the peripheral clock is running at 26 MHz, or 24 MHz when the peripheral clock is running at 13 MHz.
BB_OB_STB	Output	Outbound signal qualifier. This signal indicates that a channel identifier is on the data pins when it is asserted and that a data nibble is on the data pins when it is deasserted.
BB_OB_WAIT	Input	Wait indicator for outbound link. This signal is sent from the target to the source and indicates whether a channel is available to accept more data. An attempted data transfer with BB_xx_WAIT asserted cancels the transfer.
BB_IB_DAT<3:0>	Input	Inbound data bits.
BB_IB_CLK	Input	Inbound clock strobe.
BB_IB_STB	Input	Inbound signal qualifier.
BB_IB_WAIT	Output	Wait indicator for inbound link.

16.4 Operation

This section describes the functions of the major modules in the mobile scalable link (MSL). The interface consists of two independent unidirectional links: one outbound link and one inbound link. Each physical link supports seven logical data channels. These channels are multiplexed over their respective physical link but are independent and do not interact in any way.

Figure 16-2 shows the block diagram for the interface.

Figure 16-2. Mobile Scalable Link Block Diagram



16.4.1 Transmit Operation

Each of the seven identical channels in the outbound link contains a 64-byte FIFO buffer, into which data can be written either directly by the processor or by DMA (see Figure 16-14). Data written to a FIFO buffer is transmitted through its associated channel by the shared physical outbound link. The FIFO buffers can be accessed a byte, half-word, or word at a time. Word or half-word accesses write the bytes in little-endian order.

The MSL Interface Width register (BBITFC) determines how many bits are transmitted over the outbound link at a time: one, two, or four bits wide (see Table 16-13). Bytes are sent over the link in little-endian order. For example, if the link is set to four-bit mode, the least significant nibble is transmitted first.

Each channel has an associated MSL Channel Configuration register (BBCFGx) that controls activity across the channel. These registers contain trigger thresholds for each FIFO. When the amount of data in a FIFO drops below this trigger threshold, service for the FIFO is initiated according to the type of service specified in the MSL Channel Configuration registers, BBCFGx[TxService] (see Section 16.5.2): interrupt, DMA, or none. DMA service supports transfers of up to 32 bytes.

Another configuration bit, BBCFGx[TxEnable], enables or disables a channel, regardless of whether data remains in the FIFO buffer. A disabled transmit channel generates interrupts and allows DMA access to the FIFO, but no data is transmitted from it.

Each channel also has an associated MSL Channel Status register, BBSTATx (see Section 16.5.3), which includes the following information about the channel:

- TxWait—Channel is in wait state
- TxEmpty—FIFO buffer is empty
- TxEmpty—FIFO buffer is full
- TxFullness—Number of data bytes remaining in FIFO buffer

The transmit block determines by arbitration the channel from which data is to be transmitted. Transmit channels are selected in channel-number order (1 to 7, then back to 1), skipping channels that do not contain data or are in a wait state (see Section 16.4.3 for more on wait states). A new channel is selected for one of the following reasons:

- The maximum number of bytes for the current channel has been sent.
- The current channel's transmit FIFO buffer is empty.
- A stop message was received for the current channel, putting it in a wait state.

The rate at which data is transmitted from each channel over the physical link depends on two factors: arbitration between channels for the link and the occurrence of full receive FIFOs in the target.

16.4.2 Receive Operation

Each of the seven identical channels in the inbound link contains a 64-byte FIFO buffer, from which data can be read either directly by the processor or by DMA (see Figure 16-14). Data received over the link goes into the FIFO buffer for the channel from which it was received. The FIFO buffers can be accessed a byte, a half-word, or a word at a time. Word or half-word accesses read the bytes in little-endian order.

The MSL Interface Width register (BBITFC) determines how many bits are received over the inbound link at a time: one, two, or four bits wide (see Table 16-13). Bytes received over the link are assumed to be in little-endian order. For example, if the link is set to four-bit mode, the least-significant nibble is received first.

Each channel has an associated MSL Channel Configuration register that controls activity across the channel. These registers contain trigger thresholds for each FIFO buffer. When the amount of data in a FIFO exceeds this trigger threshold, service for the FIFO is initiated according to the type of service specified in the MSL Channel Configuration registers, BBCFGx[RxService] (see Section 16.5.2): interrupt, DMA, or none. DMA service supports transfers of up to 32 bytes.

Another configuration bit, BBCFGx[RxEnable], enables or disables a channel, regardless of whether data remains in the FIFO buffer. A disabled receive channel generates interrupts and allows DMA access to the FIFO, but no data from the inbound link goes into it.

Each channel also has an associated MSL Channel Status register, BBSTATx (see Section 16.5.3), which includes the following information about the channel:

- RxWait—Channel is in wait state
- RxEmpty—FIFO buffer is empty
- RxEmpty—FIFO buffer is full
- RxFullness—Number of data bytes remaining in FIFO buffer

16.4.3 Channel Flow Control

The channel through which data is sent must first be activated to transfer its data over the interface. A channel is activated by asserting the BB_xx_STB signal and putting the channel number on the data pins before the next rising-edge clock transition. Data is then transferred from the data pins on each of the following rising-edge clock transitions.

Table 16-2 shows the three different interface widths that are supported. Data on any unused data pins is ignored by the inbound link and held to zero by the outbound link. A whole number of bytes must always be transferred. Transfer cycles must be in multiples, as shown in Table 16-2.

Table 16-2. Supported MSL Interface Widths

Mode	Width in Bits	BB_xx_DAT Pins	Transfer Cycle Whole-Byte Multiple
Serial	1	<0>	8
Two-bit	2	<1:0>	4
Nibble	4	<3:0>	2

When the transmission of a message through a channel is complete, the link must change the active channel to channel 0 (the EOM channel). Switching to the EOM channel signifies the end of a message (EOM) and thus initiates service at the target for the corresponding receive FIFO. The type of service initiated depends on the MSL Channel Configuration register. Figure 16-3, Figure 16-4, and Figure 16-5 show examples of a basic data transfer of 0x3D7B over channel 3 in nibble, two-bit, and serial modes, respectively

Figure 16-3. Example MSL Waveform for Basic Transmission in Nibble Mode

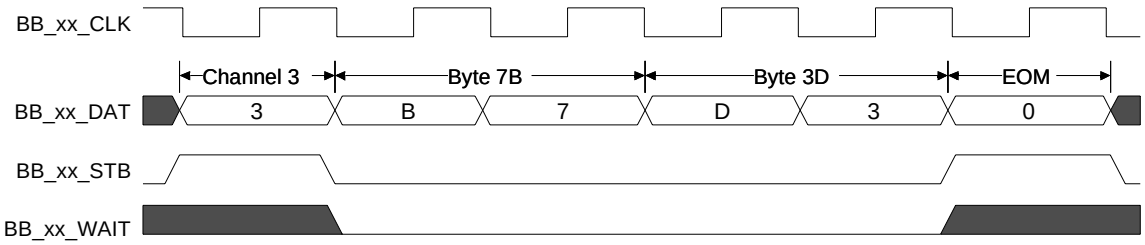


Figure 16-4. Example MSL Waveform for Basic Transmission in Two-Bit Mode

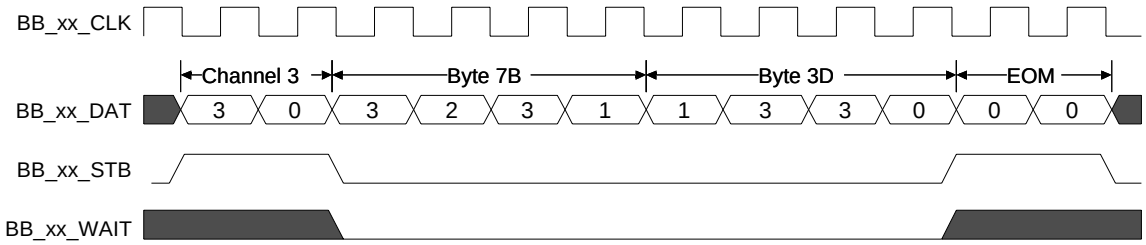
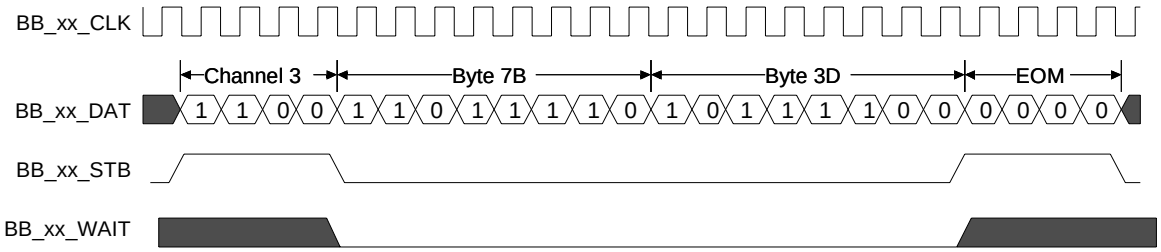


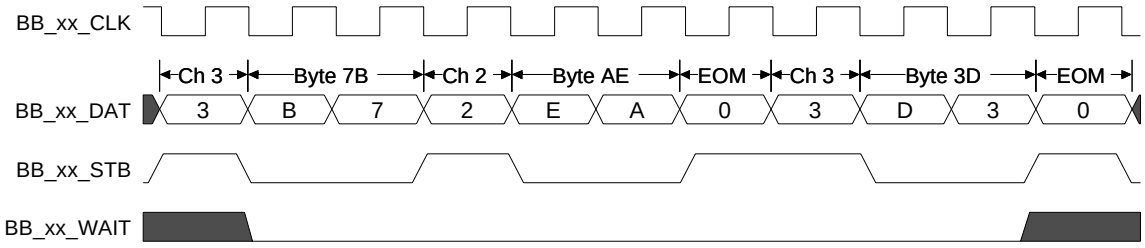
Figure 16-5. Example MSL Waveform for Basic Transmission in Serial Mode



Activating a new channel requires the reassertion of the BB_xx_STB signal with the value of the new channel on the data pins. New channels can be activated immediately following the activation of the EOM channel when no transfer is occurring, in the middle of a transfer, or just after the current transfer has finished. Figure 16-6 shows an example of data transfers in nibble mode for two different channels, 3 and 2:

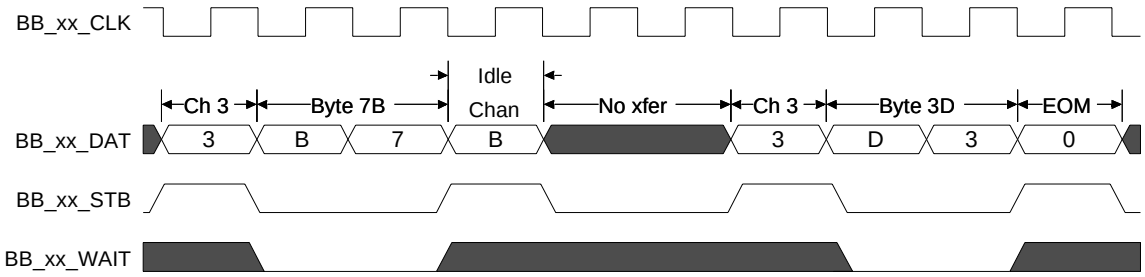
1. Channel 3 sends one byte, followed by a switch to channel 2, which sends one byte as well.
2. After channel 2 sends one byte, the transfer is complete, so the EOM channel is activated.
3. Immediately following the end of the message from channel 2, channel 3 sends one more byte and ends its message.

Figure 16-6. Example MSL Waveform for New Channel Selection



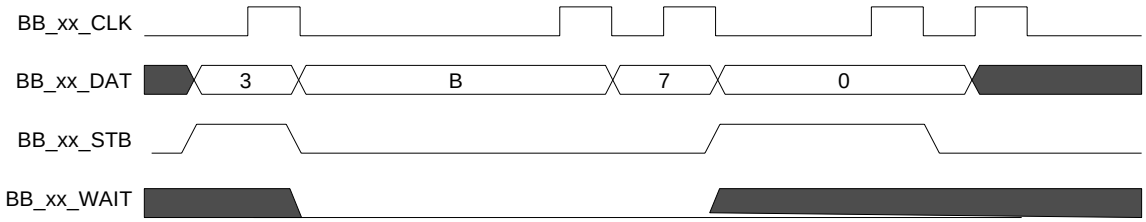
If the source must stop transmitting a message without activating a new channel, it does so by activating channel 11 (the idle channel). This stops the data transmission without initiating service until a data channel is activated. Figure 16-7 shows an example waveform in nibble mode where a two-byte transmission through channel 3 is interrupted by an activation of the idle channel.

Figure 16-7. Example MSL Waveform for Idle Channel



The BB_xx_CLK signal is not a clock in the true sense, as it does not need to change regularly. It can be kept low for an indefinite time if no data is to be transmitted. Figure 16-8 shows an example waveform where the clock does not toggle regularly.

Figure 16-8. Example MSL Waveform for Clock

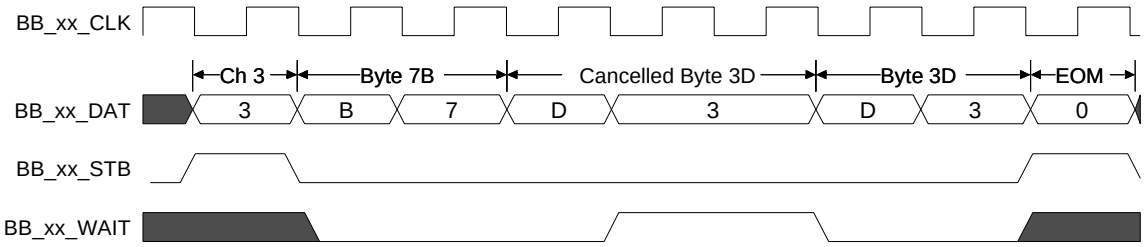


While receiving data, the target channel's receive FIFO can become full, which prevents the channel from accepting any new data. The WAIT pin notifies the source of this event.

In direct flow control, the target asserts its BB_xx_WAIT signal when the active channel is disabled, invalid, or if its receive FIFO is full. Additionally, the BB_xx_WAIT signal is asserted after reset and while the link is idle (no data or messages being transmitted). The source samples the BB_xx_WAIT signal on the rising edge of the BB_xx_CLK. If the BB_xx_WAIT signal is

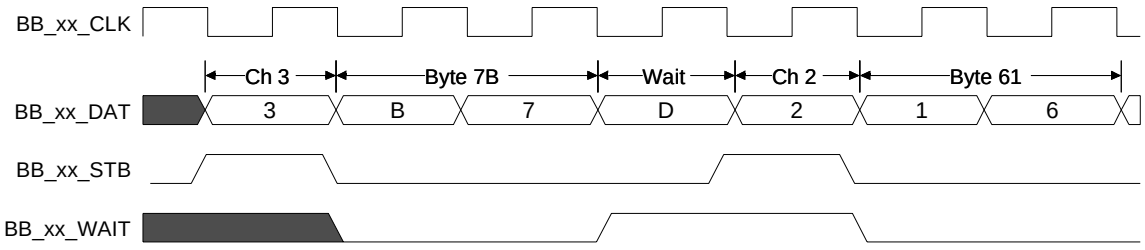
sampled high, the active channel is in a wait state, and data transfers are ignored. Once the BB_xx_WAIT signal is sampled low, the wait state is over, and data transmission can resume as normal. Note that the wait signal is recognized in the middle of a byte transfer. For example, if the wait signal is asserted during the transfer of the second nibble of a byte, that byte does not count as a valid transfer. No data is transferred until the BB_xx_WAIT signal is deasserted. Figure 16-9 illustrates this example in nibble mode where 0x3D must be retransmitted.

Figure 16-9. Example MSL Waveform with Wait State



Instead of waiting for the BB_xx_WAIT signal to deassert, the source can change to a new channel and transmit data for that channel. Note that this requires the assertion of the BB_xx_STB signal while the BB_xx_WAIT signal is still asserted. Although changing the active channel allows transmission through one channel while the other is in a wait state, the target can assert the BB_xx_WAIT signal as soon as the new channel is activated, thus placing that channel in wait state as well. There is no limit to the number of times a new channel can be activated while other channels are in wait states. Figure 16-10 shows a sample waveform in nibble mode for a new channel activation after a BB_xx_WAIT signal is asserted. In Figure 16-10, 0x61 is transmitted through channel 2 after channel 3 is placed in a wait state.

Figure 16-10. Example MSL Waveform for New Channel Selection After Wait State



The source does not know when a channel exits a wait state until it activates the channel and the BB_xx_WAIT signal is not asserted. Therefore, if a channel does not remain active during the entire wait state, it must be polled to determine when it has exited the wait state. This polling consists of a periodic activation of the channel in wait state. Since continuous polling can be inefficient and a significant power draw, it is limited. Once a channel is in wait state, the transmit control block does not activate that channel again until a user-specified time has elapsed. This value is programmed in the MSL Wait Count register (BBWAIT; see Section 16.5.7). Waiting before retries avoids continual activity on the interface while the channel is unavailable.

16.4.4 Power Management

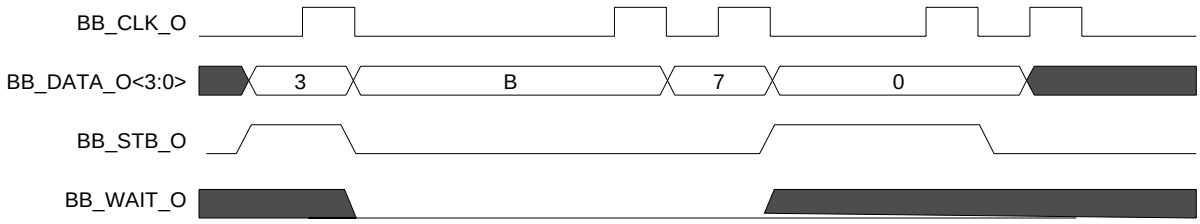
Several power management events can occur across the interface. These events include a device's going to sleep, waking up, and requesting the other device to wake up. It is expected that software provide the overall mechanism for inter-device power management policy.

- The specification also requires the target device to wake up upon detecting a low-to-high transition of the input CLK.
- The Intel® MSL specification provides the wake-up channel to indicate to a target device that it must exit any low-power state and be ready to accept commands.

16.4.4.1 Power Managing Clocks

The clocks do not need to change regularly. They can be kept low for an indefinite time if no channels or data need to be transmitted. Figure 16-11 shows an example waveform where the clock does not toggle regularly.

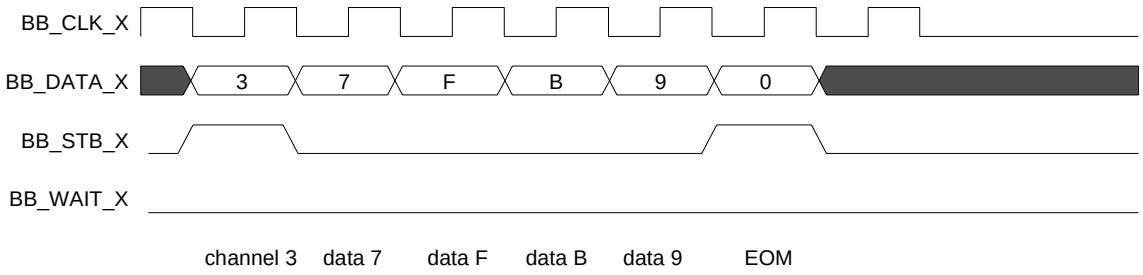
Figure 16-11. Power Managing Clocks Waveform



For low-power link modes, it is expected that devices will use CLK_X as a strobe, negating it when there is no activity on the link or when the link is idle. To save power, devices can detect idle conditions on the link and power off the clock. A link is idle when all channels have either no data to transmit, or have data to transmit, but the individual channels are in a wait state. While a link is idle, it is legal to either continue to toggle the clock or to stop it until the link is no longer idle. It is recommended that implementations continue toggling the clock for a programmable number of transmit clock cycles. There should be a mode in which the clocks immediately stop upon an idle condition (for example, EOM channel, idle channel), stop after a programmable number of transmit clocks, or never stop.

Figure 16-12 shows an example waveform in which the link becomes idle and the clock is stopped one cycle after the idle time begins. Idle time begins on the first cycle WAIT_X is sampled asserted, or when the active channel is changed to the EOM or Idle channel. In this example, the clock stop time is set to one, therefore the clock turns off one cycle after the link becomes idle.

Figure 16-12. Example Waveform of Clock Stopping One Cycle After EOM



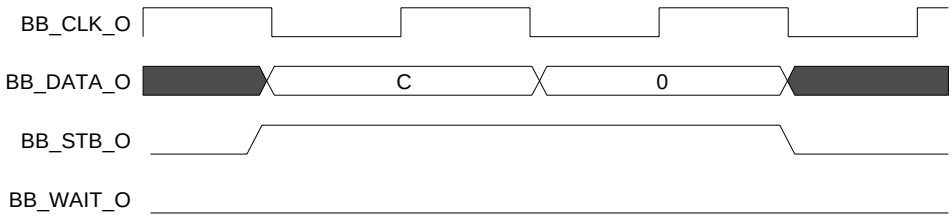
16.4.4.2 Wake-Up Channel (Channel 12)

It is expected that software will manage the power management policy for the links. The hardware defines only the wake-up channel for causing the target to exit a low-power mode. A source generates the wake-up channel (Channel 12) using a software initiated event. Devices must provide software with the capability of initiating a wake-up channel on the link. It is expected that target devices decode the wake-up channel, exit any low-power link state, and be ready to accept incoming commands, messages, or data.

- Devices must generate an interrupt upon the detection of the wake-up channel so that software can acknowledge/respond to this channel.
- Devices can keep the clocks running while they are in low-power states. For these devices, they must exit the low-power state after detecting the wake-up channel.
- A device in a low power state may not be able to decode the wake-up channel. For devices in this state, any active clock (low-to-high transition) should cause the device to wake up (for example, by the generation of an internal interrupt). Data-link software should treat the resulting event as a wake-up indication. It is expected that the source's data-link software ensure that the target device has exited any low power state, and only then, should send data or requests to the target.

Whenever channel 12 is activated across the link, the target wakes up. Figure 16-13 shows the waveform for this wake-up channel in quad bit-mode. No data is transmitted. The link simply switches to the wake-up channel.

Figure 16-13. Wake-Up Channel Waveform



16.4.5 Channel Allocation

Sixteen logical channels are defined for each link, as listed in Table 16-3. Channel numbers are sent across the BB_xx_DAT pins. A channel is valid when the BB_xx_STB pin is asserted and on a rising clock edge.

Table 16-3. Summary of MSL Channel Allocation

Channel Number	Description
0	End of message (EOM)
1	Data 1
2	Data 2
3	Data 3
4	Data 4
5	Data 5
6	Data 6
7	Data 7
8	reserved
9	reserved
10	reserved
11	Idle
12	Wake-up

16.4.6 DMA Interaction

DMA can be used to send and receive data to and from the FIFOs without any CPU interaction. When using DMA to write data to the transmit FIFOs, the end of a DMA descriptor chain implies the end of a message. Conversely, when using DMA to read data from a receive FIFO, reading the last byte of a message from the receive FIFO can cause the descriptor chain to end prematurely if the descriptor chain length is less than the message length. Therefore, if the length of an incoming message is unknown, DMA can be programmed to receive a message as large as the maximum possible incoming message and to stop on an end-of-message indicator. If the received message is shorter than the DMA chain length, the DMA descriptor chain ends after the last byte of the message.

The following sequences describe an example where DMA sends and receives a message. This example assumes that a 40-byte message is to be sent from the multimedia chip to the baseband chip over channel 1. The receive and transmit service trigger thresholds (BBCFGx[RxThreshLevel and BBCFGx[TxThreshLevel]—see Section 16.5.2) are set to 16 and 32 bytes, respectively. The baseband chip does not know the size of the incoming message. DMA burst size is 32 bytes.

The following steps set up the DMA interaction on both chips:

1. On the baseband chip, program DMA to read N bytes from FIFO1, where N is large enough to fit the incoming packet. Program DMA to stop on end-of-message (EOM).
2. On the multimedia chip, program DMA to write 40 bytes to FIFO1.

3. On the baseband chip, program the channel 1 channel configuration register to use DMA. Enable the channel and set the service trigger threshold to 16.
4. On the multimedia chip, program the channel 1 channel configuration register to use DMA. Enable the channel, and set the service trigger threshold to 32.

The following events occur in the multimedia chip:

1. The transmit FIFO requests DMA service because the channel 1 transmit FIFO is empty.
2. DMA writes 32 bytes to the transmit FIFO.
3. Transmission of the channel 1 FIFO data to the baseband chip begins over channel 1.
4. Because the transmit FIFO now contains less than 32 bytes, it requests another DMA service.
5. DMA writes the last eight bytes to the transmit FIFO. Because this is the end of the descriptor, the last byte in the FIFO is marked as the last byte in the message.
6. The multimedia chip continues sending data to the baseband chip until it reaches the last byte in the channel 1 FIFO. After the last byte is transmitted, the multimedia chip changes its outbound link to channel 0 to signify the end of the message.

The following events occur in the baseband processor:

1. The baseband inbound link switches to channel 1 (the baseband's outbound link). It begins receiving data into the channel 1 receive FIFO.
2. After receiving 16 bytes, the receive FIFO requests DMA service.
3. DMA reads 16 bytes from the receive FIFO.
4. After another 16 bytes are read, the receive FIFO requests another DMA service.
5. DMA reads another 16 bytes from the receive FIFO.
6. The last eight bytes are received and the link channel is changed to 0, indicating the end of the packet. This causes the last byte in the receive FIFO to be marked as the EOM.
7. DMA attempts to read another 16 bytes from the receive FIFO. On the eighth read, it sees the EOM. Since DMA is programmed to stop on an EOM, the descriptor chain ends.

16.5 Register Descriptions

The following sections describe the registers used by the MSL interface to manage data reception and transmission.

Note: Do not modify the reserve bit fields. Unpredictable behavior may occur.

Since these registers work closely with DMA memory transfers, familiarity with Chapter 5, "DMA Controller" is recommended.

16.5.1 MSL FIFO Registers (BBFIFOx)

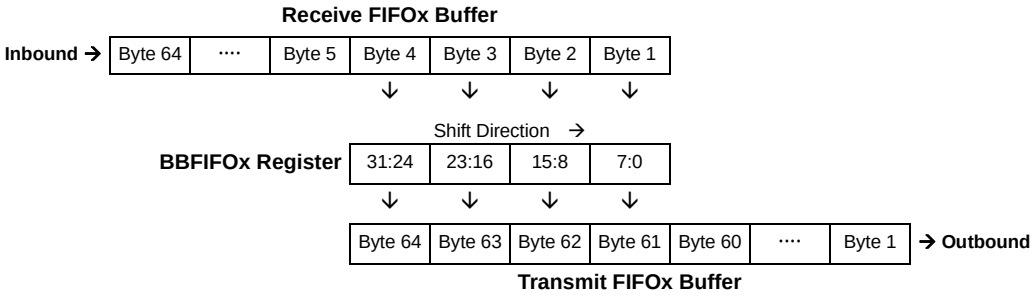
Figure 16-14 illustrates a dual-ported receive/transmit FIFO register and its relationship to the receive and transmit FIFO buffers. There is one such register/buffer set for each of the seven channels. Multiple-byte data order is little endian.

The data accessed by the register depends upon the mode of addressing the register:

- **Reading** from the register pulls new data from the receive FIFO buffer.
- **Writing** to the register pushes new data into the transmit FIFO buffer.

The single set of BBFIFOx registers are used in either receive or transmit mode, and the determination of what the contents are to be interpreted as depends on what mode (receive or transmit) that the MSL interface is operating.

Figure 16-14. MSL FIFO Structure and Organization



16.5.1.1 Receive Operation

Data received over the inbound link is right-shifted into the channel's receive FIFO buffer. One of the following conditions signals the need for service (which is performed by reading the BBFIFOx register):

- The amount of data in the FIFO buffer exceeds the trigger threshold specified in BBCFGx[RxThreshLevel] (see Table 16-5)
- An end-of-message (EOM) is received (see Section 16.4.3)

Reading a BBFIFOx register transfers up to four bytes of FIFO buffer data into memory. Both the register and the receive FIFO buffer are right-shifted by the number of bytes read.

16.5.1.2 Transmit Operation

Data in a channel's transmit FIFO buffer is right-shifted out for transmission over the outbound link. When the amount of data in the FIFO buffer falls below the trigger threshold specified in BBCFGx[TxThreshLevel] (see Table 16-5), the need for service is signaled, which is performed by writing to the BBFIFOx register.

Writing to a BBFIFOx register transfers up to four bytes of data from memory into the channel's transmit FIFO buffer. The register is right-shifted by the number of bytes written.

16.5.1.3 Memory Transfer Modes

Data transfers between memory and a FIFO register take place in one of two ways:

- Direct CPU access after an interrupt or a poll
- DMA transfers

For DMA transfers, FIFO register accesses occur according to the settings in the DMA Command register (see Section 5.5.5, "DMA Command Registers (DCMDx)" on page 5-35). DCMDx[WIDTH] must be set to one word (four bytes).

16.5.1.4 Partial-Word Transfers

Access to the BBFIFOx registers takes place in multiples of bytes: word, half-word, and byte reads and writes. This allows access to partial words in the FIFOs. Partial words read from a receive FIFO are stored in the least-significant bytes of the word transferred to memory.

When accessing partial words, always address a BBFIFOx register at its base address (for example, 0x4140_0004 for channel 1). This is required because the least-significant bytes are right-shifted out of the register after they are accessed. There are three ways, for example, to access three bytes in a FIFO, always using the same BBFIFOx address:

- A half-word followed by a byte
- A byte followed by a half-word
- Three successive bytes

Data read from an empty receive FIFO is undefined, and the FIFO state does not change. Word and half-word reads performed with less than a word or half-word of data in the FIFO transfer the partial word to memory in the least significant bytes of the intended word or half-word. Therefore, always program DMA for the correct descriptor length (see Chapter 5, "DMA Controller") or verify that a FIFO contains data (see Section 16.5.3).

Partial words are also read when the EOM is aligned in the middle of a word. For example, a word read of a receive FIFO containing the last two bytes of a message transfers only those two bytes to memory, even if there is a second message in the FIFO. To detect this situation, monitor the appropriate channel status register (see Section 16.5.3).

Writes to a transmit FIFO without enough room to hold the data are ignored, and the FIFO state does not change. Therefore, always program DMA for the correct descriptor length (see Chapter 5, "DMA Controller") or verify that a FIFO has enough room to hold the data (see Section 16.5.3).

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 16-4. BBFIFO1/2/3/4/5/6/7 Bit Definitions

Physical Address		MSL Interface
0x4140_0004	BBFIFO1	
0x4140_0008	BBFIFO2	
0x4140_000C	BBFIFO3	
0x4140_0010	BBFIFO4	
0x4140_0014	BBFIFO5	
0x4140_0018	BBFIFO6	
0x4140_001C	BBFIFO7	

User Settings																															
Bit	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
	DATA																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:0	R/W	DATA	**Reading** this register transfers DATA from the channel's receive FIFO buffer into memory. **Writing** to this register transfers DATA from memory into the transmit FIFO buffer.

16.5.2 MSL Channel Configuration Registers (BBCFGx)

Each channel has associated with it a channel configuration register, which configures the channel. Table 16-5 shows the bit layout of these registers.

The transmit control block attempts to transmit a programmable amount of data from each transmit channel before moving on to the next one. This amount is set in each channel's BBCFGx[TxBlock] bit field.

When a receive FIFO buffer exceeds the trigger threshold defined in BBCFGx[RxThreshLevel], or when a transmit FIFO drops below the trigger threshold defined in BBCFGx[TxThreshLevel], that FIFO requests service of the type specified in BBCFGx[RxService] and BBCFGx[TxService]. The choices are interrupt, DMA, or no service.

BBCFGx[RxEnable] and BBCFGx[TxEnable] enable and disable each channel's receive and transmit paths independently. A disabled transmit channel transmits no data from its transmit FIFO. A disabled receive channel asserts wait signals in response to all incoming transmissions on that channel (see Section 16.4.3).

The Rx/TxEnable and Rx/TxService fields operate independently of each other. For example, setting a channel's service to "none" disables only the initiation of the channel's FIFO service—its channel status register can still be polled and its FIFO register accessed as usual. Conversely, disabling a channel does not prevent FIFO services from being initiated.

WAIT pin are enabled/disabled with BBCFGx[xxWAITenable]. When BBCFGx[RxWAITenable] is cleared, the inbound link never asserts the wait signal. When BBCFGx[TxWAITenable] is cleared, the outbound link ignores the wait signal when the channel corresponding to the cleared bit is activated.

If DMA reads from a receive FIFO, and **only** if the end of a descriptor chain is reached before EOM is read, an interrupt is generated when BBCFGx[EOCservice] = 01. No service is initiated if:

- the end of a descriptor chain is reached when the last byte of a message is read, or
- EOM is reached before the end of the descriptor chain has been reached.

EOCservice does not depend on the service type. However, it does not make sense to set the EOCservice bits to “interrupt” unless the RxService bits are set to DMA.

Note: Normally, RxEnable and RxWAITenable are updated only once after reset. They can be updated at other times, but **only when the inbound link is idle** (to avoid synchronization problems). No such restriction applies to TxEnable or TxWAITenable.

Note: Ensuring that the inbound link is idle requires the data link protocol between the PXA27x processor and the baseband processor.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 16-5. BBCFG1/2/3/4/5/6/7 Bit Definitions (Sheet 1 of 3)

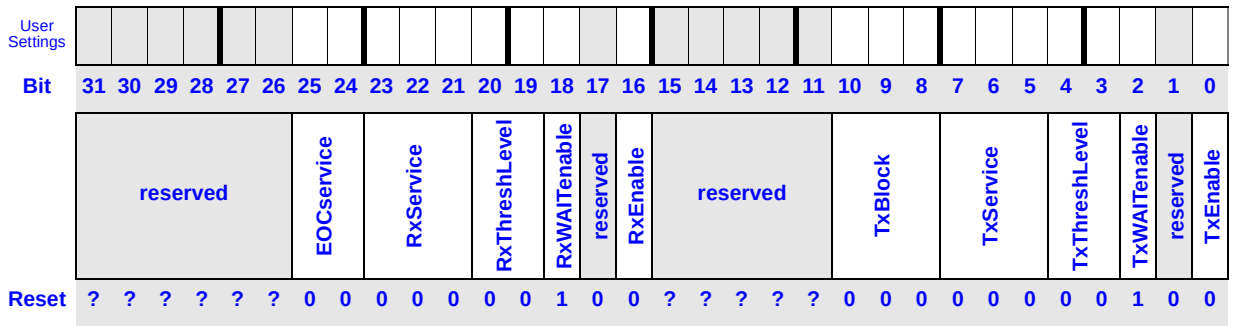
Physical Address		MSL Interface
0x4140_0044	BBCFG1	
0x4140_0048	BBCFG2	
0x4140_004C	BBCFG3	
0x4140_0050	BBCFG4	
0x4140_0054	BBCFG5	
0x4140_0058	BBCFG6	
0x4140_005C	BBCFG7	

User Settings																															
Bit	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						EOCservice		RxService			RxThreshLevel		RxWAITenable	reserved	RxEnable	reserved					TxBlock			TxService				TxThreshLevel		TxWAITenable	reserved	TxEnable
Reset	?	?	?	?	?	?	0	0	0	0	0	0	0	1	0	0	?	?	?	?	?	0	0	0	0	0	0	0	0	1	0	0

Bits	Access	Name	Description
31:26	—	—	reserved
25:24	R/W	EOCservice	Early EOC Service Select 0b00 = None 0b01 = Interrupt 0b1x = reserved
23:21	R/W	RxService	Receive FIFO Service Select 0b000 = None 0b001 = DMA 0b010 = Interrupt 0b011 = reserved 0b1xx = reserved

Table 16-5. BBCFG1/2/3/4/5/6/7 Bit Definitions (Sheet 2 of 3)

Physical Address		
0x4140_0044	BBCFG1	MSL Interface
0x4140_0048	BBCFG2	
0x4140_004C	BBCFG3	
0x4140_0050	BBCFG4	
0x4140_0054	BBCFG5	
0x4140_0058	BBCFG6	
0x4140_005C	BBCFG7	



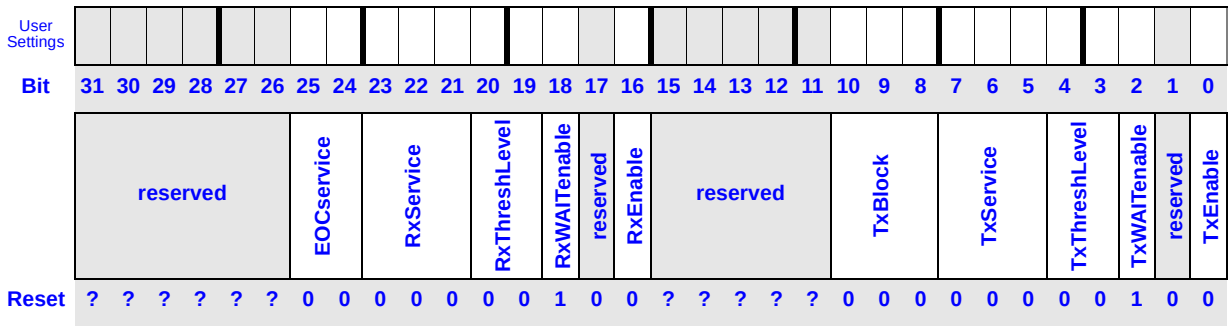
Bits	Access	Name	Description
20:19	R/W	RxThresh Level	Receive FIFO Service Trigger Threshold 0b00 = 4 bytes 0b01 = 8 bytes 0b10 = 16 bytes 0b11 = 32 bytes
18	R/W	RxWAIT enable	Direct Flow Control Enable 0 = Disabled 1 = Enabled
17	—	—	reserved
16	R/W	RxEnable	Receive FIFO Channel Enable 0 = Channel disabled 1 = Channel enabled
15:11	—	—	reserved
10:8	R/W	TxBlock	Transmit Block Size 0b000 = 4 bytes 0b001 = 8 bytes 0b010 = 16 bytes 0b011 = 32 bytes 0b1xx = reserved
7:5	R/W	TxService	Transmit FIFO Service Select 0b000 = None 0b001 = DMA 0b010 = Interrupt 0b011 = reserved 0b1xx = reserved

Table 16-5. BBCFG1/2/3/4/5/6/7 Bit Definitions (Sheet 3 of 3)

Physical Address

0x4140_0044	BBCFG1
0x4140_0048	BBCFG2
0x4140_004C	BBCFG3
0x4140_0050	BBCFG4
0x4140_0054	BBCFG5
0x4140_0058	BBCFG6
0x4140_005C	BBCFG7

MSL Interface



Bits	Access	Name	Description
4:3	R/W	TxThresh Level	Transmit FIFO Service Trigger Threshold 0b00 = 4 bytes 0b01 = 8 bytes 0b10 = 16 bytes 0b11 = 32 bytes
2	R/W	TxWAIT-enable	Direct Flow Control Enable 0 = Disabled 1 = Enabled
1	—	—	reserved
0	R/W	TxEnable	Transmit FIFO Channel Enable 0 = Channel disabled 1 = Channel enabled

16.5.3 MSL Channel Status Registers (BBSTATx)

Each channel has associated with it a channel status register, which contains status information about the channel. Table 16-6 shows the bit layout of these registers.

The RxEOM_x bits correspond to the next four bytes to be read from the receive FIFO. If one of these bits is set, the corresponding byte is the last in a message. For example, if the receive FIFO is initially empty and a four-byte message is received, the RxEOM_x bits have the value 0b1000. If there are less than four bytes left in the receive FIFO, extra RxEOM bits remain clear. For example, if the receive FIFO is initially empty and a two-byte message is received, the RxEOM_x bits have the value 0b0010. RxEOM_3 and RxEOM_2 remain clear, since they do not correspond to any data in the FIFO.

The RxEOM_FIFO bit indicates if the end of a message is in the 64-byte receive FIFO buffer. This bit is set when an EOM notification is received. It is cleared when the last byte of a message is read and there are no more EOM bytes in the FIFO.

Use RxEOM and RxEOM_FIFO together when software does not know the length of a message or the location of the EOM:

1. Before reading the receive FIFO, check the RxEOM_FIFO bit.
2. If RxEOM_FIFO is set, check the RxEOM bits to see if the end-of-message is in the next four bytes of the FIFO.

Assume, for example, that there are two messages in the receive FIFO and the RxEOM_x bits have the value 0b0010. Without knowing this value, software cannot determine that it has read the last two bytes of the first message and is ready to start reading the second message.

The RxWait and TxWait bits, if set, indicate that a channel is in a wait state.

- For a transmit channel, this occurs when either:
  - a wait signal was received on the last attempt to transmit to a channel, or
  - a stop message but no start message has been received for the channel
- For a receive channel, this occurs when either:
  - BBCFGx[RxWAITenable] is set and the receive FIFO is full, or
  - a stop message but no start message has been sent for the channel.

Note: The RxWait and TxWait bits depend on the setting of BBCFGx[xxWAITenable]. They are *not* affected by BBCFGx[RxEnable] or BBCFGx[TxEnable].

RxEmpty and TxEmpty, when set, indicate that the corresponding FIFO has no data in it.

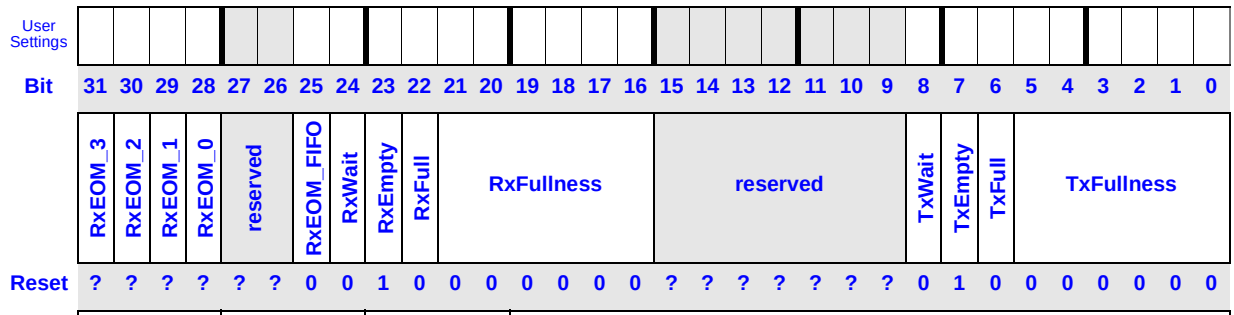
RxFull and TxFull, when set, indicate that the corresponding FIFO has no more room for data.

The RxFullness and TxFullness fields indicate the number of bytes of data remaining in the corresponding FIFO, *except* when the FIFO is full. In this case, the xxFullness field has a value of zero and the xxFull bit is set. Thus, the combined bits xxFull and xxFullness, with xxFull as the most-significant bit, have the value 0b1000000 (64), which is the number of bytes in a full FIFO.

These are read-only registers. Ignore reads from reserved bits.

Table 16-6. BBSTAT1/2/3/4/5/6/7 Bit Definitions (Sheet 1 of 2)

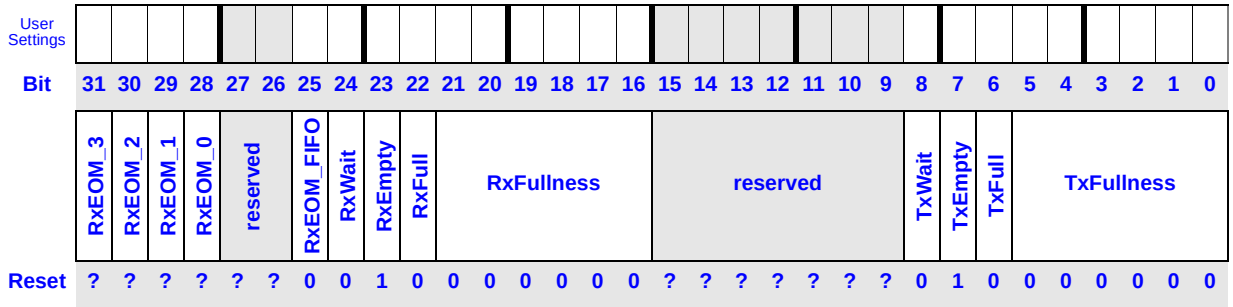
Physical Address		
0x4140_0084	BBSTAT1	MSL Interface
0x4140_0088	BBSTAT2	
0x4140_008C	BBSTAT3	
0x4140_0090	BBSTAT4	
0x4140_0094	BBSTAT5	
0x4140_0098	BBSTAT6	
0x4140_009C	BBSTAT7	



Bits	Access	Name	Description
31	R	RxEOM_3	Fourth Byte to be Read from FIFO is Last Byte in a Message 0 = Fourth byte is not the last in a message. 1 = Fourth byte is the last in a message.
30	R	RxEOM_2	Third Byte to be Read from FIFO is Last Byte in a Message 0 = Third byte is not the last in a message. 1 = Third byte is the last in a message.
29	R	RxEOM_1	Second Byte to be Read from FIFO is Last Byte in a Message 0 = Second byte is not the last in a message. 1 = Second byte is the last in a message.
28	R	RxEOM_0	Next Byte to be Read from FIFO is Last Byte in a Message 0 = Next byte is not the last in a message. 1 = Next byte is the last in a message.
27:26	—	—	reserved
25	R	Rx EOM_FIFO	Receive FIFO Contains an EOM 0 = FIFO does not contain an EOM. 1 = FIFO does contain an EOM.
24	R	RxWait	Receive Channel in Wait State 0 = Channel not in wait state. 1 = Channel in wait state.
23	R	RxEmpty	Receive FIFO Empty 0 = Not empty 1 = Empty
22	R	RxFull	Receive FIFO Full 0 = Not full 1 = Full
21:16	R	RxFullness	Fullness of Receive FIFO 000000 = FIFO is either full or empty (see RxFull and RxEmpty bits). nonzero = Number of bytes of data in receive FIFO.
15:9	—	—	reserved

Table 16-6. BBSTAT1/2/3/4/5/6/7 Bit Definitions (Sheet 2 of 2)

Physical Address		
0x4140_0084	BBSTAT1	MSL Interface
0x4140_0088	BBSTAT2	
0x4140_008C	BBSTAT3	
0x4140_0090	BBSTAT4	
0x4140_0094	BBSTAT5	
0x4140_0098	BBSTAT6	
0x4140_009C	BBSTAT7	



Bit	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
	RxEOM_3	RxEOM_2	RxEOM_1	RxEOM_0	reserved		RxEOM_FIFO	RxWait	RxEmpty	RxFull	RxFullness						reserved							TxWait	TxEmpty	TxFull	TxFullness					
Reset	?	?	?	?	?	?	0	0	1	0	0	0	0	0	0	0	?	?	?	?	?	?	?	0	1	0	0	0	0	0	0	0

Bits	Access	Name	Description
8	R	TxWait	Transmit Channel in Wait State 0 = Channel not in wait state. 1 = Channel in wait state.
7	R	TxEmpty	Transmit FIFO Empty 0 = Not empty 1 = Empty
6	R	TxFull	Transmit FIFO Full 0 = Not full 1 = Full
5:0	R	TxFullness	Fullness of Transmit FIFO 0b000000 = FIFO is either full or empty (see TxFull and TxEmpty bits) nonzero = number of bytes of data in transmit FIFO

16.5.4 MSL Channel EOM Registers (BBEOMx)

Each channel has associated with it an end-of message (EOM) register, which indicates that an entire message has been written to the corresponding transmit FIFO. Table 16-7 shows the BBEOMx bit layout. These are write-only registers; reads yield undefined results.

Any write to this register indicates an EOM by setting an EOM bit in the transmit FIFO. When data marked with a set EOM bit is transmitted, the interface switches to the EOM channel, signifying an EOM.

Use this register with processor-initiated messages only. DMA-initiated messages automatically signal the EOM and do not require this step.

These are write-only registers. Write 0b0 to reserved bits.

Table 16-7. BBEOM1/2/3/4/5/6/7 Bit Definitions

Physical Address		MSL Interface
0x4140_00C4	BBEOM1	
0x4140_00C8	BBEOM2	
0x4140_00CC	BBEOM3	
0x4140_00D0	BBEOM4	
0x4140_00D4	BBEOM5	
0x4140_00D8	BBEOM6	
0x4140_00DC	BBEOM7	

User Settings																															
Bit	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
	EOM																															
Reset	N/A																															

Bits	Access	Name	Description
31:0	W	EOM	Write any value to indicate EOM.

16.5.5 MSL Interrupt ID Register (BBIID)

The MSL Interrupt ID register, shown in Table 16-8, identifies the type of interrupt that occurred.

Interrupts can be generated when a FIFO reaches its trigger threshold BBCFGx[xxThreshLevel], as set in its channel configuration register, an EOM is received, or the end of a DMA descriptor chain is reached before the end of a message is read. Generating an interrupt at the EOM is required to service any trailing bytes. Generating an interrupt on an early DMA end-of-channel is required to inform the processor of improper DMA programming. Each type of interrupt has a bit associated with it in register BBIID. When a interrupt occurs, the corresponding BBIID bit is set. Once a *BBIID* bit is set, the CPU must clear it by writing one to it. Writing zero to a BBIID bit has no effect. If a different interrupt occurs before the previous one has been serviced, both BBID bits are set, each of which must be cleared.

Note: An interrupt bit cannot be cleared until the condition that caused the interrupt has been cleared. For example, if receive FIFO 1 exceeds its trigger threshold and sets the RX_INT1 bit, data must be read from that FIFO until the trigger threshold is no longer exceeded before RX_INT1 bit can be cleared.

This a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 16-8. BBIID Bit Definitions

Physical Address 0x4140_0108								BBIID																MSL Interface							
User Settings																															
Bit	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								TX_INT7	TX_INT6	TX_INT5	TX_INT4	TX_INT3	TX_INT2	TX_INT1	reserved	EOC_INT7	EOC_INT6	EOC_INT5	EOC_INT4	EOC_INT3	EOC_INT2	EOC_INT1	reserved	RX_INT7	RX_INT6	RX_INT5	RX_INT4	RX_INT3	RX_INT2	RX_INT1	reserved
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:24	—	—	reserved
23:17	R/W†	TX_INTx	Transmit FIFO Interrupt for Channel x
16	—	—	reserved
15:9	R/W†	EOC_INTx	EOC Interrupt for Channel x
8	—	—	reserved
7:1	R/W†	RX_INTx	Receive FIFO Interrupt for Channel x
0	—	—	reserved

† Write 0b1 to clear the bit.

16.5.6 MSL Transmit Frequency Select Register (BBFREQ)

Register BBFREQ, shown in Table 16-9, determines the clock speed (transmit frequency) for the outbound link.

The transmit clock frequency is the interface clock (48-MHz) divided by the decimal value of the DIV bit field:

clock frequency = 48 MHz / BBFREQ[DIV]

DIV = 0 is illegal; if zero is stored, one is actually used. Since up to one nibble is transmitted each clock cycle, the peak transfer rate is 192 Mbps.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 16-9. BBFREQ Bit Definitions

	Physical Address 0x4140_0110												BBFREQ												MSL Interface											
User Settings																																				
Bit	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																				DIV								reserved							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	?	?	?	?				

Bits	Access	Name	Description
31:12	—	—	reserved
11:4	R/W	DIV	BB_INT_CLK Clock Divider
3:0	—	—	reserved

16.5.7 MSL Wait Count Register (BBWAIT)

Register BBWAIT, shown in Table 16-10, determines the time, in transmit clock cycles, that the transmit control block waits before retrying a transmit to a channel that has sent a wait signal. BBWAIT[Count] applies to all seven channels, each of which has an independent wait counter.

During the wait period (BBWAIT[Count] clock cycles), either the link remains idle or another channel is activated. If another wait signal is received during a subsequent transmit attempt, the source waits for another period and repeats until the data has been successfully transmitted.

Reprogramming this register clears all wait counters, allowing all channels in wait state to be retried immediately.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 16-10. BBWAIT Bit Definitions

Physical Address 0x4140_0114 — BBWAIT — MSL Interface

	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
User Settings																																
Bit	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																						Count									
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

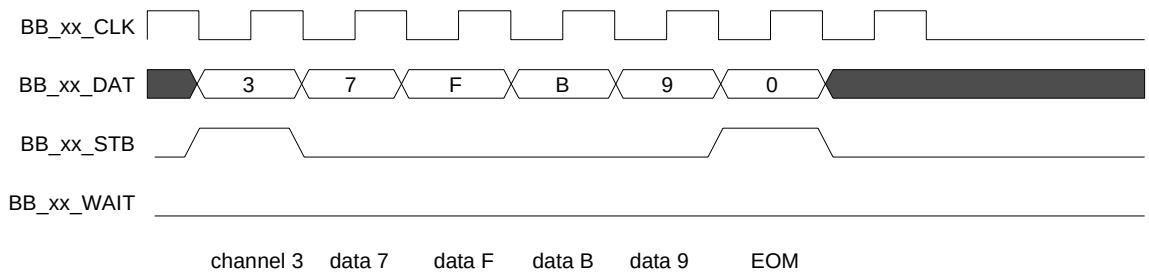
Bits	Access	Name	Description
31:10	—	—	reserved
9:0	R/W	Count	Number of Transmit Clock Cycles Waited before Retrying a Waiting Channel

16.5.8 MSL Clock Stop Time Register (BBCST)

Register BBCST, shown in Table 16-11, determines the time (in transmit clock cycles) that the BB_OB_CLK signal stops toggling after the outbound link becomes idle.

Figure 16-15 shows an example waveform in which the link becomes idle and the clock is stopped one cycle after the idle time begins. Idle time begins on the first cycle the BB_xx_WAIT signal is sampled high, or when the active channel is changed to the null or idle channel. Since, in this example, the clock stop time is one, the clock turns off one cycle after the link becomes idle.

Figure 16-15. Example Waveform of Clock Stopping One Cycle After Idle



This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 16-11. BBCST Bit Definitions

Physical Address 0x4140_0118 — BBCST — MSL Interface

	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
User Settings																																
Bit	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																												Count			
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0

Bits	Access	Name	Description
31:4	—	—	reserved
3:0	R/W	Count	Count Number of TX clock cycles to wait before turning off clock when there is no data to transmit. 0xF: Clock never stops. 0xN: Number of clock cycles to wait.

16.5.9 MSL Wake-Up Register (BBWAKE)

Use register BBWAKE, shown in Table 16-12, to wake up the connected chip. Any write to this register initiates a wake-up message by causing a transmission over channel 12, the wake-up channel. See Section 16.4.4 for more on power management.

This is a write-only register. Reads from it yield undefined results.

Table 16-12. BBWAKE Bit Definitions

Physical Address 0x4140_00140 — BBWAKE — MSL Interface

	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
User Settings																																
Bit	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
	WAKE																															
Reset	N/A																															

Bits	Access	Name	Description
31:0	W	WAKE	Write any value to initiate wake-up message transmission.

16.5.10 MSL Interface Width Register (BBITFC)

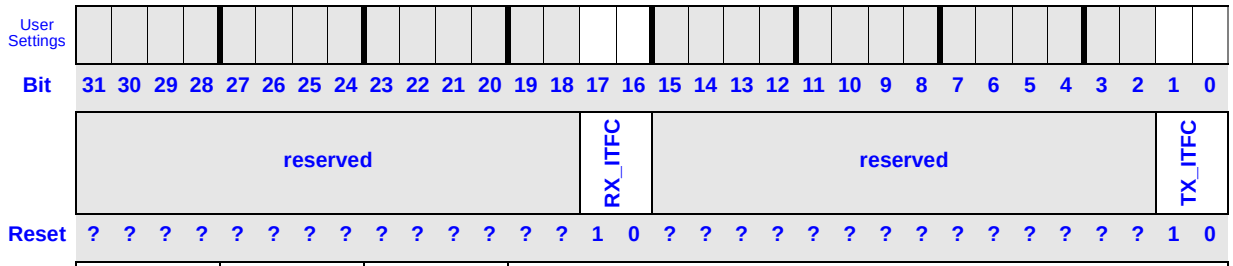
Register BBITFC, shown in Table 16-13, contains configuration bits that specify the width of the interface.

Update this register only when the affected link is completely idle. Changing the interface width while any activity is occurring on the link can yield unpredictable results. Reads from this register do not affect link activity.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 16-13. BBITFC Bit Definitions

Physical Address 0x4140_0144 — BBITFC — MSL Interface



Bits	Access	Name	Description
31:18	—	—	reserved
17:16	R/W	RX_ITFC	Receive (Inbound) Link Interface Width 0b00 = 1 bit 0b01 = 2 bits 0b10 = 4 bits 0b11 = reserved
15:2	—	—	reserved
1:0	R/W	TX_ITFC	Transmit (Outbound) Link Interface Width 0b00 = 1 bit 0b01 = 2 bits 0b10 = 4 bits 0b11 = reserved

16.6 Register Summary

Table 16-14 summarizes the registers for the MSL interface and their addresses.

Table 16-14. MSL Interface Register Summary (Sheet 1 of 2)

Address	Name	Description	Page
0x4140_0004	BBFIFO1	MSL Channel 1 Receive/Transmit FIFO register	16-13
0x4140_0008	BBFIFO2	MSL Channel 2 Receive/Transmit FIFO register	16-13
0x4140_000C	BBFIFO3	MSL Channel 3 Receive/Transmit FIFO register	16-13
0x4140_0010	BBFIFO4	MSL Channel 4 Receive/Transmit FIFO register	16-13
0x4140_0014	BBFIFO5	MSL Channel 5 Receive/Transmit FIFO register	16-13
0x4140_0018	BBFIFO6	MSL Channel 6 Receive/Transmit FIFO register	16-13
0x4140_001C	BBFIFO7	MSL Channel 7 Receive/Transmit FIFO register	16-13
0x4140_0020–0x4140_0040	—	reserved	
0x4140_0044	BBCFG1	MSL Channel 1 Configuration register	16-15
0x4140_0048	BBCFG2	MSL Channel 2 Configuration register	16-15
0x4140_004C	BBCFG3	MSL Channel 3 Configuration register	16-15

Table 16-14. MSL Interface Register Summary (Sheet 2 of 2)

Address	Name	Description	Page
0x4140_0050	BBCFG4	MSL Channel 4 Configuration register	16-15
0x4140_0054	BBCFG5	MSL Channel 5 Configuration register	16-15
0x4140_0058	BBCFG6	MSL Channel 6 Configuration register	16-15
0x4140_005C	BBCFG7	MSL Channel 7 Configuration register	16-15
0x4140_0060–0x4140_0080	—	reserved	
0x4140_0084	BBSTAT1	MSL Channel 1 Status register	16-19
0x4140_0088	BBSTAT2	MSL Channel 2 Status register	16-19
0x4140_008C	BBSTAT3	MSL Channel 3 Status register	16-19
0x4140_0090	BBSTAT4	MSL Channel 4 Status register	16-19
0x4140_0094	BBSTAT5	MSL Channel 5 Status register	16-19
0x4140_0098	BBSTAT6	MSL Channel 6 Status register	16-19
0x4140_009C	BBSTAT7	MSL Channel 7 Status register	16-19
0x4140_00A0–0x4140_00C0	—	reserved	
0x4140_00C4	BBEOM1	MSL Channel 1 EOM register	16-22
0x4140_00C8	BBEOM2	MSL Channel 2 EOM register	16-22
0x4140_00CC	BBEOM3	MSL Channel 3 EOM register	16-22
0x4140_00D0	BBEOM4	MSL Channel 4 EOM register	16-22
0x4140_00D4	BBEOM5	MSL Channel 5 EOM register	16-22
0x4140_00D8	BBEOM6	MSL Channel 6 EOM register	16-22
0x4140_00DC	BBEOM7	MSL Channel 7 EOM register	16-22
0x4140_00E0–0x4140_00FC	—	reserved	
0x4140_0100–0x4140_0104	—	reserved	
0x4140_0108	BBIID	MSL Interrupt ID register	16-23
0x4140_010C	—	reserved	
0x4140_0110	BBFREQ	MSL Transmit Frequency Select register	16-4
0x4140_0114	BBWAIT	MSL Wait Count register	16-24
0x4140_0118	BBCST	MSL Clock Stop Time register	16-25
0x4140_011C–0x4140_0138	—	reserved	
0x4140_013C	—	reserved	
0x4140_0140	BBWAKE	MSL Wake-Up register	16-26
0x4140_0144	BBITFC	MSL Interface Width register	16-4
0x4140_0148–0x414F_FFFC	—	reserved	—

Memory Stick Host Controller 17

This chapter describes the memory stick host controller (MSHC) for Sony Corporation's Memory Stick.

Note: Additional technical information about this subject matter can be obtained directly from Sony Corporation. Sony may require the execution of a license or other documents prior to releasing such information to a recipient. Intel Corporation expressly disclaims all liability and responsibility for information disclosed by Sony Corporation to a recipient, and makes no representation or warranty respecting such information whatsoever.

17.1 Overview

The Memory Stick is a medium for storing and transferring data. In its simplest form, the Memory Stick is a small, pluggable card containing flash (or other similar) memory. This memory can store multiple content types—for example, audio data or stored image data. In addition to this basic form, other devices are available that use the standard Memory Stick definition (for example, camera modules). The memory stick host controller provides the interface between the PXA27x processor and one Memory Stick.

Note: The *Sony Memory Stick Standard, Format Specification Version 1.3* (the Sony MS standard) defines the Memory Stick interface and provides the basis for this chapter. For all details of Memory Stick operation, refer to this Sony standard.

17.2 Features

- Compliance with the Sony Memory Stick standard
- Built-in transmit and receive FIFO buffers
- Built-in CRC calculation and checking
- Transfer clock up to 20 MHz
- Data transfer using programmed I/O, interrupt to processor, and DMA
- Automatic command execution when an interrupt from the Memory Stick is detected

17.3 Signal Descriptions

Table 17-1 summarizes the signals used by the memory stick host controller.

Table 17-1. Memory Stick Host Controller I/O Signal Descriptions

Name	Type	Description
MSBS	Output	Serial protocol bus state signal
MSSDIO	Bidirectional	Serial protocol data signal
nMSINS	Input	Stick insertion/removal detect terminal
MSSCLK	Output	Serial protocol clock signal

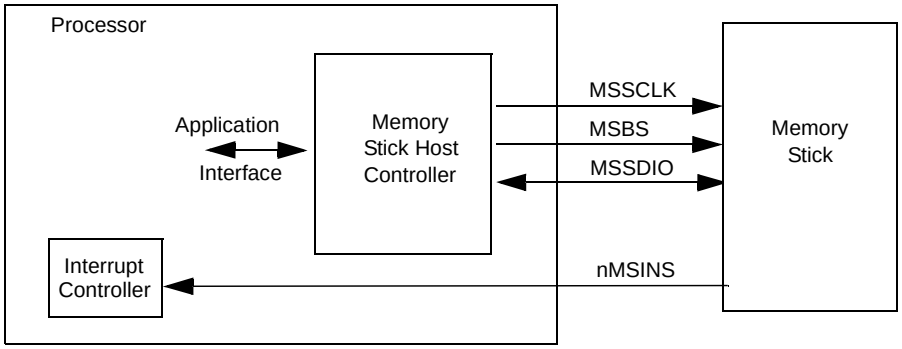
17.4 Operation

The memory stick host controller communicates with the Memory Stick using a half-duplex serial protocol. This section presents an overview of the protocol. Always refer to the Sony MS standard for details.

17.4.1 Functional Description

The Memory Stick system, depicted in Figure 17-1, consists of the memory stick host controller and an attached Memory Stick card.

Figure 17-1. Memory Stick System Block Diagram



The memory stick host controller interfaces with the Memory Stick using a 32-bit internal application interface. It allows:

- Sending of *transfer protocol commands* (TPCs) to the Memory Stick using the MSHC Command register.
- Data transfer using the two separate receive (RX) and transmit (TX) FIFOs (this data can be transferred using polling, interrupts to the processor, or DMA).
- Responding directly to Memory Stick interrupts by issuing a predefined command, the *AutoCommand* (ACD).
- Placing the card into a low-power mode.

17.4.2 Interrupts

The memory stick host controller generates a single interrupt to the interrupt controller. The cause of the interrupt can be determined by reading the Interrupt and Status register (Section 17.5.3). Status bits in this register indicate which event caused the interrupt to be generated. Interrupts can be disabled (either individually or in total) by setting bits in the Interrupt Enable register (Section 17.5.4).

To process an interrupt, MSINTEN[INTEN] must be set, and the enable bit for the specific interrupt in register MSINTEN must be set.

It is possible for multiple secondary interrupts to occur if more than one MSINTEN register bit besides INTEN are set. To determine the specific cause of the interrupt, read the Interrupt and Status register, MSINT. To handle more than one interrupt at a time, set MSINTEN[INTEN] and the specific MSINTEN interrupt enable bits.

17.4.3 Memory Stick Insertion and Removal

The nMSINS signal indicates the insertion and removal of a Memory Stick. nMSINS is connected to the interrupt controller, but not to the memory stick host controller. Thus, the interrupt controller must be programmed to generate an nMSINS interrupt. See Chapter 25, “Interrupt Controller” for details.

Software must monitor the nMSINS interrupt. When an interrupt occurs due to the removal of the memory stick, the software must halt all MHSC activity and reset the memory stick host controller.

17.4.4 Reset

The memory stick host controller is reset in either of two ways:

- Any PXA27x processor reset causes all of the MSHC registers to be reset.
- Setting MSCRSR[RST] causes the memory stick host controller to enter and remain in reset until MSCRSR[RST] is cleared. In this case, all of the registers except for the *RST* bit are reset, and the output signals BS, SDIO, and SCLK are driven low. Any currently-executing protocol is terminated when MSCRSR[RST] is asserted.

These methods do not cause a reset TPC to be sent to the Memory Stick.

Note: Before writing any of the control registers for a new bus protocol, always set MSCRSR[RST] and then clear it.

17.4.5 TPC Code Errors

The memory stick host controller does not check the validity of the TPC that it receives. Therefore, the memory stick host controller responds to invalid TPCs as follows:

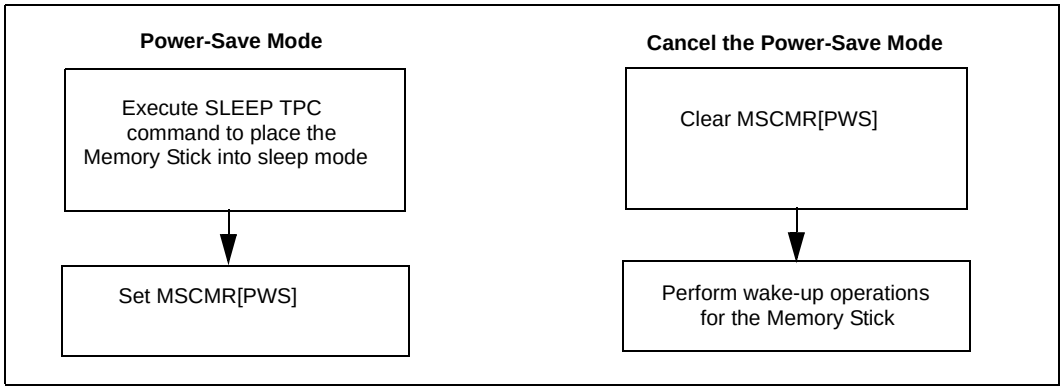
- If the TPC is invalid and TPC[3] = 0, then RDY and TOE occur.
- If the TPC is invalid, TPC[3] = 1, and the data size is zero, then RDY and TOE occur.
- If the TPC is invalid, TPC[3] = 1, and the data size does not equal zero, then SCLK goes high and the MSHC hangs until it is reset by writing to MSCRSR[RST].

17.4.6 Power-Save Mode

When the memory stick is not being used, software may optionally place it into a low-power mode. This is controlled entirely by software and is not related in any way to the PXA27x processor low-power modes.

Figure 17-2 shows how to enter and exit the Memory Stick power-save mode. To maximize overall system power saving, software must place the Memory Stick in power-save mode prior to entering any of the system low-power modes.

Figure 17-2. Procedure in Power-Save Mode



17.4.7 FIFO Operation

The memory stick host controller has a transmit FIFO and a receive FIFO, each holding 16 bytes of data. There are three separate methods for moving data to and from the FIFOs: interrupts, polling, and DMA.

Each FIFO has a trigger threshold that causes data-transfer service requests for interrupt and DMA operation. For the memory stick host controller, the threshold is fixed at eight bytes (half of the FIFO). Therefore, for the transmit FIFO, a service request is generated when the amount of empty space reaches eight bytes; for the receive FIFO, a service request is generated when the amount of data in the FIFO reaches eight bytes. There is no mode in which the FIFOs are not enabled.

Reads from MSRXFIFO and writes to MSTXFIFO transfer the number of bytes read or written, regardless of whether all bytes are valid. Use the appropriate instructions and combinations of instructions to ensure that only valid data is read from or written to the RX/TX FIFOs. Use the following instructions to transfer the corresponding number of bytes.

- To read a byte (8-bits) from the FIFO, use LDRB.
- To read a half-word (16-bits) from the FIFO, use LDRH.
- To read a word (32-bits) from the FIFO, use LDR.
- To write a byte (8-bits) to the FIFO, use STRB.
- To write a half-word (16-bits) to the FIFO, use STRH.
- To write a word (32-bits) to the FIFO, use STR.

To read only valid data from the receive FIFOs, do not read more bytes than indicated by MSCRSR[RXAVAIL]. If more bytes are read than are indicated by MSCRSR[RXAVAIL], then any data read beyond the indicated number of available bytes is indeterminate, and the value of MSCRSR[RXAVAIL] also takes on invalid values.

To read only valid data, read MSCRSR[RXAVAIL] first to determine how many bytes are available from MSRXFIFO. Then, use the proper combination of LDR, LDRH and LDRB instructions to read only the number of bytes that MSCRSR[RXAVAIL] indicates. For example, if MSCRSR[RXAVAIL] = 5, then no more than five bytes may be read from MSRXFIFO; otherwise, indeterminate behavior results. This can be accomplished by any of the following methods:

- Perform one LDR instruction and then one LDRB instruction, or
- Perform two LDRH instruction and then one LDRB instruction, or
- Perform five LDRB instructions.

After these bytes have been read, software must check MSCRSR[RXAVAIL] again to see if any new data has arrived in MSRXFIFO.

Similarly, to transfer only valid data to the Memory Stick, the exact number of bytes must be written to MSTXFIFO. If more than the exact number of bytes of valid data are written to the transmit FIFO, the extra (invalid) bytes are also transferred to the Memory Stick. For example, if five bytes are to be transferred to the Memory Stick, then only five bytes may be written to MSTXFIFO. This can be accomplished by any of the following methods:

- Perform one STR instruction, and then one STRB instruction, or
- Perform two STRH instructions, and then one STRB instruction, or
- Perform five STRB instructions.

17.4.7.1 FIFO Interrupt Mode Operation

Receive Interrupt

When the receive (RX) FIFO interrupt is enabled with MSINTEN[RXDAVEN], a receive interrupt occurs as follows:

- The receive data available interrupt (MSINT[RXDAV]) is asserted when the RX FIFO has reached the trigger threshold. The interrupt is cleared when the amount of data in the FIFO drops below the trigger threshold.
- The RX available data count (MSCRSR[RXAVAIL]) continuously indicates how much valid data is available in the FIFO (0 to 16 bytes). MSCRSR[RXAVAIL] indicates the amount of data that must be read from the FIFO.

Transmit Interrupt

When the transmit (TX) FIFO interrupt is enabled with MSINTEN(TXDAVEN), a transmit interrupt occurs as follows:

- The transmit data request interrupt (MSINT[TXDAV]) occurs when the TX FIFO has reached the trigger threshold. The interrupt is cleared when the amount of free space in the FIFO drops below the trigger threshold.
- The TX available data count (MSCRSR[TXAVAIL]) continuously indicates the amount of free space available in the FIFO (0 to 16 bytes). MSCRSR[TXAVAIL] controls the amount of data that is written into the FIFO.

17.4.7.2 FIFO Polled-Mode Operation

When neither interrupt nor DMA transfer mode is enabled for a FIFO, the FIFO operates in polled mode. In this mode, the FIFOs are accessed using processor read and writes in words, half words, and bytes.

Receive Data Service

Poll RX available data count (MSCRSR[RXAVAIL]) value to determine how much data is required (0 to 16 bytes).

Transmit Data Service

Poll the TX available data count (MSCRSR[TXAVAIL]) value to indicate how much space is available in the FIFO (0 to 16 bytes). This does **not** indicate how much data is required to complete the current transfer (this information must be separately maintained).

17.4.7.3 FIFO DMA-Mode Operation

There are two MSHC DMA service requests, one for receive data and one for transmit data.

Receive Data Service

To enable this service, set MSCMR2[RXDMAEN]. When the receive FIFO reaches its trigger threshold, the receive DMA request is generated if enabled. The DMA controller then reads data from the RX FIFO. For each DMA request, the DMA controller must read a total of eight bytes of data from the FIFO. The number of bytes to be read must be programmed in the DMA controller along with a bus width of 32 bits. The DMA controller typically attempts to read four bytes of data per transfer. In the case where less than four bytes are being transferred, the DMA controller is informed of the number of bytes transferred. The memory stick host controller can send one, two, or four bytes of data per bus transaction.

Transmit Data Service

To enable this service, set MSCMR2[TXDMAEN]. When the transmit FIFO has eight or more empty byte locations, the transmit DMA request is generated if enabled. The DMA controller then writes data to the TX FIFO. For each DMA request, the DMA controller must send no more than eight bytes of data to the FIFO (fewer bytes of data are acceptable and results in a new DMA request once the FIFO returns to eight empty bytes). This must be programmed in the DMA controller along with a bus width of 32 bits. The memory stick host controller accepts partial- or full-word transfers of one, two, or four consecutive bytes from the DMA controller.

Special Conditions

If an error occurs when DMA RX service requests are enabled:

- The receive DMA requests are disabled.
- The relevant error interrupt is generated if enabled.

The processor must now read previously-received bytes through programmed input/output (PIO). When all valid data entries have been removed from the FIFO, the memory stick host controller again enables the receive DMA requests.

Note: If an error occurs when the receive FIFO trigger threshold has been reached so that a receive DMA request is set, software must wait for the DMA controller to finish the transfer before reading the error bytes using PIO. If not, FIFO underflow could occur.

17.4.7.4 Trailing Bytes in the Receive FIFO

When the number of entries in the receive FIFO is less than its trigger threshold and no additional data is received, the remaining bytes (trailing bytes) are handled according to the enabled FIFO mode of operation. Trailing bytes are indicated when the FIFO contains less data than the trigger threshold and the current transfer has completed. If enabled by MSINTEN[RDYEN], the MSINT[RDY] interrupt is generated to indicate the end of a packet.

The mechanism for removing trailing bytes from the receive FIFO depends upon the FIFO mode of operation, as follows:

- **Interrupt mode**—No receive DMA request or *receive data available* interrupt is generated. To read the trailing bytes, wait for the MSINT[RXDAV] interrupt. Then, read all remaining bytes as indicated by MSCRSR[RXAVAIL]. The MSINT[RXDAV] interrupt must be enabled.
- **Polled mode**—No interrupts are enabled, and the software must remove data normally. The MSINT[RDY] bit must be checked to determine whether a packet has completed.
- **DMA mode**—When an end-of-packet occurs, the processor may receive an MSINT[RDY] interrupt, and a DMA request is automatically issued for the remaining number of bytes left in the receive FIFO. The DMA controller then empties the contents of the receive FIFO unless the DMA controller reaches the end of its descriptor chain. The MSINT[RDY] interrupt is for notification purposes only; the processor must take no action. If required, the interrupt can be disabled by setting MSINTEN[RDYEN].

17.5 Register Descriptions

The Sony MS standard (see Section 17.1) defines the control, status, and data registers used by the memory stick host controller. The register descriptions in this section outline and summarize the MSHC registers, but these descriptions provide only the implementation details that are specific to the PXA27x processor memory stick host controller. For full details of the registers and protocols, always refer to the Sony MS standard.

17.5.1 MSHC Command Register (MSCMR)

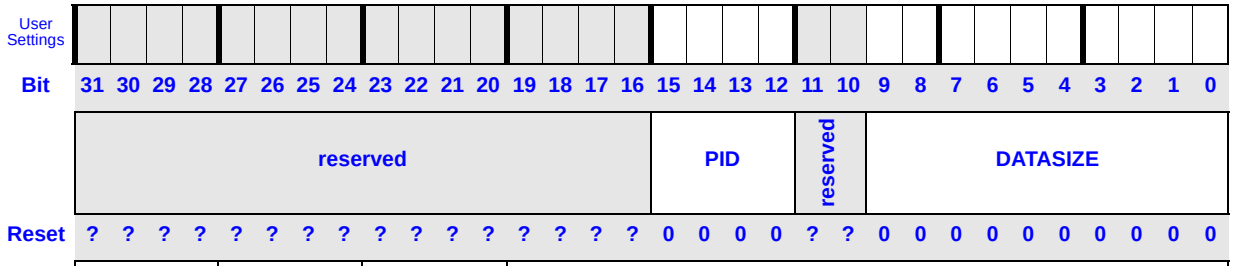
Writing to MSCMR, defined in Table 17-2, starts the protocol. Data cannot be written while a protocol is already in progress, as indicated by MSINT[RDY] being clear.

The DATASIZE is the number of data bytes to be transferred. The value must be greater than zero. Writing a value of zero to DATASIZE, a zero-byte transfer is not valid.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 17-2. MSCMR Bit Definitions

Physical Address 0x4180_0000 — **MSCMR** — **Memory Stick Host Controller**



Bits	Access	Name	Description
31:16	—	—	reserved
15:12	R/W	PID†	Packet ID
11:10	—	—	reserved
9:0	R/W	DATASIZE†	Data Size

† Refer to the Sony MS standard.

17.5.2 MSHC Control and Status Register (MSCRSR)

MSCRSR, defined in Table 17-3, contains control bits and status indicators to manage the data flow.

If MSCRSR[RST] is set, the memory stick host controller enters and remains in reset until MSCRSR[RST] is cleared. While RST is set, all MSHC registers except MSCRSR[RST] and MSCR2[BSYCNT] are cleared, and the output signals, BS, SDIO and SCLK are driven low.

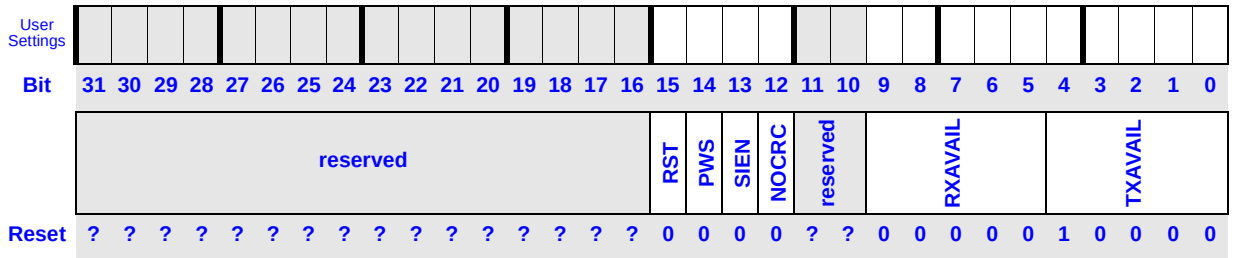
MSCRSR[RST] must always be set and then cleared before writing to any of the control registers for a new bus protocol.

Note: If the application software reads more bytes from the receive FIFO than specified in MSCRSR[RXAVAIL], then RXAVAIL becomes an illegal value, and data corruption results. If software reads more bytes from the transmit FIFO than specified in MSCRSR[TXAVAIL], then TXAVAIL becomes an illegal value, and data corruption results.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 17-3. MSCRSR Bit Definitions

Physical Address 0x4180_0004 — **MSCRSR** — **Memory Stick Host Controller**



Bits	Access	Name	Description
31:16	—	—	reserved
15	R/W	RST†	Reset
14	W	PWS†	Power Save
13	R/W	SIEN†	Serial I/F
12	R/W	NOCRC†	Controls CRC Generation and Checking
11:10	—	—	reserved
9:5	R/W	RXAVAIL	Receive Available Data Count The number of valid data bytes in the RX FIFO (0-16)
4:0	R/W	TXAVAIL	Transmit Available Data Count The number of available (unused) bytes in the TX FIFO (0-16)

† Refer to the Sony MS standard.

17.5.3 MSHC Interrupt and Status Register (MSINT)

MSINT, defined in Table 17-4, contains interrupt and error status for the memory stick host controller.

When an interrupt is disabled, the corresponding MSINT bit remains set, but no interrupt is issued to the processor. To disable interrupts, use register MSINTEN (see Section 17.5.4).

This is a read-only register. Ignore reads from reserved bits.

Table 17-4. MSINT Bit Definitions

Physical Address 0x4180_0008 — **MSINT** — **Memory Stick Host Controller**

Bit	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
User Settings																																
	reserved																INT	RXDAV	TXDAV	reserved								RDY	SIF	CRC	TOE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	?	?	?	?	?	?	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15	R	INT	Interrupt Present 0 = No interrupts are present. 1 = At least one interrupt is present.
14	R	RXDAV	Receive Data Available Interrupt 0 = Less than 8 bytes of valid data are available in the RX FIFO. 1 = At least 8 bytes of valid data are available in the RX FIFO.
13	R	TXDAV	Transmit Space Available Interrupt 0 = Less than 8 bytes of unused space is available in the TX FIFO. 1 = At least 8 bytes of unused space is available in the transmit FIFO.
12:4	—	—	reserved
3	R	RDY†	—
2	R	SIF†	—
1	R	CRC†	—
0	R	TOE†	—

† Refer to the Sony MS standard.

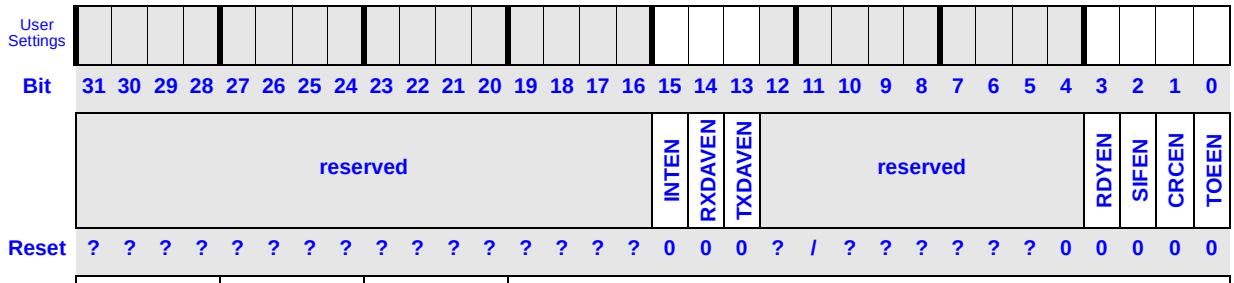
17.5.4 MSHC Interrupt Enable Register (MSINTEN)

MSINTEN, defined in Table 17-5, enables and disables the memory stick host controller interrupts.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 17-5. MSINTEN Bit Definitions

Physical Address 0x4180_000C — **MSINTEN** — **Memory Stick Host Controller**



Bits	Access	Name	Description
31:16	—	—	reserved
15	R/W	INTEN	Interrupts Enabled 0 = No interrupts are permitted from the memory stick host controller. 1 = Interrupts are enabled from the memory stick host controller.
14	R/W	RXDAVEN	Receive Data Available Interrupt Enable 0 = Interrupt is disabled. 1 = Interrupt is enabled.
13	R/W	TXDAVEN	Transmit Space Available Interrupt Enable 0 = Interrupt is disabled. 1 = Interrupt is enabled.
12:4	—	—	reserved
3	R/W	RDYEN	End of Packet Interrupt Enable 0 = Interrupt is disabled. 1 = Interrupt is enabled.
2	R/W	SIFEN	Stick Interface Interrupt Enable 0 = Interrupt is disabled. 1 = Interrupt is enabled.
1	R/W	CRCEN	CRC Error Interrupt Enable 0 = Interrupt is disabled. 1 = Interrupt is enabled.
0	R/W	TOEEN	Time-Out Error Interrupt Enable 0 = Interrupt is disabled. 1 = Interrupt is enabled.

17.5.5 MSHC Control Register 2 (MSCR2)

MSCR2, defined in Table 17-6, contains controls for enabling auto-command (ACD) mode, enabling DMA operation, and selecting the positive or negative clock edge for data loading.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 17-6. MSCR2 Bit Definitions

Physical Address 0x4180_0010 — **MSCR2** — **Memory Stick Host Controller**

Bit	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
User Settings																																
	reserved																ACD	reserved	TXDMAEN	RXDMAEN	reserved										BSYCNT		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	?	?	?	?	?	?	?	?	?	1	0	1

Bits	Access	Name	Description
31:16	—	—	reserved
15	R/W	ACD†	—
14	—	—	reserved
13	R/W	TXDMAEN	Transmit FIFO DMA Enable 0 = DMA operation is not enabled for the transmit FIFO. 1 = DMA operation is enabled for the transmit FIFO.
12	R/W	RXDMAEN	Receive FIFO DMA Enable 0 = DMA operation is not enabled for the receive FIFO 1 = DMA operation is enabled for the receive FIFO
11:3	—	—	reserved.
2:0	R/W	BSYCNT†	Busy Count Specifies the number of cycles the controller waits for 5 cycles of RDY. The number of cycles is calculated as (BSYCNT*4 + 1) BSYCNT = 1 always generates a TOE RDY time-out detection is not performed when BSYCNT = 0. BSYCNT is not reset with the assertion of MSCRSR[RST].

† Refer to the Sony MS standard.

17.5.6 MSHC ACD Command Register (MSACD)

MSACD, defined in Table 17-7, specifies the command to execute in response to a Memory Stick interrupt when ACD mode is enabled.

The ADATASIZE is the number of data bytes that are to be transferred. The value must be greater than 0. Writing a value of 0 to ADATASIZE, specifying a 0-byte transfer, is not valid.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 17-7. MSACD Bit Definitions

Physical Address 0x4180_0014																MSACD														Memory Stick Host Controller															
User Settings																																												
Bit	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																APID				reserved		ADATASIZE																			
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	1	1	?	?	0	0	0	0	0	0	0	0	0	1												

Bits	Access	Name	Description
31:16	—	—	reserved
15:12	R/W	APID†	—
11:10	—	—	reserved
9:0	R/W	ADATASIZE†	—

† Refer to the Sony MS standard.

17.5.7 MSHC Receive FIFO Register (MSRXFIFO)

MSRXFIFO, defined in Table 17-8, contains the data received from the Memory Stick.

MSRXFIFO is readable only in 1-, 2- or 4-byte quantities. 1-, 2- and 4-byte reads must be done with the following assembler directives: LDRB (byte read), LDRH (half word read), and LDR (word read), respectively.

Reads of a word or a half-word read multiple entries within the FIFO input buffer. A half-word read (LDRH) returns a single 16-bit value comprised of the first two entries of the input FIFO. A word read (LDR) returns a single 32-bit value comprised of the first four entries of the input FIFO. Refer to Section 17.4.7 for important information concerning the proper procedure for reading the MSHC Receive FIFO register.

This is a read-only register. Ignore reads from reserved bits.

Table 17-8. MSRXFIFO Bit Definitions

Physical Address 0x4180_0018 — **MSRXFIFO** — **Memory Stick Host Controller**

	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
User Settings																																
Bit	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																								RX Data							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R	RX Data	Data Received from Memory Stick

17.5.8 MSHC Transmit FIFO Register (MSTXFIFO)

MSTXFIFO, defined in Table 17-9, contains the data to be transmitted to the Memory Stick.

MSTXFIFO is writable only in 1-, 2-, or 4-byte quantities. 1-, 2- and 4-byte writes must be done with the following assembler directives: STRB (byte write), STRH (half word write), and STR (word write), respectively.

Stores of a word or a half-word write multiple entries within the FIFO output buffer. A half-word store (STRH) writes a single 16-bit value as two 8-bit entries to the output FIFO. A word read (LDR) writes a single 32-bit value as four 8-bit entries in the output FIFO.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 17-9. MSTXFIFO Bit Definitions

Physical Address 0x4180_001C	MSTXFIFO	Memory Stick Host Controller

User Settings																															
Bit	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																								TX Data							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	TX Data	Data to be Transmitted to Memory Stick

17.6 Register Summary

Table 17-10 summarizes the registers and memory mapping associated with the memory stick host controller for the Sony Memory Stick.

Table 17-11 provides cross references between the Sony register names and bit fields, as described in the Sony MS standard, and the equivalent PXA27x processor memory stick host controller register names and bit fields.

Table 17-10. Memory Stick Register Summary

Address	Name	Description	Page
0x4180_0000	MSCMR	MSHC Command register	17-8
0x4180_0004	MSCRSR	MSHC Control and Status register	17-9
0x4180_0008	MSINT	MSHC Interrupt and Status register	17-10
0x4180_000C	MSINTEN	MSHC Interrupt Enable register	17-11
0x4180_0010	MSCR2	MSHC Control register 2	17-12
0x4180_0014	MSACD	MSHC ACD Command register	17-13
0x4180_0018	MSRXFIFO	MSHC Receive FIFO register	17-14
0x4180_001C	MSTXFIFO	MSHC Transmit FIFO register	17-15
0x4180_0020– 0x418F_FFFC	—	reserved	—

Table 17-11. Sony-to-PXA27x Processor Memory Stick Register Reference

Sony MS Standard		PXA27x Processor MSHC Register and Bit Field
Register Name	**Bit Field**	
Command register	PID DATASIZE	MSCMR[15:12] MSCMR[9:0]
Status register	INT other bits	MSINT[15] not supported
Control register 1	RST PWS SIEN DAKEN NOCRC BSYCNT	MSCSR[15] MSCSR[14] MSCSR[13] not supported MSCSR[12] MSCR2[2:0]
Receive Data Buffer	—	not supported
Transmit Data Buffer	—	not supported
Interrupt Data register	RDY SIF CRC TOE other bits	MSINT[3] MSINT[2] MSINT[1] MSINT[0] not supported
Interrupt Control register	INTEN other bits	MSINTEN[15] not supported
PP Data register	—	not supported
PP Control register	—	not supported
Control register 2	ACD	MSCR2[15]
ACD Command register	APID ADATASIZE	MSACD[15:12] MSACD[9:0]

Keypad Interface 18

This chapter describes the keypad interface included in the PXA27x processor. This interface supports up to an 8 x 8 matrix keypad, up to eight direct keys, and up to two rotary encoders, which can implement items such as scroll keys, jog-dials, and thumbwheels.

18.1 Overview

The keypad interface provides an interface to two styles of keypads simultaneously:

- Matrix keypad interface, described in Section 18.1.1
- Direct keypad interface, described in Section 18.1.2

18.1.1 Matrix Keypad Interface

The matrix-keypad interface supports manual and automatic scans of the keypad array. For details of the matrix interface, see Section 18.4.1.

18.1.1.1 Manual Scan

Stable keypad activity that lasts longer than the key debounce interval generates an interrupt. A manual matrix scan can then be conducted by setting appropriate bits in the Keypad Interface Control register to sequentially assert the scan lines. The row readings for each of the columns are read from the Keypad Interface Matrix Keypad register as they are being scanned.

If Ignore Multiple Keypress is selected, only one interrupt is generated for a debounced keypress. For example, if three keys are depressed and held, only one interrupt is generated after the first key is depressed and held.

18.1.1.2 Automatic Scan

In automatic scan, the scan signals are automatically asserted in sequence by the automatic scan logic in the keypad interface, and the row readings are stored in the automatic scan registers. Automatic scans can be initiated by either of the following methods:

- If there is stable keypad activity for a period greater than the specified key-debounce interval while the Automatic Scan on Activity bit is set. Completion of the scan generates an interrupt.
- Setting the Automatic Scan bit. This method does not generate an interrupt, since user software determines when to initiate an automatic scan.

Depending on whether the Ignore Multiple Keypress bit is selected, manual scan and automatic scan by keypad activity take place as follows:

- **Ignore multiple keypress:** Multiple keypresses are ignored and not considered as new keypad activity. For example, if an automatic scan is conducted for a keypress activity, another automatic scan is not conducted until all keys are released.

- **Do not ignore multiple keypress:** For automatic scan on keypad activity, an automatic scan is conducted after a debounce interval for all of the time that:
  - — One or more keys have been pressed (new activity), or
  - — All keys have been released.

If the scanned activity is new, an interrupt is generated.

18.1.2 Direct Keypad Interface

The direct keypad interface supports the direct keys and the rotary encoders that implement features such as scroll keys, jog-dials, and thumbwheels. For details of the direct interface, see Section 18.4.2.

18.2 Features

The keypad interface is divided into two blocks: one each for the matrix and direct keypads. The direct-keypad block supports eight input pins, while the matrix-keypad block supports eight output and eight input pins.

- Direct-keypad interface:
  - — Eight inputs
  - — Supports up to eight direct keys and up to two rotary encoders
    - Eight direct keys, or
    - Six direct keys and one rotary encoder (two pins for the rotary encoder), or
    - Four direct keys and two rotary encoders (two pins each for the two rotary encoders)
- Matrix-keypad interface:
  - — Eight scan outputs and eight inputs (returns)
  - — Supports up to 64 keys
  - — Supports manual and automatic scan
- Simultaneous operation of direct and matrix keypads
- Interrupt generated on keypad activity:
  - — Separate matrix- and direct-key interrupt enables
  - — One interrupt signal, generated by merging the matrix and direct interrupts.
- Continuous keypad polling
- Key-debounce logic for both matrix and direct keypads

18.3 Signal Descriptions

The keypad interface I/O signals are summarized in Table 18-1.

Table 18-1. Keypad Interface I/O Signal Descriptions

Name	Type	Description
KP_DKIN<7:0>	Input	Direct key inputs signals from the direct keypad and the rotary encoder sensors. • KP_DKIN<7:4> are dedicated input pins for direct keys 7– 4. • KP_DKIN<3:2> are used as input pins for direct keys 3 and 2 or for rotary-encoder sensor readings for rotary encoder 1, if it is enabled. • KP_DKIN<1:0> are used as input pins for direct keys 1 and 0 or for rotary-encoder sensor readings for rotary encoder 0, if it is enabled.
KP_MKIN<7:0>	Input	Matrix-key return input signals from the matrix keypad and are the matrix-keypad row readings.
KP_MKOUT<7:0>	Output	Matrix-key outputs. The keypad interface sends scan signals to the columns of the matrix keypad to detect any key(s) pressed. If an automatic scan is occurring, these outputs are driven by the automatic scan logic. At other times, they are driven by the settings of bits MS7 to MS0 in the Keypad Interface Control register (KPC described in Table 18-3).

18.4 Operation

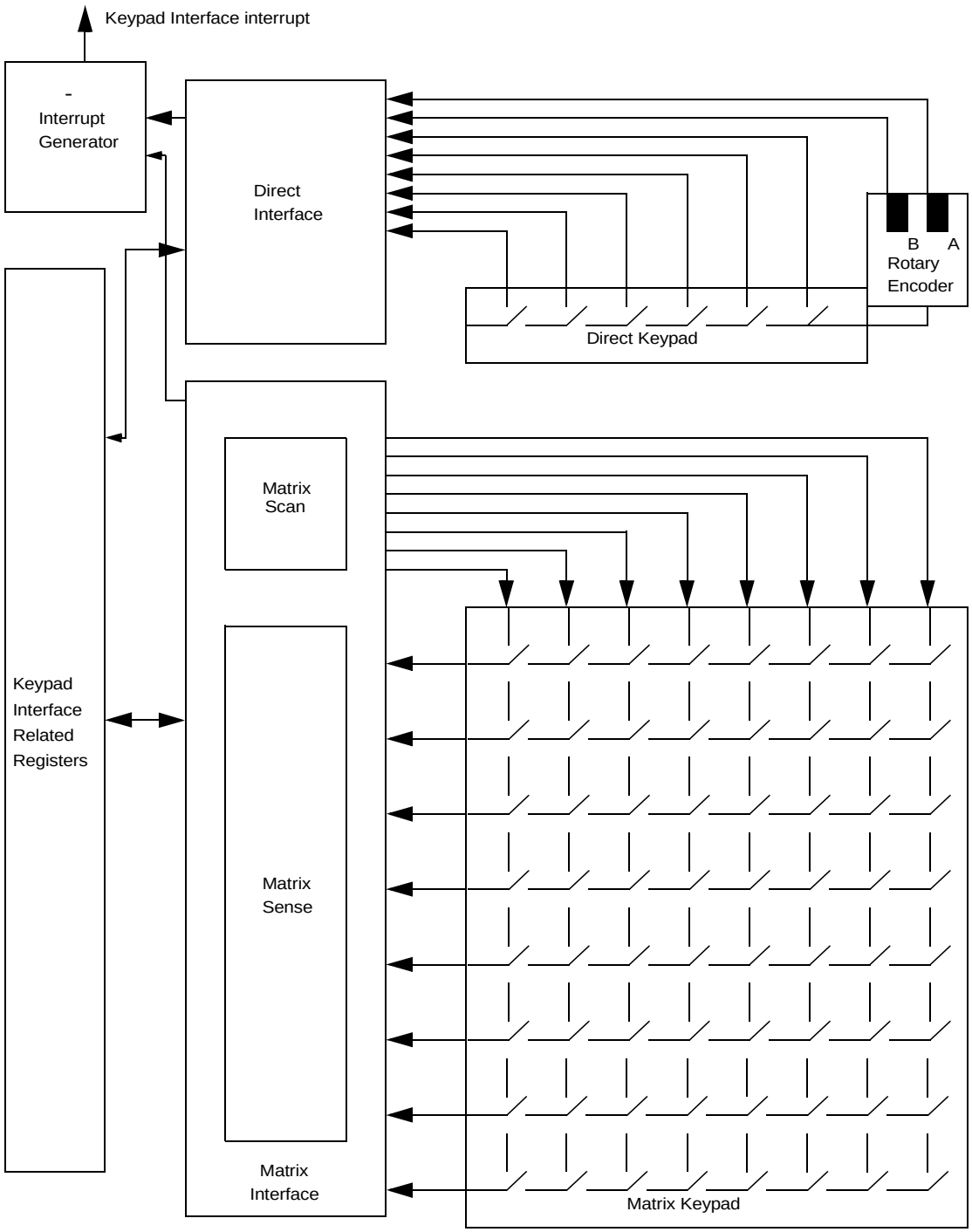
This section describes the functions of the keypad interface module. The interface supports up to an eight-by-eight matrix keypad, up to eight direct keys, and up to two rotary encoders. The Keypad Interface Control register see Table 18-3) controls the following operational items:

- Automatic scan initiation
- Enable/disable automatic-scan-on-activity
- Number of rows and columns in the matrix keypad
- Enable/disable ignore-multiple-keypresses (applies to matrix-keypad interface only)
- Enable/disable matrix keypad and matrix interrupt
- Number of keys in the direct keypad
- Enable/disable rotary encoders 0 and 1
- Enable/disable direct keypad and direct interrupt

The keypad interface connects the PXA27x processor to a matrix keypad and a direct keypad with rotary encoders. Direct and matrix operation can be enabled independently or simultaneously.

Figure 18-1 shows the keypad interface block connected to a typical 8 x 8 matrix keypad and to a direct keypad with six direct keys and one rotary encoder.

Figure 18-1. Keypad Interface Block Diagram



18.4.1 Matrix Interface

The matrix interface uses eight scan outputs to and eight sense inputs from the matrix keypad (see Section 18.3 for signal names). This eight-by-eight organization allows support for up to 64 keys.

Up to eight scan signals, corresponding to columns in the matrix keypad, are sent to the keypad to activate the columns. Up to eight input (sense) signals, corresponding to rows in a column that has been activated, are read into the following registers, depending on the mode of operation:

- Manual scans—Section 18.5.4
- Automatic scans—Section 18.5.5 and Section 18.5.6

The matrix-keypad interface has three scanning modes, described in the following sections:

- Manual scan
- Automatic scan initiated by keypad activity
- Automatic scan initiated by user software.

18.4.1.1 Manual Matrix Scan

To enable manual matrix-keypad scanning:

1. Clear the Automatic Scan on Activity bit, KPC[ASACT].
2. Clear the Automatic Scan bit, KPC[AS].
3. Assert all column (scan output) lines by setting bits KPC[MS0:MS7].

In this polling mode of operation, all keypad columns remain constantly activated. Any keypress that exceeds the key-debounce interval generates an interrupt. Service the interrupt as follows:

1. Deassert all of the columns by clearing bits KPC[MS0:MS7].
2. Sequentially assert bits KPC[MS0:MS7] so as to scan all keypad columns.
3. As the columns are being scanned, read each column's row (scan input) values, which appear in bits KPMK[MR0:MR7].
4. After servicing the interrupt, set bits KPC[MS0:MS7]. This leaves the scanning logic ready to detect the next keypress.

Manual scanning operates in one of two modes, depending upon the setting of the Ignore Multiple Keypress bit, KPC[IMKP]:

- **Ignore multiple keypresses**

  Set KPC[IMKP]. After a keypad interrupt is generated for a keypress activity, no further interrupts occur until all keys pressed have been released, for example if three keys are depressed and held, only one interrupt is generated after the first key is depressed and held.

- **Do *not* ignore multiple keypresses**

  Clear KPC[IMKP]. For the following keypad activities:

  — If a key is pressed, or multiple keys are pressed and held, an interrupt is generated after every debounce interval expires for as long as any key is pressed, for example, if three keys are pressed and held for four debounce intervals, four interrupts are generated.

  — An interrupt is generated after all keys are released.

18.4.1.2 Automatic Matrix Scan Initiated by Keypad Activity

To enable automatic matrix-keypad scanning initiated by keypad activity:

1. Set the Automatic Scan on Activity bit, KPC[ASACT].
2. Clear the Automatic Scan bit, KPC[AS].

In this polling mode of operation, the setting of the Ignore Multiple keypress bit (KPC[IMKP] defines keypress activity. An interrupt is generated after the automatic scan is completed and new keypad activity is detected. The scanning logic performs the following sequence of events automatically:

Note: User software cannot access the internal registers used in this procedure, KPASMKP_0x and KPASMKP_1x.

1. Detect matrix keypad activity.
2. Assert the column scan lines sequentially, storing the row readings for each in KPASMKP_0x.
3. Allow one debounce interval to elapse, as defined in KPKDI[Interval].
4. Assert the column scan lines sequentially, storing the row readings for each in KPASMKP_1x.
5. If internal registers KPASMKP_0x match the corresponding internal registers KPASMKP_1x (the activity is stable) **and** the internally stored activity differs from user-accessible registers KPASMKPx (the activity is new), then the internal values are copied into the user-readable registers KPAS and KMASMKPx.

The KPAS[MUKP] bit field identifies whether a single key or multiple keys were pressed. For **single keypresses**, read the column and row information directly from KPAS[CP] and KPAS[RP]. For **multiple keypresses**, read the column and row information from registers KPASMKP0-3.

Automatic scanning operates in one of two modes, depending upon the setting of the Ignore Multiple Keypress bit, KPC[IMKP]:

- **Ignore multiple keypresses**

  Set KPC[IMKP]. After a keypad activity has been scanned, no further automatic scans occur until all keys pressed have been released.

- **Do *not* ignore multiple keypresses**

  Clear KPC[IMKP]. Automatic scans occur for the following keypad activities:

  — Every keypress that exceeds the key-debounce interval, for as long as any key is pressed

  — All keys released.

18.4.1.3 Automatic Matrix Scan Initiated by Software

To enable automatic matrix-keypad scanning initiated by user software, set the Automatic Scan bit, KPC[AS]. In this mode of operation, the keypad is scanned only once, and the row readings are stored in registers KPAS and KPASMKPx. No interrupt is generated, since user software initiates the scan.

18.4.2 Direct Keypad Interface

The direct-key interface receives eight input signals (KP_DKIN<7:0>) from direct keys and rotary encoders and stores them in the Keypad Interface Direct Key register (KPDK). When the direct-key interface detects debounced activity in the direct keys or the rotary encoders, it sets KPC[DI], which causes a keypad interrupt.

There are two options for the direct key and rotary encoder debounce intervals depending on the state of the direct keypad debounce select bit (DK_DEB_SEL) of the KPC register:

- Regular debounce interval—This is the default debounce interval. The direct-key debounce interval is the same as the matrix-key debounce interval field of the KPKDI register. The rotary-encoder debounce interval can be either the matrix-key debounce interval or zero, depending on the state of the rotary-encoder zero debounce select bit (RE_ZERO_DEB) of the KPC register.
- Direct-key debounce interval—The direct-key debounce interval is different from the matrix-key debounce interval and is specified in the direct-key debounce interval field of the KPKDI register. This is the debounce interval for the direct key and rotary encoder logic if the DK_DEB_SEL bit of the KPC register is set. The rotary-encoder debounce interval can be either the direct-key debounce interval or zero, depending on the state of the rotary-encoder zero debounce select bit (RE_ZERO_DEB) of the KPC register.

It is not necessary for every direct-key input to be connected to the keypad interface if the corresponding GPIO pins are being used as inputs or outputs for other blocks. As an example, if GPIO pins corresponding to direct-key inputs 2 and 3 are being used as inputs/outputs for other blocks, they would be unavailable for the keypad interface. In such a case, the KP_DKIN<3:2> input signals are guaranteed a logic 0 at all times, which means that no activity is detected on direct keys 2 and 3. The remaining direct-key inputs (KP_DKIN<1:0> and KP_DKIN<7:4>) are utilized by software and are connected to either rotary encoders or direct keys, by specifying the number of direct keys in the Keypad Control register as eight (KPC[DKN] = 0b111). Although the direct-key inputs 2 and 3 are used for other applications, the keypad interface always detects no activity on them.

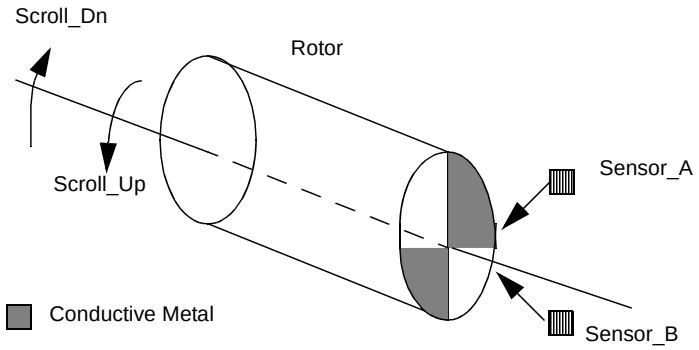
18.4.2.1 Direct Keys

The direct-key interface reads up to eight direct keys and stores them in KPDK[7:0]. Power and reset keys can be implemented as direct keys.

18.4.2.2 Rotary Encoders

A rotary encoder consists of a cylindrical rotor with metal strips and a pair of sensors, as shown in Figure 18-2. It can implement items such as scroll keys, jog-dials, and thumbwheels. The direct-key interface supports up to two rotary encoders and stores the sensor outputs in KPDK[3:0].

Figure 18-2. Rotary Encoder



Rotary Encoder

When the encoder is rotated, the sensors send logic highs or logic lows, depending on whether they sense the metal strip. The direct-key interface incorporates an up/down counter to determine the amount of scroll. Table 18-2 shows the effect of rotary-encoder sensor outputs on count values.

Table 18-2. Effect of Rotary-Encoder Sensor Outputs on Count Value

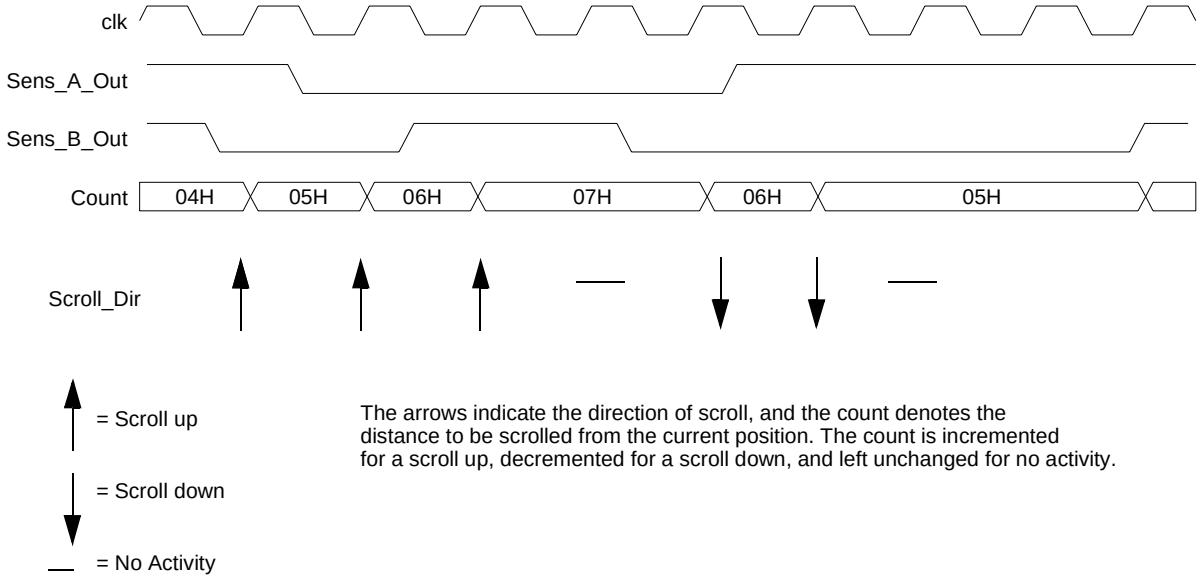
Previous A	Previous B	Current A	Current B	Effect
L	L	L	H	Count Up
L	L	H	L	Count Dn
L	H	L	L	Count Dn
L	H	H	H	Count Up
H	L	L	L	Count Up
H	L	H	H	Count Dn
H	H	L	H	Count Dn
H	H	H	L	Count Up
All Other Combinations				No effect

Figure 18-3 illustrates the waveforms generated by the rotary-encoder sensors. The sensor output determines the scroll direction. At every rising edge of the keypad clock, the rotary-encoder counter increments, decrements, or remains unchanged, depending on the scroll direction. The Rotary Encoder Count register (KPREC) stores the following items for each of the two encoders:

- Count value
- Underflow (count value < 0, the minimum possible value)
- Overflow (count value > 255, the maximum possible value)

The change in these values over any period of time determines the direction and magnitude of scroll during that period.

Figure 18-3. Waveforms Illustrating Operation of Rotary Encoder



18.4.3 Debounce Check

The debounce interval for the matrix and direct keys is specified in the Keypad Key Debounce Interval register, KPKDI[Interval]. To disable debounce checking, clear all bits in the Interval field.

The input signals, direct or matrix, are considered debounced if they remain stable for the debounce interval specified in KPKDI[Interval]. The optimum debounce interval depends on the type of keypad and the scan mode. For detailed information on the intervals available, see Section 18.5.7.

The following debouncing procedures, described here to clarify operation, occur transparently. No action is required by user software.

Matrix Keypad, Manual Scan Procedure

In manual matrix scan, only the keypress or no-keypress information is detected and stored. The user software must scan the entire keypad to determine the individual states of all keys.

1. Read the matrix keypad inputs; determine if any key(s) or no keys were pressed.
2. Wait one debounce interval.
3. Read the matrix keypad inputs again; determine if any key(s) or no keys were pressed.
4. If the results from the two successive reads match, the keys have debounced. Otherwise, repeat steps 2 and 3 until two successive reads match.

Matrix Keypad, Automatic Scan Procedure

The debounce check occurs only for automatic scans initiated by keypad activity (KPC[ASACT] set), **not** for automatic scans initiated by setting KPC[AS].

1. Automatically conduct a full scan of the matrix keypad to obtain the state of all keys.

2. Automatically wait one debounce interval.
3. Automatically conduct another full scan of the matrix keypad to obtain the state of all keys.
4. If the results from the two successive scans match, the keys have debounced. Otherwise, automatically repeat steps 2 and 3 until two successive scans match.

Direct Keypad Procedure

1. Read the direct keypad inputs.
2. Wait one debounce interval.
3. Read the direct keypad inputs again.
4. If the results from the two successive reads match, the keys have debounced. Otherwise, repeat steps 2 and 3 until two successive reads match.

18.4.4 Interrupt Generation

The keypad interface interrupt is generated by merging the matrix interrupt signal (KPC[MI] and the direct interrupt signal (KPC[DI]. Use the interrupt-enable bits KPC[MIE] (matrix) and KPC[DIE] (direct) to turn each type of interrupt on or off.

The following referenced sections describe interrupt generation:

Section 18.4.1.1—Manual Matrix Scan
Section 18.4.1.2—Automatic Matrix Scan Initiated by Keypad Activity
Section 18.4.1.3—Automatic Matrix Scan Initiated by Software
Section 18.4.2—Direct Keypad Interface

18.4.5 Entry Into or Exit from Standby or Sleep Mode

When keypad activity is enabled to cause exit from standby or sleep mode, it is important to assure that, between the time that a decision is made to enter standby or sleep mode and the time that the decision is executed, any keypad activity has first been determined to be completed. This is to avoid the undesirable case where abrupt entry into standby or sleep mode during keypad activity might miss a keypress, especially if the key is in the process of being debounced. Since the keypad controller runs at 32 KHz, determination that keypad activity has been completed can occur before the keypad controller is able to sense a new keypress.

The keypad controller itself must remain enabled (with all selected features remaining enabled) during entry into standby or sleep mode. This is to assure that keypad activity can successfully cause exit from standby or sleep mode. Scroll-wheel activity must not be used to cause exit from standby or sleep mode.

The example pseudocode describes a software procedure for entering and exiting standby or sleep mode, with particular emphasis on keypress activity. If a key-up event must be sensed, record all keypress interrupts in the associated ISR, and until the keypad-driver interrupt-service thread determines that it received a key-up event, block entry into standby or sleep mode.

Many operating systems typically disable interrupts during preparation for entry into sleep mode to prevent undesired side-effects as various peripherals are shut down. Accordingly, the pseudocode shows this disabling of interrupts as part of preparation for entry into sleep mode, but uses the term “probably” to indicate that disabling of interrupts for a particular peripheral is not absolutely

necessary if the user requires otherwise. On the other hand, many operating systems typically do not disable interrupts during preparation for entry into standby mode. Accordingly, the pseudocode shows that interrupts are not to be disabled as part of preparation for entry into standby mode, but uses the term “probably” to indicate that not disabling interrupts for a particular peripheral is not absolutely necessary if the user requires otherwise.

Pseudocode:

```
Begin low-power mode sequence
    Begin low-power mode preparation
        Prepare all devices for low-power mode entry
...
        For keypad controller:
            If MODE = sleep (probably) disable interrupts in ICMR
            Endif
            If MODE = standby (probably) do not disable interrupts in ICMR
            Endif
            Leave keypad controller fully configured and enabled

        Configure PMU registers for entry into standby mode or sleep mode

        Evaluate existing keypad activity:
            Read all GPLRs that report MKIN or MKOUT pins
            Keypad is active if any of the following are true:
                Any MKOUTs are low // indicates matrix scan in progress
                Any MKINs are high // indicates matrix key is being pressed
                Any DKIN not assigned to a rotary encoder is high
                        // direct key or button is being pressed
            End keypad activity determination

    End low-power mode preparation

    If keypad is not active, enter low-power mode
    Endif
    If keypad is active, do not enter (skip) low-power mode
    Endif

Record reason for exit from or skipping of low-power mode

Restore all peripherals after exit from low-power mode
    For keypad controller:
        If MODE = Standby, re-enable interrupts // restore when appropriate
                            // (OS-dependent) if any were optionally disabled
        Endif
        If MODE = sleep, OS-specific restoration // with possible modification if
                            // low-power mode was not entered (skipped).
...     Endif
End low-power mode sequence
```

18.5 Register Descriptions

The keypad interface has ten 32-bit registers, summarized in Table 18-14. Up to three keypad cycles are required for a value written to a keypad register from the peripheral interface to take effect. After a value has been written to a keypad register, do not write a new value to it until another six keypad clock cycles have elapsed.

18.5.1 Keypad Interface Control Register (KPC)

The Keypad Interface Control register specifies the keypad settings that allow independent enabling of the direct (DE set) and matrix (ME set) keypad interfaces. Setting or clearing the DIE and the MIE bits individually enables or disables interrupt generation for each of the keypads. Setting or clearing the rotary encoder enable bits REE0 and REE1 enables or disables the rotary encoders for the direct keypad. The MKRN and MKCN bits specify the number of rows and columns of the matrix keypad. DKN specifies the number of direct keys.

When not scanning the matrix keypad, set bits MS0–MS7. This assures that new keypad activity does not get missed. KPC[MS0–MS7] drive the KP_MKOUT<7:0> outputs at all times other than when an automatic matrix scan is being conducted. Software must set these bits to 0b1 at all times except when a manual matrix scan is being conducted. This ensures that all of the matrix keypad's column lines are activated and that activity in the matrix keypad does not go undetected. On detection of new activity, a new manual or automatic matrix scan can be initiated, depending on the state of KPC[ASACT].

Setting AS initiates an automatic scan of the matrix keypad. Setting ASACT and detecting keypad activity initiate an automatic scan of the matrix keypad. When the scan completes, MI is set, which asserts the keypad interrupt signal to the interrupt controller.

When ASACT is cleared, detecting keypad activity sets MI. An interrupt is sent to the interrupt controller, and an interrupt service routine can initiate a manual scan of the matrix keypad by setting the MS0-MS7 bits in the appropriate order.

If IMKP is set for the matrix-keypad interface, multiple keypresses are ignored. Only when all keys are released is a new scan initiated to detect activity (in case of automatic scan by keypad activity) or an interrupt generated (in case of manual scan).

Activity detected in the direct keypad or the rotary encoders sets DI. The keypad interrupt signal to the interrupt controller is asserted, the direct-key interrupt service routine is initiated, and the Direct Key register (KPDK) is read to determine the key(s) pressed. Table 18-3 shows the register organization and the individual bit definitions.

This is a read/write register. Ignore reads from reserved bits. Write reserved bits with zeros.

Table 18-3. KPC Bit Definitions

Physical Address 0x4150_0000 — KPC — Keypad Interface

Bit	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
User Settings																																
	reserved	AS	ASACT	MKRN			MKCN			MI	IMKP	MS7	MS6	MS5	MS4	MS3	MS2	MS1	MS0	ME	MIE	reserved	DK_DEB_SEL	DKN			DI	RE_ZERO_DEB	REE1	REE0	DE	DIE
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

Bits	Access	Name	Description
31	—	—	reserved
30	R/W	AS	Automatic Scan 0 = No effect 1 = Writes cause the keypad to be scanned *once*, and then the AS bit is reset to zero.
29	R/W	ASACT	Automatic Scan on Activity 0 = No automatic scan on activity. 1 = If KPC[AS] is clear, the keypad is scanned to detect the keypress whenever there is any keypad activity and the key is pressed for a duration longer than the key debounce interval specified by KPKDI[Interval].
28:26	R/W	MKRN	Matrix Keypad Row Number This octal value plus 1 indicates the number of rows in the matrix keypad. Examples: 000: Number of keypad rows = 1 111: Number of keypad rows = 8
25:23	R/W	MKCN	Matrix Keypad Column Number This octal value plus 1 indicates the number of columns in the matrix keypad. Examples: 000: Number of keypad columns = 1 111: Number of keypad columns = 8
22	R	MI	Matrix Keypad Interrupt This bit is reset when read. Writes to it are ignored.
21	R/W	IMKP	Ignore Multiple keypress (Matrix-Keypad Interface Only) 0 = In case of automatic scan by keypad activity and manual scan, do not ignore multiple keypresses. 1 = In case of automatic scan by keypad activity and manual scan, ignore multiple keypresses after a keypress activity has been detected and scanned.
20	R/W	MS7†	Manual Matrix Scan Line 7 Asserted to scan column 7 of the matrix keypad.
19	R/W	MS6†	Manual Matrix Scan Line 6 Asserted to scan column 6 of the matrix keypad.
18	R/W	MS5†	Manual Matrix Scan Line 5 Asserted to scan column 5 of the matrix keypad.

Table 18-3. KPC Bit Definitions

Physical Address 0x4150_0000 — **KPC** — **Keypad Interface**

User Settings																															
Bit	**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	AS	ASACT	MKRN			MKCN			MI	IMKP	MS7	MS6	MS5	MS4	MS3	MS2	MS1	MS0	ME	MIE	reserved	DK_DEB_SEL	DKN			DI	RE_ZERO_DEB	REE1	REE0	DE	DIE
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

Bits	Access	Name	Description
17	R/W	MS4†	Manual Matrix Scan Line 4 Asserted to scan column 4 of the matrix keypad.
16	R/W	MS3†	Manual Matrix Scan Line 3 Asserted to scan column 3 of the matrix keypad.
15	R/W	MS2†	Manual Matrix Scan Line 2 Asserted to scan column 2 of the matrix keypad.
14	R/W	MS1†	Manual Matrix Scan Line 1 Asserted to scan column 1 of the matrix keypad.
13	R/W	MS0†	Manual Matrix Scan Line 0 Asserted to scan column 0 of the matrix keypad.
12	R/W	ME	Matrix Keypad Enable 0 = Disabled 1 = Enabled
11	R/W	MIE	Matrix Interrupt Enable 0 = Disabled 1 = Enabled
10	—	—	reserved
9	R/W	DK_DEB_SEL	Direct Keypad Debounce Select 0 = Direct keypad debounce interval equal to the Matrix Key Debounce Interval Field of the KPKDI. 1 = Direct keypad debounce interval equal to the Direct Key Debounce Interval field of the KPKDI.
8:6	R/W	DKN	Direct Key Number This octal value + 1 indicates the number of keys in the direct keypad plus the number of rotary-encoder sensor inputs. Examples: 000: number of direct keys plus rotary-encoder sensor inputs = 1 111: number of direct keys plus rotary-encoder sensor inputs = 8
5	R	DI	Direct Keypad Interrupt This bit is reset when read. Writes to it are ignored.
4	R/W	RE_ZERO_DEB	Rotary Encoder Zero Debounce 0 = Rotary encoder logic debounce interval equal to the direct keypad debounce interval. 1 = Rotary encoder logic debounce interval equal to zero.

Table 18-3. KPC Bit Definitions

Physical Address 0x4150_0000													KPC										Keypad Interface									
User Settings																																
Bit	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	AS	ASACT	MKRN			MKCN			MI	IMKP	MS7	MS6	MS5	MS4	MS3	MS2	MS1	MS0	ME	MIE	reserved	DK_DEB_SEL	DKN			DI	RE_ZERO_DEB	REE1	REE0	DE	DIE
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

Bits	Access	Name	Description
3	R/W	REE1	Rotary Encoder 1 Enable 0 = Not enabled 1 = Enabled
2	R/W	REE0	Rotary Encoder 0 Enable 0 = Not enabled 1 = Enabled
1	R/W	DE	Direct Keypad Enable 0 = Disabled 1 = Enabled
0	R/W	DIE	Direct Keypad interrupt Enable 0 = Disabled 1 = Enabled
† All of the KPC[MS7:MS0] bits must be set simultaneously to detect matrix keypad activity correctly.			

18.5.2 Keypad Interface Direct Key Register (KPDK)

The Direct Key register, shown in Table 18-4, contains details of the last read of the direct-keypad inputs, if the direct-keypad is enabled (KPC[DE] set). The status of all the direct keys are stored in KPDK on each read of the direct keypad. This register is reset only on power-up.

This is a read-only register. Ignore reads from reserved bits.

Table 18-4. KPDK Bit Definitions

Physical Address 0x4150_0008 — KPDK — Keypad Interface

Bit	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
User Settings																																
	DKP	reserved																							DK7	DK6	DK5	DK4	RB1-DK3	RA1-DK2	RB0-DK1	RA0-DK0
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	RO	DKP	Direct Key Pressed Since Last Read Reset on register read.
30:8	—	—	reserved
7	RO	DK7	Direct Key 7 Input
6	RO	DK6	Direct Key 6 Input
5	RO	DK5	Direct Key 5 Input
4	RO	DK4	Direct Key 4 Input
3	RO	RB1-DK3	Rotary Encoder B/Direct Key 3 Input Rotary Encoder 1 disabled: input = direct key 3 Rotary Encoder 1 enabled: input = rotary encoder 1, sensor B
2	RO	RA1-DK2	Rotary Encoder A/Direct Key 2 Input Rotary Encoder 1 disabled: input = direct key 2 Rotary Encoder 1 enabled: input = rotary encoder 1, sensor A
1	RO	RB0-DK1	Rotary Encoder B/Direct Key 1 Input Rotary Encoder 0 disabled: input = direct key 1 Rotary Encoder 0 enabled: input = rotary encoder 0, sensor B
0	RO	RA0-DK0	Rotary Encoder A/Direct Key 0 Input Rotary Encoder 0 disabled: input = direct key 0 Rotary Encoder 0 enabled: input = rotary encoder 0, sensor A

18.5.3 Keypad Interface Rotary Encoder Count Register (KPREC)

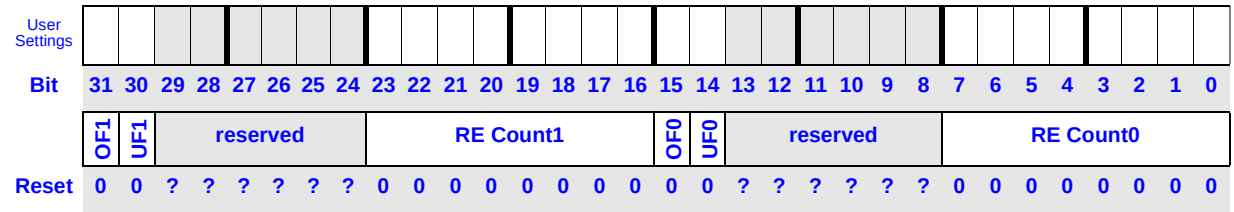
Table 18-5 shows the organization and bit definitions of the Keypad Interface Rotary Encoder Count register (KPREC), which stores the values of the counters pertaining to the respective rotary encoders. These count values serve as indications of any activity in the keys implemented using the rotary encoders. For example, with a scroll key implemented using a rotary encoder, any activity increments or decrements the rotary encoder counter, depending on the direction of the scroll. This count value is stored in KPREC every 32 kHz keypad clock. Software must periodically read the count value in KPREC and compare it with the last value read to determine the direction and magnitude of the scroll.

The KPREC register can store up to two 8-bit count values with an overflow and an underflow bit corresponding to each of the two counts. The underflow bits UFx are set when the count goes below zero. The overflow bits OFx are set when it goes beyond the maximum 8-bit value of 255. The overflow and underflow bits are reset when KPREC is read.

This is a read/write register. Ignore reads from reserved bits. Write reserved bits with zeros.

Table 18-5. KPREC Bit Definitions

Physical Address 0x4150_0010	KPREC	Keypad Interface



Bits	Access	Name	Description
31	R/W	OF1	Overflow for Rotary Encoder 1 Set when count value goes above the maximum 8-bit value (255). This bit is reset on read.
30	R/W	UF1	Underflow for Rotary Encoder 1 Set when count value goes below zero. This bit is reset on read.
29:24	—	—	reserved
23:16	R/W	RE Count1	Count Value for Rotary Encoder 1
15	R/W	OF0	Overflow for Rotary Encoder 0 Set when count value goes above the maximum 8-bit value (255). This bit is reset on read.
14	R/W	UF0	Underflow for Rotary Encoder 0 Set when count value goes below zero. This bit is reset on read.
13:8	—	—	reserved
7:0	R/W	RE Count0	Count Value for Rotary Encoder 0

18.5.4 Keypad Interface Matrix Key Register (KPMK)

When enabled (KPC[ME] set), the Matrix Key register contains the row details of the keys pressed on the matrix keypad when the last manual scan was completed. The MKP bit, if set, indicates that a matrix key was pressed since the register was last read. Table 18-6 shows the register organization and bit definitions.

This is a read-only register. Ignore reads from reserved bits.

Table 18-6. KPMK Bit Definitions

Physical Address 0x4150_0018 — **KPMK** — **Keypad Interface**

	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
User Settings																																
Bit	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
	MKP	reserved																							MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	RO	MKP	Matrix Key Pressed Since Last Read Reset on register read.
30:8	—	—	reserved
7	RO	MR7	Matrix Row 7
6	RO	MR6	Matrix Row 6
5	RO	MR5	Matrix Row 5
4	RO	MR4	Matrix Row 4
3	RO	MR3	Matrix Row 3
2	RO	MR2	Matrix Row 2
1	RO	MR1	Matrix Row 1
0	RO	MR0	Matrix Row 0

18.5.5 Keypad Interface Automatic Scan Register (KPAS)

The KPAS register, defined in Table 18-7, contains row and column details for a single keypress or information about multiple keypresses and invalid data.

Automatic scan of the keypad is initiated either when there is stable keypad activity for longer than the key debounce interval while KPC[ASACT] is set or when KPC[AS] is set.

If no key was pressed, the Multiple Keys Pressed (MUKP) field contains the value 00000.

If a single key was pressed, the row and column details of the key are stored in the RP and CP bit fields. The MUKP bit field contains the value 00001.

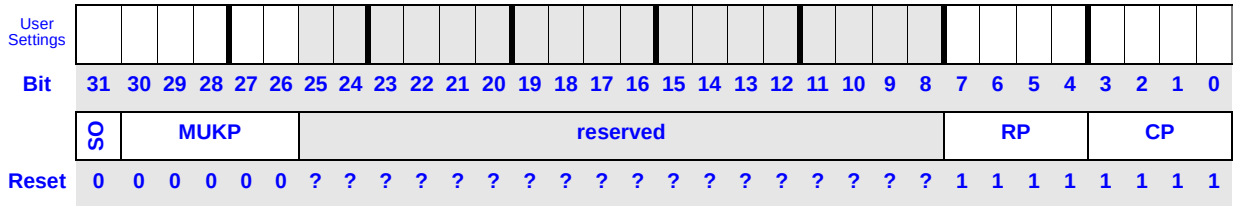
If multiple keys were pressed, the MUKP bit field contains a value greater than 00001, read the details about the keys pressed from the Keypad Interface Automatic Scan Multiple keypress registers KPASMKP0-3.

The Scan On (SO) bit, if set, indicates that a scan is in progress and that the data in this register is invalid.

This is a read-only register. Ignore reads from reserved bits.

Table 18-7. KPAS Bit Definitions

Physical Address 0x4150_0020 — **KPAS** — **Keypad Interface**



Bits	Access	Name	Description
31	RO	SO	Scan On This bit is set at the beginning of the automatic scan or automatic scan on activity and cleared when the scan is completed. When set, the data in this register is invalid.
30:26	RO	MUKP	Multiple Keys Pressed 00000 = No key pressed. 00001 = Single key pressed. Read row and column information from the RP and CP bit fields. >00001 = Multiple keys pressed. Read the keypress information from the KPASMKP0, KPASMKP1, KPASMKP2 and KPASMKP3 registers.
25:8	—	—	reserved
7:4	RO	RP	Row Pressed 1111 = Data invalid (also the reset value) 0000 = Key pressed is in row 0 ...continuing sequentially through... 0111 = Key pressed is in row 7
3:0	RO	CP	Column Pressed 1111 = Data invalid (also the reset value) 0000 = Key pressed is in column 0 ...continuing sequentially through... 0111 = Key pressed is in column 7

18.5.6 Keypad Interface Automatic Scan Multiple Keypress Registers 0–3 (KPASMKPx)

The Keypad Interface Automatic Scan Multiple keypress registers contain details of the row readings from automatic scans when multiple keys have been pressed. Each register deals with two keypad columns, as shown in Table 18-8.

Table 18-8. Keypad Columns Used by KPASMKPx Registers

Register	Definition Table	Keypad Column Numbers
KPASMKP0	Table 18-9	1–0
KPASMKP1	Table 18-10	3–2
KPASMKP2	Table 18-11	5–4
KPASMKP3	Table 18-12	7–6

Read these registers after:

- An automatic scan interrupt
- A scan initiated by setting KPC[AS] has completed.

Row readings are recorded in 8-bit MKCx bit fields, two per register, where x corresponds to the keypad column number. Each bit in the MKCx field corresponds to a row under the column, with bit 0 representing row 0 and bit 7 representing row 7. A set bit in MKCx indicates that a key has been pressed under column x. For example:

01000000 in KPASMKP0[MKC1] indicates that a key in row 6, column 1 has been pressed.

00000010 in KPASMKP3[MKC7] indicates that a key in row 1, column 7 has been pressed.

Each KPASMKPx register contains a Scan On (SO) bit, which is set on initiation of an automatic scan and cleared when the scan is completed. When SO is set, the data in the register is invalid.

These are read-only registers. Ignore reads from reserved bits.

Table 18-9. KPASMKP0 Bit Definitions

Physical Address 0x4150_0028	KPASMKP0	Keypad Interface

Bit	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
	SO	reserved							MKC1								reserved								MKC0							
Reset	0	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	RO	SO	Scan On This bit is set at the beginning of the automatic scan or automatic scan on activity and cleared when the scan is completed. When set, the data in this register is invalid.
30:24	—	—	reserved
23:16	RO	MKC1	Matrix Keypad Column 1 Reading A set bit identifies a key in the corresponding row and column 1: Bit 23 = row 7, sequentially through Bit 16 = row 0
15:8	—	—	reserved
7:0	RO	MKC0	Matrix Keypad Column 0 Reading A set bit identifies a key in the corresponding row and column 0. Bit 7 = row 7, sequentially through Bit 0 = row 0

Table 18-10. KPASMKP1 Bit Definitions

Physical Address 0x4150_0030	KPASMKP1	Keypad Interface

Bit	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
	SO	reserved							MKC3								reserved								MKC2							
Reset	0	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	RO	SO	Scan On This bit is set at the beginning of the automatic scan or automatic scan on activity and cleared when the scan is completed. When set, the data in this register is invalid.
30:24	—	—	reserved
23:16	RO	MKC3	Matrix Keypad Column 3 Reading A set bit identifies a key in the corresponding row and column 3: Bit 23 = row 7, sequentially through Bit 16 = row 0
15:8	—	—	reserved
7:0	RO	MKC2	Matrix Keypad Column 2 Reading A set bit identifies a key in the corresponding row and column 2: Bit 7 = row 7, sequentially through Bit 0 = row 0

Table 18-11. KPASMKP2 Bit Definitions

Physical Address 0x4150_0038	KPASMKP2	Keypad Interface

Bit	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
	SO	reserved							MKC5								reserved								MKC4							
Reset	0	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	RO	SO	Scan On This bit is set at the beginning of the automatic scan or automatic scan on activity and cleared when the scan is completed. When set, the data in this register is invalid.
30:24	—	—	reserved
23:16	RO	MKC5	Matrix Keypad Column 5 Reading A set bit identifies a key in the corresponding row and column 5: Bit 23 = row 7, sequentially through Bit 16 = row 0
15:8	—	—	reserved
7:0	RO	MKC4	Matrix Keypad Column 4 Reading A set bit identifies a key in the corresponding row and column 4: Bit 7 = row 7, sequentially through Bit 0 = row 0

Table 18-12. KPASMKP3 Bit Definitions

Physical Address 0x4150_0040	KPASMKP3	Keypad Interface

Bit	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
	SO	reserved							MKC7								reserved								MKC6							
Reset	0	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	RO	SO	Scan On This bit is set at the beginning of the automatic scan or automatic scan on activity and cleared when the scan is completed. When set, the data in this register is invalid.
30:24	—	—	reserved
23:16	RO	MKC7	Matrix Keypad Column 7 Reading A set bit identifies a key in the corresponding row and column 7: Bit 23 = row 7, sequentially through Bit 16 = row 0
15:8	—	—	reserved
7:0	RO	MKC6	Matrix Keypad Column 6 Reading A set bit identifies a key in the corresponding row and column 6: Bit 7 = row 7, sequentially through Bit 0 = row 0

18.5.7 Keypad Interface Key Debounce Interval Register (KPKDI)

The signals generated by a key do not stabilize for some tens of microseconds after the key is pressed. Reading those signals before the key signals stabilize could lead to faulty detection. To avoid such a condition, a time period known as a *key debounce interval* must elapse before any reading is performed to determine the key pressed.

The key debounce interval is the time interval between the time keypad activity is first detected and the time the key value is stored in the appropriate register (Matrix Key register, Direct Key register, or an Automatic Scan register). This interval is specified in the Key Debounce Interval register (KPKDI), defined in Table 18-13. The key debounce interval is specified as an 8-bit number in the Interval field. The unit for the debounce interval is ms (the interval is typically between 32 ms and 128 ms). The Interval field defaults to 100 ms upon reset. To read the key(s) pressed without waiting for the debounce interval, set the interval value in the KPKDI register to zero.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 18-13. KPKDI Bit Definitions

Physical Address 0x4150_0048																KPKDI								Keypad Interface							
User Settings																															
Bit	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																Direct-Key Debounce Interval								Matrix-Key Debounce Interval							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15:8	R/W	Direct-Key Debounce Interval	Direct-Key Debounce Interval (in ms)
7:0	R/W	Matrix-Key Debounce Interval	Matrix-Key Debounce Interval (in ms)

18.6 Register Summary

Table 18-14 shows the registers associated with the keypad interface and the physical addresses to access them. These registers can be accessed only with word accesses. They are grouped together within one page, and all have the same memory protections.

Table 18-14. Keypad Interface Register Summary

Address	Name	Description	Page
0x4150_0000	KPC	Keypad Interface Control register	18-12
0x4150_0004	—	reserved	—
0x4150_0008	KPDK	Keypad Interface Direct Key register	18-16
0x4150_000C	—	reserved	—
0x4150_0010	KPREC	Keypad Interface Rotary Encoder Count register	18-17
0x4150_0014	—	reserved	—
0x4150_0018	KPMK	Keypad Interface Matrix Key register	18-18
0x4150_001C	—	reserved	—
0x4150_0020	KPAS	Keypad Interface Automatic Scan register	18-18
0x4150_0024	—	reserved	—
0x4150_0028	KPASMKP0	Keypad Interface Automatic Scan Multiple Keypress register 0	18-20
0x4150_002C	—	reserved	—
0x4150_0030	KPASMKP1	Keypad Interface Automatic Scan Multiple Keypress register 1	18-20
0x4150_0034	—	reserved	—
0x4150_0038	KPASMKP2	Keypad Interface Automatic Scan Multiple Keypress register 2	18-20
0x4150_003C	—	reserved	—
0x4150_0040	KPASMKP3	Keypad Interface Automatic Scan Multiple Keypress register 3	18-20
0x4150_0044	—	reserved	—
0x4150_0048	KPKDI	Keypad Interface Key Debounce Interval register	18-23
0x4150_004C–0x415F_FFFC	—	reserved	—

Universal Subscriber ID Interface 19

This chapter describes the Universal Subscriber Identity Module (USIM) interface block and the USIM interface-related registers supported by the PXA27x processor.

19.1 Overview

The USIM interface is a primary device and communications interface for a GSM mobile handset. The USIM interface supports communication with smart cards as specified in the standard *ISO 7816-3* and technical specification 3G *TS 31.101* of the 3rd Generation Partnership Project.

Today, smart cards are used in many applications, and the GSM network USIM card is only one of many applications. Smart cards usually consist of CPU, flash memory, and a serial-communications interface device similar to the one described in this chapter. More sophisticated cards contain PLL for frequency enhancement. Encryption accelerators can also exist in a smart card, because many of their applications are security oriented. In all smart card applications, the physical layer and data link layer are identical, so this module may serve them as well. Familiarity with the above standards will help readers better understand this chapter.

Software controls the session between the USIM interface and the card by updating the USIM interface registers. Choosing protocol type and parameters, receiving or sending a byte to/from the card, activating/deactivating the card, setting transmitter/receiver baud rates, and similar operations are accomplished with read/write operations to the USIM interface registers. Transforming byte convention (inverse to direct and vice-versa, according to the session convention) is performed within the USIM interface. Hence, software does not have to perform format inversion before character receipt. The USIM interface provides functionality to support the above standards, but it is the responsibility of software to ensure the standards are met.

19.2 Features

The USIM interface has the following key features:

- Compatible with any USIM card that is compliant with standard *ISO 7816-3* and *3G TS 31.101* and operates in voltages of 1.8 V or 3 V
- Supports control lines for two-level voltage supply (1.8 V and 3 V)
- Supports USIM card reset pin control (using reset pin control and power supply control, warm/cold reset can be software-initiated)
- Supports T = 0 and T = 1 protocols
- Programmable card clock frequency
- Supports any combination of the following clock-rate conversion factor *F*, and bit-rate adjustment factor *D*:
  - *F* = {372, 512, 558}
  - *D* = {1,2,4,8,16, 32, 12, 20}

- Auto-error signal in T = 0 receive mode
- Auto-character repeat in T = 0 transmit mode
- Transforms inverted format to regular format and vice-versa
- Programmable block guard time period
- Programmable extra guard time period
- Programmable character waiting time period
- Programmable block waiting time period
- Programmable time-out period
- Programmable CPU interrupt on an error-signal detection
- Programmable CPU interrupt when a smart card is connected

19.3 Signal Descriptions

This section describes the I/O signals used by the USIM interface. Table 19-1 lists all processor pins that connect between the USIM interface and the USIM card.

Table 19-1. USIM Interface I/O Signal Descriptions (Sheet 1 of 2)

Signal Name	Type	Description†††
UIO	Bidirectional	USIM I/O Data. Receive and transmit data connection. The bidirectional pad is connected directly to the off-chip USIM card. When asserted, the I/O line is forced to V_{LOW}. When deasserted, the I/O line is pulled-up by a $20K\Omega$ pull-up resistor. (If the USIM function is not used, the pull-up resistor is not needed, decreasing power consumption in Standby mode.) The USIM card can also act as a pull-down on the I/O line. RXD is an input to the USIM interface logic and an output from the processor UIO pad. RXD reflects the status of the I/O to the USIM interface. **NOTE:** When the USIM card and the processor operate at different voltage levels, the I/O voltage level of the USIM card must be at a voltage level the PXA27x processor supports.
UVS0†	Output	Select for power transistor controlling voltage level supplied for card. Selects zero voltage on the USIM card power-supply pad (VCC).
nUVS1†	Output	Select for power transistor controlling voltage level supplied for card. Selects 1.8 V on the USIM card power-supply pad (VCC).
nUVS2†	Output	Input for power transistor controlling voltage level supplied for card. Selects 3.0 V on the USIM card power-supply pad (VCC).
UCLK	Output	Clock supplied to the card. This pin connects directly to the card clock pin. The card cannot use any other clock
nURST	Output	Card reset pin. Connects to card reset pin. The card is reset when this output is asserted.

Table 19-1. USIM Interface I/O Signal Descriptions (Sheet 2 of 2)

Signal Name	Type	Description†††
UDET††	Output	USIM detection for card present
UEN††	Output	USIM enable for VCC_USIM connection

NOTES:

† Care must be taken when placing the PXA27x processor into deep-sleep low-power mode. The voltage control pins are powered from the VCC_IO power domain, and in deep-sleep mode, this is turned off. Powering off the voltage select pins (UVS0, nUVS1, nUVS2) can cause shorting of the power supplies that are switched to provide power to the card. To prevent these electrical problems:

- The external power supply must be turned off before (or at the same time as) the processor enters deep-sleep power mode using USIM[USCCR] bit field.
- GPIOs must be correctly configured prior to entering deep-sleep mode with software writes to USIM[USCCR]. See Chapter 24, "General-Purpose I/O Controller" for more details.

†† The control and status of this signal is defined using the Section 3.8.1.14, "Power Manager USIM Card Control/Status Register (PUCR)" on page 3-90.

†††The voltage level of the pads is set off-chip. The pads work at either 1.8 V or 3.0 V.

19.4 Operation

The following sections describes the connection to the USIM card connect, coding conversion, protocols, clock control, card management, and FIFO operation.

19.4.1 USIM Card Interface

Actual connection to the USIM card is buffered by standard analog circuits. The card itself has only five I/O pads (listed in Table 19-2).

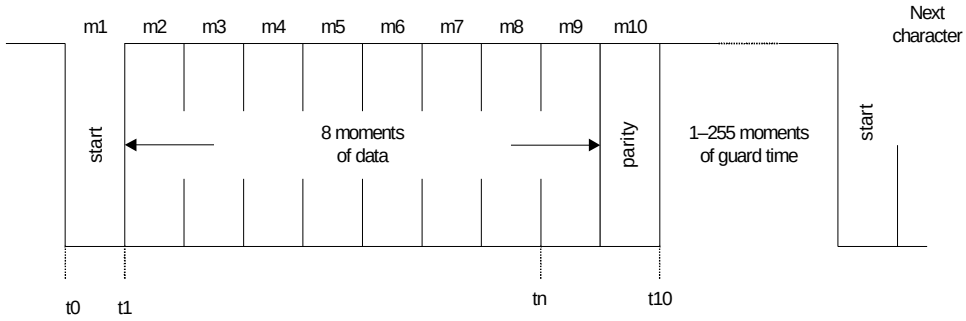
Table 19-2. USIM Card Pinout

Card Pin	Function	Pin Direction	Processor Pin
I/O	Data input/output (bidirectional) pad with a pull-up transistor. See *ISO 7816-3* for pad specifications.	USIM card <--> Processor	UIO
CARD_RST	USIM card reset	Processor --> USIM card	nURST
CLK_CARD	Clock input. Frequencies are between 1–5 MHz. See *ISO 7816-3* for pad specifications. Note that cards manufactured before April 2000 may have frequency limitation of 4 MHz. Clock-stops on low or high phase are supported.	Processor --> USIM card	UCLK
VCC	Card power supply. Supplies 0 V, 1.8 V, 3 V with max currents 0 mA, 30 mA, 50 mA, respectively according to card class (B, C). See *ISO 7816-3* for pad specifications.	Power supply --> Card	VCC_USIM
GND	Mutual USIM card, SIM interface, and VCC ground reference voltage.	Power supply --> Card	VSS_IO

19.4.2 Coding Conventions

The USIM interface performs serial-to-parallel conversion with optional format inversion on data characters received from the USIM card, and parallel-to-serial conversion with optional format inversion on data characters received from the processor. Every byte sent or received consists of a start bit followed by an 8-bit data string, parity bit, and guard time. The period of time devoted for transmission/reception of a single bit is called an *elementary time unit (etu)* or *moment.*

Figure 19-1. Byte Sent Versus Time



NOTE: m1,m2...m10 stands for moments 1–10.

The start bit is mandatory, as is the first guard bit. Extra guard time is optional. The USIM interface enables software to add up to 255 extra guard-time moments. See Section 19.5.11.

Data bits may appear in two possible conventions:

- **Direct Convention**—Data bits sent on moments m2–m9 (see Figure 19-1) are transmitted LSB first-MSB last (the least significant bit is transmitted on m2 and the most significant bit on m9). During moments m2–m10 (data and parity bits), transmission of V_{HIGH} is interpreted as a logic 1, while transmission of V_{LOW} is interpreted as a logic 0.
- **Inverse Convention**—Data bits sent on moments m2–m9 (see Figure 19-1) are transmitted MSB first-LSB last (the most significant bit is transmitted on m2 and the least significant bit on m9). During moments m2–m10 (data and parity bits), transmission of V_{LOW} is interpreted as a logic 1, while transmission of V_{HIGH} is interpreted as a logic 0.

The first byte sent from the card to the USIM interface after reset deassertion determines the convention to be used throughout the whole session. A card that encodes/decodes data in the direct convention sends 0x3B. A card that is working in inverse convention sends 0x3F.

When receiving this first character, one of the following scenarios is possible:

- Both the USIM interface and card are set to the same convention, and the first character is received correctly (received data was either 0x3F or 0x3B with no parity error).
- USIM interface and card are set to opposite conventions—the USIM interface asserts a parity error because it expects the opposite parity bit convention.
- Some other communication error occurred (for example, parity error due to thermal noise, wrong voltage level supplied to card).

19.4.3 Protocols

Both T = 0 and T = 1 protocols, as defined in the standards *ISO 7816-3* and *3G TS 31.101*, are supported by the USIM interface. The protocols can be set in the USIM Line Control register (LCR) (Section 19.5.8). The USIM interface does not analyze/generate data content or block structure sent in the protocols. It is the responsibility of software to generate and analyze command headers and block structure to decode data meaning. The protocols begin after either the Answer-To-Rest (ATR) or a successful Protocol and Parameter Selection (PPS) exchange. See the specifications for a more detailed explanation of the protocols.

In the T = 0 protocol, the USIM interface initiates every command with software writes to the transmit FIFO (TX-FIFO) containing a five-byte command header. The header tells the card what to do. The software-controlled command processing continues with the transfer of a variable number of data bytes in one direction under the control of procedure bytes sent by the card. See Figure 19-2. The T = 0 protocol allows for character retransmission when errors are detected. The protocol also requires two extra guard-time moments to follow the parity bit.

In the T = 1 protocol, data is sent using a sequence of bytes known as a *block*. By analyzing block content, software knows at any given moment the proper direction of data flow (transmit/receive). For example, software knows the length of the block sent in a T = 1 protocol from the card to the USIM interface by analyzing data in the third byte of the sent block (marked as a LEN byte). Note that each character has a special meaning according to its position in the block. See Figure 19-3. There is no character retransmission in the T = 1 protocol. Errors must be handled by software. The T = 1 protocol requires one extra guard-time moment following the *parity* bit. The USIM interface device supports DMA transfers that can be used in long T = 1 protocol block transfers.

Receive and transmit activity for each mode can be monitored in one of three possible ways: interrupt, polled, or DMA mode. See Section 19.4.6 for more details.

Protocol switching can occur during the ATR, immediately after the ATR, or after a successful PPS. Unpredictable results may occur when changing the protocol under other scenarios.

Figure 19-2. T = 0 Protocol Communications Method

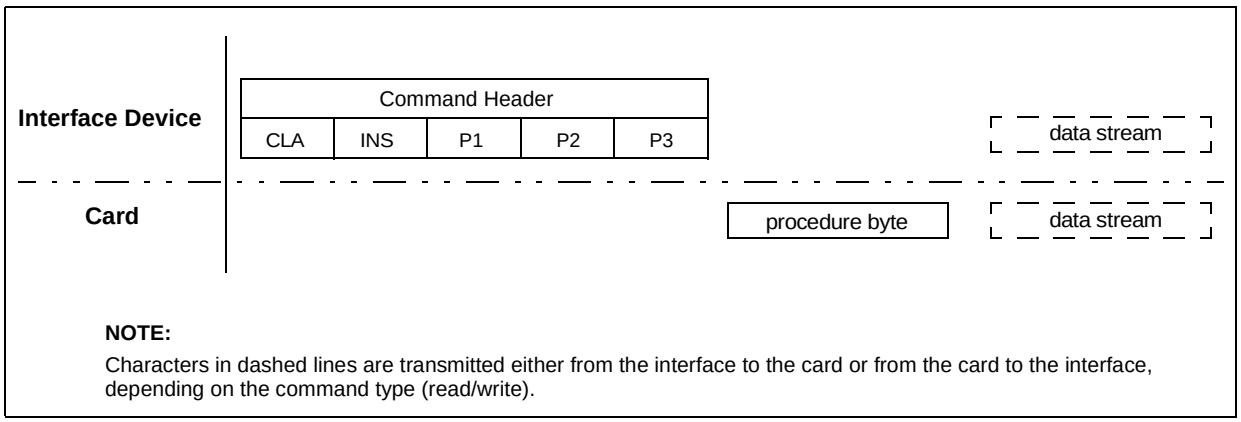
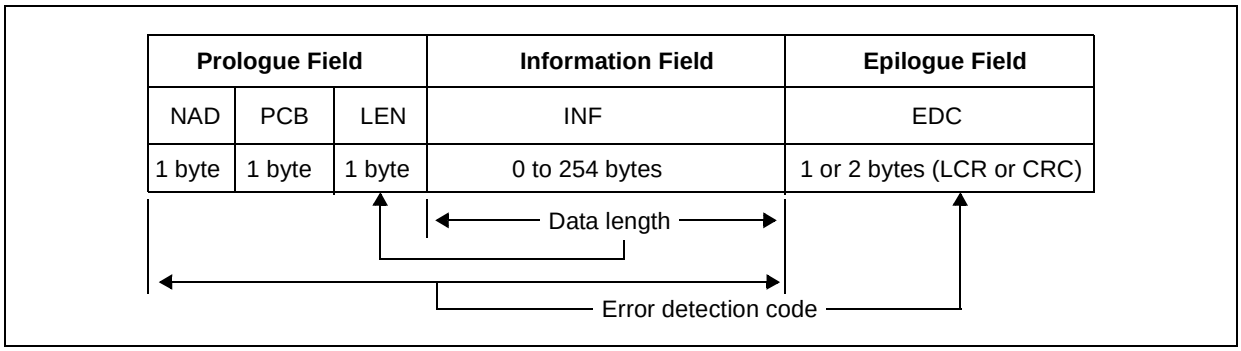


Figure 19-3. Complete Block Structure in T = 1 Protocol



19.4.3.1 Errors

The USIM Error Control register (ECR) (Section 19.5.7) and the USIM Interrupt Enable register (IER) (Section 19.5.3) allow software to control how errors are handled.

19.4.3.1.1 T = 0 Error

The appearance of an error signal during transmission in T = 0 mode is referred to as a T = 0 error. While transmitting a character, the USIM interface senses for an error signal V_{LOW} during the second mandatory guard-time moment, on the I/O line. In the event an error signal is identified, the USIM interface transmitter repeats the last byte transmitted. When the number of repetitions meets the T = 0 error trigger level ECR[T0ERR_TL], transmission stops, an interrupt occurs if enabled, and the LSR[T0ERR] bit is asserted. Software can then decide whether to proceed with the next byte or re-transmit the same byte using the control bits ECR[T0_CLR] and ECR[T0_REPEAT], respectively. In T = 1 mode, there is no repetition of data, so this interrupt is not applicable.

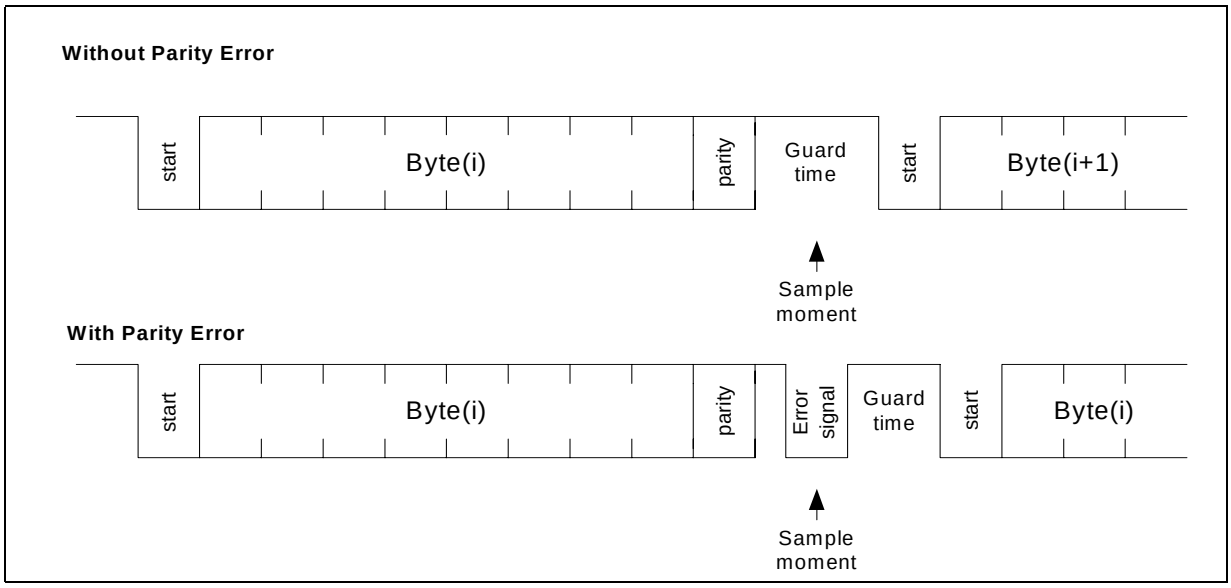
19.4.3.1.2 Parity Error

In T = 0 mode, the USIM interface receiver automatically signals the card in the event of a parity error using the I/O line with a V_{LOW} during guard time, see Figure 19-4. The erroneous byte does not enter the receive FIFO (RX-FIFO). Retransmission by the card of the previous byte is expected so no data is lost. When the number of repetitions meets the Parity Error trigger level ECR[PE_TL] an interrupt occurs (if enabled) and the LSR[PERR] bit is asserted. A parity error generates an error signal to the card, regardless of the error trigger value.

Note: Reception does not stop after receiving a parity error interrupt. If the parity error continues to repeat, the USIM interface continues to signal the card on each occurrence. Software can either reset the card or receive the character with a parity error by switching to T = 1 mode and then reverting back to T = 0 mode. Software can check the RBR[PERR] bit to validate the character was received with parity from the T = 1 mode.

In the T = 1 protocol, error signaling or sensing is not performed. When a parity error occurs, the erroneous byte enters the RX-FIFO, the LSR[PERR] bit is asserted, and a parity error interrupt occurs if enabled. The parity error trigger level has no effect in T = 1 mode. The erroneous bytes are indicated in the RX-FIFO by the assertion of RBR[PERR]. See Section 19.5.1. The total number of bytes with parity errors in the RX-FIFO can be determined by FSR[PERR_NUM].

Figure 19-4. T = 0 Character Transmission Format



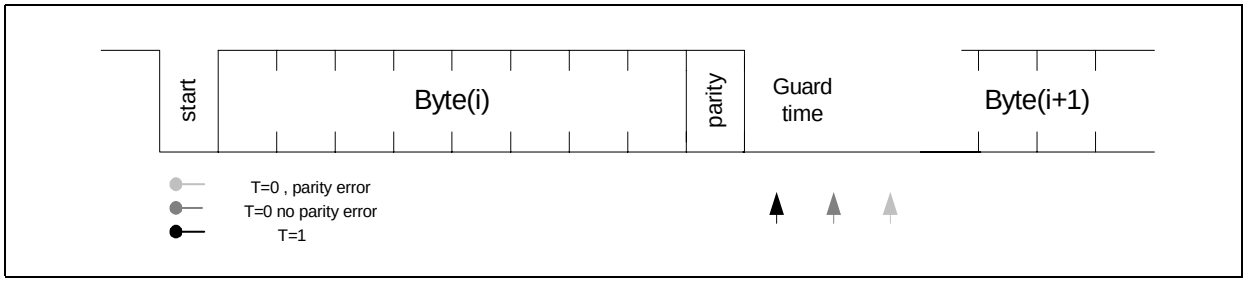
19.4.3.1.3 Overrun Error

The receive FIFO can store up to 16 bytes. If the receive FIFO is not emptied before the 17th byte is transmitted by the card, the LSR[OVRN] bit is asserted and an overrun interrupt is generated (if enabled), and the 17th byte does not enter the FIFO. In T = 0 mode, the USIM interface signals the card about the error using the I/O line with a V_{LOW} during guard time, which causes the card to retransmit the data.

19.4.3.1.4 Framing Error

After a byte is received, the USIM interface verifies that during the last moment of mandatory guard time the I/O line is kept at V_{HIGH}. If the I/O line drops to V_{LOW}, the LSR[FRAMERR] bit asserts and an interrupt (if enabled) occurs. See Figure 19-5.

Figure 19-5. Framing Error



19.4.3.2 Waiting Times

Waiting Times are protocol defined parameters from standard *ISO 7816-3* and specification *3G TS 31.101*. These parameters can be set in the USIM interface registers. Software must set the waiting times before changing the baud rate, see Section 19.4.4.2, to avoid unpredictable results.

19.4.3.2.1 Character Waiting Time (CWT)

In the T = 0 and T = 1 protocols, the number of moments separating consecutive characters in the same block must not exceed the Character Waiting Time (CWT). The ISO standard defines CWT as Work Waiting Time (WWT) for the T = 0 protocol. The USIM interface measures the CWT differently from the way it is defined in the ISO standard. In T = 0 mode, CWT is measured 12 etu after the start bit. In T = 1 mode, the CWT is measured 11 etu after the start bit. See Section 19-22 and Figure 19-6.

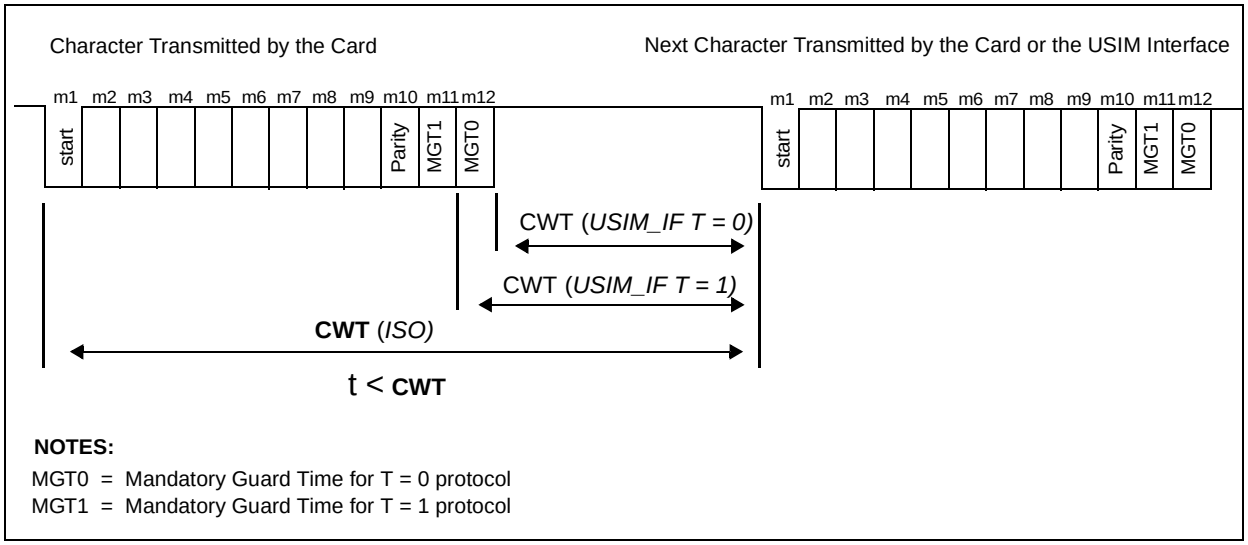
As shown in Figure 19-6, CWT does not measure the time between two characters transmitted by the USIM interface. It is software's responsibility to ensure that data is delivered continuously and that the upper software layers (such as data and application) are standard-compliant.

Also, if the CWT is less than the Block Guard Time (BGT) (Section 19.4.3.2.2), a CWT violation occurs each time the USIM interface transmits data after the card has finished transmitting. This is not an issue for the T = 0 protocol because the ISO specification defines a WWT/CWT of 9600 etu. For the T = 1 protocol, this may be an issue because the 3G specification defines 12etu<CWT<43etu. To avoid this, software must ignore CWT violations during transmission by the USIM interface.

Measurement of the time-out commences after reception of a character is complete. It stops with the renewal of transmit or receive activity before the time-out period has expired or when the time-out expires. Upon expiration of the time-out, the LSR[CWT] bit asserts and a CWT interrupt occurs, if enabled.

When using the CWT interrupt in receive mode, software can identify either the end of a block or lose of synchronization in communication (software expects more characters to come after the card finishes transmitting). By using a CWT interrupt in transmit mode, software can monitor its performance, verifying that the transmission was not stopped for more than CWT etu.

Figure 19-6. Character Waiting Time



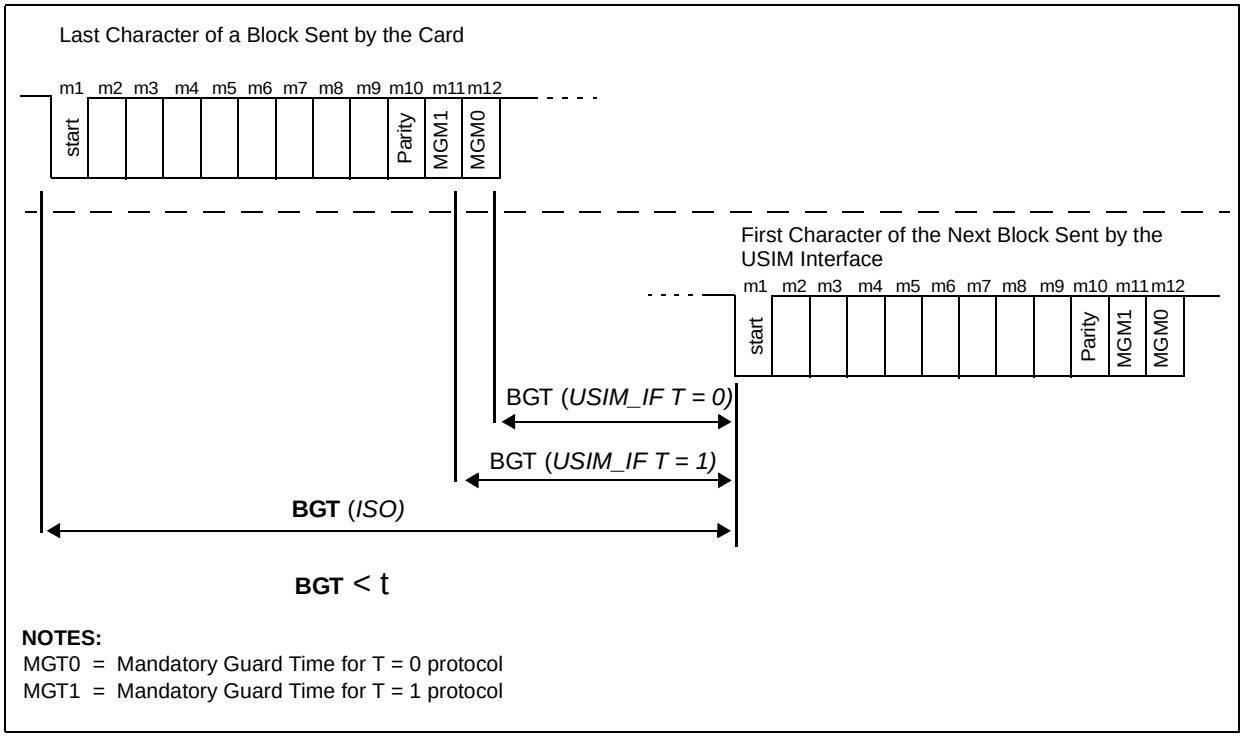
19.4.3.2.2 Block Guard Time (BGT)

The T = 0 and T = 1 protocols require different guard times between bytes transmitted in opposite directions. This guard time is called Block Guard Time (BGT). The USIM interface enables setting any block guard time to between 0–255 moments. The USIM interface automatically ensures BGT

is kept before transmission is initiated. The USIM interface is ready to receive a character immediately after transmission even when the card does not obey BGT. The USIM interface measures the BGT differently from the way it is defined in the ISO standard. The overall result is the same. In T = 0 mode, BGT is measured 12 etu after the start bit. In T = 1 mode, the BGT is measured 11 etu after the start bit. See Section 19.5.12 and Figure 19-7.

When receiving a character, the USIM interface does not initiate an access to the bidirectional I/O line until the character is fully received and the BGT is finished (even if the TX-FIFO is full). To start transmitting on the I/O line, software must write at least 1 byte to the 16-byte transmit FIFO that holds data from the processor to be transmitted on the serial link. As soon as the BGT is finished, the USIM interface starts transmitting. Transmission continues until the TX-FIFO is emptied or the FCR[TX_HOLD] bit is asserted.

Figure 19-7. Block Guard Time

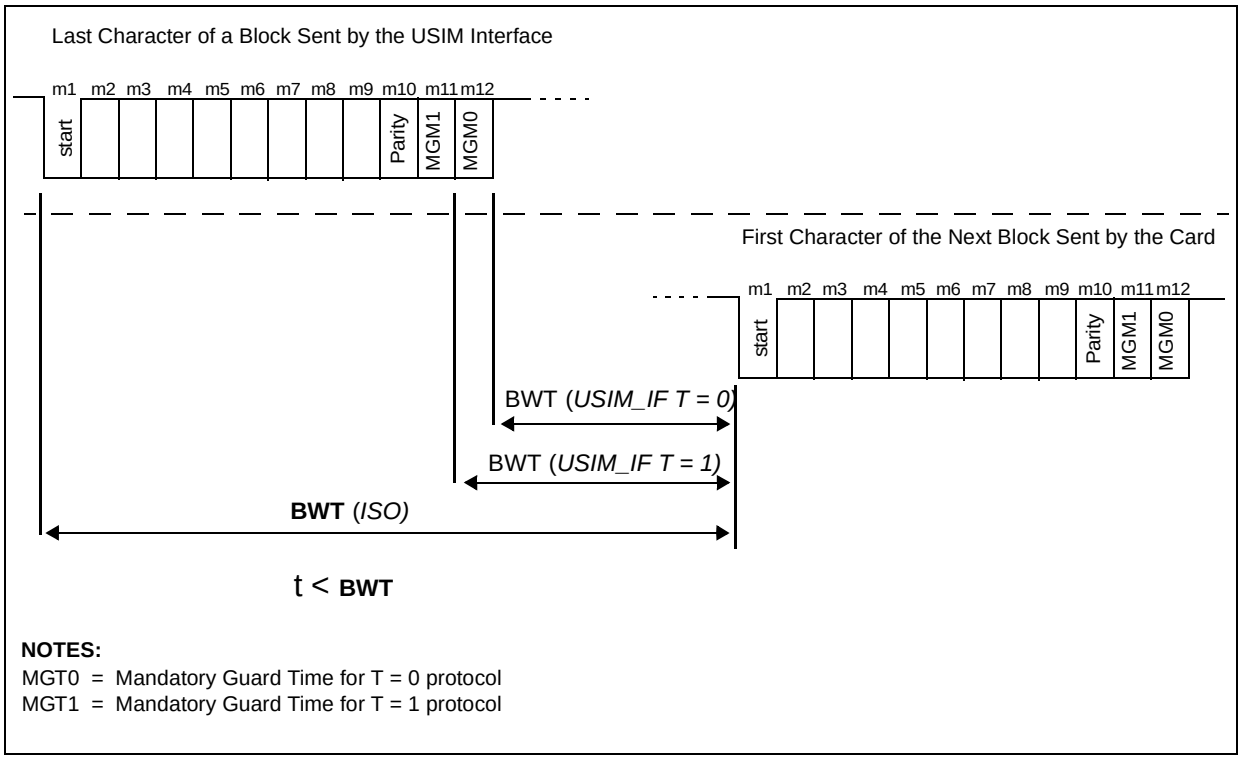


19.4.3.2.3 Block Waiting Time (BWT)

When the card is expected to reply, it must do so no later than the Block Waiting Time (BWT) period. The USIM interface measures the BWT differently from the way it is defined in the ISO standard. The overall result is the same. In T = 0 mode, BWT is measured 12 etu after the start bit. In T = 1 mode, BWT is measured 11 etu after the start bit. See Section 19.5.18 and Figure 19-8.

Measurement of the time-out commences after a card reset or the end of a transmission. It stops with the renewal of transmit or receive activity before the time-out period has expired or when the time-out period expires. Upon expiration of the time-out, the LSR[BWT] bit asserts and a BWT interrupt occurs, if enabled. BWT is useful in detecting an unresponsive card.

Figure 19-8. Block Waiting Time



19.4.4 Clock Control

The USIM _IF generates and controls the clock supplied to the card, UCLK, based on the values in the USIM Clock register (CLKR) (Section 19.5.14). UCLK is defined in Equation 19-1. The frequency of the clock driving the USIM internal logic is 48 MHz, as noted in Equation 19-2.

Equation 19-1. USIM Card Clock: $$UCLK = \frac{48\,MHz}{CLKR[DIVISOR] \times 2}$$

Equation 19-2. USIM Internal Clock: $$CLK\_USIM = 48MHZ$$

The specifications *ISO 7816-3* and *3G TS 31.101* limit the UCLK to a range of 1–4 MHz. UCLK is set to 4 MHZ after a system reset. When switching the card clock frequency, do so immediately after the Answer-To-Reset (ATR) or immediately after a successful Parameter and Protocol Selection (PPS). See the ISO and 3G specifications for more information on the ATR and PPS.

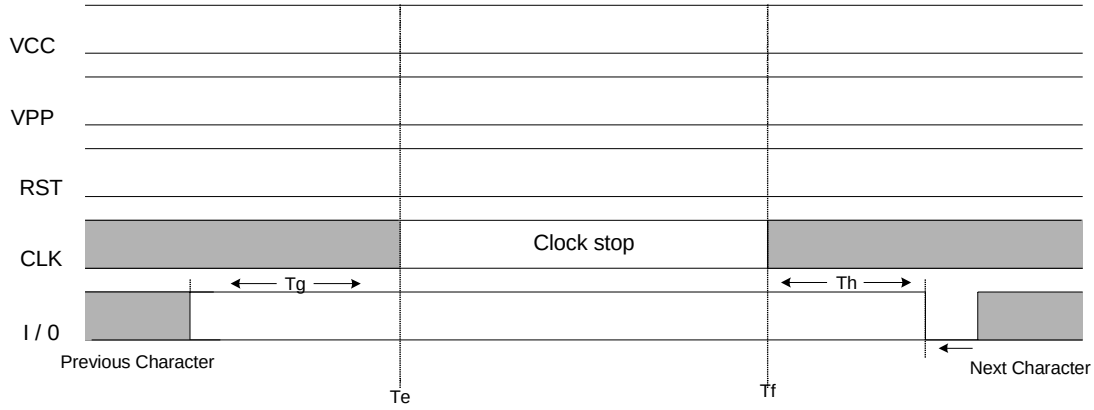
Changing the card clock affects the etu period. See Section 19.4.4.2. Software can write a new value to the CLKR only when CLKR[RQST] is deasserted. The CLKR does not update when CLKR[RQST] is asserted. CLKR[RQST] ensures the USIM interface has completed all pending transmission, reception, and block guard time before the card clock is changed.

19.4.4.1 Clock Stop

It is possible to stop the clock signal supplied to the card. The USIM interface enables the stop-clock signal either on V_{LOW} or V_{HIGH}. The voltage level of the clock signal during a stop period can be configured with the STOP_LEVEL bit of the USIM Clock register. Asserting the STOP_UCLK bit in the USIM Clock register stops the UCLK signal only after the USIM interface completes all pending transmission, reception, and block guard time. When deasserting the STOP_UCLK bit, the clock is reactivated. *ISO 7816-3* defines guard time periods before stopping the clock and after re-activating it, as specified in Figure 19-9.

For power savings, the USIM interface clock, CLK_USIM, can also be disabled by asserting bit CLKR[15]. After reset, CLK_USIM is enabled by default. The USIM interface registers can still be read/written when CLK_USIM is disabled.

Figure 19-9. Card Clock Stop



NOTE: Tg and Th have minimum values:
Tg > 1860 cycles
Th > 700 cycles

19.4.4.2 Programmable Baud-Rate Generator

The USIM interface contains a programmable baud-rate generator. The same baud rate is used in transmitting and receiving data. Although the baud rate can be configured on any given moment, the actual change takes place only when the TX-FIFO is empty, no receive or transmit activities are in process, and block guard time is over. The USIM Divisor Latch register must be written when changing the baud rate even if the value is the same, see Section 19.4.4.2.2.

19.4.4.2.1 Determining the Baud Rate

When setting a new baud rate, software must specify three parameters in the following order:

1. RATIO—Ratio of the USIM internal clock verses the card clock (UCLK) (Equation 19-3). The card clock is defined in the USIM Clock register (CLKR) (Section 19.5.14).
2. FACTOR—Number of samples/bit minus one (ISO standard is a minimum of 6 samples/bit) (Figure 19-10). This parameter is loaded in the USIM Factor Latch register (FLR) (Section 19.5.16).

3. DIVISOR—Number of CLK_USIM cycles between samples, see Figure 19-11. This parameter is loaded in the USIM Divisor Latch register (DLR) (Section 19.5.15).

Figure 19-10. Baud-Rate Sampling Pulses When Number of Samples per Bit Is 6

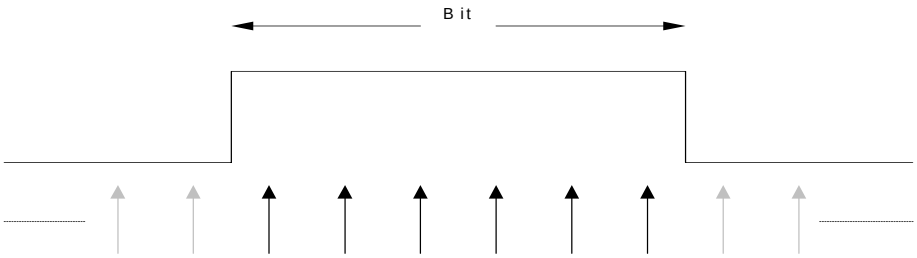
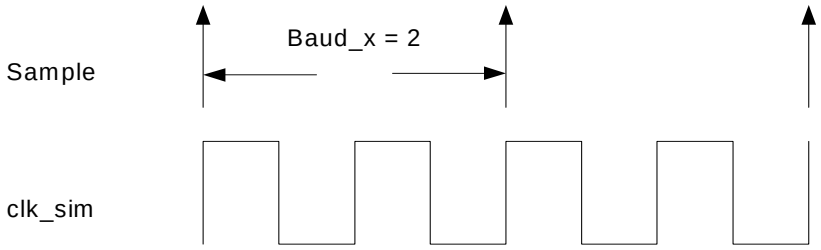


Figure 19-11. Spacing Between Samples When Baud Divisor Is 2



The baud rate, defined in Equation 19-5, is determined by using the above parameters with the following equations. The *F* and *D* parameters in Equation 19-4 are delivered by the card in the Answer-to-Reset (ATR) or negotiated in the Parameter and Protocol Selection (PPS) defined in the ISO standard.

Equation 19-3. $$\text{RATIO} = \frac{\text{CLK\_USIM}}{UCLK \times 2}$$

Equation 19-4. $$\frac{DIVISOR \times (FACTOR + 1)}{RATIO \times 2} = \frac{F}{D}$$

Equation 19-5. $$BaudRate = \frac{\text{CLK\_USIM (in Hz)}}{(DIVISOR) \times (FACTOR + 1)} = \frac{\text{UCLK} \times RATIO \times 2}{(DIVISOR) \times (FACTOR + 1)}$$

19.4.4.2.2 Handling a Parameter and Protocol Selection (PPS)

After a successful Parameter and Protocol Selection (PPS), the USIM interface is ready to start transmitting with the new protocol parameters. When writing to the DLR the USIM interface restarts counting of all waiting times (BGT, CWT, EGT, and BWT), so software must set the waiting times before writing the DLR. The FLR must also be written before the DLR. Software must write the DLR, even if the value has not changed, to restart the baud rate. The USIM interface starts transmitting only after the new BGT has passed.

19.4.4.2.3 Examples of Setting the Baud Rate

The following two examples cover the cases where F/D is an integer and non-integer number.

Example 19-1. Setting a New Baud Rate—F/D is an Integer Number

Consider the case where the Fi and Di parameters delivered in the ATR (TA(1) byte of the ATR) are 512 and 32, respectively. The USIM interface is expected to start the session in the new baud rate specified by the card. Because F/D is an integer number, F/D = 16, set any value to RATIO that results in a card clock approved by the standard. For this example, set RATIO = 6, resulting in a 4-MHz clock for the card. Now, software must set DIVISOR and FACTOR that obey Equation 19-4:

$$\frac{\text{DIVISOR} \times (\text{FACTOR} + 1)}{6 \times 2} = 16$$

Possible combinations are: DIVISOR = 12 and FACTOR = *15 or* DIVISOR = 24 and FACTOR = *7*. Other combinations are allowed by the standard as long as the number of samples per bit (FACTOR*)* is not less than 5.

With UCLK set to 4 MHz, the baud rate is $\frac{4MHz}{16} = 250KHz$.

Example 19-2. Setting a New Baud Rate—F/D is Not an Integer Number

Consider the case where Fi and Di parameters delivered in the ATR (TA(1) byte of the ATR) were 372 and 20, respectively. Because $\frac{F}{D} = 18\frac{3}{5}$, choose *RATIO = 10.*

(RATIO = *5* is also possible but not all cards operate at a clock frequency that is greater than 4 MHz). Now, software must set DIVISOR and FACTOR that obey Equation 19-4:

$$\frac{\text{DIVISOR} \times (\text{FACTOR} + 1)}{10 \times 2} = \frac{372}{20}$$

A possible combination is: DIVISOR = 31 and FACTOR = *11*.

The baud rate would be $\frac{48MHz}{31 \times 12} = 129.032KHz$.

19.4.5 Card Management

The USIM Card Control register (Section 19.5.9) controls the USIM card voltage supply, card reset, and I/O pad. Software uses this register to manage startup and shutdown of the card. In particular, it is used while executing the following routines:

1. Card activation (cold reset)
2. Card warm reset
3. Card deactivation

The following subsections describe the above procedures. Refer to USIM Card Control register (USCCR) (Section 19.5.9) for a description of the register.

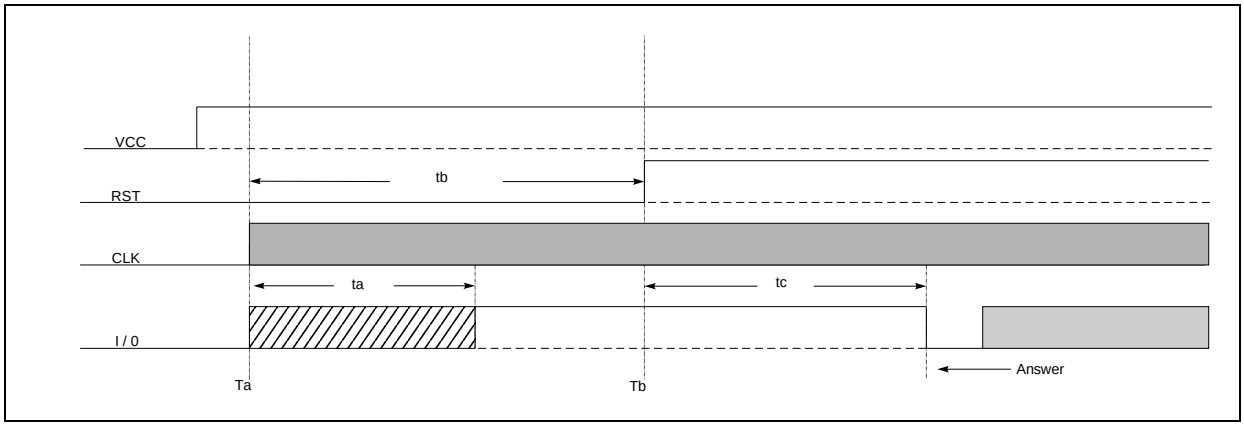
19.4.5.1 Card Activation (Cold Reset)

Perform the following in sequence to activate the USIM card:

1. Clear CLKR[STOP_CLK_USIM] in the USIM Clock register (CLKR) (Section 19.5.14) (after a system reset, this bit is cleared by default, so this step can be skipped).

2. Clear USCCR[RST_CARD_N] bit.
3. Turn on the VCC voltage by setting the USCCR[VCC] bits. On first activation, set the lowest voltage level. (Set the USCCR[VCC] bits to 0b10)
4. Enable the I/O line to return to V_{HIGH} by clearing the USCCR[TXD_FORCE] bit.
5. Activate the card clock by clearing the stop bit CLKR[STOP_UCLK] (Section 19.5.14).
6. Verify that USCCR[RST_CARD_N] was asserted for at least 400 UCLK cycles. When UCLK is set to the lowest frequency allowed by the standard, 400 cycles take 0.4 ms (400µsec). (See tb in Figure 19-12.)
7. Deassert nURST by setting USCCR[RST_CARD_N]. The card will Answer To Reset (ATR) within 400–40000 card clock cycles.(See tc in Figure 19-12.) If an ATR is not returned after 40000 card clock cycles, then steps 1–7 must be repeated with VCC voltage level increased from 1.8 V to 3.0 V in step 3. (set the USCCR[VCC] bits to 0b01).

Figure 19-12. Card Activation



19.4.5.2 Warm Reset

Warm reset is performed to reset the card after activation. A card reset does not reset the USIM FIFOs. The FIFOs can be reset in the USIM FIFO Control register (FCR) (Section 19.5.5). To execute a warm reset, perform the following operation:

1. Clear USCCR[RST_CARD_N] bit.
2. Verify that USCCR[RST_CARD_N] was cleared for at least 400 UCLK cycles. Note that when UCLK is set to the lowest frequency allowed by the standard, 400 cycles take 0.4 ms (400µsec).
3. Deassert nURST by setting USCCR[RST_CARD_N] bit.

19.4.5.3 Card Deactivation

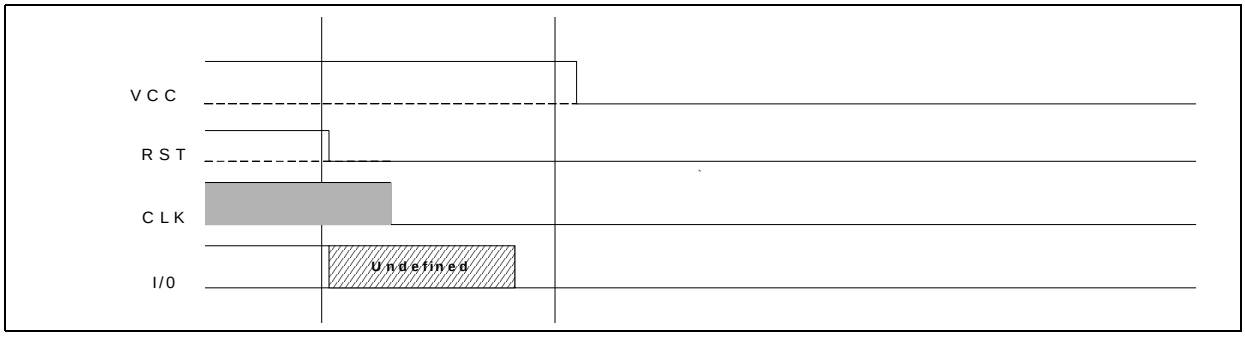
Perform the following in sequence to deactivate the USIM card:

1. Clear USCCR[RST_CARD_N] bit.
2. Stop the clock on V_{LOW} by clearing CLKR[STOP_LEVEL] and setting the CLKR[STOP_UCLK] bit.
3. Force the I/O line to ground level by setting USCCR[TXD_FORCE] bit.

4. Turn the VCC voltage to ground level by setting the USCCR[VCC] bits to 0b00 at least 200 UCLK cycles after the I/O line is forced low. The I/O line can be checked in the USIM Extra Guard Time register (EGTR), bit LSR[RXD] (Section 19.5.11). Operating system timers can determine the amount of time that has elapsed since LSR[RXD] went low.
5. For power savings, set CLKR[STOP_CLK_USIM] (Section 19.5.14).

Note: Any write to the CLKR is ignored while CLKR[RQST] is asserted.

Figure 19-13. USIM Card Deactivation



19.4.6 FIFO Operation

The USIM interface contains two 16-byte long FIFOs, a receive FIFO (RX-FIFO) and a transmit FIFO (TX-FIFO). Each is accessed with USIM interface registers RBR and THR, respectively. The USIM FIFO Control register (FCR) (Section 19.5.5), allows software to reset the FIFOs, hold transmission, mask parity errors in the RX-FIFO, and set the trigger levels for each. The data transmitted and received can be monitored using three modes: interrupt, polled, and DMA.

19.4.6.1 Interrupt Mode

In interrupt mode, software tracks and handles FIFO activities with interrupts. The interrupts are enabled in the USIM Interrupt Enable register (IER) (Section 19.5.3). The interrupt source can be determined by reading the USIM Interrupt Identification register (IIR) (Section 19.5.4).

19.4.6.1.1 Receiver-Related Interrupts

During data reception, LSR[RX_WORKING] is set, the following interrupts can occur when enabled:

- **Receiver Data Ready Interrupt**—The interrupt indicates the number of bytes in the RX-FIFO is equal to or greater than FCR[RX_TL]. The interrupt and its corresponding status bit is cleared when the number of bytes drops below FCR[RX_TL]. Empty the RX-FIFO by reading the RBR.
- **Character Waiting Time Interrupt—**This is an ISO protocol parameter and indicates the maximum delay between the leading edges of two consecutive characters has been violated. See Section 19.4.3.2.1 and the ISO standard for full details. Clearing the interrupt is performed by writing a 0b1 to IIR[CWT].
- **Receiver Time-Out Interrupt**—The receiver time-out interrupt helps software to detect when data is left in the receive FIFO for some time with no I/O activity occurring. This is known as *trailing bytes*. It occurs when at least one character remains in the RX-FIFO and the time-out period programmed in the USIM Time-Out register (TOR) (Section 19.5.13) has

expired. The interrupt and its corresponding status bit is cleared by reading the RBR, resetting the RX-FIFO, or when the USIM interface receives another character.

- **Framing Error Interrupt**—Occurs when the I/O line is asserted low during guard time. See Section 19.4.3.1.4 for full details. Clear the interrupt by setting IIR[FRAMERR].
- **Parity Error Interrupt in T = 0 mode**—The received data contains a parity error. See Section 19.4.3.1.2 for full details. Clear the interrupt by setting IIR[PERR].
- **Parity Error Interrupt in T = 1 mode**—The received data contains a parity error. See Section 19.4.3.1.2 for full details. Clear the interrupt by setting IIR[PERR].
- **Receiver Data Overrun Interrupt**—The receive FIFO is full and cannot accept any more data. See Section 19.4.3.1.3 for full details. Clear the interrupt by setting IIR[OVRN].

19.4.6.1.2 Transmitter-Related Interrupts

During transmission, LSR[TX_WORKING] is set, the following interrupts can occur when enabled:

- **Transmitter Data Refill Interrupt**—The interrupt indicates the number of bytes in the TX-FIFO is less than FCR[TX_TL]. The interrupt and its corresponding status bit is cleared when the number of bytes is at or above FCR[TX_TL].
- **Block Waiting Time Interrupt**—This is an ISO protocol parameter and indicates the maximum delay between the leading edge of the last character of a block sent by the USIM interface and the leading edge of the first character of the next block sent by the card has been violated. See Section 19.4.3.2.3 and the ISO standard for full details. Clear the interrupt by setting IIR[BWT].
- **T = 0 Error Interrupt**—In T = 0 mode, the card signals the transmitter to repeat the transmission. See Section 19.4.3.1.1 for full details. The interrupt is cleared using bits T0_REPEAT or T0_CLR from the USIM Error Control register.

19.4.6.1.3 Other Interrupts

- **Smart Card Detection Interrupt**—The interrupt indicates when a new smart card has been connected and remains asserted until the interrupt is cleared. The interrupt only asserts upon detection of a new card. This interrupt is tied to the PUCR[UDETS], see Section 3.8.1.14, “Power Manager USIM Card Control/Status Register (PUCR)” on page 3-90. The interrupt is cleared by setting IIR[CARD_DET].

19.4.6.2 Polled Mode

In FIFO polled mode, interrupt and DMA requests are disabled. FIFOs are accessed with reads and writes of the registers RBR and THR, respectively. Software checks the receiver and transmitter status by reading the LSR and FSR registers. The USIM FIFO Status register (FSR)(Section 19.5.6) holds information regarding FIFO status. The USIM Extra Guard Time register (EGTR) (Section 19.5.11) contains status information on data transfers. Because the receiver and the transmitter are controlled separately, either one or both can operate in FIFO polled mode.

19.4.6.3 DMA Mode

In DMA mode, data is entered and removed from the FIFOs by DMA requests. DMA requests are enabled in the USIM Interrupt Enable register (IER) (Section 19.5.3). The USIM interface has two DMA requests: one for transmit data service and one for receive data service.

Warning: The FIFO trigger levels in the USIM FIFO Control register must be set to 8 bytes to avoid unpredictable results when working in DMA mode. Other trigger levels are not supported in this mode.

19.4.6.3.1 Transmit Data Service

When the DMA_TX request is enabled in the IER and the number of bytes in the transmit FIFO is below eight, the DMA transmit request is generated. The DMA controller then writes data to the FIFO. For each DMA request, the DMA controller must send no more than eight bytes of data to the TX-FIFO. Sending fewer than eight is allowed. Each Write access of the DMA to the THR adds a single byte to the transmit FIFO.The number of bytes to be transmitted is programmed in the DMA controller.

19.4.6.3.2 Receive Data Service

The DMA receive request is generated in two situations:

1. When the DMA_RX request is enabled in the IER and the number of bytes in the receive FIFO is equal to or greater than eight, a DMA receive request is generated.
2. A time-out situation occurs when the DMA_TIME bit is enabled in the IER, the receive FIFO is not empty but the trigger level was not reached, and the RBR was not accessed within the time-out period defined in the USIM Time-Out register (TOR) ((Section 19.5.11) has expired. A DMA receive request is then generated to remove the trailing bytes.

When one of these situations occurs, the DMA controller then reads data from the RX-FIFO. For each DMA request, the DMA controller can read up to eight bytes of data. The number of bytes to be read is programmed in the DMA controller.

It is software's responsibility to ensure any remaining data in the RX-FIFO is handled properly when the DMA reaches the end of its chain. This is not be an issue because software knows how much data is being received via the *ISO 7816-3* protocol. The USIM interface asserts an end-of-receive (EOR) whenever the TOR or CWTR time-outs have expired and the DMA is reading the last byte in the RX-FIFO. To avoid having the DMA stop servicing the current chain when an EOR occurs and the end of a block has not yet been reached (recommended), software must setup the DMA Channel Control/Status register, DCSR, as noted in Table 19-3.

Table 19-3. DCSR Setup to Ignore EOR

DCSR Bit Name	Bit Value	Comment
StopIrqEn	1	Interrupt when the descriptor is finished.
EORIrqEn	0	No DMA interrupt after an EOR.
EORJmpEn	1	DMA services another channel if an EOR occurs.
EORStopEn	0	Keep waiting for more bytes until the descriptor is finished.

Refer to Figure 5-9, "Descriptor Behavior on End-of-Receive (EOR)" on page 5-47 for more information.

Special Condition:

In T = 1 DMA mode, parity errors enter the RX-FIFO. A DMA receive request only reads the first eight bits from the RBR, discarding the parity error bit. To determine if a block contains a parity error in this mode, software must follow one of two possible approaches. Each approach requires the DMA interrupt DCMDx[EndIrqEn] to be asserted. Refer to Section 5.4.3.1, “Servicing Internal Peripherals” on page 5-10 for more details on the DMA interrupt.

- With the USIM interface parity error interrupt disabled, the DMA reads the whole block regardless of parity errors. After the block is read the DMA interrupt occurs and software checks LSR[PERR] to see if the block contains any parity errors.
- With the USIM interface parity error interrupt enabled, when the interrupt occurs the software interrupt service routine waits for the DMA interrupt. Upon receiving the DMA interrupt, indicating the whole block has been read, software then requests a block retransmission.

19.5 Register Descriptions

The following registers describe the control, status, and data functionality of the USIM controller. These registers are listed in Table 19-24. The Power Manager USIM Card Control/Status register (PUCR) is also needed for control/status control of UDET and UEN signals, as described Section 3.8.1.14 on page 3-90.

19.5.1 USIM Receive Buffer Register (RBR)

The RBR contains the next byte to be read from the receive FIFO (see Table 19-4). In T = 1 mode, a byte containing parity errors has bit 8 asserted. Parity errors do not enter the FIFO in T = 0 mode.

This is a read-only register. Ignore reads from reserved bits.

Table 19-4. RBR Bit Definitions

Physical Address 0x4160_0000	RBR	USIM

User Settings																															
Bit	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																							PERR	RB[7:0]							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:9	—	—	reserved
8	R	PERR	Parity Error When not masked by the PEM bit in the FCR register, the parity error indicator appears in this bit. 0 = Either there was no parity error when receiving the character or the parity bit was masked. 1 = Character was received with a parity error.
7:0	R	RB[7:0]	Data Byte Received

19.5.2 USIM Transmit Holding Register (THR)

The THR contains the next byte to be written to the transmit FIFO. Data bytes written to this register are stored in the transmit FIFO and are transmitted to the card on the I/O pin in a first-in-first-out manner. After writing at least one byte to the transmit FIFO with the THR register, the USIM interface initiates access to the I/O line as soon as the Block Guard Time period is over. Software must keep track of whose turn it is to access the I/O line, the USIM interface or the card.

This is a write-only register. Write 0b0 to reserved bits.

Table 19-5. THR Bit Definitions

Physical Address 0x4160_0004	THR	USIM

User Settings																															
Bit	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																								TB[7:0]							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	W	TB[7:0]	Data Byte Transmitted Least significant bit first.

19.5.3 USIM Interrupt Enable Register (IER)

The USIM Interrupt Enable register (IER) enables USIM interface interrupts that independently activate the interrupt signal, and sets a value in the USIM Interrupt Identification register. IER is used in FIFO operation (see Section 19.4.6) and DMA requests (see Section 19.4.6.3). Each interrupt is explained in Section 19.4.6.1. DMA requests are also enabled in the IER and each is explained in Section 19.4.6.3.

Each interrupt type can be individually enabled or disabled using the IER register. The receiver time-out interrupt is separated from the receiver data ready interrupt to avoid having the CPU and the DMA controller serve the receive FIFO simultaneously.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-6. IER Bit Definitions (Sheet 1 of 2)

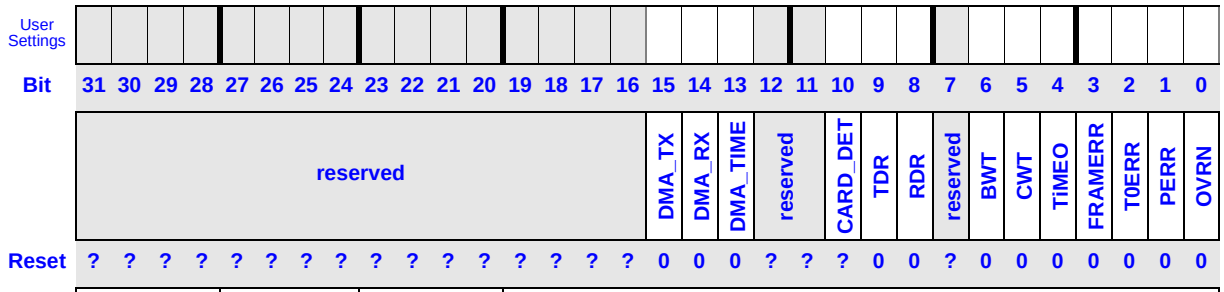
Physical Address 0x4160_0008 — **IER** — **USIM**

Bit	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
User Settings																																
	reserved																DMA_TX	DMA_RX	DMA_TIME	reserved		CARD_DET	TDR	RDR	reserved	BWT	CWT	TiMEO	FRAMERR	T0ERR	PERR	OVRN
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	0	0	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15	R/W	DMA_TX	DMA Transmitter Request Enable 0 = Disabled 1 = Enabled
14	R/W	DMA_RX	DMA Receiver Request Enable 0 = Disabled 1 = Enabled
13	R/W	DMA_TIME	Enable DMA Receiver Requests in Event of Time-Out 0 = Disabled 1 = Enabled
12:11	—	—	reserved
10	R/W	CARD_DET	Smart card Detection 0 = Disabled 1 = Enabled
9	R/W	TDR	Transmitter Data Refill Interrupt 0 = Disabled 1 = Enabled
8	R/W	RDR	Receiver Data Ready Interrupt 0 = Disabled 1 = Enabled
7	—	—	reserved

Table 19-6. IER Bit Definitions (Sheet 2 of 2)

Physical Address 0x4160_0008 — IER — USIM



Bits	Access	Name	Description
6	R/W	BWT	Block Waiting Time Interrupt 0 = Disabled 1 = Enabled
5	R/W	CWT	Character Waiting Time Interrupt 0 = Disabled 1 = Enabled
4	R/W	TIMEO	Receiver Time-Out Interrupt 0 = Disabled 1 = Enabled
3	R/W	FRAMERR	Framing Error Interrupt 0 = Disabled 1 = Enabled
2	R/W	TDR	Transmitter Data Refill Interrupt Enable 0 = Disabled 1 = Enabled
1	R/W	RDR	Receiver Data Ready Interrupt Enable 0 = Disabled 1 = Enabled
0	R/W	OVRN	Receiver Data Overrun Interrupt Enable 0 = Disabled 1 = Enabled

19.5.4 USIM Interrupt Identification Register (IIR)

The USIM Interrupt Identification register specifies the interrupt source. The IIR is used in the FIFO operation (Section 19.4.6). Each interrupt is explained in Section 19.4.6.1. After software handles the interrupt source, it must clear the interrupt by writing a 0b1 to the corresponding bit. There are several exceptions to this rule:

- The Receiver Data Ready Interrupt (RDR) is cleared only when the number of bytes in the receive FIFO drop below eight. The receive FIFO is unloaded by reading data from the RBR or by clearing the receive FIFO.
- Receiver Time-Out Interrupt (TIMEO) is cleared when reading the RBR or clearing the receive FIFO.
- The Transmitter Data Refill Interrupt (TDR) is cleared automatically when the number of bytes in the TX-FIFO is at or above eight. To fill the TX-FIFO, data must be written to the THR.

A T = 0 error is cleared by either setting a request to repeat the transmission of the last byte or by setting a request to proceed with transmission of the next byte. Both requests are set by writing the USIM Error Control register (ECR) (Section 19.5.7).

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-7. IIR Bit Definitions (Sheet 1 of 3)

Physical Address 0x4160_000C																															
	IIR																					USIM									
User Settings																															
Bit	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																					CARD_DET	TDR	RDR	reserved	BWT	CWT	TIMEO	FRAMERR	T0ERR	PERR	OVRN
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:11	—	—	reserved
10	R/W	CARD_DET	Smart card Detection Read Values: 0 = No interrupt. 1 = Interrupt has occurred. Write Values: 0 = No change. 1 = Clear this bit.
9	R	TDR	Transmitter Data Refill Interrupt 0 = No interrupt. 1 = Interrupt has occurred.
8	R	RDR	Receiver Data Ready Interrupt 0 = No interrupt. 1 = Interrupt has occurred.
7	—	—	reserved

Table 19-7. IIR Bit Definitions (Sheet 2 of 3)

	Physical Address 0x4160_000C												IIR										USIM									
User Settings																																
Bit	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																					CARD_DET	TDR	RDR	reserved	BWT	CWT	TIMEO	FRAMERR	T0ERR	PERR	OVRN
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
6	R/W	BWT	Block Waiting Time Interrupt Read Values: 0 = No interrupt. 1 = Interrupt has occurred. Write Values: 0 = No change. 1 = Clear this bit.
5	R/W	CWT	Character Waiting Time Interrupt: Read Values: 0 = No interrupt. 1 = Interrupt has occurred. Write Values: 0 = No change. 1 = Clear this bit.
4	R	TIMEO	Receiver Time-Out Interrupt 0 = No interrupt. 1 = Interrupt has occurred.
3	R/W	FRAMERR	Framing Error Interrupt Read Values: 0 = No interrupt. 1 = Interrupt has occurred. Write Values: 0 = No change. 1 = Clear this bit.

Table 19-7. IIR Bit Definitions (Sheet 3 of 3)

	Physical Address 0x4160_000C												IIR										USIM								
User Settings																															
Bit	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																					CARD_DET	TDR	RDR	reserved	BWT	CWT	TIMEO	FRAMERR	T0ERR	PERR	OVRN
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	R	T0ERR	T = 0 Error Interrupt 0 = No interrupt. 1 = Interrupt has occurred.
1	R/W	PERR	Parity Error Interrupt Read Values: 0 = No interrupt. 1 = Interrupt has occurred. Write Values: 0 = No change. 1 = Clear this bit.
0	R/W	OVRN	Receiver Data Overrun Interrupt Read Values: 0 = No interrupt. 1 = Interrupt has occurred. Write Values: 0 = No change. 1 = Clear this bit.

19.5.5 USIM FIFO Control Register (FCR)

The FCR allows the transmit and receive FIFOs to be reset, transmission to be stopped temporarily, and parity error masking to occur in T = 1 mode. When the Parity Error Mask (PEM) is asserted, parity errors are masked and do not appear in the USIM Receive Buffer register. When the Parity Error Mask is enabled, the USIM Line Status register (LSR) and the USIM FIFO Status register (FSR) can be examined to determine if parity errors have occurred. The transmitter and receiver trigger threshold must be set for eight bytes when in DMA mode to avoid unpredictable results.

This is a write-only register. Write 0b0 to reserved bits.

Table 19-8. FCR Bit Definitions

Physical Address 0x4160_0010 — FCR — USIM

User Settings

Bit	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																							TX_TL	RX_TL		reserved		PEM	TX_HOLD	RESETTF	RESETRF
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	0	0	?	?	0	0	0	0

Bits	Access	Name	Description
31:9	—	—	reserved
8	Read-unpredictable/ Write	TX_TL	Transmitter Interrupt Trigger Level (threshold) Set interrupt/DMA request trigger threshold in the transmit FIFO: 0 = Trigger threshold is 0 byte in TX-FIFO. 1 = Trigger threshold is 8 byte in TX-FIFO.
7:6	Read-unpredictable/ Write	RX_TL	Receiver Interrupt Trigger Level (threshold) Set interrupt/DMA request trigger threshold in the receive FIFO: 0b00 = Trigger threshold is 1 byte in RX-FIFO. 0b01 = Trigger threshold is 4 byte in RX-FIFO. 0b10 = Trigger threshold is 8 byte in RX-FIFO. 0b11 = Trigger threshold is 12 byte in RX-FIFO.
5:4	—	—	reserved
3	Read-unpredictable/ Write	PEM	Parity Error Mask 0 = Bit 8 of the USIM Receive Buffer register is asserted when a character enters the RX-FIFO with a parity error in T = 1 mode. 1 = A parity error coming out of the RX-FIFO is masked and does not appear on bit 8 of the USIM Receive Buffer register (RBR). Evidence of a parity error can still be found in: • USIM Line Status register, LSR[PERR]. • USIM FIFO Status register, FSR[PERR_NUM]. • USIM Interrupt Identification register, IIR[PERR] (provided the parity error interrupt is enable).
2	Read-unpredictable/ Write	TX_HOLD	Transmit Hold When asserted, transmission stops after the current character is over. All the remaining characters are stored in the transmitter FIFO. Transmission resumes when the bit is deasserted. 0 = Transmission continues. 1 = Holds transmission.
1	Read-unpredictable/ Write	RESETTF	Reset Transmit FIFO 0 = No effect. 1 = Transmit FIFO is cleared.
0	Read-unpredictable/ Write	RESETRF	Reset Receive FIFO 0 = No effect. 1 = Receive FIFO is cleared.

19.5.6 USIM FIFO Status Register (FSR)

The FSR provides software with the number of bytes in each of the FIFOs and the number of parity errors in the USIM receive FIFO. This register is used when working in polled mode (see Section 19.4.6.2).

This is a read-only register. Ignore reads from reserved bits.

Table 19-9. FSR Bit Definitions

Physical Address 0x4160_0014 — **FSR** — **USIM**

	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
User Settings																																
Bit	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																	PERR_NUM					TX_LENGTH					RX_LENGTH				
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:15	—	—	reserved
14:10	R	PERR_NUM	Parity Error Number Number of characters with a parity error in the RX-FIFO.
9:5	R	TX_LENGTH	Transmit FIFO Length Number of characters in the TX-FIFO.
4:0	R	RX_LENGTH	Receive FIFO Length Number of characters in the RX-FIFO.

19.5.7 USIM Error Control Register (ECR)

The USIM Error Control register allows software to determine the USIM interface behavior in parity and T = 0 error situations when working in T = 0 mode. Table 19-10 shows the format of the USIM Error Control register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-10. ECR Bit Definitions (Sheet 1 of 2)

Physical Address 0x4160_0018	ECR	USIM

User Settings																															
Bit	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																								T0_REPEAT	T0_CLR	reserved	PE-TL		reserved	T0ERR_TL	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	?	1	1

Bits	Access	Name	Description
31:8	—	—	reserved
7	R/W	T0_REPEAT	Repeat Character Transmission Repeat character transmission until error trigger threshold (see below) is met again. Read Values: 0 = Character has been repeated. 1 = Character repeat in process. Write Values: 0 = No effect 1 = Repeat character transmission until trigger threshold is met.
6	R/W	T0_CLR	Clear T = 0 Error Read Values: 0 = T = 0 error has been cleared. 1 = Clearing of T = 0 error in process. Write Values: 0 = No effect 1 = Clears T = 0 error indicator and transmission continues with the net character in the TX-FIFO.
5	—	—	reserved

Table 19-10. ECR Bit Definitions (Sheet 2 of 2)

Physical Address 0x4160_0018 — **ECR** — **USIM**

Bit	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
User Settings																																
	reserved																								T0_REPEAT	T0_CLR	reserved	PE-TL		reserved	T0ERR_TL	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	?	1	1

Bits	Access	Name	Description
4:3	R/W	PE_TL	Parity Error Trigger Level 0b00 = Reception of a single parity error causes an interrupt to occur, if enabled 0b01 = Reception of two consecutive parity error causes an interrupt to occur, if enabled. 0b01 = Reception of three consecutive parity error causes an interrupt to occur, if enabled. 0b11 = Reception of four consecutive parity error causes an interrupt to occur, if enabled.
2	—	—	reserved
1:0	R/W	T0ERR_TL	T = 0 Error Trigger Level (threshold) Specifies number of repetition upon detection of T = 0 error. When trigger threshold is met, both transmission hold and a T = 0 interrupt are generated. 0b00 = Reception of single T = 0 error would cause interrupt and transmission hold. 0b01 = Reception of two consecutive T = 0 error would cause interrupt and transmission hold. 0b10 = Reception of three consecutive T = 0 error would cause interrupt and transmission hold. 0b11 = Reception of a four consecutive T = 0 error would cause interrupt and transmission hold.

19.5.8 USIM Line Control Register (LCR)

The LCR allows software to define protocol and coding conventions for the transmitter and receiver. The serial data format consists of a start bit (logic 0), eight data bits that can appear in different order and polarity, and a closing even parity bit. When the T = 0 protocol is selected, error signaling can appear over the line through guard time.

Bits [1:0] of the LCR determine the data bit order and polarity. After receiving the first byte of the ATR, programmers must decide whether the data order should be mirrored and whether data polarity should change. Change of polarity and order is not limited to the period of time prior to second character arrival, so there is no need to reset the card in case the transmission was not in the expected order.

Bit [2] determines whether the parity bit is even. Set this bit when no activity on the I/O line is expected.

Bit [3] determines protocol type when transmitting a character. If the LCR.RX_T = 1_MODE is 0, then the protocol is T = 0. If the bit is set, then the protocol is T = 1. In T = 0 protocol, the error signal (V_{LOW} through guard time) may appear on the line.While in receive mode (rx_working), the USIM interface device generates T = 0 error signaling to the card in two cases:

- The parity check result differs from that expected. In other words, exclusive OR-ing of 8 data bits and a parity bit yield the same value as the EPS bit (see bit 1)
- The receive FIFO is full when data transmission is complete.

In both of these cases, the data read is not placed in the receive FIFO because retransmission of the data is expected.

LCR[TX_T1] determines the protocol type when transmitting a character. If the bit is clear, then the protocol is T = 0. If the bit is set, then the protocol is T = 1. In T = 0 protocol, the error signal (V_{LOW} through guard time) may appear on the line.While in transmit mode (tx_working), the USIM interface device senses for a T = 0 error signaled by the card and repeats transmission of the previous byte in case such an error signal was detected on I/O. When the number of errors sensed in any data transmission exceeds the error trigger threshold (TQERR_TL bits in the ECR register), transmission stops and software intervention is required.

Protocol switch policy: When communicating using the USIM, protocol switching can occur at three times:

- Through ATR—After receiving the T = 0 byte, programmers can change the protocol to T = 0 if the card supports this protocol. The advantage of using the T = 0 protocol is an elimination of a card reset when a single parity error occurs.
- After the card finishes the ATR, the USIM is expecting transmission using the protocol defined by T = 0 and TA2 (if it exists).
- After PPS—the USIM interface can initiate a Protocol and Parameter Selection session. If the session ended successfully, the SIM interface protocol can be changed.

Avoid any change in protocol under other scenarios.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-11. LCR Bit Definitions

Physical Address 0x4160_001C — LCR — USIM

	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
User Settings																																
Bit	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																											TX_T1	RX_T1	EPS	ORDER	INVERSE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	1	0	0

Bits	Access	Name	Description
31:5	—	—	reserved
4	R/W	TX_T1	Transmitter Protocol 0 = T = 0 protocol set for the transmitter. 1 = T = 1 protocol set for the transmitter.
3	R/W	RX_T1	Receiver Protocol 0 = T = 0 protocol set for the receiver. 1 = T = 1 protocol set for the receiver.
2	R/W	EPS	Even Parity Select (EPS) When EPS is a logic 0, an odd number of logic ones is transmitted or checked in the data word bits and the parity bit. When EPS is a logic 1, an even number of logic ones is transmitted or checked in the data word bits and parity bit. 0 = Sends or checks for odd parity as required by inverse convention. 1 = Sends or checks for even parity as required by direct convention. See Section 19.4.2.
1	R/W	ORDER	Transmit/Receive Bit Order 0 = Regular format—LSB of data transmitted/received first (after start bit), MSB of data transmitted/received last (and then guard time). Used in Direct Convention. See Section 19.4.2. 1 = Inverted Format—MSB of data transmitted/received first (after start bit), LSB of data transmitted/received last (and then guard time). Used in Inverse Convention. See Section 19.4.2.
0	R/W	INVERSE	Bit Inversion 0 = Logic 0 transmitted/received as V_{LOW}, logic 1 transmitted/received as V_{HIGH}. Used in Direct Convention. See Section 19.4.2. 1 = Logic 1 transmitted/received as V_{LOW}, logic 0 transmitted/received as V_{HIGH}. Used in Inverse Convention. See Section 19.4.2.

19.5.9 USIM Card Control Register (USCCR)

The USCCR is used for card management (Section 19.4.5). Table 19-12 shows the format of the USIM Card Control register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-12. USCCR Bit Definitions

Physical Address 0x4160_0020 — **USCCR** — **USIM**

	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
User Settings																																
Bit	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																											TXD_FORCE	reserved	VCC		RST_CARD_N
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	?	0	0	0

Bits	Access	Name	Description
31:5	—	—	reserved
4	R/W	TXD_FORCE	Force TXD This bit must always remain non-active unless running a deactivation process. In deactivation I/O must be turned low before turning down card's voltage. 0 = TXD indicates transmission. 1 = TXD is forced to V_{LOW}.
3	—		reserved
2:1	R/W	VCC	Card Voltage 0b00 = USIM card at 0 V (GND). 0b01 = USIM card at 3 V. 0b10 = USIM card at 1.8 V. 0b11 = reserved
0	R/W	RST_CARD_N	Card Reset. 0 = Reset the USIM card. 1 = Normal USIM card operation.

19.5.10 USIM Line Status Register (LSR)

The LSR provides status information concerning data transfers. Most of the status bits have a corresponding interrupt bit and are explained in detail in Section 19.4.6. The LSR bits that do not have a corresponding interrupt are: RXD, RX_WORKING, TX_WORKING, RX_EMPTY_N, TX_REFILL. The waiting time and error status bits (BWT, CWT, FRAMERR, PERR, OVRN) are cleared after the LSR is read or when their corresponding interrupt is cleared. The T = 0 error is an exception and is cleared only when ECR[T0_REPEAT] or ECR[T0_CLR] is set.

An interrupt associated with a waiting time or error status bit is not cleared when the corresponding status bit is read from the LSR. The interrupt is only cleared when writing a 0b1 to the proper bit in the USIM Interrupt Identification register (IIR) (Section 19.5.4).

This is a read-only register. Ignore reads from reserved bits.

Table 19-13. LSR Bit Definitions (Sheet 1 of 2)

Physical Address 0x4160_0024 — LSR — USIM

	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
User Settings																																
Bit																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved																RXD	RX_WORKING	TX_WORKING	RX_EMPTY_N	TDR	reserved				BWT	CWT	TIMEO	FRAMERR	T0ERR	PERR	OVRN
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	0	0	0	1	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15	R	RXD	Reflects Sampled Data from I/O Pad 0 = I/O line is currently on logic 0. 1 = I/O line is currently on logic 1.
14	R	RX_WORKING	Receiver Working 0 = Receiver idle. 1 = Receiver is in process.
13	R	TX_WORKING	Transmitter Working 0 = Transmitter idle. 1 = Transmitter in process.
12	R	RX_EMPTY_N	Receiver FIFO Not Empty 0 = Receiver FIFO is empty. 1 = Receiver FIFO is not empty.
11	R	TDR	Transmitter Data Refill 0 = Number of characters in the TX-FIFO is at or above FCR[TX_TL]. 1 = Number of characters in the TX-FIFO is less than FCR[TX_TL].
10:7	—	—	reserved
6	R	BWT	Block Waiting Time 0 = BWT time-out period is not over. 1 = BWT time-out period is over.

Table 19-13. LSR Bit Definitions (Sheet 2 of 2)

Physical Address 0x4160_0024 — **LSR** — **USIM**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User Settings	reserved	RXD	RX_WORKING	TX_WORKING	RX_EMPTY_N	TDR	reserved			BWT	CWT	TIMEO	FRAMERR	T0ERR	PERR	OVRN
Reset	? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?	1	0	0	0	1	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
5	R	CWT	Character Waiting Time 0 = CWT time-out period is not over. 1 = CWT time-out period is over.
4	R	TIMEO	Receiver Time-Out 0 = TOR time-out period is not over. 1 = TOR time-out period is over.
3	R	FRAMERR	Framing Error 0 = Framing error has not occurred. 1 = Framing error has occurred.
2	R	T0ERR	T = 0 Error 0 = During transmission, the number of consecutive T = 0 errors did not meet the repetition trigger level, ECR[T0ERR_TL]. 1 = During transmission, the number of consecutive T = 0 errors met the repetition trigger level, ECR[T0ERR_TL].
1	R	PERR	Parity Error 0 = During reception, the number of consecutive parity errors did not met the repetition trigger threshold, ECR[PE_TL]. 1 = During reception, the number of consecutive parity errors met the repetition trigger threshold, ECR[PE_TL].
0	R	OVRN	Receiver Data Overrun Error 0 = Overrun error has not occurred. 1 = Overrun error has occurred.

19.5.11 USIM Extra Guard Time Register (EGTR)

The EGTR specifies the number of Extra Guard-Time Moments (EGTM) between bytes transmitted by the USIM interface. This register can hold any value between 0–255. In the T = 0 protocol, the USIM interface starts measuring the extra guard time period 12 moments after the last byte's start bit. In T = 1, counting of the Extra Guard Time period starts 11 etu after transmission of the last byte's start bit. Table 19-14.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-14. Number of ETU between a Transmitted Byte's Leading Edge in Different Protocols

Protocol	Spacing Between a Transmitted Byte's Leading Edge (in etu)
T = 0	12 + EGTM
T = 1	11 + EGTM

Table 19-15. EGTR Bit Definitions

Physical address 0x4160_0028 — EGTR — USIM

User Settings																															
Bit	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																								EGTM							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	EGTM	Extra Guard-Time Moments Number of total guard-time moments.

19.5.12 USIM Block Guard Time Register (BGTR)

Block guard time, is a protocol parameter from standard *ISO 7816-3* and specification *3G TS 31.101*. It is defined as the minimum number of guard-time moments (starting from the last mandatory guard-time moment) in which the USIM interface transmitter cannot access the I/O port after the card finishes its transmission.

Note: The value used in the BGT register is not the same as the value specified for ISO 7816-3 and 3G TS 31.101. The BGT register uses a value that is 12 moments less in T = 0 mode and 11 moments less in T = 1 mode than the value specified by the standards. See Table 19-16.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-16. Block Guard Time Values

Block Guard Time	T = 0	T = 1
Value as specified in protocol BGT definition	16	22
Value converted to BGTR definition	5	11

Table 19-17. BGTR Bit Definitions

Physical address 0x4160_002C — BGTR — USIM

	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
User Settings																																
Bit	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																								BGT							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	1	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	BGT	Block Guard Time Number of total block guard-time moments.

19.5.13 USIM Time-Out Register (TOR)

The TOR lets software detect when data is left in the receive FIFO for some time with no I/O activity occurring. See Section 19.4.6 for more information. Time-Out is measured from the number of moments that have passed either since the last character entered the receive FIFO or since the receive FIFO was read. The count stops when the receive FIFO is empty or I/O activity is occurring. Table 19-18 shows the format of the USIM Time-Out register, which sets the trigger threshold of the time-out interrupt.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-18. TOR Bit Definitions

Physical address 0x4160_0030 — TOR — USIM

	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
User Settings																																
Bit	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																								TO							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	1	0	1	1	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	TO	Time-Out Number of total time-out moments.

19.5.14 USIM Clock Register (CLKR)

The CLKR allows software to set the values of the card clock, UCLK, using the methodology described in Section 19.4.4. Note the CLKR cannot be updated while CLKR[RQST] is set. Any writes to the CLKR when CLKR[RQST] is set is ignored. After reset, the card clock is disabled and its frequency is set to 4 MHz by default. For power savings, the USIM interface clock CLK_USIM can be disabled by asserting bit CLKR[15]. After reset, CLK_USIM is enabled by default. Note, the USIM interface registers can be read/written when CLK_USIM is disabled. Table 19-19 shows the format of the USIM Clock register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-19. CLKR Bit Definitions

Physical Address 0x4160_0034	CLKR																USIM													
User Settings																														
Bit	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																STOP_CLK_UCLK	STOP_LEVEL	STOP_UCLK	RQST	reserved				DIVISOR[7:0]							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	0	1	0	?	?	?	?	0	0	0	0	0	1	1	0

Bits	Access	Name	Description
31:16	—	—	reserved
15	R/W	STOP_CLK_UCLK	Stop USIM Interface Clock 0 = Starts the USIM clock, CLK_USIM. 1 = Stops the USIM clock. Does not stop the clock that controls the USIM registers.
14	R/W	STOP_LEVEL	Stop Level 0 = Clock stops on low. 1 = Clock stops on high.
13	R/W	STOP_UCLK	Stop Clock 0 = Starts the card's clock. 1 = Stops the card's clock.
12	R	RQST	Clock Change Request 0 = Request not occurring, CLKR can be updated. 1 = Request occurring, do not update CLKR.
11:8	—	—	reserved
7:0	R/W	DIVISOR	Clock Divisor The card's clock frequency is 48 MHz/DIVISOR. The value zero results in unpredictable behavior.

Note: Changing UCLK affects the etu period. See Section 19.4.4.2. Software can enter a new value to CLKR at any given moment. The new value is validated only when the USIM interface completes all previous jobs that are pending (transmission/reception, block guard time and extra guard time periods are finished, and the receive FIFO is empty).

Note: Intel recommends that the input clock frequency be 48 MHz. However, it is possible to use other frequencies. In such cases, it is important to change the value of the CLKR register and follow the values of the USIM programmable baud-rate generator registers.

19.5.15 USIM Divisor Latch Register (DLR)

The DLR contains the Divisor used in controlling the baud rate. See Section 19.4.4.2. The reset value conforms to the *ISO 7816-3* requirement that the etu is 372 clock cycles during the ATR. Table 19-20 shows the format of the USIM Divisor Latch register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-20. DLR Bit Definitions

	Physical Address 0x4160_0038								DLR								USIM											
User Settings																												
Bit	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																DIVISOR															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0

Bits	Access	Name	Description
31:16	reserved	—	reserved
15:0	R/W	DIVISOR	Baud Divisor Determines the number of USIM clock in between samples, forcing the total length of a bit to be DIVISOR*(FACTOR+1)*(USIM_CYCLE). The value of DIVISOR must not be set to zero.

Note: Change in the actual baud rate is not immediate.

19.5.16 USIM Factor Latch Register (FLR)

The FLR contains the Factor used in controlling the baud rate. See Section 19.4.4.2. The reset value conforms to the *ISO 7816-3* requirement that the etu is 372 clock cycles during the ATR. Table 19-21 shows the format of the USIM Factor Latch register. The FLR must be modified before the DLR.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-21. FLR Bit Definitions

	Physical Address 0x4160_003C																FLR								USIM							
User Settings																																
Bit	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																								FACTOR							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	1	0	1	1

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	FACTOR	Baud Factor Determines the number of samples per bit. Number of samples would be (FACTOR + 1). ISO standard demands minimum of 6 samples; therefore 5 is the minimum value of FACTOR bits.

19.5.17 USIM Character Waiting Time Register (CWTR)

Character waiting time, see Section 19.4.3.2.1, is a protocol parameter from standard *ISO 7816-3* and specification *3G TS 31.101*. It is defined as the maximum delay between the leading edges of two consecutive characters in a block. During the ATR, the ISO standard specifies an initial waiting time of 9600 etu. This initial waiting time is equivalent to CWT, and so the reset value of CWT is 9600 – 12 = 9588 etu. The 12 is a result of how the CWT is measured and because the FIFOs are reset to T = 0 mode. For a receive occurring in T = 1 mode, the value would be: 9600 – 11 = 9589.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-22. CWTR Bit Definitions

	Physical address 0x4160_0040																CWTR								USIM							
User Settings																																
Bit	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																CWT															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	1	0	0	1	0	1	0	1	1	1	0	1	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R/W	CWT	Character Waiting Time Number of total character waiting time moments.

19.5.18 USIM Block Waiting Time Register (BWTR)

Block Waiting Time, see Section 19.4.3.2.3, is a protocol parameter from standard *ISO 7816-3* and specification *3G TS 31.101*. It is defined as the maximum delay between the leading edge of the last character of the block received by the card and the leading edge of first character of the next block sent by the card.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 19-23. BWTR Bit Definitions

Physical address 0x4160_0044	BWTR	USIM

	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
User Settings																																
Bit	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																BWT															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R/W	BWT	Block Waiting Time Number of total block waiting time moments.

19.6 Register Summary

There are eighteen 32-bit-wide registers in the USIM interface. Table 19-24 shows the registers and their addresses.

Table 19-24. USIM Interface Register Summary

Address	Name	Description	Page
0x4160_0000	RBR	USIM Receive Buffer register	19-18
0x4160_0004	THR	USIM Transmit Holding register	19-19
0x4160_0008	IER	USIM Interrupt Enable register	19-20
0x4160_000C	IIR	USIM Interrupt Identification register	19-22
0x4160_0010	FCR	USIM FIFO Control register	19-24
0x4160_0014	FSR	USIM FIFO Status register	19-26
0x4160_0018	ECR	USIM Error Control register	19-27
0x4160_001C	LCR	USIM Line Control register	19-29
0x4160_0020	USCCR	USIM Card Control register	19-31
0x4160_0024	LSR	USIM Line Status register	19-32
0x4160_0028	EGTR	USIM Extra Guard Time register	19-34
0x4160_002C	BGTR	USIM Block Guard Time register	19-34
0x4160_0030	TOR	USIM Time-Out register	19-35
0x4160_0034	CLKR	USIM Clock register	19-36
0x4160_0038	DLR	USIM Divisor Latch register	19-37
0x4160_003C	FLR	USIM Factor Latch register	19-37
0x4160_0040	CWTR	USIM Character Waiting Time register	19-38
0x4160_0044	BWTR	USIM Block Waiting Time register	19-39
0x4160_0048–0x4160_FFFC	—	reserved	reserved

USB Host Controller 20

This chapter describes the Universal Serial Bus host controller (UHC) implemented in the PXA27x processor.

20.1 Overview

The Universal Serial Bus (USB) supports serial data exchanges between a host computer and a variety of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host-scheduled, token-based protocol. Peripherals can be attached, configured, used, and detached, while the host and other peripherals continue operation. Familiarity with the *Universal Serial Bus Specification*, Revision 1.1[1] and the OHCI specification[2] are necessary to fully understand the material contained in this section

See the *Universal Serial Bus Specification Revision 1.1* and the *Open HCI—Open Host controller Specification for USB* for details of the interface operation.

20.2 Features

- USB Rev. 1.1 compatible
- Supports both low-speed and full-speed USB devices
- Open Host Controller Interface (OHCI) Rev 1.0a compatible
- Root hub supports two downstream ports

20.3 Signal Descriptions

This section describes the signal pins that are inputs or outputs from the USB host controller (see Table 20-1).

1. The latest revision of the *Universal Serial Bus Specification Revision 1.1* can be accessed at: http://www.usb.org/
2. Open Host Controller Interface Specification for USB, Release 1.0a.

Table 20-1. USB Host Controller I/O Signal Descriptions

Name	Type	Description
USBHPWR<3:1>	Input	Over-current indicator from ports 3, 2 and 1
USBH_P<3:1>	Inout	Data positive to ports 3, 2 and 1
USBH_N<3:1>	Inout	Data minus to ports 3, 2 and 1.
USBHPEN<3:1>	Output	Controls power to the USB ports

A USB host must supply 5.0 volts (per the USB specification); however, the PXA27x processor does not have 5.0-volt-tolerant pads. Therefore, system designers must provide an external device to interface the USBHPEN and USBHPWR pins to the power supply and over-current detection circuits.

20.3.1 Input Signals

The USBHPWR<3:1> signals are active-high signals that indicate an over-current fault condition on the USB power supply. The USB host can be programmed to change the polarity of these signals to be active low. These signals are multiplexed with GPIO pins; please refer to Chapter 24, “General-Purpose I/O Controller” for details on configuration.

20.3.2 Output Signals

The USBHPEN<3:1> signals reflect the status of the UHCRHPSx[PPS] bits and control an external power-switching device that supplies power to USB peripherals. The USB host can be programmed to change the polarity of these signals to be active low. These signals are mixed with GPIO pins; refer to Chapter 24, “General-Purpose I/O Controller” for details on configuration.

20.3.3 Bidirectional Signals

USBH_P<3:1> and USBH_N<3:1> are the differential signals of the USB ports.

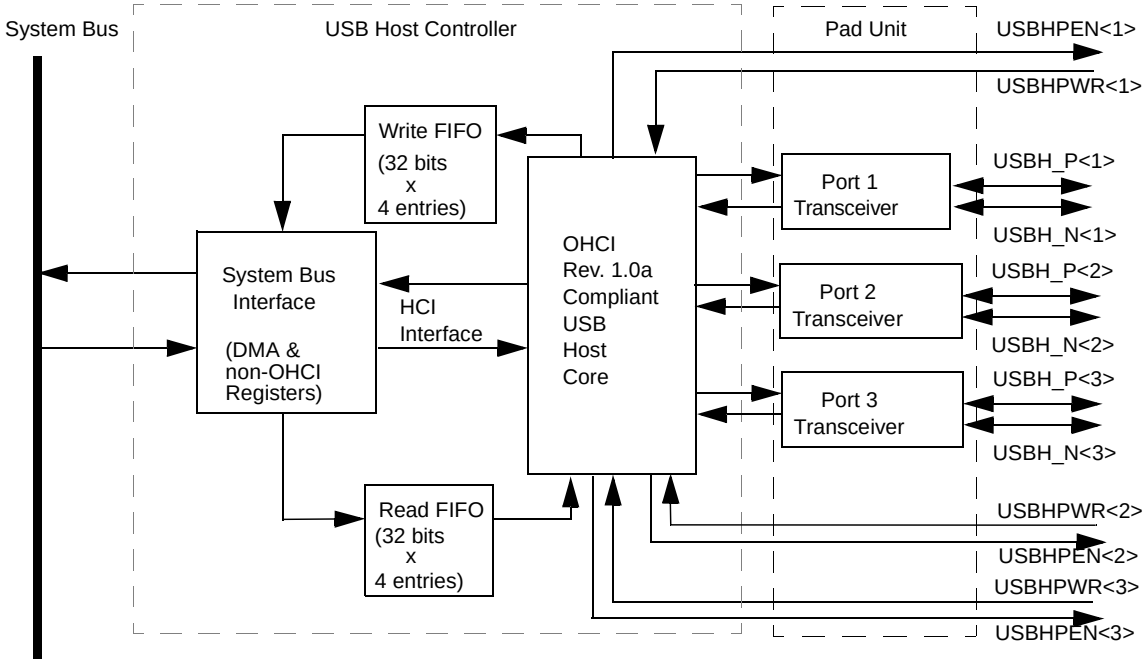
20.4 Operation

Note: Although the PXA27x processor provides three USB host ports, the second and third USB host ports (USBH2 and USBH3) must be configured through the USB client registers. The first port (USBH1) is assigned purely to the host, while the second and third ports are programmed using both the USB host and client modules.

A USB system consists of four main components: two of these, the client software and the host controller driver, are implemented in software; the other two areas (host controller and the device controller) are implemented in hardware. The host controller driver and the host controller work together to serially transfer data between a shared memory data structure and the USB controller. Figure 20-1 shows a block diagram of the USB host controller. This module consists of an OHCI-compliant core, a bus interface unit to connect it the system bus, two small FIFOs for buffering data in and out, 2 input pins, 2 output pins, and the transceivers located in the pad unit. The bus interface unit connects to the system bus for register access and for writing and reading of the FIFOs. The registers must be accessed as 32-bit entities on 32-bit aligned addresses.

The serial information transmitted and received by the USB host (USBH) contains layers of communications protocols, the most basic of which are fields. USBH fields include sync, packet identifier, address, Endpoint, Frame Number, data, and CRC fields. Fields are used to produce packets. Depending on the packet function, a different combination and number of fields can be used. Packet types include token, start of frame, data, and handshake. There are four data transfer types define in USB: bulk, control, interrupt, and isochronous. Packets are assembled into groups to produce frames. Data transfers can be grouped into two categories: periodic (isochronous and interrupt) and non-periodic (control and bulk). Fields inside of the Endpoint Descriptor (ED) and Transfer Descriptor (TD) memory structures define what type of transfer is to take place.

Figure 20-1. USB Host Controller Block Diagram

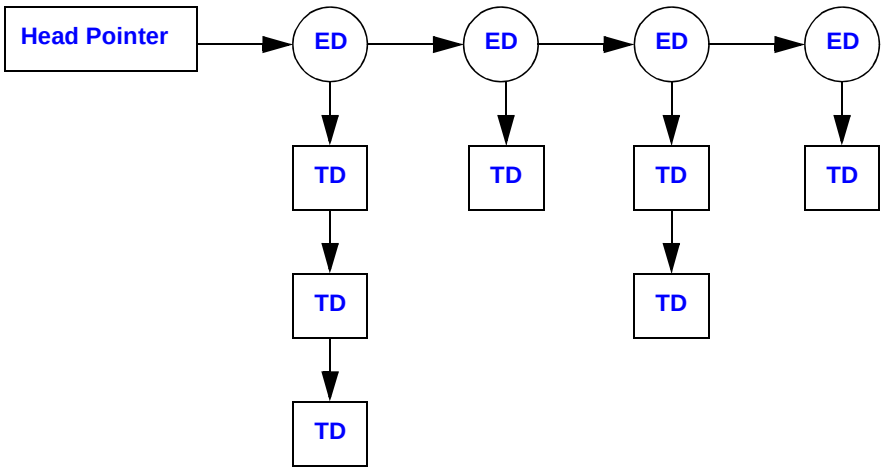


There are two communication channels between the host controller and the host controller driver. The first channel uses a set of registers for control, status and list pointers (described in Section 20.9). The second communication channel is the Host Controller Communication Area (HCCA). The HCCA contains pointers to the interrupt ED, done queue, and status information associated with start of frame processing.

The USB host controller functions as a “smart” DMA operating on linked lists (EDs and TDs) created by the HCD and located in system memory. The HCD assigns an Endpoint Descriptor to each Endpoint in the system. The information in these descriptors include: maximum packet size, the Endpoint address, the speed of the Endpoint, and the direction of data flow. A queue of TDs is linked to the ED for a specific Endpoint. The TDs contain information on data toggle, shared memory buffer location and completion status codes. The HCD creates these ED lists and TD queues then passes control to the UHC for processing (by setting bits two through five of the UHCHCON register, described in Section 20.8.2). The HCD adds to the TD queues and the UHC removes from the queues by linking a finished TD with the Done Queue. The UHC updates fields (such as Current Buffer Pointer and Condition Code) in the TD in system memory space upon completion of a TD.

Head pointers to the Bulk and Control ED lists are maintained in the UHC (UHCBHED and UHCCHED registers). The HCD must initialize these registers. Figure 20-2 illustrates a typical list structure.

Figure 20-2. Typical List Structure



Interrupt EDs are maintained in the HCCA[1]. The HCD must also maintain the state of the UHC (operational, resume, suspend, reset), the list processing pointers (the UHCBHED and UHCCHED registers), list processing enables (bits 5-2 of the UHCHCON register), and interrupt enables (in the UHCINTE register).

Note: Remote host wake from sleep may not be absolutely USB 1.1 specification-compliant. The USB host controller may not be able to feed back resume signaling to downstream devices within 100 microseconds.

Note: In isochronous transfer mode, the latency associated with using VLIO and PC Card memory accesses may violate the 10-µs time limit. As a result, the USB host controller sends a corrupted CRC (an OUT packet) or not issue an ACK; in the latter case, an interrupt occurs (if enabled) and the ISO packet is dropped by software.

20.5 Interrupts

The USB host controller generates two interrupts to the interrupt controller (see Chapter 25, "Interrupt Controller" for details).

- An OHCI USB interrupt (generated from the Interrupt Status register; see Section 20.8.4)
- All other USB host controller event interrupts (see Section 20.8.23)
  - — Buffer-access interrupt
  - — Remote wake-up interrupt
  - — Port-resume signal interrupt

1. See the OHCI Rev.1.0a specification, page 9, section 3.2.2 for details.

— OHCI-initiated interface-clear signal interrupt (transfer abort)

— USB port-power-over-current-exception interrupts

20.6 Programming Considerations

Programming the USB host controller is very similar to programming the USB host in the Intel® StrongARM* SA-1111 microprocessor companion chip.[1] The major exceptions are in the base address of the OHC registers and in functions of the Status register (UHCSTAT). In the SA-1111 companion chip, this register (referred to as USB Status register at address 0x0000518) monitors the host controller events, no events were latched, and a total of five different interrupts could be generated. In the USB host controller unit, two interrupts can be sent (OHCI and non-OHCI). The non-OHCI interrupt is generated from the UHCSTAT register, which latches all events (enabled interrupts), and the bits remain set until cleared by the HCD.

On system reset, the default condition is for the UHC Reset register (UHCHR) to enable each port by clearing either UHCHR[SSEP1] or UHCHR[SSEP2] or UHCHR[SSEP3], or globally by clearing UHCHR[SSE]. If either USB port 1, port 2, or USB port 3 is not physically available, it must be disabled by setting the appropriate bit (see Section 20.8.24) for more information.

20.6.1 USB Reset

The USB host controller is not fully reset following a PXA27x processor reset. The full-chip processor reset leaves the USB force host controller reset bit UHCHR[FHR] set. To initialize the USB host controller, use the following sequence:

1. Start the USB clocks
2. Wait at least 10 µs
3. Clear the UHCHR[FHR]

The USB host controller is then operational.

20.6.2 USB Suspend

The USB specification defines a power-conservation mechanism called UsbSuspend[2]. Devices and hubs can be placed into a suspend state; and if required, the whole system can be suspended (global suspend).

A USB device can enter the suspend state if it does not receive an SOF (keep awake) packet in greater than 3 ms. Software needs to explicitly invoke the suspend state; otherwise, devices automatically receive 1-ms “keep-awake” SOF packets.

If there is no USB activity, the host can invoke the global suspend state, which causes all connected devices and hubs to enter into their suspend states within 5 ms. The host can re-enable the bus by invoking the Resume state. If a device requires attention, it can cause a resume by sending the host a remote wake-up event. See Section 12.4.7, “Suspend and Resume” for additional information.

1. *Intel® StrongARM* SA-1111 Microprocessor Companion Chip Developer’s Manual*, Order No: 278242-003.
2. *Open Host Controller Interface Specification for USB, Release 1.0a*

20.7 Power Management

The *Open Host Controller Interface Specification* for the USB defines a port power-switching mechanism.

- All ports can be continuously powered, or the power can be switched.
- Power switching can be globally switched for all ports or individually switched.

The power switching is independent of the port state, such as its speed, or whether it is connected or enabled. Power-enable features can be operated only while the USB host clocks are running (the USB clocks must be running to disable or enable power-enable features).

This USB host controller has the following features that support power-conservation features of the Open Host Controller Interface:

- USB clock stopping
- Port power-enable
- Port-resume interrupt

20.7.1 USB Clock Stopping and Power Enable

The USB clocks can be stopped at any time; however, stopping the clocks is recommended only when the USB is in the global-suspend state, which is reached when there has been no activity on the USB for more than 5 ms, and the host controller is in the suspend state. In the suspend state, the port (single-ended receivers on the USBH_P and USBH_N pads) is also in low-power mode, but is still able to detect the USB port resume condition and asynchronously generate an interrupt to the wake-up controller of the clock unit (refer to Section 3.8.1.4, “Power Manager Wake-Up Enable Register (PWER)” on page 3-74 for details).

If a device is not connected to the USB port, the host controller cannot be suspended, and the port (PAD) is not in its low-power mode. To save power when a device is not connected to the USB port (PAD), the port power enable can be disabled before the USB clock is stopped (refer to OHCI specification). To stop the USB clocks, refer to Section 3.8.2.2, “Clock Enable Register (CKEN)” on page 3-98.

20.7.2 Port-Resume Interrupt

The port-resume interrupt does not depend on the USB or the system bus clocks to be running. This interrupt indicates that an event has occurred on the USB bus that requires the USB clocks to be restarted. If the USB clocks are not running and UHCHIE[RWIE] is set, then a port-resume event asynchronously sends an interrupt to the wake-up controller of the clock unit. Once the clocks have been restarted, a non-OHCI interrupt is sent to the interrupt controller. If the USB clocks are running, the host controller OHCI interrupt handles any interrupt conditions. For the port-resume interrupt to function, both the USB pad’s single-ended receiver and the port power supply must be powered on.

Three events can cause a UsbPortResume interrupt:

- A device is connected (indicated by one of the USB port signals being pulled high)
- A port is disconnected (indicated by both the USB port signals being pulled low)

- A remote resume signal from a currently connected device (indicated by an idle → *K*-state transition on the USB bus).

In addition to the above, an over-current condition on either port 1 or port 2 can also asynchronously be sent to the wake-up controller of the clock unit (see Section 3.8.1.4, "Power Manager Wake-Up Enable Register (PWER)" on page 3-74) to restart clocks.

20.7.3 Summary of USB Host Low-Power Operation

Table 20-2 lists the power status of each of the potential power-drain components in the USB interface for the various modes of operation. If the USB clocks, the transmitter drivers, single-ended receivers, or the differential receivers are to be turned off, software must first ensure that either there are no USB devices connected, or that the USB has been placed into the USB suspend state.

Table 20-2. Operational Power Status

Mode of Operation	Notes	Power Status				
		USB Core	Tx Driver	Differential Receiver	Single-Ended Receiver	Port Power Supply
Fully operational	1	Powered	Powered	Powered	Powered	Powered
USB global suspend	2, 3, 6	Under user control	Low power	Low power	Powered	Powered
USB port power disabled (as specified in OHCI spec).	3,4	Under user control	Low power	Low power	Low power	Low power
Processor in sleep or standby mode	5	Low power	Low power	Low power	Low power	Under user control
Processor in idle mode	2, 3, 4, 7	Under user control	Under user control	Under user control	Under user control	Under user control

NOTES:
1. Fully operational is defined as: USB host controller is in the operational state, the sleep mode is not active, and the port power is enabled.
2. The user can stop the USB clocks and still be able to receive a USB port resume interrupt when a device sends a remote resume interrupt or a device is disconnected.
3. When the PortPowerStatus bit in the UHC core is disabled the USB port is considered to be disabled, and therefore not in use, all components are put into low-power mode. Users can stop the USB clocks to further reduce power drain.
4. When the USB pads singled-ended receiver is in low-power mode, the remote resume interrupt does not function; this can also be the case if the device does not receive power from the USB port.
5. When the Intel® PXA27x processor Sleep or Standby modes are active, the USB host is disabled and does not respond to a port resume, connect, or disconnect event. USB ports must be powered off before entering either of these modes. However, the PWER register can be programmed to recognize activity on these ports, such as a cable connect, to use as a wakeup source for bringing the Intel® PXA27x processor out of Sleep or Standby modes. The USB must be re-initialized upon exiting Sleep or Standby modes.
6. A device must be connected before the UHC can be put into the suspend state.
7. The single-ended port receiver and USB power supply are powered on if the *port power enable* was enabled before the clock was stopped. The differential receivers are disabled (powered off) if the USB clocks are stopped. The USB host is capable of recognizing a remote wake-up event in this mode even if the USB host clocks are off as long as the single-ended receivers are still enabled.

20.7.4 Suggested Power-Management Routines

The following applies to all of the suggested power-management routines in this section:

- The UsbPortResume interrupt is not generated when a device is connected unless UHCHR[SSE], the Sleep Standby Enable bit, is cleared.
- To set UHCRHS[LPS], the global power enable bit, it is necessary to first start the USB clock.

20.7.4.1 Initial Port Power-Down Sequence

To initiate a port power-down sequence, follow this procedure:

1. Wait for the device's hardware reset sequence to complete. For more information on USB reset, see Section 20.6.1. The USB clock must be enabled.
2. Send a software reset to the host controller by setting UHCCOMS[HCR] and then wait 10 µs.
3. Set the global power enable bit (UHCRHS[LPS]) to turn off power to all ports.
4. Clear the Sleep Standby Enable bit (UHCHR[SSE]) to enable power to the downstream ports.
5. Stop the USB clock. The USB is now in standby mode.

Note: Software waits for the UsbPortResume interrupt before restarting the USB clock.

20.7.4.2 Low-Power Mode in Suspend State Sequence

To put the USB system into low-power mode, do the following in sequence:

1. Wait for the USB to go to global suspend state.
2. Set the port suspend bit (UHCRHPS1[PSS] or UHCRHPS2[PSS] or UHCRHPS3[PSS]). This can be automatic after 5 ms of inactivity on the USB bus.
3. Wait for the port suspend bit (UHCRHPS1[PSS] or UHCRHPS2[PSS] or UHCRHPS3[PSS]) to be set.
   The USB is now in suspend mode.
4. Clear the Sleep Standby Enable bit (UHCHR[SSE]), enabling power to the USB single-ended receivers and the USB port power supplies.
5. Stop the USB clocks.

The USB is now in standby mode.

Note: Software waits for the UsbPortResume interrupt to go active. Start the USB clock, and the host controller restarts.

20.7.4.3 Low-Power Mode after Device Disconnect Sequence

To activate low-power mode, follow this sequence:

1. Wait for port-connection status to clear (UHCRHPSx[CCS]), indicating no device is connected; or wait for the USB port resume interrupt if the USB clock is stopped and check the connection status.
2. Clear the Sleep Standby Enable bit (UHCHR[SSE]) to enable power to the downstage ports.

3. Stop the USB clocks.

The USB is now in standby mode.

Note: The port is powered down. For more information on the initial port power-down sequence, see Section 20.7.4.1. Software waits for the USB port resume interrupt before restarting the USB clock. If the USB clocks are stopped in any mode other than that described above, the host controller and any connected device might need re-initialization.

20.7.4.4 Disable USB Port

To initiate a port-disable sequence, follow this procedure:

1. Wait for the device's hardware-reset sequence to complete. For more information on USB Reset, see Section 20.6.1. The USB clock is enabled
2. Send a software reset to the host controller by setting UHCCOMS[HCR] and then wait 10 µs.
3. Clear the global power enable bit (UHCRHS[LPS]) by writing a 1 to it. The hub must be in port power switching mode. The USB is now in port power-disabled mode.
4. Clear the Sleep Standby Enable bit (UHCHR[SSE]) to enable power to the downstream ports
5. Stop the USB clocks.

The USB is now in standby mode and the port power is disabled.

Note: The UsbPortResume interrupt is not generated when a device is connected unless the Sleep Standby Enable bit is cleared. To set the global power enable bit, it is necessary to first start the USB clock.

20.8 Register Descriptions

20.8.1 UHC HCI Spec Revision Register (UHCREV)

This read-only register contains the binary-coded decimal (BCD) representation of the HCI specification version that the PXA27x processor implements. The UHC complies with the *OHCI Rev. 1.0a* specification, so this register always has a value of 0x0000_0010.

The register organization and individual bit definitions are shown in Table 20-3.

Table 20-3. UHCREV Bit Definitions

Physical Address 0x4C00_0000																UHCREV												USB Host Controller			
User Settings																															
Bit	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
	Rev																															
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	1	0	0	0	0

Bits	Access	Name	Description
31:0	R	Rev	OHCI Specification Revision Number This USB host is compliant to OHCI Rev. 1.0a, so this register reads 0x0000_0010.

20.8.2 UHC Host Control Register (UHCHCON)

The UHC Host Control (UHCHCON, shown in Table 20-4) register defines the operating modes for the host controller. Most of the fields in this register are modified only by the host controller driver, except for the HostControllerFunctionalState and RemoteWakeupConnected fields.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 20-4. UHCHCON Bit Definitions (Sheet 1 of 3)

Physical Address 0x4C00_0004 — **UHCHCON** — **USB Host Controller**

Bit	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
User Settings																																
	reserved																					RWE	RWC	IR	HCFS		BLE	CLE	IE	PLE	CBSR	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:11	—	—	reserved
10	R/W	RWE	RemoteWakeupEnable The host controller driver uses this bit to enable or disable the remote wake-up feature upon the detection of upstream resume signaling. When this bit is set and the ResumeDetected bit in HcInterruptStatus (Section 20.8.4) is set, a remote wake-up is signaled to the host system. Setting this bit has no impact on the generation of hardware interrupt. 0 = Disable remote wake-up is signaling to the host system. 1 = Enable remote wake-up is signaling to the host system.
9	R/W	RWC	RemoteWakeupConnected This bit indicates whether the UHC supports remote wake-up signaling. If remote wake-up is supported and used by the system, it is the responsibility of system firmware to set this bit after reset. The UHC clears the bit upon a hardware reset, but does not alter it upon a software reset 0 = UHC does not support remote wake-up signaling. 1 = UHC supports remote wake-up signaling.
8	R/W	IR	InterruptRouting This bit determines the routing of interrupts generated by events registered in the UHC Interrupt Status register, UHCINTS (Section 20.8.4). If clear, all interrupts are routed to the normal host bus-interrupt mechanism. If set, interrupts are routed to the system management interrupt (SMI). The host controller driver clears this bit on a hardware reset, but it does not alter this bit on a software reset. The host controller driver uses this bit as a tag to indicate the ownership of UHC. This implementation of the OHCI host controller does not support SMI. Therefore, the host controller driver must never set this bit. 0 = All interrupts are routed to the normal host bus interrupt mechanism. 1 = Interrupts are routed to the SMI.

Table 20-4. UHCHCON Bit Definitions (Sheet 2 of 3)

Physical Address 0x4C00_0004 — **UHCHCON** — **USB Host Controller**

Bit	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
User Settings																																
	reserved																					RWE	RWC	IR	HCFS		BLE	CLE	IE	PLE	CBSR	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
7:6	R/W	HCFS	HostControllerFunctionalState This field of two bits displays the current functional state of the USB host controller. 0b00 = USBRESET 0b01 = USBRESUME 0b10 = USBOPERATIONAL 0b11 = USBSUSPEND A transition to USBOPERATIONAL from another state causes an SOF generation to begin 1 ms later. The host controller driver can determine whether UHC has begun sending SOFs by reading the StartofFrame field of HcInterruptStatus (Section 20.8.4). This field can be changed by UHC only when in the USBSUSPEND state. UHC can move from the USBSUSPEND state to the USBRESUME state after detecting the resume signaling from a downstream port. HC enters USBSUSPEND after a software reset, whereas it enters USBRESET after a hardware reset. The latter also resets the root hub and asserts subsequent reset signaling to downstream ports. The *root hub* is defined as the USB hub attached directly to the USB host controller (physically it is a part of the host). Root hubs can have from 1 to 15 downstream ports (this implementation in the PXA27x processor has two downstream ports). Refer to Section 6.2 of the *OHCI Rev. 1.0a Specification* for more details.
5	R/W	BLE	BulkListEnable The host controller driver sets this bit to enable the processing of the bulk list in the next frame. If cleared by the host controller driver, processing of the bulk list does not occur after the next SOF. UHC checks this bit whenever it needs to process the list. When disabled, the host controller driver can modify the list. If HcBulkCurrentED (Section 20.8.12) is pointing to an ED to be removed, the host controller driver must advance the pointer by updating HcBulkCurrentED before re-enabling list processing. 0 = Disables processing of the bulk list after the next SOF. 1 = Enables the processing of the bulk list in the next frame.
4	R/W	CLE	ControlListEnable The host controller driver sets this bit to enable the processing of the control list in the next frame. If cleared by the host controller driver, processing of the control list does not occur after the next SOF. UHC must check this bit whenever it determines to process the list. When disabled, the host controller driver can modify the list. If HcControlCurrentED (Section 20.8.10) is pointing to an ED to be removed, the host controller driver must advance the pointer by updating HcControlCurrentED before re-enabling list processing. 0 = Disables processing of the control list after the next SOF. 1 = Enables the processing of the control list in the next frame.

Table 20-4. UHCHCON Bit Definitions (Sheet 3 of 3)

Physical Address 0x4C00_0004 — **UHCHCON** — **USB Host Controller**

Bit	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
User Settings																																
	reserved																					RWE	RWC	IR	HCFS		BLE	CLE	IE	PLE	CBSR	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
3	R/W	IE	IsochronousEnable The host controller driver uses this bit to enable/disable processing of isochronous EDs. While processing the periodic list in a frame, UHC checks the status of this bit when it finds an isochronous ED (F = 1). If set (enabled), UHC continues processing the EDs. If cleared (disabled), UHC halts processing of the periodic list, and begins processing the bulk/control lists. Setting this bit is guaranteed to take effect in the next frame (not the current frame). 0 = Disables processing of isochronous EDs. 1 = Enables processing of isochronous EDs.
2	R/W	PLE	PeriodicListEnable This bit is set to enable the processing of the periodic list in the current frame. If cleared by the host controller driver, processing of the periodic list does not halt until after the next SOF. UHC must check this bit before it starts processing the list. 0 = Disables processing of the periodic list. 1 = Enables processing of the periodic list.
1:0	R/W	CBSR	ControlBulkServiceRatio This bit specifies the service ratio between control and bulk EDs. Before processing any of the non-periodic lists, UHC must compare the ratio specified with its internal count on how many non-empty control EDs have been processed, in determining whether to continue serving another control ED or switching to bulk EDs. The internal count is retained when crossing the frame boundary. In case of reset, the host controller driver is responsible for restoring this value. The number of control EDs over bulk EDs served is as follows: 0b00 = 1:1 0b01 = 2:1 0b10 = 3:1 0b11 = 4:1

20.8.3 UHC Command Status Register (UHCCOMS)

The USB host controller uses the UHC Command Status (UHCCOMS, shown in Table 20-5) register to receive commands that the host controller driver issues, as well as reflecting the current status of the host controller. To the host controller driver, it appears to be a write-to-set register. Bits written as 1 become set in the register, while bits written as 0 remain unchanged in the register. In this way, the host controller driver can issue multiple distinct commands to the host controller without concern for corrupting previously issued commands. The host controller driver has read access to all of these bits.

Table 20-5. UHCCOMS Bit Definitions (Sheet 1 of 2)

Physical Address 0x4C00_0008 — **UHCCOMS** — **USB Host Controller**

	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
User Settings																																
Bit	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														SOC		reserved												OCR	BLF	CLF	HCR
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0

Bits	Access	Name	Description
31:18	—	—	reserved
17:16	R	SOC	SchedulingOverrunCount The SchedulingOverrunCount field indicates the number of frames with which the host controller has detected the scheduling overrun error, which occurs when the periodic list does not complete before EOF. When a scheduling-overrun error is detected, the host controller increments the counter and sets the SchedulingOverrun field in the UHC Interrupt Status register, UHCINTS. These bits are incremented on each scheduling overrun error. It is initialized to 0b00 and wraps around at 0b11. This is incremented when a scheduling overrun is detected even if SchedulingOverrun in HcInterruptStatus (Section 20.8.4) has already been set. This bit field and the scheduling overrun interrupt are used by the host controller driver to monitor any persistent scheduling problems. These bits are incremented on each scheduling overrun error.
15:4	—	—	reserved
3	R/W	OCR	OwnershipChangeRequest The host controller driver sets this bit to request a change of control of the UHC. When set, UHC sets OwnershipChange in HcInterruptStatus (Section 20.8.4). When read, this bit always returns 0. This implementation of the OHCI host does not support SMI. Therefore, software must never write 0b1 to this bit. 0 = No HCD request for a change in control of the UHC is pending. 1 = HCD is requesting a change of control of the UHC.

Table 20-5. UHCCOMS Bit Definitions (Sheet 2 of 2)

Physical Address 0x4C00_0008 — **UHCCOMS** — **USB Host Controller**

Bit	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
User Settings																																
	reserved														SOC		reserved												OCR	BLF	CLF	HCR
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0

Bits	Access	Name	Description
2	R/W	BLF	BulkListFilled This bit indicates whether there are any TDs on the bulk list. It is set by the host controller driver whenever it adds a TD to an ED in the bulk list. When UHC begins to process the head of the bulk list, it checks the BulkListFilled bit (BLF). As long as BulkListFilled is 0, UHC does not start processing the bulk list. If BulkListFilled is 1, UHC starts processing the bulk list and sets BLF to 0. If UHC finds a TD on the list, then UHC sets BulkListFilled to 1, causing the bulk list processing to continue. If no TD is found on the bulk list, and if the host controller driver does not set BulkListFilled, then BulkListFilled is still 0 when UHC completes processing the bulk list, and bulk list processing stops. 0 = Bulk list processing stops. 1 = HC starts processing the bulk list and sets BLF to 0.
1	R/W	CLF	ControlListFilled This bit indicates whether there are any TDs on the control list. It is set by the host controller driver whenever it adds a TD to an ED in the control list. When UHC begins to process the head of the control list, it checks the control list filled (CLF) bit. As long as CLF is 0, UHC does not start processing the Control list. If CF is 1, UHC starts processing the control list and sets CLF to 0. If UHC finds a TD on the list, then UHC sets CLF to 1 causing the control list processing to continue. If no TD is found on the control list, and if the host controller driver does not set CLF, then it is still 0 when UHC completes processing the control list, and control list processing stops. 0 = UHC does not start processing the control list. 1 = UHC starts processing the control list and sets ControlListFilled to 0.
0	R/W	HCR	HostControllerReset The host controller driver sets this bit to initiate a software reset of UHC. Regardless of the functional state of the UHC, it moves to the USBSUSPEND state in which most of the operational registers are reset. The exceptions to that are stated in the OHCI Spec (for example, the InterruptRouting field of the UHC Host Control register). While this bit is set, no USB host bus accesses are allowed (writes or reads of the OHCI registers). This bit is cleared by the UHC upon the completion of the reset operation (which is completed within 10 µs.). This bit, when set, does not cause a reset to the root hub and no subsequent reset signaling is asserted to its downstream ports. 0 = Controller is not in reset. 1 = Reset the host controller.

20.8.4 UHC Interrupt Status Register (UHCINTS)

The UHC INTerrupt Status (UHCINTS) register provides status on various events that cause hardware interrupts. When an event occurs, the host controller sets the corresponding bit in this register. When a bit is set, a hardware interrupt is generated if the interrupt is enabled in the HcInterruptEnable register (see Section 20.8.5) and the MasterInterruptEnable bit is set. The host controller driver can clear specific bits in this register by writing 1 to bit positions to be cleared. The host controller driver cannot set any of these bits. The host controller never clears these bits (except for during a system reset). Because of the difference in clock frequencies between the system bus and the USB host controller core, there is a latency from the time a write occurs on the system bus until the actual register bit is cleared. The HCD needs to take into this delay into consideration.

Table 20-6. UHCINTS Bit Definitions (Sheet 1 of 2)

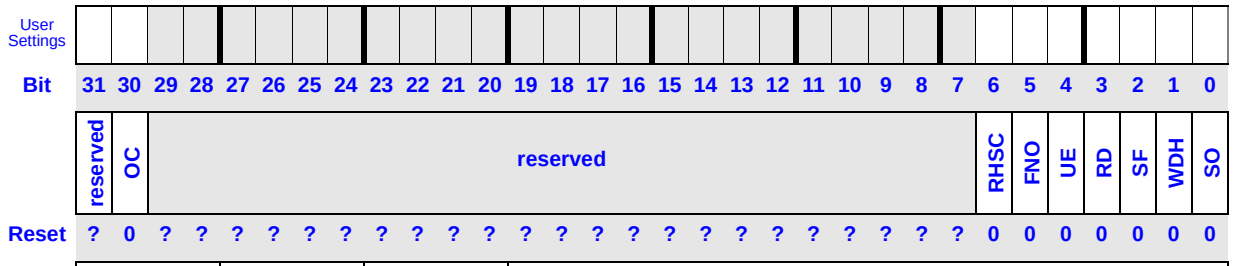
Physical Address 0x4C00_000C — **UHCINTS** — **USB Host Controller**

Bit	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
User Settings																																
	reserved	OC	reserved																							RHSC	FNO	UE	RD	SF	WDH	SO
Reset	?	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	—	—	reserved
30	R/WC†	OC	OwnershipChange This bit is set by UHC when the host controller driver sets the OwnershipChangeRequest field in HcCommandStatus (Section 20.8.3). However, this implementation of the OHCI host does not support SMI. Therefore, software must never write 0b1 to the UHCCOMS[OCR] bit. This bit is tied to 0 because the SMI pin is not implemented in this design.
29:7	—	—	reserved
6	R/WC†	RHSC	RootHubStatusChange This bit is set when the content of HcRhStatus (Section 20.8.21) or the content of any of HcRhPortStatus (Section 20.8.22) for port 1, 2, or 3 has changed. 0 = The contents of the above four registers has not changed. 1 = The contents of HcRhStatus or HcRhPortStatus[3:1] has changed.
5	R/WC†	FNO	FrameNumberOverflow This bit is set when the most significant bit (15) of HcFmNumber (Section 20.8.16) changes value, from 0 to 1or from 1 to 0, and after HccaFrameNumber has been updated. 0 = Otherwise 1 = HcFmNumber (bit 15) changed in value, from 0 to 1 or from 1 to 0, or HccaFrameNumber has been updated

Table 20-6. UHCINTS Bit Definitions (Sheet 2 of 2)

Physical Address 0x4C00_000C — **UHCINTS** — **USB Host Controller**



Bits	Access	Name	Description
4	R/WC†	UE	UnrecoverableError This bit is set when UHC detects a system error not related to USB. UHC must not proceed with any processing or signaling before the system error has been corrected. System errors such as incorrect buffer addressing offsets or condition codes (as described by section 4.3.2.3.1 on pg. 27 of the *OHCI Rev 1.0a spec.*). The HCD clears this bit by either writing a 1 to it, or by resetting the UHC OHCI core by generating a software reset (writing a 1 to the UHCCOMS[HCR] bit). 0 = No system error was detected. 1 = UHC detected a system error not related to USB.
3	R/WC†	RD	ResumeDetected This bit is set when UHC detects that a device on the USB is asserting resume-signaling. It is the transition from no-resume-signaling to resume-signaling that sets this bit. This bit is not set when HCD sets the USBRESUME state. 0 = No device is asserting resume-signaling. 1 = A device on the USB is asserting resume-signaling.
2	R/WC†	SF	StartofFrame This bit is set by UHC at each start of a frame and after the update of HccaFrameNumber. UHC also generates a SOF token at the same time. 1 = A start of a frame or an update of HccaFrameNumber has occurred
1	R/WC†	WDH	WritebackDoneHead This bit is set immediately after the UHC has written to the value of the UHCDHead to an external the memory location referred to (by the OHCI spec) as HccaDoneHead. Additional updates of the HccaDoneHead do not occur until this bit has been cleared. HCD must clear this bit only after it has saved the content of HccaDoneHead. 1 = UHC has written HcDoneHead to HccaDoneHead.
0	R/WC†	SO	SchedulingOverrun This bit is set when the USB schedule for the current frame overruns, and after the update of HccaFrameNumber. A scheduling overrun also causes the SchedulingOverrunCount of UHCCOMS to be incremented To avoid this condition, take care to enable the periodic lists as early in the frame as possible. Processing of the periodic lists occurs with no TDs scheduled. If a list is enabled at the end of the frame, and the reading of the lists occurs beyond the frame boundary the SO bit is set to 0b1 and the SOC counter is incremented. 1 = The USB schedule for the current frame has overrun or the update of HccaFrameNumber has occurred.

NOTE:
† To clear this bit, write 0b1 to it.

20.8.5 UHC Interrupt Enable Register (UHCINTE)

Each *enable* bit in the UHC Interrupt Enable (UHCINTE) register corresponds to an associated *interrupt* bit in the UHCINTS register. The UHC Interrupt Enable register controls which events generate an OHCI hardware interrupt. When a bit is set in UHCINTS, the UHC Interrupt Status register, the corresponding bit in the UHC Interrupt Enable register is set, and UHCINTE[MIE], the MasterInterruptEnable bit is set, an OCHI interrupt request is sent to the processor interrupt controller.

Writing a 1 to a bit in this register sets the corresponding bit, whereas writing a 0 to a bit in this register leaves the corresponding bit unchanged. On a read, the current value of this register is returned. To clear a bit in this register, write 0b1 to the Interrupt Disable register (UHCINTD), see Section 20.8.6.

The register organization and individual bit definitions are shown in Table 20-7. All reserved bits are read as unknown values and must be written with only a 0. A question mark indicates the value is unknown at reset.

Table 20-7. UHCINTE Bit Definitions (Sheet 1 of 2)

Physical Address 0x4C00_0010 — **UHCINTE** — **USB Host Controller**

	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
User Settings																																
Bit	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
	MIE	OC	reserved																							RHSC	FNO	UE	RD	SF	WDH	SO
Reset	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	MIE	Master Interrupt Enable 0 = Ignore 1 = Enable interrupt generation due to events specified in the other bits of this register.
30	Read/Write 1 to set	OC	Ownership Change 0 = Ignore 1 = Enable interrupt generation due to ownership change.
29:7	—	—	reserved
6	Read/Write 1 to set	RHSC	Root Hub Status Change 0 = Ignore 1 = Enable interrupt generation due to root hub status change.
5	Read/Write 1 to set	FNO	Frame Number Overflow 0 = Ignore 1 = Enable interrupt generation due to frame number overflow.
4	Read/Write 1 to set	UE	Unrecoverable Error 0 = Ignore 1 = Enable interrupt generation due to unrecoverable error.
3	Read/Write 1 to set	RD	Resume Detect 0 = Ignore 1 = Enable interrupt generation due to resume detect.

Table 20-7. UHCINTE Bit Definitions (Sheet 2 of 2)

Physical Address 0x4C00_0010 — **UHCINTE** — **USB Host Controller**

	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
User Settings																																
Bit	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
	MIE	OC	reserved																							RHSC	FNO	UE	RD	SF	WDH	SO
Reset	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

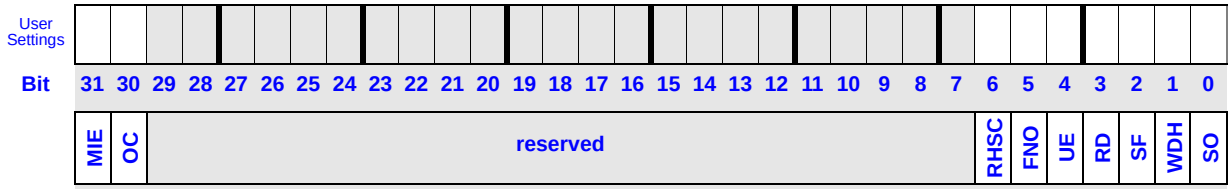
Bits	Access	Name	Description
2	Read/Write 1 to set	SF	Start of Frame 0 = Ignore 1 = Enable interrupt generation due to start of frame.
1	Read/Write 1 to set	WDH	Writeback HcDoneHead 0 = Ignore 1 = Enable interrupt generation due to HcDoneHead writeback.
0	Read/Write 1 to set	SO	Scheduling Overrun 0 = Ignore 1 = Enable interrupt generation due to scheduling overrun.

20.8.6 UHC Interrupt Disable Register (UHCINTD)

Each disable bit in the UHC Interrupt Disable (UHCINTD, shown in Table 20-8) register corresponds to an associated interrupt bit in UHCINTS, the UHC Interrupt Status register. The UHC Interrupt Disable register is coupled with the UHC Interrupt Enable register. Thus, writing a 1 to a bit in this register clears the corresponding bit in the UHC Interrupt Enable register, whereas writing a 0 to a bit in this register leaves the corresponding bit in the UHC Interrupt Enable register unchanged. On a Read, the current value of the UHC Interrupt Enable register is returned.

Table 20-8. UHCINTD Bit Definitions

Physical Address 0x4C00_0014 — **UHCINTD** — **USB Host Controller**



Bits	Access	Name	Description
31	R/W	MIE	Master Interrupt Enable 0 = Ignore 1 = Disable interrupt generation due to events specified in the other bits of this register. This field is set after a hardware or software reset
30	R/W	OC	Ownership Change 0 = Ignore 1 = Disable interrupt generation due to ownership change.
29:7	—	—	reserved
6	R/W	RHSC	Root Hub Status Change 0 = Ignore 1 = Disable interrupt generation due to root hub status change.
5	R/W	FNO	Frame Number Overflow 0 = Ignore 1 = Disable interrupt generation due to frame number overflow.
4	R/W	UE	Unrecoverable Error 0 = Ignore 1 = Disable interrupt generation due to unrecoverable error.
3	R/W	RD	Resume Detect 0 = Ignore 1 = Disable interrupt generation due to resume detect.
2	R/W	SF	Start of Frame 0 = Ignore 1 = Disable interrupt generation due to start of frame.
1	R/W	WDH	Writeback HcDoneHead 0 = Ignore 1 = Disable interrupt generation due to HcDoneHead Writeback.
0	R/W	SO	Scheduling Overrun 0 = Ignore 1 = Disable interrupt generation due to scheduling overrun.

20.8.7 UHC Host Controller Communication Area (UHCHCCA)

The UHCHCCA register contains the exact physical address of the host controller communication area.

Table 20-9. UHCHCCA Bit Definitions

Physical Address 0x4C00_0018																UHCHCCA												USB Host Controller							
User Settings																																			
Bit	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			
	HCCA																																0		
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			

Bits	Access	Name	Description
31:8	R/W	HCCA	Host Controller Communication Area Base address of the host controller communication area. This register is a pointer to an area that holds the control structures and the interrupt table that both the host controller and the host controller driver access. The minimum alignment for the HCCA is 256 bytes; therefore, bits 0 through 7 of this register always return 0 when read. Detailed descriptions are located in Chapter 4 of the *OHCI Rev. 1.0a Spec*.
7:0	R	—	Fixed at 0

20.8.8 UHC Period Current Endpoint Descriptor (UHCPCED)

The UHCPCED register contains the exact physical address of the current Isochronous or Interrupt Endpoint Descriptor. The register organization and individual bit definitions are shown in Table 20-10. The lower 4 bits are read as 0 and are unaffected by writes.

Table 20-10. UHCPCED Bit Definitions

Physical Address 0x4C00_001C																UHCPCED												USB Host Controller							
User Settings																																			
Bit	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			
	PCED																																0		
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			

Bits	Access	Name	Description
31:4	R	PCED	PeriodCurrent Endpoint Descriptor This register is used by the UHC to point to the head of one of the Periodic lists which is processed in the current frame. The content of this register is updated by the UHC after a Periodic ED has been processed. The HCD can read the content in determining which ED is currently being processed at the time of reading. This address is 32 byte aligned, therefore, the lower 4 bits are always 0
3:0	R	—	Fixed at 0

20.8.9 UHC Control Head Endpoint Descriptor (UHCCHED)

The UHCCHED register contains the physical address of the first Endpoint Descriptor of the control list. The register organization and individual bit definitions are shown in Table 20-11. The lower 4 bits are read as 0 and are unaffected by writes.

Table 20-11. UHCCHED Bit Definitions

Physical Address 0x4C00_0020	UHCCHED	USB Host Controller

User Settings																															
Bit	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
	CHED																												0			
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

Bits	Access	Name	Description
31:4	R/W	CHED	ControlHead Endpoint Descriptor The UHC traverses the control list starting with the HcControlHeadED pointer (the UHCCHED register, Section 20.8.9); the content is loaded from the HCCA during the initialization of the UHC. This address pointer is 32 byte aligned, therefore, the lower 4 bits are always 0.
3:0	R	—	Fixed at 0

20.8.10 UHC Control Current Endpoint Descriptor (UHCCCED)

The UHCCCED register contains the physical address of the current Endpoint Descriptor of the control list. The register organization and individual bit definitions are shown in Table 20-12. All reserved bits are read as 0 and are unaffected by writes.

Table 20-12. UHCCCED Bit Definitions

Physical Address 0x4C00_0024																UHCCCED												USB Host Controller			
User Settings																															
Bit	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
	CCED																												0			
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

Bits	Access	Name	Description
31:4	R/W	CCED	ControlCurrent Enpoint Descriptor This pointer is advanced to the next ED after serving the present one. The UHC continues processing the list from where it left off in the last frame. When it reaches the end of the Control list, the UHC checks the ControlListFilled bit of in HcCommandStatus (UHCCOMS[CLF]). If set, it copies the content of HcControlHeadED (the UHCCHED register) to the UHC Control Current ED (this register) and clears the UHCCOMS[CLF] bit. If UHCCOMS[CLF] is not set, the UHC does nothing. The HCD is allowed to modify this register only when the ControlListEnable bit of HcControl (UHCHCON[CLE]) is cleared. When UHCHCON[CLE] is set, the HCD only reads the instantaneous value of this register. Initially, this is set to zero to indicate the end of the Control list. This pointer/address is 32-byte aligned, therefore, the lower 4 bits are always 0.
3:0	R	—	Fixed at 0

20.8.11 UHC Bulk Head Endpoint Descriptor (UHCBHED)

The UHCBHED register contains the physical address of the first Endpoint Descriptor of the Bulk list. The register organization and individual bit definitions are shown in Table 20-13. The lower 4 bits are read as 0 and are unaffected by Writes.

Table 20-13. UHCBHED Bit Definitions

Physical Address 0x4C00_0028																UHCBHED												USB Host Controller			
User Settings																															
Bit	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
	BHED																												0			
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

Bits	Access	Name	Description
31:4	R/W	BHED	BulkHead Endpoint Descriptor The UHC traverses the Bulk list starting with the HcBulkHeadED pointer (UHCBHED register, Section 20.8.11); the content is loaded from the HCCA during the initialization of the UHC. This pointer/address is 32 byte aligned, therefore, the lower 4 bits are always 0.
3:0	R	—	Fixed at 0

20.8.12 UHC Bulk Current Endpoint Descriptor (UHCBCED)

The UHCBCED register contains the physical address of the current Endpoint of the Bulk list. The register organization and individual bit definitions are shown in Table 20-14. The lower 4 bits are read as 0 and are unaffected by Writes.

Table 20-14. UHCBCED Bit Definitions

Physical Address 0x4C00_002C	UHCBCED	USB Host Controller

User Settings																															
Bit	**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**
	BCED																												**0**			
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

Bits	Access	Name	Description
31:4	R/W	BCED	Bulk Current Endpoint Descriptor This pointer is advanced to the next ED after the UHC has served the present one. The UHC continues processing the Bulk list from where it left off in the last frame. When it reaches the end of the Bulk list, the UHC checks the ControlListFilled bit of HcControl (UHCHCON[CLF]). If set, the UHC copies the content of the UHC Bulk Head ED register to UHC Bulk Current ED and clears the UHCHCON[CLF] bit. If the UHCHCON[CLF] bit is not set, the UHC does nothing. The HCD is only allowed to modify this register when the BulkListEnable bit of the UHC Control register (UHCHCON[BLE]) is cleared. When the UHCHCON[BLE] bit is set, the HCD only reads the instantaneous value of this register. This register is initially set to zero to indicate the end of the Bulk list. As the Bulk list is served in a round-robin fashion, the Endpoints are ordered according to their insertion to the list. This address is 32 byte aligned, therefore, the lower 4 bits are always 0
3:0	R	—	Fixed at 0

20.8.13 UHC Done Head Register (UHCDHEAD)

The UHCDHEAD register contains the physical address of the last completed Transfer Descriptor that was added to the Done queue. The register organization and individual bit definitions are shown in Table 20-15. The lower 4 bits are read as 0 and are unaffected by Writes.

Table 20-15. UHCDHEAD Bit Definitions

Physical Address 0x4C00_0030	UHCDHEAD	USB Host Controller

	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
User Settings																																
Bit	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
	DHED																												0			
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

Bits	Access	Name	Description
31:4	R	DHED	DoneHead When a TD is completed, the UHC writes the content of the UHC Done Head register to the NextTD field of the TD. The UHC then overwrites the content of the UHCDHead with the address of this TD. Whenever the UHC writes the content of this register to HCCA, this register is set to zero, and the UHC also sets the WritebackDoneHead bit of the UHC Interrupt Status register (UHCINTS[WBDH]). In normal operation, the host controller driver must not need to read this register as its content is periodically written to the HCCA. This address is 32 byte aligned, therefore, the lower 4 bits are always 0.
3:0	R	—	Fixed at 0

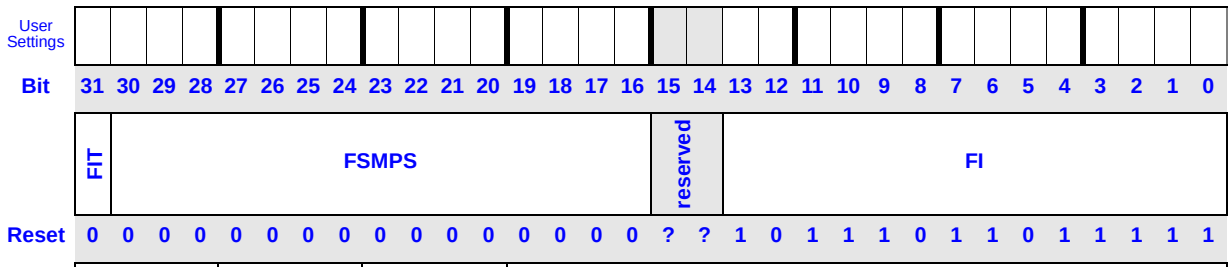
20.8.14 UHC Frame Interval Register (UHCFMI)

The UHC Frame Interval (UHCFMI) register contains a 14-bit value that indicates the bit-time interval in a frame (between two consecutive SOFs), and a 15-bit value indicating the full-speed, maximum packet size that the host controller can transmit or receive without causing scheduling overrun. The host controller driver can carry out a minor adjustment on the frame interval by writing a new value over the present one at each SOF. This mechanism provides the programmability necessary for the host controller to synchronize with an external clocking resource and to adjust any unknown local clock offset. A write to this register by the HCD takes affect in the next frame (the host controller delays updating its frame interval counter until the next SOF).

The register organization and individual bit definitions are shown in Table 20-16. All *reserved* bits are read as unknown values and must be written with only a 0. A question mark indicates the value is unknown at reset.

Table 20-16. UHCFMI Bit Definitions

Physical Address 0x4C00_0034	UHCFMI	USB Host Controller



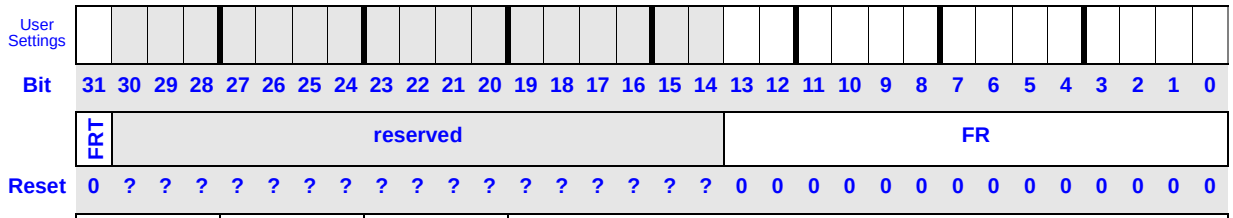
Bits	Access	Name	Description
31	R/W	FIT	FrameIntervalToggle HCD toggles this bit whenever it loads a new value to FrameInterval
30:16	R/W	FSMPS	FSLargestDataPacket This field specifies a value that is loaded into the Largest Data Packet counter at the beginning of each frame. The counter value represents the largest amount of data in bits that can be sent or received by the UHC in a single transaction at any given time without causing scheduling overrun. The field value is calculated by the HCD.
15:14	—	—	reserved
13:0	R/W	FI	FrameInterval This specifies the interval between two consecutive SOFs in bit-times. The nominal value is set at 11,999. HCD must store the current value of this field before resetting UHC (by setting the HostControllerReset field of HcCommandStatus, Section 20.8.3) as this causes the UHC to reset this field to its nominal value. The HCD can choose to restore the stored value upon the completion of the reset sequence.

20.8.15 UHC Frame Remaining Register (UHCFMR)

The UHC Frame Remaining (UHCFMR) register is a 14-bit down counter showing the bit-time remaining in the current frame.

Table 20-17. UHCFMR Bit Definitions

Physical Address 0x4C00_0038	UHCFMR	USB Host Controller



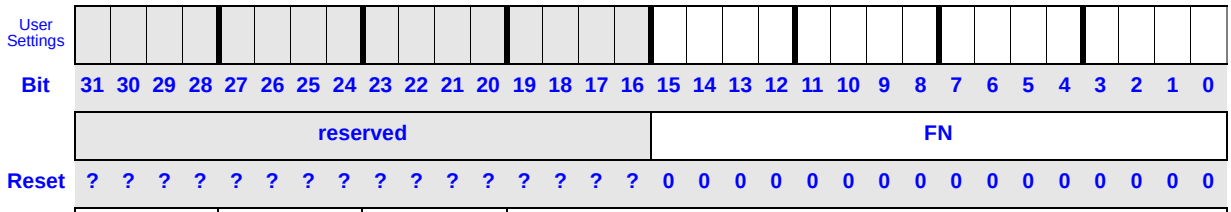
Bits	Access	Name	Description
31	R	FRT	Frame Remaining Toggle Loaded from the Frame Interval Toggle field of the UHC Frame Interval register (UHCFMI[FIT]) whenever the Frame Remaining value (UHCFMR[FR]) reaches 0. This bit is used by the HCD for the synchronization between frame interval and frame remaining.
30:14	—	—	reserved. Read as unknown and must be written as zero.
13:0	R	FR	Frame Remaining This counter is decremented at each bit-time. When it reaches zero, it is reset by loading the frame interval value specified in the UHC Frame Interval register at the next bit-time boundary. When entering the USBOPERATIONAL state, the UHC re-loads the content with the frame interval (UHCFMI[FI]) and uses the updated value from the next SOF.

20.8.16 UHC Frame Number Register (UHCFMN)

The UHC Frame Number (UHCFMN) register is a 16-bit counter which provides a timing reference.

Table 20-18. UHCFMN Bit Definitions

Physical Address 0x4C00_003C — UHCFMN — USB Host Controller



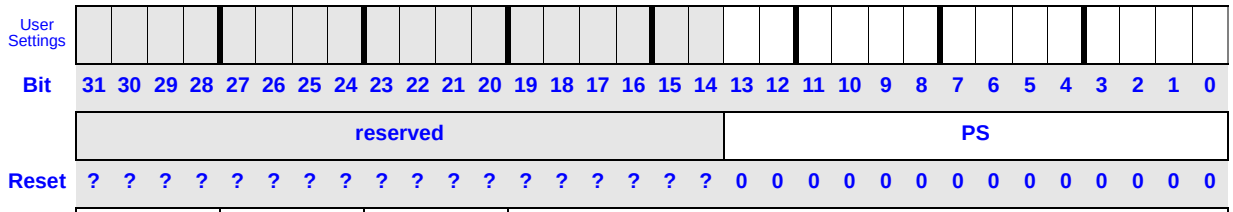
Bits	Access	Name	Description
31:16	—	—	reserved. Read as unknown and must be written as zero.
15:0	R	FN	Frame Number This 16-bit counter provides a timing reference among events happening in the host controller and the host controller driver. The host controller driver can use the 16-bit value specified in this register and generate a 32-bit Frame Number without requiring frequent register access. This is incremented when UHCFR is re-loaded. It is rolled over to 0x0 after 0xFFFF. When entering the USBOPERATIONAL state, this is incremented automatically. The content is written to the HCCA after the UHC has incremented the Frame Number at each frame boundary and sent a SOF but before the UHC reads the first ED in that Frame. After writing to the HCCA, the UHC sets the Start of Frame bit in the UHC Interrupt Status register (UHCINTS[SOF])

20.8.17 UHC Periodic Start Register (UHCPERS)

The UHC Periodic Start (UHCPERS) register has a 14-bit programmable value that determines the earliest time the UHC must start processing the periodic list.

Table 20-19. UHCPERS Bit Definitions

Physical Address 0x4C00_0040	UHCPERS	USB Host Controller



Bits	Access	Name	Description
31:14	—	—	reserved. Read as unknown and must be written as zero.
13:0	R/W	PS	PeriodicStart After a hardware reset, this field is cleared. This is then set by the HCD during the UHC initialization. The value is calculated roughly as ten percent off from the UHC frame interval value. A typical value is 0x3E67. When UHC Frame Remaining (UHCFMR[FR]) value reaches the value specified in this register, processing of the periodic lists has priority over Control/Bulk processing. The UHC therefore starts processing the interrupt list after completing the current control or bulk transaction that is in progress

20.8.18 UHC Low-Speed Threshold Register (UHCLST)

The UHC Low-Speed Threshold register is used by the UHC to determine whether to commit to the transfer of a maximum of 8-byte LS packet before EOF.

The register organization and individual bit definitions are shown in Table 20-20. All *reserved* bits are read as unknown values and must be written with only a 0. A question mark indicates the value is unknown at reset.

Table 20-20. UHCLST Bit Definitions

Physical Address 0x4C00_0044 — **UHCLST** — **USB Host Controller**

	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
User Settings																																
Bit	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																				LST											
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	1	0	0	0	1	0	1	0	0	0

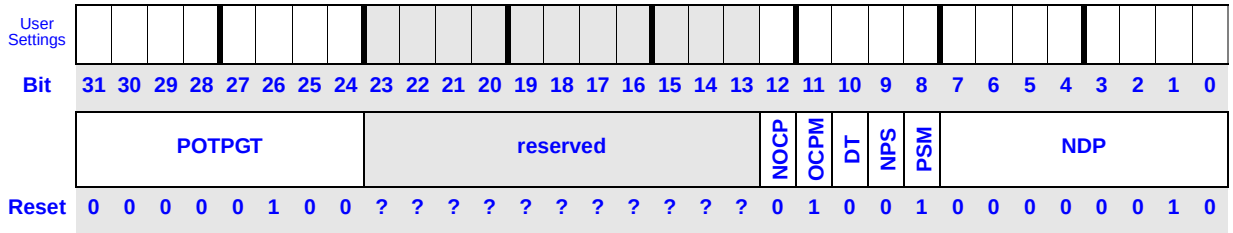
Bits	Access	Name	Description
31:12	—	—	reserved. Read as unknown and must be written as zero.
11:0	R/W	LST	LSThreshold This field contains an 12-bit value that is compared to the Frame Remaining field (UHCFMR[FR]) prior to initiating a low-speed transaction. The transaction is started only if the Frame Remaining value is greater than or equal to this field. The value is calculated by HCD with the consideration of transmission and setup overhead.

20.8.19 UHC Root Hub Descriptor A Register (UHCRHDA)

The Root Hub Descriptor A register is the first register of two describing the characteristics of the root hub. The descriptor length, descriptor type, and hub controller current fields of the Hub Class descriptor are emulated by the HCD. All other fields are located in the Root Hub Descriptor A and Root Hub Descriptor B registers.

Table 20-21. UHCRHDA Bit Definitions (Sheet 1 of 2)

Physical Address 0x4C00_0048	UHCRHDA	USB Host Controller



Bits	Access	Name	Description
31:24	R/W	POTPGT	PowerOnToPowerGoodTime The duration that the HCD must wait before accessing a powered on port (in 2-ms units)
23:13	—	—	reserved. Read as unknown and must be written as zero.
12	R/W	NOCP	NoOverCurrentProtection This bit describes if over-current protection is supported. 0 = Over-current protection is supported. 1 = No over-current protection supported.
11	R/W	OCPM	OverCurrentProtectionMode This bit describes how the over-current status for the root hub ports are reported. At reset, this field reflects the same mode as PowerSwitchingMode. This field is valid only if the No Over Current Protection field is cleared. 0 = Over-current status is reported collectively for all downstream ports (global). 1 = Over-current status is reported on a per-port basis.
10	R	DT	DeviceType This bit specifies that the root hub is not a compound device. The root hub is not permitted to be a compound device. This field always reads 0. This bit always reads 0.

Table 20-21. UHCRHDA Bit Definitions (Sheet 2 of 2)

Physical Address 0x4C00_0048 — **UHCRHDA** — **USB Host Controller**

Bit	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
User Settings																																
	POTPGT								reserved											NOCP	OCPM	DT	NPS	PSM	NDP							
Reset	0	0	0	0	0	1	0	0	?	?	?	?	?	?	?	?	?	?	?	0	1	0	0	1	0	0	0	0	0	0	1	0

Bits	Access	Name	Description
9	R/W	NPS	NoPowerSwitching This bit specifies if power-switching is supported or if ports are always powered. When this bit is cleared, the PowerSwitchingMode (UHCRHDA[PSM]) bit specifies global or per-port switching (the ports are power-switched). If this bit is set, ports are always powered on when the UHC is powered on. 0 = Ports are power-switched. 1 = Ports are always powered on when the UHC is powered on.
8	R/W	PSM	PowerSwitchingMode This bit specifies how the power switching of the root hub ports is controlled. This field is only valid if the NoPowerSwitching bit UHCRHDA[NPS] is clear. If UHCRHDA[PSM] is set, each port is powered individually. This mode allows port power to be controlled by either the global switch or per-port switching. If the Port Power Control Mask bit (UHCRHDB[PPCM]) is set, the port responds only to port power commands (Set/ClearPortPower: to Set Port Power, write 0b1 to UHCRHPS1/2/3[PPS], to Clear Port Power, write 0b1 to UHCRHPS1/2/3[LDA]). If the port mask is cleared, the port is controlled only by the global power switch (to Set/ClearGlobalPower: write 1 to UHCRHDB[LPSC] to Set Global Power, write 1 to UHCRHDB[LPS] to Clear Global Power). If the UHCRHDA[PSM] bit is cleared, all ports are powered at the same time. 0 = All ports are powered at the same time. 1 = Each port is powered individually.
7:0	R	NDP	NumberDownstreamPorts The actual number of downstream ports supported by the root hub is equal to NDP + 1. **NOTE:** Although the USB host controller supports three (3) downstream ports, this field always contains a value of 0x2.

20.8.20 UHC Root Hub Descriptor B Register (UHCRHDB)

The UHC Root Hub Descriptor B register is the second register of two describing the characteristics of the root hub. These fields are written during initialization to correspond with the system implementation. Only bits 1, 2, 17, and 18 are writable; all other bits always read as 0.

Table 20-22. UHCRHDB Bit Definitions

Physical Address 0x4C00_004C	UHCRHDB	USB Host Controller

User Settings																															
Bit	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
	PPCM																DR															
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

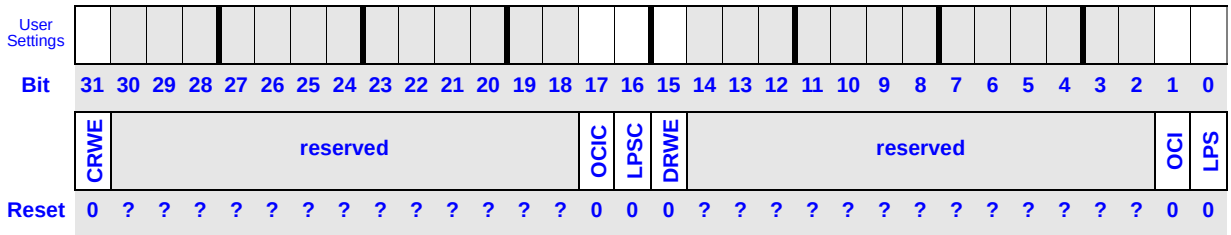
Bits	Access	Name	Description
31:16	R/W	PPCM	PortPowerControlMask Each bit in this field indicates if a port is affected by a global power control command when power switching mode (UHCRHDA[PSM]) is set. When this bit (UHCRHDB[PPCM]) is set, the port power state is affected only by per-port power control (Set/Clear Port Power). When cleared, the port is controlled by the global power switch (Set/Clear Global Power). If the device is configured to global switching mode (UHCRHDA[PSM] = 0), this field is not valid. Bit 17 corresponds to ganged-power mask on port 1, and bit 18 corresponds to ganged-power mask on port 2, and bit 19 corresponds to ganged-power mask on port 3. Bits 31 to 20, and bit 16 are not used. bit 16: Unused, always reads as 0 bit 17: Ganged-power mask on port 1 bit 18: Ganged-power mask on port 2 bit 19: Ganged-power mask on port 3 bits 20–31: Unused, always reads as 0
15:0	R/W	DNR	DeviceNotRemovable Each bit is dedicated to a port of the root hub. When cleared, the attached device is removable; when set, it is not removable. Bit 1 refers to port 1, bit 2 refers to port 2, and bit 3 refers to port 3. Bits 15–4 and bit 0 are unused. If set, this bit forces the corresponding Port Status register current connection status bit (UHCRHPS1/2/3[CSC]) to 1. bit 0: Unused, always reads as 0 bit 1: 1 if device attached to port 1 is not removable bit 2: 1 if device attached to port 2 is not removable bit 3: 1 if device attached to port 3 is not removable bits 4–15: Unused, always reads as 0

20.8.21 UHC Root Hub Status Register (UHCRHS)

The UHC Root Hub Status register is divided into two parts. The lower word of a word represents the Hub Status field and the upper word represents the Hub Status Change field.

Table 20-23. UHCRHS Bit Definitions (Sheet 1 of 2)

Physical Address 0x4C00_0050 — **UHCRHS** — **USB Host Controller**



Bits	Access	Name	Description
31	R/W	CRWE	ClearRemoteWakeupEnable Writing a 1 clears the device remote wake-up enable (UHCRHS[DRWE]) bit. Writing a 0 has no effect. This bit always reads as 0.
30:18	—	—	reserved. Read as unknown and must be written as zero.
17	R/WC†	OCIC	OverCurrentIndicatorChange This bit is set by hardware when a change has occurred to the OCI field of this register. The HCD clears this bit by writing a 1. Writing a 0 has no effect. 1 = A change has occurred to the OCI field of this register
16	Read/Write 1 to set SGP	LPSC	(read) LocalPowerStatusChange / (write) SetGlobalPower: wake-up The root hub does not support the local power status feature; thus, this bit is always read as 0. In global power mode (UHCRHDA[PSM] is clear), this bit is written to 1 to turn on power to all ports (set Goal Port Power). In per-port power mode, this bit sets PortPowerStatus (UHCRHPS1/2/3[PPS]) only on ports whose port power control mask bit (UHCRHDB[PPCM]) is not set. Writing a 0 has no effect. This bit is always read as 0.
15	Read/Write 1 to set SRWE	DRWE	(read) DeviceRemoteWakeupEnable / (write) SetRemoteWakeupEnable This bit enables a ConnectStatusChange bit (UHCRHPS1/2/3[CSC]) as a resume event, causing a USBSUSPEND-to-USBRESUME state transition and setting the resume-detected interrupt. If it reads 0, a ConnectStatusChange is not a remote wake-up event. If it is read as a 1, ConnectStatusChange is a remote wake-up event. Writing a 1 sets the device remote wake-up enable (this bit). Writing a 0 has no effect. 0 = ConnectStatusChange is not a remote wake-up event. 1 = ConnectStatusChange is a remote wake-up event.
14:2	—	—	reserved. Read as unknown and must be written as zero.

Table 20-23. UHCRHS Bit Definitions (Sheet 2 of 2)

Physical Address 0x4C00_0050 — **UHCRHS** — **USB Host Controller**

Bit	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
User Settings																																
	CRWE	reserved													OCIC	LPSC	DRWE	reserved													OCI	LPS
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
1	R	OCI	OverCurrentIndicator This bit reports over-current conditions when the global reporting is implemented. When set, an over-current condition exists. When cleared, all power operations are normal. If per-port over-current protection is implemented, this bit is always 0
0	Read/ Write 1 to Clear CGP	LPS	(read) LocalPowerStatus/ (write) ClearGlobalPower The root hub does not support the local power-status feature; thus, this bit is always read as 0. In global power mode (when UHCRHDA[PSM] is clear), this bit is set to turn off power to all ports (clear PortPowerStatus). In per-port power mode, it clears PortPowerStatus only on ports whose port power control mask bit is not set. Clearing to 0 has no effect.

NOTE:
† To clear this bit, write 0b1 to it.

20.8.22 UHC Root Hub Port Status 1/2/3 Registers (UHCRHPS1, UHCRHPS2, and UHCRHPS3)

The UHC Root Hub Port Status[3:1] registers control and report USB ports 1, 2, and 3 events on a per-port basis. The lower word of UHCRHPS1/2/3 reflects the port status, whereas the upper word reflects the status change bits. Some status bits are implemented with special write behavior (see Table 20-24). If a transaction (token through handshake) is in progress when a write to change port status occurs, the resulting port-status change must be postponed until the transaction completes.

The register organization and individual bit definitions are shown in Table 20-24.

Table 20-24. UHCRHPS1/2/3 Bit Definitions (Sheet 1 of 4)

Physical Address
0x4C00_0054
0x4C00_0058
0x4C00_005C

UHCRHPS1
UHCRHPS2
UHCRHPS3

USB Host Controller

	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
User Settings																																
Bit	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											PRSC	POCIC	PSSC	PESC	CSC	reserved						LSDA	PPS	reserved			PRS	POCI	PSS	PES	CCS
Reset	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	?	?	?	?	?	?	0	0	?	?	?	0	0	0	0	0

Bits	Access	Name	Description
31:21	—	—	reserved. Read as unknown and must be written as zero.
20	R/W	PRSC	PortResetStatusChange This bit is set at the end of the 10-ms port reset signal. A 1 indicates that the port reset has completed. A 0 indicates that the port reset is not yet complete. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. 0 = Port reset is not complete. 1 = Port reset is complete.
19	R/W	POCIC	PortOverCurrentIndicatorChange This bit is valid only if over-current conditions are reported on a per-port basis. This bit is set when root hub changes the port over-current indicator bit. The HCD writes a 1 to clear this bit. Writing a 0 has no effect 0 = No change in PortOverCurrentIndicator. 1 = PortOverCurrentIndicator has changed.
18	R/W	PSSC	PortSuspendStatusChange This bit is set by the UHC when the full resume sequence has been completed. This sequence includes the 20-ms resume pulse, LS EOP, and 3-ms re-synchronization delay. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. This bit is also cleared when PortResetStatusChange is set. 0 = Resume is not completed. 1 = Resume completed.
17	R/W	PESC	PortEnableStatusChange This bit is set when events (such as over-current condition, disconnect, switched-off power, or operational bus error-babble), cause the PortEnableStatus bit to be cleared. Changes from HCD writes do not set this bit. The HCD writes a 1 to clear this bit. Writing a 0 has no effect 0 = No change in PortEnableStatus. 0 = Change in PortEnableStatus.

Table 20-24. UHCRHPS1/2/3 Bit Definitions (Sheet 2 of 4)

Physical Address
0x4C00_0054
0x4C00_0058
0x4C00_005C

UHCRHPS1
UHCRHPS2
UHCRHPS3

USB Host Controller

Bit	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
User Settings																																
	reserved											PRSC	POCIC	PSSC	PESC	CSC	reserved						LSDA	PPS	reserved			PRS	POCI	PSS	PES	CCS
Reset	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	?	?	?	?	?	?	0	0	?	?	?	0	0	0	0	0

Bits	Access	Name	Description
16	R/W	CSC	ConnectStatusChange This bit is set by hardware whenever a connect or disconnect event occurs. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared when a set-port-reset, set-port-enable, or set-port-suspend write occurs, this bit is set to force the driver to re-evaluate the connection status since these writes must not occur if the port is disconnected. 0 = No change in CurrentConnectStatus. 1 = Change in CurrentConnectStatus. **NOTE:** If the device removable bit in the Root Hub Descriptor B register is set (indicating that this device is not removable), then the CSC bit is set only after a root hub reset to inform the system that the device is attached.
15:10	—	—	reserved. Read as unknown and must be written as zero.
9	R/WC† CPP	LSDA	(read) LowSpeedDeviceAttached / (write) ClearPortPower When read, this bit indicates the speed of the device attached to this port. When set, a low-speed device is attached to this port. When clear, a full-speed device is attached to this port. This field is valid only when the CurrentConnectStatus is set. When the HCD writes a 1 to this bit, the controller clears the PortPowerStatus bit. Writing a 0 has no effect. 0 = Full-speed device attached. 1 = Low-speed device attached.
8	R/W	PPS	(read) PortPowerStatus / (write) SetPortPower This bit reflects the PortPowerStatus, regardless of the type of power-switching implemented. This bit is cleared if an over-current condition is detected. HCD sets this bit by writing Set Port Power (writing a 1 to this bit, UHCRHPS1/2/3[PPS]) or Set Global Power (UHCRHS[LPSC]). HCD clears this bit by writing Clear Port Power (writing a 1 to UHCRHPS1/2/3[LSDA]) or Clear Global Power (UHCRHS[LPS]). 0 = Port power is off. 1 = Port power is on. The HCD writes a 0b1 to set the PortPowerStatus bit. Writing 0b0 has no effect
7:5	—	—	reserved. Read as unknown and must be written as zero.

Table 20-24. UHCRHPS1/2/3 Bit Definitions (Sheet 3 of 4)

Physical Address
0x4C00_0054
0x4C00_0058
0x4C00_005C

UHCRHPS1
UHCRHPS2
UHCRHPS3

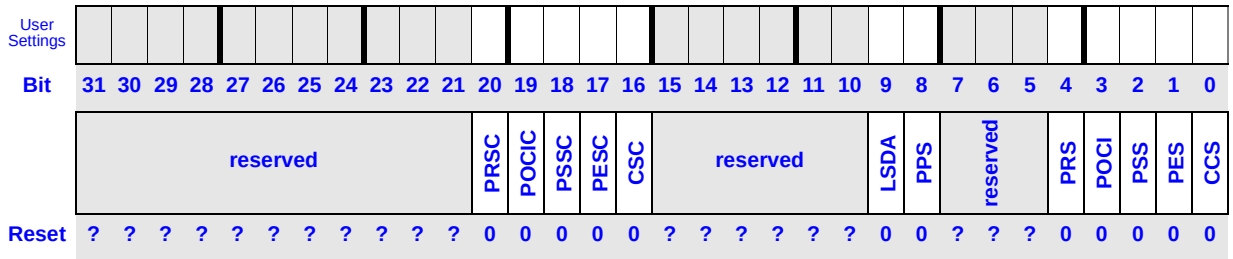
USB Host Controller

Bit	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
User Settings																																
	reserved											PRSC	POCIC	PSSC	PESC	CSC	reserved						LSDA	PPS	reserved			PRS	POCI	PSS	PES	CCS
Reset	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	?	?	?	?	?	?	0	0	?	?	?	0	0	0	0	0

Bits	Access	Name	Description
4	R/W	PRS	(read) PortResetStatus / (write) SetPortReset When this bit is set by an HCD write to set port reset, port reset signaling is asserted. When reset is completed, this bit is cleared when PortResetStatusChange is set. This bit cannot be set if CurrentConnectStatus is cleared. The HCD sets the port reset signaling by writing a 1 to this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set port-reset status, but instead sets connect-status-change. This informs the driver that it attempted to reset a disconnected port. 0 = Port reset signal is not active. 1 = Port reset signal is active.
3	R/W	POCI	(read) PortOverCurrentIndicator / (write) ClearSuspendStatus When read, this bit is valid only when the root hub is configured in such a way that over-current conditions are reported on a per-port basis. If per-port over-current reporting is not supported, this bit is forced to 0. If this bit is cleared, all power operations are normal for this port. If set, an over-current condition exists on this port. This bit always reflects the over-current input signal. The HCD writes a 1 to this bit to initiate a resume. Writing a 0 has no effect. A resume is initiated only if PortSuspendStatus is set. 0 = No over-current condition. 1 = Over-current condition detected.
2	R/W	PSS	(read) PortSuspendStatus / (write) SetPortSuspend This bit indicates if the port is suspended or in the resume sequence (1). If it is a 0, the port is not suspended or in the resume sequence. It is set by a set suspend state write and cleared when PortSuspendStatus change is set at the end of the resume interval. This bit cannot be set if CurrentConnectStatus is cleared. This bit is also cleared when PortResetStatus change is set at the end of the port reset, or when the UHC is placed in the USBRESUME state. If an upstream resume is in progress, it must propagate to the HC. The HCD sets the PortSuspendStatus bit by writing a 1 to this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortSuspendStatus; instead it sets ConnectStatusChange. This informs the driver that it attempted to suspend a disconnected port. 0 = Port is not suspended. 1 = Port is suspended.

Table 20-24. UHCRHPS1/2/3 Bit Definitions (Sheet 4 of 4)

Physical Address 0x4C00_0054 0x4C00_0058 0x4C00_005C	UHCRHPS1 UHCRHPS2 UHCRHPS3	USB Host Controller



Bits	Access	Name	Description
1	R/W	PES	(read) PortEnableStatus / (write) SetPortEnable This bit indicates whether the port is enabled (1) or disabled (0). The root hub can clear this bit when an over-current condition, disconnect event, switched-off power, or operational bus error, such as babble, is detected. This change also causes the port enabled status change bit (UHCRHPS1/2/3[PESC] to be set. The HCD sets the PortEnableStatus bit, by writing a 1 to it, and clears this bit by writing a 1 to clear port enable (UHCRHPS1/2/3[CCS], bit 0 of this register). If CurrentConnectStatus is cleared, this write does not set PortEnableStatus, but instead sets ConnectStatusChange. This informs the driver that it attempted to enable a disconnected port. Writing a 0 to the PortEnableStatus bit has no effect. This bit is also set, if not already, at the completion of a port reset when reset status change (UHCRHPS1/2/3[PRSC]) is set or port suspend when suspend status change (UHCRHPS1/2/3[PSSC]) is set 0 = Port is disabled. 1 = Port is enabled.
0	R/W	CCS	(read) CurrentConnectStatus / (write) ClearPortEnable This bit reflects the current state of the downstream port. If a device is connected, this bit reads as 1, and if no device is connected, it is 0. The HCD writes a 1 to this bit to clear the PortEnableStatus bit. Writing a 0 has no effect. The CurrentConnectStatus is not affected by any write. **NOTE:** This bit is always read as 1 when the attached device is non-removable This bit reflects the current state of the downstream port. 0 = No device connected. 1 = Device connected.

NOTE:
† To clear this bit, write 0b1 to it.

20.8.23 UHC Status Register (UHCSTAT)

This register lets users monitor the non-OHCI-specific interrupt sources. When an event occurs, if the interrupt enable bit (in UHCHIE register) for that event is set, then the corresponding bit in the UHC Status register (UHCSTAT) is set, and an interrupt is sent. UHC Status register bits are cleared only by writing a 1 to the bit. If the corresponding interrupt enable bit (in UHCHIE register) is not set, then no interrupt is sent, and the status of the event is not seen in this register. Refer to Chapter 25, “Interrupt Controller” for more details on the two USB host interrupts.

USB Power Sense Port 3—This bit reflects the state of the USB Power Sense over-current pin (USBHPWR<3>). This status bit is also reflected in the over-current condition status bits of the HCI controller (UHCRHS[OCI]), and (UHCRHPS3[POCI]) when the USB host controller clocks are running. A 1 indicates that port 3 is over current. This interrupt is sent asynchronously to the wake-up controller of the clock unit, and then, after the clocks have restarted, to the interrupt controller. HCD can clear this bit by writing a 1 to it.

System Bus Master Abort Interrupt—This bit indicates that a master abort event has occurred on a USB host-initiated system bus transfer. This interrupt is non-maskable. HCD can clear this bit by writing a 1 to it.

System Bus Target Abort Interrupt—This bit indicates that a target abort event has occurred on a USB host-initiated system bus transfer. This interrupt is non-maskable. HCD can clear this bit by writing a 1 to it.

USB Port Resume Interrupt—This bit indicates that a resume, a connect, or a disconnect event has occurred on either port 1, 2, or 3. HCD can clear this bit by writing a 1 to it. This interrupt can only occur if the USB clocks have been stopped (see Section 3.8.2.2, “Clock Enable Register (CKEN)” on page 3-98 for details). An asynchronous signal is sent to the wake-up controller of the clock unit, which then begins sending USB clocks after which, the UHC sends this interrupt to the interrupt controller.

USB Power Sense Port 2—This bit reflects the state of the USB Power Sense over-current pin (USBHPWR<2>). This status bit is also reflected in the over-current condition status bits of the HCI controller (UHCRHS[OCI]) and (UHCRHPS2[POCI]) when the USB host controller clocks are running. A 1 indicates that port 2 is over current. This interrupt is sent asynchronously to the wake-up controller of the clock unit, and then, after the clocks have restarted, to the interrupt controller. HCD can clear this bit by writing a 1 to it.

USB Power Sense Port 1—This bit reflects the state of the USB Power Sense over-current pin (USBHPWR<1>). This status bit is also reflected in the over-current condition status bits of the HCI controller (UHCRHS[OCI]) and (UHCRHPS1[POCI]) when the USB host controller clocks are running. A 1 indicates that port 1 is over current. This interrupt is sent asynchronously to the wake-up controller of the clock unit, and then, after the clocks have restarted, to the interrupt controller. HCD can clear this bit by writing a 1 to it.

HCI Transfer Abort—This bit is asserted by hardware when the host controller can no longer start or continue the current host controller interface (HCI) transfer (for example, during a reset while an HCI transfer is in progress). When asserted, it indicates that the application must terminate the transfer, and that the FIFOs are cleared. This event is active for a minimum of 10 µs. The HCD must not try to force the host controller to re-enter the Operational state from the suspend state until after a minimum of 10 µs after this interrupt occurs. The HCD can clear this bit by writing a 1 to it.

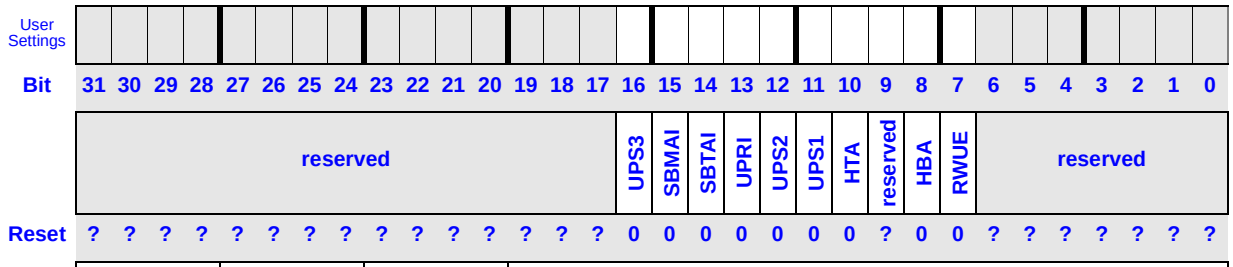
HCI Buffer Active—This bit is active while the host controller is accessing the data buffer for the current TD or when an ED is being accessed (this informs the application whether the host controller is accessing the shared memory). HCD can clear this bit by writing a 1 to it.

HCI Remote Wake-Up Event—If enabled (UHCHIE[RWIE] bit and the UHCHCON[RWE] bit are set), this bit is asserted (1) when a remote wake-up event occurs on one of the downstream ports of the root hub and the USB clocks are running. If the UHCINTE[RD] bit is set, an OHCI interrupt is also generated. HCD can clear this bit by writing a 1 to it.

The register organization and individual bit definitions are shown in Table 20-25. All reserved bits are read as unknown values and must be written with only a 0. A question mark indicates the value is unknown at reset.

Table 20-25. UHCSTAT Bit Definitions

Physical Address 0x4C00_0060	UHCSTAT	USB Host Controller



Bits	Access	Name	Description
31:17	—	—	reserved. Read as unknown and must be written as zero.
16	R/WC†	UPS3	USB Power Sense Port 3 1 = Port 3 is over current.
15	R/WC†	SBMAI	System Bus Master Abort Interrupt 1 = A master abort event has occurred on a USB host-initiated system bus transfer.
14	R/WC†	SBTAI	System Bus Target Abort Interrupt 1 = A target abort event has occurred on a USB host-initiated system bus transfer.
13	R/WC†	UPRI	USB Port Resume Interrupt 1 = A resume, a connect, or a disconnect event has occurred on either port 1, 2, or 3.
12	R/WC†	UPS2	USB Power Sense Port 2 1 = Port 2 is over-current.
11	R/WC†	UPS1	USB Power Sense Port 1 1 = Port 1 is over-current.
10	R/WC†	HTA	HCI Transfer Abort 1 = The HCI interface clear signal has gone active.
9	—	—	reserved. Read as unknown and must be written as zero.
8	R/WC†	HBA	HCI Buffer Active 1 = The host controller is accessing the shared memory
7	R/WC†	RWUE	HCI Remote Wake-Up Event 1 = A RemoteWakeUp event has occurred on one of the downstream ports of the root hub.
6:0	—	—	reserved. Read as unknown and must be written as zero.

NOTE:
† To clear this bit, write 0b1 to it.

20.8.24 UHC Reset Register (UHCHR)

This register provides software with a mechanism to reset the USB core and system bus interface. This register also controls the polarity of the Power Control and Power Sense signals. A processor reset (full chip reset) sets the Force Host Controller Reset bit (FHR) and must be cleared by the HCD for normal operation. The Clock Generation Reset bit (CGR) is used only for simulation and

test, and user software must never set this bit. This register must not be written to after the USB host begins operation unless an error condition occurs that requires a reset to the system bus interface or to the USB OHCI core (setting bits UHCHR[FSBIR] or UHCHR[FHR]).

Table 20-26. UHCHR Bit Definitions (Sheet 1 of 2)

Physical Address 0x4C00_0064 — **UHCHR** — **USB Host Controller**

Bit	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
User Settings																																
	reserved																				SSEP3	SSEP2	SSEP1	reserved	PCPL	PSPL	SSE	UIT	SSDC	CGR	FHR	FSBIR
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	1	?	0	0	1	0	0	1	1	0

Bits	Access	Name	Description
31:12	—	—	reserved. Read as unknown and must be written as zero.
11	R/W	SSEP3	Sleep Standby Enable for Port 3 0 = Enables power to the USB single-ended receivers and the USB port 3 power supply. 1 = Disables power to the USB single-ended receivers and the USB port 3 power supply.
10	R/W	SSEP2	Sleep Standby Enable for Port 2 0 = Enables power to the USB single-ended receivers and the USB port 2 power supply. 1 = Disables power to the USB single-ended receivers and the USB port 2 power supply.
9	R/W	SSEP1	Sleep Standby Enable for Port 1 0 = Enables power to the USB single-ended receivers and the USB port 1 power supply. 1 = Disables power to the USB single-ended receivers and the USB port 1 power supply.
8	—	—	reserved. Read as unknown and must be written as zero.
7	R/W	PCPL	Power Control Polarity Low This bit selects the polarity of the USBHPEN<3:1> output signals. When this bit is set, the output signals are considered active low. 0 = Polarity of the USBHPEN<3:1> output signals is active high. 1 = Output signals are considered active low.
6	R/W	PSPL	Power Sense Polarity Low This bit selects the polarity of the USBHPWR<3:1> input signals. When this bit is set, the input signals are considered active low. 0 = Polarity of the USBHPWR<3:1> input signals is active high. 1 = Input signals are considered active low.
5	R/W	SSE	Sleep Standby Enable When set, this bit disables the USB single-ended receivers and the USB port power supplies for all USB host downstream ports. 0 = Enables power to the USB single-ended receivers and the USB port power supply. 1 = Disables power to the USB single-ended receivers and the USB port power supply.

Table 20-26. UHCHR Bit Definitions (Sheet 2 of 2)

Physical Address 0x4C00_0064 — **UHCHR** — **USB Host Controller**

Bit	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
User Settings																																
	reserved																				SSEP3	SSEP2	SSEP1	reserved	PCPL	PSPL	SSE	UIT	SSDC	CGR	FHR	FSBIR
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	1	?	0	0	1	0	0	1	1	0

Bits	Access	Name	Description
4	R/W	UIT	USB Interrupt Test When set and test mode is active, this bit enables the interrupt test bits that directly force the interrupt signals to the UHCSTAT register. 0 = Disables the interrupt test bits in the UHCHIT register. 1 = Enables the interrupt test bits in the UHCHIT register.
3	R/W	SSDC	Simulation Scale Down Clock When set, this bit scales down the 1-ms interval clock in the host controller. Only used for test and simulation, it must be 0 for normal operation. 0 = Normal operation. 1 = 1-ms interval clock in the UHC is switched to 1µs.
2	R/W	CGR	Clock Generation Reset When cleared, this bit resets the host controller clock-generation block inside of the OHCI-compliant core. It can be used by the test interface to guarantee deterministic behavior for the host controller block. If cleared, this bit needs to remain active for at least 4 periods of the 12-MHz USB host clock. This bit is to be used only for simulation and testing purposes, and the host controller driver can leave this bit as inactive. This bit is reset to the inactive state, 1. 0 = Reset to the DPLL circuit is active. 1 = Reset to the DPLL is inactive.
1	R/W	FHR	Force Host Controller Reset When set, forces a reset to the USB host controller OHCI core and all 12-MHz clocked logic that interfaces to it (for example, the write FIFO). This bit must be cleared by the HCD for the host controller to function. This bit is also set active (to 1) by the processor chip-reset signal and remains in the active state until software clears this bit. This bit needs to remain active (1) for at least 10 µs. 0 = Normal operation. 1 = OHCI core and 12-MHz logic is in reset.
0	R/W	FSBIR	Force System Bus Interface Reset When a 1 is written to this bit, a reset is sent to all of the system interface logic. This includes the system bus interface, the DMA, and part of the FIFOs. The UHC automatically clears this bit after three system-bus clock cycles. This bit is used when a software or hardware reset is written to the host controller after the USB host controller has requested a system-bus cycle. The software or hardware reset sent to the host controller partially flushes the FIFOs, but this reset will be needed if the system bus interface has already requested a bus cycle. This bit always reads as 0b0.

20.8.25 UHC Interrupt Enable Register (UHCHIE)

This register can individually enable the non-OHCI-defined interrupts sent from the USB host controller to the interrupt controller. They generate only interrupts if the USB clocks have been disabled. The register organization and individual bit definitions are shown in Table 20-27.

Table 20-27. UHCHIE Bit Definitions (Sheet 1 of 2)

Physical Address 0x4C00_0068	UHCHIE	USB Host Controller

	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
User Settings																																
Bit	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																	UPS3IE	UPRIE	UPS2IE	UPS1IE	TAIE	reserved	HBAIE	RWIE	reserved						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	?	0	0	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:15	—	—	reserved. Read as unknown and must be written as zero.
14	R/W	UPS3IE	USB Power Sense Port 3 Interrupt Enable This bit enables the port 3 over-current indicator pin to cause an UHCSTAT[UPS3] type interrupt. 0 = Disable interrupt. 1 = Enable interrupt.
13	R/W	UPRIE	USB Port Resume Interrupt Enable If enabled, an interrupt is generated if a resume condition is generated on either port 1, 2, or 3. This is similar to UHCINTS[RD], the OHCI Resume Interrupt bit, except this allows the interrupt to be sent even if the USB host's 12- and 48-MHz clocks are turned off. 0 = Disable interrupt. 1 = Enable interrupt.
12	R/W	UPS2IE	USB Power Sense Port 2 Interrupt Enable This bit enables the port 2 over-current indicator pin to cause an USBHStatus[UPS2] type interrupt. 0 = Disable interrupt. 1 = Enable interrupt.
11	R/W	UPS1IE	USB Power Sense Port 1 Interrupt Enable This bit enables the port 1 over-current indicator pin to cause an USBHStatus[UPS1] type interrupt. 0 = Disable interrupt. 1 = Enable interrupt.
10	R/W	TAIE	HCI Interface Transfer Abort Interrupt Enable This corresponds to the Transfer Abort bit 10 of the USBHStatus register. When the USB host controller decides to abort a transaction, it sends out a clear signal to reset the FIFOs, their counters, and the DMA logic. 0 = Disable interrupt. 1 = Enable interrupt.
9	—	—	reserved. Read as unknown and must be written as zero.

Table 20-27. UHCHIE Bit Definitions (Sheet 2 of 2)

Physical Address 0x4C00_0068	UHCHIE	USB Host Controller

User Settings																															
Bit	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																	UPS3IE	UPRIE	UPS2IE	UPS1IE	TAIE	reserved	HBAIE	RWIE	reserved						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	?	0	0	?	?	?	?	?	?	?

Bits	Access	Name	Description
8	R/W	HBAIE	HCI Buffer Active Interrupt Enable This bit enables the HCI buffer active interrupt. 0 = Disable interrupt. 1 = Enable interrupt.
7	R/W	RWIE	HCI Remote Wake-Up Interrupt Enable This bit enables the remote wake-up interrupt. 0 = Disable interrupt. 1 = Enable interrupt.
6:0	—	—	reserved. Read as unknown and must be written as zero.

20.8.26 UHC Interrupt Test Register (UHCHIT)

The UHCHIT register is used for test purposes only. It enables the interrupt path to be exercised independently of the interrupt source (without actually creating conditions in the hardware). With any of the interrupt enable bits (UHCHIE) or any of the interrupt test bits (UHCHIT) set and the USB interrupt test bit (UHCHR[UIT]) set, these bits send an interrupt signal to the interrupt controller. The interrupts are created from rising-edge detects; therefore, software must clear an individual bit before that bit can cause a second interrupt. The exception to this requirement is bit 9 (IRQT), which does not need an enable bit. Also, bit 9 (IRQT) is a level-detect and must be cleared before the interrupt can be cleared. The UHCHIT[IRQT] bit enables software developers to create interrupts in their software so that they can debug interrupt handlers.

The register organization and individual bit definitions are shown in Table 20-28.

Table 20-28. UHCHIT Bit Definitions

Physical Address 0x4C00_006C — UHCHIT — USB Host Controller

	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
User Settings																																
Bit	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															UPS3T	SMAT	STAT	UPRT	UPS2T	UPS1T	TAT	IRQT	BAT	RWUT	reserved						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:17	—	—	reserved. Read as unknown and must be written as zero.
16	R/W	UPS3T	USB Power Sense Port3 Interrupt Test 0 = No effect. 1 = Force interrupt strobe if UHCHIE[UPS3IE] bit is set.
15	R/W	SMAT	System Bus Master Abort Interrupt Test 0 = No effect. 1 = Force interrupt strobe (non-maskable).
14	R/W	STAT	System Bus Target Abort Interrupt Test 0 = No effect. 1 = Force interrupt strobe (non-maskable).
13	R/W	UPRT	USB Port Resume Interrupt Test 0 = No effect. 1 = Force interrupt strobe if UHCHIE[UPRIE] bit is set.
12	R/W	UPS2T	USB Power Sense Port2 Interrupt Test 0 = No effect. 1 = Force interrupt strobe if UHCHIE[UPS2IE] bit is set.
11	R/W	UPS1T	USB Power Sense Port1 Interrupt Test 0 = No effect. 1 = Force interrupt strobe if UHCHIE[UPS1IE] bit is set.
10	R/W	TAT	HCI Interface Transfer Abort Interrupt Test 0 = No effect. 1 = Force interrupt strobe if UHCHIE[TAIE] bit is set.
9	R/W	IRQT	Normal OHC Interrupt Test 0 = No effect. 1 = Force interrupt strobe.
8	R/W	BAT	HCI Buffer Active Interrupt Test 1 = Force interrupt strobe if UHCHIE[BAIE] bit is set. 0 = No effect
7	R/W	RWUT	HCI Remote Wake-Up Interrupt Test 0 = No effect 1 = Force interrupt strobe if UHCHIE[RWUIE] bit is set.
6:0	—	—	reserved. Read as unknown and must be written as zero.

20.9 Register Summary

The USB host controller block contains control and status registers in addition to registers required by the OHCI specification (0x4C00_0000–0x4C00_0058). Table 20-29 lists these registers and their memory-mapped locations. All registers must be accessed as 32-bit entities.

Table 20-29. USB Host Controller Register Summary

Address	Name	Description	Page
0x4C00 0000	UHCREV	UHC HCI Spec Revision register	20-10
0x4C00 0004	UHCHCON	UHC Host Control register	20-10
0x4C00 0008	UHCCOMS	UHC Command Status register	20-14
0x4C00 000C	UHCINTS	UHC Interrupt Status register	20-16
0x4C00 0010	UHCINTE	UHC Interrupt Enable register	20-18
0x4C00 0014	UHCINTD	UHC Interrupt Disable register	20-20
0x4C00 0018	UHCHCCA	UHC Host Controller Communication Area register	20-21
0x4C00 001C	UHCPCED	UHC Period Current Endpoint Descriptor register	20-21
0x4C00 0020	UHCCHED	UHC Control Head Endpoint Descriptor register	20-22
0x4C00 0024	UHCCCED	UHC Control Current Endpoint Descriptor register	20-22
0x4C00 0028	UHCBHED	UHC Bulk Head Endpoint Descriptor register	20-23
0x4C00 002C	UHCBCED	UHC Bulk Current Endpoint Descriptor register	20-24
0x4C00 0030	UHCDHEAD	UHC Done Head register	20-25
0x4C00 0034	UHCFMI	UHC Frame Interval register	20-26
0x4C00 0038	UHCFMR	UHC Frame Remaining register	20-27
0x4C00 003C	UHCFMN	UHC Frame Number register	20-28
0x4C00 0040	UHCPERS	UHC Periodic Start register	20-29
0x4C00 0044	UHCLST	UHC Low-Speed Threshold register	20-30
0x4C00 0048	UHCRHDA	UHC Root Hub Descriptor A register	20-31
0x4C00 004C	UHCRHDB	UHC Root Hub Descriptor B register	20-33
0x4C00 0050	UHCRHS	UHC Root Hub Status register	20-34
0x4C00 0054	UHCRHPS1	UHC Root Hub Port 1 Status register	20-35
0x4C00 0058	UHCRHPS2	UHC Root Hub Port 2 Status register	20-35
0x4C00 005C	UHCRHPS3	UHC Root Hub Port 3 Status register	20-35
0x4C00 0060	UHCSTAT	UHC Status register	20-39
0x4C00 0064	UHCHR	UHC Reset register	20-41
0x4C00 0068	UHCHIE	UHC Interrupt Enable register	20-44
0x4C00 006C	UHCHIT	UHC Interrupt Test register	20-45
0x4C00 0070–0x4FFF FFFF	—	reserved	

Real-Time Clock (RTC) 21

This chapter describes the real-time clock (RTC) controller included in the PXA27x processor.

21.1 Overview

The PXA27x processor contains a real-time clock (RTC) that provides a general-purpose, real-time reference for use by the system. The RTC provides five basic functions:

- Timer
- Wristwatch
- Stopwatch
- Periodic interrupt
- Trimmer

21.2 Features

- Timer Section
  - — User-programmable, free-running counter
  - — User-programmable alarm register
  - — Resolution of one second
- Wristwatch Section
  - — User-programmable, free-running counter containing time of the day in terms of hours, minutes, seconds, day of week, week of month, day of month, month, and year
  - — User-programmable alarm registers to generate alarms in terms of hours, minutes, seconds, day of week, week of month, day of month, month, and year
  - — Resolution of one second
- Stopwatch Section
  - — Programmable counter register that contains the time elapsed between two events in terms of hours, minutes, seconds, and hundredths of a second
  - — Two user-programmable alarm registers to generate alarms in terms of hours, minutes, seconds, and hundredths of a second
  - — Resolution of one 100^{th} of a second
- Periodic Interrupt Section
  - — Programmable alarm register to generate periodic interrupts at regular intervals
  - — Resolution of one millisecond
- Trimmer Section

— User-programmable trimmer register to generate a precise 1-Hz clock for the timer and wristwatch sections

21.3 Signal Descriptions

Table 21-1 describes the one external signal associated with the RTC controller.

Table 21-1. Real-Time Clock Controller I/O Signal Descriptions

Signal Name	Type	Polarity	Description
HZ_CLK	Output	NA	1-Hz clock generated by the trimmer section of the RTC. It is available in all power modes and is used by the timer section and the wristwatch section. The accuracy of the 1-Hz clock is one 32.768-kHz clock cycle.

21.4 Operation

The functionality of the RTC controller is defined primarily by five sections:

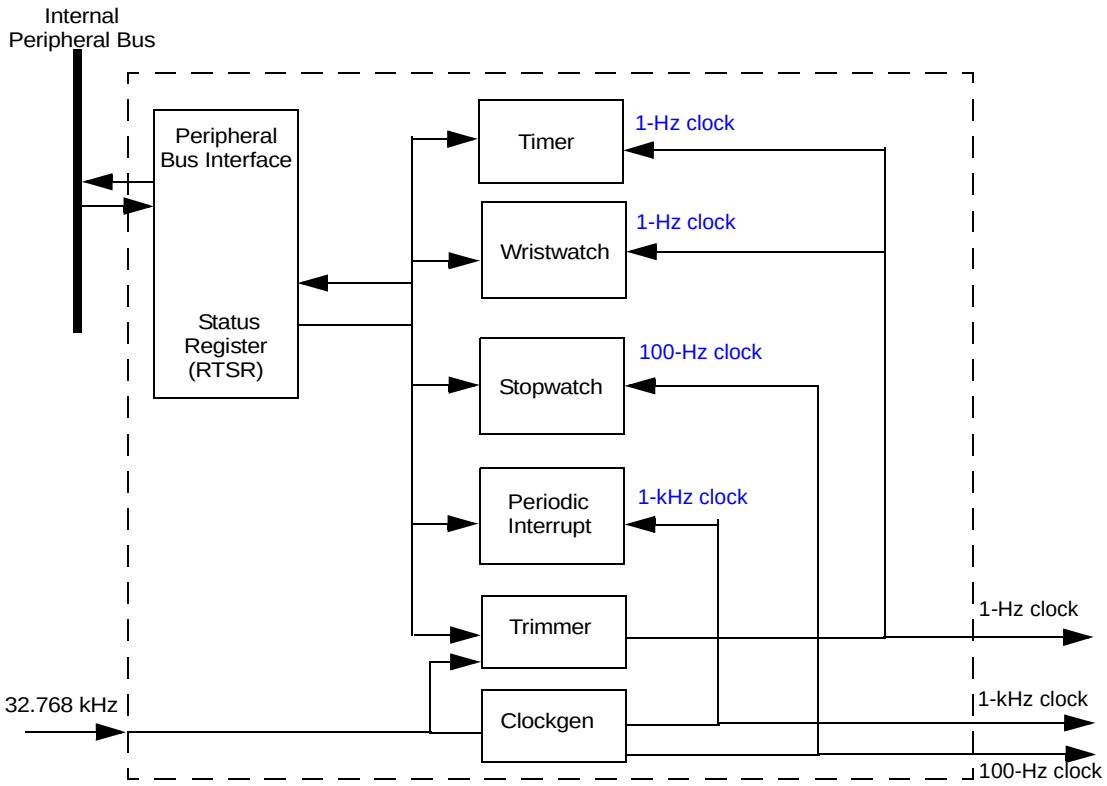
- Timer
- Wristwatch
- Stopwatch
- Periodic interrupt
- Trimmer

The peripheral bus interface block contains the RTC Status register (RTSR), which provides the interface between the core and the RTC data. Figure 21-1 is a block diagram of the RTC controller and the RTC dedicated I/O.

With the exception of the RTC Trim register, all registers in the RTC controller are reset by hardware reset and watchdog reset. The RTC Trim register (RTTR) is reset by the hardware reset only. The other resets—sleep-exit and GPIO— do not affect any of the RTC controller blocks.

The operation of all five sections, except the trimmer, is the same. Figure 21-2 shows the operational flow of one of these four identical sections.

Figure 21-1. RTC Block Diagram



Each section has one or more counter registers and one or more corresponding alarm registers. For example, the timer section has one counter register and one alarm register.

First, the desired alarm set conditions are written to the alarm register. The corresponding alarm-enable bit in the RTC Status register (RTSR) is then set. If a counter has a count-enable bit, the corresponding count-enable bit must be set to enable the counter to start counting. The count-enable bits for the corresponding counter registers reside in the RTSR. The stopwatch and periodic-interrupt sections have count-enable bits, while the timer and wristwatch sections consist of free-running counters without any count-enable bits.

As shown in Figure 21-2, when the data in the counter register and the alarm register are equal, the RTC controller signals the power manager, regardless of the state of the corresponding alarm-enable bit in the RTSR. The RTC controller then checks whether the corresponding alarm-enable bit in the RTSR has been set. If so, the corresponding alarm-detect bit in the RTSR is set, indicating an alarm has been detected. This information is forwarded to the interrupt controller, as shown in Figure 21-2. The corresponding alarm-detect bit in the RTSR is cleared by writing a one to it. This process is identical for all alarm-detection events.

Figure 21-2. Operational Flow of RTC Sections

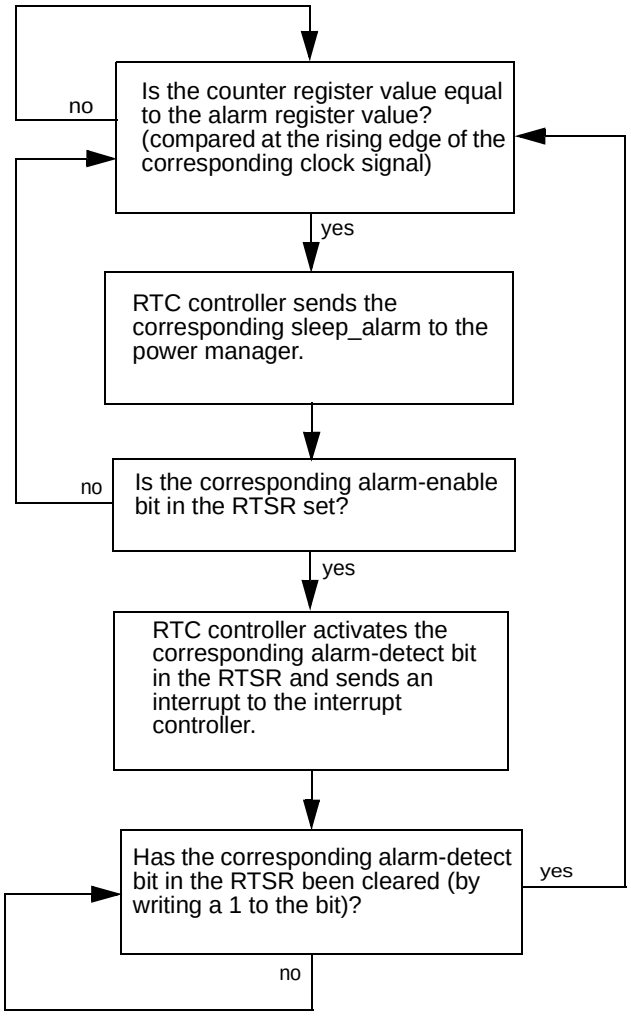


Table 21-2 lists the registers and their associated section of the RTC controller. Refer to Section 21.6 for the locations of each of these registers.

Table 21-2. RTC Controller Alarm Bit Location Summary

Section	Counter Register	Alarm Register(s)	Alarm Detect Bit	Alarm Enable Bit
Timer	RCNR	RTAR	RTSR[AL]	RTSR[ALE]
Wristwatch	RDCR RYCR	RDAR1, RYAR1 RDAR2, RYAR2	RTSR[RDAL1] RTSR[RDAL2]	RTSR[RDALE1] RTSR[RDALE2]
Stopwatch	SWCR	SWAR1 SWAR2	RTSR[SWAL1] RTSR[SWAL2]	RTSR[SWALE1] RTSR[SWALE2]
Periodic Interrupt	RTCPICR	PIAR	RTSR[PIAL]	RTSR[PIALE]
Trimmer	RTTR	N/A	N/A	N/A

21.4.1 Timer

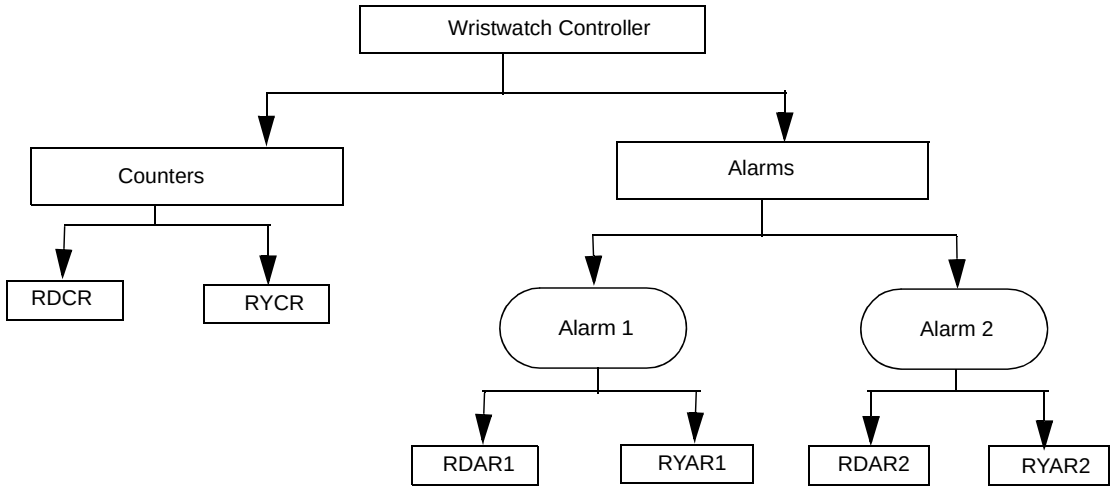
The Timer section consists of a free-running counter, the RTC Counter register (RCNR), which starts incrementing after the deassertion of hardware reset or watchdog reset. The count value is incremented at each rising edge of the 1-Hz clock. This value of this counter can be altered by writing to it. The value of the counter is unaffected by transitions into and out of sleep or idle modes.

The corresponding RTC Alarm register, RTAR, can be written with a value to be compared against the counter. On each rising edge of the 1-Hz clock, the counter is incremented and then compared to the value of RTAR. If the values match, and if the alarm-enable bit (RTSR[ALE]) is set, the corresponding alarm-detect bit (RTSR[AL]) is set.

21.4.2 Wristwatch

The wristwatch section consists of two subsections—counters and alarms—as shown in Figure 21-3.

Figure 21-3. Block Diagram of Wristwatch Section



RDCR = RTC Day Counter Register
RYCR = RTC Year Counter Register
RDAR1 = RTC Day Alarm Register 1
RYAR1 = RTC Year Alarm Register 1
RDAR2 = RTC Day Alarm Register 2
RYAR2 = RTC Year Alarm Register 2

The counters section of the wristwatch consists of two 32-bit, free-running counters (RDCR and RYCR) and two alarms in the alarms subsection. Each of the alarms consists of two 32-bit registers. Alarm 1 contains RDAR1 and RYAR1. Alarm 2 contains RDAR2 and RYAR2.

The wristwatch section runs in all power modes. The counters subsection counts the current time and year. The time of the day in terms of seconds, minutes, hours, day-of-week, and week-of-month is available through the RDCR. The year, month, and day-of-month is available through the RYCR.

Alarm 1 has a corresponding alarm-detect bit (RTSR[RDAL1]) and a corresponding alarm-enable bit (RTSR[RDALE1]). Refer to columns labeled “Alarm Detect Bit” and “Alarm Enable Bit” in Table 21-2.

Similarly, alarm 2 has a corresponding alarm-detect bit (RTSR[RDAL2]) and a corresponding alarm-enable bit (RTSR[RDALE2]). Refer to columns labeled “Alarm Detect Bit” and “Alarm Enable Bit” in Table 21-2.

21.4.2.1 Programming Wristwatch Registers

There are two counter registers (RDCR and RYCR) and four alarm registers (RDAR1, RYAR1, RDAR2, and RYAR2) in the wristwatch section.

The RDCR (see Table 21-14) and RDAR1/2 (see Table 21-9) each have five fields:

- Seconds
- Minutes

- Hour
- Day-of-week (DOW)
- Week-of-month (WOM)

The RYCR counter (see Table 21-15) and the RYAR1/2 (see Table 21-10) each have three fields:

- Day-of-month (DOM)
- Month
- Year

21.4.2.2 Allowable Values for Wristwatch Register Fields

For reference, Table 21-3 provides all the valid and invalid values for each of the fields of the counter registers and the alarm registers.

Table 21-3. Valid and Invalid Data for Wristwatch Register Fields

Field Name	Number of Bits Needed in Binary	Valid Data	Invalid Data
Seconds	6	0 to 59	60,61,62,63
Minutes	6	0 to 59	60,61,62,63
Hours	5	0 to 23	24 to 31
Day of week (DOW)	3	1 to 7	0
Week of month (WOM)	3	1 to 5	0,6,7
Day of month (DOM)	5	refer to Table 21-4	refer to Table 21-4
Month	4	1 to 12	0,13,14,15
Year	12	0 to 4095	NA

Note: The Day of Month (DOM) field is a special case. Depending on the month and the type of year, the validity of the data in this field varies. Table 21-4 lists the valid values for this field for each month.

Table 21-4. Valid Data for Day of Month (DOM) Field In RYCR

Month	Year	Valid Data	Invalid Data
January, March, May, July, August, October, December	Any	1 to 31	0
April, June, September, November	Any	1 to 30	0,31
February	leap year	1 to 29	0,30,31
	nonleap year	1 to 28	0,29,30,31

For the months January, March, May, July, August, October, and December, the count can go from 1 to 31, because there are 31 days in these months. Since 31 is the maximum decimal number for the DOM field, five bits are needed to represent this number in binary. With five bits, the maximum decimal number that can be represented is 31. Zero is the only invalid data for these months.

For the months April, June, September, and November, the count goes from 1 to 30, because there are only 30 days in these particular months. The invalid data in this case is 31 and zero.

If the year is a leap year, the count in February goes from 1 to 29; and the invalid data in this case is 30, 31, and zero. If the year is not a leap year, the count in February goes from 1 to 28; and the invalid data in this case is 29, 30, 31, and zero.

21.4.2.3 Effects of Data Written to Wristwatch Register Fields

This section describes the effects of writing the wristwatch register fields with valid and invalid data. The discussion is separated for counter registers and alarm registers. Zero and non-zero data is described separately for the alarm registers.

21.4.2.3.1 Writing RDCR and RYCR Counter Registers with Valid Data

The counter registers are updated only when there is a write to the RTC Day Counter register (RDCR). When the write for RYCR is executed, the new data for RYCR is first written into an internal register. This new data is *only* written into the RYCR register when a write to the RDCR occurs. If a write to the RDCR is not issued, the new data in the internal register is never written into RYCR, and users read the original data from RYCR register. Therefore, the protocol to be followed in each of the three possible cases follows.

- Update the RYCR only and not the RDCR:
  - — Write the RYCR with new data—which writes the new data into the internal register.
  - — Read the RDCR.
  - — Write the RDCR with data just read—new data in the internal register is written into the RYCR and the current data from the RDCR is re-written.
- Update the RDCR only and not the RYCR:
  - — Write the RDCR with new data—the RYCR (with current data) and the RDCR (with new data) is written.
- Update both the RDCR and the RYCR:
  - — Write the RYCR with new data—which writes the new data into the internal register.
  - — Write the RDCR with new data—new data in the internal register is written into the RYCR and the new data from the RDCR is written.

21.4.2.3.2 Writing Counter Registers with Invalid Data

Any attempt to write invalid or incorrect data to the counter registers results in unpredictable behavior.

Valid and correct data must be written to the wristwatch registers (RDCR and RYCR) initially. Subsequently, the registers are normally to be treated as read-only registers. Only rarely do these registers need updating.

21.4.2.3.3 Writing Alarm Registers with Valid Data

Alarm 1 consists of the alarm registers RDAR1 and RYAR1. Alarm 2 consists of the alarm registers RDAR2 and RYAR2. A write to these alarm registers with valid data follows the same procedures as writing valid data to the counter registers. Refer to Section 21.4.2.3.1 for the proper procedure.

21.4.2.3.4 Writing Alarm Registers with Invalid Data (Other than Zero)

Except for the WOM field, writing invalid data to the alarm registers results in no match ever occurring between the counter and alarm registers.

For the RDARx[WOM] fields, the invalid data is 6 and 7. If 6 is written to the RDARx[WOM] field, the first and third weeks are matched. If 7 is written to the RDARx[WOM] field, the second and fourth weeks are matched.

An exception in the use of the alarm registers is if RYARx[DOM] contains valid data, RDSRx[DOW] and RDSRx[WOM] are ignored for matching the data.

For example, the following data is written:

- RDAR1[SECONDS] contains 0b00_0000
- RDAR1[MINUTES]contains 0b00_0000
- RDAR1[HOURS]contains 0b00_0000
- RDAR1[DOW] contains 0b111
- RDAR1[WOM]contains 0b110
- RYAR1[DOM] contains 0b0_0001
- RYAR1[MONTH] contains 0b0001
- RYAR1[MONTH] contains 0b0111_1101_0000

In this example, the RYAR1[DOM] field contains valid data. As a result, RDAR1[DOW] and RDAR1[WOM] are ignored and the alarm is set on 01/01/2000 at 0:00:00 hours.

If invalid data is written to RYARx[DOM], the data in RDARx[DOW] and RDARx[WOM] is considered only for generating the alarm. For example, using the same data as the example above, RYAR1[DOM] instead contains invalid data. Since the data written in RDAR1[DOW] is 7 (Saturday) and the data written in RDAR1[WOM] field is 6, the alarm is set on the first and the third Saturday of the first month (January) in the year 2000 at 0:00:00 hours.

21.4.2.3.5 Writing Alarm Registers with Invalid (Zero) Data

In some of the fields of the alarm registers, zero is an invalid value. The effects of zero in the fields of the alarm registers are as follows:

- Hours, Minutes and Seconds fields—Zero is valid for these fields.
- Day-Of-Week (DOW), or Week-Of-Month (WOM), Day of Month (DOM), Month or Year fields—Zero is not valid for these fields. If zero is written into any of these fields, it is ignored while generating the alarm. For example, if a zero is written into a DOW field, the alarm is set every day at the time written in the Hours, Minutes, and Seconds field. If zeroes are written into all the fields of the alarm register, the alarm occurs at 0:00:00 hours every day.

  If the same data is written (as given in the previous item) in all fields except for the Month field, where a zero was entered. In this case, the hardware sets the alarm every month. So, the alarm will be set on the first day of every month in the year 2000 at 0:00:00 hours.

Note: The hour field works in 24-hour mode only. Software must convert the time to 12-hour mode, if needed.

21.4.3 Stopwatch

The stopwatch section consists of a Stopwatch Counter register (SWCR) that measures the elapsed time between two events. This counter is activated with the corresponding stopwatch count-enable bit, RTSR[SWCE].

The Stopwatch has two alarm registers: Stopwatch Alarm register1 (SWAR1) and Stopwatch Alarm register 2 (SWAR2). The corresponding alarm-detect bits and alarm-enable bits are given in Table 21-2. The stopwatch counter is clocked with a 100-Hz clock signal incrementing SWCR at each rising edge of the 100-Hz clock signal.

To reset the SWCR counter to zero, write a new value to SWAR1. Any value written to SWAR1 register clears SWCR to zero.

To use the stopwatch function as a stopwatch:

1. Set the stopwatch count-enable bit (RTSR[SWCE]). This starts the SWCR counter incrementing.
2. Clear RTSR[SWCE]—the counter stops counting and SWCR contains the elapsed time between the setting and clearing of RTSR[SWCE].
3. (optional) If RTSR[SWCE] is set again, the counter starts counting from where it stopped.

If another session is desired, (that is, with the counter starting to count from zero), a new value must be written into SWAR1.

21.4.3.1 Interval Interrupts

The SWCR can also generate an interval interrupt. The required elapsed time is written to an alarm register (SWAR1 or SWAR2), and then set the corresponding alarm-enable (RTSR[SWALE1] or RTSR[SWALE2]) is set. Set the stopwatch count-enable bit (RTSR[SWCE]). When the alarm register and the counter register are equal, the corresponding alarm-detect bit (RTSR[SWAL1] or RTSR[SWAL2]) is set by the RTC controller logic.

Note: The value written to SWAR1 or SWAR2 is the elapsed time required, not the time (based on the current time) when the alarm should trigger. This is because any write to a Stopwatch Alarm register 1 (SWAR1) clears the Stopwatch Counter register (SWCR).

For example:

- 60 minutes is written into SWAR1 (0x0007_8000) and 90 minutes into SWAR2 (0x0013_4000).
- The alarm-enable bits are set (RTRS[SWALE1] and RTSR[SWALE2].
- The count-enable bit (RTSR[SWCE]) is set, which enables the counter (the count starts from zero).
- When the count in the SWCR reaches 60 minutes, the first alarm-detect bit in the RTSR[SWAL1] is set.
- The counter continues counting. When the count in the SWCR reaches 90 minutes, the second alarm-detect bit in the RTSR[SWAL2] is set.
- If the count enable bit (RTSR[SWCE]) is still set, the counter continues to count. When the RTSR[SWCE] is cleared, the counter stops counting.

In this example, if RTSR[SWCE] had been cleared when the counter reached 100 minutes, the counter would have stopped counting and SWCR would contain a value of 100 minutes.

Whenever new data is written into SWAR1, SWCR is reset, and a new session begins. For example, in the above example, after clearing RTSR[SWCE], if new data—say, 70 minutes—is written into SWAR1, the counter resets from 100 minutes to zero and is ready for a new session. Setting RTSR[SWCE] again would start the counter incrementing from zero.

Note: Whenever a new session is required, RTSR[SWCE] must be set only after the new data is written into the alarm register. The delay between the two writes must be at least two CPU clock cycles.

21.4.3.2 Programming Stopwatch Registers

There is one counter register (SWCR) and two alarm registers (SWAR1and SWAR2) for the stopwatch.

The SWCR (Table 21-16), SWAR1 Alarm register (Table 21-11), and SWAR2 Alarm register (Table 21-11) each have four different fields: Hundredths, Seconds, Minutes, and Hour.

Table 21-5 provides the valid and invalid values for each of the fields of the counter register and the alarm registers for the stopwatch.

Table 21-5. Valid and Invalid Data for SWCR, SWAR1, and SWAR2

Field Name	Number of Bits Needed in Binary	Valid Data	Invalid Data
Hundredths	7	0 to 99	100 to 127
Seconds	6	0 to 59	60,61,62,63
Minutes	6	0 to 59	60,61,62,63
Hours	5	0 to 23	24 to 31

21.4.3.2.1 Writing Counter Register (SWCR) with Invalid Data

Writing invalid data into any field of the counter register results in unpredictable behavior of the RTC controller.

21.4.3.2.2 Writing Alarm Registers (SWAR1 and SWAR2) with Invalid Data

Writing invalid data to the alarm registers prevents a match between the counter register and the alarm register.

21.4.4 Periodic Interrupt

The periodic interrupt controller section of the RTC controller generates alarms periodically, depending on the interval programmed into the Periodic Interrupt Alarm register (PIAR). The counter is clocked on the positive edge of a 1-kHz clock. The Periodic Interrupt Counter register (RTCPICR) contains the current value of the periodic interrupt counter.

The periodic interrupt counter only increments when the periodic interrupt count enable bit (RTSR[PICE]) is set. If disabled, (by clearing RTSR[PICE]), the counter stops counting at the value it reaches. If then re-enabled (by setting RTSR[PICE]), the counter continues counting from the value it had previously reached.

The periodic interrupt alarm-enable bit (RTSR[PIALE]) must be set for the periodic interrupt alarm bit (RTSR[PIAL]) to be set. Whenever the Periodic Interrupt Counter register (RTCPICR) equals the Periodic Interrupt Alarm register (PIAR), RTSR[PIAL] is set (if RTSR[PIALE] is set), and the RTCPICR resets to zero. This process repeats as long as RTSR[PICE] remains set.

For example, if the PIAR is written with 78 (0x4E), an alarm occurs (the RTSR[PIAL] is set) every 78 ms if RTSR[PIALE] is set. Whenever a new session is required, a write to the Periodic Interrupt Alarm register (PIAR) must occur. The PICE bit must be enabled only after PIAR is written with new data. There must be at least two CPU cycles delay between the two actions.

Note: If the PIAR is zero and the periodic interrupt alarm enable bit (RTSR[PIALE]) is set, the hardware behavior is unpredictable. Zero is not a valid value for the Periodic Interrupt Counter register. The general use of this alarm is to generate interrupts to the processor in sleep mode. Therefore, a valid value (non-zero) must be written into PIAR. The maximum value that can be written is 65,535. The correct use of the periodic interrupt alarm detect bit is as follows

- First, write to PAIAR (the Match register) with a non-zero value. This also resets the PICR (the counter)
- IF a user wants the RTSR[PIAL] bit (alarm-detect bit) to be set to 1 after wake-up, then write a 1 to the RTSR[PIALE] bit.
- If a user prefers an RTC interrupt to the core after wake-up, then the RTC interrupt-enable bits in the interrupt controller block must be set.
- Assert the PICE so that the counter increments

21.4.5 Trimmer

The trimmer section generates the 1-Hz clock by dividing down the 32.768-kHz crystal oscillator output by approximately 32,768.

The inherent inaccuracies of crystals, aggravated by varying capacitance of the board connections, cause the time base to be somewhat inaccurate, requiring a periodic adjustment in the 1-Hz clock period. The PXA27x processor, through the RTTR, allows the 1-Hz timebase to be trimmed to an accuracy of ± 5 seconds per month. The trimming procedure is described in Section 21.4.5.1.

21.4.5.1 Trim Procedure

The 1-Hz clock driving the RTC is obtained by dividing the output of the oscillator multiplexor. At reset, the RTTR contains 0x0000_7FFF, which yields an approximate 1-Hz clock. When the divider count in RTTR[CLK_DIV] is set to zero, the 1-Hz clock driving the RTC maintains a high-level signal. Setting RTTR[CLK_DIV] = 0x0001 divides the frequency of the oscillator multiplexor output by two. The RTTR is reset to its default value of 0x0000_7FFF each time the hardware reset bit is asserted.

To generate the value to be entered into the RTTR, the output frequency of the oscillator multiplexor must first be measured (approximately 32 kHz) using an accurate timebase, such as a frequency counter. Refer to Chapter 24, “General-Purpose I/O Controller” for details on the procedure to select the appropriate GPIO alternate function to make the oscillator multiplexor externally visible. The trim is accomplished by dividing the output of the oscillator by an integer value, and then performing fine-grain fractional adjustment by periodically deleting clocks from the stream driving this integer divider.

21.4.5.2 RTTR Value Calculations

After the true oscillator frequency is known, it must be split into integer and fractional portions. The integer portion of the value (minus one) is written into RTTR[CLK_DIV]. This value is compared against a 16-bit counter clocked by the output of the oscillator multiplexor (approximately 32 kHz). The counter resets and generates a pulse when the two values are equal. This pulse constitutes the raw 1-Hz signal.

The fractional part of the adjustment is performed by periodically deleting clocks from the clock stream driving the integer counter. The period, called the *trim interval*, is hardwired to be 2^{10}–1 seconds (approximately 17 minutes). The number of clocks deleted, called the *trim delete value*, is a 10-bit programmable counter allowing from 0 to (2^{10}–1) 32-kHz clocks to be deleted from the input-clock stream once per trim interval. RTTR[DEL] represents the number of clocks deleted per trim operation. In summary, every 2^{10} –1 seconds, the integer counter stops clocking for a period equal to the fractional error that has accumulated. If this counter is programmed to zero, no trim operations occur, and the RTC is clocked with the raw 32-kHz clock. The relationship between the nominal 1-Hz clock frequency and the nominal 32-kHz clock (f1 and f32K, respectively) is shown in the following equation.

$$f1 = \frac{(2\text{^}10\text{-}1)*(RTTR[CK\_DIV]+1) - RTTR[DEL]}{(2\text{^}10\text{-}1)*(RTTR[CK\_DIV]+1)} * \frac{f32k}{(RTTR[CK\_DIV]+1)}$$

where:

f1 = HZ_CLK frequency

f32k = RTC internal clock—either the 32.678-kHz crystal output or the 13-MHz crystal output divided down to 32.754 kHz

RTTR[DEL] = RTTR(25:16)

RTTR[CK_DIV] = RTTR(15:0)

21.4.5.2.1 Trim Example #1—Measured Value Has No Fractional Component

In this example, the desired HZ_CLK frequency is 1 Hz. The oscillator output is measured as 36045.000 cycles/s (Hz). This output is exactly 3277 cycles over the nominal frequency of the crystal (32.768 kHz) and has no fractional component. As such, only the integer trim function is needed—no fractional trim is required. Accordingly, RTTR[CK_DIV] is loaded with the binary equivalent of 36045–1, or 0x0000_8CCC. RTTR[DEL] is left at zero (power-up state) to disable fractional trimming. This trim exercise leaves an error of zero in trimming.

21.4.5.2.2 Trim Example #2—Measured Value Has a Fractional Component

This example is more common in that the measured frequency of the oscillator has a fractional component. Again, the desired HZ_CLK output frequency is 1 Hz. If the oscillator output is measured as 32768.92 cycles/s (Hz), an integer trim is necessary so that the *average* number of cycles counted before generating one 1-Hz clock is 32768.92. Similar to the previous example, the integer field RTTR[CK_DIV] is loaded with the hexadecimal equivalent of 32768–1 or 0x7FFF (reset value).

Because the actual clock frequency is 0.92 cycles per second faster than the integer value, the 1-Hz clock generated by just the integer trimming is slightly faster than needed and must be slowed down. Accordingly, program the fractional trim to delete 0.92 cycles per second on average to bring the 1-Hz output frequency down to the proper value. Since the trimming procedure is performed every 1023 (2^{10}–1) seconds, the trim must be set to delete 941.16 clocks every 1023

seconds (.92 x 1023 = 941.16). Load RTTR[DEL] with the hexadecimal equivalent of 941, or 0x3AD. The fractional component of this value cannot be trimmed out and constitutes the error in trimming, described below.

This trim setting leaves an error of 0.16 cycles per 1023 seconds. The error calculation yields (in parts-per-million or ppm):

$$\text{Error} = \frac{0.16 \text{ cycles}}{1023 \text{ sec}} X \frac{1 \text{ sec}}{32768 \text{ cycles}} = 0.002 \text{ ppm}$$

21.4.5.2.3 Maximum Error Calculation Versus Real-Time Clock Accuracy

As seen from trim example #2, the maximum possible error approaches 1 clock per 2^{10}–1 seconds. Calculating the ppm error for this scenario yields:

$$\text{Error (maximum)} = \frac{1 \text{ cycle}}{1023 \text{ sec}} X \frac{1\text{sec}}{1024 \text{ sec}} = 0.03 \text{ ppm}$$

To maintain an accuracy of ± 5 seconds per month, the required accuracy is calculated to be:

$$\text{Error} = \frac{5 \text{ sec}}{\text{month}} X \frac{1 \text{ month}}{2592000 \text{ sec}} = 1.9 \text{ ppm}$$

This calculation indicates the 1-Hz clock output can be made very accurate through the use of the trim procedure. Likewise, use the trim procedure to compensate for a range of factors that can affect crystal oscillators. Such factors can include, but are not limited to:

- Manufacturing and supplier variance in the crystals
- Crystal aging effects
- System voltage differences
- System manufacturing variance

The trim procedure can counteract these factors by providing a highly accurate mechanism to remove the variance and shifts from the manufacturing and static environment variables on an individual system level. However, since this is a calibration solution, it is not a practical solution for dynamic changes in the system and environment and can most likely only be done in a factory setting due the equipment required.

21.4.6 Low-Power Modes

Predefined RTC events result in the PXA27x processor exiting from low-power modes.

21.4.6.1 RTC Recovery from Idle Mode

The PXA27x processor exits from idle mode when an RTC interrupt is generated if ICPR[RTC_AL] is set to one and any of the following groups of RTC conditions occur:

- RTSR[PICE] is set to one and RTSR[PIALE] is set to one and a match between PICR and PIAR occurs.
- RTSR[SWCE] is set to one and RTSR[SWALE1] is set to one and a match between SWCR and SWAR1 occurs.
- RTSR[SWCE] is set to one and RTSR[SWALE2] is set to one and a match between SWCR and SWAR2 occurs.
- RTSR[ALE] is set to one and a match between RCNR and RTAR occurs.
- RTSR[RDALE1] is set to one and matches between both (RDCR, RYCR) and (RDAR1, RYAR1) occur, respectively.
- RTSR[RDALE2] is set to one and matches between both (RDCR, RYCR) and (RDAR2, RYAR2) occur, respectively.

21.4.6.2 RTC Recovery from Standby, Sleep, and Deep-Sleep Modes

The PXA27x processor exits from standby, sleep, and deep-sleep modes (deep-sleep mode must have been entered using a software write to the PWRMODE register) if PWER[WERTC] is set to one and any of the following conditions occur:

- If RTSR[PICE] is set to one and a match between PICR and PIAR occurs. Wake-up of the processor occurs regardless of the state of the RTSR[PIALE] bit.
- If RTSR[SWCE] is set to one and a match between SWCR and SWAR1 occurs. Wake-up of the processor occurs regardless of the state of the RTSR[SWALE1] bit.
- If RTSR[SWCE] is set to one and a match between SWCR and SWAR2 occurs. Wake-up of the processor occurs regardless of the state of the RTSR[SWALE2] bit.
- Anytime a match between RCNR and RTAR occurs. Wake-up of the processor occurs regardless of the state of the RTSR[ALE] bit.
- Anytime matches between both (RDCR, RYCR) and (RDAR1, RYAR1) occur, respectively. Wake-up of the processor occurs regardless of the state of the RTSR[RDALE1] bit.
- Anytime matches between both (RDCR, RYCR) and (RDAR2, RYAR2) occur, respectively. Wake-up of the processor occurs regardless of the state of the RTSR[RDALE2] bit.

21.5 Register Descriptions

The RTC controller contains 15 registers used for the control and communication of the status of the RTC controller.

Most registers in the RTC controller are read/write registers. Intel strongly recommends that the operating system use the core's memory management unit (MMU) protection mechanisms to prevent inadvertent writes to the RCNR. For more information about the MMU, see the *Intel XScale® Core Developer's Manual*.

Because of the asynchronous nature of the 1-Hz clock relative to the processor clock, writes to these registers are controlled by a hardware mechanism which delays the actual write until the data can be properly synchronized. In the case of multiple writes to RTC counter and alarm registers in quick succession, the final update to the RTC register may be delayed by a maximum of six 32 kHz clock cycles.

The RTC counter and alarm registers can be read at any time. Reads reflect the value in the register after it increments or after it is written.

21.5.1 RTC Trim Register (RTTR)

RTTR, defined in Table 21-6, configures the frequency of the 1-Hz clock. The reset value of this register (0x0000_7FFF) is such that a perfect 32.768-kHz crystal would result in a 1-Hz clock. (See Section 21.4.5.1 for details on how to calculate the value for this register.)

The RTTR register is only reset by hardware reset. To ensure the validity of the data written into the RTTR, RTTR[31] is used as a lock bit. The data in RTTR can be changed only if RTTR[LCK] is zero. Once RTTR[LCK] is set, only a hardware reset can clear the RTTR.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 21-6. RTTR Bit Definitions

Physical Address 0x4090_000C — **RTTR** — **RTC Controller**

	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
User Settings																																
Bit	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
	LCK	reserved					DEL										CK_DIV															
Reset	0	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Access	Name	Description
31	R/W	LCK	Lock for RTTR 0 = Value of RTTR can be overwritten. 1 = Value of RTTR cannot be overwritten. **NOTE:** Once LCK has been set, the value of RTTR can be changed only after a hardware reset.
30:26	—	—	reserved
25:16	R/W†	DEL	Trim Delete Count The number of 32-kHz clocks to delete when clock trimming begins.
15:0	R/W†	CK_DIV	Clock Divider Count The integer portion of the true 32-kHz oscillator frequency minus one. Example: If true frequency = 32768.92 Hz, then CLK_DIV = 32768 –1 = 32767 = 0x7FFF.
† These bit fields can be written to only if RTTR[LCK] is clear.			

21.5.2 RTC Status Register (RTSR)

The bits within RTSR, defined in Table 21-7, are categorized as alarm-enable bits, count-enable bits, and alarm-detect bits.

- The alarm-enable bits enable and disable the alarm functions of the RTC. The RTSR contains the alarm-enable bits for the timer (ALE), wristwatch (RDALE1 and RDALE2), and stopwatch (SWALE1 and SWALE2). The RTSR also contains a 1-Hz clock edge-detection-enable bit (HZE).
- The count-enable bits enable and disable the counter functions of the RTC. The RTSR contains two count-enable bits: one for the stopwatch (SWCE) and one for the periodic interrupt (PICE). The counters increment only if their corresponding count-enable bits are set.
- The alarm-detect bits indicate whether the corresponding alarm has occurred. The alarm-detect bits are set by the RTC controller logic if the corresponding enable bits are set and the alarm conditions have been met. The alarm-detect bits in this register are routed to the interrupt controller where they can be enabled to cause a second-level interrupt. The alarm-detect bits are reset by writing 0b1 to the bits to be cleared. The RTSR contains alarm-detect bits for the timer (AL), wristwatch (RDAL1 and RDAL2), stopwatch (SWAL1 and SWAL2), periodic interrupt (PIAL), and the 1-Hz clock edge-detect (HZ).

When the PXA27x processor is in the sleep mode, the alarm-detect bit in the RTSR is updated if an RTC alarm is detected and the corresponding alarm-enable bit in the RTSR is set.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 21-7. RTSR Bit Definitions (Sheet 1 of 3)

Physical Address 0x4090_0008	RTSR	RTC

User Settings																															
Bit	**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																**PICE**	**PIALE**	**PIAL**	**SWCE**	**SWALE2**	**SWAL2**	**SWALE1**	**SWAL1**	**RDALE2**	**RDAL2**	**RDALE1**	**RDAL1**	**HZE**	**ALE**	**HZ**	**AL**
Reset	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Bits	Access	Name	Description
31:16	—	—	reserved
15	R/W	PICE	Periodic Interrupt Count Enable for RTCPICR Count Register 0 = RTCPICR counter incrementing is disabled. 1 = RTCPICR counter is incrementing is enabled.
14	R/W	PIALE	Periodic Interrupt Alarm Enable 0 = PIALE is not enabled; PIAL is not set. 1 = PIALE is enabled; when there is a match between PIAR and RTCPICR, then PIAL is set.
13	R/WC†	PIAL	Periodic Interrupt Alarm Status 0 = Periodic interrupt alarm not detected. 1 = Alarm match occurred and PIALE is enabled.
12	R/W	SWCE	Stopwatch Count Enable for SWCR Count Register 0 = SWCR counter incrementing is disabled. 1 = SWCR counter incrementing is enabled.

Table 21-7. RTSR Bit Definitions (Sheet 2 of 3)

	Physical Address 0x4090_0008												RTSR															RTC			
User Settings																															
Bit	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																PICE	PIALE	PIAL	SWCE	SWALE2	SWAL2	SWALE1	SWAL1	RDALE2	RDAL2	RDALE1	RDAL1	HZE	ALE	HZ	AL
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
11	R/W	SWALE2	Stopwatch Alarm Enable for Stopwatch Alarm 2 0 = Stopwatch alarm 2 is not enabled. 1 = Stopwatch alarm 2 is enabled.
10	R/WC†	SWAL2	Stopwatch Alarm 2 Status 0 = No stopwatch alarm 2 has been detected. 1 = Stopwatch alarm 2 has been detected (SWAR2 matches SWCR) and SWALE2 is set.
9	R/W	SWALE1	Stopwatch Alarm Enable for Stopwatch Alarm 1 0 = Stopwatch alarm 1 is not enabled. 1 = Stopwatch alarm 1 is enabled.
8	R/WC†	SWAL1	Stopwatch Alarm 1 Status 0 = No stopwatch alarm 1 has been detected. 1 = Stopwatch alarm 1 has been detected (SWAR1 matches SWCR) and SWALE1 is set.
7	R/W	RDALE2	Wristwatch Alarm Enable for Wristwatch Alarm 2 0 = Wristwatch alarm 2 is not enabled. 1 = Wristwatch alarm 2 is enabled.
6	R/WC†	RDAL2	Wristwatch Alarm 2 Status 0 = No wristwatch alarm 2 has been detected. 1 = Wristwatch alarm 2 has been detected (RDAR2 matches RDCR and RYAR2 matches RYCR) and RDALE2 bit is set.
5	R/W	RDALE1	Wristwatch Alarm Enable for Wristwatch Alarm 1 0 = Wristwatch alarm 1 is not enabled. 1 = Wristwatch alarm 1 is enabled.
4	R/WC†	RDAL1	Wristwatch Alarm 1 Status 0 = No wristwatch alarm 1 has been detected. 1 = Wristwatch alarm 1 has been detected (RDAR1 matches RDCR and RYAR1 matches RYCR) and RDALE1 bit is set.
3	R/W	HZE	HZ interrupt Enable 0 = The HZ interrupt is not enabled. 0 = The HZ interrupt is enabled.
2	R/W	ALE	RTC Alarm interrupt Enable 0 = The RTC alarm interrupt is not enabled. 1 = The RTC alarm interrupt is enabled.

Table 21-7. RTSR Bit Definitions (Sheet 3 of 3)

Physical Address 0x4090_0008	RTSR	RTC

User Settings																															
Bit	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																PICE	PIALE	PIAL	SWCE	SWALE2	SWAL2	SWALE1	SWAL1	RDALE2	RDAL2	RDALE1	RDAL1	HZE	ALE	HZ	AL
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
1	R/WC†	HZ	HZ Rising Edge Detected 0 = No 1-Hz rising edge has been detected. 1 = A 1-Hz rising edge has been detected and HZE is set.
0	R/WC†	AL	RTC Alarm Detected 0 = No RTC alarm has been detected. 1 = An RTC alarm has been detected (RTAR matches RCNR) and RTSR[ALE] bit is set.

† To clear this bit, write 0b1 to it.

21.5.3 RTC Alarm Register (RTAR)

RTAR, defined in Table 21-8, is a 32-bit register. Following each rising edge of the 1-Hz clock, this register is compared to the RCNR. If the two are equal and RTSR[ALE] is set, then RTSR[AL] is set.

Table 21-8. RTAR Bit Definitions

Physical Address 0x4090_0004	RTAR	RTC

User Settings																															
Bit	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
	RTMV																															
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

Bits	Access	Name	Description
31:0	R/W	RTMV	RTC Target Match Value The value compared with the RTC counter (RCNR).

21.5.4 RTC Wristwatch Day Alarm Registers (RDARx)

The RDARx registers are defined in Table 21-9. Following each rising edge of the 1-Hz clock, these registers along with the Wristwatch Year Alarm registers (RYAR1/2) are compared to the RDCR and RYCR, respectively. If the conditions result in a match and the corresponding Wristwatch Alarm Enable bit (RTSR[RDALE1/2]) is set, the RTC controller logic sets the corresponding Wristwatch Alarm Detect bit, RTSR[RDAL1/2].

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 21-9. RDAR1/2 Bit Definitions

Physical Address	RDAR1 / RDAR2	RTC
0x4090_0018 0x4090_0020		

	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
User Settings																																
Bit	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									WOM			DOW			HOURS					MINUTES						SECONDS					
Reset	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:23	—	—	reserved
22:20	R/W	WOM	Match Value for Week of Month
19:17	R/W	DOW	Match Value for Day of Week
16:12	R/W	HOURS	Match Value for Hours
11:6	R/W	MINUTES	Match Value for Minutes
5:0	R/W	SECONDS	Match Value for Seconds

21.5.5 RTC Wristwatch Year Alarm Registers (RYARx)

RYARx is defined in Table 21-10. Following each rising edge of the 1-Hz clock, these registers along with the Wristwatch Day Alarm registers (RDAR1/2) are compared to the RYCR and RDCR, respectively. If the conditions result in a match and the corresponding Wristwatch Alarm Enable bit (RTSR[RDALE1/2]) is set, the RTC controller logic sets the corresponding Wristwatch Alarm Detect bit, RTSR[RDAL1/2].

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 21-10. RYAR1/2 Bit Definitions

Physical Address	RYAR1 / RYAR2	RTC
0x4090_001C 0x4090_0024		

User Settings																															
Bit	**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											**YEAR**												**MONTH**				**DOM**				
Reset	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Bits	Access	Name	Description
31:21	—	—	reserved
20:9	R/W	YEAR	Match Value for Year Count, from 1 to 4096
8:5	R/W	MONTH	Match Value for Month
4:0	R/W	DOM	Match Value for Day of Month

21.5.6 RTC Stopwatch Alarm Registers (SWARx)

The RTC Stopwatch Alarm registers (SWARx), defined in Table 21-11, are 32-bit registers. Following each rising edge of the 100-Hz clock, these registers are compared to the SWCR. If either match and the corresponding Stopwatch Alarm Enable bit (RTSR[SWALE1/2]) is set, the RTC controller logic sets the corresponding Stopwatch Alarm Detect bit, RTSR[SWAL1/2].

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 21-11. SWAR1/2 Bit Definitions

Physical Address		SWAR1		RTC	
0x4090_002C		SWAR2			
0x4090_0030					

User Settings

Bit	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								HOURS					MINUTES							SECONDS					HUNDRETHS							
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

Bits	Access	Name	Description
31:24	—	—	reserved
23:19	R/W	HOURS	Match Value for Stopwatch Time in Hours
18:13	R/W	MINUTES	Match Value for Stopwatch Time in Minutes
12:7	R/W	SECONDS	Match Value for Stopwatch Time in Seconds
6:0	R/W	HUNDRETHS	Match Value for Stopwatch Time in Hundredths of a Second

21.5.7 RTC Periodic Interrupt Alarm Register (PIAR)

PIAR, defined in Table 21-12, is a 32-bit register. Following each rising edge of the 1-kHz clock, this register is compared to the RTCPICR. If the two are equal and RTSR[PIALE] is set, then RTSR[PIAL] is set, and RTCPICR is reset to zero. This process repeats as long as the count-enable RTSR[PICE] remains set.

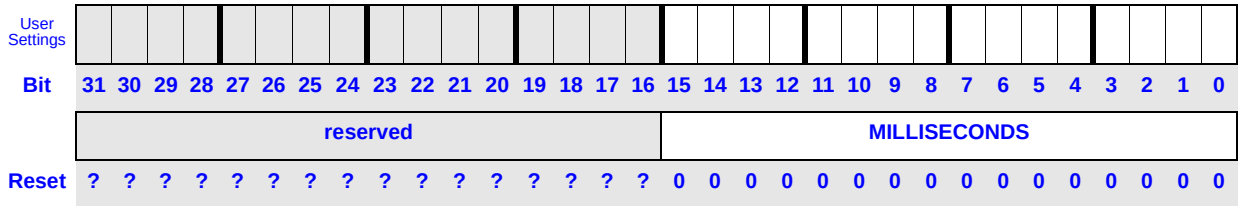
Any write to the PIAR resets the periodic interrupt counter and RTCPICR to zero. The RTSR[PICE] bit must be enabled only after PIAR is written with new data. There must be at least two CPU cycles delay between the two actions.

Note: Zero is an invalid value for the PIAR when RTSR[PICE] is set and results in unpredictable behavior. The maximum value that can be written is 65535.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 21-12. PIAR Bit Definitions

Physical Address 0x4090_0038 — **PIAR** — **RTC**



Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R/W	MILLISECONDS	Match value for the periodic interrupt time in milliseconds. This value is compared with the periodic interrupt counter (RTCPICR).

21.5.8 RTC Counter Register (RCNR)

RCNR, defined in Table 21-13, reflects the current value of the RTC counter.

RCNR is incremented at each rising edge of a 1-Hz clock signal, which is obtained after trimming the approximate 32-kHz clock signal in the trimmer section. A crystal clock oscillator provides the 32.768-kHz clock.

Table 21-13. RCNR Bit Definitions

	Physical Address 0x4090_0000																RCNR														RTC							
User Settings																																					
Bit	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
	RCV																															
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

Bits	Access	Name	Description
31:0	R/W	RCV	Count Value Current value of the RTC counter in seconds.

21.5.9 RTC Day Counter Register (RDCR)

RDCR, defined in Table 21-14, reflects the current time in seconds, minutes, hours, day-of-the-week, and week-of-the-month.

This Week-Of-Month (WOM) and Day-Of-Month (DOM) fields are used together to represent the current day and week of the month. For example: DOW = 3 indicates Tuesday and WOM = 1 indicates the first week of the month, so together this indicates Tuesday of the first week of the month.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 21-14. RDCR Bit Definitions (Sheet 1 of 2)

	Physical Address 0x4090_0010																RDCR														RTC							
User Settings																																					
Bit	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									WOM			DOW			HOURS					MINUTES								SECONDS							
Reset	?	?	?	?	?	?	?	?	?	0	0	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

Bits	Access	Name	Description
31:23	—	—	Reserved
22:20	R/W	WOM	Week of Month—1 (first week) through 5 (fifth week)
19:17	R/W	DOW	Day of Week—1 (Sunday) through 7 (Saturday)

Table 21-14. RDCR Bit Definitions (Sheet 2 of 2)

	Physical Address 0x4090_0010																RDCR												RTC															
User Settings																																											
Bit	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									WOM			DOW				HOURS				MINUTES								SECONDS															
Reset	?	?	?	?	?	?	?	?	?	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										

Bits	Access	Name	Description
16:12	R/W	HOURS	Hours—0 to 23
11:6	R/W	MINUTES	Minutes—0 to 59
5:0	R/W	SECONDS	Seconds—0 to 59

21.5.10 RTC Year Counter Register (RYCR)

RYCR, defined in Table 21-15, contains the current time in the day-of-the-month, month, and year. The year count starts at 0 and ends at 4095.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 21-15. RYCR Bit Definitions

	Physical Address 0x4090_0014																RYCR												RTC															
User Settings																																											
Bit	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											YEAR												MONTH				DOM															
Reset	?	?	?	?	?	?	?	?	?	?	?	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1										

Bits	Access	Name	Description
31:21	—	—	reserved
20:9	R/W	YEAR	Current Year—0 to 4095
8:5	R/W	MONTH	Month—1 (January) through 12 (December)
4:0	R/W	DOM	Day of Month—starting with 1. Zero is not allowed. The maximum count varies, based on the month and the type of year. Table 21-4 lists the allowable values.

21.5.11 RTC Stopwatch Counter Register (SWCR)

SWCR, defined in Table 21-16, contains the elapsed time of the stopwatch in hours, minutes, seconds, and hundredths of a second.

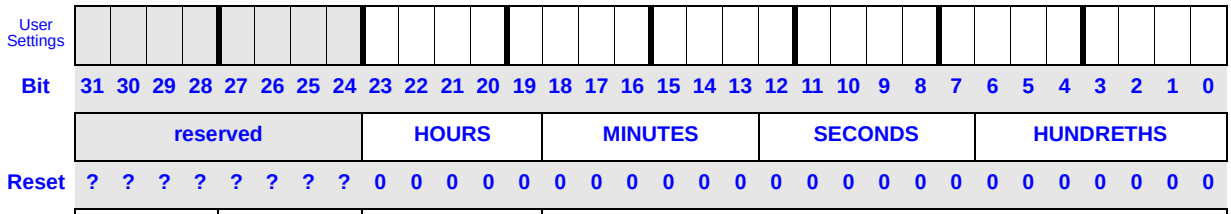
SWCR increments only when the count-enable bit (RTSR[SWCE]) is set. While RTSR[SWCE] is set, the counter continuously counts. When the counter reaches the maximum possible time (24 hours minus 1/100sec), it reverts to zero and starts counting again.

SWCR can be programmed to be used as a stopwatch or to generate an interval interrupt.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 21-16. SWCR Bit Definitions

Physical Address 0x4090_0028	SWCR	RTC



Bits	Access	Name	Description
31:24	—	—	reserved
23:19	R/W	HOURS	Number of Elapsed Hours in Stopwatch Counter
18:13	R/W	MINUTES	Number of Elapsed Minutes in Stopwatch Counter
12:7	R/W	SECONDS	Number of Elapsed Seconds in Stopwatch Counter
6:0	R/W	HUNDRETHS	Hundreths of a Second in Stopwatch Counter

21.5.12 RTC Periodic Interrupt Counter Register (RTCPICR)

RTCPICR, defined in Table 21-17, contains the current count of the 1-kHz periodic interrupt counter and is clocked on the positive edge of the 1-kHz clock. The RTCPICR increments only when the periodic interrupt count enable bit (RTSR[PICE]) is set.

If RTSR[PICE] is disabled, the periodic interrupt counter stops incrementing at the value last reached. If re-enabled, the counter resumes counting from the value it stopped at previously.

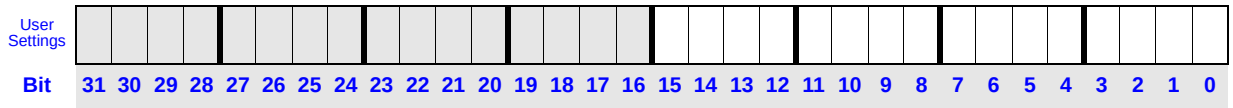
Whenever the RTCPICR equals the PIAR and RTSR[PIALE] is set, RTSR[PIAL] is set by the RTC controller logic, and RTCPICR resets to zero. This process repeats as long as RTSR[PICE] is set.

Whenever a new session (count started from zero) is required, the periodic interrupt counter must be reset to zero by executing a write to the PIAR.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 21-17. RTCPICR Bit Definitions

Physical Address 0x4090_0034	RTCPICR	RTC Controller



Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R/W	MILLISECONDS	Periodic Interrupt Time (in milliseconds)

21.6 Register Summary

Table 21-18 describes the location of the real-time clock registers.

Table 21-18. RTC Controller Register Summary

Address	Name	Description	Page
0x4090_0000	RCNR	RTC Counter register	21-24
0x4090_0004	RTAR	RTC Alarm register	21-19
0x4090_0008	RTSR	RTC Status register	21-17
0x4090_000C	RTTR	RTC Timer Trim register	21-16
0x4090_0010	RDCR	RTC Day Counter register	21-24
0x4090_0014	RYCR	RTC Year Counter register	21-25
0x4090_0018	RDAR1	RTC Wristwatch Day Alarm register 1	21-20
0x4090_001C	RYAR1	RTC Wristwatch Year Alarm register 1	21-21
0x4090_0020	RDAR2	RTC Wristwatch Day Alarm register 2	21-20
0x4090_0024	RYAR2	RTC Wristwatch Year Alarm register 2	21-21
0x4090_0028	SWCR	RTC Stopwatch Counter register	21-26
0x4090_002C	SWAR1	RTC Stopwatch Alarm register 1	21-22
0x4090_0030	SWAR2	RTC Stopwatch Alarm register 2	21-22
0x4090_0034	RTCPICR	RTC Periodic Interrupt Counter register	21-27
0x4090_0038	PIAR	RTC Periodic Interrupt Alarm register	21-23
0x4090_003C– 0x409F_FFFC	—	Reserved	—

Operating System Timers 22

This chapter describes the operating-system timers included in the PXA27x processor.

22.1 Overview

The operating-system timers block provides a set of timer channels that allows software to generate timed interrupts (or wake-up events). In the PXA27x processor, these interrupts are generated by two sets of timer channels. The first set, which provides one counter and four match registers, is clocked from a 3.25-MHz clock. The other set, which provides eight counters and eight match registers, can be clocked from either the 32.768-kHz timer clock, a 13-MHz clock, or an externally supplied clock, providing a wide range of timer resolutions. The latter set of interrupts can be used as a wake-up source, if wake-up from a power island (PI) domain event is enabled in the Power Manager Wake Enable register (PWER[WEP1] is set) (see Table 3-17, “PWER Bit Definitions” on page 3-74).

22.2 Features

The timers provide the following features:

- Single counter operating at 3.25 MHz
- Four Match registers
- Watchdog function.

The PXA27x processor also provides eight additional timer channels that supply the following additional features:

- Eight independent channels, each consisting of
  - Counter
  - Match register
  - Control register
- Independent clock for each counter, selectable by software
  - 32.768-kHz clock for low power
  - 13-MHz clock for high accuracy
  - Externally-supplied clock for network synchronization
- Counter resolutions of 1/32768th of a second, 1 millisecond, 1 second, and 1 microsecond
- Periodic and one-shot timers
- Two external synchronization events
- Operation during reduced-power modes (standby, sleep, and deep-sleep modes)

22.3 Signal Descriptions

This section describes the external signals the timers use. Table 22-1 summarizes the signals.

Table 22-1. Operating System Timers I/O Signal Descriptions

Name	Type	Description
EXT_SYNC<1:0>	Input	External synchronization signals that reset timers. These signals are provided by the GPIO block and must be held low until the appropriate alternate functions have been programmed. Once programmed, these signals reflect the state of the external pin. When the OMCRx[S] field is enabled for either of these external-synchronization signals, the corresponding OSCRx register is reset when a rising edge is detected on the appropriate EXT_SYNC signal (See Table 22-2, Table 22-3, and Table 22-4 for information on these registers).
CHOUT<1:0>	Output	CHOUT<1> is a periodic output clock generated from channel 11. CHOUT<0> is a periodic output clock generated from channel 10. These active-high output signals generate output clocks from timer channels 11 and 10, respectively, if these channels have been programmed to be periodic. The signal for an active channel programmed as periodic changes state each time a match occurs. The signal is low for any non-periodic channel.

22.4 Operation

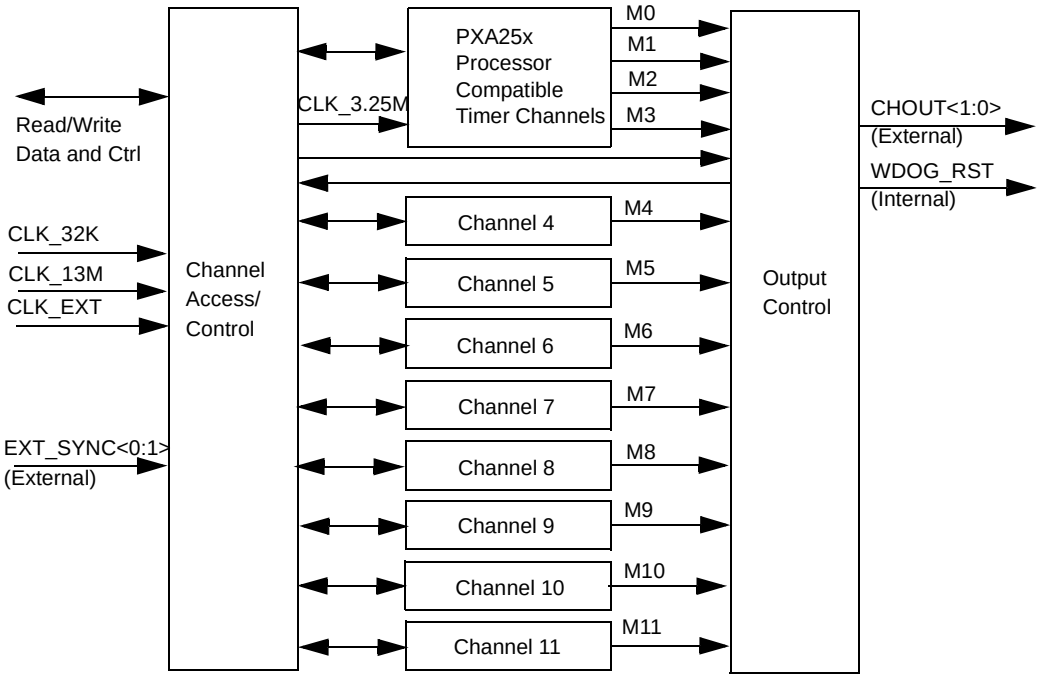
This section describes the operation of the operating system timers. Throughout this section, the terms *compare* and *match* describe when a counter register, OSCRx, is being compared to a match register, OSMRx.

- A compare occurs at the rising edge of every corresponding clock, CKCx.
- A match occurs when a compare is being performed, and the value in OSCRx is the same as the value in OSMRx. A match triggers an interrupt if the corresponding bit is set in the OIER register (see Table 22-7).

22.4.1 Block Diagrams

Figure 22-1 shows the OS timers block.

Figure 22-1. Operating System Timers Block Diagram



22.4.1.1 Channel Access and Control

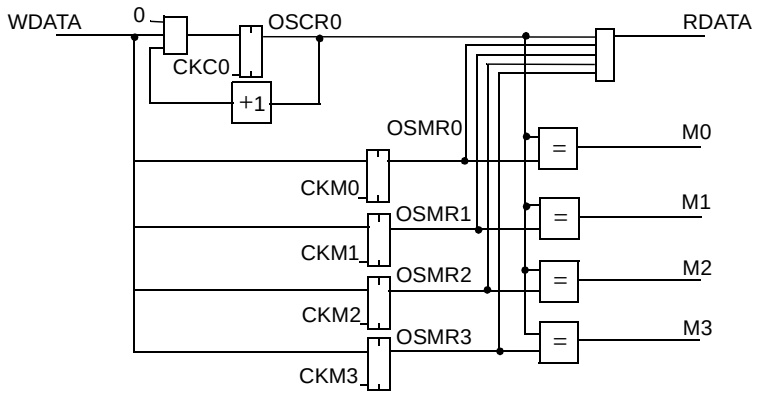
This block is responsible for controlling reads and writes to registers within the operating system timers block. It is also responsible for maintaining the Match Control registers (see Section 22.5.1) and generating the appropriate clocks and control signals for each timer channel.

22.4.1.2 PXA25x Processor-Compatible Timer Channels

This block is responsible for maintaining the four PXA25x processor-compatible timer channels and for generating the appropriate channel-match signals. Figure 22-2 diagrams this timer channel block.

Note: The diagram in Figure 22-2 does not show the clock synchronizers in the register read and write data path.

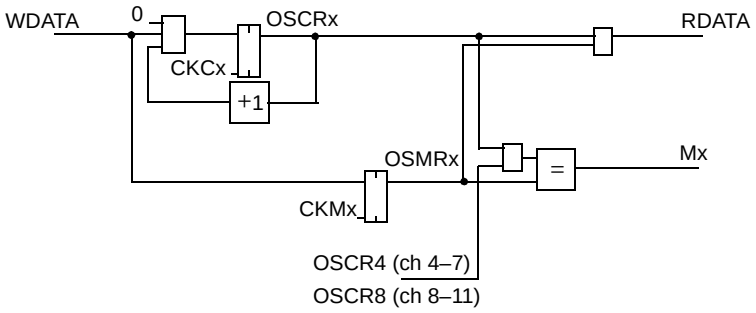
Figure 22-2. PXA25x Processor-Compatible Timer Channel Block (Channels 0 to 3)



22.4.1.3 Channels 4–11

Figure 22-3 is a block diagram of a timer channel.

Figure 22-3. Timer Channel Block Diagram



22.4.1.4 Output Control

This block is responsible for collecting the match signals from each timer channel and generating the match signals to the interrupt controller, the channel-output signals to the GPIO block, and the watchdog-reset signal.

22.4.2 Compares and Matches

Each of the OS Timer Counter registers (OSCRx) is incremented on the rising edge of the corresponding clock. The OS Timer Counter register, OSCRx, is compared with the appropriate OS Timer Match register, OSMRx. When a match occurs, the match bit in the OS Timer Status register (OSSRx) is set only if the corresponding interrupt-enable bit is set in the OS Timer Interrupt Enable register (OIER). There are two exceptions to this rule:

- Watchdog reset—When the OWER[WME] watchdog match-enable bit has been set, the OIER[E3] interrupt-enable bit does not need to be set for a match to generate a watchdog reset. The watchdog reset is sent out on the WDOG_RST pin instead of setting the OSSR[M3] bit.

- If OMCRx[C] bit is not set, another exception occurs (see Table 22-2). In this case, only OSCR4 is compared to OSMR[4:7] to generate OSSR[M4:M7] if the corresponding interrupt-enable bit is set in the OIER register. Similarly, only OSCR8 is compared to OSMR[8:11] to generate OSSR[M8:M11] if the corresponding interrupt-enable bit is set in the OIER register.

For channels 5–7 and channels 9–11, when OMCRx[C] is clear, the corresponding OSCRx register is not incremented.

For channels 5–7 and channels 9–11, when OMCRx[C] is set, a write to the corresponding OSCRx register starts the channel.

For channel 4 and channel 8 only, a write to the corresponding OSCRx register always starts the channel.

22.4.3 PXA25x Processor Compatibility

The one Count register OSCR0 and four Match registers OSMR0–3 are identical to those in the PXA25x processor. The watchdog-reset functionality is also unchanged. However, the input clock that increments OSCR0 has changed. For the PXA25x processor, this clock was 3.6864 MHz. For the PXA27x processor, the clock frequency is 3.25 MHz. Software must recalculate any time periods that must be exact.

22.4.4 Timer Channels

In addition to the OSCR0 and four match registers OSMR0–3, the processor provides eight additional timer channels (OSCR4–11) with a wider range of counter resolutions, interval and periodic timers, synchronization features, output waveform generation, and low-power mode operability. Each of the features is described in this section.

22.4.5 Counter Resolutions

The following sections discuss the possible resolutions for each counter register in the OS timer.

22.4.5.1 Clock Generation for Counter 0 Register

The OSCR0 Counter register always increments on the rising edge of the 3.25-MHz clock. This clock is generated in the Channel Access/Control block by dividing the 13-MHz input clock (CLK_13M) by four.

22.4.5.2 Clock Generation for Channels 4–7

Each of the four counters (OSCR7:4) can be configured to run at the following speeds:

- 32.768 kHz (1/32768th of a second)—This is the 32.768-kHz clock.
- 1 kHz (1 millisecond)—This clock is generated using the 32.768-kHz clock; it approximates 1 ms. The interval between clock increments averages one millisecond, but the time between individual ticks varies because the counter resolution is derived from the 32.768-kHz clock.

Note: This clock may induce a gain of about 1 ms per minute (0.0017%) due to the approximation in deriving this clock from the 32.768 kHz clock when the Timekeeping Oscillator is enabled (OSCC[OON]=1).

- 1 Hz (1 second)—This clock is generated using the 32.768-kHz clock; it approximates 1 second. The interval between ticks averages one second, but the time between individual ticks varies because the counter resolution is derived from the 32.768-kHz clock.
- 1 MHz (1 microsecond)—This clock is derived from the 13-MHz clock.
- The external clock input, CLK_EXT.

22.4.5.3 Clock Generation for Channels 8–11

Each of the four counters (OSCR8:11) can be configured to run at the following speeds:

- 32.768 kHz (1/32768th of a second)—This is the 32.768-kHz clock.
- 1 kHz (1 millisecond)—This clock is generated using the 32.768-kHz clock and approximates 1 ms. The interval between clock increments averages one millisecond, but the time between individual ticks varies because the counter resolution is derived from the 32.768-kHz clock. ***Note:*** This clock may induce a gain of about 1 ms per minute (0.0017%) due to the approximation in deriving this clock from the 32.768 kHz clock when the Timekeeping Oscillator is enabled (OSCC[OON]=1).
- 1 Hz (1 second)—This clock is generated using the 32.768-kHz clock; it approximates 1 second. The interval between ticks averages one second, but the time between individual ticks varies because the counter resolution is derived from the 32.768-kHz clock.
- 1 MHz (1 microsecond)—Derived from the 13-MHz clock.
- Same as the external clock input, CLK_EXT.
- Same as the Frame Detect signal from SSP1.
- Same as the Frame Detect signal from SSP2.
- Same as the Frame Detect signal from SSP3.
- 1 kHz (1 millisecond)—Start-of-Frame signal from UDC.

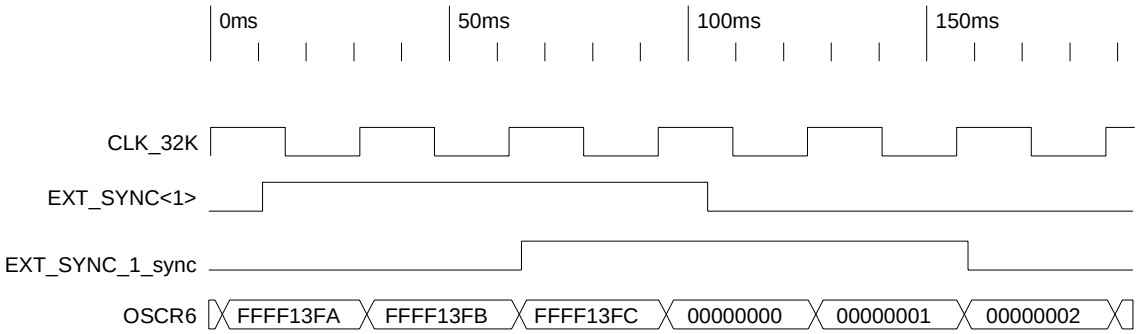
22.4.6 External Synchronization (EXT_SYNC<1:0>)

For channel counters 4–11, when programmed with the OMCRx[S] field, the rising edge of either of the external synchronization input signals EXT_SYNC<1:0> can be used to reset the corresponding Channel Counter register, OSCRx. The timing diagram in Figure 22-4 shows an example where:

- OMCR6[S] field is programmed to 0b10 (reset on rising edge of EXT_SYNC<1>).
- OMCR6[CES] field is programmed to 0b001 (the 32.768-kHz clock).

The EXT_SYNC<1:0> signals are brought into the OS timers block using two-stage synchronizers.

Figure 22-4. Reset of OSCR6 Based on Rising Edge of EXT_SYNC<1> and Counter Configured for 32-kHz Operation



EXT_SYNC_1_sync is an internal signal that is EXT_SYNC<1> after going through a two-stage synchronizer.

CLK_32K is a 32-kHz internal clock signal.

22.4.7 Output Generation

There are two outputs from the OS timers block—CHOUT<1:0>. Channel 11 generates the periodic output CHOUT<1>. Channel 10 generates the periodic output CHOUT<0>.

22.4.7.1 Output Generation for CHOUT<1:0>

For channels 11 and 10, if OMCRx[P] is set, the channel counter OSCRx is configured to be *periodic*. This means that when a match occurs, the channel counter continues counting. If, in addition, OMCRx[R] is set, the counter OSCRx is reset when this match occurs. The effect on the output pin CHOUT<x> follows.

- If OMCRx[R] and OMCRx[P] are cleared, the counter starts at zero and increments at the selected clock rate (see Section 22.4.5.3) until OSCRx matches the value programmed in OSMRx. Once this match occurs, the counter OSCRx stops incrementing and holds its match value. The corresponding CHOUT<x> output pin is held low.
- If OMCRx[R] is set, and OMCRx[P] is cleared, the counter starts at 0x0000_0000 and increments at the selected clock rate (see Section 22.4.5.3) until OSCRx matched the value programmed in OSMRx. Once this match occurs, the counter OSCRx resets to 0x0000_0000 and stops incrementing. The corresponding CHOUT<x> output pin is held low.
- If OMCRx[R] is cleared, and OMCRx[P] is set, the counter starts at 0x0000_0000 and increments at the selected clock rate (see Section 22.4.5.3) until OSCRx matches the value programmed in OSMRx. Once this match occurs, CHOUT<x> is inverted, and the counter OSCRx continues counting. When the counter reaches 0xFFFF_FFFF, it rolls over and continues counting from 0x0000_0000.
- If both OMCRx[R] and OMCRx[P] are set, the counter starts at 0x0000_0000 and increments at the selected clock rate (see Section 22.4.5.3) until OSCRx matches the value programmed in OSMRx. Once this match occurs, CHOUT<x> is inverted, the counter OSCRx is reset to 0x0000_0000, and incrementing of OSCRx resumes.

22.4.8 Snapshot Mode

For channels 11 and 9, enable snapshot mode by setting OMCRx[N] (available only in these two channels). While in snapshot mode, when a read operation of OSCR_11 is performed, a snapshot of the value of OSCR_10 is taken and is transferred to OS Timer Snapshot register OSNR. Similarly, if snapshot mode is enabled for channel 9, a read from OSCR_9 results in copying the current value of OSCR_8 to OSNR. This mode allows software to read the timer values in two channels simultaneously.

22.4.9 Operation in Low-Power Modes

Counter operation for low-power modes (standby and sleep) is summarized in this section. Counters are not stopped but not reset when they are not operating in low-power modes. Counters continue incrementing and checking for matches when operating in low-power modes. The 13-MHz input clock is configurable (either on or off) in standby mode.

Note: No counters operate in deep-sleep mode.

- The OS timers block is in the PI power island.

  To wake up using the OS timers, or if the OS timers block must be functional while in a low-power mode, software must program PSTR[PI] or PSLR[PI] in standby or sleep mode, respectively, to the RUN setting.
- OSCR0 does not operate during any low-power mode.
- The counter register OSCRx (only applies to OSCR4–11) operates in the low-power modes if the clock is generated from the 32.768-kHz input clock, CLK_32K. The counter is configured using OMCRx[CRES].
- The counter register OSCRx (only applies to OSCR4–11) does not operate in the low-power modes if the clock is generated from the external-input clock, CLK_EXT. The counter is configured with OMCRx[CRES].
- The counter register OSCRx (only applies to OSCR4–11) does not operate in the low-power modes if PCFR[OPDE] and OSCC[OOK] are set and the clock is generated from the 13-MHz input clock, CLK_13M. The counter is configured with OMCRx[CRES].

22.5 Register Descriptions

The following sections describe the registers that configure the OS timers.

22.5.1 OS Match Control Registers (OMCRx)

The OS Match Control registers, shown in Table 22-2, Table 22-3, and Table 22-4, control the operation of timer channels 4–11. Periodic control of the timer channels using OMCRx[P] and OMCRx[R] affects timer channels 4–11. The registers have different capabilities and are grouped as follows:

- OMCR[[4–7]
- OMCR[8] and OMCR[10]
- OMCR[9] and OMCR[11]

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 22-2. OMCR4/5/6/7 Bit Definitions (Sheet 1 of 2)

0x40A0_00C0 0x40A0_00C4 0x40A0_00C8 0x40A0_00CC	OMCR4 OMCR5 OMCR6 OMCR7	OS TImer

User Settings																															
Bit	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																								C	P	S		R	CRES		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7	R/W	C	Channel 4–7 Match Against 0 = Channel x is compared to OSCR4 and OSCRx is not incremented. 1 = Channel x is compared to OSCRx and a write to OSCRx starts the channel. NOTE: For channel 4, the counter always operates as if this bit is set.
6	R/W	P	Periodic Timer 0 = The channel stops incrementing after detecting a match. 1 = The channel continues incrementing after detecting a match.
5:4	R/W	S	External Synchronization Control 0b00 = No external synchronization 0b01 = Reset OSCRx on the rising edge of EXT_SYNC<0>. 0b10 = Reset OSCRx on the rising edge of EXT_SYNC<1>. 0b11 = reserved.
3	R/W	R	Reset OSCRx on Match 0 = Do not reset OSCRx on match. 1 = Reset OSCRx on match.

Table 22-2. OMCR4/5/6/7 Bit Definitions (Sheet 2 of 2)

0x40A0_00C0
0x40A0_00C4
0x40A0_00C8
0x40A0_00CC

OMCR4
OMCR5
OMCR6
OMCR7

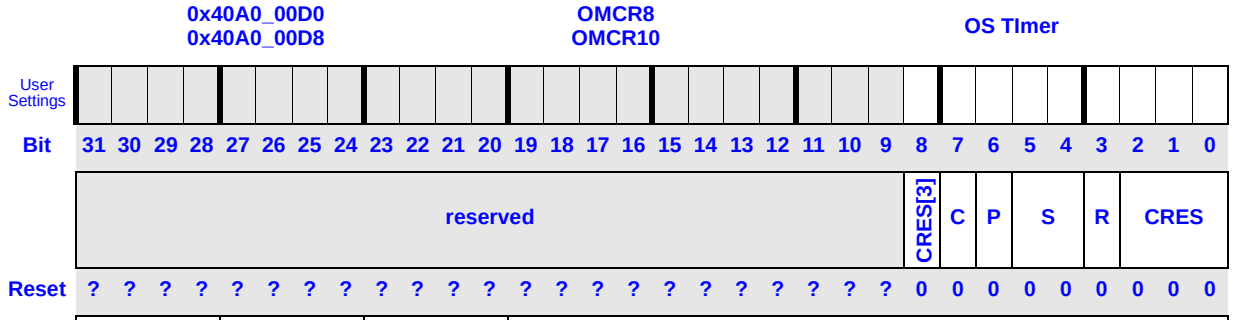
OS TImer

User Settings

Bit	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																								C	P	S		R	CRES		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2:0	R/W	CRES	Counter Resolution 0b000 = The counter is disabled. 0b001 = 1/32768th of a second 0b010 = 1 millisecond. The interval between clock increments *averages* one millisecond, but the time between individual clock increments varies because the counter resolution is derived from the 32.768-kHz clock. 0b011 = 1 second 0b100 = 1 microsecond 0b101 = Externally supplied clock. The counter resolution is the clock period of the externally supplied clock. 0b110–0b111 = reserved Any channel using a counter that is derived from the 32.768-kHz clock continues to operate in standby and sleep mode. This applies to the values of CRES between 0x1 and 0x3. Any channel not using a counter that is derived from the 32.768-kHz clock stops incrementing in standby, sleep, or deep-sleep mode. This applies to the values of CRES of 0b100 and 0b101.

Table 22-3. OMCR8/10 Bit Definitions (Sheet 1 of 2)



Bits	Access	Name	Description
31:9	—	—	reserved
8	R/W	CRES[3]	Most significant bit to be appended to CRES (OMCR8/10[2:0])
7	R/W	C	Channel 8 and 10 Match Against 0 = Channel x is compared to OSCR8 and OSCRx is not incremented. 1 = Channel x is compared to OSCRx and a write to OSCRx starts the channel. NOTE: For channel 8 the counter always operates as if this is set.
6	R/W	P	Periodic Timer 0 = The channel stops incrementing after detecting a match. 1 = The channel continues incrementing after detecting a match.
5:4	R/W	S	External Synchronization Control 0b00 = No external synchronization 0b01 = Reset OSCRx on the rising edge of EXT_SYNC<0>. 0b10 = Reset OSCRx on the rising edge of EXT_SYNC<1>. 0b11 = reserved
3	R/W	R	Reset OSCRx on Match 0 = Do not reset OSCRx on match. 1 = Reset OSCRx on match.

Table 22-3. OMCR8/10 Bit Definitions (Sheet 2 of 2)

	0x40A0_00D0 0x40A0_00D8								OMCR8 OMCR10														OS TImer								
User Settings																														
Bit	**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																							**CRES[3]**	**C**	**P**	**S**		**R**	**CRES**		
Reset	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Bits	Access	Name	Description
2:0	R/W	CRES	Counter Resolution 0b0000 = The counter is disabled. 0b0001 = 1/32768th of a second 0b0010 = 1 millisecond. The interval between clock increments *averages* one millisecond, but the time between individual clock increments varies because the counter resolution is derived from the 32.768-kHz clock. 0b0011 = 1 second 0b0100 = 1 microsecond 0b0101 = Externally supplied clock. The counter resolution is the clock period of the externally supplied clock. 0b0110 = SSP1 Frame Detect. The counter resolution is the SSP1 frame detect rate. 0b0111 = SSP2 Frame Detect. The counter resolution is the SSP2 frame detect rate. 0b1000 = SSP3 Frame Detect. The counter resolution is the SSP3 frame detect rate. 0b1001 = UDC Frame Detect. The counter resolution is the UDC frame detect rate. 0b1010–0b1111 = reserved Any channel using a counter that is derived from the 32.768-kHz clock continues to operate in standby and sleep mode. This applies to the values of CRES between 0x1 and 0x3. Any channel not using a counter that is derived from the 32.768-kHz clock stops incrementing in standby, sleep, or deep-sleep mode. This applies to the values of CRES between 0x4 and 0x9. **NOTE:** The values shown in this field are the result of concatenating CRES[3] with CRES.

Table 22-4. OMCR9/11 Bit Definitions (Sheet 1 of 2)

0x40A0_00D4
0x40A0_00DC

OMCR9
OMCR11

OS TImer

User Settings

Bit	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																						N	CRES[3]	C	P	S		R	CRES		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:10	—	—	reserved
9	R/W	N	Snapshot Mode Channel 9: 0 = Snapshot mode is disabled. 1 = Read from OSCR9 copies contents of OSCR8 to OSNR. Channel 11: 0 = Snapshot mode is disabled. 1 = Read from OSCR11 copies contents of OSCR10 to OSNR.
8	R/W	CRES[3]	Most significant bit to be appended to CRES (OMCR8/10[2:0])
7	R/W	C	Channel 9 and 11 Match Against 0 = Channel x is compared to OSCR8 and OSCRx is not incremented. 1 = Channel x is compared to OSCRx and a write to OSCRx starts the channel.
6	R/W	P	Periodic Timer 0 = The channel stops incrementing after detecting a match. 1 = The channel continues incrementing after detecting a match.
5:4	R/W	S	External Synchronization Control 0b00 = No external synchronization 0b01 = Reset OSCRx on the rising edge of EXT_SYNC<0>. 0b10 = Reset OSCRx on the rising edge of EXT_SYNC<1>. 0b11 = reserved.
3	R/W	R	Reset OSCRx on Match 0 = Do not reset OSCRx on match. 1 = Reset OSCRx on match.

Table 22-4. OMCR9/11 Bit Definitions (Sheet 2 of 2)

	0x40A0_00D4 0x40A0_00DC								OMCR9 OMCR11												OS TImer									
User Settings																														
Bit	**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																						**N**	**CRES[3]**	**C**	**P**	**S**		**R**	**CRES**		
Reset	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Bits	Access	Name	Description
2:0	R/W	CRES	Counter Resolution 0b0000 = The counter is disabled. 0b0001 = 1/32768th of a second 0b0010 = 1 millisecond. The interval between clock increments *averages* one millisecond, but the time between individual clock increments varies because the counter resolution is derived from the 32.768-kHz clock. 0b0011 = 1 second 0b0100 = 1 microsecond 0b0101 = Externally supplied clock. The counter resolution is the clock period of the externally supplied clock. 0b0110 = SSP1 Frame Detect. The counter resolution is the SSP1 frame detect rate. 0b0111 = SSP2 Frame Detect. The counter resolution is the SSP2 frame detect rate. 0b1000 = SSP3 Frame Detect. The counter resolution is the SSP3 frame detect rate. 0b1001 = UDC Frame Detect. The counter resolution is the UDC frame detect rate. 0b1010–0b1111 = reserved Any channel using a counter that is derived from the 32.768-kHz clock continues to operate in standby and sleep mode. This applies to the values of CRES between 0x1 and 0x3. Any channel not using a counter that is derived from the 32.768-kHz clock stops incrementing in standby, sleep, or deep-sleep mode. This applies to the values of CRES between 0x4 and 0x9. **NOTE:** The values shown in this field are the result of concatenating CRES[3] with CRES.

22.5.2 OS Timer Match Registers (OSMRx)

OSMR0–OSMR3 are compared against the OSCR0 register after every rising edge of the 3.25-MHz clock.

The OSMR3 register can also serve as a watchdog timer (see Section 22.5.7).

OSMR4–OSMR11 are compared against an OS Counter register after every rising edge of the clock selected by the CRES field of the appropriate Match Control register (see Section 22.5.1).

If any of these registers matches the configured counter register (OSCRx), the corresponding status bit in OSSR is set. The status bits are routed to the interrupt controller where they generate a CPU interrupt (see Section 22.5.4) it the corresponding interrupt is enabled.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 22-5. OSMR 0–11 Bit Definitions

0x40A0_0000–0x40A0_000C 0x40A0_0080–0x40A0_009C																OSMR0–OSMR3 OSMR4–OSMR11																OS Timer															
User Settings																																												
Bit	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												
	Match_Value																																											
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												

Bits	Access	Name	Description
31:0	R/W	Match_Value	Match Value Value that is compared with OSCRx.

22.5.3 OS Timer Watchdog Match Enable Register (OWER)

OSCR0 can be programmed to generate a watchdog-reset signal. When OWER[WME] is set, OSCR0 is compared to OSMR3 every rising edge of the 3.25-MHz clock. If a match is detected, the output pin nRESET_OUT is asserted; a reset is applied to the PXA27x processor and most internal states are cleared. The only way to clear this pin is with a reset function (hardware reset, sleep-exit reset, watchdog reset, or GPIO reset).

The following procedure is suggested when using OSMR3 as a watchdog:

- Each time the operating system services the register, the current value of the counter is read, and a number is then added to the value read, corresponding to the amount of time before the next time-out (be sure to account for counter wraparound)
- This number is then written back to OSMR3.
- The OS code must repeat this procedure periodically before each match occurs. If the match occurs, the OS timer asserts the WDOG_RST pin. This pin can be reset only by one of the reset functions (hardware reset, sleep-exit reset, watchdog reset, or GPIO reset)

The Watchdog Match Enable register contains a single control bit (bit 0) that enables the watchdog function by setting this bit. Once enabled, the watchdog function can be disabled only by one of the reset functions (hardware reset, sleep-exit reset, watchdog reset, or GPIO reset). Writing a zero to the Watchdog Match Enable bit after it has been set has no effect. Table 22-6 shows the bit locations for the OWER register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 22-6. OWER Bit Definitions

0x40A0_0018	OWER	OS Timer

User Settings																															
Bit	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																															WME
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0

Bits	Access	Name	Description
31:1	reserved	—	reserved Read as unknown and must be written as zero.
0	R/W	WME	Watchdog Match Enable 0 = OSMR3 match with OSCR0 does not cause a reset of the processor. 1 = OSMR3 match with OSCR0 causes a reset of the processor.

22.5.4 OS Timer Interrupt Enable Register (OIER)

This register contains 12 enable bits that configure whether a match between one of the OS Match registers and an OS Counter register sets a status bit in the OSSR. Each match register has a corresponding enable bit. Clearing an enable bit does not clear the corresponding interrupt status bit if that bit is already set. See Table 22-7.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 22-7. OIER Bit Definitions

0x40A0_001C	OIER	OS Timer

User Settings																															
Bit	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																				E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:12	—	—	reserved
11:0	R/W	E{*n*}	Interrupt Enable Channel *n* 0 = No interrupt bit is asserted on a match. 1 = Assert the interrupt bit M(*n*) in the OSSR if a match occurs between OSMR*n* and the OS timer.

22.5.5 OS Timer Count Register 0 (OSCR0)

This register is incremented on rising edges of the 3.25-MHz clock. The counter can be read or written at any time. Table 22-8 shows the bit definitions.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 22-8. OSCR0 Bit Definitions

0x40A0_0010	OSCR0	OS Timer

User Settings

Bit	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
	Count																															
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

Bits	Access	Name	Description
31:0	R/W	Count	Count This field increments on every rising edge of the 3.25-MHz clock.

22.5.6 OS Timer Count Registers (OSCR4–11)

These OS Timer Count registers increment on rising edges of the clock selected by the CRES field of the appropriate Match Control register (see Section 22.5.1). A counter can be read or written at any time. Table 22-9 shows the bit definitions.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 22-9. OSCR4–11 Bit Definitions

0x40A0_0040 0x40A0_0044 0x40A0_0048 0x40A0_004C 0x40A0_0050 0x40A0_0054 0x40A0_0058 0x40A0_005C	OSCR4 OSCR5 OSCR6 OSCR7 OSCR8 OSCR9 OSCR10 OSCR11	OS Timer

User Settings

Bit	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
	Count																															
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

Bits	Access	Name	Description
31:0	R/W	Count	Count This field increments on every rising edge of the clock specified by the CRES field of the appropriate OMCR register.

22.5.7 OS Timer Status Register (OSSR)

This status register contains status bits to indicate if a match has occurred between any of the 12 OS Match registers and the OS Counter registers. These bits are set when a match event occurs and the corresponding interrupt-enable bit is set in the OIER register. If a match value is loaded that equals any of the current OS Counter registers, or a Counter register is loaded with a value that

equals any of the match registers, the status bit is set immediately, before the next rising clock edge for the counters. The OSSR bits are cleared by setting the proper bit position. Clearing any bit locations in this register has no effect. Table 22-11 shows the bit locations for OSSR.

Match registers 0–3 generate separate interrupts to the interrupt controller but match registers 4–11 generate only a single interrupt, and software must determine the source from the value in the OSSR register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 22-10. OSSR Bit Definitions

0x40A0_0014	OSSR	OS TImer																													
User Settings																															
Bit	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																				M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:12	reserved	—	reserved
11:0	R/W	M{*n*}	Match Status Channel *n* If OIER(*n*) is set then: 0 = No OSMR*n* match has occurred since the last clear. 1 = An OSMR*n* match has occurred since the last clear.

22.5.8 OS Timer Snapshot Register (OSNR)

This status register contains snapshot value of either OSCR8 or OSCR10. When snapshot mode is enabled in OMCR9 and OSCR9 is read, the contents of OSCR8 are copied to OSNR. Similarly, when snapshot mode is enabled in OMCR11 and OSCR11 is read, the contents of OSCR10 are copied to OSNR.

This is a read-only register. Ignore reads from reserved bits.

Table 22-11. OSNR Bit Definitions

0x40A0_0020	OSNR	OS TImer																													
User Settings																															
Bit	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
	Snapshot_Value																															
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

Bits	Access	Name	Description
31:0	R	Snapshot_ Value	Snapshot Value The snapshot value of channel 8 or channel 10

22.6 Register Summary

Table 22-12 shows the registers associated with the OS timers and the physical addresses to access them.

Table 22-12. OS Timers Register Summary (Sheet 1 of 2)

Address	Name	Description	Page
0x40A0_0000	OSMR0	OS Timer Match 0 register	22-15
0x40A0_0004	OSMR1	OS Timer Match 1 register	22-15
0x40A0_0008	OSMR2	OS Timer Match 2 register	22-15
0x40A0_000C	OSMR3	OS Timer Match 3 register	22-15
0x40A0_0010	OSCR0	OS Timer Counter 0 register	22-17
0x40A0_0014	OSSR	OS Timer Status register (used for all counters)	22-18
0x40A0_0018	OWER	OS Timer Watchdog Enable register	22-16
0x40A0_001C	OIER	OS Timer Interrupt Enable register (used for all counters)	22-16
0x40A0_0020	OSNR	OS Timer Snapshot register	22-19
0x40A0_0024–0x40A0_003C	—	reserved	—
0x40A0_0040	OSCR4	OS Timer Counter 4–11 registers	22-17
0x40A0_0044	OSCR5		
0x40A0_0048	OSCR6		
0x40A0_004C	OSCR7		
0x40A0_0050	OSCR8		
0x40A0_0054	OSCR9		
0x40A0_0058	OSCR10		
0x40A0_005C	OSCR11		
0x40A0_0060–0x40A0_007C	—	reserved	—
0x40A0_0080	OSMR4	OS Timer Match 4–11 registers	22-15
0x40A0_0084	OSMR5		
0x40A0_0088	OSMR6		
0x40A0_008C	OSMR7		
0x40A0_0090	OSMR8		
0x40A0_0094	OSMR9		
0x40A0_0098	OSMR10		
0x40A0_009C	OSMR11		
0x40A0_00A0–0x40A0_00BC	—	reserved	—

Table 22-12. OS Timers Register Summary (Sheet 2 of 2)

Address	Name	Description	Page
0x40A0_00C0	OMCR4	OS Match Control 4–7 registers	22-9
0x40A0_00C4	OMCR5		
0x40A0_00C8	OMCR6		
0x40A0_00CC	OMCR7		
0x40A0_00D0	OMCR8	OS Match Control 8 register	22-11
0x40A0_00D4	OMCR9	OS Match Control 9 register	22-13
0x40A0_00D8	OMCR10	OS Match Control 10 register	22-11
0x40A0_00DC	OMCR11	OS Match Control 11 register	22-13
0x40A0_00E0–0x40AF_FFFC	—	reserved	—

Pulse Width Modulator Controller 23

The pulse width modulator (PWM) controller generates four independent PWM outputs. Specific applications of the PWM controller vary. In general, the PWM controller provides a basic digital-to-analog converter with an appropriate analog filter. Examples include:

- Controlling the brightness of an LED output by modulating the “on” time
- LCD contrast control

23.1 Overview

The PXA27x processor contains four PWMs, PWM0–PWM3. Each PWM operates independently of the others, is configured by its own set of registers, and provides a pulse-width modulated signal on an external pin. Because each PWM contains identical circuitry, a generic PWM<x>, where x is 0, 1, 2, or 3, is described.

Each PWM function enables the control of leading- and falling-edge timing of a single output channel. The edge timing can be set up to run indefinitely or adjusted on the fly to adapt to variable requirements. Power-saving modes include the ability to stop the PSCLK_PWM used to source the PWM<x> and drive the PWM_OUT<x> signals to a steady high or low state.

The frequency range supporting a 50% duty cycle varies from 49.6 kHz to 1.625 MHz. Other duty-cycle options depend on the choice of desired frequency.

23.2 Features

- Four pulse-width modulated output channels
- Enhanced period control through 6-bit clock divider and 10-bit period counter
- 10-bit pulse control

23.3 Signal Descriptions

Output signals are the four single-bit output channels defined as PWM_OUT<0>, PWM_OUT<1>, PWM_OUT<2>, and PWM_OUT<3> (see Table 23-1). These signals are sent to GPIO multiplexers.

Table 23-1. Pulse Width Modulator I/O Signal Descriptions

Signal Name	Direction	Description
PWM_OUT<0>	Output	Pulse-width modulated output signal for PWM 0
PWM_OUT<1>	Output	Pulse-width modulated output signal for PWM 1
PWM_OUT<2>	Output	Pulse-width modulated output signal for PWM 2
PWM_OUT<3>	Output	Pulse-width modulated output signal for PWM 3

23.4 Operation

Pulse-width modulated outputs, once programmed, output a specified waveform until the value in any associated register is altered.

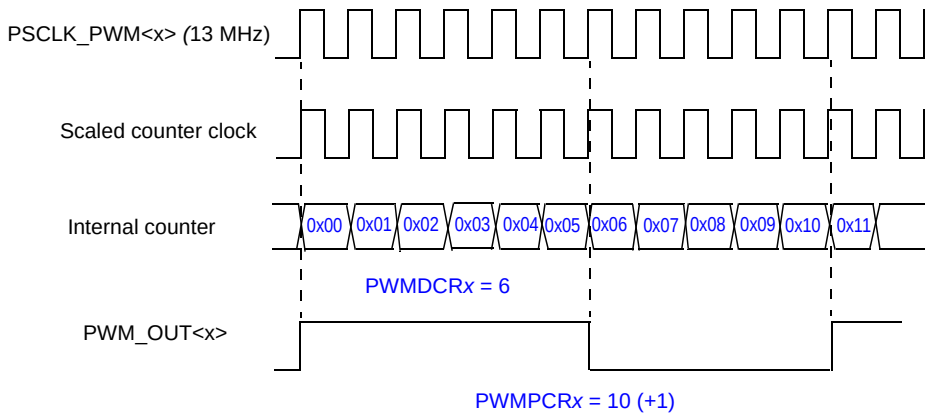
- For PWMPCRx[PV] values 0x005 and larger—After a register value is altered, the PWM_OUT<x> output changes when the previously programmed waveform cycle is complete.
- For PWMPCRx[PV] values less than 0x005—After a register value is altered, the PWM_OUT<x> output changes after two waveform cycles.

To program the PWM controller, determine the period and pulse-width values. The period value is based on two registers, PWMPCR*x* and PWMCR*x*.

Each of the four PWM channels can be independently shut down to save power.

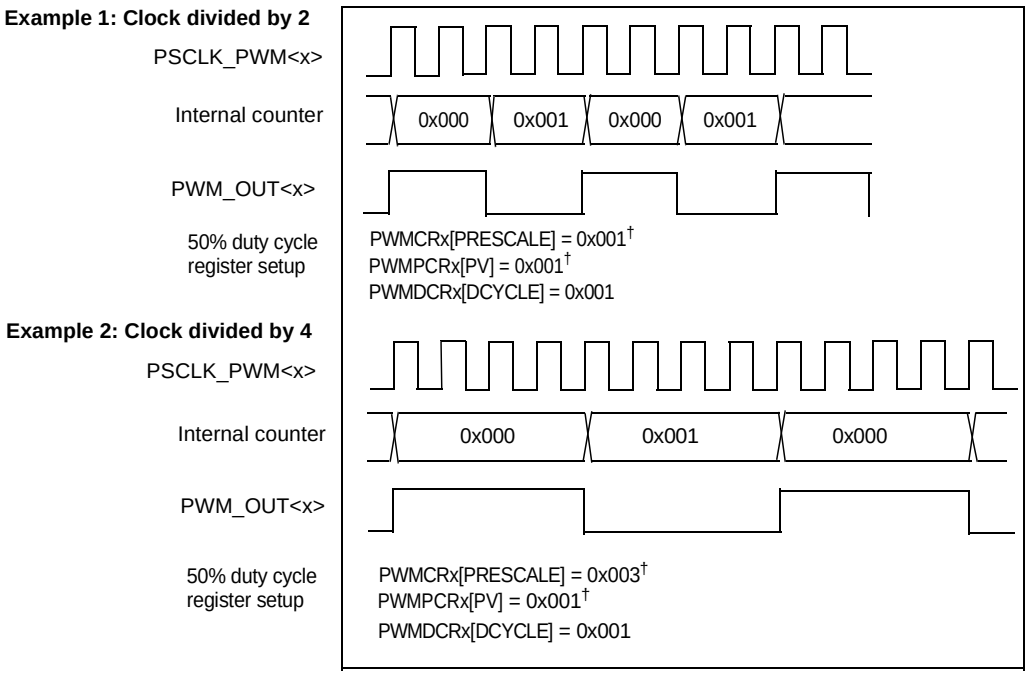
The output waveform in Figure 23-1 is derived by writing the PWMPCR*x* register with a value of 10(0x00A) and writing the PWMDCR*x* register with 0x06.

Figure 23-1. Basic Pulse Width Modulated Waveform



Programming PWMCRx[PRESCALE] configures the Scaled Counter Clock (refer to Figure 23-3). Two timing examples are provided in Figure 23-2. Both examples have the PWMDCR*x* and PWMPCR*x* registers set with the same 50% duty cycle setting. The first example shows the effect on the Scaled Counter Clock effectively being divided by two with a setting of 0x01, while the second example shows the Scaled Counter Clock being divided by four with a setting of 0x03. For more information regarding the calculation of waveform values, see Section 23.4.4 and Figure 23-3.

Figure 23-2. Effects of PWMCRx Settings

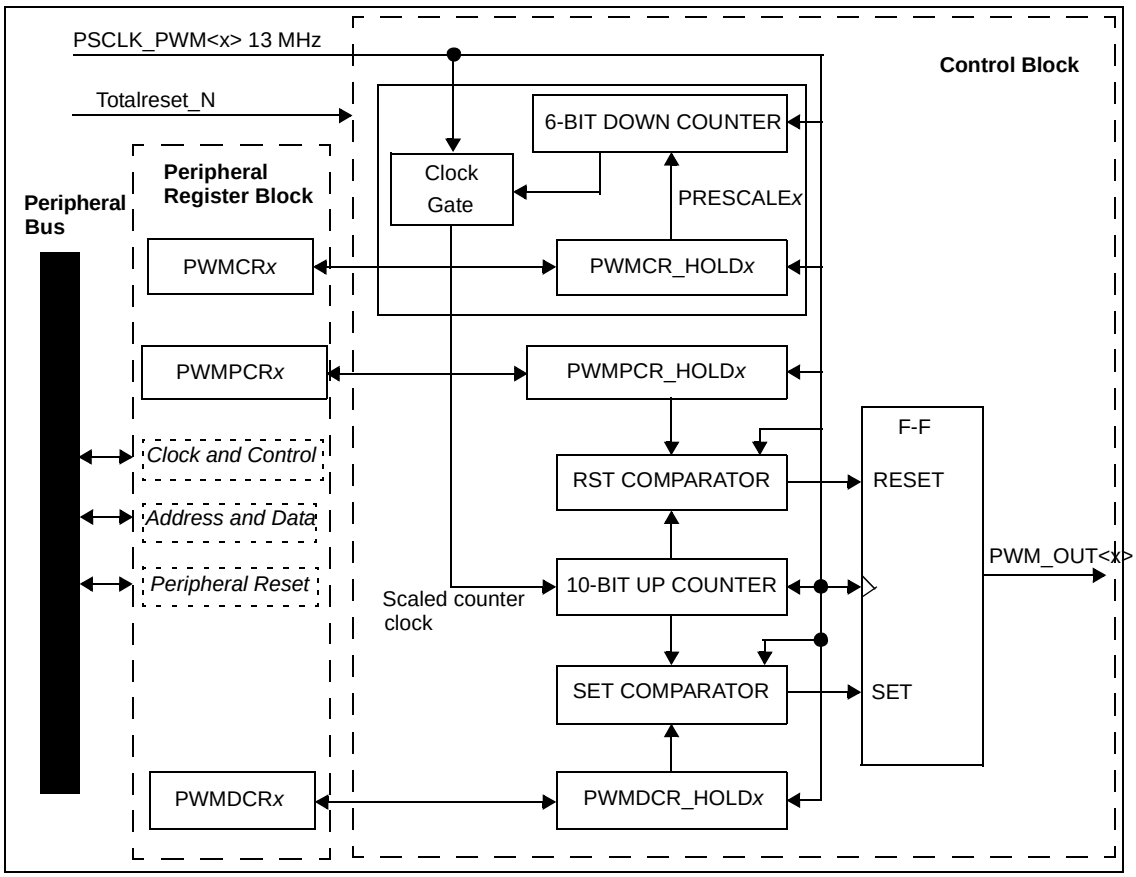


† The effective value of this field is one greater than the programmed value. Refer to register descriptions.

23.4.1 Block Diagram

Figure 23-3 shows the block diagram for the PWM control logic.

Figure 23-3. PWM<x> Block Diagram



23.4.2 Interdependencies

PWM<x> module clocks are supplied by the clock unit. The signals are based on the 13-MHz clock and must be a minimum of two clock cycles wide. These signals are sent from the PXA27x processor by configuring the GPIOs.

Each PWM<x> is controlled by three registers: the Pulse Width Control register (PWMCR*x*), the Duty-Cycle Control register (PWMDCR*x*), and the Period Control register (PWMPCR*x*). Using the values in these registers, the PWM<x> unit produces a pulse-width modulated output signal. The registers contain the values for PWM<x> counters and PWM<x> power-management mode.

23.4.3 Reset Sequence

During system reset, the PWMCR*x* and PWMDCR*x* registers are reset to 0 and the PWMPCR*x* register is set to 0x004. Externally, reset places the PWM_OUT<x> channels in a steady low state. Internally, the 6-bit down counter is reset to 0x0 and passes PSCLK_PWM<x> on to the Scaled

Count Clock and to the 10-bit up counter, which causes the 10-bit up counter to count continuously from 0x0 to 0x4. The PWM_OUT<x> channel remains reset to 0 until the PWMDCR*x* register is programed with a non-zero value. Therefore, system reset results in no pulse-width modulated signal.

The register interface can be reset independently during operation using the peripheral bus reset. Each PWM<x> contains three registers that control the clock, the period, and the duty cycle timing of the PWM_OUT<x>. The following sections provide a summary (Table 23-5) and detailed register descriptions (Table 23-2, Table 23-3, and Table 23-4).

23.4.4 Programming Considerations

The PWMs use three registers to configure the output of each PWM<x> signal: PWMCRx, PWMDCRx, and PWMPCRx.

PWM<x> timing is based on the input clock to the PWM<x> controller, PSCLK_PWM<x>, which is fixed at 13 MHz. This signal is divided by (PWMCRx[PRESCALE] + 1) to generate the Scaled Counter Clock. The 6-bit PRESCALE field allows the input clock to be divided by values between 1 (PRESCALE = 0) and 64 (PRESCALE = 63). The Scaled Counter Clock is further divided by contents of the PWMDCRx and PWMPCRx registers to generate the duty cycle (time asserted) and period of the PWM<x> signal.

To calculate the frequency of the Scaled Counter Clock, use the following equation:

Scaled Counter Clock frequency = 13 MHz / (PWMCRx[PRESCALE] + 1)

To calculate the cycle time of the Scaled Counter Clock, use the following equation:

Scaled Counter Clock cycle time = 76.9 ns x (PWMCRx[PRESCALE] + 1)

Both the period and the duty cycle of the PWM are based on the Scaled Counter Clock cycle time. The PWM_OUT<x> signal is asserted for the number of Scaled Counter Clock cycles equal to PWMDCRx[DCYCLE].

To calculate the duty cycle time of the PWM, use the following equation:

Duty cycle time = Scaled Counter Clock cycle time x PWMDCRx[DCYCLE]

which also equals:

Duty cycle time = 76.9nS x (PWMCRx[PRESCALE] + 1) x PWMDCRx[DCYCLE]

The PWM Period Control register (PWMPCRx) determines the number of Scaled Counter Clock cycles each PWM period contains. The actual number of clocks is the value of PWMPCRx[PV] plus one. When the RST comparator in Figure 23-3 equals (PWMPCRx[PV] + 1), the comparators and the flip-flop are reset, and the values of the PWMDCR_HOLD*x*, PWMCR_HOLD*x*, and PWMPCR_HOLD*x* registers are loaded from the control block.

To calculate the period of the PWM, use the following equation:

PWM period = Scaled Counter Clock period x (PWMPCRx[PV] + 1)

which also equals:

PWM cycle time = 76.9nS x (PWMCRx[PRESCALE] + 1) x (PWMPCRx[PV] + 1)

Note: The PWMDCRx[FD] bit determines if PWM_OUT<x> is always asserted. When this bit is set, PWM_OUT<x> remains high until PWMDCRx[FD] is cleared.

To produce a toggle of the output, ensure that the value of the PWMPCR*x*[PV] remains equal to or greater than PWMDCR*x*[DCYCLE]. If PWMPCR*x*[PV] is less than PWMDCR*x*[DCYCLE], the PWM_OUT<x> output remains high. Ensure that the PWMDCR*x*[DCYCLE] is greater than one. If it equals zero, the output remains low.

23.4.5 Power Management

Each PWM<x> can be disabled through a pair of clock-enable bits (see Section 3.8.2.2, “Clock Enable Register (CKEN)” on page 3-98). If the clock is disabled, the unit shuts down abruptly or gracefully (as selected by the PWMPCRx[SD] described in Section 23.5.1).

- For a graceful shutdown, PWM_OUT<x> completes the current cycle before it stops.
- For an abrupt shutdown, PWM_OUT<x> stops immediately.

Shutdown for the PWM modules is defined as the clock stopping. If power is removed after the clock is stopped, the registers are placed in an unknown state and must be rewritten.

23.5 Register Descriptions

23.5.1 PWM Control Registers (PWMCRx)

The PWM<x> Control registers (PWMCR*x*), defined in Table 23-2, configure the behavioral characteristics of the PWM<x> shut-down response and the divisor for the input clocks to the PWM<x> control unit that configures the frequency of the Scaled Counter Clock.

The 13-MHz clock (PSCLK_PWM<x>) is divided by (PWMCRx[PRESCALE] + 1) to generate the Scaled Counter Clock (see Figure 23-3). The Scaled Counter Clock clocks the counters and the flip-flops that control the level of PWM<x>.

Each PWM<x> can shut down gracefully or abruptly, depending on the PWMCR*x*[SD] setting. If PWMCRx[SD] is set, a graceful shutdown is configured and the duty-cycle counter completes its count before PWM<x> enters the power-management mode. If PWMCRx[SD] is cleared, an abrupt shutdown is configured, the prescale and duty-cycle counters are immediately reset to the reload values in their associated registers, and PWM<x> immediately enters the shutdown mode. The PWM_OUT<x> signal may be delayed by at most one PSCLK_PWM<x> clock period. See Section 3.6, “Power Manager Operation” on page 3-34 for power-management information.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 23-2. PWMCR0/1/2/3 Bit Definitions

Physical Address		PWM Controller
0x40B0_0000	PWMCR0	
0x40C0_0000	PWMCR1	
0x40B0_0010	PWMCR2	
0x40C0_0010	PWMCR3	

User Settings																															
Bit	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																									SD	PRESCALE					
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:7	—	—	reserved
6	R/W	SD	Pulse Width Modulator Shutdown Mode 0 = Graceful shutdown of PWM<x> when the processor transitions to power management mode. 1 = Abrupt shutdown of PWM<x> when the processor transitions to power management mode.
5:0	R/W	PRESCALE	The scaled counter clock frequency is: PSCLK_PWM<x> / (PRESCALE*x +1)*

23.5.2 PWM Duty Cycle Registers (PWMDCRx)

The PWM<x> Duty Cycle registers (PWMDCR*x*), defined in Table 23-3, configure the duty cycle of the corresponding PWM_OUT<x> signals.

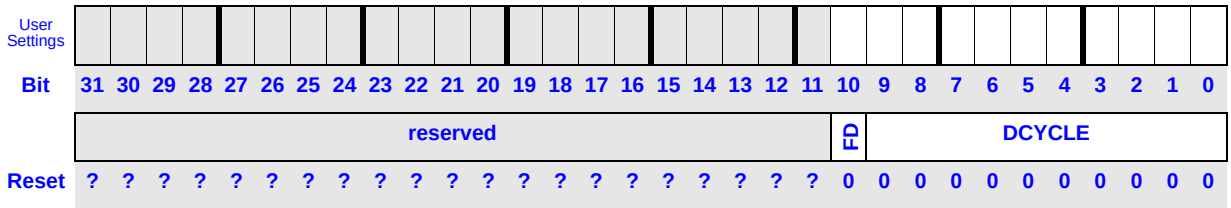
PWMDCRx[DCYCLE] specifies the number of Scaled Counter Clocks that PWM_OUT<x> is asserted during each cycle of the PWM_OUT<x>. Refer to Section 23.4.4 and Figure 23-3 for details on calculating the value of PWMDCRx[DCYCLE].

If PWMDCRx[FD] is set, PWM_OUT<x> remains high until PWMDCRx[FD] is cleared. This results in a duty cycle of 100%. Typically, PWMDCRx[FD] is cleared and the duty cycle of PWM_OUT<x> is a function of PWMDCRx[DCYCLE].

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 23-3. PWMDCR0/1/2/3 Bit Definitions

Physical Address		
0x40B0_0004	PWMDCR0	PWM Controller
0x40C0_0004	PWMDCR1	
0x40B0_0014	PWMDCR2	
0x40C0_0014	PWMDCR3	



Bits	Access	Name	Description
31:11	—	—	reserved
10	R/W	FD	Full Duty Cycle 0 = PWM_OUT<x> is determined by DCYCLE value. 1 = PWM_OUT<x> is continuously asserted.
9:0	R/W	DCYCLE	Duty Cycle of PWM_OUT<x> 0 = PWM_OUT<x> is continuously deasserted. 1 = PWM_OUT<x> is high for the number of 13-MHz clock periods equal to PWMDCRx[DCYCLE] x (PWMCRx[PRESCALE]+ 1), which is a time equal to PWMDCRx[DCYCLE] x (PWMCRx[PRESCALE] + 1) x 76.9 ns If FD is set, DCYCLE has no effect on the output of PWM<x>.

23.5.3 PWM Period Control Registers (PWMPCRx)

The Period Control registers (PWMPCR*x*), defined in Table 23-4, configure the cycle time of the corresponding PWM_OUT<x> signals.

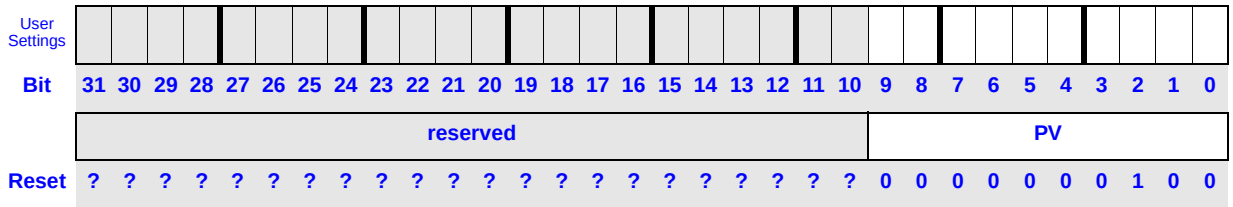
PWMPCRx[PV] specifies the number of Scaled Counter Clocks (plus one) in each cycle of the PWM_OUT<x>. Refer to Section 23.4.4 and Figure 23-3 for details on calculating the value of PWMPCRx[PV].

If these registers are cleared, the period is two scaled clock cycles, and the PWM_OUT<x> output maintains a high state.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 23-4. PWMPCR0/1/2/3 Bit Definitions

Physical Address		PWM Controller
0x40B0_0008	PWMPCR0	
0x40C0_0008	PWMPCR1	
0x40B0_0018	PWMPCR2	
0x40C0_0018	PWMPCR3	



Bits	Access	Name	Description
31:10	—	—	reserved
9:0	R/W	PV	Period Value The value of scaled clock cycles per cycle of PWM_OUT<x> plus one. If all zeros are written to this register, a single clock cycle count is produced, and the output remains high.

23.6 Register Summary

Table 23-5 shows the registers associated with the PWM controller and the physical addresses to access them.

Table 23-5. PWM Control Registers

Address	Name	Description	Page
0x40B0_0000	PWMCR0	PWM 0 Control register	23-7
0x40B0_0004	PWMDCR0	PWM 0 Duty Cycle register	23-8
0x40B0_0008	PWMPCR0	PWM 0 Period register	23-9
0x40B0_000C	—	reserved	—
0x40B0_0010	PWMCR2	PWM 2 Control register	23-7
0x40B0_0014	PWMDCR2	PWM 2 Duty Cycle register	23-8
0x40B0_0018	PWMPCR2	PWM 2 Period register	23-9
0x40B0_001C– 0x40BF_FFFC	—	reserved	—
0x40C0_0000	PWMCR1	PWM 1 Control register	23-7
0x40C0_0004	PWMDCR1	PWM 1 Duty Cycle register	23-8
0x40C0_0008	PWMPCR1	PWM 1 Period register	23-9
0x40C0_000C	—	reserved	—
0x40C0_0010	PWMCR3	PWM 3 Control register	23-7
0x40C0_0014	PWMDCR3	PWM 3 Duty Cycle register	23-8
0x40C0_0018	PWMPCR3	PWM 3 Period register	23-9
0x40C0_001C–0x40CF_FFFC	—	reserved	—

General-Purpose I/O Controller 24

This chapter describes the operation of the general-purpose I/O (GPIO) controller.

24.1 Overview

The PXA27x processor provides 121[1] highly-multiplexed general-purpose I/O (GPIO) pins for use in generating and capturing application-specific input and output signals. Each pin can be programmed as an output, an input, or as bidirectional for certain alternate functions (that override the value programmed in the GPIO direction registers). When programmed as an input, a GPIO can also serve as an interrupt source. All GPIO pins are configured as inputs during the assertion of all resets, and they remain inputs until configured otherwise. In addition, select special-function GPIO pins serve as bidirectional pins where the I/O direction is driven from the respective unit (overriding the GPIO direction register).

To minimize power consumption, configure all unused GPIOs as outputs.

A subset of the GPIO pins can generate wake-up events to bring the processor out of sleep and deep-sleep modes, as described in Section 24.4.1. When the PXA27x processor comes out of sleep or reset (hardware reset, power-on reset, GPIO reset, watch dog reset, sleep, or deep-sleep), the GPIO input path is disabled until the read disable hold (PSSR[RDH]) bit is cleared. PSSR[RDH] must not be cleared until the GPIO registers have been configured for their respective functions.

GPIO pins may have alternate input and output functions. A pin may serve either as GPIO or as an alternate function, but not as both at the same time, as described in Section 24.4.2.

24.2 Features

Most of the peripheral pins double as GPIO pins. This section lists the general features of the GPIO.

- As inputs, the GPIOs can be sampled or programmed to generate interrupts from either rising or falling edges.
- As outputs, the GPIOs can be individually cleared or set. They can be preprogrammed to either state when entering sleep or deep-sleep mode.
- Each GPIO can be programmed to alternate functions, providing system flexibility.

24.3 Signal Descriptions

Table 24-1 describes the signals associated with the GPIO controller. The GPIO signals are multiplexed with other signal pins that are described in the corresponding chapters.

1. Although the PXA27x processor has 121 GPIO pins, only 119 are bonded out in the PXA270 processor. All 121 GPIOs are available in the PXA271, PXA272, and PXA273 processors. Additionally, five of these GPIOs have dedicated functions and are not available as GPIOs.

Table 24-1. GPIO Controller I/O Signal Descriptions

Signal Name	Type	Description
GPIO<120:0>	Input or Output	Programmable for: • Inputs or outputs • Interrupts or wake-up sources • Rising or falling edge • GPIO or one of several alternate functions
NOTE: GPIO<120:119> are supported in the PXA271, PXA272, and PXA273 processors only.		

24.4 Operation

The GPIO signals operate as either general-purpose I/O or as one of their alternate functions. This section describes operation in both modes.

24.4.1 GPIO Operation as Application-Specific GPIO

Use the GPIO Pin Direction registers GPDR0/1/2/3 (see Section 24.5.1) to program the GPIO pins as inputs or outputs. For a pin configured as an output, write to the GPIO Pin Output Set register (GPSR0/1/2/3) to set the pin high; write to the GPIO Pin Output Clear register (GPCR0/1/2/3) to clear the pin to a low level (see Section 24.5.2). Writes to GPDRx and GPSRx can take place whether the pin is configured as an input or an output. If a pin is configured as an input, the programmed output state occurs when the pin is reconfigured as an output.

To validate the state of a GPIO pin, read the GPIO Pin Level register GPLR0/1/2/3 (see Section 24.5.5). Software can read this register at any time to confirm the state of a pin, even if the pin is configured as an output.

To detect either a rising or a falling edge on each GPIO pin, use the GPIO Rising-Edge Detect Enable registers (GRER0/1/2/3) and GPIO Falling Edge Detect Enable registers (GFER0/1/2/3). For register details, see Section 24.5.3. Use the GPIO Edge Detect Status register GEDR0/1/2/3 (see Section 24.5.6) to read edge-detect state. To program these edge-detects to generate interrupts, see Chapter 25, “Interrupt Controller”.

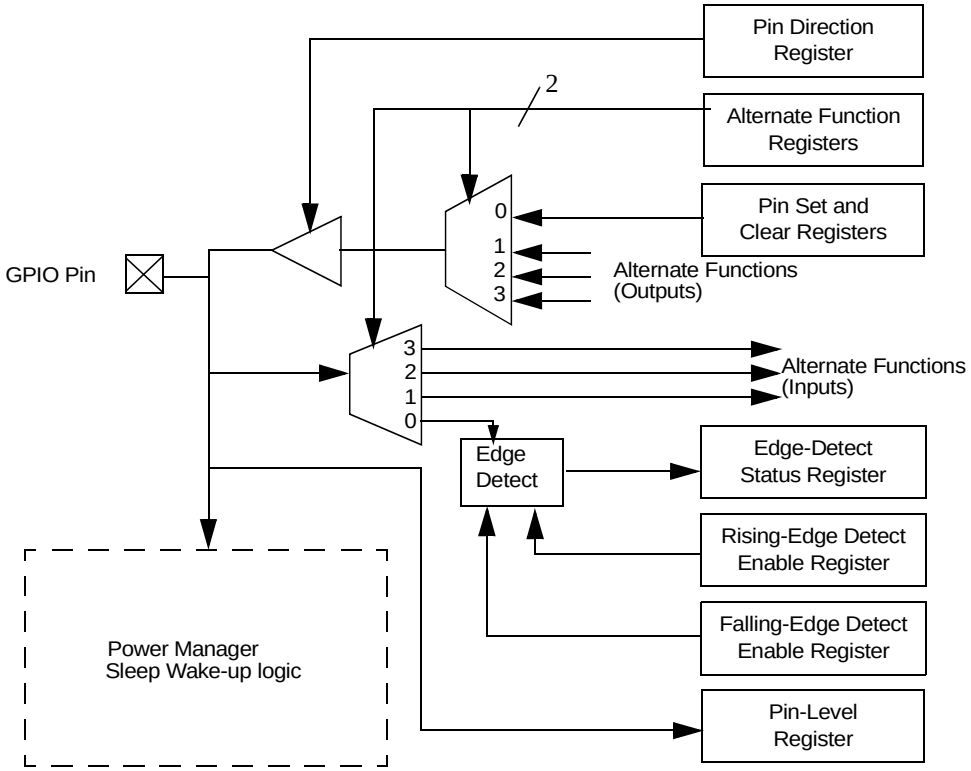
GPIO<116>, GPIO<113>, GPIO<102:93>, GPIO<91:90>, GPIO<83>, GPIO<53>, GPIO<40:34>, GPIO<31>, GPIO<17:9>, GPIO<4:3>, and GPIO<1:0> can generate wake-up events to bring the processor out of sleep and standby modes (see Section 3.6.9, “Sleep Mode” on page 3-45 and Section 3.6.10, “Deep-Sleep Mode” on page 3-48), in addition to the keypad, USB, and MSL wake-up events. Additionally, GPIO<3> and GPIO<1:0> can bring the processor out of deep-sleep mode.

When the processor enters sleep mode, the contents of the Power Manager Sleep State registers (PGSR0/1/2/3) are loaded into the output data registers. If the particular pin is programmed as an output, then the value in the PGSR is driven onto the pin before entering sleep mode. When the PXA27x processor exits sleep mode, these values remain driven until the GPIO pins are reprogrammed by writing to the GPDR, GPSR or GPCR, and setting the GPIO bit in the Power Manager Sleep Status register (PSSR) to indicate that the GPIO registers have been re-initialized after sleep mode. This is necessary since the GPIO logic loses power during sleep mode.

Most GPIO pins are multiplexed with alternate functions of the PXA27x processor. Certain modes within the serial controllers and LCD controller require extra pins. These functions are externally available through specific GPIO pins, and their use is described in the following paragraphs. Even though a GPIO pin is used for an alternate function, the proper direction of that pin must still be programmed through the GPDR. Details on alternate functions are provided in Section 24.4.2.

Figure 24-1 shows a block diagram of a single GPIO pin.

Figure 24-1. General-Purpose I/O Block Diagram



24.4.2 GPIO Operation as Alternate Function

GPIO pins can have as many as three alternate input and three alternate output functions. If a GPIO is used for an alternate function, then it cannot be used as a GPIO at the same time. When using an alternate function of a GPIO signal, first configure the alternate function and then enable the corresponding unit. Also, disable the unit prior to changing the alternate function signals in the GPIO control registers.

- GPIO<0> is reserved because of its special use during sleep mode and is not available for alternate functions.
- GPIO<116>, GPIO<113>, GPIO<102:93>, GPIO<91:90>, GPIO<83>, GPIO<53>, GPIO<40:34>, GPIO<31>, GPIO<17:9>, GPIO<4:3>, and GPIO<1:0>, in addition to the keypad, USB, and MSL wake-up events, are used for wake-up from sleep or standby mode.
- GPIO<3> and GPIO<1:0> can be used exclusively for wake-up from deep-sleep mode; if BATT_FAULT or nVDD_FAULT is asserted, GPIO<1:0> are used for wake-up.

The wake-up functionality is described in Section 3.6.9.4, “Sleep Exit” on page 3-47. Table 24-2 shows each GPIO pin and its corresponding alternate functions.

Note: Configuring a GPIO for an alternate function that is not defined for it causes unpredictable results.

The Power Manager is capable of overriding the alternate function of GPIO<10:9> and GPIO<4:3> directly. See Section 3.4.6.1, “Enabling GPIO Reset” on page 3-10 for more details.

Note: PSSR[RDH] must be cleared before the MBREQ alternate function is enabled. Otherwise, unpredictable MBGNT behavior results. For more information, see Chapter 3, “Clocks and Power Manager”.

For more information on alternate functions, refer to “Pin Usage and Mapping” in the *Intel® PXA270 Processor Electrical, Mechanical, and Thermal Specification* and *Intel® PXA27x Processor Family Electrical, Mechanical, and Thermal Specification* (*Intel® PXA27x Processor Family EMTS*) for a summary of all block function pins.

MMCMD, MMDAT<1:0>, MMDAT<2>/MMCCS<0>, MMDAT<3>/MMCCS<1>, MSSDIO, SSPSCLK, SSPSCLK2, SSPSCLK3, SSPSFRM, SSPSFRM2, SSPSFRM3, LDD<17:0>, CIF_LV, CIF_FV, and the I^2C pins PWR_SDA, PWR_SCL, SDA, and SCL are special bidirectional GPIOs where the direction of the pin is controlled by the peripheral directly overriding the GPIO direction settings for these pins (GPDR). Notice also that the alternate function for input and output variants of these functions is the same. For example, configuring GPIO<112> with the alternate function 01 selects MMCMD in both input and output mode.

24.4.2.1 Special Function Bidirectional GPIOs

Table 24-2 shows the alternate mapping of all the GPIOs, with the special-function bidirectional GPIOs shown shaded. These GPIOs act like the special bidirectional GPIOs described above, except that the input and output function of the alternate function are not the same.

For example, GPIO<35> can be configured as SSPSFRM3 if it is configured for alternate function 3 AND it is configured as an input. Since this is a special-function bidirectional GPIO (as indicated by the shading), SSPSFRM3 functions as an input or as an output depending on how this signal is configured in the SSP3 controller, not how it is configured in the GPIO direction register. To configure SSPSFRM3 as an *output* on GPIO<35>, the GPIO Direction register must be configured as an *input*, the GPIO Alternate Function register must select alternate function 3 and the SSP3 Control register 1 must configure SSPSFRM3 as an output. If however, GPIO<35> was configured as an output and Alternate Function 3 was configured in the Alternate Function register, GPIO<35> would function as SSPTXD3. The other combinations of direction and alternate function selection for this signal function normally.

SSPTXD, SSPTXD2, and SSPTXD3 are not true bidirectional signals; however, they are listed as special-function bidirectional GPIOs because the SSP unit has the ability to three-state these signals (effectively overriding the direction register settings).

KP_MKOUT<7:0> are also not true bidirectional signals; however, they are listed as special-function bidirectional GPIOs because their function is altered during a matrix-scan cycle to three-state and pull-down the signal when not driving a logic 1 value. This modification prevents the potential of high current drain when multiple keys are pressed simultaneously.

Table 24-2. GPIO Alternate Functions (Sheet 1 of 4)

GPIO Pin	Pin Name	Alternate Function 1 (In)	Alternate Function 2 (In)	Alternate Function 3 (In)	Alternate Function 1 (Out)	Alternate Function 2 (Out)	Alternate Function 3 (Out)
0	GPIO<0>						
1	GPIO<1>/ nRESET_GPIO[4]						
2	SYS_EN[5]						
3	GPIO<3>/ PWR_SCL						
4	GPIO<4>/ PWR_SDA						
5	PWR_CAP<0>[5]						
6	PWR_CAP<1>[5]						
7	PWR_CAP<2>[5]						
8	PWR_CAP<3>[5]						
9	GPIO<9>			FFCTS	HZ_CLK		CHOUT<0>
10	GPIO<10>	FFDCD		USB_P3_5[7]	HZ_CLK		CHOUT<1>
11	GPIO<11>	EXT_SYNC<0>	SSPRXD2	USB_P3_1	CHOUT<0>	PWM_OUT<2>	48_MHz
12	GPIO<12>	EXT_SYNC<1>	CIF_DD<7>		CHOUT<1>	PWM_OUT<3>	48_MHz
13	GPIO<13>	CLK_EXT	KP_DKIN<7>	KP_MKIN<7>	SSPTXD2		
14	GPIO<14>	L_VSYNC	SSPSFRM2			SSPSFRM2	UCLK
15	GPIO<15>				nPCE<1>	nCS<1>	
16	GPIO<16>	KP_MKIN<5>				PWM_OUT<0>	FFTXD
17	GPIO<17>	KP_MKIN<6>	CIF_DD<6>			PWM_OUT<1>	
18	GPIO<18>	RDY					
19	GPIO<19>	SSPSCLK2		FFRXD	SSPSCLK2	L_CS	nURST
20	GPIO<20>	DREQ<0>	MBREQ		nSDCS<2>		
21	GPIO<21>				nSDCS<3>	DVAL<0>	MBGNT
22	GPIO<22>	SSPEXTCLK2	SSPSCLKEN2	SSPSCLK2	KP_MKOUT<7>	SSPSYSCLK2	SSPSCLK2
23	GPIO<23>		SSPSCLK		CIF_MCLK	SSPSCLK	
24	GPIO<24>	CIF_FV	SSPSFRM		CIF_FV	SSPSFRM	
25	GPIO<25>	CIF_LV			CIF_LV	SSPTXD	
26	GPIO<26>	SSPRXD	CIF_PCLK	FFCTS			
27	GPIO<27>	SSPEXTCLK	SSPSCLKEN	CIF_DD<0>	SSPSYSCLK		FFRTS
28	GPIO<28>	AC97_BITCLK	I2S_BITCLK	SSPSFRM	I2S_BITCLK		SSPSFRM
29	GPIO<29>	AC97_ SDATA_IN_0	I2S_SDATA_IN	SSPSCLK	SSPRXD2		SSPSCLK
30	GPIO<30>				I2S_SDATA_OUT	AC97_ SDATA_OUT	USB_P3_2
31	GPIO<31>				I2S_SYNC	AC97_SYNC	USB_P3_6
32	GPIO<32>				MSSCLK	MMCLK	

Table 24-2. GPIO Alternate Functions (Sheet 2 of 4)

GPIO Pin	Pin Name	Alternate Function 1 (In)	Alternate Function 2 (In)	Alternate Function 3 (In)	Alternate Function 1 (Out)	Alternate Function 2 (Out)	Alternate Function 3 (Out)
33	GPIO<33>	FFRXD	FFDSR		DVAL<1>	nCS<5>	MBGNT
34	GPIO<34>	FFRXD	KP_MKIN<3>	SSPSCLK3	USB_P2_2		SSPSCLK3
35	GPIO<35>	FFCTS	USB_P2_1	SSPSFRM3		KP_MKOUT<6>	SSPTXD3
36	GPIO<36>	FFDCD	SSPSCLK2	KP_MKIN<7>	USB_P2_4	SSPSCLK2	
37	GPIO<37>	FFDSR	SSPSFRM2	KP_MKIN<3>	USB_P2_8	SSPSFRM2	FFTXD
38	GPIO<38>	FFRI	KP_MKIN<4>	USB_P2_3	SSPTXD3	SSPTXD2[8]	PWM_OUT<0>
39	GPIO<39>	KP_MKIN<4>		SSPSFRM3	USB_P2_6	FFTXD	SSPSFRM3
40	GPIO<40>	SSPRXD2		USB_P2_5	KP_MKOUT<6>	FFDTR	SSPSCLK3
41	GPIO<41>	FFRXD	USB_P2_7	SSPRXD3	KP_MKOUT<7>	FFRTS	
42	GPIO<42>	BTRXD	ICP_RXD				CIF_MCLK
43	GPIO<43>			CIF_FV	ICP_TXD	BTTXD	CIF_FV
44	GPIO<44>	BTCTS		CIF_LV			CIF_LV
45	GPIO<45>			CIF_PCLK	AC97_SYSCLK	BTRTS	SSPSYSCLK3
46	GPIO<46>	ICP_RXD	STD_RXD			PWM_OUT<2>	
47	GPIO<47>	CIF_DD<0>			STD_TXD	ICP_TXD	PWM_OUT<3>
48	GPIO<48>	CIF_DD<5>			BB_OB_DAT<1>	nPOE	
49	GPIO<49>					nPWE	
50	GPIO<50>	CIF_DD<3>		SSPSCLK2	BB_OB_DAT<2>	nPIOR	SSPSCLK2
51	GPIO<51>	CIF_DD<2>			BB_OB_DAT<3>	nPIOW	
52	GPIO<52>	CIF_DD<4>	SSPSCLK3		BB_OB_CLK	SSPSCLK3	
53	GPIO<53>	FFRXD	USB_P2_3		BB_OB_STB	CIF_MCLK	SSPSYSCLK
54	GPIO<54>		BB_OB_WAIT	CIF_PCLK		nPCE<2>	
55	GPIO<55>	CIF_DD<1>	BB_IB_DAT<1>			nPREG	
56	GPIO<56>	nPWAIT	BB_IB_DAT<2>		USB_P3_4		
57	GPIO<57>	nIOIS16	BB_IB_DAT<3>				SSPTXD
58	GPIO<58>		LDD<0>			LDD<0>	
59	GPIO<59>		LDD<1>			LDD<1>	
60	GPIO<60>		LDD<2>			LDD<2>	
61	GPIO<61>		LDD<3>			LDD<3>	
62	GPIO<62>		LDD<4>			LDD<4>	
63	GPIO<63>		LDD<5>			LDD<5>	
64	GPIO<64>		LDD<6>			LDD<6>	
65	GPIO<65>		LDD<7>			LDD<7>	
66	GPIO<66>		LDD<8>			LDD<8>	
67	GPIO<67>		LDD<9>			LDD<9>	
68	GPIO<68>		LDD<10>			LDD<10>	

Table 24-2. GPIO Alternate Functions (Sheet 3 of 4)

GPIO Pin	Pin Name	Alternate Function 1 (In)	Alternate Function 2 (In)	Alternate Function 3 (In)	Alternate Function 1 (Out)	Alternate Function 2 (Out)	Alternate Function 3 (Out)
69	GPIO<69>		LDD<11>			LDD<11>	
70	GPIO<70>		LDD<12>			LDD<12>	
71	GPIO<71>		LDD<13>			LDD<13>	
72	GPIO<72>		LDD<14>			LDD<14>	
73	GPIO<73>		LDD<15>			LDD<15>	
74	GPIO<74>					L_FCLK_RD	
75	GPIO<75>					L_LCLK_A0	
76	GPIO<76>					L_PCLK_WR	
77	GPIO<77>					L_BIAS	
78	GPIO<78>				nPCE<2>	nCS<2>	
79	GPIO<79>				PSKTSEL	nCS<3>	PWM_OUT<2>
80	GPIO<80>	DREQ<1>	MBREQ			nCS<4>	PWM_OUT<3>
81	GPIO<81>		CIF_DD<0>		SSPTXD3	BB_OB_DAT<0>	
82	GPIO<82>	SSPRXD3	BB_IB_DAT<0>	CIF_DD<5>			FFDTR
83	GPIO<83>	SSPSFRM3	BB_IB_CLK	CIF_DD<4>	SSPSFRM3	FFTXD	FFRTS
84	GPIO<84>	SSPCLK3	BB_IB_STB	CIF_FV	SSPCLK3		CIF_FV
85	GPIO<85>	FFRXD	DREQ<2>	CIF_LV	nPCE<1>	BB_IB_WAIT	CIF_LV
86	GPIO<86>	SSPRXD2	LDD<16>	USB_P3_5	nPCE<1>	LDD<16>	
87	GPIO<87>	nPCE<2>	LDD<17>	USB_P3_1	SSPTXD2	LDD<17>	SSPSFRM2
88	GPIO<88>	USBHPWR<1>	SSPRXD2	SSPSFRM2		SSPTXD2[8]	SSPSFRM2
89	GPIO<89>	SSPRXD3		FFRI	AC97_SYSCLK	USBHPEN<1>	SSPTXD2
90	GPIO<90>	KP_MKIN<5>	USB_P3_5	CIF_DD<4>		nURST	
91	GPIO<91>	KP_MKIN<6>	USB_P3_1	CIF_DD<5>		UCLK	
92	GPIO<92>	MMDAT<0>			MMDAT<0>	MSBS	
93	GPIO<93>	KP_DKIN<0>	CIF_DD<6>		AC97_ SDATA_OUT		
94	GPIO<94>	KP_DKIN<1>	CIF_DD<5>		AC97_SYNC		
95	GPIO<95>	KP_DKIN<2>	CIF_DD<4>	KP_MKIN<6>	AC97_RESET_n		
96	GPIO<96>	KP_DKIN<3>	MBREQ	FFRXD		DVAL<1>	KP_MKOUT<6>
97	GPIO<97>	KP_DKIN<4>	DREQ<1>	KP_MKIN<3>		MBGNT	
98	GPIO<98>	KP_DKIN<5>	CIF_DD<0>	KP_MKIN<4>	AC97_SYSCLK		FFRTS
99	GPIO<99>	KP_DKIN<6>	AC97_ SDATA_IN_1	KP_MKIN<5>			FFTXD
100	GPIO<100>	KP_MKIN<0>	DREQ<2>	FFCTS			
101	GPIO<101>	KP_MKIN<1>					
102	GPIO<102>	KP_MKIN<2>		FFRXD	nPCE<1>		
103	GPIO<103>	CIF_DD<3>				KP_MKOUT<0>	

Table 24-2. GPIO Alternate Functions (Sheet 4 of 4)

GPIO Pin	Pin Name	Alternate Function 1 (In)	Alternate Function 2 (In)	Alternate Function 3 (In)	Alternate Function 1 (Out)	Alternate Function 2 (Out)	Alternate Function 3 (Out)
104	GPIO<104>	CIF_DD<2>			PSKTSEL	KP_MKOUT<1>	
105	GPIO<105>	CIF_DD<1>			nPCE<2>	KP_MKOUT<2>	
106	GPIO<106>	CIF_DD<9>				KP_MKOUT<3>	
107	GPIO<107>	CIF_DD<8>				KP_MKOUT<4>	
108	GPIO<108>	CIF_DD<7>			CHOUT<0>	KP_MKOUT<5>	
109	GPIO<109>	MMDAT<1>	MSSDIO		MMDAT<1>	MSSDIO	
110	GPIO<110>	MMDAT<2>/ MMCCS<0>			MMDAT<2>/ MMCCS<0>		
111	GPIO<111>	MMDAT<3>/ MMCCS<1>			MMDAT<3>/ MMCCS<1>		
112	GPIO<112>	MMCMD	nMSINS		MMCMD		
113	GPIO<113>			USB_P3_3	I2S_SYSCLK	AC97_RESET_n	
114[1]	GPIO<114>	CIF_DD<1>			UEN	UVS0	
115[2]	GPIO<115>	DREQ<0>	CIF_DD<3>	MBREQ	UEN	nUVS1	PWM_OUT<1>
116[3]	GPIO<116>	CIF_DD<2>	AC97_ SDATA_IN_0	UDET	DVAL<0>	nUVS2	MBGNT
117	GPIO<117>	SCL			SCL		
118	GPIO<118>	SDA			SDA		
119	GPIO<119>	USBHPWR<2>					
120	GPIO<120>					USBHPEN<2>	

NOTES:
1. GPIO<114> signal is driven by UEN regardless of the alternate function selection if PUCR[USIM114] is set. GPDR3[PD114] must be set must be cleared for this configuration. See Section 3.8.1.14, “Power Manager USIM Card Control/Status Register (PUCR)” on page 3-90.
2. GPIO<115> signal is driven by UEN regardless of the alternate function selection if PUCR[USIM115] is set. GPDR3[PD115] must be set must be cleared for this configuration. See Section 3.8.1.14, “Power Manager USIM Card Control/Status Register (PUCR)” on page 3-90.
3. GPIO<116> signal is configured as UDET regardless of the alternate function selection if PUCR[EN_UDET] bit is set. GPDR3[PD116] must be cleared for this configuration. See Section 3.8.1.14, “Power Manager USIM Card Control/Status Register (PUCR)” on page 3-90.
4. nRESET_GPIO is not selected through use of the GPIO registers. nRESET_GPIO is configured through the Power Manager. For more details, see Section 3.4.6.1, “Enabling GPIO Reset” on page 3-10.
5. This signal is dedicated to the function shown and is not available as a GPIO.
6. GPIO<120:119> are supported in the PXA271, PXA272, and PXA273 processors only.
7. Shaded cells indicate special-function bidirectional GPIOs, as described in Section 24.4.2.1.
8. When GPIO<88> alternate function 2 out is selected, GPIO<38> alternate function 2 out is automatically changed from SSP2TXD2 to SSPRXD2. This operation is enabled as of the C5 stepping.

24.4.2.2 Exceptions to GPDR Direction Bits

Typically, the GPIO output enable is controlled by the GPIO Direction register (GPDR). However there are two instances when GPDR is used like an additional alternate-function select function. For these two cases, the combination of the alternate function selected in the GAFR and the direction configured in the GPDR cause an output alternate-function to operate as an input and vice verse. These two cases are detailed below.

- Configuring GPIO<29> alternate function 1 *output* configures the SSPRXD2 *input* function. When the SSPRXD2 function is configured on GPIO<29>, the GPIO<29> pad output enable is inverted to configure the pad as an input.
- Configuring GPIO<87> alternate function 1 *input* configures the nPCE<2> *output* function. When the nPCE<2> function is configured on GPIO<87>, the GPIO<87> pad input enable is inverted to configure the pad as an output.

24.4.2.3 Alternate Function Programming Example

The procedure for configuring the alternate function registers is described here with an example.

Table 24-3 contains the list of alternate functions and an example function selection (shaded). After the deassertion of any RESET, GPDR0[15:0] configures GPIO pins in this example as inputs. GAFR0_L[31:0] is 0x0000_0000 to indicate normal GPIO function.

In this example, the boxes that are shaded in Table 24-3 are to be selected as alternate functions. For simplicity, assume that GPIO<31:16> are inputs configured to perform the normal GPIO function.

This programming sequence must be followed to program the GPIO alternate functions out of reset:

1. WRITE GPSR0 0x0000_8000—This sets GPIO<15> (active-low chip-select) when it is configured as an output.
2. WRITE GPDR0 0x0000_8604—GPIO<2>, GPIO<9>, GPIO<10> and GPIO<15> as outputs. GPIO<15> drives 1 even before the alternate function information is programmed. This is required for active-low outputs.
3. WRITE GAFR0_L 0x9D74_0004—This maps the correct alternate functions of GPIO<15:0>.

For GPIOs that need to be configured as outputs, first program the GPSR and GPCR signals such that when the pin direction is changed by means of setting the bit in the GPDR register, 1 is driven for active-high signals, and 0 is driven for active-low signals.

Table 24-3. GPIO Alternate Function Programming Example (Sheet 1 of 2)

GPIO Pin	Alternate Function 1 (In)	Alternate Function 2 (In)	Alternate Function 3 (In)	Alternate Function 1 (Out)	Alternate Function 2 (Out)	Alternate Function 3 (Out)	Pin Name
0							GPIO<0>
1							GPIO<1>/ nRESET_GPIO[1]
2							SYS_EN[2]
3							GPIO<3>/ PWR_SCL
4							GPIO<4>/ PWR_SDA
5							PWR_CAP<0>[2]
6							PWR_CAP<1>[2]
7							PWR_CAP<2>[2]
8							PWR_CAP<3>[2]

Table 24-3. GPIO Alternate Function Programming Example (Sheet 2 of 2)

GPIO Pin	Alternate Function 1 (In)	Alternate Function 2 (In)	Alternate Function 3 (In)	Alternate Function 1 (Out)	Alternate Function 2 (Out)	Alternate Function 3 (Out)	Pin Name
9			FFCTS	HZ_CLK		CHOUT<0>	GPIO<9>
10	FFDCD		USB_P3_5	HZ_CLK		CHOUT<1>	GPIO<10>
11	EXT_SYNC<0>	SSPRXD2	USB_P3_1	CHOUT<0>	PWM_OUT<2>	48_MHz	GPIO<11>
12	EXT_SYNC<1>	CIF_DD<7>		CHOUT<1>	PWM_OUT<3>	48_MHz	GPIO<12>
13	CLK_EXT	KP_DKIN<7>	KP_MKIN<7>	SSPTXD2			GPIO<13>
14	L_VSYNC	SSPSFRM2			SSPSFRM2	UCLK	GPIO<14>
15				nPCE<1>	nCS<1>		GPIO<15>

NOTES:
1. RESET_GPIO is not selected through use of the GPIO registers. nRESET_GPIO is configured through the Power Manager. For more details, see Section 3.4.6.1, “Enabling GPIO Reset” on page 3-10.
2. This signal is dedicated to the function shown and is not available as a GPIO.

24.5 Register Descriptions

There are a total of thirty-six 32-bit registers within the GPIO control block. There are nine distinct register functions, and there are four sets of each of the nine registers to serve the 121 GPIOs. The various functions of the nine registers corresponding to each GPIO pin include:

- Four monitor pin states (GPLRx).
- Eight control output pin states (GPSRx, GPCRx).
- Four control pin directions (GPDRx).
- Eight control whether rising edges or falling edges are detected (GRERx & GFERx).
- Four indicate when specified edge types have been detected on pins (GEDRx).
- Eight determine whether a pin is used as a normal GPIO or whether it is to be taken over by one of three possible alternate functions (GAFR_Lx, GAFR_Ux).

Table 24-4. GPIO Register Definitions

Register Type	Register Function	GPIO <15:0>	GPIO <31:16>	GPIO <47:32>	GPIO <63:48>	GPIO <79:64>	GPIO <95:80>	GPIO <111:96>	GPIO <120:112>
GPLR	Monitor Pin State	GPLR0		GPLR1		GPLR2		GPLR3	
GPSR	Control Output Pin State	GPSR0		GPSR1		GPSR2		GPSR3	
GPCR		GPCR0		GPCR1		GPCR2		GPCR3	
GPDR	Set Pin Direction	GPDR0		GPDR1		GPDR2		GPDR3	
GRER	Detect Rising/Falling Edge	GRER0		GRER1		GRER2		GRER3	
GFER		GFER0		GFER1		GFER2		GFER3	
GEDR	Detect Edge Type	GEDR0		GEDR1		GEDR2		GEDR3	
GAFR	Set Alternate Functions	GAFR0_L	GAFR0_U	GAFR1_L	GAFR1_U	GAFR2_L	GAFR2_U	GAFR3_L	GAFR3_U

NOTES:
1. For the alternate function registers, the designator _L signifies that the lower 16 GPIOs' alternate functions are configured by that register and _U designates that the upper 16 GPIOs' alternate functions are configured by that register.
2. GPLR0<8:5>, GPLR0<2>, GPLR3<31:25>, GPSR0<8:5>, GPSR0<2>, GPSR3<31:25>, GPCR0<8:5>, GPCR0<2>, GPCR3<31:25>, GPDR0<8:5>, GPDR0<2>, GPDR3<31:25>, GRER0<8:5>, GRER0<2>, GRER3<31:25>, GFER0<8:5>, GFER0<2>, GFER3<31:25>, GEDR0<8:5>, GEDR0<2>, GEDR3<31:25> and GAFR0_L<8:5>, GAFR0_L<2>, GAFR3_U<31:18> are reserved bits. Write 0b0 to these bits and ignore all reads from these bits.
3. All GPIO registers are initialized to 0x0 at reset, which results in all GPIO pins being initialized as inputs.

24.5.1 GPIO Pin-Direction Registers (GPDR)

Whether a pin is an input or an output is controlled by programming the GPIO pin direction registers (GPDR0/1/2/3). The GPDR registers contain one direction-control bit for each of the 121[1]GPIO pins. If a direction bit is programmed to 0b1, the GPIO is an output. If it is programmed to 0b0, it is an input.

Note: A reset clears all bits in the GPDR0/1/2/3 registers and configures all GPIO pins as inputs. For correct operation as an input, PSSR[RDH] must be clear (see 'Section 3.8.1.2, "Power Manager Sleep Status Register (PSSR)" on page 3-70).

Table 24-5 through Table 24-8 show the bit descriptions of the GPIO Pin Direction registers.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

1. Although the PXA27x processor has 121 GPIO pins, only 119 are bonded out in the PXA270 processor. All 121 GPIOs are available in the PXA271, PXA272, and PXA273 processors. Additionally, five of these GPIOs have dedicated functions and are not available as GPIOs

Table 24-5. GPDR0 Bit Definitions

Physical Address 0x40E0_000C — GPDR0 — GPIO Controller

	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
User Settings																																
Bit	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
	PD31	PD30	PD29	PD28	PD27	PD26	PD25	PD24	PD23	PD22	PD21	PD20	PD19	PD18	PD17	PD16	PD15	PD14	PD13	PD12	PD11	PD10	PD9	reserved				PD4	PD3	reserved	PD1	PD0
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

Bits	Access	Name	Description
<31:9>	R/W	PD*x*	GPIO Pin Direction ‘x’ (where x = 9 to 31) This field configures the direction of each GPIO. 0 = Pin configured as an input. 1 = Pin configured as an output.
<8:5>	—	—	reserved
<4:3>	R/W	PD*x*	GPIO Pin Direction ‘x’ (where x = 3 to 4) This field configures the direction of each GPIO. 0 = Pin configured as an input. 1 = Pin configured as an output.
<2>	—	—	reserved
<1:0>	R/W	PD*x*	GPIO Pin Direction ‘x’ (where x = 0 to 1) This field configures the direction of each GPIO. 0 = Pin configured as an input. 1 = Pin configured as an output.

Table 24-6. GPDR1 Bit Definitions

Physical Address 0x40E0_0010 — GPDR1 — GPIO Controller

	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
User Settings																																
Bit	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
	PD63	PD62	PD61	PD60	PD59	PD58	PD57	PD56	PD55	PD54	PD53	PD52	PD51	PD50	PD49	PD48	PD47	PD46	PD45	PD44	PD43	PD42	PD41	PD40	PD39	PD38	PD37	PD36	PD35	PD34	PD33	PD32
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

Bits	Access	Name	Description
<31:0>	R/W	PD*x*	GPIO Pin Direction ‘x’ (where x = 32 to 63) This field configures the direction of each GPIO. 0 = Pin configured as an input. 1 = Pin configured as an output.

Table 24-7. GPDR2 Bit Definitions

Physical Address 0x40E0_0014	GPDR2	GPIO Controller

	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
User Settings																																
Bit	PD95	PD94	PD93	PD92	PD91	PD90	PD89	PD88	PD87	PD86	PD85	PD84	PD83	PD82	PD81	PD80	PD79	PD78	PD77	PD76	PD75	PD74	PD73	PD72	PD71	PD70	PD69	PD68	PD67	PD66	PD65	PD64
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

Bits	Access	Name	Description
<31:0>	R/W	PD*x*	GPIO Pin Direction ‘x’ (where x = 64 to 95) This field configures the direction of each GPIO. 0 = Pin configured as an input. 1 = Pin configured as an output.

Table 24-8. GPDR3 Bit Definitions

Physical Address 0x40E0_010C	GPDR3	GPIO Controller

	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
User Settings																																
User Settings																																
Bit	reserved							PD120	PD119	PD118	PD117	PD116	PD115	PD114	PD113	PD112	PD111	PD110	PD109	PD108	PD107	PD106	PD105	PD104	PD103	PD102	PD101	PD100	PD99	PD98	PD97	PD96
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

Bits	Access	Name	Description
<31:25>	—	—	reserved
<24:0>	R/W	PD*x*	GPIO Pin Direction ‘x’ (where x = 96 to 120) This field configures the direction of each GPIO. 0 = Pin configured as an input. 1 = Pin configured as an output.

24.5.2 GPIO Pin-Output Set Registers (GPSR) and GPIO Pin-Output Clear Registers (GPCR)

When a GPIO is configured as an output, the state of the pin is controlled by writing to either the GPIO Pin Output Set registers (GPSR0/1/2/3) or the GPIO Pin Output Clear registers (GPCR0/1/2/3). An output pin is set by setting to its corresponding bit within the GPSR. To clear an output pin, set the corresponding GPCR bit.

Writing 0b0 to any of the GPSR or GPCR bits has no effect on the state of the pin. Writing 0b1 to a GPSR or GPCR bit corresponding to a pin that is configured as an input is effective only after the pin is configured as an output.

Note: When a GPIO pin is reconfigured as an output, the most recent write to GPSR or GPCR determines the pin's output state.

The GPIO Pin-Output Set registers (GPSR0/1/2/3) and GPIO Pin-Output Clear registers (GPCR0/1/2/3) are write-only registers. Reads return unpredictable values.

Table 24-9 through Table 24-12 show the bit descriptions of the GPIO Pin-Output Set registers. Table 24-13 through Table 24-16 show the bit descriptions of the GPIO Pin-Output Clear registers.

These are write-only registers. Write 0b0 to reserved bits.

Table 24-9. GPSR0 Bit Definitions

Physical Address 0x40E0_0018 — **GPSR0** — **GPIO Controller**

	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
User Settings																																
Bit	PS31	PS30	PS29	PS28	PS27	PS26	PS25	PS24	PS23	PS22	PS21	PS20	PS19	PS18	PS17	PS16	PS15	PS14	PS13	PS12	PS11	PS10	PS9	reserved				PS4	PS3	reserved	PS1	PS0
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

Bits	Access	Name	Description
<31:9>	W	PS*x*	GPIO Pin Set 'x' (where x = 9 to 31) This field configures the output level of each GPIO. 0 = Pin level unaffected. 1 = If pin configured as an output, set pin level high (one).
<8:5>	—	—	reserved
<4:3>	W	PS*x*	GPIO Pin Set 'x' (where x = 3 to 4) This field configures the output level of each GPIO. 0 = Pin level unaffected. 1 = If pin configured as an output, set pin level high (one).
<2>	—	—	reserved
<1:0>	W	PS*x*	GPIO Pin Set 'x' (where x = 0 to 1) This field configures the output level of each GPIO. 0 = Pin level unaffected. 1 = If pin configured as an output, set pin level high (one).

Table 24-10. GPSR1 Bit Definitions

Physical Address 0x40E0_001C	GPSR1	GPIO Controller

User Settings																															
Bit	**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**
	PS63	PS62	PS61	PS60	PS59	PS58	PS57	PS56	PS55	PS54	PS53	PS52	PS51	PS50	PS49	PS48	PS47	PS46	PS45	PS44	PS43	PS42	PS41	PS40	PS39	PS38	PS37	PS36	PS35	PS34	PS33	PS32
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

Bits	Access	Name	Description
<31:0>	W	PS*x*	GPIO Pin Set ‘x’ (where x = 32 to 63) This field configures the output level of each GPIO. 0 = Pin level unaffected. 1 = If pin configured as an output, set pin level high (one).

Table 24-11. GPSR2 Bit Definitions

Physical Address 0x40E0_0020	GPSR2	GPIO Controller

User Settings																															
Bit	**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**
	PS95	PS94	PS93	PS92	PS91	PS90	PS89	PS88	PS87	PS86	PS85	PS84	PS83	PS82	PS81	PS80	PS79	PS78	PS77	PS76	PS75	PS74	PS73	PS72	PS71	PS70	PS69	PS68	PS67	PS66	PS65	PS64
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

Bits	Access	Name	Description
<31:0>	W	PS*x*	GPIO Pin Set ‘x’ (where x = 64 to 95) This field configures the output level of each GPIO. 0 = Pin level unaffected. 1 = If pin configured as an output, set pin level high (one).

Table 24-12. GPSR3 Bit Definitions

Physical Address 0x40E0_0118 | GPSR3 | GPIO Controller

	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
User Settings																																
User Settings																																
Bit	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								PS120	PS119	PS118	PS117	PS116	PS115	PS114	PS113	PS112	PS111	PS110	PS109	PS108	PS107	PS106	PS105	PS104	PS103	PS102	PS101	PS100	PS99	PS98	PS97	PS96
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

Bits	Access	Name	Description
<31:25>	—	—	reserved
<24:0>	W	PS*x*	GPIO Pin Set ‘x’ (where x = 96 to 120) This field configures the output level of each GPIO. 0 = Pin level unaffected. 1 = If pin configured as an output, set pin level high (one).

Table 24-13. GPCR0 Bit Definitions

Physical Address 0x40E0_0024 | GPCR0 | GPIO Controller

	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
User Settings																																
Bit	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
	PC31	PC30	PC29	PC28	PC27	PC26	PC25	PC24	PC23	PC22	PC21	PC20	PC19	PC18	PC17	PC16	PC15	PC14	PC13	PC12	PC11	PC10	PC9	reserved				PC4	PC3	reserved	PC1	PC0
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

Bits	Access	Name	Description
<31:9>	W	PC*x*	GPIO Pin Clear ‘x’ (where x = 9 to 31) This field configures the output level of each GPIO. 0 = Pin level unaffected. 1 = If pin configured as an output, clear pin level low (zero).
<8:5>	—	—	reserved
<4:3>	W	PC*x*	GPIO Pin Clear ‘x’ (where x = 3 to 4) This field configures the output level of each GPIO. 0 = Pin level unaffected. 1 = If pin configured as an output, clear pin level low (zero).
<2>	—	—	reserved
<1:0>	W	PC*x*	GPIO Pin Clear ‘x’ (where x = 0 to 1) This field configures the output level of each GPIO. 0 = Pin level unaffected. 1 = If pin configured as an output, clear pin level low (zero).

Table 24-14. GPCR1 Bit Definitions

Physical Address 0x40E0_0028	GPCR1	GPIO Controller

User Settings																															
Bit	**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**
	PC63	PC62	PC61	PC60	PC59	PC58	PC57	PC56	PC55	PC54	PC53	PC52	PC51	PC50	PC49	PC48	PC47	PC46	PC45	PC44	PC43	PC42	PC41	PC40	PC39	PC38	PC37	PC36	PC35	PC34	PC33	PC32
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

Bits	Access	Name	Description
<31:0>	W	PC*x*	GPIO Pin Clear ‘x’ (where x = 32 to 63) This field configures the output level of each GPIO. 0 = Pin level unaffected. 0 = If pin configured as an output, clear pin level low (zero).

Table 24-15. GPCR2 Bit Definitions

Physical Address 0x40E0_002C	GPCR2	GPIO Controller

User Settings																															
Bit	**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**
	PC95	PC94	PC93	PC92	PC91	PC90	PC89	PC88	PC87	PC86	PC85	PC84	PC83	PC82	PC81	PC80	PC79	PC78	PC77	PC76	PC75	PC74	PC73	PC72	PC71	PC70	PC69	PC68	PC67	PC66	PC65	PC64
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

Bits	Access	Name	Description
<31:0>	W	PC*x*	GPIO Pin Clear ‘x’ (where x = 64 to 95) This field configures the output level of each GPIO. 0 = Pin level unaffected. 1 = If pin configured as an output, clear pin level low (zero).

Table 24-16. GPCR3 Bit Definitions

Physical Address 0x40E0_0124								GPCR3																GPIO Controller							
User Settings																															
User Settings																															
Bit	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							PC120	PC119	PC118	PC117	PC116	PC115	PC114	PC113	PC112	PC111	PC110	PC109	PC108	PC107	PC106	PC105	PC104	PC103	PC102	PC101	PC100	PC99	PC98	PC97	PC96
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

Bits	Access	Name	Description
<31:25>	—	—	reserved
<24:0>	W	PC*x*	GPIO Pin Clear ‘x’ (where x = 96 to 120) This field configures the output level of each GPIO. 0 = Pin level unaffected. 1 = If pin configured as an output, clear pin level low (zero).

24.5.3 GPIO Rising-Edge Detect Enable Registers (GRER0/1/2/3) and Falling-Edge Detect Enable Registers (GFER0/1/2/3)

Each GPIO can also be programmed to detect a rising-edge, falling-edge, or either transition on a pin. When an edge is detected that matches the type of edge programmed for the pin, a status bit is set. The interrupt controller can be programmed so that an interrupt is signaled to the core when any of these status bits is set. Additionally, the interrupt controller can be programmed so that a subset of the status bits causes the PXA27x processor to wake from sleep/deep-sleep mode when it is set. Refer to Chapter 3, “Clocks and Power Manager” for more information on which status bits can cause a wake-up from sleep/deep-sleep mode.

The GPIO Rising-Edge Detect Enable register (GRER0/1/2/3) and Falling Edge Detect Enable register (GFER0/1/2/3) select the type of transition on a GPIO pin that causes a bit within the GPIO Edge Detect Enable Status register (GEDR0/1/2/3) to be set. For a given GPIO pin, its corresponding GRER bit is set, causing a GEDR status bit to be set when the pin transitions from logic level zero to logic level one. Likewise, GFER sets the corresponding GEDR status bit when a transition from logic level one to logic level zero occurs. When the corresponding bits are set in both registers, either a falling- or a rising-edge transition causes the corresponding GEDR status bit to be set.

Note: Minimum pulse-width timing requirements exist to guarantee that an edge or level transition is detected. For more information, refer to the *Intel® PXA27x Processor Family EMTS*, “GPIO AC Timing Specifications.”

Table 24-17 through Table 24-20 show the descriptions of the GPIO Rising-Edge Detect Enable registers. Table 24-21 through Table 24-24 show the bit descriptions of the GPIO Falling Edge Detect Enable registers.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 24-17. GRER0 Bit Definitions

Physical Address 0x40E0_0030	GRER0	GPIO Controller

User Settings																															
Bit	**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**
	RE31	RE30	RE29	RE28	RE27	RE26	RE25	RE24	RE23	RE22	RE21	RE20	RE19	RE18	RE17	RE16	RE15	RE14	RE13	RE12	RE11	RE10	RE9	reserved				RE4	RE3	reserved	RE1	RE0
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

Bits	Access	Name	Description
<31:9>	R/W	RE*x*	GPIO Pin Rising-Edge Detect Enable Bit ‘x’ (where x = 9 to 31) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a rising edge is detected on the GPIO pin.
<8:5>	—	—	reserved.
<4:3>	R/W	RE*x*	GPIO Pin Rising-Edge Detect Enable Bit ‘x’ (where x = 3 to 4) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a rising edge is detected on the GPIO pin.
<2>	—	—	reserved
<1:0>	R/W	RE*x*	GPIO Pin Rising-Edge Detect Enable Bit ‘x’ (where x = 0 to 1) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a rising edge is detected on the GPIO pin.

Table 24-18. GRER1 Bit Definitions

Physical Address 0x40E0_0034	GRER1	GPIO Controller

User Settings																															
Bit	**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**
	RE63	RE62	RE61	RE60	RE59	RE58	RE57	RE56	RE55	RE54	RE53	RE52	RE51	RE50	RE49	RE48	RE47	RE46	RE45	RE44	RE43	RE42	RE41	RE40	RE39	RE38	RE37	RE36	RE35	RE34	RE33	RE32
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

Bits	Access	Name	Description
<31:0>	R/W	RE*x*	GPIO Pin Rising-Edge Detect Enable Bit ‘x’ (where x = 32 to 63) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a rising edge is detected on the GPIO pin.

Table 24-19. GRER2 Bit Definitions

Physical Address 0x40E0_0038	GRER2	GPIO Controller

	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
User Settings																																
Bit	RE95	RE94	RE93	RE92	RE91	RE90	RE89	RE88	RE87	RE86	RE85	RE84	RE83	RE82	RE81	RE80	RE79	RE78	RE77	RE76	RE75	RE74	RE73	RE72	RE71	RE70	RE69	RE68	RE67	RE66	RE65	RE64
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

Bits	Access	Name	Description
<31:0>	R/W	RE*x*	GPIO Pin Rising-Edge Detect Enable Bit ‘x’ (where x = 64 to 95) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a rising edge is detected on the GPIO pin.

Table 24-20. GRER3 Bit Definitions

Physical Address 0x40E0_0130	GRER3	GPIO Controller

	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
User Settings																																
Bit	reserved							RE120	RE119	RE118	RE117	RE116	RE115	RE114	RE113	RE112	RE111	RE110	RE109	RE108	RE107	RE106	RE105	RE104	RE103	RE102	RE101	RE100	RE99	RE98	RE97	RE96
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

Bits	Access	Name	Description
<31:25>	—	—	reserved
<24:23>	R/W	RE*x*	GPIO Pin Falling Edge detect Enable bit ‘x’ (where x = 119 to 120) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a falling edge is detected on the GPIO pin. **NOTE:** These GPIO selections are supported on the PXA271, PXA272, and PXA273 processors only. These bits are reserved in the PXA270 processor.
<22:0>	R/W	RE*x*	GPIO Pin Rising-Edge Detect Enable Bit ‘x’ (where x = 96 to 118) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a rising edge is detected on the GPIO pin.

Table 24-21. GFER0 Bit Definitions

Physical Address 0x40E0_003C — GFER0 — GPIO Controller

Bit	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
User Settings																																
	FE31	FE30	FE29	FE28	FE27	FE26	FE25	FE24	FE23	FE22	FE21	FE20	FE19	FE18	FE17	FE16	FE15	FE14	FE13	FE12	FE11	FE10	FE9	reserved				FE4	FE3	reserved	FE1	FE0
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

Bits	Access	Name	Description
<31:9>	R/W	FE*x*	GPIO Pin Falling Edge detect Enable bit ‘x’ (where x = 9 to 31) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a falling edge is detected on the GPIO pin.
<8:5>	—	—	reserved
<4:3>	R/W	FE*x*	GPIO Pin Falling Edge detect Enable bit ‘x’ (where x = 3 to 4) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a falling edge is detected on the GPIO pin.
<2>	—	—	reserved
<1:0>	R/W	FE*x*	GPIO Pin Falling Edge detect Enable bit ‘x’ (where x = 0 to 1) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a falling edge is detected on the GPIO pin.

Table 24-22. GFER1 Bit Definitions

Physical Address 0x40E0_0040 — GFER1 — GPIO Controller

Bit	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
User Settings																																
	FE63	FE62	FE61	FE60	FE59	FE58	FE57	FE56	FE55	FE54	FE53	FE52	FE51	FE50	FE49	FE48	FE47	FE46	FE45	FE44	FE43	FE42	FE41	FE40	FE39	FE38	FE37	FE36	FE35	FE34	FE33	FE32
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

Bits	Access	Name	Description
<31:0>	R/W	FE*x*	GPIO Pin Falling Edge detect Enable bit ‘x’ (where x = 32 to 63) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a falling edge is detected on the GPIO pin.

Table 24-23. GFER2 Bit Definitions

Physical Address 0x40E0_0044	GFER2	GPIO Controller

Bit	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
User Settings																																
	FE95	FE94	FE93	FE92	FE91	FE90	FE89	FE88	FE87	FE86	FE85	FE84	FE83	FE82	FE81	FE80	FE79	FE78	FE77	FE76	FE75	FE74	FE73	FE72	FE71	FE70	FE69	FE68	FE67	FE66	FE65	FE64
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

Bits	Access	Name	Description
<31:0>	R/W	FE*x*	GPIO Pin Falling Edge detect Enable bit ‘x’ (where x = 64 to 95) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a falling edge is detected on the GPIO pin.

Table 24-24. GFER3 Bit Definitions

Physical Address 0x40E0_013C	GFER3	GPIO Controller

Bit	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
User Settings																																
	reserved							FE120	FE119	FE118	FE117	FE116	FE115	FE114	FE113	FE112	FE111	FE110	FE109	FE108	FE107	FE106	FE105	FE104	FE103	FE102	FE101	FE100	FE99	FE98	FE97	FE96
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

Bits	Access	Name	Description
<31:25>	—	—	reserved
<24:23>	R/W	FE*x*	GPIO Pin Falling Edge detect Enable bit ‘x’ (where x = 119 to 120) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a falling edge is detected on the GPIO pin. **NOTE:** These GPIO selections are supported on the PXA271, PXA272, and PXA273 processors only. These bits are reserved in the PXA270 processor.
<22:0>	R/W	FE*x*	GPIO Pin Falling Edge detect Enable bit ‘x’ (where x = 96 to 118) This field configures the output level of each GPIO. 0 = Disables rising-edge detect enable. 1 = Sets corresponding GEDR status bit when a falling edge is detected on the GPIO pin.

24.5.4 GPIO Alternate Function Register (GAFR)

The GPIO alternate function registers (GAFR0/1/2/3) contain select bits that correspond to the 121 GPIO pins. Each GPIO can be configured to be either a generic GPIO pin, one of three alternate input functions, or one of three alternate output functions. To select any of the alternate functions, the GPDR register must configure the GPIO to be an input. Similarly, only GPIOs configured as outputs by the GPDR can be configured for alternate output functions. Each GPIO pin has a pair of bits assigned to it whose values determine which function (normal GPIO, alternate function 1, alternate function 2 or alternate function 3) the GPIO performs. The function selected is determined by writing the GAFR bit pair as below:

- 0b00 indicates normal GPIO function
- 0b01 selects alternate input function 1 (ALT_FN_1_IN) or alternate output function 1 (ALT_FN_1_OUT)
- 0b10 selects alternate input function 2 (ALT_FN_2_IN) or alternate output function 2 (ALT_FN_2_OUT)
- 0b11 selects alternate input function 3 (ALT_FN_3_IN) or alternate output function 3 (ALT_FN_3_OUT)

See Section 24.4.2 for details on alternate functions. The assignment of bits in the GAFR registers that correspond to the GPIO pins is defined in Table 24-2. For additional information on an alternate function, see “Pin Usage and Mapping” in the *Intel® PXA27x Processor Family EMTS* for a summary of all block function pins.

GPIO<0> is reserved because of its special use during sleep mode and is not available for alternate functions. Other GPIO pins are used for wake-up from sleep mode (see Section 24.4.1 for additional information). GPIO<3> and GPIO<1:0> are used exclusively for wake-up from deep-sleep mode. If nBATT_FAULT or nVDD_FAULT is asserted, GPIO<1:0> can be use for wake-up. The wake-up functionality is described in Section 3.6.9.4, “Sleep Exit” on page 3-47.

All GAFR registers are cleared to all zeroes on reset conditions. GAFR0_L[5:4], GAFR0_L[17:10], and GAFR3_U[31:18] are reserved.

Note: The power manager can override the alternate function of GPIO<10:9> and GPIO<4:3> directly. See Section 3.4.6.1, “Enabling GPIO Reset” on page 3-10 for more details.

Note: Configuring a GPIO to map to an alternate function that is not available causes indeterminate results.

A subset of the alternate functions is the special bidirectional GPIOs where the direction of the pin is controlled by the peripheral directly—overriding the GPIO direction settings for these pins (GPDR). This occurs with signals for which the alternate function for input and output variants of these functions is the same and for some signals having different input and output alternate functions. For further information on these signals, refer to Section 24.4.2.

Table 24-25 through Table 24-32 show the bitmaps of the GPIO Alternate Function Select registers.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 24-25. GAFR0_L Bit Definitions

Physical Address 0x40E0_0054 — GAFR0_L — GPIO Controller

User Settings

Bit	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
	AF15		AF14		AF13		AF12		AF11		AF10		AF9		reserved								AF4		AF3		reserved		AF1		AF0	
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

Bits	Access	Name	Description
<31:18>	R/W	AF*x*	GPIO ‘x’ Alternate Function Select Bits (where x = 9 to 15) A bit-pair in this register determines the corresponding GPIO pin’s functionality as one of the alternate functions that is mapped to it or as a generic GPIO pin. 0b00 = The corresponding GPIO pin (GPIO<x>) is used as a general-purpose I/O. 0b01 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 1. 0b10 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 2. 0b11 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 3.
<17:10>	—	—	reserved
<9:6>	R/W	AF*x*	GPIO ‘x’ Alternate Function Select Bits (where x = 3 to 4) A bit-pair in this register determines the corresponding GPIO pin’s functionality as one of the alternate functions that is mapped to it or as a generic GPIO pin. 0b00 = The corresponding GPIO pin (GPIO<x>) is used as a general-purpose I/O. 0b01 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 1. 0b10 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 2. 0b11 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 3.
<5:4>	—	—	reserved
<3:0>	R/W	AFx	GPIO ‘x’ Alternate Function Select Bits (where x = 0 through 1) A bit-pair in this register determines the corresponding GPIO pin’s functionality as one of the alternate functions that is mapped to it or as a generic GPIO pin. 0b00 = The corresponding GPIO pin (GPIO<x>) is used as a general-purpose I/O. 0b01 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 1. 0b10 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 2. 0b11 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 3.

Table 24-26. GAFR0_U Bit Definitions

Physical Address 0x40E0_0058																GAFR0_U												GPIO Controller							
User Settings																																			
Bit	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			
	AF31		AF30		AF29		AF28		AF27		AF26		AF25		AF24		AF23		AF22		AF21		AF20		AF19		AF18		AF17		AF16				
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			

Bits	Access	Name	Description
<31:0>	R/W	AFx	GPIO ‘x’ Alternate Function Select Bits (where x = 16 through 31) A bit-pair in this register determines the corresponding GPIO pin’s functionality as one of the alternate functions that is mapped to it or as a generic GPIO pin. 0b00 = The corresponding GPIO pin (GPIO<x>) is used as a general-purpose I/O. 0b01 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 1. 0b10 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 2. 0b11 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 3.

Table 24-27. GAFR1_L Bit Definitions

Physical Address 0x40E0_005C																GAFR1_L												GPIO Controller							
User Settings																																			
Bit	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			
	AF47		AF46		AF45		AF44		AF43		AF42		AF41		AF40		AF39		AF38		AF37		AF36		AF35		AF34		AF33		AF32				
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			

Bits	Access	Name	Description
<31:0>	R/W	AFx	GPIO ‘x’ Alternate Function Select Bits (where x = 32 through 47) A bit-pair in this register determines the corresponding GPIO pin’s functionality as one of the alternate functions that is mapped to it or as a generic GPIO pin. 0b00 = The corresponding GPIO pin (GPIO<x>) is used as a general-purpose I/O. 0b01 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 1. 0b10 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 2. 0b11 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 3.

Table 24-28. GAFR1_U Bit Definitions

Physical Address 0x40E0_0060																GAFR1_U												GPIO Controller			
User Settings																															
Bit	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
	AF63		AF62		AF61		AF60		AF59		AF58		AF57		AF56		AF55		FA54		AF53		AF52		AF51		AF50		AF49		AF48	
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

Bits	Access	Name	Description
<31:0>	R/W	AFx	GPIO ‘x’ Alternate Function Select Bits (where x = 48 through 63) A bit-pair in this register determines the corresponding GPIO pin’s functionality as one of the alternate functions that is mapped to it or as a generic GPIO pin. 0b00 = The corresponding GPIO pin (GPIO<x>) is used as a general-purpose I/O. 0b01 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 1. 0b10 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 2. 0b11 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 3.

Table 24-29. GAFR2_L Bit Definitions

Physical Address 0x40E0_0064																GAFR2_L												GPIO Controller			
User Settings																															
Bit	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
	AF79		AF78		AF77		AF76		AF75		AF74		AF73		AF72		AF71		AF70		AF69		AF68		AF67		AF66		AF65		AF64	
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

Bits	Access	Name	Description
<31:0>	R/W	AFx	GPIO ‘x’ Alternate Function Select Bits (where x = 64 through 79) A bit-pair in this register determines the corresponding GPIO pin’s functionality as one of the alternate functions that is mapped to it or as a generic GPIO pin. 0b00 = The corresponding GPIO pin (GPIO<x>) is used as a general-purpose I/O. 0b01 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 1. 0b10 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 2. 0b11 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 3.

Table 24-30. GAFR2_U Bit Definitions

Physical Address 0x40E0_0068	GAFR2_U	GPIO Controller

User Settings																															
Bit	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
	AF95		AF94		AF93		AF92		AF91		AF90		AF89		AF88		AF87		AF86		AF85		AF84		AF83		AF82		AF81		AF80	
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

Bits	Access	Name	Description
<31:0>	R/W	AFx	GPIO ‘x’ Alternate Function Select Bits (where x = 80 through 95) A bit-pair in this register determines the corresponding GPIO pin’s functionality as one of the alternate functions that is mapped to it or as a generic GPIO pin. 0b00 = The corresponding GPIO pin (GPIO<x>) is used as a general-purpose I/O. 0b01 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 1. 0b10 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 2. 0b11 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 3.

Table 24-31. GAFR3_L Bit Definitions

Physical Address 0x40E0_006C	GAFR3_L	GPIO Controller

User Settings																															
Bit	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
	AF111		AF110		AF109		AF108		AF107		AF106		AF105		AF104		AF103		AF102		AF101		AF100		AF99		AF98		AF97		AF96	
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

Bits	Access	Name	Description
<31:0>	R/W	AFx	GPIO ‘x’ Alternate Function Select Bits (where x = 96 through 111) A bit-pair in this register determines the corresponding GPIO pin’s functionality as one of the alternate functions that is mapped to it or as a generic GPIO pin. 0b00 = The corresponding GPIO pin (GPIO<x>) is used as a general-purpose I/O. 0b01 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 1. 0b10 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 2. 0b11 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 3.

Table 24-32. GAFR3_U Bit Definitions

Physical Address 0x40E0_0070																GAFR3_U												GPIO Controller			
User Settings																															
User Settings																															
Bit	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														AF120		AF119		AF118		AF117		AF116		AF115		AF114		AF113		AF112	
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

Bits	Access	Name	Description
<31:18>	—	—	reserved
<17:0>	R/W	AFx	GPIO ‘x’ Alternate Function Select Bits (where x = 112 through 120) A bit-pair in this register determines the corresponding GPIO pin’s functionality as one of the alternate functions that is mapped to it or as a generic GPIO pin. 0b00 = The corresponding GPIO pin (GPIO<x>) is used as a general-purpose I/O. 0b01 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 1. 0b10 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 2. 0b11 = The corresponding GPIO pin (GPIO<x>) is used for its alternate function 3.

24.5.5 GPIO Pin-Level Registers (GPLRx)

Check the state of each of the GPIO pins by reading the GPIO Pin Level register (GPLR). Each bit in the GPLR corresponds to one pin in the GPIO. GPLR0<31:0> correspond to GPIO<31:0>, GPLR1<31:0> correspond to GPIO<63:32>, GPLR2<31:0> correspond to GPIO<95:64>, and GPLR3<24:0> correspond to GPIO<120:96>. Use the GPLRx read-only registers to determine the current value of a particular pin (regardless of the programmed pin direction). For reserved bits (GPLR3<31:25>), reads return zero.

Table 24-33 through Table 24-36 show the bitmaps of the GPIO Pin Level registers.

These are read-only registers. Ignore reads from reserved bits.

Table 24-33. GPLR0 Bit Definitions

Physical Address 0x40E0_0000 — GPLR0 — GPIO Controller

	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
User Settings																																
Bit	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
	PL31	PL30	PL29	PL28	PL27	PL26	PL25	PL24	PL23	PL22	PL21	PL20	PL19	PL18	PL17	PL16	PL15	PL14	PL13	PL12	PL11	PL10	PL9	reserved				PL4	PL3	reserved	PL1	PL0
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

Bits	Access	Name	Description
<31:9>	R	PL*x*	GPIO Pin Level ‘x’ (where x = 9 to 31) This read-only field indicates the current value of each GPIO. 0 = Pin state is low. 1 = Pin state is high.
<8:5>	—	—	reserved
<4:3>	R	PL*x*	GPIO Pin Level ‘x’ (where x = 3 to 4) This read-only field indicates the current value of each GPIO. 0 = Pin state is low. 0 = Pin state is high.
<2>	—	—	reserved
<1:0>	R	PL*x*	GPIO Pin Level ‘x’ (where x = 0 to 1) This read-only field indicates the current value of each GPIO. 0 = Pin state is low. 1 = Pin state is high.

Table 24-34. GPLR1 Bit Definitions

Physical Address 0x40E0_0004 — GPLR1 — GPIO Controller

	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
User Settings																																
Bit	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
	PL63	PL62	PL61	PL60	PL59	PL58	PL57	PL56	PL55	PL54	PL53	PL52	PL51	PL50	PL49	PL48	PL47	PL46	PL45	PL44	PL43	PL42	PL41	PL40	PL39	PL38	PL37	PL36	PL35	PL34	PL33	PL32
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

Bits	Access	Name	Description
<31:0>	R	PL*x*	GPIO Pin Level ‘x’ (where x = 32 to 63) This read-only field indicates the current value of each GPIO. 0 = Pin state is low. 1 = Pin state is high.

Table 24-35. GPLR2 Bit Definitions

Physical Address 0x40E0_0008	GPLR2	GPIO Controller

	Bit	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
User Settings																																	
		PL95	PL94	PL93	PL92	PL91	PL90	PL89	PL88	PL87	PL86	PL85	PL84	PL83	PL82	PL81	PL80	PL79	PL78	PL77	PL76	PL75	PL74	PL73	PL72	PL71	PL70	PL69	PL68	PL67	PL66	PL65	PL64
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

Bits	Access	Name	Description
<31:0>	R	PL*x*	GPIO Pin Level ‘x’ (where x = 64 to 95) This read-only field indicates the current value of each GPIO. 0 = Pin state is low. 1 = Pin state is high.

Table 24-36. GPLR3 Bit Definitions

Physical Address 0x40E0_0100	GPLR3	GPIO Controller

	Bit	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
User Settings																																	
User Settings																																	
		reserved							PL120	PL119	PL118	PL117	PL116	PL115	PL114	PL113	PL112	PL111	PL110	PL109	PL108	PL107	PL106	PL105	PL104	PL103	PL102	PL101	PL100	PL99	PL98	PL97	PL96
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

Bits	Access	Name	Description
<31:25>	—	—	reserved
<24:0>	R	PL*x*	GPIO Pin Level ‘x’ (where x = 96 to 120) This read-only field indicates the current value of each GPIO. 0 = Pin state is low. 1 = Pin state is high.

24.5.6 GPIO Edge Detect Status Register (GEDR)

The GPIO Edge Detect Status registers (GEDR0/1/2/3) contain a total of 121 status bits that correspond to the 121 GPIO pins. When an edge-detect occurs on a pin that matches the type of edge programmed in the GRER and/or GFER registers, the corresponding status bit is set in GEDR. Once a GEDR bit is set by an edge event, the bit remains set until it is cleared by writing a one to the status bit. Writing a zero to a GEDR status bit has no effect.

Each edge-detect that sets the corresponding GEDR status bit for GPIO<120:0> can trigger an interrupt request. GPIO<120:2> together form a group that can cause one interrupt request to be triggered when any one of GEDR<120:2> is set. GPIO<0> and GPIO<1> cause independent first-level interrupts. Refer to Chapter 25, “Interrupt Controller” for a description of the programming of GPIO interrupts.

Table 24-37 through Table 24-40 show the bit descriptions of the GPIO Edge Detect Status registers.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 24-37. GEDR0 Bit Definitions

Physical Address 0x40E0_0048																GEDR0												GPIO Controller										
User Settings																																				
Bit	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
	ED31	ED30	ED29	ED28	ED27	ED26	ED25	ED24	ED23	ED22	ED21	ED20	ED19	ED18	ED17	ED16	ED15	ED14	ED13	ED12	ED11	ED10	ED9	reserved				ED4	ED3	reserved	ED1	ED0
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

Bits	Access	Name	Description
<31:9>	R/W	ED*x*	GPIO Pin ‘x’ Edge Detect Status (where x = 9 to 31) READ 0 = No edge detect has occurred on pin as specified in GRER and/or GFER. 1 = Edge detect has occurred on pin as specified in GRER and/or GFER. WRITE 0 = No effect. 1 = Clear edge detect status field.
<8:5>	—	—	reserved
<4:3>	R/W	ED*x*	GPIO Pin ‘x’ Edge Detect Status (where x = 3 to 4) READ 0 = No edge detect has occurred on pin as specified in GRER and/or GFER. 1 = Edge detect has occurred on pin as specified in GRER and/or GFER. WRITE 0 = No effect. 1 = Clear edge detect status field.
<2>	—	—	reserved
<1:0>	R/W	ED*x*	GPIO Pin ‘x’ Edge Detect Status (where x= 0 through 1) READ 0 = No edge detect has occurred on pin as specified in GRER and/or GFER. 1 = Edge detect has occurred on pin as specified in GRER and/or GFER. WRITE 0 = No effect. 1 = Clear edge detect status field.

Table 24-38. GEDR1 Bit Definitions

Physical Address 0x40E0_004C — GEDR1 — GPIO Controller

	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
User Settings																																
Bit	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
	ED63	ED62	ED61	ED60	ED59	ED58	ED57	ED56	ED55	ED54	ED53	ED52	ED51	ED50	ED49	ED48	ED47	ED46	ED45	ED44	ED43	ED42	ED41	ED40	ED39	ED38	ED37	ED36	ED35	ED34	ED33	ED32
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

Bits	Access	Name	Description
<31:0>	R/W	ED*x*	GPIO Pin ‘x’ Edge Detect Status (where x = 32 through 63) READ 0 = No edge detect has occurred on pin as specified in GRER and/or GFER. 1 = Edge detect has occurred on pin as specified in GRER and/or GFER. WRITE 0 = No effect. 1 = Clear edge detect status field.

Table 24-39. GEDR2 Bit Definitions

Physical Address 0x40E0_0050 — GEDR2 — GPIO Controller

	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
User Settings																																
Bit	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
	ED95	ED94	ED93	ED92	ED91	ED90	ED89	ED88	ED87	ED86	ED85	ED84	ED83	ED82	ED81	ED80	ED79	ED78	ED77	ED76	ED75	ED74	ED73	ED72	ED71	ED70	ED69	ED68	ED67	ED66	ED65	ED64
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

Bits	Access	Name	Description
<31:0>	R/W	ED*x*	GPIO Pin ‘x’ Edge Detect Status. (where x = 64 through 95) READ 0 = No edge detect has occurred on pin as specified in GRER and/or GFER. 1 = Edge detect has occurred on pin as specified in GRER and/or GFER. WRITE 0 = No effect. 1 = Clear edge detect status field.

Table 24-40. GEDR3 Bit Definitions

Physical Address 0x40E0_0148								GEDR3																GPIO Controller							
User Settings																															
User Settings																															
Bit	**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								**ED120**	**ED119**	**ED118**	**ED117**	**ED116**	**ED115**	**ED114**	**ED113**	**ED112**	**ED111**	**ED110**	**ED109**	**ED108**	**ED107**	**ED106**	**ED105**	**ED104**	**ED103**	**ED102**	**ED101**	**ED100**	**ED99**	**ED98**	**ED97**	**ED96**
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**

Bits	Access	Name	Description
<31:25>	—	—	reserved
<24:0>	R/W	ED*x*	GPIO Pin ‘x’ Edge Detect Status (where x = 96 through 120) READ 0 = No edge detect has occurred on pin as specified in GRER and/or GFER. 1 = Edge detect has occurred on pin as specified in GRER and/or GFER. WRITE 0 = No effect. 1 = Clear edge detect status field.

24.6 Register Summary

Table 24-41 describes the location of the GPIO registers.

Table 24-41. GPIO Controller Register Summary (Sheet 1 of 2)

Physical Address	Name	Description	Page
0x40E0_ 0000	GPLR0	GPIO Pin-Level register GPIO<31:0>	24-28
0x40E0_ 0004	GPLR1	GPIO Pin-Level register GPIO<63:32>	24-28
0x40E0_ 0008	GPLR2	GPIO Pin-Level register GPIO<95:64>	24-28
0x40E0_ 000C	GPDR0	GPIO Pin Direction register GPIO<31:0>	24-11
0x40E0_ 0010	GPDR1	GPIO Pin Direction register GPIO<63:32>	24-11
0x40E0_ 0014	GPDR2	GPIO Pin Direction register GPIO<95:64>	24-11
0x40E0_ 0018	GPSR0	GPIO Pin Output Set register GPIO<31:0>	24-14
0x40E0_ 001C	GPSR1	GPIO Pin Output Set register GPIO<63:32>	24-14
0x40E0_ 0020	GPSR2	GPIO Pin Output Set register GPIO<95:64>	24-14
0x40E0_ 0024	GPCR0	GPIO Pin Output Clear register GPIO<31:0>	24-14
0x40E0_ 0028	GPCR1	GPIO Pin Output Clear register GPIO <63:32>	24-14
0x40E0_ 002C	GPCR2	GPIO pin Output Clear register GPIO <95:64>	24-14
0x40E0_ 0030	GRER0	GPIO Rising-Edge Detect Enable register GPIO<31:0>	24-18

Table 24-41. GPIO Controller Register Summary (Sheet 2 of 2)

Physical Address	Name	Description	Page
0x40E0_ 0034	GRER1	GPIO Rising-Edge Detect Enable register GPIO<63:32>	24-18
0x40E0_ 0038	GRER2	GPIO Rising-Edge Detect Enable register GPIO<95:64>	24-18
0x40E0_ 003C	GFER0	GPIO Falling-Edge Detect Enable register GPIO<31:0>	24-18
0x40E0_ 0040	GFER1	GPIO Falling-Edge Detect Enable register GPIO<63:32>	24-18
0x40E0_ 0044	GFER2	GPIO Falling-Edge Detect Enable register GPIO<95:64>	24-18
0x40E0_ 0048	GEDR0	GPIO Edge Detect Status register GPIO<31:0>	24-30
0x40E0_ 004C	GEDR1	GPIO Edge Detect Status register GPIO<63:32>	24-30
0x40E0_ 0050	GEDR2	GPIO Edge Detect Status register GPIO<95:64>	24-30
0x40E0_ 0054	GAFR0_L	GPIO Alternate Function register GPIO<15:0>	24-23
0x40E0_ 0058	GAFR0_U	GPIO Alternate Function register GPIO<31:16>	24-23
0x40E0_ 005C	GAFR1_L	GPIO Alternate Function register GPIO<47:32>	24-23
0x40E0_ 0060	GAFR1_U	GPIO Alternate Function register GPIO<63:48>	24-23
0x40E0_ 0064	GAFR2_L	GPIO Alternate Function register GPIO<79:64>	24-23
0x40E0_ 0068	GAFR2_U	GPIO Alternate Function register GPIO <95:80>	24-23
0x40E0_ 006C	GAFR3_L	GPIO Alternate Function register GPIO<111:96>	24-23
0x40E0_ 0070	GAFR3_U	GPIO Alternate Function register GPIO<120:112>	24-23
0x40E0_ 0074– 0x40E0_ 00FC	—	reserved	
0x40E0_ 0100	GPLR3	GPIO Pin-Level register GPIO<120:96>	24-28
0x40E0_ 0104– 0x40E0_ 0108	-—	reserved	
0x40E0_ 010C	GPDR3	GPIO Pin Direction register GPIO<120:96>	24-11
0x40E0_ 0110– 0x40E0_ 0114	—	reserved	
0x40E0_ 0118	GPSR3	GPIO Pin Output Set register GPIO<120:96>	24-14
0x40E0_ 011C– 0x40E0_ 0120	—	reserved	
0x40E0_ 0124	GPCR3	GPIO Pin Output Clear register GPIO<120:96>	24-14
0x40E0_ 0128– 0x40E0_ 012C	—	reserved	
0x40E0_ 0130	GRER3	GPIO Rising-Edge Detect Enable register GPIO<120:96>	24-18
0x40E0_ 0134– 0x40E0_ 0138	—	reserved	
0x40E0_ 013C	GFER3	GPIO Falling-Edge Detect Enable register GPIO<120:96>	24-18
0x40E0_ 0140– 0x40E0_ 0144	—	reserved	
0x40E0_ 0148	GEDR3	GPIO Edge Detect Status register GPIO<120:96>	24-18
0x40E0_ 014C– 0x40EF_FFFC	-—	reserved	

Interrupt Controller 25

This chapter describes the interrupt controller included in the PXA27x processor, explains its modes of operation, and defines its registers. The interrupt controller controls the interrupt sources available to the processor and contains the location of the interrupt source to allow software to determine the first-level source of all interrupts. It also determines whether the interrupts cause an IRQ or an FIQ to occur and masks the interrupts.

25.1 Overview

The interrupt controller provides a two-level hierarchy of interrupts. It can receive interrupts from peripherals or PXA27x processor devices. *Processor devices* are the *primary* sources of interrupts. The internal events that occur in a peripheral and cause an interrupt are called *secondary* sources of interrupts.

Multiple secondary sources are usually mapped to a single primary source. For example, the DMA controller is a primary source of interrupts to the interrupt controller, with 32 possible secondary sources.

Each interrupt source is set to generate either an IRQ or an FIQ. The setting that determines whether an IRQ or an FIQ generated is called the *level* of the interrupt. The interrupt controller can be programmed to individually mask interrupts from the different sources. If an interrupt is masked, the controller does not cause the interrupt. The software can read registers in the interrupt controller, which identifies all the active IRQ and FIQ interrupts.

The controller assigns a unique priority to each primary source. It uses the assigned priority values to determine the highest priority peripheral when more than one interrupt is pending. Software can determine the peripheral ID with the highest active priority by reading a register in the controller.

The registers can be accessed by two methods: as coprocessor registers or as memory-mapped I/O registers. Coprocessor-register mode has less access latency than memory-mapped I/O-register mode access.

25.2 Features

The interrupt controller unit has the following features:

- Peripheral interrupt sources can be mapped to FIQ or IRQ
- Each interrupt source can be independently enabled
- Priority mechanism to indicate highest priority interrupt
- Accessible from the coprocessor interface
- Accessible as a memory-mapped peripheral for backward compatibility

25.3 Signal Descriptions

No external I/O signals are associated with the interrupt controller.

25.4 Operation

The Interrupt Controller Pending register (ICPR) (see Section 25.5.1) has a bit for each of the peripherals (primary interrupt sources). Each active interrupt sets the corresponding bit. The Interrupt Controller IRQ Pending register (ICIP) (see Section 25.5.2) and the Interrupt Controller FIQ Pending register (ICFP) (see Section 25.5.3) identify the active, unmasked pending interrupt sources that are causing IRQ- and FIQ-level interrupts, respectively. Interrupts are set at their source and masked or unmasked in the interrupt controller. The programmable Interrupt Controller Mask register (ICMR) (see Section 25.5.4) chooses which interrupt source is masked. Masked interrupt sources do not cause an IRQ or FIQ interrupt to the core and only update the ICPR. The Interrupt Control Level register (ICLR) (see Section 25.5.5) chooses whether an interrupt causes an IRQ or an FIQ.

Because more than one unmasked interrupt can be active at the same time, the processor must select the one with the highest priority. Software can determine that through the ICIP and ICFP or by reading the Interrupt Controller Highest Priority (ICHP) (see Section 25.5.8) register at the controller that contains the peripheral ID with the highest priority value among the active unmasked interrupts.

The second level of the interrupt structure is represented by registers contained in the source device (the device that generated the first-level interrupt bit). Second-level interrupt status provides additional information about the interrupt and is used inside the interrupt-service routine. After it reads the first-level registers, software reads the registers in the device to determine the function that is causing the interrupt. In general, multiple second-level interrupts are ORed to produce a first-level interrupt bit. Interrupts are enabled inside the source device.

Interrupt Priority registers (IPRs) (see Section 25.5.7) specify the mapping between the different priority levels and the peripheral IDs. The interrupt controller uses these set values to prioritize the active unmasked interrupts and to update the ICHP register. The core can read the highest priority peripheral ID for both IRQ- and FIQ-level from the ICHP.

Because the highest priority register is automatically updated with the peripheral ID, the software is not required to examine each of the pending interrupts to determine which peripheral ID has the highest priority. ICHP is updated with both the highest priority IRQ and FIQ peripherals.

Most interrupt controller registers can be accessed using coprocessor-register-access mode or memory-mapped register-access mode. The software can use either access method. Anomalies that can occur due to the varying access latencies in each mode must be considered. For example, a coprocessor-mapped read of a register that immediately follows a memory-mapped register write to the register may not yield the correct value. The memory-mapped register I/Os must be read immediately after the write to ensure that the memory-mapped write has been performed properly.

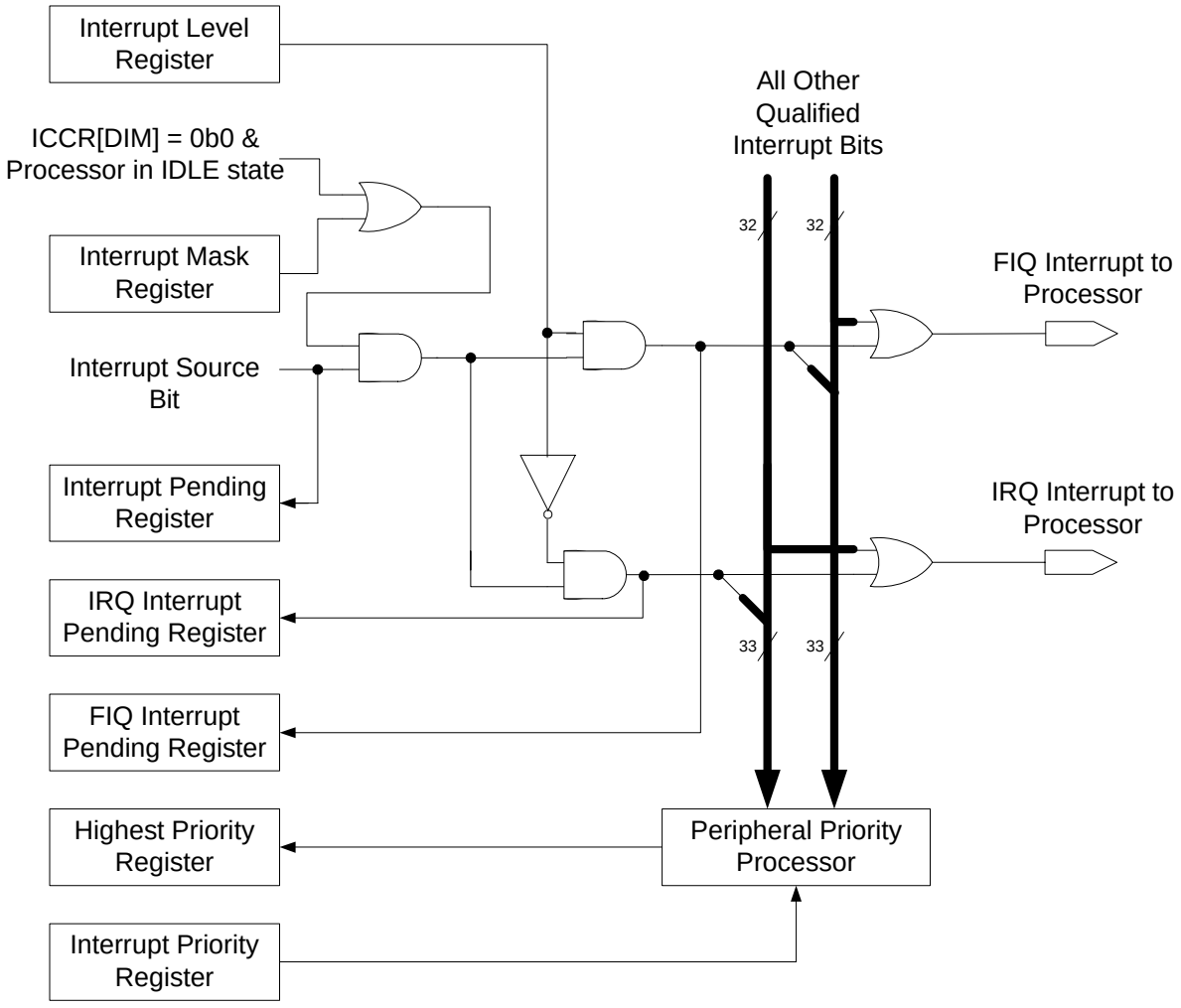
For coprocessor-register-access mode, coprocessor space number six is used. Interrupt controller registers can be accessed only in supervisory mode. If any software process attempts to access the registers in the coprocessor mode, it receives an undefined access error. A software process that attempts to access the registers in memory-mapped register-access mode triggers a data-abort error.

Software sets the enabling and access options for the coprocessor. Detailed information on setting access and enable options is located in the *Intel XScale® Microarchitecture Programmer's Reference Manual.*

When the processor is in idle state, the occurrence of any enabled interrupt causes the processor to resume operation. When ICCR disable-idle mode (DIM) is cleared—its default state—the interrupt controller ignores interrupt masks when the processor is in idle.

Figure 25-1 is a block diagram of the interrupt controller's operation. All of the registers depicted in Figure 25-1 are described in detail in this chapter.

Figure 25-1. Interrupt Controller Block Diagram



25.4.1 Accessing Interrupt Controller Registers

Most of the interrupt controller registers can be accessed through coprocessor registers. Accessing the interrupt controller registers through the coprocessor registers significantly reduces access times. The coprocessor must be accessed in the supervisory mode only. Attempts to access the interrupt coprocessor in user mode result in an undefined instruction exception.

Table 25-1 shows the interrupt controller registers and the coprocessor register numbers that correspond to them. The registers are mapped to the register space of Coprocessor 6.

Table 25-1. Interrupt Controller Register Mapping When Mapped to Coprocessor Space

Coprocessor 6 Register Number	Name	Access Mode	Description
CR0	ICIP	R	Interrupt Controller IRQ Pending register
CR1	ICMR	R/W	Interrupt Controller Mask register
CR2	ICLR	R/W	Interrupt Controller Level register
CR3	ICFP	R	Interrupt Controller FIQ Pending register
CR4	ICPR	R	Interrupt Controller Pending register
CR5	ICHP	R	Interrupt Controller Highest Priority register
CR6	ICIP2	R	Interrupt Controller IRQ Pending register 2
CR7	ICMR2	R/W	Interrupt Controller Mask register 2
CR8	ICLR2	R/W	Interrupt Controller Level register 2
CR9	ICFP2	R	Interrupt Controller FIQ Pending register 2
CR10	ICPR2	R	Interrupt Controller Pending register 2
Not Mapped to coprocessor	ICCR	R/W	Interrupt Controller Control register
Not Mapped to coprocessor	IPR0–IPR39	R/W	Interrupt Priority registers for Interrupts 0 to 39

25.4.2 Enabling and Accessing the Coprocessor

To access the interrupt controller as Coprocessor 6, the corresponding coprocessor must be enabled in the core CPAR register. See the *Intel XScale® Core Developer's Manual* for more information about enabling the coprocessors.

The example code sequences that follow assume that Coprocessor 6 is enabled:

To read from the coprocessor register, use an MRC instruction:

```
MRC P6, 0, Rd, CRn, C0, 0 ; Rd:Arm register and CRn:Coprocessor register
```

Note: All registers except IPR and ICCR are readable in coprocessor-access mode.

To write the interrupt controller register, use an MCR instruction:

```
LDR Rd, =0xdesired_val; Loading core register with desired value
MCR P6, 0, Rd, CRn, C0, 0 write the ARM* register to CP6 register CRn
```

Note: Only ICMR and ICLR can be written to in coprocessor-access mode.

25.4.3 Bit Positions and Peripheral IDs

Table 25-2 contains a list of sources for the interrupts. Each source can be associated with a number of second-level sources. Refer to the appropriate chapters for more detailed information on the second-level interrupt sources in each device. Each interrupt source has a peripheral ID. For the sake of simplicity, the peripheral ID is assigned according to bit position. The IPRs are set using these values as the peripheral ID.

Table 25-2. Bit Positions for Primary Interrupt Sources (Sheet 1 of 2)

Bit Position	Source Module	Bit Field Description	Peripheral ID
IP[39]	NA	reserved	39
IP[38]	NA	reserved	38
IP[37]	NA	reserved	37
IP[36]	NA	reserved	36
IP[35]	NA	reserved	35
IP[34]	NA	reserved	34
IP[33]	Quick capture interface	Quick capture interface interrupt	33
IP[32]	Trusted Platform Module	Trusted Platform Module interrupt	32
IP[31]	Real-time clock	RTC equals Alarm register	31
IP[30]		One Hz clock TIC occurred	30
IP[29]	Operating system timers	OS timer equals Match register 3	29
IP[28]		OS timer equals Match register 2	28
IP[27]		OS timer equals Match register 1	27
IP[26]		OS timer equals Match register 0	26
IP[25]	DMA controller	DMA Channel service request	25
IP[24]	Synchronous serial port 1	SSP_1 service request	24
IP[23]	Flash card interface/MMC	Flash Card status/Error detection	23
IP[22]	FFUART	Transmit or receive error in FFUART	22
IP[21]	BTUART	Transmit or receive error in BTUART	21
IP[20]	STUART	Transmit or receive error in STUART	20
IP[19]	Infrared communications port	Transmit or receive error in infrared communications port	19
IP[18]	I^2C	I^2C service request	18
IP[17]	LCD controller	LCD controller service request	17
IP[16]	Synchronous serial port 2	SSP_2 service request	16
IP[15]	USIM interface	Smart card interface status/error	15
IP[14]	AC '97	AC '97 interrupt	14
IP[13]	I^2S	I^2S interrupt	13
IP[12]	PMU	PMU (performance monitor) interrupt	12
IP[11]	USB client	USB client interrupt	11

Table 25-2. Bit Positions for Primary Interrupt Sources (Sheet 2 of 2)

Bit Position	Source Module	Bit Field Description	Peripheral ID
IP[10]	GPIO	GPIO_x “OR” of the GPIO edge detect (except 0 and 1)	10
IP[9]		GPIO<1> detects an edge	9
IP[8]		GPIO<0> detects an edge	8
IP[7]	Operating system timers	OS timer matches registers 4–11	7
IP[6]	Power I^2C	Power I^2C interrupt	6
IP[5]	Memory Stick	Memory stick interrupt	5
IP[4]	Keypad controller	Keypad controller interrupt	4
IP[3]	USB host controller	USB host interrupt 1 (OHCI)	3
IP[2]		USB host interrupt 2	2
IP[1]	Mobile scalable link	MSL Interface interrupt	1
IP[0]	Synchronous serial port 3	SSP_3 service request	0

25.5 Register Descriptions

25.5.1 Interrupt Controller Pending Registers (ICPR and ICPR2)

ICPR (Table 25-3) and ICPR2 (Table 25-4) are read-only registers that show all active interrupts in the system. The contents are not affected by the state of the mask registers (ICMR and ICMR2). Both the IRQ-level and FIQ-level active interrupts are read as set in this register. The register is cleared during reset.

Table 25-3. ICPR Bit Definitions (Sheet 1 of 3)

Physical Address: 0x40D0_0010 **ICPR** **Interrupt Controller**
Coprocessor Register: CR4

	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
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
31	R	RTC_AL	Real-Time Clock Alarm: 0 = RTC equals alarm register interrupt has not occurred. 1 = RTC equals alarm register interrupt has occurred.
30	R	RTC_HZ	One Hz Clock: 0 = One Hz clock TIC has not occurred. 1 = One Hz clock TIC has occurred.
29	R	OST_3	OS Timer 3: 0 = OS timer does not equal match register 3. 1 = OS timer equals match register 3.
28	R	OST_2	OS Timer 2: 0 = OS timer does not equal match register 2. 1 = OS timer equals match register 2.
27	R	OST_1	OS Timer 1: 0 = OS timer does not equal match register 1. 1 = OS timer equals match register 1.
26	R	OST_0	OS Timer 0: 0 = OS timer does not equal match register 0. 1 = OS timer equals match register 0.
25	R	DMAC	DMA Controller: 0 = DMA Channel service request has not occurred. 1 = DMA Channel service request has occurred.
24	R	SSP1	SSP 1: 0 = SSP 1 has not requested service. 1 = SSP 1 has requested service.
23	R	MMC	MultiMediaCard: 0 = Flash card status has not changed/error detection has not occurred. 1 = Flash card status change/error detection has occurred.
22	R	FFUART	FFUART: 0 = A transmit or receive error in FFUART has not occurred. 1 = A transmit or receive error in FFUART has occurred.
21	R	BTUART	BTUART: 0 = A transmit or receive error in BTUART has not occurred. 1 = A transmit or receive error in BTUART has occurred.
20	R	STUART	STUART: 0 = A transmit or receive error in STUART has not occurred. 1 = A transmit or receive error in STUART has occurred.

Table 25-3. ICPR Bit Definitions (Sheet 2 of 3)

Physical Address: 0x40D0_0010 — **ICPR** — **Interrupt Controller**
Coprocessor Register: CR4

	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
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
19	R	ICP	Infrared Communications Port: 0 = A transmit or receive error in infrared port has not occurred. 1 = A transmit or receive error in infrared port has occurred.
18	R	I2C	I2C: 0 = I^2C service request has not occurred. 1 = I^2C service request has occurred.
17	R	LCD	LCD: 0 = LCD controller has not requested service. 1 = LCD controller has requested service.
16	R	SSP2	SSP2: 0 = SSP 2 has not requested service. 1 = SSP 2 has requested service.
15	R	USIM	USIM: 0 = Smart card interface status changes/error has occurred. 1 = Smart card interface status changes/error has occurred.
14	R	AC97	AC97: 0 = AC '97 interrupt has not occurred. 1 = AC '97 interrupt has occurred.
13	R	I2S	I2S: 0 = ^2S interrupt has not occurred. 1 = I^2S interrupt has occurred.
12	R	PMU	PMU: 0 = PMU (performance monitor) interrupt has not occurred. 1 = PMU (performance monitor) interrupt has occurred.
11	R	USBC	USB Client: 0 = USB client interrupt has not occurred. 1 = USB client interrupt has occurred.
10	R	GPIO_x	GPIO_x 0 = A GPIO_x edge (other than GPIO_0 or GPIO_1) has not been detected. 1 = A GPIO_x edge (other than GPIO_0 or GPIO_1) has been detected.
9	R	GPIO_1	GPIO_1: 0 = GPIO<1> edge has not been detected. 1 = GPIO<1> edge has been detected.
8	R	GPIO_0	GPIO_0: 0 = GPIO<0> edge has not been detected. 1 = GPIO<0> edge has been detected.

Table 25-3. ICPR Bit Definitions (Sheet 3 of 3)

Physical Address: 0x40D0_0010
Coprocessor Register: CR4

ICPR

Interrupt Controller

	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
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
7	R	OST_4_11	OS Timers 4–11: 0 = OS timer match 4-11 has not occurred. 1 = OS timer match 4-11 has occurred.
6	R	PWR_I2C	Power I^2C: 0 = I^2C power unit interrupt has not occurred. 1 = I^2C power unit interrupt has occurred.
5	R	MEM_STK	Memory Stick: 0 = Memory stick host controller request has not occurred. 1 = Memory stick host controller request has occurred.
4	R	KEYPAD	Keypad: 0 = Keypad controller interrupt has not occurred. 1 = Keypad controller interrupt has occurred.
3	R	USBH1	USB Host 1: 0 = USB host interrupt 1 (OHCI) has not occurred. 1 = USB host interrupt 1 (OHCI) has occurred.
2	R	USBH2	USB Host 2: 0 = USB host interrupt 2 has not occurred. 1 = USB host interrupt 2 has occurred.
1	R	MSL	MSL: 0 = MSL interface interrupt 1 has not occurred. 1 = MSL interface interrupt 1 has occurred.
0	R	SSP3	SSP 3: 0 = SSP3 has not requested service. 1 = SSP3 has requested service.

Table 25-4. ICPR2 Bit Definitions

Physical Address 0x40D0_00AC Coprocessor Register Number CR10	ICPR2	Interrupt Controller

Bit	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																														CIF	TPM
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31-2	—	—	reserved
1	R	CIF	Quick Capture Interface: 0 = Quick capture Interface interrupt has not occurred 1 = Quick capture Interface interrupt has occurred = 0 otherwise
0	R	TPM	Trusted Platform Module: 0 = Trusted Platform Module interrupt has not occurred 1 = Trusted Platform Module interrupt has occurred

Several units have more than one source per interrupt signal. When one of these units signals an interrupt, the interrupt-service routine identifies the interrupt through the registers (ICPR and ICPR2) or by reading the ICHP that contains the peripheral ID with the interrupt that has highest active, unmasked priority. The handler then reads the interrupting unit's status register to identify which source in the unit signaled the interrupt. For all interrupts that have only one corresponding source, the interrupt-handler routine must use only the interrupt controller registers to identify the exact cause of the interrupt. After the interrupt from a source is served, the corresponding bit is cleared.

25.5.2 Interrupt Controller IRQ Pending Registers (ICIP and ICIP2)

ICIP and ICIP2 have one bit per interrupt source. A bit is set when the corresponding peripheral has a pending unmasked IRQ interrupt that is waiting to be served. In general, software reads status registers in the interrupting device for detailed information to determine how to service the interrupt. Interrupt bits in the ICPR and ICPR2 are read-only and represent the logical OR of the status bits for a given interrupt in the source unit. After an interrupt has been serviced, the handler clears the pending interrupt at the source by writing to the required status bit. Refer to the respective peripheral chapters for details.

Clearing the interrupt-status bit at the source automatically clears the corresponding ICIP (or ICIP2) bit. Table 25-5 and Table 25-5 show the bit locations that correspond to the independent Interrupt-Pending Status flags in the ICIP and ICIP2. An interrupting source creates an IRQ if the corresponding bit in the ICLR or ICLR2 is cleared.

Table 25-5. ICIP Bit Definitions (Sheet 1 of 4)

Physical Address: 0x40D0_0000
Coprocessor Register: CR0

ICIP — **Interrupt Controller**

	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
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USB1	USB2	MSL	SSP3
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

Bits	Access	Name	Description
31	R	RTC_AL	Real-Time Clock Alarm 0 = One of the requirements for setting the bit has not been met. 1 = RTC equals Alarm register, interrupt level<31> = 0, and either Mask Bit <31> = 1 or DIM Bit = 0.
30	R	RTC_HZ	One Hz Clock 0 = One of the requirements for setting the bit has not been met. 1 = One Hz clock TIC occurred, interrupt level<30> = 0, and either Mask Bit <30> = 1 or DIM Bit = 0.
29	R	OST_3	OS Timer 3 0 = One of the requirements for setting the bit has not been met. 1 = OS timer equals match register 3, interrupt level<29> = 0, and either Mask Bit <29> = 1 or DIM Bit = 0.
28	R	OST_2	OS Timer 2 0 = One of the requirements for setting the bit has not been met. 1 = OS timer equals match register 2, interrupt level<28> = 0, and either Mask Bit <28> = 1 or DIM Bit = 0.
27	R	OST_1	OS Timer 1 0 = One of the requirements for setting the bit has not been met. 1 = OS timer equals match register 1, interrupt level<27> = 0, and either Mask Bit <27> = 1 or DIM Bit = 0.
26	R	OST_0	OS Timer 0 0 = One of the requirements for setting the bit has not been met. 1 = OS timer equals match register 0, interrupt level<26> = 0, and either Mask Bit <26> = 1 or DIM Bit = 0.
25	R	DMAC	DMA Controller 0 = One of the requirements for setting the bit has not been met. 1 = DMA Channel service request has occurred, interrupt level <23> = 0, and either Mask Bit<23> = 1 or DIM Bit = 0.
24	R	SSP1	SSP 1 0 = One of the requirements for setting the bit has not been met. 1 = SSP 1 service request has occurred, interrupt level <22> = 0, and either Mask Bit<22> = 1 or DIM Bit = 0.
23	R	MMC	MultiMediaCard 0 = One of the requirements for setting the bit has not been met. 1 = Flash card status has changed or any error has been detected, interrupt level<19> = 0, and either Mask Bit<19> = 1 or DIM Bit = 0.

Table 25-5. ICIP Bit Definitions (Sheet 2 of 4)

Physical Address: 0x40D0_0000
Coprocessor Register: CR0

ICIP

Interrupt Controller

	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
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USB1	USB2	MSL	SSP3
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

Bits	Access	Name	Description
22	R	FFUART	FFUART 0 = One of the requirements for setting the bit has not been met. 1 = A transmit or receive error has occurred in FFUART, interrupt level<18> = 0, and either Mask Bit<18> = 1 or DIM Bit = 0.
21	R	BTUART	BTUART 0 = One of the requirements for setting the bit has not been met. 1 = A transmit or receive error has occurred in BTUART, interrupt level<17> = 0, and either Mask Bit<17> = 1or DIM Bit = <0>.
20	R	STUART	STUART 0 = One of the requirements for setting the bit has not been met. 1 = A transmit or receive error has occurred in STUART, interrupt level<16> = 0, and either Mask Bit<16> = 1 or DIM Bit = 0.
19	R	ICP	Infrared Communications Port 0 = One of the requirements for setting the bit has not been met. 1 = A transmit or receive error has occurred in infrared port, interrupt level<15> = 0, and either Mask Bit<15> = 1 or DIM Bit = 0.
18	R	I2C	I^2C 0 = One of the requirements for setting the bit has not been met. 1 = I^2C service request has occurred, interrupt level<14> = 0, and either Mask Bit<14> = 1 or DIM Bit = 0.
17	R	LCD	LCD 0 = One of the requirements for setting the bit has not been met. 1 = LCD controller service request has occurred, interrupt level<13> = 0, and either Mask Bit<13> = 1 or DIM Bit = 0.
16	R	SSP2	SSP 2 0 = One of the requirements for setting the bit has not been met. 1 = SSP 2 service requests has occurred, interrupt level<21> = 0, and either Mask Bit<21> = 1 or DIM Bit = 0.
15	R	USIM	USIM 0 = One of the requirements for setting the bit has not been met. 1 = Smart card interface status/error has occurred, interrupt level<20> = 0, and either Mask Bit<20> = 1 or DIM Bit = 0.
14	R	AC97	AC97 0 = One of the requirements for setting the bit has not been met. 1 = AC '97 interrupt has occurred, interrupt level<12> = 0, and either Mask Bit<12> = 1, or DIM Bit = 0.
13	R	I2S	I^2S 0 = One of the requirements for setting the bit has not been met. 1 = I^2S interrupt has occurred, interrupt level<11> = 0, and either Mask Bit<11> = 1 or DIM Bit = 0.

Table 25-5. ICIP Bit Definitions (Sheet 3 of 4)

Physical Address: 0x40D0_0000
Coprocessor Register: CR0

ICIP

Interrupt Controller

	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
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USB1	USB2	MSL	SSP3
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

Bits	Access	Name	Description
12	R	PMU	Power Management Unit 0 = One of the requirements for setting the bit has not been met. 1 = PMU (Performance Monitor) interrupt has occurred, interrupt level<6> = 0, and either Mask Bit<6> = 1 or DIM Bit = 0.
11	R	USBC	USB Client 0 = One of the requirements for setting the bit has not been met. 1 = USB client interrupt has occurred, interrupt level<5> = 0, and either Mask Bit<5> = 1 or DIM Bit = 0.
10	R	GPIO_x	GPIO_x 0 = One of the requirements for setting the bit has not been met. 1 = GPIO_x (other than GPIO_0 or GPIO_1) edge detect = 1, interrupt level<10> = 0, and either Mask Bit<10> = 1 or DIM Bit = 0.
9	R	GPIO_1	GPIO_1 0 = One of the requirements for setting the bit has not been met. 1 = GPIO<1> has detected an edge, interrupt level<9> = 0, and either Mask Bit<9> = 1 or DIM Bit = 0.
8	R	GPIO_0	GPIO_0 0 = One of the requirements for setting the bit has not been met. 1 = GPIO<0> has detected an edge, interrupt level<8> = 0, and either Mask Bit<8> = 1 or DIM Bit = 0.
7	R	OST_4_11	OS Timer 4-11 0 = One of the requirements for setting the bit has not been met. 1 = OS timer match 4-11 has occurred, interrupt level<7> = 0, and either Mask Bit<7> = 1 or DIM Bit = 0.
6	R	PWR_I2C	Power I^2C 0 = One of the requirements for setting the bit has not been met. 1 = I^2C power unit interrupt has occurred, interrupt level<0> = 0, and either Mask Bit<0> = 1 or DIM Bit = 0.
5	R	MEM_STK	Memory Stick 0 = One of the requirements for setting the bit has not been met. 1 = Memory stick host controller request has occurred, interrupt level<24> = 0, and either Mask Bit<24> = 1 or DIM Bit = 0.
4	R	KEYPAD	Keypad 0 = One of the requirements for setting the bit has not been met. 1 = Keypad controller interrupt has occurred, interrupt level<4> = 0, and either Mask Bit<4> = 1 or DIM Bit = 0.
3	R	USBH1	USB Host 1 0 = One of the requirements for setting the bit has not been met. 1 = USB host interrupt 1 (OHCI) interrupt has occurred, interrupt level<3> = 0, and either Mask Bit<3> = 1 or DIM Bit = 0.

Table 25-5. ICIP Bit Definitions (Sheet 4 of 4)

Physical Address: 0x40D0_0000
Coprocessor Register: CR0

ICIP — **Interrupt Controller**

	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
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USB1	USB2	MSL	SSP3
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

Bits	Access	Name	Description
2	R	USBH2	USB Host 2 0 = One of the requirements for setting the bit has not been met. 1 = USB host interrupt 2 interrupt has occurred, interrupt level<2> = 0, and either Mask Bit<2> = 1 or DIM Bit = 0.
1	R	MSL	MSL 0 = One of the requirements for setting the bit has not been met. 1 = MSL interrupt has occurred, interrupt level<1> = 0, and either Mask Bit<1> = 1 or DIM Bit = 0.
0	R	SSP3	SSP 3 0 = One of the requirements for setting the bit has not been met. 1 = SSP 3 service request has occurred, interrupt level<0> = 0, and either Mask Bit<0> = 1 or DIM Bit = 0.

Table 25-6. ICIP2 Bit Definitions

Physical Address
0x40D0_009C
Coprocessor Register Number
CR6

ICIP2 — **Interrupt Controller**

	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
User Settings																																
Bit	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																														CIF	TPM
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31-2	—	—	reserved
1	R	CIF	Quick Capture Interface 0 = No interrupt notification 1 = Quick capture Interface interrupt occurs and ((Mask Bit(0) = 0b1)OR (DIM Bit = 0b0)) and (interrupt level (33) = 0b0)
0	R	TPM	Trusted Platform Module 0 = One of the requirements for setting the bit has not been met. 1 = Trusted Platform Module service request has occurred, interrupt level <0> = 0, and either Mask Bit<0> = 1 or DIM Bit = 0.

25.5.3 Interrupt Controller FIQ Pending Registers (ICFP and ICFP2)

ICFP and ICFP2 have one bit per interrupt source. A bit is set if the corresponding peripheral has a pending unmasked FIQ interrupt waiting to be served (see Table 25-7 and Table 25-8).

Table 25-7. ICFP Bit Definitions (Sheet 1 of 4)

Physical Address 0x40D0_000C
Coprocessor Register CR3

ICFP

Interrupt Controller

	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
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
31	R	RTC_AL	Real-Time Clock Alarm 0 = No interrupt notification 1 = RTC equals alarm register has occurred, interrupt level<31> = 1, and either Mask Bit<31> = 1 or DIM Bit = 0.
30	R	RTC_HZ	One Hz Clock 0 = No interrupt notification 1 = One Hz clock TIC has occurred, interrupt level<30> = 1, and either Mask Bit<30> = 1 or DIM Bit = 0.
29	R	OST_3	OS Timer 3 0 = No interrupt notification 1 = OS timer equals match register 3, interrupt level<29> = 1, and either Mask Bit<29> = 1 or DIM Bit = 0.
28	R	OST_2	OS Timer 2 0 = No interrupt notification 1 = OS timer equals match register 2, interrupt level<28> = 1, and either Mask Bit<28> = 1 or DIM Bit = 0.
27	R	OST_1	OS Timer 1 0 = No interrupt notification 1 = OS timer equals match register 1, interrupt level<27> = 1, and either Mask Bit<27> = 1 or DIM Bit = 0.
26	R	OST_0	OS Timer 0 0 = No interrupt notification 1 = OS timer equals match register 0, interrupt level<26> = 1, and either Mask Bit<26> = 1 or DIM Bit = 0.
25	R	DMAC	DMA Controller 0 = No interrupt notification 1 = DMA Channel service request has occurred, interrupt level<23> = 1, and either Mask Bit<23> = 1 or DIM Bit = 0.

Table 25-7. ICFP Bit Definitions (Sheet 2 of 4)

Physical Address 0x40D0_000C Coprocessor Register CR3 — **ICFP** — **Interrupt Controller**

Bit	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
User Settings																																
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
24	R	SSP1	SSP 1 0 = No interrupt notification 1 = SSP 1 service request has occurred, interrupt level<22> = 1, and either Mask Bit<22> = 1 or DIM Bit = 0.
23	R	MMC	MultiMediaCard 0 = No interrupt notification 1 = Flash Card status has changed or any error has been detected, interrupt level<19> = 1, and either Mask Bit<19> = 1 or DIM Bit = 0.
22	R	FFUART	FFUART 0 = No interrupt notification 1 = A transmit or receive error has occurred in FFUART, interrupt level<18> = 1, and either Mask Bit<18> = 1 or DIM Bit = 0.
21	R	BTUART	BTUART 0 = No interrupt notification 1 = A transmit or receive error has occurred in BTUART, interrupt level<17> = 1, and either Mask Bit<17> = 1 or DIM Bit = 0.
20	R	STUART	STUART 0 = No interrupt notification 1 = A transmit or receive error in STUART has occurred, interrupt level<16> = 1, and either Mask Bit<16> = 1 or DIM Bit = 0.
19	R	ICP	Infrared Communications Port 0 = No interrupt notification 1 = A transmit or receive error in ICP has occurred, interrupt level<15> = 1, and either Mask Bit<15> = 1 or DIM Bit = 0.
18	R	I2C	I^2C 0 = No interrupt notification 1 = I^2C service request has occurred, interrupt level<14> = 1, and either Mask Bit<14> = 1 or DIM Bit = 0.
17	R	LCD	LCD Controller 0 = One of the requirements for setting the bit has not been met. 1 = LCD controller service request. has occurred, interrupt level<13> = 1, and either Mask Bit<13> = 1 or DIM Bit = 0.
16	R	SSP2	SSP 2 0 = No interrupt notification 1 = SSP 2 service request has occurred, interrupt level<21> = 1, and either Mask Bit<21> = 1 or DIM Bit = 0.

Table 25-7. ICFP Bit Definitions (Sheet 3 of 4)

	Physical Address 0x40D0_000C Coprocessor Register CR3				ICFP																Interrupt Controller											
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
15	R	USIM	USIM 0 = No interrupt notification 1 = Smart card interface status/error has occurred, interrupt level<20> = 1, and either Mask Bit<20> = 1 or DIM Bit = 0.
14	R	AC97	AC97 0 = No interrupt notification 1 = AC '97 interrupt has occurred, interrupt level<12> = 1, and either Mask Bit<12> = 1 or DIM Bit = 0.
13	R	I2S	I^2S 0 = No interrupt notification 1 = I^2S interrupt has occurred, interrupt level<11> = 1, and either Mask Bit<11> = 1 or DIM Bit = 0.
12	R	PMU	Performance Monitor Unit 0 = No interrupt notification 1 = PMU interrupt has occurred, interrupt level<6> = 1, and either Mask Bit<6> = 1 or DIM Bit = 0.
11	R	USBC	USB Client 0 = No interrupt notification 1 = USB client interrupt has occurred, interrupt level<5> = 1, and either Mask Bit<5> = 1 or DIM Bit = 0.
10	R	GPIO_x	GPIO x 0 = No interrupt notification. 1 = GPIO_x (other than GPIO_0 and GPIO_1) edge detect =1, interrupt level<10> = 1, and either Mask Bit<10> = 1 or DIM Bit = 0.
9	R	GPIO_1	GPIO 1 0 = No interrupt notification 1 = GPIO<1> detected an edge, interrupt level<9> = 1, and either Mask Bit<9> = 1 or DIM Bit = 0.
8	R	GPIO_0	GPIO 0 0 = No interrupt notification 1 = GPIO<0> detected an edge, interrupt level<8> = 1, and either Mask Bit<8> = 1 or DIM Bit = 0.
7	R	OST_4_11	OS Timer 4–11 0 = No interrupt notification 1 = OS timer match 4-11 has occurred, interrupt level<7> = 1, and either Mask Bit<7> = 1 or DIM Bit = 0.

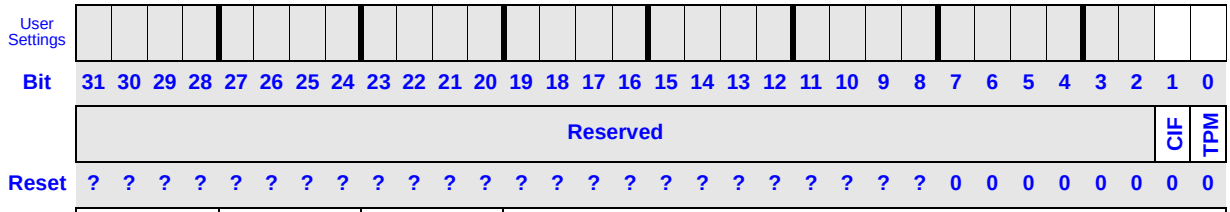
Table 25-7. ICFP Bit Definitions (Sheet 4 of 4)

	Physical Address 0x40D0_000C Coprocessor Register CR3								ICFP								Interrupt Controller															
User Settings																																
Bit	**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**
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
6	R	PWR_I2C	Power I^2C 0 = No interrupt notification 1 = I^2C Power Unit interrupt has occurred, interrupt level<0> = 1, and either Mask Bit<0> = 1 or DIM Bit = 0.
5	R	MEM_STK	Memory Stick 0 = No interrupt notification 1 = Memory Stick Host controller request has occurred, interrupt level<24> = 1, and either Mask Bit<24> = 1 or DIM Bit = 0.
4	R	KEYPAD	Keypad 0 = No interrupt notification 1 = Keypad controller interrupt has occurred, interrupt level<4> = 1, and either Mask Bit<4> = 1 or DIM Bit = 0.
3	R	USBH1	USB Host 1 0 = No interrupt notification 1 = USB host interrupt 1 (OHCI) has occurred, interrupt level<3> = 1, and either Mask Bit<3> = 1 or DIM Bit = 0.
2	R	USBH2	USB Host 2 0 = No interrupt notification 1 = USB host interrupt 2 has occurred, interrupt level<2> = 1, and either Mask Bit<2> = 1 or DIM Bit = 0.
1	R	MSL	MSL 0 = No interrupt notification 1 = MSL interrupt 1 has occurred, interrupt level<1> = 1, and either Mask Bit<1> = 1 or DIM Bit = 0.
0	R	SSP3	SSP 3 0 = No interrupt notification 1 = SSP three service request has occurred, interrupt level<0> = 1, and either Mask Bit<0> = 1 or DIM Bit = 0.

Table 25-8. ICFP2 Bit Definitions

Physical Address 0x40D0_00A8 Coprocessor Register Number CR9	ICFP2	Interrupt Controller



Bits	Access	Name	Description
31-2	—	—	reserved
1	R	CIF	Quick Capture Interface 0 = No interrupt notification 1 = Quick capture Interface interrupt occurs and ((Mask Bit(0) = 0b1)OR (DIM Bit = 0b0)) and (interrupt level (33) = 0b1)
0	R	TPM	Trusted Platform Module 0 = No interrupt notification 1 = Trusted Platform Module service request has occurred, interrupt level<0> = 1, and either Mask Bit<0> = 1 or DIM Bit = 0.

25.5.4 Interrupt Controller Mask Registers (ICMR and ICMR2)

The read/write Interrupt Controller Mask registers (ICMR and ICMR2) contains one mask bit per pending interrupt. Any active interrupt from the source is sent to the IRQ or FIQ (depending on the level set for the interrupt) if the corresponding mask bit in ICMR or ICMR2 is set, unless the processor is in the Idle state and ICCR[DIM] is clear.

The ICMR and ICMR2 bits are initialized and reset to 0s, which indicates that all interrupts are masked at reset and that ICMR and ICMR2 must be configured by software.

Table 25-9 and Table 25-10 shows the ICMR and the ICMR2 register bit locations that correspond to the interrupt mask bits.

Table 25-9. ICMR Bit Definitions (Sheet 1 of 3)

Physical Address 0x40D0_0004 Coprocessor Register CR1 — **ICMR** — **Interrupt Controller**

	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
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
31	R/W	RTC_AL	Real-Time Clock Alarm 0 = Masked. 1 = RTC equals alarm register interrupt is not to be masked.
30	R/W	RTC_HZ	One Hz Clock 0 = Masked. 1 = One Hz clock TIC interrupt is not to be masked.
29	R/W	OST_3	OS Timer 3 0 = Masked. 1 = OS timer equals match register 3 interrupt is not to be masked.
28	R/W	OST_2	OS Timer 2 0 = Masked. 1 = OS timer equals match register 2 interrupt is not to be masked.
27	R/W	OST_1	OS Timer 1 0 = Masked. 1 = OS timer equals match register 1 interrupt is not to be masked.
26	R/W	OST_0	OS Timer 0 0 = Masked. 1 = OS timer equals match register 0 interrupt is not to be masked.
25	R/W	DMAC	DMA Controller 0 = Masked. 1 = DMA Channel service request interrupt is not to be masked.
24	R/W	SSP1	SSP 1 0 = Masked. 1 = SSP 1 service request interrupt is not to be masked.
23	R/W	MMC	MultiMediaCard 0 = Masked. 1 = Flash card interrupt is not to be masked.
22	R/W	FFUART	FFUART 0 = Masked. 1 = FFUART interrupt is not to be masked.
21	R/W	BTUART	BTUART 0 = Masked. 1 = BTUART interrupt is not to be masked.

Table 25-9. ICMR Bit Definitions (Sheet 2 of 3)

Physical Address 0x40D0_0004 Coprocessor Register CR1 — **ICMR** — **Interrupt Controller**

	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
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
20	R/W	STUART	STUARTS 0 = Masked. 1 = STUART interrupt is not to be masked.
19	R/W	ICP	Infrared Communications Port 0 = Masked. 1 = ICP interrupt is not to be masked.
18	R/W	I2C	I^2C 0 = Masked. 1 = I^2C interrupt is not to be masked.
17	R/W	LCD	LCD Controller 0 = Masked. 1 = LCD controller interrupt is not to be masked.
16	R/W	SSP2	SSP 2 0 = Masked. 1 = SSP 2 service request interrupt is not to be masked.
15	R/W	USIM	USIM 0 = Masked. 1 = Smart card interface status/error interrupt is not to be masked.
14	R/W	AC97	AC97 0 = Masked. 1 = AC '97 interrupt is not to be masked.
13	R/W	I2S	I^2S 0 = Masked. 1 = I^2S interrupt is not to be masked.
12	R/W	PMU	Performance Monitor Unit 0 = Masked. 1 = PMU interrupt is not to be masked.
11	R/W	USBC	USB Client 0 = Masked. 1 = USB client interrupt is not to be masked.
10	R/W	GPIO_x	GPIO_x 0 = Masked. 1 = GPIO_x (other than GPIO_0 and GPIO_1) edge detected interrupt is not to be masked.

Table 25-9. ICMR Bit Definitions (Sheet 3 of 3)

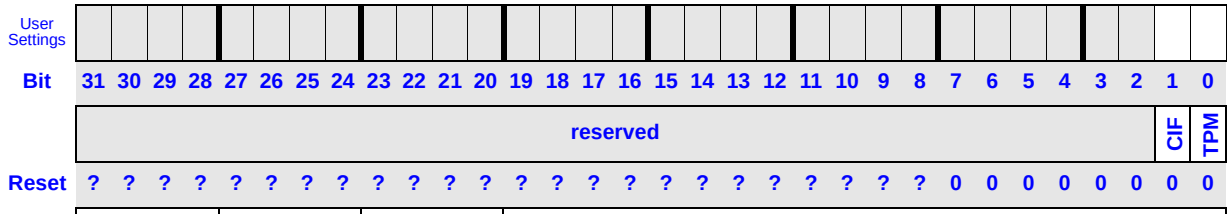
Physical Address 0x40D0_0004 Coprocessor Register CR1	ICMR	Interrupt Controller

	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
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	MEM_STK	KEYPAD	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
9	R/W	GPIO_1	GPIO_1 0 = Masked. 1 = GPIO<1> edge detect interrupt is not to be masked.
8	R/W	GPIO_0	GPIO_0 0 = Masked. 1 = GPIO<0> edge detect interrupt is not to be masked.
7	R/W	OST_4_11	OS Timer 4–11 0 = Masked. 1 = OS timer match 4-11 interrupt is not to be masked.
6	R/W	PWR_I2C	Power Manager I^2C 0 = Masked. 1 = I^2C power interrupt is not to be masked.
5	R/W	MEM_STK	Memory Stick 0 = Masked. 1 = Memory stick host controller service request is not to be masked.
4	R/W	KEYPAD	Keypad Controller 0 = Masked. 1 = Keypad controller interrupt is not to be masked.
3	R/W	USBH1	USB Host 1 0 = Masked. 1 = USB host interrupt 1 (OHCI) is not to be masked.
2	R/W	USBH	USB Host 2 0 = Masked. 1 = USB host interrupt 2 is not to be masked.
1	R/W	MSL	MSL 0 = Masked. 1 = MSL interrupt is not to be masked.
0	R/W	SSP3	SSP 3 0 = Masked. 1 = SSP 3 service request interrupt is not to be masked.

Table 25-10. ICMR2 Bit Definitions

Physical Address 0x40D0_00A0 Coprocessor Register Number CR7	ICMR2	Interrupt Controller



Bits	Access	Name	Description
31-2	—	—	reserved
1	R/W	CIF	Quick Capture Interface 0 = Masked. 1 = CIF interrupt is not to be masked.
0	R/W	TPM	Trusted Platform Module 0 = Masked. 1 = TPM interrupt is not to be masked.

25.5.5 Interrupt Controller Level Registers (ICLR and ICLR2)

The Interrupt Controller Level registers (ICLR and ICLR2) controls whether a pending interrupt generates an FIQ or an IRQ interrupt. If a pending interrupt is unmasked, the corresponding ICLR (or ICLR2) bit field is decoded to select which CPU interrupt is asserted. Table 25-11 and Table 25-12 shows the location of all interrupt-level bits in the ICLR and the ICLR2. The ICLR and the ICLR2 registers is initialized to all zeros at reset and software must configure them to reflect the correct value for normal operation.

Table 25-11. ICLR Bit Definitions (Sheet 1 of 3)

Physical Address 0x40D0 0008 Coprocessor Register CR2 — **ICLR** — **Interrupt Controller**

	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
User Settings																																
Bit	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Mem_stk	Keypad	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
31	R/W	RTC_AL	Real-Time Clock Alarm 0 = RTC equals alarm register interrupt creates an IRQ. 1 = RTC equals alarm register interrupt creates an FIQ.
30	R/W	RTC_HZ	One Hz Clock 0 = One Hz clock TIC interrupt creates an IRQ. 1 = One Hz clock TIC interrupt creates an FIQ.
29	R/W	OST_3	OS Timer 3 0 = OS timer equals match register 3 interrupt creates an IRQ. 1 = OS timer equals match register 3 interrupt creates an FIQ.
28	R/W	OST_2	OS Timer 2 0 = OS timer equals match register 2 interrupt creates an IRQ. 1 = OS timer equals match register 2 interrupt creates an FIQ.
27	R/W	OST_1	OS Timer 1 0 = OS timer equals match register 1 interrupt creates an IRQ. 1 = OS timer equals match register 1 interrupt creates an FIQ.
26	R/W	OST_0	OS Timer 0 0 = OS timer equals match register 0 interrupt creates an IRQ. 1 = OS timer equals match register 0 interrupt creates an FIQ.
25	R/W	DMAC	DMA Controller 0 = DMA Channel service request interrupt creates an IRQ. 1 = DMA Channel service request interrupt creates an FIQ.
24	R/W	SSP1	SSP 1 0 = SSP 1 service request interrupt creates an IRQ 1 = SSP 1 service request interrupt creates an FIQ
23	R/W	MMC	MultiMediaCard 0 = MMC interrupt creates an IRQ 1 = MMC card interrupt creates an FIQ
22	R/W	FFUART	FFUART 0 = FFUART interrupt creates an IRQ. 1 = FFUART interrupt creates an FIQ.
21	R/W	BTUART	BTUART 0 = BTUART interrupt creates an IRQ. 1 = BTUART interrupt creates an FIQ.

Table 25-11. ICLR Bit Definitions (Sheet 2 of 3)

Physical Address 0x40D0 0008 Coprocessor Register CR2	ICLR	Interrupt Controller

User Settings																															
Bit	**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**
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Mem_stk	Keypad	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
20	R/W	STUART	STUART 0 = STUART interrupt creates an IRQ. 1 = STUART interrupt creates an FIQ.
19	R/W	ICP	Infrared Communications Port 0 = ICP interrupt creates an IRQ. 1 = ICP interrupt creates an FIQ.
18	R/W	I2C	I^2C 0 = I^2C interrupt creates an IRQ. 1 = I^2C interrupt creates an FIQ.
17	R/W	LCD	LCD Controller 0 = LCD controller interrupt creates an IRQ. 1 = LCD controller interrupt creates an FIQ.
16	R/W	SSP2	SSP 2 0 = SSP 2 service request interrupt creates an IRQ. 1 = SSP 2 service request interrupt creates an FIQ.
15	R/W	USIM	USIM 0 = Smart card interface status/error interrupt creates an IRQ. 1 = Smart card interface status/error interrupt creates an FIQ.
14	R/W	AC97	AC97 0 = AC '97 interrupt creates an IRQ. 1 = AC '97 interrupt creates an FIQ.
13	R/W	I2S	I^2S 0 = I^2S interrupt creates an IRQ. 1 = I^2S interrupt creates an FIQ.
12	R/W	PMU	Performance Monitor Unit 0 = PMU interrupt creates an IRQ. 1 = PMU interrupt creates an FIQ.
11	R/W	USBC	USB Client 0 = USB client interrupt creates an IRQ. 1 = USB client interrupt creates an FIQ.
10	R/W	GPIO_x	GPIO_x 0 = GPIO_x (other than GPIO_0 and GPIO_1) edge detected interrupt creates an IRQ. 1 = GPIO_x (other than GPIO_0 and GPIO_1) edge detected interrupt creates an FIQ.

Table 25-11. ICLR Bit Definitions (Sheet 3 of 3)

Physical Address 0x40D0 0008 Coprocessor Register CR2 — ICLR — Interrupt Controller

	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
User Settings																																
Bit	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
	RTC_AL	RTC_HZ	OST_3	OST_2	OST_1	OST_0	DMAC	SSP1	MMC	FFUART	BTUART	STUART	ICP	I2C	LCD	SSP2	USIM	AC97	I2S	PMU	USBC	GPIO_x	GPIO_1	GPIO_0	OST_4_11	PWR_I2C	Mem_stk	Keypad	USBH1	USBH2	MSL	SSP3
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

Bits	Access	Name	Description
9	R/W	GPIO_1	GPIO_1 0 = GPIO<1> edge detect interrupt creates an IRQ. 1 = GPIO<1> edge detect interrupt creates an FIQ.
8	R/W	GPIO_0	GPIO_0 0 = GPIO<0> edge detect interrupt creates an IRQ. 1 = GPIO<0> edge detect interrupt creates an FIQ.
7	R/W	OST_4_11	OS Timer 4–11 0 = OS timer match 4-11 interrupt creates an IRQ. 1 = OS timer match 4-11 interrupt creates an FIQ.
6	R/W	PWR_I2C	Power I^2C 0 = I^2C power unit interrupt creates an IRQ. 1 = I^2C power unit interrupt creates an FIQ.
5	R/W	MEM_STK	Memory Stick Host 0 = Memory stick host controller interrupt creates an IRQ. 1 = Memory stick host controller interrupt creates an FIQ.
4	R/W	KEYPAD	Keypad 0 = Keypad controller interrupt creates an IRQ. 1 = Keypad controller interrupt creates an FIQ.
3	R/W	USBH1	USB Host 1 0 = USB host interrupt 1 (OHCI) creates an IRQ. 1 = USB host interrupt 1 (OHCI) creates an FIQ.
2	R/W	USBH2	USB Host 2 0 = USB host interrupt 2 creates an IRQ. 1 = USB host interrupt 2 creates an FIQ.
1	R/W	MSL	MSL 0 = MSL interrupt creates an IRQ. 1 = MSL interrupt creates an FIQ.
0	R/W	SSP3	SSP 3 0 = SSP 3 service request interrupt creates an IRQ. 1 = SSP 3 service request interrupt creates an FIQ.

Table 25-12. ICLR2 Bit Definitions

Physical Address 0x40D0 00A4 Coprocessor Register Number CR8	ICLR2	Interrupt Controller

User Settings																															
Bit	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																														CIF	TPM
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31-2	—	—	reserved
1	R	CIF	Quick Capture Interface 0 = Quick capture interface service request interrupt creates an IRQ. 1 = Quick capture interface service request interrupt creates a FIQ.
0	R	TPM	Trusted Platform Module 0 = Trusted Platform Module service request interrupt creates an IRQ. 1 = Trusted Platform Module service request interrupt creates a FIQ.

25.5.6 Interrupt Controller Control Register (ICCR)

The Interrupt Controller Control register (ICCR) contains a single control bit, the disable idle mask bit (DIM). When set, this bit inhibits the idle mode in which any active interrupt can interrupt the CPU, regardless of the value in ICMR. Table 25-13 shows the location of the DIM bit in the ICCR.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 25-13. ICCR Bit Definitions

Physical Address 0x40D0 0014	ICCR	Interrupt Controller

User Settings																															
Bit	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																															DIM
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0

Bits	Access	Name	Description
31:1	—	—	reserved
0	R/W	DIM	0 = Any interrupt in ICPR brings the processor out of idle mode. 1 = Only active, unmasked interrupts (as defined in the ICMR) bring the processor out of idle mode. Cleared during resets.

25.5.7 Interrupt Priority Registers 0–39 (IPRx)

These registers set a peripheral ID for each priority value. At reset, these registers are marked invalid. Software must program the values in these registers after reset, before it unmasks the interrupts.

These registers allow unique priority values to be defined for each peripheral. The interrupt controller uses the priority values to resolve situations in which multiple active unmasked interrupts occur at the same time. Each register contains the peripheral ID assigned to the corresponding priority. The priority value for the register is the register number. Priority 0 is the highest priority and 39 is the lowest priority. The peripheral ID with priority 0 is written to IPR0. IPR39 contains the peripheral ID with priority 39. The contents of the IPRs typically are not changed during execution.

If the contents of these registers are changed during execution, all interrupts must be masked. The interrupts must be masked until the changes written to the IPRs have taken effect. Interrupts can be disabled by writing to the Core Program Status register (CPSR) and altering its F-bit and I-bit.

The IPRx[31] is the valid bit (0b1 = valid and 0b0 = invalid). The peripherals IDs range between 0 and 39. Peripheral IDs for different sources are shown in Table 25-2. The lower six bits assign peripheral IDs. A valid peripheral ID must be in to the priority registers. However, if an invalid or non-existing peripheral ID (40 or 52, for example) is programmed in to one of the priority registers, it will be ignored and not have any effect on the determination of the highest priority FIQ and IRQ interrupts. Valid peripheral IDs are 0 through 39, however there are currently only 34 interrupts incoming to the interrupt controller, and hence IDs 34 through 39 correspond to interrupt lines that will be tied to 0 for now. The interrupt controller logic can extend to 40 interrupts if future expansions require new interrupts to be added.

To ease the highest priority selection process in the hardware, only one peripheral ID can be assigned to a single priority value. In addition, hardware does not offer any protection against assigning one peripheral ID to multiple priority values. Software must ensure that such cases are avoided. In a case in which multiple-priority values are assigned to one peripheral, the hardware uses only the highest priority values and ignores the others.

Table 25-14 shows the format for IPR registers.

This is a read/write register. Ignore reads from reserved bits. Reserved bits must be written with zeros.

Table 25-14. IPR0/39 Bit Definitions

Physical Addresses 0x40D0_001C-0098 (IPR 0–31) and 0x40D0_00B0-00CC (IPR 32–39) Not Mapped to Coprocessor Registers	IPR0–IPR39	Interrupt Controller

	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
User Settings																																
Bit	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
	VAL	Reserved																									PID					
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	?	?	?	?	?

Bits	Access	Name	Description
31	R/W	VAL	Valid Bit 0 = Peripheral ID contained in the register is not valid. 1 = Peripheral ID contained in the register is valid.
30:6	—	—	reserved
5:0	R/W	PID	Peripheral ID for this Priority IPR[n] Peripheral ID of peripheral with priority n (n=IPR number). Valid IDs: 0 through 39

25.5.8 Interrupt Control Highest Priority Register (ICHP)

This register contains the highest priority peripheral ID that caused an interrupt. The register contains peripheral IDs for the level of interrupts for FIQ and IRQ. If no interrupting peripheral for a particular interrupt level exists, the corresponding peripheral ID field is invalidated. The register is updated when an unmasked interrupt occurs. If ICPs are partially defined, the ICHP uses the partial information to determine the highest priority peripheral. If none of the ICP fields is defined, ICHP contains invalidated values in both fields. After an interrupt is served and the status bit is set in the Peripheral Status register, the ICHP register is updated with the peripheral ID with the next highest priority.

Changes in the IPR registers can also lead to an update in ICHP.

Table 25-15 shows the format for ICHP register.

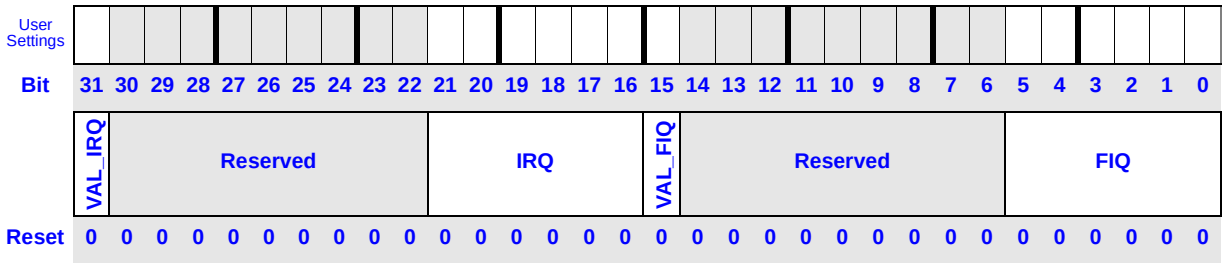
This is a read-only register. Ignore reads from reserved bits.

Table 25-15. ICHP Bit Definitions

Physical Address
0x40D0_0018
Coprocessor Register Number
CR5

ICHP

Interrupt Controller



Bits	Access	Name	Description
31	R	VAL_IRQ	Valid IRQ 0 = No valid peripheral ID that is causing an IRQ interrupt. 1 = Valid peripheral ID is causing an IRQ interrupt.
30:22	—	—	reserved
21:16	R	IRQ	IRQ Highest Priority Field Peripheral ID with the highest IRQ priority. When no interrupt has occurred, this bit is set to –1.
15	R	VAL_FIQ	Valid FIQ 0 = No valid peripheral ID that is causing an FIQ interrupt. 1 = Valid peripheral ID is causing an FIQ interrupt.
14:6	—	—	reserved
5:0	R	FIQ	FIQ Highest Priority Field Peripheral ID with the highest FIQ priority. When no interrupt has occurred, this bit is set to –1.

25.6 Register Summary

Table 25-16 shows the registers that are associated with the interrupt controller and their physical addresses.

Table 25-16. Interrupt Controller Register Summary

Address	Name	Description	Page
0x40D0_0000	ICIP	Interrupt Controller IRQ Pending register	25-11
0x40D0_0004	ICMR	Interrupt Controller Mask register	25-20
0x40D0_0008	ICLR	Interrupt Controller Level register	25-24
0x40D0_000C	ICFP	Interrupt Controller FIQ Pending register	25-15
0x40D0_0010	ICPR	Interrupt Controller Pending register	25-6
0x40D0_0014	ICCR	Interrupt Controller Control register	25-27
0x40D0_0018	ICHP	Interrupt Controller Highest Priority register	25-30
0x40D0_001C–0x40D0_0098	IPR0–IPR31	Interrupt Priority registers for Priorities 0–31	25-29
0x40D0_009C	ICIP2	Interrupt Controller IRQ Pending register 2	25-10
0x40D0_00A0	ICMR2	Interrupt Controller Mask register 2	25-23
0x40D0_00A4	ICLR2	Interrupt Controller Level register 2	25-27
0x40D0_00A8	ICFP2	Interrupt Controller FIQ Pending register 2	25-19
0x40D0_00AC	ICPR2	Interrupt Controller Pending register 2	25-6
0x40D0_00B0–0x40D0_00CC	IPR32–IPR39	Interrupt Priority registers for Priorities 32–39	25-29
0x40D0_00D0–0x40DF_FFFC	—	reserved	—

Software Debug 26

This chapter describes the debug architecture and related features implemented in the PXA27x processor.

26.1 Overview

This chapter refers to a *debugger* and a *debug handler.* These key terms require clear definition in debugging. They concern the differences between the host and target portions of a debugging scenario:

- The *debugger* is software that runs on a host system outside the PXA27x processor.
- The *debug handler* is a software event handler that runs on the processor when a debug event occurs.

The core debug unit, when used with a debugger application, allows software running on a core target to be debugged. The debug unit allows the debugger to stop program execution and redirect execution to a debug handling routine. When the program has stopped, the debugger can examine or modify the processor state, coprocessor state, or memory. The debugger can then restart the program.

The external debug interface to the PXA27x processor uses the JTAG port. For information on using the JTAG interface, see the “JTAG Debug Interface” chapter in the *Intel® PXA27x Processor Family Design Guide*.

26.2 Features

The debug unit provides the following features:

- A mechanism to load the instruction cache through JTAG
- A serial debug communications link using the JTAG interface
- A trace buffer
- Four hardware instruction and data breakpoints
- A mini-instruction cache
- Debug modes, registers, and exceptions

26.3 Signal Descriptions

The JTAG signals that interface with the Test Access Port (TAP) controller are described in the “JTAG Debug Interface” chapter in the *Intel® PXA27x Processor Family Design Guide*.

26.4 Operation

This section describes the debug architecture, its operation, and related features that are implemented in the PXA27x processor.

26.4.1 Debug Exceptions

A debug exception causes the processor to redirect program execution to a debug event handling routine. The core debug architecture defines the following debug exceptions:

- Instruction breakpoint
- Data breakpoint
- Software breakpoint
- External debug break
- Exception vector trap
- Trace-buffer full break

When a debug exception occurs, the action taken by the processor depends on whether the debug unit is configured for halt or monitor mode.

Table 26-1 shows the priority of debug exceptions relative to other processor exceptions.

Table 26-1. Event Priority

Event	Priority
Reset	1
Vector trap†	2
Data abort (precise)	3
Data breakpoint	4
Data abort (imprecise)	5
External debug break, trace-buffer full	6
FIQ	7
IRQ	8
Instruction breakpoint	9
Prefetch abort	10
Undefined, **SWI**, **BKPT**	11
† See Table 26-7 for vector trap options.	

26.4.1.1 Halt Mode

When the debug unit is configured for halt mode, the reset vector is overloaded to serve as the debug vector. Another processor mode, debug mode (CPSR[4:0] = 0x15), allows debug exceptions to be handled similarly to other types of ARM* exceptions.

The debugger turns on halt mode through the JTAG interface by scanning in a value that sets the bit in the Debug Control and Status register (DCSR; see Section 26.5.2). The debugger turns off halt mode through JTAG, either by scanning in a new DCSR value or by a JTAG reset (TRST). Processor reset does not affect the value of the halt mode bit.

When a debug exception occurs and halt mode is active, the processor switches to debug mode and uses the reset vector as the debug vector. After the debug handler starts, the debugger can communicate with the debug handler to examine or alter processor state or memory through the JTAG interface.

Through the JTAG interface, the debug handler and exception vectors can be downloaded and locked directly into the instruction cache to intercept the default vectors and reset handler. Downloading them into the instruction cache means that external memory is not required to contain debug handler code. The debug handler and exception vectors can also reside in external memory. Downloading into the instruction cache allows a system with memory problems or no external memory to be debugged. Refer to Section 26.4.6.3 for details about downloading code into the instruction cache.

During halt mode, software running on the core cannot access DCSR or any of the hardware breakpoint registers unless the processor is in the special debug state (SDS).

When a debug exception occurs during halt mode, the processor takes the following actions:

- Disables the trace buffer
- Sets DCSR[MOE] encoding (see Table 26-7)
- Enters SDS
- For data breakpoints, trace-buffer full break, and external debug break:
  R14_dbg = PC of the next instruction to execute + 4
- For instruction breakpoints, software breakpoints, and vector traps:
  R14_dbg = PC of the aborted instruction + 4
- SPSR_dbg = CPSR
- CPSR[4:0] = 0b1_0101 (debug mode)
- CPSR[5] = 0
- CPSR[6] = 1
- CPSR[7] = 1
- PC = 0x0[1]

Following a debug exception, the processor switches to debug mode and enters SDS, which allows the following special functionality:

- All events are disabled. SWI and undefined instructions have unpredictable results. The processor ignores prefetch aborts, FIQ, and IRQ (SDS disables FIQ and IRQ regardless of the enable values in the CPSR). The processor reports data aborts detected during SDS by setting DCSR[SA] but does not generate an exception. The processor also sets up FSR and FAR as it normally would for a data abort.
- Normally, software cannot write to the hardware breakpoint registers or the DCSR during halt mode. However, during SDS, software has write access to the breakpoint registers (see Section 26.4.2) and the DCSR (see Table 26-7).

1. When the vector table is relocated (CP15 Control register[13] = 1), the debug vector is relocated to 0xFFFF_0000

- The core's instruction memory management unit (IMMU) is disabled. For more information about the IMMU, see the *Intel XScale® Core Developer's Manual*. In halt mode, the debug handler is typically downloaded directly into the instruction cache and it is not appropriate to perform TLB accesses or translation walks, because there may not be any external memory or, if there is, the translation table or TLB may not contain a valid mapping for the debug handler code. To avoid these problems, the processor internally disables the IMMU during SDS.
- The PID is disabled for instruction fetches. This prevents fetches of the debug handler code from being remapped to an address other than the address from which the code was downloaded. For more details on the PID, see "Register 13: Process ID" in the *Intel XScale® Core Developer's Manual*.

The SDS remains active regardless of the processor mode. This allows the debug handler to switch to other modes and maintain SDS functionality. Entering user mode causes unpredictable behavior. The processor exits SDS after a CPSR restore operation.

To exit, the debug handler uses:

```
su  bs pc, lr, #4
```

This restores CPSR, turns off SDS functionality, and branches to the target instruction.

26.4.1.2 Monitor Mode

In monitor mode, debug exceptions are handled as ARM* prefetch aborts or ARM* data aborts. If debug functionality is enabled (DCSR[GE] set) and the processor is in monitor mode, debug exceptions cause either a data abort or a prefetch abort.

When a debug exception occurs, the processor switches to abort mode and branches to a debug handler using the prefetch abort vector or data abort vector. The debugger then communicates with the debug handler to access processor state or memory contents.

In monitor mode, the processor handles debug exceptions as normal ARM* exceptions.

The following debug exceptions cause data aborts:

- Data breakpoint
- External debug break
- Trace-buffer full break

The following debug exceptions cause prefetch aborts:

- Instruction breakpoint
- BKPT instruction

The processor ignores vector traps during monitor mode.

When an exception occurs in monitor mode, the processor takes the following actions:

- Disables the trace buffer
- Sets DCSR[MOE] encoding
- Sets FSR[9]
- For data aborts: R14_abt = PC of the next instruction to execute + 4.
  For prefetch aborts: R14_abt = PC of the faulting instruction + 4.

- SPSR_abt = CPSR
- CPSR[4:0] = 0b1_0111 (abort mode)
- CPSR[5] = 0
- CPSR[6] = unchanged
- CPSR[7] = 1
- For data aborts: PC = 0x10.
  For prefetch aborts: PC = 0xC.

During abort mode, external debug breaks and trace-buffer full breaks are internally postponed. When the processor exits abort mode, either through a CPSR restore or a write directly to the CPSR, the postponed debug breaks immediately generate a debug exception. Any of these postponed debug breaks are cleared once any one debug exception occurs.

When exiting abort mode, the debug handler must perform a CPSR restore operation that branches to the next instruction to be executed in the program under debug.

26.4.2 Hardware Breakpoint Resources

The PXA27x debug architecture defines two instruction and two data breakpoint registers, denoted IBCR0 and IBCR1 (shown in Table 26-9) and DBR0 and DBR1 (shown in Table 26-10).

The instruction and data address breakpoint registers are 32-bit registers. The instruction breakpoint causes a break *before* execution of the target instruction. The data breakpoint causes a break *after* the memory access has been issued.

In this section, Modified Virtual Address (MVA) refers to the virtual address that the PID ORs. The processor does not OR the PID with the specified breakpoint address before it performs address comparison. Address comparison must be performed by the software and written to the breakpoint register as the MVA. This applies to data and instruction breakpoints.

26.4.2.1 Instruction Breakpoints

The debug architecture defines two instruction breakpoint registers (IBCR0 and IBCR1). The format of these registers is shown on Table 26-9. In ARM* mode, the upper 30 bits contain a word-aligned MVA to break on. In Thumb mode, the upper 31 bits contain a half-word-aligned MVA to break on. In each mode, bit 0 enables and disables the instruction breakpoint register. Enabling instruction breakpoints while debug is globally disabled (DCSR[GE] clear) results in unpredictable behavior.

An instruction breakpoint generates a debug exception before the instruction at the address specified in the IBCR executes. When an instruction breakpoint occurs, the processor sets DBCR[MOE] to 0b001.

Software must disable the breakpoint before exiting the handler. This allows the instruction to execute after the exception is handled.

Single step execution is accomplished using the instruction breakpoint registers and must be completely handled in software, either on the host or by the debug handler.

26.4.2.2 Data Breakpoints

The debug architecture defines two data breakpoint registers: DBR0 and DBR1. Table 26-10 shows the format of these registers.

DBR0 is a dedicated data address breakpoint register. DBR1 can be programmed for either of two operations:

- Second data address breakpoint
- Data address mask

The DBCON register controls the functionality of DBR1 and the enabling of DBR0 and DBR1. DBCON also determines the type of memory access on which to break. Table 26-8 shows the format of the DBCON register.

26.4.2.2.1 Data Address Breakpoint

A data address breakpoint is triggered if the memory access matches the access type and the address of any byte in the memory access matches the address in DBRx. For example, LDR triggers a breakpoint if DBCON[E0] is 0b10 or 0b11 and the address of any of the four bytes accessed by the load matches the address in DBR0.

The processor does not trigger data breakpoints for the PLD instruction or any CP15, register 7, 8, 9, or 10 functions. Any other type of memory access can trigger a data breakpoint. For data breakpoint purposes, the SWP and SWPB instructions are treated as stores. They do not cause a data breakpoint if the breakpoint is set up to break on loads only and an address match occurs.

On unaligned memory accesses, breakpoint address comparison occurs on a word-aligned address (aligned down to word boundary).

When a memory access triggers a data breakpoint, the breakpoint is reported after the access is issued. The memory access is not aborted by the processor. The actual timing of the completion of the access with respect to the start of the debug handler depends on the memory configuration.

On a data breakpoint, the processor generates a debug exception and redirects execution to the debug handler *before* the next instruction executes. The processor reports the data breakpoint by setting the DCSR[MOE] to 0b010. The link register of a data breakpoint is always the PC (of the next instruction to execute) + 4, whether the processor is configured for monitor mode or halt mode.

When DBR1 is programmed as a second data address breakpoint, it functions independently of DBR0 and is controlled by DBCON[E1].

26.4.2.2.2 Data Address Mask

When DBR1 is programmed as a data address mask, it is used in conjunction with the address in DBR0. The bits set in DBR1 are ignored by the processor when it compares the address of a memory access with the address in DBR0. Using DBR1 as a data address mask allows a range of addresses to generate a data breakpoint. When DBR1 is selected as a data address mask, it is unaffected by DBCON[E1]. The mask is used only when DBR0 is enabled.

26.4.3 Software Breakpoints

Mnemonics:	BKPT (See *ARM* Architecture Reference Manual*, ARMv5T)
Operation:	If DCSR[GE] is clear, BKPT is a no-operation. If DCSR[GE] is set, BKPT causes a debug exception.

The processor handles the software breakpoint as described in Section 26.4.1.

26.4.4 Normal RX Handshaking versus High-Speed Download

26.4.4.1 Normal Receive Register (RX) Handshaking

Debugger Actions

1. The debugger has data to send to the debug handler.
2. Before it writes new data to the RX register, the debugger must poll the RX register ready flag (TXRXCTRL[RR]) through JTAG until the register bit is cleared.
3. After the debugger reads 0b0 from TXRXCTRL[RR], it scans data into the JTAG TDI signal to write to the RX register and sets the valid bit. The write to the RX register automatically sets TXRXCTRL[RR].

Debug Handler Actions

1. The debug handler polls TXRXCTRL[RR] until it is set, indicating that data in the RX register is valid.
2. After TXRXCTRL[RR] is set, the debug handler reads the new data from the RX register. The read operation automatically clears TXRXCTRL[RR].

26.4.4.2 High-Speed Handshaking States

When data is being downloaded by the debugger, part of the normal handshaking can be bypassed to allow the download rate to be increased. Before the high-speed download starts, both the debugger and debug handler must be synchronized to ensure that the debug handler is executing a routine that supports the high-speed download.

Although it is similar to normal handshaking, the debugger polling of TXRXCTRL[RR] is bypassed on the assumption that the debug handler can read the previous data from the RX register before the debugger can scan in the new data.

Debugger Actions

1. The debugger has code to transfer into the processor system memory.
2. Before the download starts, the debugger polls TXRXCTRL[RR] until it is clear, which indicates that the debug handler is ready. Then, the debugger starts the download.
3. The debugger scans data into JTAG to write to the RX register with the download bit and the valid bitset. TXRXCTRL[RR] and TXRXCTRL[D] are automatically set following the write to the RX register.
4. Without polling TXRXCTRL[RR] to determine whether the debug handler has read the data just scanned in, the debugger continues to scan new data into JTAG for RX, with the download bit and the valid bit set.

5. An overflow condition occurs if the debug handler does not read the previous data before the debugger completely scans in the new data. See the TXRXCTRL[OV] bit description in Table 26-6 for details on the overflow condition.
6. After the download is complete, the debugger clears TXRXCTRL[D], which allows the debug handler to exit the download loop.

Debug Handler Actions

1. The debug handler is in a routine waiting to write data out to memory. The routine loops based on TXRXCTRL[D].
2. The debug handler polls TXRXCTRL[RR] until it is set. It then reads the register and writes its contents out to memory. The handler loops, repeating these operations until the debugger clears TXRXCTRL[D].

26.4.5 Executing Conditionally Using TXRXCTRL

The bits in the TXRXCTRL register are placed so that they can be read directly into the CC flags in the CPSR with an MRC (with Rd = PC). The subsequent instruction can then conditionally execute based on the updated CC value.

To simplify the debug handler, the TXRXCTRL register must be read using the following instruction:

```
mrc p14, 0, r15, C14, C0, 0
```

This instruction directly updates the condition codes in the CPSR. The debug handler can then conditionally execute based on each CC bit. Table 26-2 shows the mnemonic extension to conditionally execute based on whether the TXRXCTRL bit is set or clear.

Table 26-2. TXRXCTRL Mnemonic Extensions

TXRXCTRL Bit	Mnemonic Extension to Execute if Bit is Set	Mnemonic Extension to Execute if Bit is Clear
31 (to N flag)	MI	PL
30 (to Z flag)	EQ	NE
29 (to C flag)	CS	CC
28 (to V flag)	VS	VC

The following example is a code sequence in which the debug handler polls the TXRXCTRL handshaking bit to determine when the debugger has completed its write to RX and the data is ready for the debug handler to read:

```
loop:   mcr     p14, 0, r15, c14, c0, 0# read the handshaking bit in TXRXCTRL
        mcrmi   p14, 0, r0, c9, c0, 0  # if RX is valid, read it
        bpl     loop                   # if RX is not valid, loop
```

26.4.6 Debug JTAG Access

This section describes the debug interface with the PXA27x processor through the JTAG port. For more information on the JTAG instructions and controller states, see the “JTAG Debug Interface” chapter in the *Intel® PXA27x Processor Family Design Guide*.

26.4.6.1 JTAG Commands and Registers

The debugger uses four JTAG instructions during software debug: LDIC, SEL_DCSR, DBG_TX, and DBG_RX. The LDIC instruction is described in Section 26.4.6.3. The other three JTAG instructions are described in this section.

SEL_DCSR, DBG_TX and DBG_RX use a common 36-bit shift register, DBG_SR. New data is shifted in and captured data is shifted out through DBG_SR. In the *Update-DR* state, the new data is shifted into the appropriate data register. See the "JTAG Debug Interface" chapter in the *Intel® PXA27x Processor Family Design Guide* for details of the JTAG state machine.

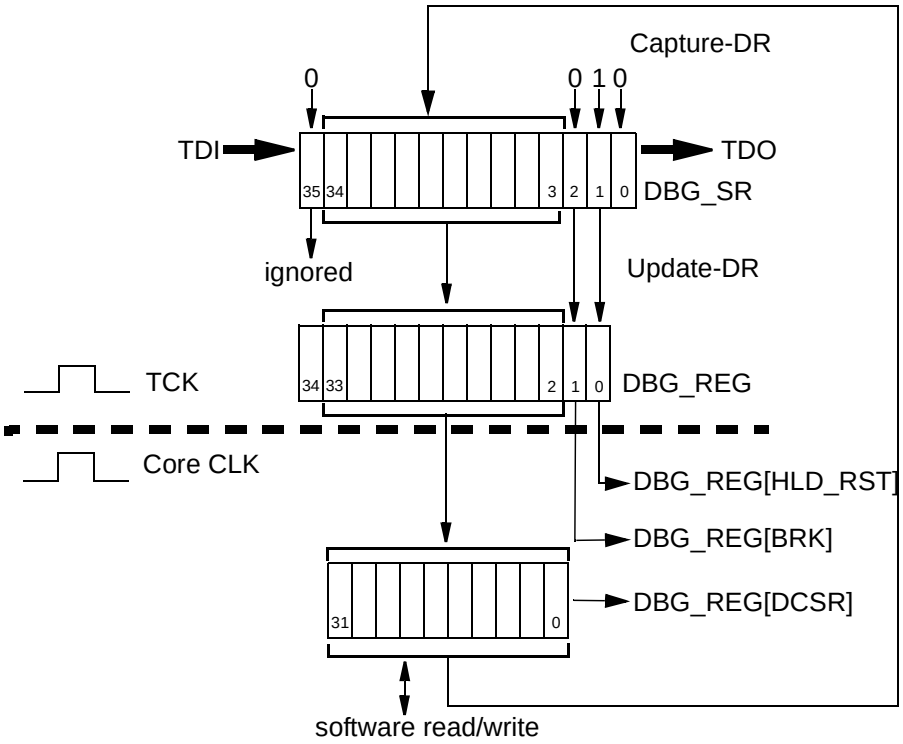
26.4.6.1.1 SEL_DCSR JTAG Command and Register

The SEL_DCSR JTAG instruction selects the DCSR JTAG data register. The JTAG opcode is 0b0_1001. When the SEL_DCSR JTAG instruction is in the JTAG instruction register, the debugger can directly access the DCSR. The debugger can only modify certain bits through JTAG, but it can read the entire register.

The SEL_DCSR instruction also allows the debugger to generate an external debug break.

Placing the SEL_DCSR JTAG instruction in the JTAG IR selects the DCSR JTAG Data register (Figure 26-1) and allows the debugger to access the DCSR, generate an external debug break, and set the hold_rst signal, which is used when loading code into the instruction cache during reset.

Figure 26-1. SEL_DCSR Hardware



A *Capture-DR* state loads the current DCSR value into DBG_SR[34:3]. The other bits in DBG_SR are loaded as shown in Figure 26-1.

During the *Shift-DR* state, a new DCSR value can be scanned into DBG_SR, and the previous value can be scanned out. When scanning a new DCSR value into DBG_SR, ensure that DBG_SR[2:1] is set up to prevent undesirable behavior.

The *Update-DR* state parallel-loads the new DCSR value into DBG_REG[33:2]. This value is then loaded into the actual DCSR register. All bits defined as JTAG writable in Table 26-7 are updated.

A debug host and the debug handler that is running on the core must synchronize access to the DCSR. If one writes to the DCSR at the same time the other reads from the DCSR, the results are unpredictable.

DBG_REG[HLD_RST]

The debugger uses DBG_REG[HLD_RST] to load code into the instruction cache during a processor reset. See Section 26.4.6.3 for details about loading code into the instruction cache.

The debugger must set DBG_REG[HLD_RST] before or during assertion of the reset pin. After DBG_REG[HLD_RST] is set, the reset pin can be deasserted and the processor remains in reset internally. The debugger then loads debug handler code into the instruction cache before the processor begins to execute any code.

After the code download is complete, the debugger must clear DBG_REG[HLD_RST]. This brings the processor out of reset and allows the code execution to begin at the reset vector.

A debugger sets DBG_REG[HLD_RST] in one of two ways:

- Placing the JTAG state machine into the *Capture-DR* state, which automatically loads DBG_SR[1] with 1, then placing the state machine into the *Exit2-DR* state, followed by the *Update-DR* state. This sets the DBG_REG[HLD_RST], clears DBG_REG[BRK], and leaves the DCSR unchanged. The DCSR bits captured in DBG_SR[34:3] are written back to the DCSR on the *Update-DR* state.
- Scanning a 1 into DBG_SR[1] and scanning in the appropriate values for the DCSR and DBG_REG[BRK].

DBG_REG[HLD_RST] can only be cleared by scanning in a 0 to DBG_SR[1] and scanning in the appropriate values for the DCSR and DBG_REG[BRK].

DBG_REG[BRK]

DBG_REG[BRK] allows the debugger to generate an external debug break and asynchronously redirect execution to a debug handling routine.

To set an external debug break, a debugger scans data into DBG_SR with DBG_SR[2] set and the desired value for the DCSR JTAG writable bits in DBG_SR[34:3].

After an external debug break is set, it remains set internally until a debug exception occurs. In monitor mode, external debug breaks detected during abort mode are postponed until the processor exits abort mode. In halt mode, breaks detected during SDS are postponed until the processor exits SDS. When an external debug break is detected outside of these two cases, the processor stops executing instructions when the current pipeline contents are completed. This improves breakpoint

accuracy by reducing the number of instructions that execute after the external debug break is requested. However, the processor continues to process any instructions that have already started. The processor does not enter debug mode until all processor activity stops.

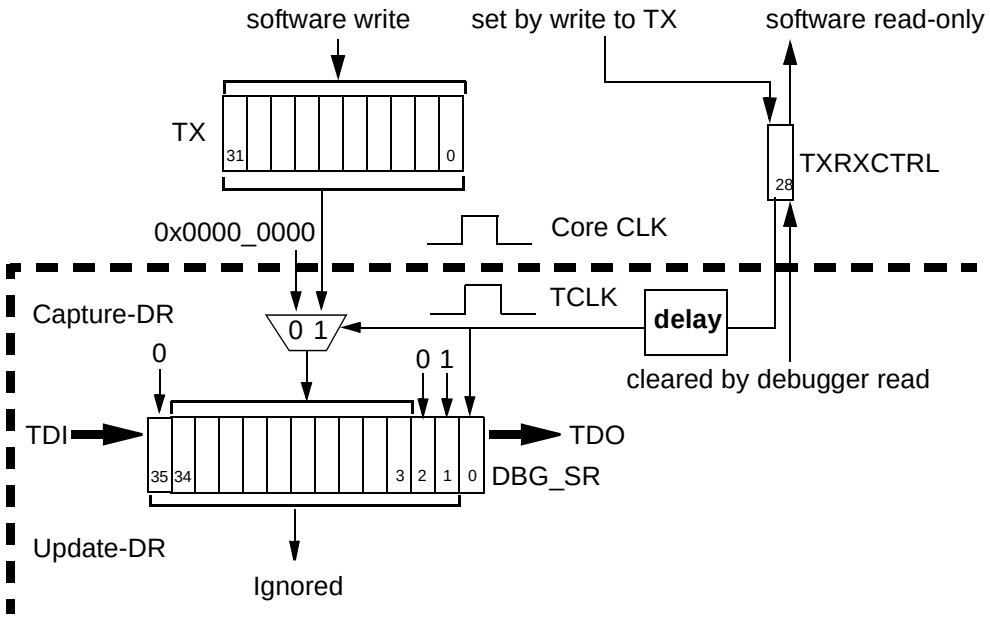
DBG_REG[DCSR]

The DCSR is updated with the value loaded into DBG_REG[DCSR] following an *Update-DR* state. Only the bit locations specified as writable by JTAG in Table 26-7 are updated.

26.4.6.1.2 DBG_TX JTAG Command and Register

The DBG_TX JTAG instruction selects the debug JTAG data register (Figure 26-2). The JTAG opcode for this instruction is 0b1_0000. The debugger uses the DBG_TX data register to poll for internal and external breaks that trigger the processor to enter debug mode. In debug mode, the debugger uses the DBG_TX data register to read data from the debug handler.

Figure 26-2. DBG_TX Hardware



A *Capture-DR* state loads the TX register value into DBG_SR[34:3] and the value in TXRXCTRL[TR] into DBG_SR[0]. The other bits in DBG_SR are loaded as shown in Table 26-2.

The captured TX value is scanned out during the *Shift-DR* state.

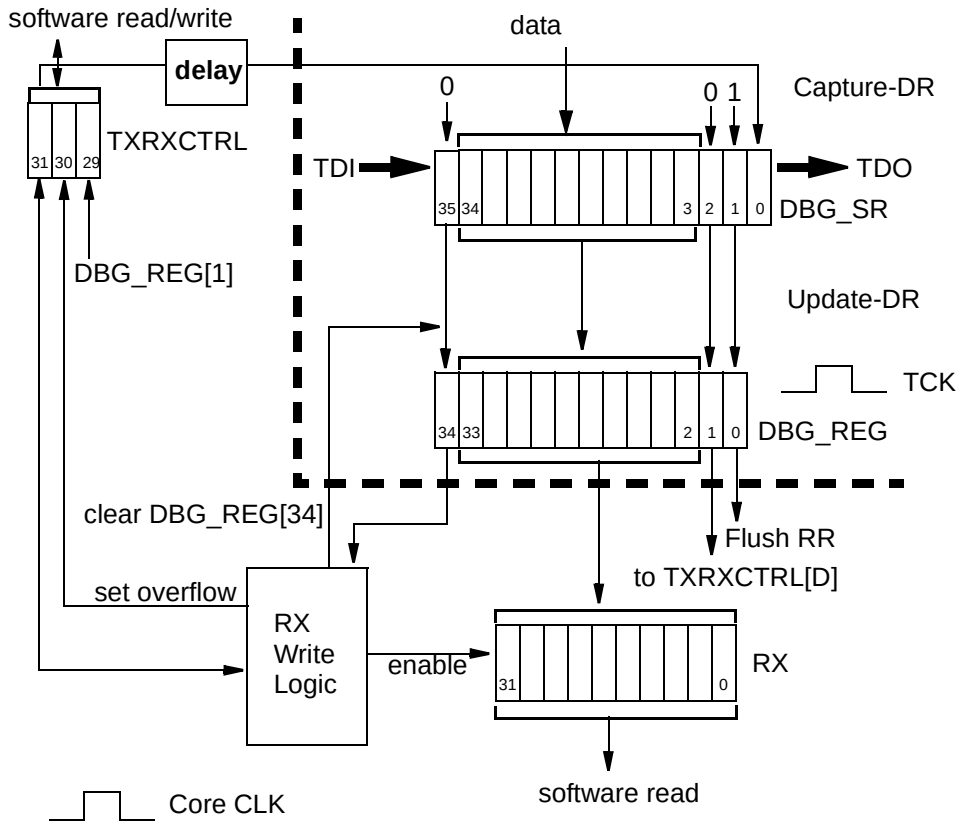
Data scanned into the TX register is ignored during the *Update-DR* state.

If DBG_SR[0] is set, the captured TX data is valid. After the debugger exits the *Capture-DR* state, it must place the JTAG state machine in the *Shift-DR* state. This ensures that the debugger performs a read, which clears TXRXCTRL[TR].

26.4.6.1.3 DBG_RX JTAG Command and Register

The DBG_RX JTAG instruction selects the DBG_RX JTAG data register. The JTAG opcode for this instruction is 0b0_0010. After the DBG_RX data register is selected, the debugger can send data or commands to the debug handler through the RX register.

Figure 26-3. DBG_RX Hardware



A *Capture-DR* state loads TXRXCTRL[RR] into DBG_SR[0]. The other bits in DBG_SR are loaded as shown in Figure 26-3.

The captured data is scanned out during the *Shift-DR* state.

Note: Incorrectly setting DBG_SR[35] or DBG_SR[1] while polling TXRXCTRL[RR] causes unpredictable behavior after an *Update-DR* state.

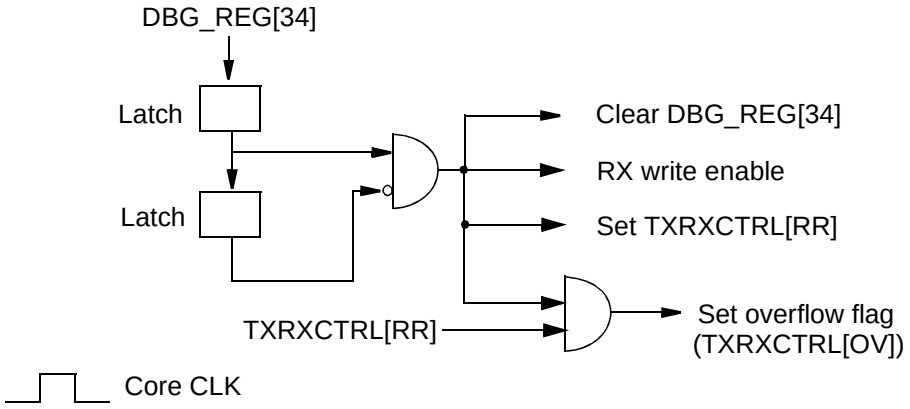
The *Update-DR* state parallel-loads DBG_SR[35:1] into DBG_REG[34:0]. The inputs to the RX write logic determine whether the new data is written to the RX register or an overflow condition is detected.

RX Write Logic

The RX write logic (Figure 26-4) serves four functions:

- Enables the debugger write to the RX register. The logic ensures that only new, valid data from the debugger is written to the RX register. In particular, when the debugger polls TXRXCTRL[RR] to determine whether the debug handler has read the previous data from the RX register. The JTAG state machine must go through *Update-DR*, which must not modify the RX register.
- Clears DBG_REG[34] to support high-speed downloads. During a high-speed download, the debugger continuously scans in data to send to the debug handler and sets DBG_REG[35], which is copied to DBG_REG[34] to signal that the data is valid. Because the debugger does not clear DBG_REG[34], the RX write logic clears DBG_REG[34], which signals the debugger write to RX.
- Sets TXRXCTRL[RR]. When the debugger writes new data to RX, the write logic sets TXRXCTRL[RR], which signals the debug handler that the data is valid.
- Sets the overflow flag (TXRXCTRL[OV]). During a high-speed download, the debugger does not poll to determine if the debug handler has read the previous data. The RX write logic sets the overflow flag when the previous data has not yet been read and the debugger has just written new data to RX. This prevents the debug handler from stalling long enough to allow the debugger to overwrite the data before it is read.

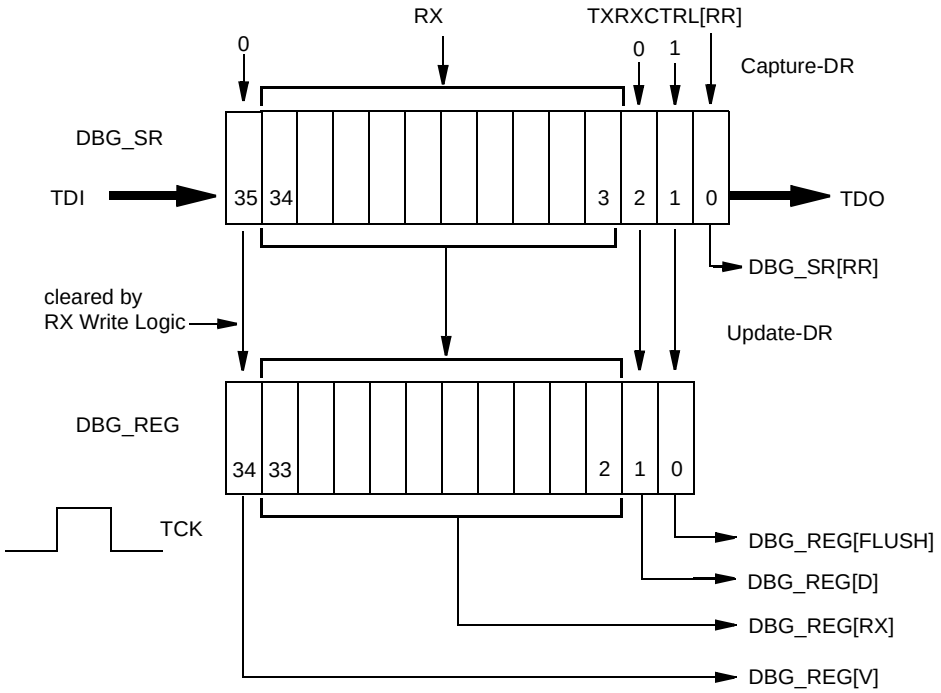
Figure 26-4. RX Write Logic



26.4.6.1.4 DBG_RX Data Register

The debugger uses the bits in the DBG_RX data register (Figure 26-5) to send data to the processor. The data register also contains a bit that flushes previously written data and a bit that is a high-speed download flag.

Figure 26-5. DBG_RX Data Register



DBG_SR[RR]

The debugger uses DBG_SR[RR] as part of the synchronization that occurs between the debugger and debug handler to allow them to access RX. This bit contains the value of TXRXCTRL[RR] after a *Capture-DR* state. The debug handler automatically sets TXRXCTRL[RR] by performing a write to RX.

The debugger polls DBG_SR[RR] to determine if the debug handler has read the previous data from the RX register.

The debugger sets TXRXCTRL[RR] by setting the DBG_REG[V] bit.

DBG_REG[V]

The debugger sets the DBG_REG[V] bit to indicate that the data scanned into DBG_SR[34:3] is valid data to write to RX. The bit is an input to the RX write logic and is cleared by the RX write logic.

When this bit is set, data scanned into DBG_SR is written to RX after an *Update-DR* state. If DBG_REG[V] is not set and the debugger performs an *Update-DR*, the RX register is not changed.

DBG_REG[V] does not affect the actions of DBG_REG[FLUSH] or DBG_REG[D].

DBG_REG[RX]

DBG_REG[RX] is written into the RX register based on the output of the RX write logic. Data to be sent from the debugger to the processor must be loaded into DBG_REG[RX] with DBG_REG[V] set. DBG_REG[RX] is loaded from DBG_SR[34:3] when the JTAG enters the *Update-DR* state.

DBG_REG[RX] is written to RX when the RX write logic enables the RX register after an *Update-DR* state.

DBG_REG[D]

DBG_REG[D] is used during high-speed downloads. It is written directly to TXRXCTRL[D]. The debugger sets DBG_REG[D] when it downloads a block of code or data to the system memory. The debug handler then uses TXRXCTRL[D] as a branch flag to determine the end of the loop.

Using DBG_REG[D] as a branch flags eliminates the need for a loop counter in the debug handler code. Eliminating the loop counter prevents a situation in which a debugger loop counter is out of synchronization with the debug handler's counter because an overflow condition has occurred.

DBG_REG[FLUSH]

DBG_REG[FLUSH] allows the debugger to flush data that has been previously written to RX. Setting DBG_REG[FLUSH] clears TXRXCTRL[RR].

26.4.6.2 Debug JTAG Data Register Reset Values

When TRST is asserted, the debug data register is reset. Asserting the nRESET pin does not affect the debug data register. Table 26-3 shows the reset and TRST values for the data register.

Note: These values apply for DBG_REG for SEL_DCSR, DBG_TX, and DBG_RX.

Table 26-3. Debug Data Register Reset Values

Bit	TRST	RESET
DBG_REG[0]	0	unchanged
DBG_REG[1]	0	unchanged
DBG_REG[33:2]	unpredictable	unpredictable
DBG_REG[34]	0	unchanged

26.4.6.3 Downloading Code into Instruction Cache

The processor core provides a 2-Kbyte mini-instruction cache that is physically separate from the 32-Kbyte main instruction cache and can be used as an on-chip instruction RAM. For additional information about the mini-instruction cache, see Section 26.4.6.3.6.

Note: The mini-instruction cache cannot be written with a cache line fill from external memory. It can be written only through JTAG.

A debug host can download code directly into either instruction cache through JTAG. The core also allows either instruction cache to be loaded during reset or program execution. Loading the instruction cache during normal program execution requires a strict handshaking protocol between the software running on the core and the debug host.

Throughout the remainder of Section 26.4.6.3, the term “instruction cache” applies to both the main and mini-instruction caches.

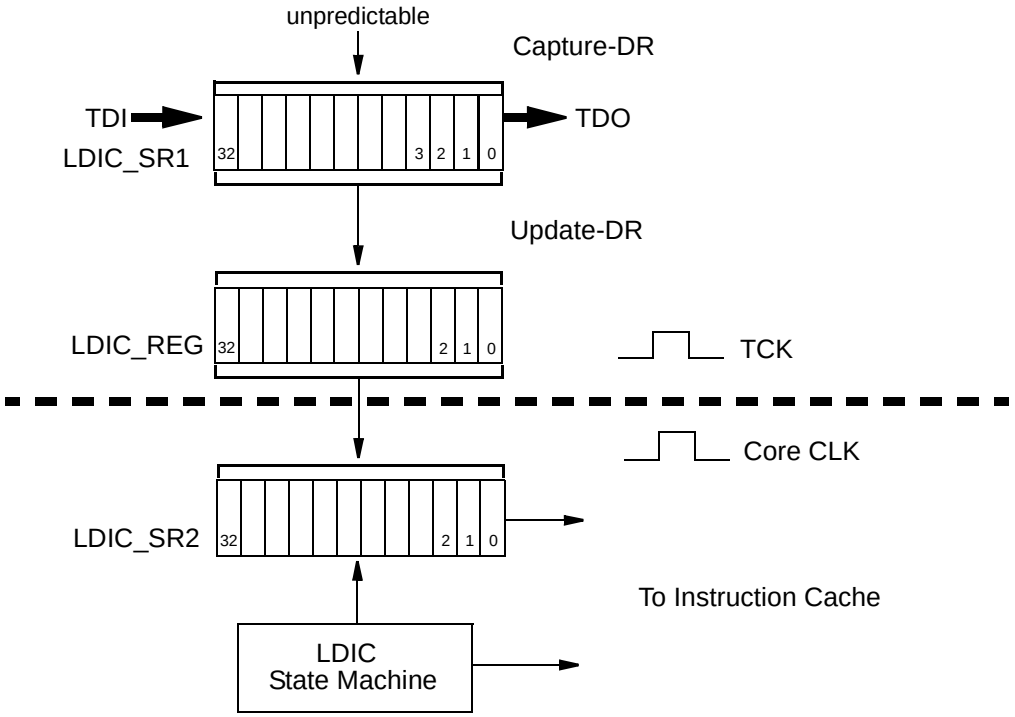
26.4.6.3.1 LDIC JTAG Command

The LDIC JTAG command selects the JTAG data register to load code into the instruction cache (IC). The JTAG opcode for this command is 0b0_0111. The LDIC command must be in the JTAG instruction register to load code directly into the instruction cache through JTAG.

26.4.6.3.2 LDIC JTAG Data Register

The LDIC JTAG data register is selected when the LDIC JTAG instruction is in the JTAG IR. A debug host can load and invalidate lines in the instruction cache through this data register.

Figure 26-6. LDIC JTAG Data Register Hardware



Note: The data loaded into LDIC_SR1 during a *Capture-DR* state is unpredictable.

LDIC functions and data consist of 33-bit packets that are scanned into LDIC_SR1 during the *Shift-DR* state.

The *Update-DR* state parallel-loads the value in LDIC_SR1 into LDIC_REG. LDIC_REG is then loaded into LDIC_SR2. After the data is loaded into LDIC_SR2, the LDIC state machine serially shifts the contents of LDIC_SR2 into the instruction cache.

Note: Removing the LDIC JTAG command from the JTAG IR before the entire contents of LDIC_SR2 are sent to the instruction cache causes unpredictable behavior. To avoid this behavior, the LDIC

command must remain in the JTAG IR for a minimum of 15 TCKs after the *Update-DR* state for the last LDIC packet. This ensures that the last packet is correctly sent to the instruction cache.

26.4.6.3.3 LDIC Cache Functions

The processor core supports four cache functions that can be executed through JTAG. Two functions allow a debug host to download code into an instruction cache through JTAG. Two additional functions allow invalidating of lines in the instruction cache. Table 26-4 shows the cache functions supported through JTAG.

Table 26-4. LDIC Cache Functions

Function	Encoding	Arguments	
		Address	Number of Data Words
Invalidate Main IC Line	0b000	VA of line to invalidate	0
Invalidate Mini IC	0b001	—	0
Load Main IC	0b010	VA of line to load	8
Load Mini IC	0b011	VA of line to load	8
reserved	0b100-0b111	—	—

Invalidate Main IC Line invalidates the line in the instruction cache (IC) that contains the specified virtual address. If the line is not in the cache, the operation has no effect. Invalidate Main IC Line does not take any data arguments.

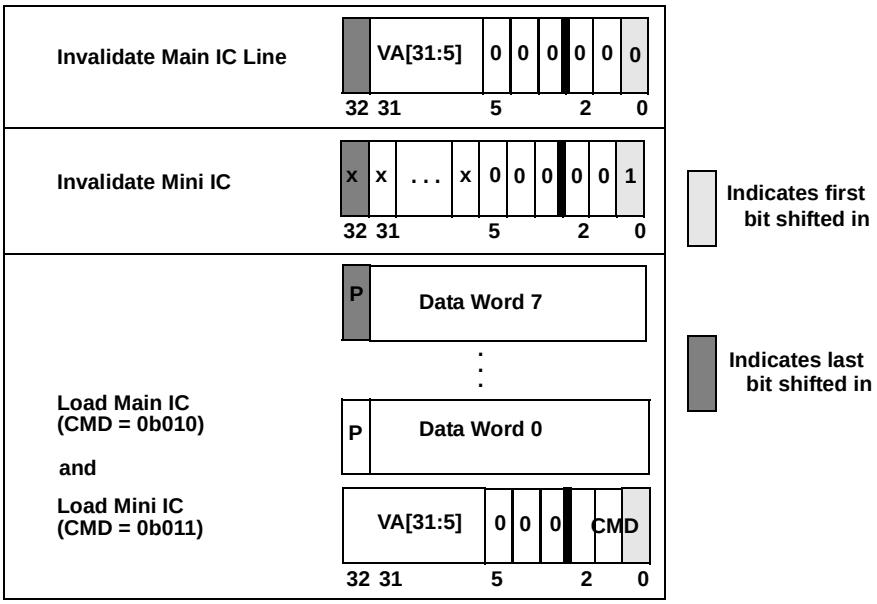
Invalidate Mini IC invalidates the entire mini-instruction cache but does not affect the main instruction cache. It does not require a virtual address or any data arguments.

Load Main IC and Load Mini IC write one line of data (eight ARM* instructions) into the specified instruction cache at the specified virtual address.

The Invalidate Mini IC function does not invalidate the ARM* branch target buffer (BTB) as the CP15 Invalidate IC function does, so software must manually invalidate the BTB where appropriate.

Each cache function is downloaded through JTAG in 33-bit packets. Figure 26-7 shows the packet formats for each of the JTAG cache functions. Invalidate Main IC Line and Invalidate Mini IC each require one packet. Load Main IC and Load Mini IC each require nine packets.

Figure 26-7. Format of LDIC Cache Functions



All packets are 33 bits long. Bits[2:0] of the first packet specify the function to execute. For functions that require an address, bits[32:6] of the first packet specify an 8-word-aligned address (Packet1[32:6] = VA[31:5]). For Load Main IC and Load Mini IC, eight additional data packets specify eight ARM* instructions to be loaded into the target instruction cache. Bits[31:0] of the data packets contain the data to download. Bit[32] of the data packets is the value of the parity for the data in that packet.

As shown in Figure 26-7, the first bit shifted in TDI is bit[0] of the first packet. After each 33-bit packet, the host must place the JTAG state machine into the *Update-DR* state. After the host performs an *Update-DR* and returns the JTAG state machine to the *Shift-DR* state, the host can immediately begin shifting in the next 33-bit packet.

26.4.6.3.4 Loading the Instruction Cache During Reset

Code can be downloaded into the instruction cache through JTAG during a processor reset. This feature is used during software debugging to download the debug handler before starting an application program. The downloaded handler can then intercept the reset vector and perform any necessary setup before the application code starts.

Any code downloaded into the instruction cache through JTAG must be downloaded to addresses in the instruction cache that are not already valid. Downloading code to valid addresses results in unpredictable behavior by the processor. During a processor reset, the instruction cache is typically invalidated, but the following modes are not:

- LDIC mode—Active when the LDIC JTAG instruction is loaded in the JTAG IR. It ensures that the instruction caches are not invalidated during reset.
- Halt mode—Active when the halt mode bit is set in the DCSR. It ensures that the mini-instruction cache is not invalidated. In this mode, the main instruction cache is invalidated by reset.

During a cold reset in which both a processor reset and a JTAG reset occur, the instruction cache is invalidated because the JTAG reset takes the processor out of either of these modes.

During a warm reset in which a JTAG reset does not occur, the instruction cache is not invalidated by reset when either of the modes is active. This situation requires special attention if code is downloaded during the warm reset.

Note: While halt mode is active, a reset invalidates the main instruction cache. Thus, debug handler code downloaded during reset must be loaded into the mini-instruction cache. However, code can be dynamically downloaded into the main instruction cache (see Section 26.4.6.3.5).

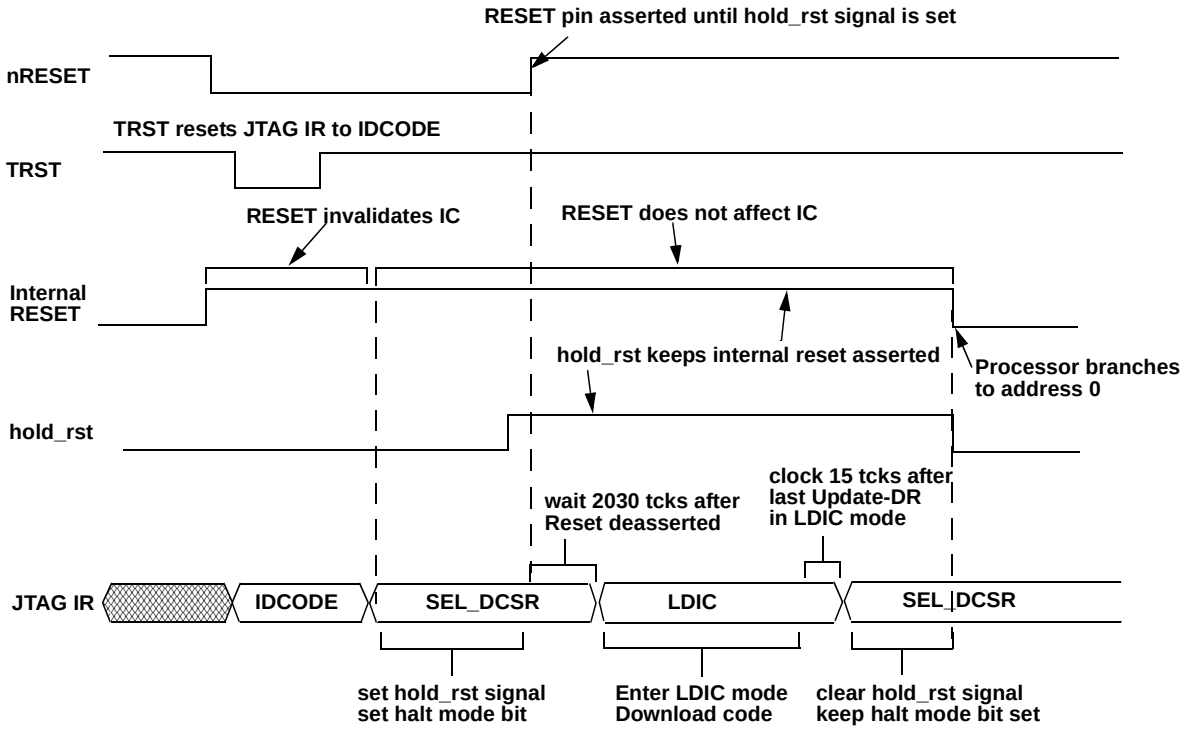
The following sections describe the steps necessary to ensure code is correctly downloaded into the instruction cache.

Loading the IC During Cold Reset for Debug

Figure 26-8 shows the actions necessary to download code into the instruction cache during a cold reset for debug.

Note: HOLD_RST is an internal signal that is set and cleared through JTAG. When the JTAG IR contains the SEL_DCSR instruction, the HOLD_RST signal is set to the value scanned into DBG_SR[1].

Figure 26-8. Code Download During a Cold Reset for Debug



A debug host must complete the following steps to load code into the instruction cache after a cold reset:

1. Assert the Reset and TRST pins. This resets the JTAG IR to IDCODE and invalidates the instruction cache.
2. Load the SEL_DCSR JTAG instruction into JTAG IR and scan in a value to set the halt mode bit in DCSR and to set the hold_rst signal. To set up the reset vector trap before the code is downloaded, scan in a DCSR value that sets the trap reset bit along with the values for the halt mode bit and the hold_rst signal. For details on the SEL_DCSR instruction, refer to Section 26.4.6.1.1.

Note: If the reset vector trap is not set up in step 2, it must be set up in step 8.

3. Deassert the reset pin. The processor is held in reset internally.
4. Wait 2030 TCKs.
5. Load the LDIC JTAG instruction into JTAG IR.
6. Download code into the instruction cache in 33-bit packets, as described in Section 26.4.6.3.3.
7. After the download is complete, wait a minimum of 15 TCKs after the last *Update-DR* state in LDIC mode.
8. If the reset vector trap was not set up in step 2, it must be set up now. The halt mode bit and the hold_rst signal must remain set while a DCSR value that sets the trap reset bit is scanned in.
9. Place the SEL_DCSR JTAG instruction into the JTAG IR and scan in a value to clear the hold_rst signal. The halt-mode bit must remain set to prevent the instruction cache from being invalidated.
10. When hold_rst is cleared, the internal reset is deasserted; the processor executes the reset vector at address 0.

Loading the IC During a Warm Reset for Debug

Loading the instruction cache during a warm reset is a slightly different situation than loading them during a cold reset. For a warm reset, the main issue is whether or not the instruction cache is invalidated by the processor reset.

The following scenarios are possible:

- While reset is asserted, TRST is also asserted.

  The instruction cache is invalidated, so the actions taken to download code are identical to those described in “Loading the IC During Cold Reset for Debug” on page 26-19

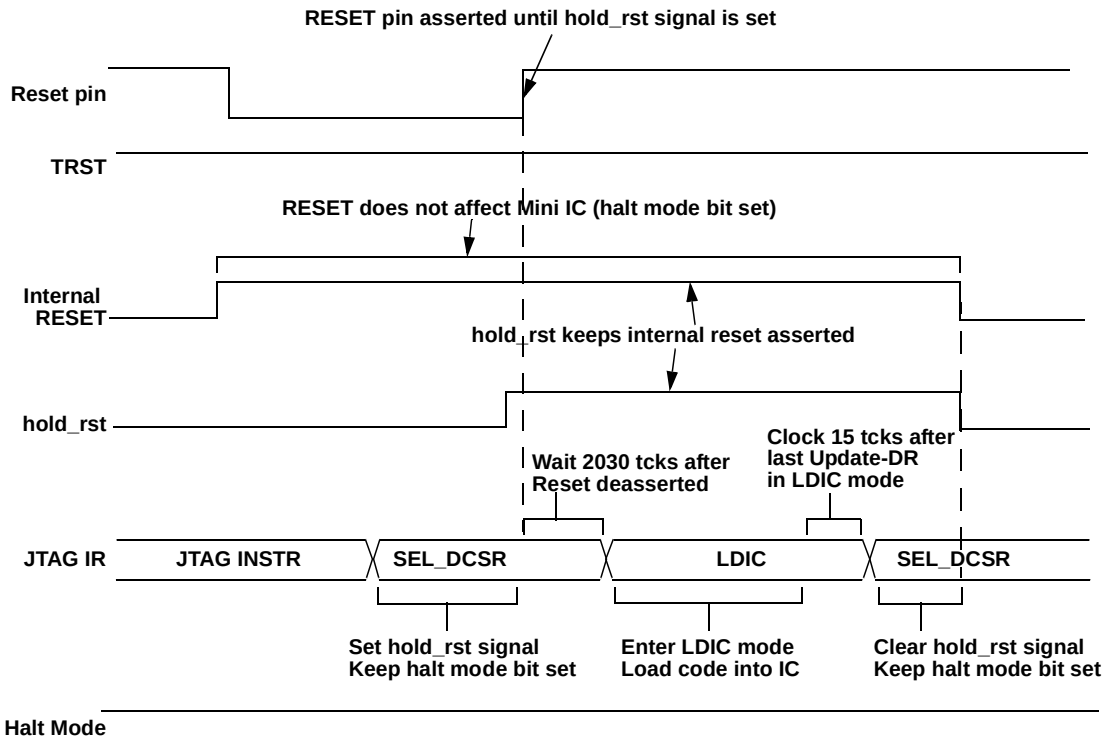
- When reset is asserted, TRST is not asserted and the processor is not in halt mode.

  The instruction cache is also invalidated, so the actions are the same as described in “Loading the IC During Cold Reset for Debug”, starting at step 6.

- When reset is asserted, TRST is not asserted and the processor is in halt mode.

  The mini-instruction cache is not invalidated by reset, because the processor is in halt mode. Figure 26-9 depicts this operation in detail.

Figure 26-9. Code Download During a Warm Reset for Debug



As shown in Figure 26-9, reset does not invalidate the mini-instruction cache because the processor is in halt mode. Because the mini-instruction cache is not invalidated, it may contain valid lines. The host must ensure that code is not downloaded to virtual addresses (VAs) that are already valid in the mini-instruction cache. If code is downloaded to a valid VA, the processor behaves unpredictably.

To ensure that code is not downloaded to a VA that already exists in the mini-instruction cache, use one of the following solutions:

- If it is not necessary to download code into the mini-instruction cache, use the code that was previously downloaded. The mini-instruction cache is not invalidated. Any code previously downloaded into the mini-instruction cache is valid and it is not necessary to download the same code again.
- If it is necessary to download code into the instruction cache:
  a. Assert TRST to halt the device that is awaiting activity on the JTAG interface.
  b. Clear the halt mode bit through JTAG to allow the instruction cache to be invalidated by reset.
  c. Place the LDIC JTAG instruction in the JTAG IR, then proceed with the normal code download, using the Invalidate Main IC Line function before loading each line. This requires that each cache line consist of 10 packets rather than nine, as described in Section 26.4.6.3.3.

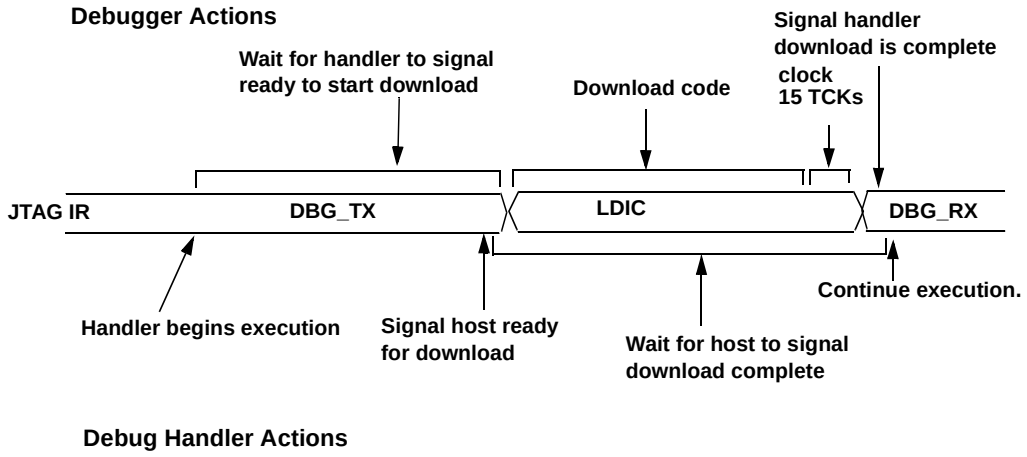
26.4.6.3.5 Dynamically Loading the Instruction Cache After Reset

A debug host can load code into the instruction cache dynamically (on-the-fly). Dynamic loads after reset occur when the host downloads code while the processor is not being reset. Such loads require strict synchronization between the code running on the processor core and the debug host. The guidelines for downloading code during program execution must be followed to ensure that the processor operates properly. The description in this section focuses on using a debug handler running on the core to synchronize with the debug host, but the details apply for any application that is running while code is dynamically downloaded.

To dynamically download code during software debug, a minimal debug handler stub that is responsible for handshaking with the host must reside in the instruction cache. This debug handler stub can be downloaded into the instruction cache during processor reset using the method described in Section 26.4.6.3.4.

Figure 26-10 shows a high-level view of the actions that the host and debug handler take during dynamic code download.

Figure 26-10. Downloading Code into Instruction Cache During Program Execution



The following steps describe the details for downloading code:

1. Because the debug handler is responsible for synchronization during the code download, the handler must be started before the host begins the download. The debug handler starts when the application running on the processor core generates a debug exception or when the host generates an external debug break.
2. While the DBG_TX JTAG instruction is in the JTAG IR (see Section 26.4.6.1.2), the host polls DBG_SR[0] and waits for the debug handler to set the bit.
3. When the debug handler is ready to begin the code download, it writes to TX, which automatically sets DBG_SR[0]. This signals the host that it can start the download. The debug handler then starts to poll TXRXCTRL[RR] and waits for the host to clear bit through the DBG_RX JTAG register, which indicates that the download is complete.
4. The host writes LDIC to the JTAG IR and downloads the code. For each line that is downloaded, the host must invalidate the target line before it can download code to that line. Writing to a line that has not been invalidated causes unpredictable processor operation.

5. When the host completes its download, it must wait a minimum of 15 TCKs, then switch the JTAG IR to DBG_RX and complete the handshaking by scanning in a value that sets DBG_SR[35]. This clears TXRXCTL[RR] and allows the debug handler code to exit the polling loop.
6. After the handler exits the polling loop, it branches to the downloaded code.

Note: The debug handler stub must reside in the instruction cache and execute out of the cache while performing the synchronization. The processor must not perform any code fetches to external memory while code is being downloaded.

Dynamic Code Download Synchronization

The debug handler must contain the following code fragments to implement the synchronization that is used during dynamic code download. The code must be in the same order as shown below:

```
Before the download can start, all outstanding instruction fetches must complete.
The MCR invalidate IC by line function serves as a barrier instruction in
the core. All outstanding instruction fetches are guaranteed to complete before
the next instruction executes.
NOTE1: The actual address specified to invalidate is implementation defined, but
it must not have any harmful effects.
NOTE2: The placement of the invalidate code is implementation defined. The only
requirement is that it must be placed such that by the time the debugger starts
loading the IC, all outstanding instruction fetches have completed.
     mov    r5, address
     mcr    p15, 0, r5, c7, c5, 1

The host waits for the debug handler to signal that it is ready for the
code download. This can be done using the TX register access handshaking
protocol. The host polls the TR bit through JTAG until it is set, then begins
the code download. The following MCR does a write to TX, automatically
setting the TR bit:
NOTE: The value written to TX is implementation defined.
     mcr    p14, 0, r6, c8, c0, 0

The debug handler waits until the download is complete before continuing. The
debugger uses the RX handshaking to signal the debug handler when the download
is complete. The debug handler polls the RR bit until it is set. A debugger write
to RX automatically sets the RR bit, allowing the handler to proceed.
NOTE: The value written to RX by the debugger is implementation defined - it can
be a bogus value signaling the handler to continue, or it can be a target
address for the handler to branch to.
 loop:
     mrc    p14, 0, r15, c14, c0, 0   @ handler waits for signal from debugger
     bpl    loop
     mrc    p14, 0, r0, c8, c0, 0     @ debugger writes target address to RX
     bx     r0
```

In a very simple debug handler stub, the pieces of code (with some handler entry and exit code) may form the complete debug handler that is downloaded during reset. When a debug exception occurs, routines can be downloaded as necessary. This allows the debug handler to be dynamic.

One way to create a more complete debug handler is to download the handler during reset. The debug handler could support some operations, such as read memory and write memory, and dynamically download other operations, such as reading from or writing to a group of coprocessor registers. This method can be used to dynamically download infrequently used debug handler functions, while allowing the more common operations to remain static in the mini-instruction cache.

26.4.6.3.6 Mini-Instruction Cache Overview

The mini-instruction cache is a smaller version of the main instruction cache. Refer to the *Intel XScale® Core Developer's Manual* for details on the main instruction cache. The mini-instruction cache is a 2-Kbyte, two way, set-associative cache. It has 32 sets and each set contains two ways. Each way contains eight words. The cache uses the round-robin replacement policy for lines overloaded from the debugger.

Code other than debug-handler code is never cached in the mini-instruction cache on an instruction fetch. The only way to download code into the mini-instruction cache is through the JTAG LDIC function. Code downloaded into the mini-instruction cache is essentially locked. The debug handler cannot be overwritten by application code running on the processor core, but it is not locked against code downloaded through the JTAG LDIC functions.

Application code can invalidate a line in the mini-instruction cache with a CP15 Invalidate Main IC Line function to an address in the mini-instruction cache. However, a CP15 global Invalidate IC function does not affect the mini-instruction cache.

The mini-instruction cache can be globally invalidated through JTAG with the LDIC Invalidate IC function or with a processor reset when the processor is not in halt or LDIC mode. A single line in the mini-instruction cache can be invalidated through JTAG with the LDIC Invalidate Main IC Line function.

The mini-instruction cache is virtually addressed and the addresses can be remapped by the PID. Because the debug handler executes in SDS, address translation and PID remapping are turned off. Application code makes accesses to the mini-instruction cache with the normal address translation and PID mechanisms.

26.4.7 Trace Buffer

The 256-entry trace buffer provides the ability to capture control flow information to be used for debugging an application. Two modes are supported:

- Fill-once mode: the buffer fills up completely and generates a debug exception. After the debug exception, software must empty the buffer.
- Wraparound mode: the buffer fills up and wraps around until it is disabled. After the buffer is disabled, software must empty the buffer.

26.4.7.1 Trace Buffer CP Registers

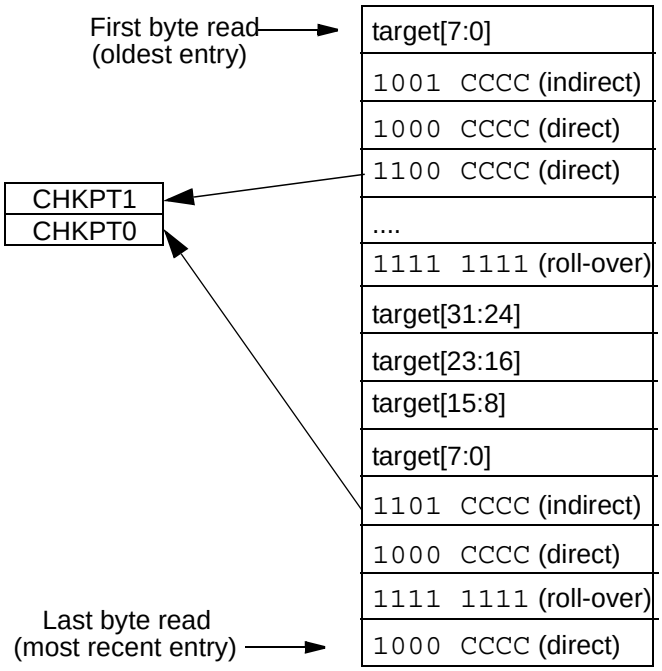
CP14 defines three registers TBREG, CHKPT0, and CHKPT1 (shown in Table 26-15) for use with the trace buffer. The CP14 registers are accessible using MRC, MCR, LDC, and STC (CDP to any CP14 registers causes an undefined instruction trap). The CRn field specifies the number of the register to access. The CRm, opcode_1, and opcode_2 fields are not used and must be cleared to 0.

Any access to the trace buffer registers in user mode causes an undefined instruction exception. Specifying registers that do not exist has unpredictable results.

26.4.7.2 Trace Buffer Usage

The trace buffer is 256 bytes long. The first byte read from the buffer represents the oldest trace history information in the buffer. The last (256th) byte read represents the most recent entry in the buffer. The last byte read from the buffer is always a message byte. This provides the debugger with a starting point for parsing the entries out of the buffer. Because the debugger requires the last byte as a starting point when parsing the buffer, the entire trace buffer (256 bytes) must be read before the buffer can be parsed. Figure 26-11 shows a high-level view of the trace buffer.

Figure 26-11. High Level View of Trace Buffer



The trace buffer must be initialized before each use, including its first use. To initialize the trace buffer, read it entirely. Reading the trace buffer also clears it by setting all entries to 0b0000_0000. This means that when the trace buffer captures a trace, the process of reading the captured trace data also re-initializes the trace buffer for its next use.

The trace buffer can capture a trace up to a processor reset. A processor reset disables the trace buffer, but does not affect the contents. The trace buffer does not capture reset events or debug exceptions.

Because the trace buffer is initialized and cleared before it is used, all entries are initially 0b0000_0000. In fill-once mode, these zeros can identify the first valid entry in the trace buffer. In wraparound mode, the zero entries identify the first valid entry and can be used to determine whether a wraparound occurred.

When the trace buffer is read, the oldest entries are read first. Reading a series of at least five consecutive 0b0000_0000 entries in the oldest entries indicates that the trace buffer has not wrapped around and the first valid entry is the first non-zero value that is read.

Reading four or fewer consecutive 0b0000_0000 values requires the host software to determine whether the zeros are part of the address of an indirect branch message or whether they are part of the 0b0000_0000 that initialized the trace buffer. If the first non-zero message byte is an indirect branch message, the zeros are part of the address because the address is always read before the indirect branch message (see Section 26.4.7.3.4). If the first non-zero entry is any other type of message byte, the zeros indicate that the trace buffer has not wrapped around and the first non-zero entry is the start of the trace.

If the oldest entry from the trace buffer is not a zero, the trace buffer has either wrapped around or is full.

After the trace buffer is read and parsed, the host software must re-create the trace history from oldest trace buffer entry to latest. Re-creating the trace going backwards from the most recent trace buffer entry is not possible in most cases, because the source of a branch message may be impossible to determine after the branch occurs.

In fill-once mode, returns from the debug handler to the application generate an indirect branch message. The address placed in the trace buffer is that of the target application instruction. Using this as a starting point, re-creating a trace going forward is relatively simple.

In wraparound mode, the host software uses the checkpoint registers (see Section 26.5.8) and address bytes from indirect branch entries to re-create the trace going forward. The drawback is that some of the oldest entries in the trace buffer may be untraceable, depending on the location of the earliest checkpoint or indirect branch entry. The best case occurs when the oldest entry in the trace buffer sets a checkpoint, allowing the entire trace buffer to re-create the trace. The worst case occurs when the first checkpoint is in the middle of the trace buffer and no indirect branch messages exist before this checkpoint. In that case, the host software has to start at its known address (the first checkpoint) midway through the buffer and work forward from there.

26.4.7.3 Trace Buffer Entries

Trace buffer entries consist of either one or five bytes. Most entries are 1-byte messages that indicate the type of control flow change. The target address of the control flow change represented by the message byte is either encoded in the message byte (as for exceptions) or can be determined by examining the instruction word (as for direct branches). Indirect branches require five bytes per entry. One byte is the message byte that identifies the indirect branch. The other four bytes make up the target address. The sections that follow describe the trace buffer entries in detail.

26.4.7.3.1 Message Byte

The message byte of an indirect address uses one of two formats: exception or non-exception. Figure 26-12 shows the two message byte formats.

Figure 26-12. Message Byte Formats

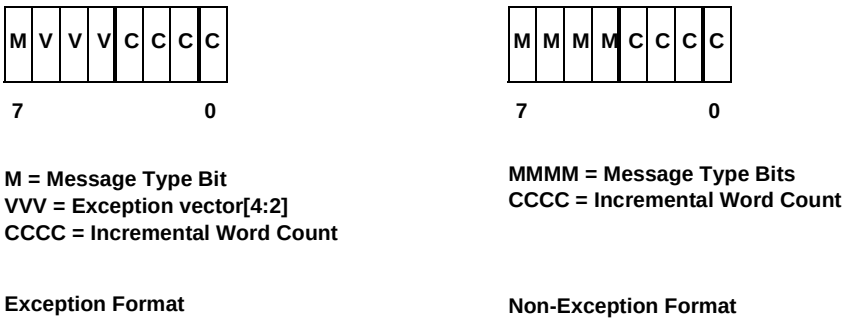


Table 26-5 shows the possible trace messages. All. other byte formats are reserved.

Table 26-5. Message Byte Formats

Message Name	Message Byte Type	Message Byte Format	Number of Address Bytes
Exception	exception	0b0VVV_CCCC	0
Direct Branch[1]	non-exception	0b1000_CCCC	0
Direct Branch with checkpoint[a2]	non-exception	0b1100_CCCC	0
Indirect Branch[3]	non-exception	0b1001_CCCC	4
Indirect Branch with checkpoint[b]	non-exception	0b1101_CCCC	4
Rollover	non-exception	0b1111_1111	0

1. Direct branches include ARM* and THUMB bl, b
2. These message types correspond to trace buffer updates to the checkpoint registers
3. Indirect branches include ARM* ldm, ldr, and dproc to PC; ARM* and THUMB bx, blx and THUMB pop.

26.4.7.3.2 Exception Message Byte

When an exception occurs, an exception message is placed in the trace buffer. In an exception message byte, the message type bit (M) is always 0.

The vector exception (VVV) field specifies bits[4:2] of the vector address (offset from the base of default or relocated vector table). The vector allows the host software to identify the type of exception that occurred.

The incremental word count (CCCC) is the instruction count since the last control flow change (not including the current instruction for undef, SWI, and prefetch abort). The instruction count includes instructions that were executed and conditional instructions that were not executed because the condition of the instruction and the CC flags did not match.

A count value of 0 indicates that no instructions have executed since the last control flow change and the current exception. For example, if a branch is immediately followed by an SWI, a direct branch exception message (for the branch) is sent. The branch exception message is followed by an exception message (for the SWI) in the trace buffer. The count value in the exception message is 0, meaning that no instructions executed after the last control flow change (the branch) and before the current control flow change (the SWI). If an IRQ immediately follows a branch, the count is also 0, because no instructions executed after the branch and before the interrupt.

A count of 0b1111 indicates that 15 instructions executed between the last branch and the exception. This indicates that an exception was either caused by the 16th instruction (if it is an undefined instruction exception, prefetch abort, or SWI) or handled before the 16th instruction executed (for FIQ, IRQ, or data abort).

26.4.7.3.3 Non-Exception Message Byte

Non-exception message bytes are used for direct branches, indirect branches, and rollovers.

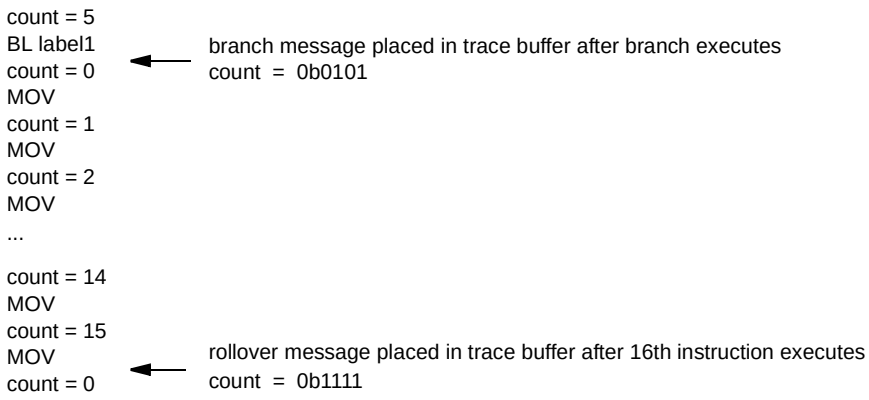
In a non-exception message byte, the 4-bit message type field (MMMM) specifies the type of message (refer to Table 26-5).

The incremental word count (CCCC) is the instruction count since the last control flow change (excluding the current branch). The instruction count includes instructions that were executed and conditional instructions that were not executed because the condition of the instruction did not match the CC flags. In the case of back-to-back branches, the word count is 0, which indicates that no instructions executed after the last branch and before the current one.

A rollover message keeps track of long traces of code that do not have control flow changes. The rollover message means that 16 instructions have executed since the last message byte was written to the trace buffer.

If the incremental counter reaches its maximum value of 15, a rollover message is written to the trace buffer after the next instruction (which is the 16th instruction to execute). This is shown in Figure 26-13. The count in the rollover message is 0b1111, indicating that 15 instructions have executed after the last branch and before the current non-branch instruction that caused the rollover message.

Figure 26-13. Rollover Message Examples

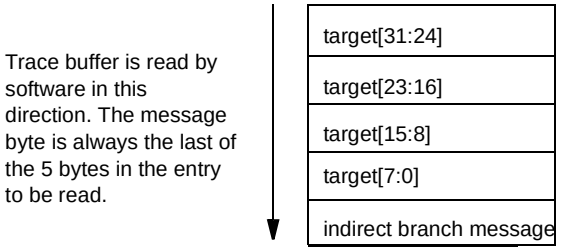


If the sixteenth instruction is a branch (direct or indirect), the appropriate branch message is placed in the trace buffer instead of the rollover message. The incremental counter is still set to 0b1111, meaning that 15 instructions executed between the last branch and the current branch.

26.4.7.3.4 Address Bytes

Only indirect branch entries contain address bytes in addition to the message byte. Indirect branch entries have four address bytes that indicate the target of the indirect branch. When a trace buffer is read, the MSB of the target address is read first. The LSB is the fourth byte read and the indirect branch message byte is the fifth. The byte organization of an indirect branch message is shown in Figure 26-14.

Figure 26-14. Indirect Branch Entry Address Byte Organization



26.4.8 Halt Mode Software Protocol

This section describes the overall debug process in halt mode. It describes how to start and end a debug session and provides details for implementing a debug handler. The code and other documentation that describes additional handler implementation techniques and requirements is intended for manufacturers of debugging tools.

26.4.8.1 Starting a Debug Session

Before starting a debug session in halt mode, the debugger must download code into the instruction cache during reset, using JTAG (see Section 26.4.6.3). The code to be downloaded typically consists of:

- A debug handler
- An override default vector table
- An override relocated vector table, if necessary

While the processor is still in reset, the debugger sets up the DCSR to trap the reset vector. This causes a debug exception to occur when the processor comes out of reset. Execution is redirected to the debug handler, which allows the debugger to perform any necessary initialization. The reset vector trap is the only debug exception that can occur with debug globally disabled (DCSR[RR] clear). Therefore, the debugger must also enable debug before it exits the handler to ensure all that subsequent debug exceptions correctly break to the debug handler.

26.4.8.1.1 Setting Up Override Vector Tables

The override default vector table intercepts the reset vector and branches to the debug handler when a debug exception occurs. If the vector table is relocated, the debug vector is relocated to address 0xFFFF_0000. Thus, an override relocated vector table is required to intercept vector 0xFFFF_0000 and branch to the debug handler.

Both override vector tables also intercept the other debug exceptions, so they must be set up to either branch to a debugger-specific handler or go to the application's handlers.

If an application modifies its vector tables in memory, the debugger cannot set up the override vector table to branch to the application's handlers. The debug handler can work around this problem by reading memory and branching to the appropriate address. The debug handler can be accessed by vector traps or the override vector tables can redirect execution to a debug handler routine that examines memory and branches to the application's handler.

26.4.8.1.2 Placing the Handler in Memory

The debug handler is not required to be placed at a specific predefined address. However, the debug handler's placement is limited by the locations of the override vector tables and the two-way set-associative mini-instruction cache.

In the override vector table, the reset vector must branch to the debug handler using one of two methods:

- A direct branch, which limits the start of the handler code to within 32 Mbytes of the reset vector.
- An indirect branch with a data processing instruction. The data processing instruction creates an address using immediate operands and then branches to the target. An LDR to the PC does cannot be performed because the debugger cannot set up data in memory before it starts the debug handler.

The two way set-associative limitation occurs because the override default and relocated vector tables that are downloaded take up both ways of set 0 (with addresses 0x0 and 0xFFFF_0000). Therefore, debug handler code cannot be downloaded to an address that maps into set 0, because doing so overwrites one of the vector tables. To accomplish this, avoid addresses in which the lower 12 bits are zeros.

The instruction cache two way set limitation is not a problem when the reset vector uses a direct branch, because the branch offset can be adjusted accordingly. However, the two way set limitation does make using indirect branches more complicated. For an indirect branch, the reset vector requires multiple data processing instructions to create the target address and branch to it.

One way to deal with an indirect branch is to set up vector traps on the non-reset exception vectors. These vector locations can then be used to extend the reset vector.

Another solution is to use the reset vector to perform a direct branch to intermediate code. The intermediate code can then use several instructions to create the debug handler start address and branch to it. This requires another line in the mini-instruction cache, because the intermediate code must also be downloaded. To use this solution, the debugger handler must have a well-planned layout so it can avoid overwriting a line of debug handler code with the intermediate code, or vice versa.

For the indirect branch cases, a temporary scratch register can hold intermediate values while computing the final target address. DBG_r13 can be used as such a scratch register. See Section 26.4.8.3.1, "Debug Handler Restrictions" for restrictions on DBG_r13 usage.

26.4.8.2 Implementing a Debug Handler

The debugger uses the debug handler to examine or modify processor state by sending commands and reading data through JTAG. The software interface between the debugger and debug handler is specific to the debugger implementation.

26.4.8.3 Debug Handler Entry

When the debugger requests an external debug break or is waiting for an internal break, it polls TXRXCTRL[TR] through JTAG to determine when the processor enters debug mode. The debug handler entry code must write to TX to signal the debugger that the processor has entered debug mode. The write to the TX register sets TXRXCTRL[TR], which signals to the host that a debug exception has occurred and the processor has entered debug mode. The value of the data written to the TX register is implementation-defined (such as debug break message and contents of register to save on host).

26.4.8.3.1 Debug Handler Restrictions

The debug handler executes in debug mode, which is similar to other privileged processor modes. The following list shows the restrictions on the debug handler code and the differences between debug mode and the other privileged processor modes:

- The processor is in SDS after a debug exception. Because it is in SDS, it has special functionality as described in Section 26.4.1.1.
- Address translation and PID remapping are disabled for instruction accesses (as defined in SDS), but data accesses use the normal address translation and PID remapping mechanisms.
- Debug mode does not have a dedicated stack pointer. DBG_r13 is not a general-purpose register and its contents are unpredictable and cannot be relied upon across any instructions or exceptions. It can be used, by data processing (non-RRX) and MCR/MRC instructions, as a temporary scratch register.
- The following instructions must not be executed in debug mode because they result in unpredictable behavior:
  - LDM
  - LDR with Rd = PC
  - LDR with RRX addressing mode
  - SWP
  - LDC
  - STC
- The debugger handler executes in debug mode and can be switched to other modes to access banked registers. The handler must not enter user mode, because doing so causes unpredictable behavior. Any user-mode registers that must be accessed must be accessed in system mode.

26.4.8.3.2 Dynamic Debug Handler

In the processor core, the debug handler and override vector tables may reside in the 2-Kbyte mini-instruction cache, separate from the main instruction cache. A static debug handler is downloaded during reset. The code downloaded during reset is the base handler code that is necessary to perform common operations such as handler entry/exit, parsing commands from the debugger, reads and writes to ARM* registers, and reads and writes to memory.

Some functions require large amounts of code or are not be used very often. As long as the mini-instruction cache has space, these functions can be downloaded as part of the static debug handler. However, if space is limited, the debug handler also has a dynamic capability that allows a function to be downloaded when it is needed. A dynamic debug handler can minimize the amount of space it requires by downloading a given function, then overwriting it when another function is required.

Because dynamic functions often perform common routines, such as the polling routines for reading RX or writing TX, the debug handler can reduce the space required by the dynamic functions by defining a set of registers that contain the addresses of the most commonly used routines. The dynamic functions can then access these routines using indirect branches (BLX). This helps to reduce the amount of space the dynamic functions require because common routines are not replicated in each dynamic function.

A dynamic debug handler can be implemented in one of three methods: using the mini-instruction cache, using the main instruction cache, or using external memory. For all three methods, the downloaded code executes in the context of the debug handler. The processor is in SDS, so the special functionality related to SDS applies. Each method has limitations and advantages. Section 26.4.6.3.5 describes how to dynamically load the mini or main instruction cache.

Using the Mini-Instruction Cache

A static debug handler can support a command that has a function that is dynamically mapped to it. A dynamic command does not have a specific function associated with it until the debugger downloads a function into the mini-instruction cache. When the debugger sends the dynamic command to the handler, the new function can be downloaded or the previously downloaded function can be used.

The debug handler can support multiple dynamic commands with each command mapped to a different dynamic function. It can also support a single dynamic command that branches to one of several downloaded dynamic functions, as determined by a parameter passed by the debugger.

Debug handlers that allow code to be dynamically downloaded into the mini-instruction cache must be written carefully to avoid overwriting a critical piece of debug handler code. Dynamic code is downloaded to the way that is pointed to by the round-robin pointer. Thus, critical debug handler code can be overwritten if the pointer does not select the proper way.

To avoid this problem, the debug handler must be written to avoid placing critical code in either of the two ways in a set that is intended for dynamic code download. Avoiding such placement allows code to be downloaded into either way and ensures that the only code that is overwritten is the previously downloaded dynamic function. To use this method, some space in the mini-instruction cache must be allocated for dynamic downloads. This limits the space available for the static debug handler, and the remaining space may be too small for a larger dynamic function.

After a dynamic function is downloaded, it essentially becomes part of the debug handler. If it is written in the mini-instruction cache, it cannot be overwritten by application code. It remains in the cache until it is replaced by another dynamic function or its lines it are invalidated.

Using the Main Instruction Cache

The steps required to download a dynamic function into the main instruction cache are similar to those for downloading it into the mini-instruction cache. Using the main instruction cache offers some advantages.

Using the main instruction cache eliminates the problem of inadvertently overwriting static debug handler code by writing to the wrong way of a set, because the instruction caches are separate. In the main instruction cache, debug handler code does not have to be specially mapped out to avoid this problem. Another advantage is that dynamic functions are not limited to an allocated size, as they are in the mini-instruction cache.

Dynamic functions downloaded into the main instruction cache can be downloaded anywhere in the address space. The debugger specifies the location of the dynamic function by writing the address to RX when it signals to the handler to continue. The debug handler then performs a branch-and-link to that address.

If a dynamic function is already downloaded in the main instruction cache, the debugger immediately downloads the address and signals the handler to continue.

The static debug handler only has to support one dynamic function command. Multiple dynamic functions can be downloaded to different addresses and accessed by the debugger using a function's address to specify which one to execute.

Because the dynamic function is downloaded into the main instruction cache, it can overwrite valid application code or be overwritten by application code. The only time that a dynamic function is guaranteed to be in the cache is from the time it is downloaded to the time the debug handler returns to the application or the debugger overwrites the function.

Using External Memory

Dynamic functions can also be downloaded to external memory. In some cases, they already exist in external memory. The debugger can download to external memory using the write-memory commands. After the debugger downloads a dynamic function to external memory, it executes the function using the function's address. This method has many of the same advantages as downloading into the main instruction cache.

Using external memory to hold dynamic functions is always slower than downloading directly into an instruction cache. Another problem with using external memory is that the application may write to the memory address that contains a function. If the software design can ensure that the application does not modify downloaded dynamic functions, the debug handler can save the time it takes to download the code again. If the software design cannot ensure that functions are not overwritten, the debugger must download a dynamic function each time it is used.

26.4.8.3.3 High-Speed Download

The PXA27x processor provides special debug hardware to support a high-speed download mode that increases the performance of downloads to system memory (when compared to writing a block of memory using the standard handshaking).

The basic assumption is that the debug handler can read the data the debugger sends and write it to memory before the debugger sends more data. Thus, in the time required for the debugger to scan in the next data word and perform an *Update-DR*, the handler is already in a polling loop, waiting for it. The debugger does not have to poll RR to see whether the handler has read the previous data because it assumes the previous data has been consumed and immediately starts scanning in the next data word.

The assumption that the debug handler can read the data the debugger sends and write it to memory before the debugger sends more data fails when the write to memory stalls long enough to allow the debugger to catch up. If the write to memory stalls that long, a download with normal handshaking can be used. A high-speed download can still be used, but extra TCKs must be added in the *Pause-DR* state to allow more time for the store to complete.

The hardware support for high-speed download includes the download bit (RXTXCTRL[D]) and the overflow flag (RXTXCTRL[OV]).

The download bit acts as a branch flag that signals the handler to continue with the download. This removes the need for a counter in the debug handler.

The overflow flag indicates that the debugger attempted to download the next word before the debug handler read the previous word.

For more details on the download bit, overflow flag, and high-speed download, see Table 26-6.

The following example code shows how the download bit and overflow flag are used in the debug handler:

```
hs_write_word_loop:
hs_write_overflow:
    bl      read_RX                 @ read data word from host

    @@ read TXRXCTRL into the CCs
    mrc     p14, 0, r15, c14, c0, 0
    bcc     hs_write_done       @ if D bit clear, download complete, exit loop.
    beq     hs_write_overflow   @ if overflow detected, loop until host clears D bit

    str     r0, [r6], #4        @ store only if there is no overflow.

    b       hs_write_word_loop      @ get next data word

hs_write_done:
    @@ after the loop, if the overflow flag was set, return error message to host
    moveq   r0, #OVERFLOW_RESPONSE
    beq     send_response
    b       write_common_exit
```

26.4.8.4 Ending a Debug Session

Before it ends a debug session, the debugger must take the following actions:

1. Clear the DCSR.
   a. Disable debug.
   b. Exit halt mode.
   c. Clear all vector traps.
   d. Disable the trace buffer.
2. Turn off all breakpoints.
3. Invalidate the mini-instruction cache if it has been altered.
4. Invalidate the main instruction cache if it has been altered.
5. Invalidate the BTB.

These actions ensure that the application program executes correctly after the debugger has been disconnected.

26.4.9 Software Debug Notes

- Trace buffer message count value on data aborts:

  LDR to non-PC that aborts is counted in the exception message. An LDR to the PC that aborts is not counted as an exception message.

- Data abort-generation in SDS:

  Avoid code that could generate precise data aborts.

  If precise data aborts cannot be avoided, the handler must be written so that a memory access is followed by one NOP. In addition, certain memory operations must be avoided: LDM, STM, STRD, LDC, SWP.

- Data abort on SDS:

  When write-back is on for a memory access that causes a data abort, the base register is updated with the write-back value. This is inconsistent with normal (non-SDS) behavior in which the base remains unchanged if write-back is on and a data abort occurs.

- Trace buffer wraps around and loses data in halt mode when configured for fill-once mode:

  Data from the trace buffer can overflow and be lost in fill-once mode when the processor is in halt mode. When the trace buffer fills up, it has space for one indirect branch message (5 bytes) and one exception message (1 byte).

  If the trace buffer fills up with an indirect branch message and generates a trace-buffer full break at the same time that a data abort occurs, the data abort has higher priority and the processor services the data abort handler first. The data abort is placed into the trace buffer without losing any data.

  However, if another imprecise data abort is detected at the start of the data abort handler, it has a higher priority than the trace-buffer full break, so the processor returns to the data abort handler. The second data abort is also written into the trace buffer. This causes the trace buffer to wrap around and lose one trace buffer entry (the oldest entry). Additional trace buffer entries can be lost if imprecise data aborts continue to be detected before the processor can handle the trace buffer full break and turn off the trace buffer).

  The trace buffer overflow problem can be avoided by enabling vector traps on data aborts.

- TXRXCTRL[OV] (the overflow flag) is not set during high-speed downloads if the handler reads the RX register at the same time the debugger writes to it.

  If the debugger writes to RX at the same time the handler reads from it, the handler read returns the newly written data and the previous data is lost. However, the overflow flag is not set, so the debugger is unaware that the download was not successful.

26.5 Register Descriptions

This section describes the control, status, and data registers that configure the PXA27x processor debug unit.

26.5.1 Transmit/Receive Control Register (TXRXCTRL)

Communication between the debug handler and debugger are controlled through handshaking bits that ensure the debugger and debug handler make synchronized accesses to TX and RX. The debugger side of the handshaking is accessed through the DBG_TX (Section 26.4.6.1.2) and DBG_RX (Section 26.4.6.1.3) JTAG Data registers, depending on the direction of the data transfer.

The debug handler uses separate handshaking bits in TXRXCTRL register to access TX and RX. The TXRXCTRL register contains two other bits that support high-speed downloads. One bit indicates an overflow condition that occurs when the debugger attempts to write the RX register before the debug handler has read the previous data written to RX. The other bit is used by the debug handler as a branch flag during high-speed download.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 26-6. TXRXCTRL Bit Definitions (Sheet 1 of 3)

	CP14, CRn14, CRm0				TXRXCTRL																				Debug							
User Settings																																
Bit	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
	RR	OV	D	TR	reserved																											
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

Bits	Access	Name	Description
31	**Software:** R only W ignored **JTAG:** W only	RR	RX Register Ready Used by the debugger and the debug handler to synchronize accesses to RX. Normally, the debugger and debug handler use a handshaking scheme that requires both sides to poll the RR bit. To support higher download performance for large amounts of data, a high-speed download handshaking scheme can be used. In this scheme, the debug handler polls the RR bit before it accesses the RX register and the debugger continuously downloads data. 0 = RX register is not ready 1 = RX register is ready

Table 26-6. TXRXCTRL Bit Definitions (Sheet 2 of 3)

CP14, CRn14, CRm0 TXRXCTRL Debug

User Settings																															
Bit	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
	RR	OV	D	TR	reserved																											
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

Bits	Access	Name	Description
30	**Software:** R/W	OV	RX Overflow This sticky flag bit is set when the debugger writes to the RX register while the RR bit is set. This flag is used during high-speed downloads to indicate that data has been lost. The assumption during high-speed downloads is that the time it takes for the debugger to shift in the next data word is greater than the time necessary for the debug handler to process the previous data word. So, before the debugger shifts in the next data word, the handler starts to poll for that data. However, if the handler stalls long enough that the handler is still processing the previous data when the debugger completes shifting in the next data word, an overflow condition occurs and the OV bit is set. When set, the overflow flag remains set until cleared by a write to TXRXCTRL with an MCR. After the debugger completes the download, it can examine the OV bit to determine if an overflow occurred. The debug handler software is responsible for saving the address of the last valid store before the overflow occurred. 0 = RX overflow has not occurred 1 = RX overflow has occurred.
29	**Software:** R only W ignored **JTAG:** W only	D	Download The value of the high-speed Download flag is set by the debugger through JTAG. This flag is asserted during high-speed download to replace a loop counter. Using the download flag, the debug handler loops until the debugger clears the flag. Therefore, when doing a high-speed download, for each data word downloaded, the debugger must set the D bit. On completing the download the debugger clears the D bit releasing the debug handler to take the data. The download flag is useful when an overflow occurs. If a loop counter is used and an overflow occurs, the debug handler cannot determine how many data words overflowed. Therefore, the debug handler counter can be out of sync with the debugger. The debugger can finish downloading the data, but the debug handler counter can indicate there is more data to be downloaded. This results in unpredictable behavior in the debug handler. 0 = Normal RX 1 = High-speed RX

Table 26-6. TXRXCTRL Bit Definitions (Sheet 3 of 3)

	CP14, CRn14, CRm0				TXRXCTRL																				Debug							
User Settings																																
Bit	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
	RR	OV	D	TR	reserved																											
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

Bits	Access	Name	Description
28	**Software:** R only W ignored **JTAG:** W only	TR	TX Register Ready Used by the debugger and debug handler to synchronize access to the TX register. The debugger and debug handler must poll the TR bit before accessing the TX register. The following handshaking schemes are used to access the TX register. Debugger actions: • The debugger is expecting data from the debug handler. • Before reading data from the TX register, the debugger polls the TR bit through JTAG until the bit is set. While the debugger polls TR, it must scan out the TR bit and the TX register data. • Reading 0b1 from the TR bit indicates that the TX data is valid • Scanning out data when the TR bit is set automatically clears TR. Debug handler actions: • The debug handler has data to send to the debugger (in response to a previous request). • The debug handler polls the TR bit to determine when the TX register is empty (any previous data has been read out by the debugger). The handler polls the TR bit until it is clear. • When the TR bit is clear, the debug handler writes new data to the TX register. The write operation automatically sets the TR bit. 0 = The TX register is not ready. 1 = The TX register is ready.
27:0	—	—	reserved

26.5.2 Debug Control and Status Register (DCSR)

DCSR is the main control register for the debug unit. Table 26-7 shows the format of the register. The DCSR register can be accessed in privileged modes by software running on the core and by a debugger through the JTAG interface. Refer to Section 26.4.6.1.1 for details about accessing DCSR through JTAG. For the trap bits in Table 26-7, setting the bits enables the trap behavior, and clearing them disables the trap.

When a trap bit is set, it acts as if an instruction breakpoint was set up on the corresponding exception vector. A debug exception is generated before the instruction in the exception vector executes.

To set up a non-reset vector trap:

- Software running on the processor core must set DCSR[GE].
- The debugger must set DCSR[H] and the appropriate vector trap bit through JTAG.

To set up a reset vector trap, the debugger sets DCSR[H] and reset vector trap bit through JTAG. DCSR[GE] does not affect the reset vector trap. A reset vector trap can be set up before or during a processor reset. When processor reset is deasserted, a debug exception occurs before the instruction in the reset vector is executed.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 26-7. DCSR Bit Definitions (Sheet 1 of 3)

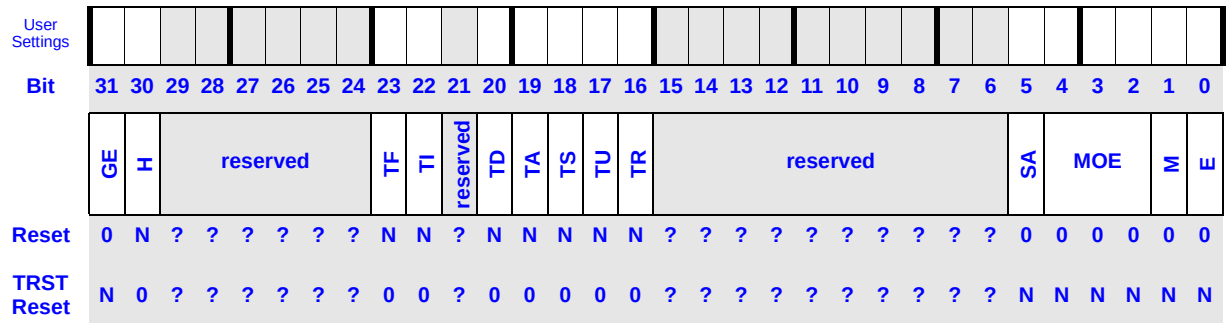
CP14, CRn10, CRm0 — DCSR — Debug

	Bit	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
User Settings		GE	H	reserved						TF	TI	reserved	TD	TA	TS	TU	TR	reserved										SA	MOE			M	E
	Reset	0	N	?	?	?	?	?	?	N	N	?	N	N	N	N	N	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0
	TRST Reset	N	0	?	?	?	?	?	?	0	0	?	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	N	N	N	N	N	N

Bits	Access	Name	Description
31	**Software:** R/W **JTAG:** R only	GE	Global Enable Enables and disables all debug functionality except the reset vector trap. After a processor reset, this bit is clear, so debug functionality is disabled. When debug functionality is disabled, the **BKPT** instruction becomes an **NOP** and external debug breaks, hardware breakpoints, and non-reset vector traps are ignored. 0 = Disables all debug functionality 1 = Enables all debug functionality
30	**Software:** R only **JTAG:** R/W	H	Halt Mode Configures the debug unit for halt or monitor mode. 0 = Monitor mode 1 = Halt mode
29:24	—	—	reserved
23	**Software:** R only **JTAG:** R/W	TF	Trap FIQ The vector trap bits allow instruction breakpoints to be set on exception vectors without using any of the breakpoint registers. 0 = Disables instruction breakpoint on FIQ 1 = Enables instruction breakpoint on FIQ
22	**Software:** R only **JTAG:** R/W	TI	Trap IRQ 0 = Disables instruction breakpoint on IRQ 1 = Enables instruction breakpoint on IRQ
21	—	—	reserved
20	**Software:** R only **JTAG:** R/W	TD	Trap Data abort: 0 = Disables instruction breakpoint on data abort 1 = Enables instruction breakpoint on data abort
19	**Software:** R only **JTAG:** R/W	TA	Trap Prefetch Abort: 0 = Disables instruction breakpoint on prefetch abort 1 = Enables instruction breakpoint on prefetch abort

Table 26-7. DCSR Bit Definitions (Sheet 2 of 3)

CP14, CRn10, CRm0 — DCSR — Debug



Bits	Access	Name	Description
18	**Software:** R only **JTAG:** R/W	TS	Trap Software Interrupt 0 = Disables instruction breakpoint on software interrupt 1 = Enables instruction breakpoint on software interrupt
17	**Software** Read-only **JTAG** Read / Write	TU	Trap Undefined Instruction: details in the bit-field description for DCSR[TF]. 0 = Disables instruction breakpoint on undefined Instruction 1 = Enables instruction breakpoint on undefined Instruction
16	**Software:** R only **JTAG:** R/W	TR	Trap Reset 0 = Disables instruction breakpoint on reset 1 = Enables instruction breakpoint on reset
15:6	—	—	reserved
5	**Software:** R/W **JTAG:** R only	SA	Sticky Abort This bit is valid in halt mode only. It indicates a data abort occurred in SDS (see Section 26.4.1.1). Because SSDS disables all exceptions, a data abort exception does not occur. However, the processor sets SA to indicate that a data abort was detected. The debugger can use this bit to determine if a data abort was detected during the SDS. The sticky abort bit must be cleared by the debug handler before exiting the debug handler.

Table 26-7. DCSR Bit Definitions (Sheet 3 of 3)

CP14, CRn10, CRm0 — DCSR — Debug

Bit	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
	GE	H	reserved						TF	TI	reserved	TD	TA	TS	TU	TR	reserved										SA	MOE			M	E
Reset	0	N	?	?	?	?	?	?	N	N	?	N	N	N	N	N	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0
TRST Reset	N	0	?	?	?	?	?	?	0	0	?	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	N	N	N	N	N	N

Bits	Access	Name	Description
4:2	**Software:** R/W **JTAG:** R only	MOE	Method Of Entry These bits specify the cause of the most recent debug exception. When multiple exceptions occur in parallel, the processor places the highest priority exception (based on the priorities in Table 26-1) in the MOE field. 0b000—processor reset 0b001—instruction breakpoint hit 0b010—data breakpoint hit 0b011—bkpt instruction executed 0b100—external debug event asserted 0b101—vector trap occurred 0b110—trace-buffer full break 0b111—reserved
1	**Software:** R/W **JTAG:** R only	M	Trace Buffer Mode This bit selects one of two trace buffer modes: 0 = Wraparound mode—the trace buffer fills up and wraps around until a debug exception occurs. 1 = Fill-once mode—the trace buffer automatically generates a debug exception (trace buffer full break) when it becomes full and stops.
0	**Software:** R/W **JTAG:** R only	E	Trace Buffer Enable Enables and disables the trace buffer. Both DCSR[E] and DCSR[GE] must be set to enable the trace buffer. The processor automatically clears this bit to disable the trace buffer when a debug exception occurs. For more details on the trace buffer, refer to Section 26.4.7. 0 = Disabled 1 = Enabled

26.5.3 Data Breakpoint Controls Register (DBCON)

The DBCON register controls the functionality of DBR1 and the enables for both DBR1 and DBR2. DBCON also controls what type of memory access to break on. Table 26.5.3 shows the format of the DBCON register.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 26-8. DBCON Bit Definitions

	CP15, CRn14, CRm4												DBCON												Debug							
User Settings																																
Bit	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																							M	reserved				E1		E0	
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

Bits	Access	Name	Description
31:9	—	—	reserved
8	R/W	M	DBR1 Mode 0 = DBR1 data address breakpoint 1 = DBR1 data address mask
7:4	—	—	reserved
3:2	R/W	E1	The DBR1 Enable bit functions only when DBCON[M] is clear. When DBCON[M] is set, this field has no effect. 0b00—DBR1 disabled 0b01—DBR1 enabled, store only 0b10—DBR1 enabled, any data access, load or store 0b11—DBR1 enabled, load only
1:0	R/W	E0	DBR0 Enable 0b00—DBR0 disabled 0b01—DBR0 enabled, store only 0b10—DBR0 enabled, any data access, load or store 0b11—DBR0 enabled, load only

26.5.4 Instruction Breakpoint Address and Control Register (IBCRx)

The PXA27x processor debug architecture defines two instruction breakpoint address and control registers (IBCR0, IBCR1). The format of the registers is shown in Table 26-9.

Table 26-9. IBCRx Bit Definitions

	CP15, CRn14, CRm8 CP15, CRn14, CRm9												IBCR0 IBCR1												Debug							
User Settings																																
Bit	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
	IBCRx																														E	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:1	Read / Write	IBCRx	Instruction Breakpoint MVA In ARM* mode, IBCRx[1] is ignored.
0	Read / Write	E	IBCRx Enable 0 = Breakpoint disabled 1 = Breakpoint enabled

26.5.5 Data Breakpoint Register (DBRx)

The debug architecture defines two data breakpoint registers (DBR0, DBR1). The format of the registers is shown in Table 26-10.

Table 26-10. DBRx Bit Definitions

	CP15, CRn14, CRm0 CP15, CRn14, CRm3												DBR0 DBR1												Debug			
User Settings																												
Bit	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
	DBRx																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:0	R/W	DBRx	DBR0: Data breakpoint MVA only DBR1: Data address mask or data breakpoint MVA

26.5.6 Transmit Register (TX)

The TX register is the debug handler transmit buffer. The debug handler sends data to the debugger through this register.

Table 26-11. TX Bit Definitions

	CP14, CRn8, CRm0												TX												Debug			
User Settings																												
Bit	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
	TX																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:0	**Software** Read / Write **JTAG** Read-only	TX	Transmit Data The debug handler writes data to send to debugger. Because the TX register is accessed by the debug handler (with MCR/MRC) and the debugger (through JTAG), handshaking is required to prevent the debug handler from writing new data before the debugger reads the previous data.The handshaking is described in the TXRXCTRL[TR] bit description in Table 26-6.

26.5.7 Receive Register (RX)

The RX register is the receive buffer used by the debug handler to receive data sent by the debugger through the JTAG interface.

Table 26-12. RX Bit Definitions

	CP14, CRn9, CRm0												RX												Debug							
User Settings																																
Bit	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
	RX																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:0	**Software:** R only **JTAG**: W only	**RX**	Software reads the RX register to receive data and commands from the debugger. Because the RX register is accessed by the debug handler (with MRC) and the debugger (through JTAG), handshaking is required to prevent the debugger from writing new data to the register before the debug handler reads the previous data out. The handshaking is described in the TXRXCNTL[RR] bit field in Table 26-6.

26.5.8 Checkpoint Registers (CHKPTx)

When the debugger reconstructs a trace history, it is required to start at the oldest trace buffer entry and construct a trace going forward. In fill-once mode and wraparound mode, when the buffer does not wrap around, the trace can be reconstructed by starting from the point in the code at which the trace buffer was first enabled.

Complications occur in wraparound mode when the trace buffer wraps around at least once. If this happens, the debugger takes a snapshot of the last *N* control flow changes in the program, where *N* is less than or equal to the size of the buffer. The debugger does not know the starting address of the oldest entry read from the trace buffer. The checkpoint registers provide reference addresses to help reduce this problem.

The two checkpoint registers (CHKPT0, CHKPT1) in the processor debug unit provide the debugger with two reference addresses to use for reconstructing the trace history.

When the trace buffer is enabled, reading or writing to either checkpoint register has unpredictable results. When the trace buffer is disabled, writing to a checkpoint register sets the register to the value written and reading the checkpoint registers returns the value of the register.

Normally, the checkpoint registers hold the target addresses of specific entries in the trace buffer. Direct and indirect entries written into the trace buffer are marked as checkpoints with the corresponding target address automatically written into the checkpoint registers. Exception and rollover messages never use the checkpoint registers. When a checkpoint register value is updated, the processor sets bit[6] of the message byte in the trace buffer to indicate that the update occurred (see Table 26-5, “Message Byte Formats”).

When the trace buffer contains only one entry that relates to a checkpoint, the corresponding checkpoint register is CHKPT0. When the trace buffer wraps around, two entries are typically marked as related to checkpoint register values. The entries marked as related are usually about

half the length of the trace buffer apart. This is always the case as the messages in the trace buffer vary in length. With two entries, the first (oldest) entry that set a checkpoint in the trace buffer corresponds to CHKPT1 and the second entry that set a checkpoint corresponds to CHKPT0.

Although the checkpoint registers are provided for wraparound mode, they are valid in fill-once mode.

Table 26-13. CHKPTx Bit Definitions

	CP14, CRn12, CRm0 CP14, CRn13, CRm0								CHKPT0 CHKPT1								Debug											
User Settings																												
Bit	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
	CHKPTx																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:0	R/W	CHKPTx	CHKPTx is the target address for corresponding entry in the trace buffer.

26.5.9 Trace Buffer Register (TBREG)

The trace buffer is read through TBREG with MRC and MCR. Software can read the trace buffer only when the buffer is disabled. Reading the trace buffer returns the oldest byte in the trace buffer in the least significant byte of TBREG. The byte is either a message byte or one byte of the 32-bit address associated with an indirect branch message.Table 26-14 shows the format of the trace buffer register.

Note: Reading the trace buffer while it is enabled causes unpredictable trace-buffer behavior. Writes to the trace buffer produce unpredictable results.

Table 26-14. TBREG Bit Definitions

	CP14, CRn11, CRm0								TBREG								Debug											
User Settings																												
Bit	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																								Data							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R†	Data	Message or Address Byte
† Read these bits only when the trace buffer is disabled.			

26.6 Register Summary

Coprocessor 15 registers are accessible with MRC and MCR. CRn and CRm specify the register to access. The opcode_1 and opcode_2 fields are not used and must be cleared to 0. Software access to debug registers must be performed in privileged mode. user-mode accesses generate an undefined instruction exception. Specifying registers that do not exist has unpredictable results.

Coprocessor 14 registers are accessible with MRC, MCR, LDC, and STC (CDP to any CP14 registers causes an undefined instruction trap). The CRn field specifies the number of the register to access. The CRm, opcode_1, and opcode_2 fields are not used and must be cleared to zero.

The RX register, the TX register, and certain bits in DCSR can be accessed by a debugger through the JTAG interface. This allows an external debugger to have access to the internal state of the processor. For details of which bits can be accessed, see Table 26-6, Table 26-2, and Table 26-7.

Table 26-15. Coprocessor Debug and Trace Register Summary

CRn	CRm	Name	Description	Page
Coprocessor 15				
14	8	IBCR0	Instruction Breakpoint register 0	26-42
14	9	IBCR1	Instruction Breakpoint register 1	26-42
14	0	DBR0	Data Breakpoint register 0	26-43
14	3	DBR1	Data Breakpoint register 1	26-43
14	4	DBCON	Data Breakpoint Control register	26-41
Coprocessor 14				
8	0	TX	TX register	26-43
9	0	RX	RX register	26-44
10	0	DCSR	Debug Control and Status register	26-39
11	0	TBREG	Trace Buffer register	26-45
12	0	CHKPT0	Checkpoint register 0	26-45
13	0	CHKPT1	Checkpoint register 1	26-45
14	0	TXRXCTRL	TXRX Control register	26-36

Quick Capture Interface 27

This chapter describes the operation and signals of the quick capture interface, which provides a connection between the processor and a camera image sensor. The quick capture interface was designed to work primarily with CMOS-type image sensors. However, it may be possible to connect some CCD-type image sensors to the PXA27x processor, depending on a specific CCD sensor's interface requirements.

27.1 Overview

The quick capture interface acquires data and control signals from the image sensor and performs the appropriate data formatting prior to routing the data to memory using direct memory access (DMA). A broad range of interface and signaling options provides direct connection. The image sensor can provide raw data through a variety of parallel and serial formats. For sensors that provide pre-processing capabilities, the quick capture interface supports several formats for RGB and YCbCr color space. The interface supports the International Telecommunication Union *Recommendation ITU-R BT.656-4*[1] Start-of-Active-Video (SAV) and End-of-Active-Video (EAV) embedded synchronization sequences for four- and eight-bit configurations.

27.2 Features

The quick capture interface provides the following features:

- Parallel interface support for 8, 9, and 10 bits
- Serial interface support for 4-bit and 5-bit device connections
- Support for ITU-R BT.656-4 SAV and EAV embedded synchronization
- Pre-processed capture modes:
  - RGB 8:8:8, RGBT 8:8:8, RGB 6:6:6, RGB 5:6:5, RGB 5:5:5, RGBT 5:5:5, RGB 4:4:4 data formats
  - YCbCr 4:2:2 data format
  - RGB component precision reductions for RGB 8:8:8
- Raw capture mode
  - A common raw format is RGGB. The quick capture interface is capable of capturing most any raw format as long as the software running on the PXA27x processor has been written to correctly interpret that particular raw format.
- Support for packing of 8-, 9-, and 10-bit raw pixel precision
- Support for both packed and planar data formatting for YCbCr 4:2:2 formats
- Programmable vertical and horizontal resolutions up to 2048 x 2048
- Two 8-entry (by 64 bits) and one 16-entry (by 64 bits) FIFOs

1. This document is available from the International Telecommunication Union at www.itu.int.

- Programmable sensor clock output from 196.777 kHz to 52 MHz
- Programmable interface timing signals for internal and external synchronization
- Programmable interrupts for FIFO overflow, end-of-line, and end-of-frame
- Programmable frame capture rate allows users to capture all frames or 1 out of every 2 to 8 frames

27.3 Signal Descriptions

The quick capture interface uses the input/output (I/O) signals described in Table 27-1.

Table 27-1. Quick Capture Interface I/O Signal Descriptions

Pin Name	Type	Definition
CIF_DD<9:0>	I	Data lines to transmit 4, 5, 8, 9, or 10 bits of data per pixel clock cycle
CIF_MCLK	O	Programmable clock output used by the imaging sensor
CIF_PCLK	I	Pixel clock used by the quick capture interface to clock pixel data into the input FIFO. Cannot exceed 1/4 of CICLK, where CICLK is the same frequency as the LCD clock. For 104-MHz CICLK, the maximum CIF_PCLK is 26 MHz.
CIF_LV	I/O	Line start or alternate synchronization signal used by the sensor to signal line readout or for external horizontal synchronization
CIF_FV	I/O	Frame start or alternate synchronization signal used by the sensor to signal frame readout or for external vertical synchronization

27.4 Operation

The quick capture interface supports a wide variety of camera imaging sensors from different manufacturers. Imaging sensors typically vary in terms of interfacing modes, degree of image pre-processing provided, and the number of pins required for a direct connection. The PXA27x processor's quick capture interface provides a high degree of flexibility for signal connections.

The quick capture interface receives the video/image data stream from the image sensor and provides all control signaling for operation with the sensor as either a master or slave device. In the sensor-master (master) modes, the processor's quick capture interface receives synchronization signals from the sensor. In sensor-slave (slave) mode, the quick capture interface provides the synchronization signaling to the sensor. Several reduced pin-count alternatives are supported as subsets of the master mode of operation. These include serialized data for 4-bit and 5-bit solutions and the elimination of the synchronization signals for sensors that embed SAV and EAV in the data stream.

The CIF_MCLK output for the sensor is programmable. The timing signals CIF_FV and CIF_LV, provided by the sensor, activate and reset the quick capture interface. The quick capture interface can be configured to provide an interrupt at the end of each line and each frame. It contains registers that provide setup configurations specific to the mode of operation for a particular sensor.

The SAV/EAV decode functionality is used when control sequences to synchronize the frame and line level timings are embedded within the data stream itself. Many image sensors provide the SAV/EAV decode functionality defined by the ITU-R BT.656-4 recommendation. The use of embedded timing reduces pin count for synchronization signaling.

The pixel data received can be in several possible formats: RAW RGGB, YCbCr 4:2:2, RGB 8:8:8, RGB 6:6:6, RGB 5:6:5, RGB 5:5:5, RGBT 5:5:5, and RGB 4:4:4. When a RAW RGGB capture mode is enabled, the data can be in 8-, 9-, or 10-bit formats. The RAW RGGB data are de-serialized if necessary and then packed as either 8- or 16-bit elements prior to transfer to memory. In a similar manner, the pre-processed image formats are packed into 16- or 32-bit elements.

The pre-processed formats can be manipulated and organized in memory in several different fashions.

- The YCbCr 4:2:2 format can be organized in either *packed* or *planar* formats.
- The RGB 8:8:8 and RGB 6:6:6 formats can be organized in memory in either *unpacked* or *packed* formats. RGB component precision reductions are supported for 24 bpp conversions to 18, 16, and 12 bpp.

Examples of unpacked, packed, and planar formats can be found in Section 27.4.5.

There is considerable overlap for control and interface among the various manufacturers of image-capture devices. Table 27-1 lists the quick capture interface signals. If the capture interface functionality is not required, all of its pins can be used for general-purpose input/output (GPIO).

27.4.1 Operating Modes

The quick capture interface provides a variety of user-programmable options allowing direct connection to image sensors targeted for low-power mobile video and imaging applications. The quick capture interface supports five modes of interconnection to a image sensor, listed in Table 27-2. The designations 8P, 9P, and 10P refer to 8-, 9-, and 10-bit parallel connections. The designation 4S and 5S refers to 4- and 5-bit serial connections.

Table 27-2. Quick Capture Interface Modes of Operation

Mode Name	Mode	Data Bus: S = Serial P = Parallel	Sync Signals	Definition
Master-Parallel	MP	8P, 9P, 10P	CIF_LV CIF_FV	The image sensor generates the synchronization signals. The data bus is parallel, either 8, 9, or 10 bits wide.
Slave-Parallel	SP	8P, 9P, 10P	CIF_LV CIF_FV	The quick capture interface generates the synchronization signals. The data bus is parallel, either 8, 9, or 10 bits wide.
Master-Serial	MS	4S, 5S	CIF_LV CIF_FV	The image sensor generates the synchronization signals. The data bus is serial, 4 or 5 bits wide.
Embedded-Parallel	EP	8P	—	The image sensor generates the synchronization signals. Start-of-Active-Video (SAV) and End-of-Active-Video (EAV) are embedded in the data stream. The data bus is parallel, 8 bits wide.
Embedded-Serial	ES	4S	—	The image sensor generates the synchronization signals. Start-of-Active-Video (SAV) and End-of-Active-Video (EAV) are embedded in the data stream. The data bus is serial, 4 bits wide.

Most sensors are programmed for exposure, frame rate, and possibly for additional parameters associated with exposure control and image processing. Once configured, the sensor begins providing data across the interface. If the sensor operates in master mode, it also generates the frame and line synchronization signals.

In slave mode, the synchronization signaling is provided externally by the quick capture interface to the sensor.

The embedded modes are master modes by default, in which the frame and line synchronization signals are embedded in the data stream (SAV and EAV). The embedded modes use either parallel or serialized data streams.

27.4.1.1 Master-Parallel (MP) Mode

The master-parallel (MP) mode of operation requires a parallel data-bus interface, two control signals for frame timing, and optionally a pixel clock for basic timing. In this mode, the sensor has been programmed for an exposure time and frame rate through a separate interface, typically the I^2C serial control interface (see Chapter 9, "I^2C Bus Interface Unit").

Acquisition of data from the sensor is initiated by transitions based on the state of the CIF_LV and CIF_FV signals, which are generated internally by the sensor. To support any required variation in delay from signal transitions to valid data, several programmable wait states are introduced, as illustrated in Figure 27-1.

Figure 27-1. Master Modes State Diagram

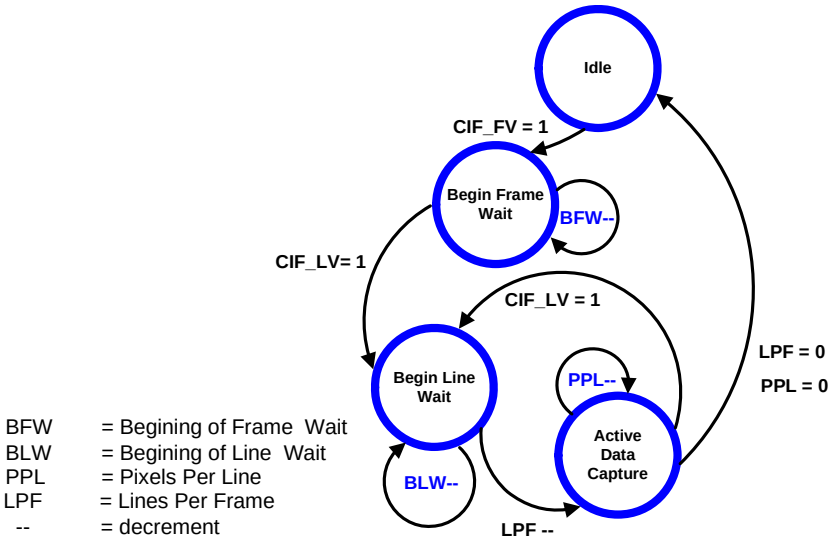


Table 27-3 identifies the control items shown in Figure 27-1 and discussed below. Labels in the drawing are identical to the register bit-field names.

Table 27-3. Master Modes State-Machine Controls

Register	Bit Field / Label	Description
CICR3	BFW	Beginning-of-frame wait

Table 27-3. Master Modes State-Machine Controls

Register	Bit Field / Label	Description
CICR2	BLW	Beginning-of-line wait
CICR1	PPL	Pixels per line
CICR3	LPF	Lines per frame

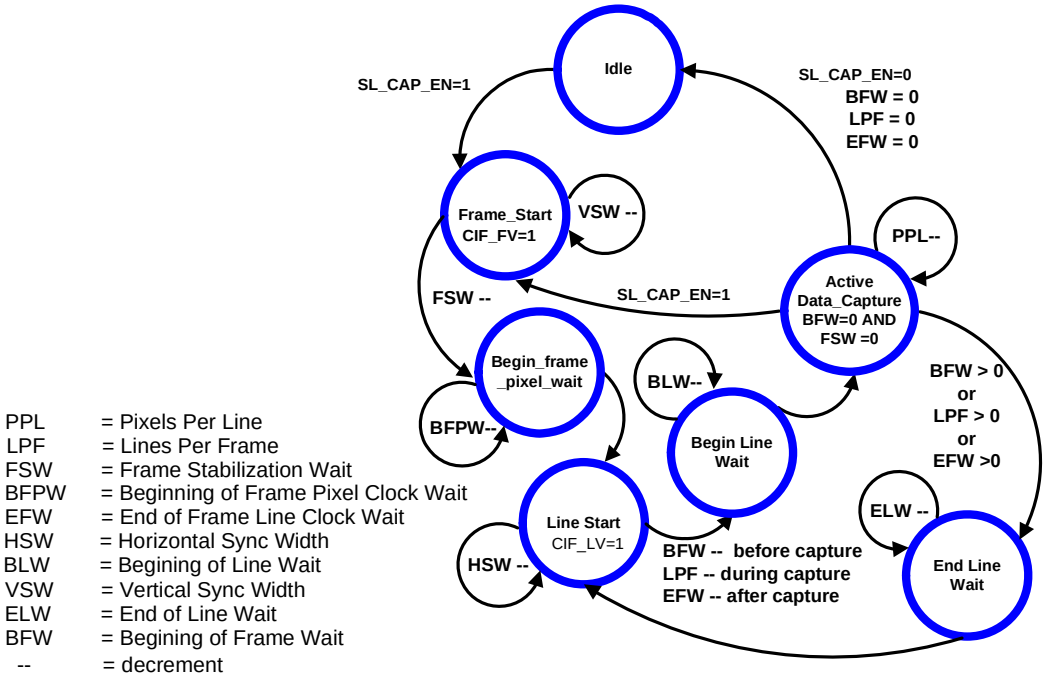
Once the quick capture interface is enabled, the capture sequence is activated by the assertion of the CIF_FV signal, which indicates that a frame read-out is about to occur. The assertion of CIF_FV causes the state machine's transition to the Begin Frame Wait state, where it then waits for the number of line clocks configured by BFW (CIF_LV). Once this BFW count has elapsed, another CIF_LV assertion brings the state machine to the Begin Line Wait state. In this state, the master mode state machine waits for BLW pixel clock cycles before moving on to the active data capture state.

Counters are maintained for both the pixels-per-line (PPL) and the lines-per-frame (LPF). Program the quick capture interface with the image sensor's specific PPL and LPF values. The End-of-Line Wait state can insert a delay equal to ELW pixel clocks after the capture of a line. Wait states are skipped if the corresponding delay values (BFW, BLW and ELW) are zero.

27.4.1.2 Slave-Parallel (SP) Mode

The slave-parallel (SP) mode of operation is an option with some image sensors. In this mode, the quick capture interface asserts the frame (CIF_FV) and line (CIF_LV) synchronization signals instead of the image sensor. Ensure that any timing requirements for the sensor are satisfied when programming the quick capture interface. The timing relationship between the capture interface and the sensor must be kept exact for proper operation. Figure 27-2 shows the slave-parallel mode state diagram.

Figure 27-2. Slave-Parallel Mode State Diagram



Once the quick capture interface is enabled and data capture is triggered by setting CICR0[SL_CAP_EN] (see Section 27.5.1), the quick capture interface begins generating synchronization signals for the sensor. The signaling sequence begins with the assertion of the CIF_FV signal, followed by the assertion of the CIF_LV signal after a delay specified by the beginning-of-frame pixel clock wait CICR2[BFPW] (see Section 27.5.3). CIF_FV is asserted for a duration of CIC3[VSW] pixel clock cycles, and CIF_LV is asserted for CIC2[HSW] pixel clock cycles. Delays are provided to skip the capture of a specified number of frames (FSW) at the beginning of a frame sequence, a specified number of lines at the beginning of a frame (BFW), and a number of pixels at the beginning of a line (BLW). Similarly, programmable delays can be inserted at the end of a line (ELW) and at the end of a frame (EFW). The quick capture interface captures data when the slave state machine has reached the active data capture state and counters corresponding to BFW and FSW have decremented to zero. Counters corresponding to PPL and LPF values indicate the end-of-line and end-of-frame captures, respectively. The wait states are skipped if the corresponding delay values (BLW, ELW, EFW, BFW and BFPW) are zero.

Note: If the sensor is configured to supply a pixel clock to the quick capture interface, (as indicated by CICR4[PCLK_EN] (see Section 27.5.5) the sensor must keep the pixel clock free running in slave mode, as long as the quick capture interface is enabled.

27.4.1.3 Master-Serial (MS) Mode

In master-serial mode of operation, only half of the parallel data bus is used, providing a lower pin-count solution. The most significant half of data is presented to the quick capture interface first, followed by the least significant half of data. The width of the data bus could be either 4 bits or 5 bits as indicated by CICR1[DATAWIDTH] (see Section 27.5.2). In 4-bit master-serial mode,

CIF_DD<3:0> transfers pixel data to the quick capture interface and the rest of CIF_DD bits are ignored. In 5-bit master-serial mode CIF_DD<4:0> transfers pixel data to the quick capture interface, and the rest of CIF_DD bits are ignored. In MS mode, the quick capture interface transitions through different wait states and data capture state as illustrated in Figure 27-1.

27.4.1.4 Embedded-Parallel (EP) and Embedded-Serial (ES) Modes

The ITU-R BT.656-4 specification provides a standard interface for unidirectional connection between a single source and a single destination for digital component video signals in 525-line and 625-line television systems operating at the 4:2:2 level of recommendation ITU-R BT.601. The specification describes the characteristics of bit-parallel and bit-serial implementations. The important component of the BT.656-4 specification for the PXA27x processor is the specification of the embedded timing reference signals, Start-of-Active-Video (SAV) and End-of-Active-Video (EAV). Use of the SAV and EAV signals allows the CIF_LV and CIF_FV signals to be eliminated from the interface, thereby reducing the pin count.

27.4.1.4.1 Embedded Control Data

To distinguish between the video/image data and the control sequences, certain levels in the video sample data must be defined as illegal. In the ITU-R BT.656-4 specification, the color space is defined according to the ITU-R BT601. In this, the luminance channel Y is defined to have a nominal range of 16 to 235; and the chrominance channels, Cb and Cr are defined to have a range of 16 to 240, with zero signal corresponding to level 128. When transferring image data using a different format, the values FF and 00 must also be defined as illegal. The values FF and 00 are reserved for use in timing reference signals formed by the 4-byte sequence FF 00 00 XY. The timing reference sequence is illustrated in Table 27-4.

Table 27-4. ITU-R BT.656-4 EAV/SAV Sequence for 8-Bit

	8-Bit Data							
	7 MSB	6	5	4	3	2	1	0 LSB
1st Byte (FF)	1	1	1	1	1	1	1	1
2nd Byte (00)	0	0	0	0	0	0	0	0
3rd Byte (00)	0	0	0	0	0	0	0	0
Status Word (XY)	1	F	V	H	P3	P2	P1	P0

The reference coding is based on three bits: F, V, and H. These bits allow the conventional video timing signals HSYNC, VSYNC, and BLANK to be embedded in the video stream. The quick capture interface routes the embedded video data to memory when the encoding indicates active video. The image data can be in any of the supported formats, in which the values FF and 00 are excluded levels in the image data. Use the parity bits P3:P0 to correct one-bit errors and detect two-bit errors. The embedded timing signals are illustrated in Table 27-5.

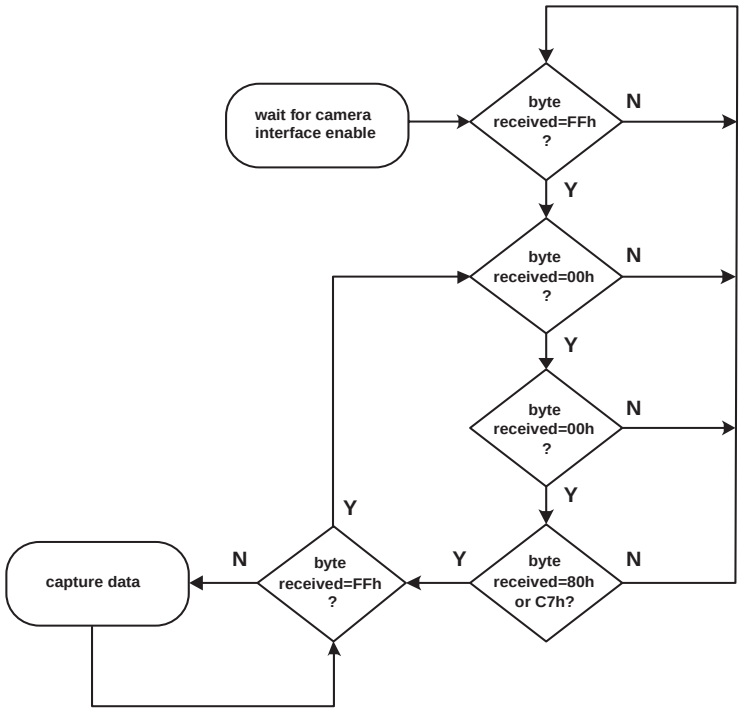
Table 27-5. ITU-R BT.656-4 Timing Reference Coding

Bit	7	6	5	4	3	2	1	0
8 Bits/Pixel (bpp)	1	F	V	H	P3	P2	P1	P0

Bits	Name	Description
6	F	**NOTE:** The quick capture interface does not utilize “fields”, so the value of F has no influence on data capture other than its value influences the value of the parity bits. Field: This specifies the frame start signal. For the 625/525-line TV systems, it is used for the even/odd fields. (A field is a defined group of lines in a TV system) 0 = First field of a frame (field 1) 1 = Second field of a frame (field 2)
5	V	For the quick capture interface, this bit must be clear to indicate scan lines are an integral part of the image. Vertical Interval: This specifies whether the data is active video, or vertical blanking. 0 = No vertical blanking 1 = Vertical blanking, no data is captured
4	H	Horizontal Interval: This bit specifies the start or end of active video. 0 = Start-of-Active-Video (SAV). 1 = End-of-Active-Video (EAV)
3:0	P3:P0	Parity Bits: These bits have states dependent on the states of the bits F, V, and H. This arrangement permits 1-bit errors to be corrected and 2-bit errors to be detected. F V H – P3 P2 P1 P0 0b000–0000 -> (SAV) Field 1 Active Video 0b001–1101 -> (EAV) Field 1 Active Video 0b010–1011 -> (SAV) Field 1 Blanking 0b011–0110 -> (EAV) Field 1 Blanking 0b100–0111 -> (SAV) Field 2 Active Video 0b101–1010 -> (EAV) Field 2 Active Video 0b110–1100 -> (SAV) Field 2 Blanking 0b111–0001 -> (EAV) Field 2 Blanking

The flow diagram for the embedded modes is illustrated in Figure 27-3.

Figure 27-3. Embedded Synchronization Mode Flow Diagram



27.4.2 Clock (CICLK and MCLK) Generation

To generate the sensor master clock (MCLK) of programmable frequency, the clock generator divides the internal input clock, CICLK, by the programmed clock divider value CICR4[DIV]. CICLK is the same frequency as the internal LCD clock, and the frequencies are as shown in the LCD Frequency column of Table 3-7, "Clock Frequencies" on page 3-20. The following equation gives the MCLK frequency.

$$MCLK = \frac{CICLK}{2(DIV + 1)}$$

Note: The sensor can use either the MCLK supplied by the quick capture interface or its own local clock as the master clock. *However, the pixel clock supplied by the sensor to the quick capture interface cannot exceed 1/4 the frequency of CICLK.* If the sensor does not supply a pixel clock to the quick capture interface (in which case sensor must use MCLK supplied by the quick capture interface), the MCLK frequency must not exceed 1/4 the frequency of CICLK.

27.4.3 Serial-to-Parallel Conversion

If serial mode of operation is selected (CICR0[SIM] is programmed for master-serial or embedded-serial), a serial-to-parallel module converts the data stream into parallel data. The serial-to-parallel module performs 5:10 conversion for 5-bit wide interfaces, and 4:8 conversion for 4-bit wide interfaces with the sensor.

27.4.4 FIFO Operation

The capture interface has three separate FIFOs to act as temporary storage for the captured video/ image data from the image sensor. The channel 0 FIFO has a storage capacity of 128 bytes, and each of the 16 FIFO entries is 8 bytes wide. The channel 1 and channel 2 FIFOs each have a storage capacity of 64 bytes. Each of their 8 entries is 8 bytes wide.

The planarized YCbCr mode uses all three channel input FIFOS. The channel 0 FIFO stores the Y data component, the channel 1 FIFO stores the Cb component, and the channel 2 FIFO stores the Cr component. Other modes of operation use the channel 0 FIFO only.

Note: All capture interface FIFOs are read-only. Any write to the capture interface FIFOs causes a target abort.

27.4.4.1 FIFO Data Packing

The input data from the sensor is packed into the required format and written into the FIFO. If serial mode is selected, output of the serial-to-parallel converter is captured, and re-arranged based on input data width and number of bits per pixel. In parallel mode, the same functions are performed on the parallel data from the sensor. The timing references for capture are provided by the CIF_LV and CIF_FV inputs (or the decoded SAV/EAV in embedded modes).

When an End of Frame is encountered during data capture, the remaining bits in the 8-byte wide FIFO location (after writing the last data) are padded with zeros. For example, if the last sample of a frame occupies byte 0 of the 8-byte FIFO entry, bytes 1 through 7 is written with zeros. The first valid data of the next frame is loaded to the next 8-byte wide FIFO entry. In RGBT modes (RGBT5:5:5 or RGBT 8:8:8), bits at the end of frame that are padded to fill to the 8-byte boundary in the FIFO have the transparency bit (CICR1[TBIT]—see Section 27.5.2) inserted at the desired bit position.

Note: The amount of time required to perform the zero-padding depends on the last valid byte position, and if planarized YCbCr mode is enabled (planarized YCbCr uses all 3 FIFOs so all 3 need to be padded). If the next frame capture starts before the zero-padding for the previous frame is complete, undefined capture interface operation results. When the number of bytes per frame is not a multiple of eight, the inter-frame delay (measured from the last valid pixel of a frame to the first valid pixel of the next frame) must be greater than or equal to 64 CICLK cycles. To avoid the undefined capture interface operation mentioned above, software must ensure the delay requirement (minimum of 64 CICLK cycles) is satisfied.

The capture-interface data output FIFOs (eight bytes wide) are padded with zeros at the end of a frame if the captured frame data does not end at an 8-byte boundary. The DMA controller must be programmed to read the entire data including any zero padding. The programmed DMA length must be a multiple of 8 bytes. Similarly, if the processor is reading data from the capture-interface output FIFOs (1,2 or 4 bytes at a time), it must read the entire FIFO entry including padded zeros (if any, at end of frame) before moving on to the next frame.

27.4.4.2 Processor-Initiated Data Transfers from FIFOs

The processor can perform data transfers from the quick capture interface FIFOs in response to an interrupt generated by the FIFO's threshold level logic. The channel 0 FIFO's threshold level is programmed in CIFR[THL_0] (see Section 27.5.8). Threshold levels of the channel 1 and channel 2 FIFOs are fixed at 32 bytes. When the FIFO threshold levels are reached, an interrupt is generated (if enabled in CICR0[RDAVM]—see Section 27.5.1), which signals the CPU to empty

the FIFOs by reading the corresponding CIBRx Receive Buffer registers (see Section 27.5.9). In response to the interrupts, the processor reads from the corresponding CIBRx. The capture interface supports processor reads of one, two or four bytes at a time. The number of bytes in the capture-interface FIFOs corresponding to a captured frame is always a multiple of eight bytes, and the processor must read the entire frame data including padded zeros (if any) before moving on to read data corresponding to the next frame.

Software can poll the FIFO control register (CIFR; see Section 27.5.8) to determine how many bytes are in a FIFO and the status register (CISR; see Section 27.5.7) to see if the FIFO is empty.

27.4.4.3 DMA Data Transfers from FIFOs

The quick capture interface has three DMA receive-data service requests, one for each of the three input-channel FIFOs.

When a FIFO issues a DMA service request, the DMA controller responds by transferring a burst of data from the FIFO to the destination address specified in the DMA descriptor. The DMA Source Address register (DSADRx; see Section 5.5.3, “DMA Source Address Register (DSADRx)” on page 5-33) must point to CIBR0, CIBR1, or CIBR2, depending on which one of the three DMA services is being employed. These are the only valid source addresses for quick capture interface DMA.

The DMA descriptors must be programmed to transfer all bytes in a frame. In planarized YCbCr mode of operation, program individual DMA channels to transfer the total number of bytes per frame for each of the components.

For DMA transfers, refer to Chapter 5, “DMA Controller”. Use the following programming model:

1. In register DCMDx, program the number of bytes to be transferred, burst sizes, and other control items. Set the descriptor chain to transfer an entire frame. The peripheral width for DMA transactions from the quick capture interface is always eight bytes, regardless of the DCMD[WIDTH] setting.
2. Program DSADRx with the address of CIBR0, CIBR1, or CIBR2, depending on which channel FIFO is to be served by DMA. Program DTADRx with the desired target address.
3. Using the quick capture interface control registers CICRx, configure the quick capture interface for the desired operation.
4. Enable the quick capture interface by setting CICR0[ENB].
5. Set the DCSRx[RUN] bits for the selected channel(s).

27.4.4.4 Trailing Bytes in FIFOs

When the number of samples in the FIFO is less than its FIFO trigger threshold level and no additional data is received, the remaining bytes are called *trailing bytes*. The quick capture interface uses a time-out based request mechanism to handle trailing bytes. A time-out condition exists when the FIFO level is below its trigger level, and the FIFO has been idle for a period of time defined by the value programmed in register CITOR (see Section 27.5.6). The time-out counter is reset when a new data sample is written to the FIFO or when a read from the FIFO occurs. When all three quick capture interface FIFOs are used (in Planarized YCbCr mode), the time-out counter starts counting after the last data sample has been written to the FIFO and is reset only after a read from all of the FIFOs. When a time-out occurs, the FIFO time-out bit CISR[FTO] is set, and an interrupt request is issued if unmasked in CICR0. Values in the Quick Capture Interface FIFO Control register (see Section 27.5.8) must be read to determine how many bytes

remain in the FIFOs. If DMA FIFO data transfers are enabled (CICR0[DMA_EN] is set) and a time-out is detected, the FIFO makes a request for DMA transfer even though the FIFO threshold is not reached. The descriptor chain must be programmed to handle the entire frame to make sure that all trailing bytes are transferred.

Note: When the quick capture interface is disabled by clearing CICR0[ENB], data capture is immediately stopped and a trailing byte situation is possible. Software must make sure that the DMA channel(s) is stopped by resetting the corresponding run bit. In this case, the data in the FIFOs is discarded.

27.4.5 Pixel Formats

The quick capture interface supports RAW, RGB 8:8:8, RGBT 8:8:8, RGB 6:6:6, RGB 5:5:5, RGBT 5:5:5, RGB 5:6:5, RGB 4:4:4, and YCbCr 4:2:2 video sampling formats.

27.4.5.1 Raw Pixel Data Formats

All image data originates with the raw sensor pixel data. Each sub-pixel has a color filter over it, passing light centered around one (or more) wavelength. As received from the sensor, each pixel has 8, 9, or 10 bits of accuracy. The most common color filter array is the Bayer pattern, which interleaves red and green pixels on the even/odd rows and blue and green pixels on the odd/even rows.

The PXA27x processor can accept raw data in most any format. When configured for raw data capture, the PXA27x processor makes no assumptions about the format of the data. The key requirement is that the image capture software running on the PXA27x processor must be written to correctly interpret data in the particular format used.

27.4.5.1.1 Raw 8-Bit Data Format

The PXA27x processor supports the acquisition of raw 8-bit sensor data that must be processed further prior to display or encode. The data type for the required processing is 8-bit unsigned integer. The 8-bit data is packed into 32-bit words prior to storage. There are no tag bits specifying the color components with raw image capture.

Table 27-6. Memory Organization for Raw 8-Bit Data

Bit	7	6	5	4	3	2	1	0
8 bits/pixel	Pixel Data<7:0>							

Memory Organization for Little Endian

Bit	31 – 24	23 – 16	15 – 8	7 – 0
Base+ 0x0	Pixel 3	Pixel 2	Pixel 1	Pixel 0
Base+ 0x4	Pixel 7	Pixel 6	Pixel 5	Pixel 4

27.4.5.1.2 Raw 9-Bit Data Format

The PXA27x processor supports the acquisition of raw 9-bit sensor data that must be processed further prior to display or encode. The data type for the required processing is 16-bit unsigned integer. The 9-bit data is packed into 16-bit half words prior to storage. There are no tag bits specifying the color components with raw image capture. The upper 7 bits of the 16-bit word are unused and written as zeros. The pixel format for 9-bit raw data capture is illustrated in Table 27-7.

Table 27-7. Memory Organization for Raw 9-Bit Data

Bit	15 14 13 12 11 10 9	8 7 6 5 4 3 2 1 0
16 bits/pixel	Unused bits, written as zeros	Pixel Data<8:0>

Memory Organization

Bit	31 – 16	15 – 0
Base+0x0	Pixel 1	Pixel 0
Base+0x4	Pixel 3	Pixel 2

27.4.5.1.3 Raw 10-Bit Data Format

The PXA27x processor supports the acquisition of raw 10-bit sensor data that must be processed further prior to display or encode. The data type for the required processing is 16-bit unsigned integer. The 10-bit data is packed into 16-bit half words prior to storage. There are no tag bits specifying the color components with raw image capture. The upper 6 bits of the 16-bit word are unused and written as zeros. The pixel format for 10-bit raw data capture is illustrated in Table 27-8.

Table 27-8. Memory Organization for Raw 10-Bit Data

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16 bits/pixel	Unused bits, written as zeros						Pixel Data<9:0>									

Memory Organization

Bit	31	16	15	0
Base+0x0	Pixel 1		Pixel 0	
Base+0x4	Pixel 3		Pixel 2	

27.4.5.2 Pre-Processed RGB Pixel Data Formats

It is important to be aware that not all image sensors output color components in the same order or sequence for a given color space. Table 27-9 illustrates the expected sequence for RGB 8:8:8. Table 27-10 illustrates the expected sequence for RGB 5:6:5. For the RGB color space, the red components are expected first, next green, and then blue.

This section also illustrates the memory organization of the various RGB packed and un-packed formats.

Table 27-9. 8-Bit Data-Capture Sequence for RGB 8:8:8

Data Bus	RGB 8:8:8 Byte Sequence					
CIF_DD<7>	$R_{7(0)}$	$G_{7(0)}$	$B_{2(0)}$	$R_{7(1)}$	$G_{7(1)}$	$B_{2(1)}$
CIF_DD<6>	$R_{6(0)}$	$G_{6(0)}$	$B_{1(0)}$	$R_{6(1}$	$G_{6(1)}$	$B_{1(1)}$
CIF_DD<5>	$R_{5(0)}$	$G_{5(0)}$	$B_{0(0)}$	$R_{5(1)}$	$G_{5(1)}$	$B_{0(1)}$
CIF_DD<4>	$R_{4(0)}$	$G_{4(0)}$	$B_{4(0)}$	$R_{4(1)}$	$G_{4(1)}$	$B_{4(1)}$
CIF_DD<3>	$R_{3(0)}$	$G_{3(0)}$	$B_{3(0)}$	$R_{3(1)}$	$G_{3(1)}$	$B_{3(1)}$
CIF_DD<2>	$R_{2(0)}$	$G_{2(0)}$	$B_{2(0)}$	$R_{2(1)}$	$G_{2(1)}$	$B_{2(1)}$
CIF_DD<1>	$R_{1(0)}$	$G_{1(0)}$	$B_{1(0)}$	$R_{1(1)}$	$G_{1(1)}$	$B_{1(1)}$
CIF_DD<0>	$R_{0(0)}$	$G_{0(0)}$	$B_{0(0)}$	$R_{0(1)}$	$G_{0(1)}$	$B_{0(1)}$
Byte sequence	0	1	3	4	5	6
Pixel	Pixel 0			Pixel 1		

Table 27-10. 8-Bit Data-Capture Sequence for RGB 5:6:5 Color Space

Data Bus	RGB 5:6:5 Packed Byte Sequence							
CIF_DD<7>	$G_{2(0)}$	$B_{4(0)}$	$G_{2(1)}$	$B_{4(1)}$	$G_{2(2)}$	$B_{4(2)}$	$G_{2(3)}$	$B_{4(3}$
CIF_DD<6>	$G_{1(0)}$	$B_{3(0)}$	$G_{1(1)}$	$B_{3(1)}$	$G_{1(2)}$	$B_{3(2)}$	$G_{1(3)}$	$B_{3(3)}$
CIF_DD<5>	$G_{0(0)}$	$B_{2(0)}$	$G_{0(1)}$	$B_{2(1)}$	$G_{0(2)}$	$B_{2(2)}$	$G_{0(3)}$	$B_{2(3)}$
CIF_DD<4>	$R_{4(0)}$	$B_{1(0)}$	$R_{4(1)}$	$B_{1(1)}$	$R_{4(2)}$	$B_{1(2)}$	$R_{4(3)}$	$B_{1(3)}$
CIF_DD<3>	$R_{3(0)}$	$B_{0(0)}$	$R_{3(1)}$	$B_{0(1)}$	$R_{3(2)}$	$B_{0(2)}$	$R_{3(3)}$	$B_{0(3)}$
CIF_DD<2>	$R_{2(0)}$	$G_{5(0)}$	$R_{2(1)}$	$G_{5(1)}$	$R_{2(2)}$	$G_{5(2)}$	$R_{2(3)}$	$G_{5(3)}$
CIF_DD<1>	$R_{1(0)}$	$G_{4(0)}$	$R_{1(1)}$	$G_{4(1)}$	$R_{1(2)}$	$G_{4(2)}$	$R_{1(3)}$	$G_{4(3)}$
CIF_DD<0>	$R_{0(0)}$	$G_{3(0)}$	$R_{0(1)}$	$G_{3(1)}$	$R_{0(2)}$	$G_{3(2)}$	$R_{0(3)}$	$G_{3(3)}$
Byte sequence	0	1	2	3	4	5	6	7
Pixel Byte order	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB
Pixel	0		1		2		3	

27.4.5.2.1 RGB 8:8:8 Pixel Data Format

The PXA27x processor supports the acquisition of pre-processed RGB 8:8:8 data. The memory organization for pre-processed image data can be in packed or unpacked formats. The memory organization for the 24-bpp RGB 8:8:8 unpacked format is illustrated in Table 27-11.

Table 27-11. Memory Organization for Unpacked RGB 8:8:8 (Pixel Depth of 24 bpp)

Bit	23 ... 16	15 ... 8	7 ... 0
24bpp	Red[23:16]	Green[15:8]	Blue[7:0]

Memory Organization

Bit	31 ... 24	23 ... 0
Base+0x0	Unused bits, written as zeros	Pixel 0
Base+0x4	Unused bits, written as zeros	Pixel 1

The memory organization for the 24-bpp RGB 8:8:8 packed format is illustrated in Table 27-12.

Table 27-12. Memory Organization for Packed RGB 8:8:8 (Pixel Depth of 24 bpp)

Bit	23 ... 16	15 ... 8	7 ... 0
24bpp	Red[23:16]	Green[15:8]	Blue[7:0]

Memory Organization

Bit	31 ... 24	23 ... 16	15 ... 8	7 ... 0
Base+0x0	Pixel1[7:0]	Pixel0[23:0]		
Base+0x4	Pixel2[15:0]		Pixel1[23:8]	
Base+0x8	Pixel3[23:0]			Pixel2[23:16]

27.4.5.2.2 RGBT 8:8:8 Pixel Data Format

The PXA27x processor supports the acquisition of pre-processed RGB 8:8:8 data for storage in RGBT 8:8:8 format. The pixel depth is 25 bpp, with the most significant bit being a programmable transparency bit. The memory organization for the 25-bpp RGBT 8:8:8 format is illustrated in Table 27-13.

Table 27-13. Memory Organization for RGBT 8:8:8 (Pixel Depth of 25 bpp)

Bit	31 … 25	24	23 … 16	15 … 8	7 … 0
25bpp	Unused bits, written as zeros	T	Red[23:16]	Green[15:8]	Blue[7:0]

Memory Organization

Bit	31 … 25	24 … 0
Base+0x0	Unused bits, written as zeros	Pixel 0
Base+0x4	Unused bits, written as zeros	Pixel 1

27.4.5.2.3 RGB 6:6:6 Pixel Data Format

The PXA27x processor supports the acquisition of pre-processed RGB 6:6:6 data. The memory organization for pre-processed image data can be in packed or unpacked formats. The memory organization for the 18-bpp RGB 6:6:6 format is illustrated in Table 27-14.

Table 27-14. Memory Organization for Unpacked RGB 6:6:6 (Pixel Depth of 18 bpp)

Bit	31 … 18	17 … 12	11 … 6	5 … 0
18bpp	Unused bits, written as zero	Red[17:12]	Green[11:6]	Blue[5:0]

Memory Organization

Bit	31 … 18	17 … 0
Base+0x0	Unused bits, written as zeros	Pixel 0
Base+0x4	Unused bits, written as zeros	Pixel 1

The memory organization for the 18-bpp RGB 6:6:6 packed format is illustrated in Table 27-15.

Table 27-15. Memory Organization for Packed RGB 6:6:6 (Pixel Depth of 18 bpp)

Bit	23 18	17 12	11 6	5 0
18bpp	Unused bits, written as zeros	Red[17:12]	Green[11:6]	Blue[5:0]

Memory Organization

Bit	31 24	23 16	15 8	7 0
Base+0x0	Pixel1[7:0]	{000000,Pixel0[17:0]}		
Base+0x4	Pixel2[15:0]		{000000, Pixel1[17:8]}	
Base+0x8	{000000,Pixel3[17:0]}			{000000,Pixel2[17:16]}

27.4.5.2.4 RGB 5:6:5 Pixel Data Format

The memory organization for pre-processed image data is illustrated in Table 27-16.

Table 27-16. Memory Organization for RGB 5:6:5 Format (Pixel Depth of 16 bpp)

Bit	15 14 13 12 11	10 9 8 7 6 5	4 3 2 1 0
	Red[15:11]	Green[10:5]	Blue[4:0]

Memory Organization

Bit	31 16	15 0
Base+0x0	Pixel 1	Pixel 0
Base+0x4	Pixel 3	Pixel 2

27.4.5.2.5 RGB 5:5:5 and RGBT 5:5:5 Pixel Data Formats

The PXA27x processor supports the acquisition of pre-processed RGB 5:5:5 data for storage in RGBT 5:5:5 format. The pixel depth is 16 bpp with the most significant bit being a programmable transparency bit. The memory organization for the 16-bpp RGBT 5:5:5 format is illustrated in Table 27-17.

Table 27-17. Memory Organization for RGBT 5:5:5 Format (Pixel Depth of 16 bpp)

Bit	15	14 13 12 11 10	9 8 7 6 5	4 3 2 1 0
	T	Red[14:10]	Green[9:5]	Blue[4:0]

Memory Organization

Bit	31 16	15 0
Base+0x0	Pixel 1	Pixel 0
Base+0x4	Pixel 3	Pixel 2

27.4.5.2.6 RGB 4:4:4 Pixel Data Format

The PXA27x processor supports the acquisition of pre-processed RGB 4:4:4 data. The memory organization for pre-processed image data is illustrated in Table 27-18.

Table 27-18. Memory Organization for RGB 4:4:4 Pixel Data (Pixel Depth of 12 bpp)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Unused bits, written as zeros				Red[11:8]				Green[7:0]				Blue[3:0]			

Memory Organization

Bit	31															16	15													0
Base+0x0	Pixel 1																Pixel 0													
Base+0x4	Pixel 3																Pixel 2													

27.4.5.3 Pre-Processed YCbCr Pixel Data Formats

The quick capture interface accepts YCbCr data in the 4:2:2 format. The YCbCr color space is important because data in this format is frequently used for input to MPEG and JPEG compression sequences or Codecs. Overlay 2 in the LCD controller provides a hardware-based color space conversion engine that can be programmed to accept the YCbCr data. This allows YCbCr data to be streamed from the quick capture interface directly to Overlay 2. The color space conversion engine converts the YCbCr image data into the RGB color space so it can be displayed on a screen.

There are several useful formats for YCbCr image data. These formats fall into two distinct categories. The packed format and the planar format. In the packed format, the Y, Cb, and Cr samples are packed together into groups of three that are stored as a single array in memory. In the planar format, the Y, Cb, and Cr components are stored as three separate arrays.

The YCbCr 4:2:2 data can be presented across either a parallel or serial interface. This format has a horizontal sub-sampling interval of two. In this particular YCbCr format, there are two luminance (Y) values for each chrominance pair (Cb, Cr). The expected 8-bit data sequence for YCbCr 4:2:2 is illustrated in Table 27-19.

Table 27-19. 8-Bit Data Capture Sequence for YCbCr 4:2:2 Color Space

Data Bus	YCbCr 4:2:2 Byte Sequence							
CIF_DD<7>	Cb_{07}	Y_{07}	Cr_{07}	Y_{17}	Cb_{27}	Y_{27}	Cr_{27}	Y_{37}
CIF_DD<6>	Cb_{06}	Y_{06}	Cr_{06}	Y_{16}	Cb_{26}	Y_{26}	Cr_{26}	Y_{36}
CIF_DD<5>	Cb_{05}	Y_{05}	Cr_{05}	Y_{15}	Cb_{25}	Y_{25}	Cr_{25}	Y_{35}
CIF_DD<4>	Cb_{04}	Y_{04}	Cr_{04}	Y_{14}	Cb_{24}	Y_{24}	Cr_{24}	Y_{34}
CIF_DD<3>	Cb_{03}	Y_{03}	Cr_{03}	Y_{13}	Cb_{23}	Y_{23}	Cr_{23}	Y_{33}
CIF_DD<2>	Cb_{02}	Y_{02}	Cr_{02}	Y_{12}	Cb_{22}	Y_{22}	Cr_{22}	Y_{32}
CIF_DD<1>	Cb_{01}	Y_{01}	Cr_{01}	Y_{11}	Cb_{21}	Y_{21}	Cr_{21}	Y_{31}
CIF_DD<0>	Cb_{00}	Y_{00}	Cr_{00}	Y_{10}	Cb_{20}	Y_{20}	Cr_{20}	Y_{30}
Y pixel components	0		1		2		3	
Cb, Cr pixel components	0,1				2,3			
Byte sequence	0	1	2	3	4	5	6	7

27.4.5.3.1 YCbCr Planar Format

The PXA27x processor supports the acquisition of YCbCr 4:2:2 data and the storage of pixel data in planar format. The planar format requires the luminance and chrominance planes to be stored in separate locations in memory, although they can be in contiguous memory areas. The memory organization for the planar format is illustrated in Table 27-20.

Table 27-20. Memory Organization for 4:2:2 YCbCr Planar Format

Luminance Channel (Y) Memory Organization

Bit	31 … 24	23 … 16	15 … 8	7 … 0
Base+ 0x00	Y_{n+3}	Y_{n+2}	Y_{n+1}	Y_n
Base + 0x04	Y_{n+7}	Y_{n+6}	Y_{n+5}	Y_{n+4}

Red Chrominance Channel (Cr) Memory Organization

	31 … 24	23 … 16	15 … 8	7 … 0
Base+ K+0x00	Cr_{n+3}	Cr_{n+2}	Cr_{n+1}	Cr_n
Base+ K+0x04	Cr_{n+7}	Cr_{n+6}	Cr_{n+5}	Cr_{n+4}

Blue Chrominance Channel (Cb) Memory Organization

	31 … 24	23 … 16	15 … 8	7 … 0
Base+ M+0x00	Cb_{n+3}	Cb_{n+2}	Cb_{n+1}	Cb_n
Base+ M+0x04	Cb_{n+7}	Cb_{n+6}	Cb_{n+5}	Cb_{n+4}

27.4.5.3.2 4:2:2 YCbCr Video Packed Format

The PXA27x processor supports the acquisition of YCbCr 4:2:2 data and its storage in packed format. The packed format requires the luminance and chrominance components to be stored in contiguous locations in memory. The memory organization for the packed format is illustrated in Table 27-21.

Table 27-21. Memory Organization for 4:2:2 YCbCr Packed Format

	Memory Organization			
Bit	31 24	23 16	15 8	7 0
Base+ 0x0	Y_{n+1}	Cr_n	Y_n	Cb_n
Base+ 0x4	Y_{n+3}	Cr_{n+1}	Y_{n+2}	Cb_{n+1}

27.4.5.4 Conversion from RGB 8:8:8 24 bpp to 12, 16, 18, and 25 bpp RGB Formats

The conversion from the pre-processed pixel format RGB 8:8:8 to RGBT 8:8:8, RGB 6:6:6, RGB 5:6:5, RGBT 5:5:5, and RGB 4:4:4 is supported to allow devices that support full image processing chains to be easily formatted for LCD controller RGB pixel formats. The conversion algorithm is straightforward and involves a scaling operation for each of the color components. For example, the format conversion from RGB 8:8:8 to RGBT 5:5:5, the five most significant bits of each of the red, green, and blue color are combined into a 16-bpp format, as shown in Table 27-22.

Table 27-22. Packing and Precision Conversion from RGB 8:8:8 to RGB 5:5:5

Bit	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGB 8:8:8	Red [7:0]								Green [7:0]								Blue [7:0]							

3 LSBs for Each Color Channel are Truncated to Form Lower Precision RGB 5:5:5 Format.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGBT 5:5:5	T	Red [4:0]					Green[4:0]					Blue[4:0]				

The supported format conversions for RGB 8:8:8 include the RGBT 8:8:8 format for 25 bpp, RGB 6:6:6 format for 18 bpp, RGB 5:6:5 format for 16 bpp, RGBT 5:5:5 format for 16 bpp, and RGB 4:4:4 format for 12 bpp.

Table 27-23. Supported Color Component Precision Conversions for RGB 8:8:8

Pixel Depth	Supported Format Conversion from RGB 8:8:8
25bpp	RGBT 8:8:8
18bpp	RGB 6:6:6
16bpp	RGB 5:6:5
16bpp	RGBT 5:5:5
12bpp	RGB 4:4:4

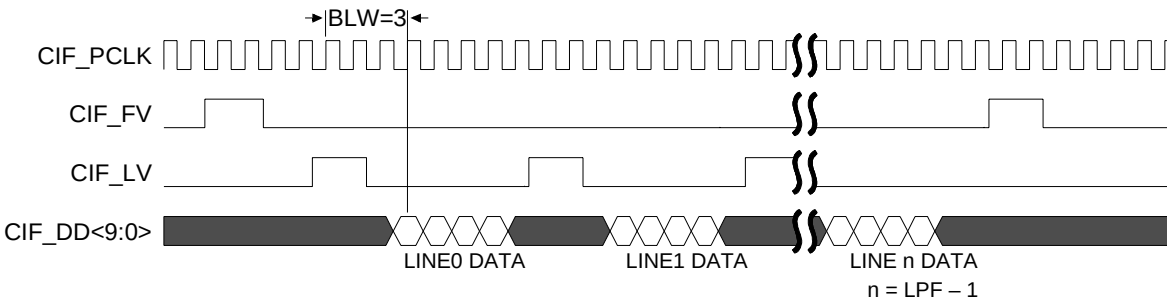
27.4.6 Functional Timing

27.4.6.1 Master-Parallel Functional Timing

The master-parallel interface timing is shown in Figure 27-4. The valid data is captured with the active edge of CIF_PCLK, after "Beginning of Line Wait Count" (programmed with CICR2[BLW]) CIF_PCLK cycles have elapsed from the CIF_LV. From the last data of a line, the interface waits a delay of "End of Line Wait" (programmed with CICR2[ELW]) CIF_PCLK cycles occurs before a new line capture can begin. At the end of the capture of the last line of a frame, the quick capture interface waits for the assertion of CIF_FV to begin the next frame capture sequence.

Note: To avoid potential loss of line data, the image sensor must assert CIF_LV only after the last pixel of the previous line.

Figure 27-4. Master-Parallel Functional Timing

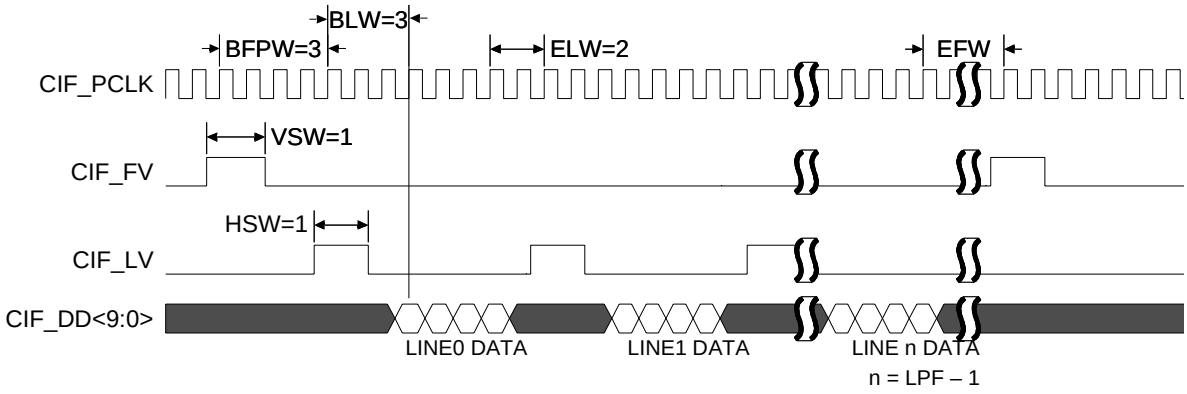


BLW = Beginning of Line Pixel Clock Wait Count

27.4.6.2 Slave-Parallel Functional Timing

In slave-parallel mode, the quick capture interface drives the synchronization signals CIF_LV and CIF_FV. Before the quick capture interface starts operating in this mode, the sensor must be configured to float the synchronization pins. Figure 27-5, shows the functional timing for slave-parallel mode. CIF_FV and CIF_LV are driven for the duration specified by Vertical Sync Width (VSW) and Horizontal Sync Width (HSW) respectively.

Figure 27-5. Slave-Parallel Functional Timing

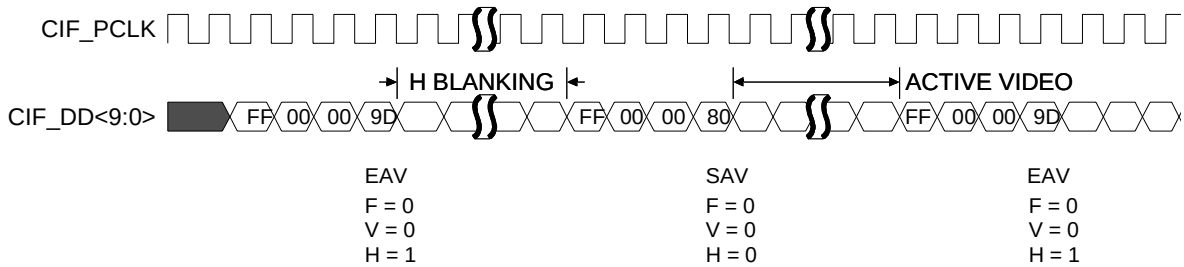


VSW = Vertical Sync Pulse Width – 1
HSW = Horizontal Sync Pulse Width – 1
BFPW= Beginning of Frame Pixel Wait Count – 1
BLW = Beginning of Line Wait
ELW = End of Line Pixel Clock Wait Count
EFW = End of Frame Line Clock Wait Count

27.4.6.3 Embedded-Parallel Functional Timing

Figure 27-6 shows the embedded-parallel functional timing for an 8-bit wide interface. The first embedded synchronization sequence (xFF00009D) signals an EAV (End of Active Video) and the second synchronization sequence (xFF000080) signals SAV (Start of Active Video). Between EAV and SAV is the Horizontal Blanking period. Any data sent during this blanking period is not captured by the quick capture interface. The active video window is between an SAV and an EAV. Similarly a set V bit in the synchronization sequence indicates Vertical Blanking period, and data presented to the quick capture interface during this interval is not captured.

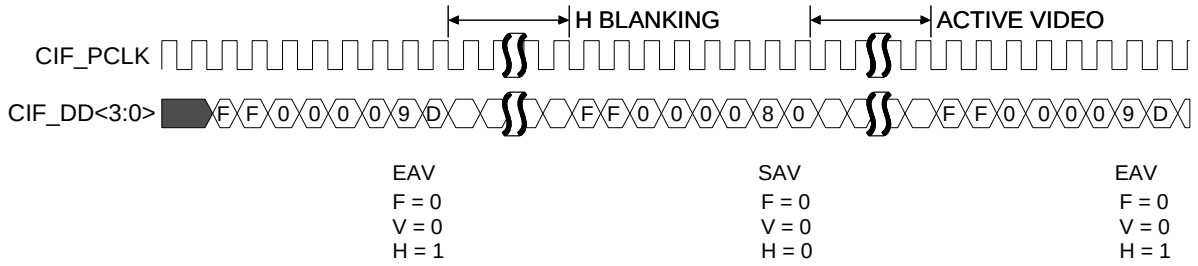
Figure 27-6. Embedded-Parallel Functional Timing for 8-Bit Interface



27.4.6.4 Embedded-Serial Functional Timing

Figure 27-7 shows the embedded serial functional timing for a 4-bit wide interface. The timing is very similar to the embedded-parallel mode, except that a single datum is presented in two CIF_PCLK cycles. The most significant half of data is sent first, followed by the least significant half. The first embedded synchronization sequence signals an EAV (End of Active Video) and the second synchronization sequence signals SAV (Start of Active Video). Between EAV and SAV is the Horizontal Blanking period. Any data sent during this blanking period is not captured by the quick capture interface. The active video window is between an SAV and an EAV. Similarly, if the V field is set in the synchronization sequence, this indicates vertical blanking period, and data presented to the quick capture interface during this interval is not captured.

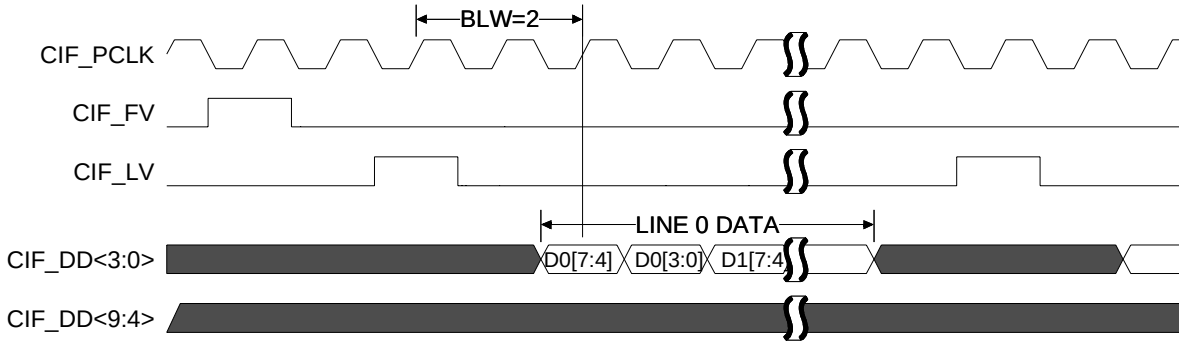
Figure 27-7. Embedded-Serial Functional Timing for 4-Bit Interface



27.4.6.5 Master-Serial Functional Timing

Figure 27-8, shows the functional timing for master-serial mode of operation with a 4 bit wide interface. The timing is very similar to the master-parallel mode, except that the a single datum is presented in two CIF_PCLK cycles. The most significant half of data is sent first, followed by the least significant half.

Figure 27-8. Master-Serial Functional Timing for 4-Bit Interface



BLW = Beginning of Line Pixel Clock Wait Count

27.5 Register Descriptions

The following sections describe the quick capture interface registers:

- Section 27.5.1 — Quick Capture Interface Control Register 0 (CICR0)
- Section 27.5.2 — Quick Capture Interface Control Register 1 (CICR1)
- Section 27.5.3 — Quick Capture Interface Control Register 2 (CICR2)
- Section 27.5.4 — Quick Capture Interface Control Register 3 (CICR3)
- Section 27.5.5 — Quick Capture Interface Control Register 4 (CICR4)
- Section 27.5.6 — Quick Capture Interface Time-Out Register (CITOR)
- Section 27.5.7 — Quick Capture Interface Status Register (CISR)
- Section 27.5.8 — Quick Capture Interface FIFO Control Register (CIFR)
- Section 27.5.9 — Quick Capture Interface Receive Buffer Registers (CIBRx)

27.5.1 Quick Capture Interface Control Register 0 (CICR0)

CICR0, defined in Table 27-24, contains the quick capture interface enable bit (ENB) in addition to other control bit fields.

To configure and enable the quick capture interface, follow these steps:

1. Disable the capture interface by clearing CICR0[ENB].
2. Write all other control items to registers CICR1–CICR4. Program CICR0 last, configuring all bit fields at the same time with a word write
3. Enable the capture interface by setting CICR0[ENB] with a word write.

Note: Before changing any other control bits in the CICR0–CICR4 registers, be sure to disable the quick capture interface by clearing CICR0[ENB].

CICR0 contains the following control items:

DMA request enable (DMAEN)—Set this bit to enable DMA service requests. When DMAEN is clear, no DMA requests are asserted.

Parity enable (PAR_EN)—Set this bit to enable parity checking in embedded synchronization modes. The protection bits P3, P2, P1, and P0 in the embedded synchronization character are used to detect and correct errors in the F, V and H bits. Multi-bit errors set the status bit CISR[PAR_ERR]. Single-bit errors are corrected and do not set CISR[PAR_ERR]. For more information on the embedded controls, see Section 27.4.1.4.1.

Capture enable for slave mode (SL_CAP_EN)—During slave-parallel operation, setting this bit triggers the slave-mode image capture. The quick capture interface continues with data capture until SL_CAP_EN is cleared. When SL_CAP_EN is cleared during data capture, the quick capture interface continues to capture until the entire frame is completed.

Quick capture interface enable (ENB)—Enables and quickly disables all quick capture interface operations. Clearing ENB disables the quick capture interface, and all quick capture interface pins can be used for general-purpose I/O (see Chapter 24, “General-Purpose I/O Controller”). Setting ENB enables the quick capture interface. Initialize all other CICRx control registers **before** setting

ENB. Program CICR0 last, configuring all bit fields at the same time using a word write. Clearing ENB while the quick capture interface is enabled stops the data acquisition immediately, and the current frame does not complete. For sleep-mode shutdown, use this quick-disable feature. For regular shutdown of the quick capture interface at the end of a frame, use CICR0[DIS].

Note: There are separate maskable interrupts for quick disable (CICR0[QDM]) and regular disable (CICR0[CDM]). The description of these interrupt masks can be found later in this section.

Quick capture interface disable (DIS) —While the quick capture interface is operating, setting DIS causes the interface to finish capturing the current frame and then shut down, signaling the shut-down by setting quick capture disable done (CISR[CDD]). Use a read-modify-write procedure to set DIS, since the current frame requires the other CICR0 configuration items until it completes. Completion of the disable command clears the quick capture interface enable (ENB) bit.

Sensor interface mode (SIM)—Selects the mode for interfacing with the quick capture sensor: master-parallel, master-serial, slave-parallel, embedded-parallel, or embedded-serial. For more information, see Section 27.4.1.

Time-out mask (TOM)—When TOM is clear, the time-out interrupt is enabled, so that if CISR[FTO] is set, an interrupt request is sent to the interrupt controller. If TOM is set, the interrupt is masked.

Receive-data-available mask (RDAVM)—Masks the interrupt requests that are asserted when a receive-data-available condition occurs in any of the FIFOs, as indicated by CISR[RDAV_x]. If RDAVM is clear, the interrupt is enabled, so that whenever a CISR[DAV_x] bit is set, an interrupt request is sent to the interrupt controller. If RDAVM is set, the interrupt is masked.

FIFO-empty mask (FEM)—Masks the interrupt requests that are asserted when a FIFO empty condition occurs in any of the FIFOs, as indicated by CISR[FEMPTY_x]. When FEM is clear, the interrupt is enabled, so that whenever a CISR [FEMPTY_x] bit is set, an interrupt request is sent to the interrupt controller. If FEM is set, the interrupt is masked.

End-of-line mask (EOLM)—Masks the interrupt requests that are asserted at the end of each line. If EOLM is clear, the interrupt is enabled, so that whenever CISR[EOL] is set, an interrupt request is sent to the interrupt controller. If EOLM is set, the interrupt is masked.

Setting EOLM does not affect the current state of CISR[EOL] or the quick capture interface's ability to set and clear CISR[EOL]. It simply blocks the generation of the interrupt request.

Parity-error mask (PERRM)—Masks the interrupt requests that are asserted when a multi-bit error is detected in the embedded synchronization character. If PERRM is clear, the interrupt is enabled, so that whenever CISR[PAR_ERR] is set, an interrupt request is sent to the interrupt controller. When PERRM is set, the interrupt is masked.

Quick-disable mask (QDM)—Intended for use with sleep-mode shutdown. This bit masks the interrupt requests that are asserted after ENB is cleared and the capture of the current pixel is completed. New data acquisition stops immediately, and the current frame is not completed. If QDM is clear, the interrupt is enabled, so that whenever the CISR[CQD] status bit is set, an interrupt request is sent to the interrupt controller. If QDM is set, the interrupt is masked.

Setting QDM does not affect the current state of CISR[CQD] or the quick capture interface's ability to set and clear CISR[CQD]. It simply blocks the generation of the interrupt request.

Quick capture interface disable-done mask (CDM)—Masks the interrupt requests that are asserted after the quick capture interface is disabled and the frame currently being captured has completed. If CDM is clear, the interrupt is enabled, so that whenever the CISR[CDD] status bit is set, an interrupt request is generated. If CDM is set, the interrupt is masked.

Setting CDM does not affect the current state of CISR[CDD] or the quick capture interface's ability to set and clear CISR[CDD]. It simply blocks the generation of the interrupt request.

Use this interrupt to ensure the quick capture interface has been disabled and that the current frame being captured has completed.

Note: Clearing ENB forces a quick disable, and CISR[CDD] is not set. CDM applies only to regular shutdowns using the disable (DIS) bit.

Start-of-frame mask (SOFM)—Masks interrupt requests that are asserted at the start of each frame. If SOFM is clear, the interrupt is enabled, so that whenever CISR[SOF] is set, an interrupt request is sent to the interrupt controller. If SOFM is set, the interrupt is masked.

Setting SOFM does not affect the current state of CISR[SOF] or the quick capture interface's ability to set and clear SOF. It simply blocks the generation of the interrupt request.

End-of-frame mask (EOFM)—Masks interrupt requests that are asserted at the end of each frame. When EOFM is clear, the interrupt is enabled, so that whenever the CISR[EOF] status bit is set, an interrupt request is sent to the interrupt controller. When EOFM is set, the interrupt is masked, so that CISR[EOF] does not generate an interrupt.

Setting EOFM does not affect the current state of CISR[EOF] or the quick capture interface's ability to set and clear CISR[EOF]. It simply blocks the generation of the interrupt request.

FIFO overrun mask (FOM)—Masks interrupt requests that are asserted when a FIFO overrun occurs in any of the FIFOs, as indicated by CISR[IFO_0], CISR[IFO_1], and CISR[IFO_2]. When FOM is clear, the interrupt is enabled, so that whenever CISR[IFO_x] is set, an interrupt request is sent to the interrupt controller. When FOM is set, the interrupt is masked, so that the CISR[IFO_x] status bits do not generate interrupts.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 27-24. CICR0 Bit Definitions (Sheet 1 of 2)

Physical Address 0x5000_0000 — CICR0 — Quick Capture Interface

	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
User Settings																																
Bit	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
	DMA_EN	PAR_EN	SL_CAP_EN	ENB	DIS	SIM			reserved														TOM	RDAVM	FEM	EOLM	PERRM	QDM	CDM	SOFM	EOFM	FOM
Reset	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1	1	1	1	1	1	1	1	1

Bits	Access	Name	Description
31	R/W	DMA_EN	DMA Request Enable 0 = DMA requests disabled. 1 = DMA requests enabled.
30	R/W	PAR_EN	Parity Enable 0 = Parity check disabled for embedded modes. 1 = Parity check enabled for embedded modes.
29	R/W	SL_CAP_EN	Quick Capture Enable for Slave Mode 0 = Quick capture interface data capture disabled in slave mode. 1 = Quick capture interface actively captures data in slave mode.
28	R/W	ENB	Quick Capture Interface Enable (and Quick Disable) 0 = Quick capture interface disabled. 1 = Quick capture interface enabled.
27	R/W	DIS	Quick Capture Interface Disable 0 = Quick capture interface has not been programmed with the sequence of being enabled and then subsequently disabled. 1 = Quick capture interface has been disabled, or is in the process of disabling.
26:24	R/W	SIM	Sensor interface Mode 0b000 = Master-parallel mode 0b001 = Slave-parallel mode 0b010 = Master-serial mode 0b011 = Embedded-parallel mode 0b100 = Embedded-serial mode Others: reserved
23:10	—	—	reserved
9	R/W	TOM	Time-Out Mask 0 = Time-out condition generates an interrupt. 1 = Time-out condition does not generate an interrupt.
8	R/W	RDAVM	Receive-Data-Available Mask 0 = Receiver data available (trigger level reached) interrupt enabled. 1 = Receiver data available (trigger level reached) interrupt disabled.
7	R/W	FEM	FIFO-Empty Mask 0 = FIFO Empty condition generates an interrupt. 1 = FIFO Empty condition does not generate an interrupt.
6	R/W	EOLM	End-of-Line Mask 0 = CISR[EOL] can generates an interrupt 1 = CISR[EOL] does not generate an interrupt

Table 27-24. CICR0 Bit Definitions (Sheet 2 of 2)

Physical Address 0x5000_0000 — CICR0 — Quick Capture Interface

	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
User Settings																																
Bit	DMA_EN	PAR_EN	SL_CAP_EN	ENB	DIS	SIM			reserved														TOM	RDAVM	FEM	EOLM	PERRM	QDM	CDM	SOFM	EOFM	FOM
Reset	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1	1	1	1	1	1	1	1	1

Bits	Access	Name	Description
5	R/W	PERRM	Parity-Error Mask 0 = CISR[PAR_ERR] can generate an interrupt. 1 = CISR[PAR_ERR] does not generate an interrupt.
4	R/W	QDM	Quick-Disable Mask 0 = CISR[CQD] can generate an interrupt. 1 = CISR[CQD] does not generate an interrupt.
3	R/W	CDM	Disable-Done Mask 0 = CISR[CDD] can generate an interrupt. 1 = CISR[CDD] does not generate an interrupt.
2	R/W	SOFM	Start-of-Frame Mask 0 = CISR[SOF] can generate an interrupt. 1 = CISR[SOF] does not generate an interrupt.
1	R/W	EOFM	End-of-Frame Mask 0 = CISR[EOF] can generate an interrupt. 1 = CISR[EOF] does not generate an interrupt.
0	R/W	FOM	FIFO Overrun Mask 0 = FIFO overrun errors generate interrupts. 1 = FIFO overrun errors do not generate interrupts.

27.5.2 Quick Capture Interface Control Register 1 (CICR1)

CICR1, detailed in Table 27-26, defines the pixel format, color space, bits per pixel, data format, precision control and the width of the interface between the quick capture interface and the image sensor. All control bits in CICR1 must be programmed before setting CICR0[ENB].

Transparency bit (TBIT)—When conversion from RGB format to RGBT is enabled (RGBT_CONV = 01 or 10), TBIT specifies the transparency. TBIT is ignored when conversion to RGBT is not enabled.

RGBT Conversion (RGBT_CONV)—Specifies the type of conversion to be performed on RGB data to generate RGBT data. Note that conversion to RGBT format is supported only for 24 bits per pixel (RGB_BPP = 100 and RGB_CONV = 000) and 15 bits per pixel (RGB_BPP = 001 or (RGB_BPP = 100 and RGB_CONV = 011). For all other settings, RGBT_CONV is ignored.

Pixels per line (PPL)—Specifies the number of pixels in each line or row of the frame. PPL is an 11-bit value that represents between 1 to 2048 pixels per line. Program PPL with the required number of pixels per line minus 1. PPL is used by the quick capture interface in master and slave modes to capture the correct number of pixels in a line.

RGB bits-per-pixel conversion (RGB_CONV)—This bit field specifies the type of conversion to be performed on RGB 8:8:8 pixel conversion. This field is used only when RGB color space is selected by COLOR_SP field and RGB_BPP field is set for 24 bits per pixel. In all other settings, the value of RGB_CONV is ignored. See Section 27.4.5.4 for more details.

RGB format (RGB_F)—When RGB Color space is selected (COLOR_SP = 01), and RGB_BPP field is set for 24 bits (RGB_BPP = 100 and RGB_CONV = 000) or 18 bits (RGB_BPP = 011 or (RGB_BPP = 100 and RGB_CONV = 001)) per pixel, RGB_F bit selects the format used for storing the captured data. When RGB_F bit is set, pixel data is stored in a packed format. When this bit is clear, the captured data is stored in unpacked format. Note that for settings other than those mentioned above, RGB_F is ignored.

YCbCr format (YCBCR_F)—When YCbCr color space is selected (COLOR_SP = 10), YCBCR_F bit selects the pixel format used for YCbCr data. If YCBCR_F is set, the captured data is arranged in a planarized format. In this case each of the image component is stored in a separate memory location. If YCBCR_F is clear, captured data is arranged in a packed format where each of the components are stored as they are received. See Section 27.4.5.3 for more details.

Note: When YCbCr color space is selected, the quick capture interface supports only 8 bits per pixel.

RGB bits per pixel (RGB_BPP)—When RGB color space is selected (COLOR_SP = 01), RGB_BPP specifies the number of bits per pixel.

Raw bits per pixel (RAW_BPP)—When raw color space is selected (COLOR_SP = 00), RAW_BPP specifies the number of bits per pixel.

Color space (COLOR_SP)—Specifies the color space used for the image data. The supported color spaces are RAW, RGB and YCbCr.

Data width (DW)—The image sensor can interface with the quick capture interface using a data width of 4 bits, 5 bits, 8 bits, 9 bits, or 10 bits, based on the mode of operation. Table 27-25 lists the number of bits per pixel (BPP) that are compatible with the data width for each mode of operation.

Table 27-25. Data Width/Bits per Pixel Compatibility

Mode of Operation	Data Width, Bits				
	4	5	8	9	10
Master-Parallel (MP)	—	—	all	all	all
Master-Serial (MS)	8 bpp	10 bpp	—	—	—
Slave-Parallel (SP)	—	—	all	all	all
Embedded-Parallel (EP)	—	—	all	—	—
Embedded-Serial (ES)	8 bpp	—	—	—	—

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 27-26. CICR1 Bit Definitions (Sheet 1 of 2)

Physical Address 0x5000_0004 — **CICR1** — **Quick Capture Interface**

User Settings

Bit	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
	TBIT	RGBT_CONV		reserved			PPL											RGB_CONV			RGB_F	YCBCR_F	RGB_BPP			RAW_BPP		COLOR_SP		DW		
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

Bits	Access	Name	Description
31	R/W	TBIT	Transparency Bit 0 = Transparency bit of 0 is inserted if RGBT conversion is selected 1 = Transparency bit of 1 is inserted if RGBT conversion is selected
30:29	R/W	RGBT_CONV	RGBT Conversion 00 = No conversion to RGBT 01 = From RGB 8:8:8 to RGBT 8:8:8 10 = From RGB 5:5:5 to RGBT 5:5:5 11 = reserved
28:26	—	—	reserved
25:15	R/W	PPL	Pixels per Line for the Frame Value from 0 through 2047. PPL specifies the number of pixels contained within each line of the image. Actual pixels per line = PPL+1.
14:12	R/W	RGB_CONV	RGB Bits per Pixel Conversion 000 = No conversion from RGB 8:8:8 001 = From RGB 8:8:8 to RGB 6:6:6 010 = From RGB 8:8:8 to RGB 5:6:5 011 = From RGB 8:8:8 to RGB 5:5:5 100 = From RGB 8:8:8 to RGB 4:4:4 Others: reserved
11	R/W	RGB_F	RGB Format 0 = Unpacked format 1 = Packed format
10	R/W	YCBCR_F	YCbCr Format 0 = Packed format 1 = Planarized format
9:7	R/W	RGB_BPP	RGB Bits per Pixel 000 = 12 bits per pixel (4:4:4) 001 = 15 bits per pixel (5:5:5) 010 = 16 bits per pixel (5:6:5) 011 = 18 bits per pixel (6:6:6) 100 = 24 bits per pixel (8:8:8) Others: reserved

Table 27-26. CICR1 Bit Definitions (Sheet 2 of 2)

Physical Address 0x5000_0004 — CICR1 — Quick Capture Interface

	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
User Settings																																
Bit	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
	TBIT	RGBT_CONV	reserved				PPL										RGB_CONV				RGB_F	YCBCR_F	RGB_BPP		RAW_BPP		COLOR_SP		DW			
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

Bits	Access	Name	Description
6:5	R/W	RAW_BPP	Raw Bits per Pixel 00 = 8 bits per pixel 01 = 9 bits per pixel 10 = 10 bits per pixel 11 = reserved
4:3	R/W	COLOR_SP	Color Space 00 = Raw 01 = RGB 10 = YCbCr 11 = reserved
2:0	R/W	DW	Data Width 000 = 4 bits wide data from the sensor 001 = 5 bits wide data from the sensor 010 = 8 bits wide data from the sensor 011 = 9 bits wide data from the sensor 100 = 10 bits wide data from the sensor Others: reserved

27.5.3 Quick Capture Interface Control Register 2 (CICR2)

CICR2, detailed in Table 27-27, defines the wait counts at the beginning and end of a line, width of horizontal sync signal and number of frames to skip at the beginning of a frame sequence.

Beginning-of-line pixel clock wait count (BLW)—Specifies the number of pixel clocks to wait from the active edge of CIF_LV before starting the data capture. This wait count is used for both master and slave modes of operation. BLW generates a wait period ranging from 0 to 255 pixel clock cycles.

End-of-line pixel clock wait count (ELW)—This 8-bit field specifies the number of pixel clocks to wait at the end of each line or row of pixels before pulsing the CIF_LV pin in slave mode. This field is not used when the quick capture interface is in master mode of operation. Program ELW with the required wait count minus 1. The resulting wait period ranges from 1 to 255 pixel clock cycles.

Horizontal sync width (HSW)—This 6-bit field specifies the pulse width (in CIF_PCLK cycles) of CIF_LV in slave mode. This field is not used when quick capture interface is in master mode of operation. CIF_LV is asserted each time quick capture interface begins to capture a line or row of pixels. HSW can be programmed to generate a CIF_LV width ranging from 1 to 64 pixel clock periods. Users must program HSW with the required number of pixel clocks minus one. Also note that the polarity (active and inactive state) of the CIF_LV pin is programmed using the horizontal sync polarity (HSP) bit in CICR4.

Beginning-of-frame pixel clock wait count (BFPW)—This 6-bit field specifies the delay between the assertion of CIF_FV and CIF_LV in slave mode. BFPW generates a delay ranging from 1 to 64 pixel clock cycles. Users must program BFPW with the preferred number of pixel clocks minus one.

Frame stabilization wait count (FSW)—This 3-bit field is used in slave mode only to specify the number of frames before the sensor starts sending valid frames. The quick capture interface starts capturing data only after FSW count has elapsed. FSW count generates a wait period ranging from 0 to 7 frames.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 27-27. CICR2 Bit Definitions

Physical Address 0x5000_0008 — CICR2 — Quick Capture Interface

	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
User Settings																																
Bit	BLW								ELW								HSW						reserved	BFPW					FSW			
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

Bits	Access	Name	Description
31:24	R/W	BLW	Beginning-of-Line Pixel Clock Wait Count Allowed values are 0 to 255. Specifies the delay from active edge of CIF_LV to the first valid data at the interface, in terms of pixel clock periods.
23:16	R/W	ELW	End-of-Line Pixel Clock Wait Count Allowed values are 0 to 255. ELW + 1 specifies the number of pixel clock periods to wait after the end of a line, before beginning the next line.
15:10	R/W	HSW	Horizontal Sync Pulse Width Allowed values are 0 to 63. In slave mode, HSW + 1 specifies the number of pixel clock periods to pulse CIF_LV at the beginning of each line. CIF_LV pulse width = (HSW+1).
9	—	—	reserved
8:3	R/W	BFPW	Beginning-of-Frame Pixel Clock Wait Count Allowed values are 0 to 63. In slave mode, BFPW +1 specifies the number of pixel clock periods to wait from CIF_FV before asserting CIF_LV.
2:0	R/W	FSW	Frame Stabilization Wait Count Value from 0 through 7. In slave mode, specifies the number of frame clock periods to wait before valid data is sent out by the sensor.

27.5.4 Quick Capture Interface Control Register 3 (CICR3)

CICR3, detailed in Table 27-28, defines the wait counts at the beginning and end of a frame, width of vertical sync signal, and number of lines per frame.

Beginning-of-frame line clock wait count (BFW)—This 8-bit field specifies the number of line clocks (CIF_LV) to wait at the beginning of each frame. The BFW count starts after the CIF_FV signal is asserted for a new frame. After CIF_FV assertion, the value in BFW counts the number of line clock periods to wait before starting the frame capture. BFW generates a wait period ranging from 0 to 255 extra line clock cycles (BFW = 0x00 disables the wait count). BFW is used by the quick capture interface in both master and slave modes of operation to delay the data capture. Note that the line clock signal, CIF_LV, does toggle during the generation of the BFW line-clock wait periods. BFW count is not used in embedded mode (internal synchronization).

End-of-frame line clock wait count (EFW)—This 8-bit field specifies the number of line clocks (CIF_LV) to wait at the end of each frame when the quick capture interface is operating in slave mode. Once a complete frame of pixels is captured by quick capture interface (as indicated by pixels-per-line and lines-per-frame counters), the value in EFW counts the number of line-clock periods to wait. After the count has elapsed, the CIF_FV signal is pulsed to start the next frame capture. EFW generates a wait period ranging from 0 to 255 line clock cycles (setting EFW = 0x00

disables the EOF wait count). Note that the line clock pin, CIF_LV, does not toggle during the generation of the EFW line clock periods. EFW is not used if quick capture interface is operating in master mode.

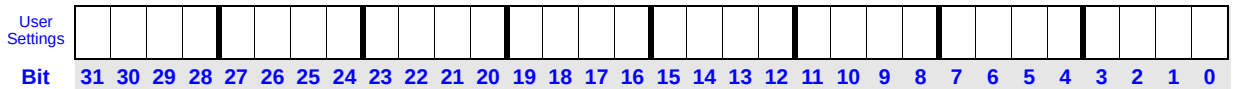
Vertical sync pulse width (VSW)—This 5-bit field specifies the width of CIF_FV in slave mode. This field is not used when the quick capture interface is in master mode of operation. CIF_FV is asserted each time quick capture interface begins to capture a frame. VSW can be programmed to generate a CIF_FV width ranging from 1 to 32 pixel clock periods. Program VSW with the required number of pixel clocks minus one. Also note that the polarity (active and inactive state) of the line clock pin is programmed using CICR4[VSP].

Lines per frame (LPF)—This 11-bit field specifies the number of lines in one frame of data.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 27-28. CICR3 Bit Definitions

Physical Address 0x5000_000C	CICR3	Quick Capture Interface



Bits	Access	Name	Description
31:24	R/W	BFW	Beginning-of-Frame Line Clock Wait Count Allowed values are 0 to 255. Specifies the number of line clock periods to wait at the beginning of a frame before the first set of pixels are captured.
23:16	R/W	EFW	End-of-Frame Line Clock Wait Count Allowed values are 0 to 255. In slave mode, specifies the number of line clock periods to add to the end of each frame.
15:11	R/W	VSW	Vertical Sync Pulse Width Allowed values are 0 through 31. In slave mode, VSW + 1 specifies the number of pixel clock periods to pulse the CIF_FV pin at the beginning of each frame. CIF_FV pulse width = (VSW+1).
10:0	R/W	LPF	Lines per Frame Allowed values are 0 to 2047. LPF+1 specifies the number of lines per frame. Lines/frame = (LPF+1).

27.5.5 Quick Capture Interface Control Register 4 (CICR4)

CICR4, defined in Table 27-29, contains different bit fields to control various functions within the quick capture interface.

Note: Before changing the state of any field within this register, disable the quick capture interface by clearing CICR0[ENB]. CICR0[DIS] may also be used to disable the quick capture interface.

Note: Clearing both MCLK_EN and PCLK_EN is an invalid clock configuration. No data is captured by the capture interface when both are clear.

Master Clock Data Capture Delay (MCLK_DLY)—When the sensor uses an MCLK supplied by the capture interface but does not supply a pixel clock to capture interface, the master clock data capture delay (MCLK_DLY) specifies the number of CICLK cycles to wait from the active edge of MCLK (as specified by pixel clock polarity, PCP) before sampling the pixel data. MCLK_DLY delays the data sampling only when CICR4[MCLK_EN] is set and CICR4[PCLK_EN] is clear. For other settings of MCLK_EN and PCLK_EN, the MCLK_DLY value is ignored. Programming the MCLK_DLY value allows software to compensate for the data skew from MCLK when the sensor does not supply a pixel clock.

Pixel clock enable (PCLK_EN)—Specifies whether or not the sensor is supplying a pixel clock to the quick capture interface. When the sensor does not supply a pixel clock to quick capture interface, PCLK_EN must be clear. The quick capture interface uses MCLK to clock in pixel data. This bit needs to be set if the sensor supplies a pixel clock.

Pixel clock polarity (PCP)—Selects which edge of the pixel clock samples data on the CIF_DD data pins. When PCP is clear, data is sampled on the CIF_DD data pins on the rising edge of CIF_PCLK. When PCP is set, data is sampled on the CIF_DD data pins on the falling edge of CIF_PCLK.

Horizontal sync polarity (HSP)—Selects the active and inactive states of the line valid signal (CIF_LV). When HSP is set, CIF_LV is active high and inactive low. When HSP is clear, CIF_LV is active low and inactive high. This polarity control applies in both master and slave modes of operation.

Vertical sync polarity (VSP)—Selects the active and inactive states of the frame valid signal (CIF_FV). When VSP is clear, CIF_FV is active high and inactive low. When VSP is set, CIF_FV is active low and inactive high. This polarity control applies in both master and slave modes of operation.

Master clock enable (MCLK_EN)—Clock enable for CIF_MCLK supplied by the quick capture interface to the sensor. MCLK_EN needs to be set to supply the clock at the rate specified by CICR4[DIV]. Turn off CIF_MCLK by clearing MCLK_EN.

Frame capture rate (FR_RATE)—Defines the rate at which the incoming frames are captured by the quick capture interface. By programming this field, users can capture frames at a rate lower than the output rate of the sensor.

Clock divisor (DIV)—Selects the frequency of the master clock (CIF_MCLK) supplied to the image sensor. DIV can be of any value from 0 to 255 and generates a range of MCLK frequencies from CICLK/2 to CICLK/512, where CICLK is the clock input to the capture interface. CICLK is the same frequency as the internal LCD clock. The master clock frequency is: determined by:.

$$MCLK = \frac{CICLK}{2(DIV + 1)}$$

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 27-29. CICR4 Bit Definitions

Physical Address 0x5000_0010 — CICR4 — Quick Capture Interface

Bit	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
User Settings																																
	reserved					MCLK_DLY			PCLK_EN	PCP	HSP	VSP	MCLK_EN	reserved								FR_RATE			DIV							
Reset	?	?	?	?	?	?	?	?	0	0	0	0	0	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:27	—	—	reserved
26:24	R/W	MCLK_DLY	MCLK Data Capture Delay Value (from 0 to 7) specifies the delay from MCLK active edge to the pixel data-sampling point in terms of CICLK cycles.
23	R/W	PCLK_EN	Pixel Clock Enable 0 = CIF_PCLK is not supplied by the sensor. 1 = CIF_PCLK is supplied by the sensor.
22	R/W	PCP	Pixel Clock Polarity 0 = Data is sampled on the quick capture interface's data pins on the rising edge of CIF_PCLK. 1 = Data is sampled on the quick capture interface's data pins on the falling edge of CIF_PCLK.
21	R/W	HSP	Horizontal Sync Polarity 0 = CIF_LV pin is active high and inactive low. 1 = CIF_LV pin is active low and inactive high.
20	R/W	VSP	Vertical Sync Polarity 0 = CIF_FV pin is active high and inactive low. 1 = CIF_FV pin is active low and inactive high.
19	R/W	MCLK_EN	MCLK Enable 0 = MCLK is not supplied by quick capture interface to the sensor. 1 = MCLK is supplied by quick capture interface to the sensor.
18:11	—	—	reserved
10:8	R/W	FR_RATE	Frame Capture Rate 0b000 = Capture all incoming frames. 0b001 = Capture 1 out of every 2 frames. 0b010 = Capture 1 out of every 3 frames. 0b011 = Capture 1 out of every 4 frames. 0b100 = Capture 1 out of every 5 frames. 0b101 = Capture 1 out of every 6 frames. 0b110 = Capture 1 out of every 7 frames. 0b111 = Capture 1 out of every 8 frames.
7:0	R/W	DIV	Clock Divisor Value from 0 to 255. Specifies the frequency of the clock supplied to the sensor MCLK. CICLK is the same frequency as the internal LCD clock. MCLK Clock Frequency = CICLK/2(DIV+1).

27.5.6 Quick Capture Interface Time-Out Register (CITOR)

CITOR, defined in Table 27-30, contains the time-out value used to specify the period of inactivity in the quick capture interface FIFOs in the number of CICLK cycles.

The TIMEOUT field sets the time-out interval described in Section 27.4.4.4. When the CITOR[TIMEOUT] value is 0, no time-out occurs, and CISR[FTO] is not set. The time-out interval is given by the equation:

$$Time-out\ interval = \frac{CITOR[TIMEOUT]}{CICLK\ frequency}$$

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 27-30. CITOR Bit Definitions

Physical Address 0x5000_001C — **CITOR** — **Quick Capture Interface**

	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
User Settings																																
Bit	TIMEOUT																															
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

Bits	Access	Name	Description
31:0	R/W	TIMEOUT	Time-Out Value from 0 through (2^{31} – 1) that defines the time-out interval. See the equation given in Section 27.5.6.

27.5.7 Quick Capture Interface Status Register (CISR)

CISR, defined in Table 27-31, contains status flag bits for hardware-detected events. Unless masked, each of these status bits signals an interrupt request to the interrupt controller for as long as the bit remains set. When the status bit is cleared, the interrupt is cleared.

To clear a CISR status bit, write 0b1 to it. Writing 0b0 to a status bit has no effect.

To mask the interrupt generated by a CISR status bit, refer to register CICR0, which contains the masks; see Section 27.5.1.

FIFO time-out (FTO)—When set, the quick capture interface FIFO has been idle (no samples received) for a period of time defined by the value programmed in register CITOR. To mask this interrupt, set CICR0[TOM].

Receive data available (RDAV_2, RDAV_1, RDAV_0)—When set, the corresponding FIFO's threshold level has been reached. Threshold levels for the channel 1 and channel 2 FIFOs are fixed (see Section 27.4.4.2). To program the threshold level for the channel 0 FIFO, use CIFR[THL_0]. To enable or mask the interrupt generated by these bits, use CICR0[RDAVM].

FIFO empty (FEMPTY_2, FEMTPY_1, FEMTPY_0)—When set, DMA has finished transferring all the contents of the respective channel's FIFO. To mask this interrupt, set CICR0[FEM].

End of line (EOL)—When set, the quick capture interface has finished capturing all pixels in a line. To enable or mask the interrupt generated by this bit, use CICR0[EOLM].

Parity error (PAR_ERR)—In embedded synchronization modes, and if parity checking is enabled (CICR0[PAR_EN] set), PAR_ERR is set when a multi-bit error is detected in the synchronization character. For more information, see CICR0[PAR_EN] in Section 27.5.1. To mask this interrupt, set CICR0[PERRM].

Quick capture interface quick disable (CQD)—Set when CICR0[ENB] has been cleared to disable the quick capture interface quickly. When CQD is set, the quick capture interface stops capturing data and quits driving the quick capture interface pins. The DMA controller and the core can still read the FIFOs. The core can still access the quick capture interface control and status registers.

This disabling method allows quick entry into sleep and deep-sleep modes. Monitor CQD to ensure that the quick capture interface is disabled. To enable this interrupt, set CICR0[QDM].

Quick capture interface disable done (CDD)—Set after CICR0[DIS] has been set to disable the quick capture interface in an orderly manner *and* after capture of the last frame has finished. When CDD is set, the quick capture interface stops capturing data and stops driving the quick capture interface pins. The DMA controller and the core can still read the FIFOs. The core can still access the quick capture interface control and status registers.

This disabling method allows orderly entry into sleep and deep-sleep modes. Software can monitor CDD and FEMPTY_x to ensure that the last frame has been transferred from quick capture interface. To enable or mask the interrupt generated by this bit, use CICR0[CDM].

Start of frame (SOF)—Set when the quick capture interface detects the start of a frame. To mask this interrupt, set CICR0[SOFM].

End of frame (EOF)—Set when the quick capture interface has finished capturing all lines in a frame. To mask this interrupt, set CICR0[EOFM].

Input FIFO overrun (IFO_2, IFO_1, IFO_0)—When set, an overrun condition has occurred in the respective input FIFO. The overrun happens when the FIFO is full and an attempt is made to write a new sample to the FIFO. This new sample and following data samples are not loaded into the FIFO. To mask this interrupt, set CICR0[FOM].

Table 27-31. CISR Bit Definitions (Sheet 1 of 2)

Physical Address 0x5000_0014 — CISR — Quick Capture Interface

Bit	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
User Settings																																
	reserved																FTO	RDAV_2	RDAV_1	RDAV_0	FEMPTY_2	FEMPTY_1	FEMPTY_0	EOL	PAR_ERR	CQD	CDD	SOF	EOF	IFO_2	IFO_1	IFO_0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15	R/WC†	FTO	FIFO Time-Out 0 = FIFO time-out has not occurred. 1 = FIFO time-out has occurred.
14	R/WC†	RDAV_2	Channel 2 Receive Data Available 0 = Received data is not available in channel 2 FIFO. 1 = Received data is available in channel 2 FIFO.
13	R/WC†	RDAV_1	Channel 1 Receive Data Available 0 = Received data is not available in channel 1 FIFO. 1 = Received data is available in channel 1 FIFO.
12	R/WC†	RDAV_0	Channel 0 Receive Data Available 0 = Received data is not available in channel 0 FIFO. 1 = Received data is available in channel 0 FIFO.
11	R/WC†	FEMPTY_2	Channel 2 FIFO Empty 0 = Channel 2 FIFO is not empty. 1 = Channel 2 FIFO is empty.
10	R/WC†	FEMPTY_1	Channel 1 FIFO Empty 0 = Channel 1 FIFO is not empty. 1 = Channel 1 FIFO is empty.
9	R/WC†	FEMPTY_0	Channel 0 FIFO Empty 0 = Channel 0 FIFO is not empty. 1 = Channel 0 FIFO is empty.
8	R/WC†	EOL	End of Line 0 = Quick capture interface has not received an end-of-line. 1 = Quick capture interface has received an end-of-line.
7	R/WC†	PAR_ERR	Parity Error 0 = No multi-bit parity error is detected in the embedded sync character. 1 = Multi-bit parity error is detected in the embedded sync character.
6	R/WC†	CQD	Quick Capture Interface Quick Disable 0 = The quick capture interface has not been quick-disabled. 1 = The quick capture interface has been quick-disabled by clearing CICR0[ENB].
5	R/WC†	CDD	Quick Capture Interface Disable Done 0 = The quick capture Interface has not been disabled. 1 = The quick capture interface has been disabled and the active frame has completed.
4	R/WC†	SOF	Start of Frame 0 = The quick capture interface has not received a start of frame. 1 = The quick capture interface has received a start of frame.

Table 27-31. CISR Bit Definitions (Sheet 2 of 2)

Physical Address 0x5000_0014 — CISR — Quick Capture Interface

User Settings																															
Bit	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																FTO	RDAV_2	RDAV_1	RDAV_0	FEMPTY_2	FEMPTY_1	FEMPTY_0	EOL	PAR_ERR	CQD	CDD	SOF	EOF	IFO_2	IFO_1	IFO_0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
3	R/WC†	EOF	End of Frame 0 = The capture Interface has not received an end of frame. 1 = The capture Interface has received an end of frame.
2	R/WC†	IFO_2	FIFO Overrun for Channel 2 0 = Overrun has not occurred in channel 2. 1 = Channel 2 input FIFO is full, and an attempts is made to write a new sample to the FIFO
1	R/WC†	IFO_1	FIFO Overrun for Channel 1 0 = Overrun has not occurred in channel 1. 1 = Channel 1 input FIFO is full, and an attempts is made to write a new sample to the FIFO
0	R/WC†	IFO_0	FIFO Overrun for Channel 0 0 = Overrun has not occurred in channel 0. 1 = Channel 0 input FIFO is full, and an attempts is made to write a new sample to the FIFO

† To clear this bit, write 0b1 to it.

27.5.8 Quick Capture Interface FIFO Control Register (CIFR)

CIFR, defined in Table 27-32, contains bits that control the behavior of the input FIFOs and bit fields that indicate the number of bytes in the FIFOs.

FIFO level (FLVL0, FLVL1, FLVL2)—Each bit field indicates the number of bytes in the FIFO for the corresponding channel. Software can poll the FLVLx values to determine the number of bytes remaining to be read in each of the FIFOs.

Threshold level for channel 0 (THL_0)—Specifies the input FIFO level that can trigger a DMA service request for channel 0. If CICR0[DMAEN] is set, DMA service requests are issued when the FIFO trigger levels are reached.

Reset input FIFO (RESETF)— When set, the FIFO pointers of all three FIFOs are reset. This bit is automatically cleared after resetting the pointers.

FIFO enable (FEN0, FEN1, FEN2)—When set, each of these bits enables the FIFO for the corresponding channel. Only the planarized YCbCr mode of operation requires that all three FIFOs be enabled (see Section 27.4.4 for information on FIFO usage). In this mode, the channel 0 FIFO holds the Y component, the channel 1 FIFO holds the Cb component, and the channel 2 FIFO holds the Cr component. All other modes use only the channel 0 FIFO to hold captured data.

NOTE: The programming of the FIFO enable bits FEN0, FEN1, and FEN2 is no longer required. The appropriate FIFOs are automatically enabled and utilized based on the programmed mode.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 27-32. CIFR Bit Definitions

Physical Address 0x5000_0018 — **CIFR** — **Quick Capture Interface**

Bit	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
User Settings																																
	reserved		FLVL2						FLVL1								FLVL0								reserved		THL_0		RESETF	FEN2	FEN1	FEN0
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

Bits	Access	Name	Description
31:30	—	—	reserved
29:23	R	FLVL2	FIFO 2 Level Value from 0 through 64 indicates the number of bytes in the channel 2 FIFO.
22:16	R	FLVL1	FIFO 1 Level Value from 0 through 64 indicates the number of bytes in the channel 1 FIFO.
15:8	R	FLVL0	FIFO 0 Level Value from 0 through 128 indicates the number of bytes in the channel 0 FIFO.
7:6	—	—	reserved
5:4	R/W	THL_0	Threshold Level for Channel 0 FIFO 0b00 = 32 bytes or more in FIFO_0 causes a DMA request. 0b01 = 64 bytes or more in FIFO_0 causes a DMA request. 0b10 = 96 bytes or more in FIFO_0 causes a DMA request. 0b11 = reserved
3	R/W	RESETF	Reset input FIFOs 0 = No effect. 1 = Input FIFOs are cleared. After clearing, this bit is automatically cleared.
2	R/W	FEN2	FIFO Enable for channel 2 0 = FIFO not enabled for channel 2. 1 = FIFO enabled for channel 2. **NOTE**: The programming of FEN2 is no longer required. The FIFO is automatically enabled and utilized based on the programmed mode.
1	R/W	FEN1	FIFO Enable for Channel 1 0 = FIFO not enabled for channel 1. 1 = FIFO enabled for channel 1. **NOTE**: The programming of FEN1 is no longer required. The FIFO is automatically enabled and utilized based on the programmed mode.
0	R/W	FEN0	FIFO Enable for Channel 0 0 = FIFO not enabled for channel 0. 1 = FIFO enabled for channel 0. **NOTE**: The programming of FEN0 is no longer required. The FIFO is automatically enabled and utilized based on the programmed mode.

27.5.9 Quick Capture Interface Receive Buffer Registers (CIBRx)

CIBRx holds the data word pointed to by the read pointer of the channel x FIFO, where *x* corresponds to channel 0, 1, or 2.

Four bytes (byte 3 through byte 0) of the channel *x* FIFO entry can be read from the CIBR*x* location before the read pointer is incremented to point to the next FIFO entry.

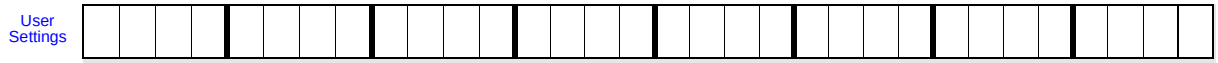
These are read-only registers. Writes to these registers may cause a target abort.

Table 27-33. CIBR0/1/2 Bit Definitions

Physical Address
0x5000_0028
0x5000_0030
0x5000_0038

CIBR0
CIBR1
CIBR2

Quick Capture Interface



Bit	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
	Byte 3								Byte 2								Byte 1								Byte 0							
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

Bits	Access	Name	Description
31:24	R	Byte 3	Byte 3 of the captured data in channel x, where *x* = CIBR*x*
23:16	R	Byte 2	Byte 2 of the captured data in channel x, where *x* = CIBR*x*
15:8	R	Byte 1	Byte 1 of the captured data in channel x, where *x* = CIBR*x*
7:0	R	Byte 0	Byte 0 of the captured data in channel x, where *x* = CIBR*x*

27.6 Register Summary

Table 27-34 summarizes the registers and memory mapping associated with the quick capture interface.

These registers must be mapped as non-cacheable and non-bufferable. They can be accessed only with word accesses. They are grouped together in one page and, therefore, all have the same memory protections.

Table 27-34. Quick Capture Interface Register Summary

Physical Address	Name	Description	Page
0x5000_0000	CICR0	Quick Capture Interface Control register 0	27-24
0x5000_0004	CICR1	Quick Capture Interface Control register 1	27-28
0x5000_0008	CICR2	Quick Capture Interface Control register 2	27-32
0x5000_000C	CICR3	Quick Capture Interface Control register 3	27-33
0x5000_0010	CICR4	Quick Capture Interface Control register 4	27-34
0x5000_0014	CISR	Quick Capture Interface Status register	27-37
0x5000_0018	CIFR	Quick Capture Interface FIFO Control register	27-40
0x5000_001C	CITOR	Quick Capture Interface Time-Out register	27-37
0x5000_0020 – 0x5000_0024	—	reserved	—
0x5000_0028	CIBR0	Quick Capture Interface Receive Buffer register 0 (channel 0)	27-42
0x5000_002C	—	reserved	—
0x5000_0030	CIBR1	Quick Capture Interface Receive Buffer register 1 (channel 1)	27-42
0x5000_0034	—	reserved	—
0x5000_0038	CIBR2	Quick Capture Interface Receive Buffer register 2 (channel 2)	27-42
0x5000_003C– 0x53FF_FFFC	—	reserved	—

Memory Map and Registers 28

This chapter describes the memory-mapped and coprocessor registers available in the PXA27x processor, as follows:

- Section 28.1 — Overview
- Section 28.2 — System Bus Unit Registers
- Section 28.3 — Peripheral Module Registers

28.1 Overview

This chapter describes the physical address map, memory-mapped registers, and coprocessor registers of the PXA27x processor. Software can use the memory management unit included in the Intel XScale® core to map these into a specific virtual address map as required.

The following diagrams describe the top-level memory map and decoding:

- Figure 28-1 shows the physical address map decode regions.
- Figure 28-2 shows the summary memory map for region 0x0000_0000–0x7FFF_FFFF.
- Figure 28-3 shows the summary memory map for region 0x8000_0000–0xFFFF_FFFF.

Note: Accessing reserved portions of the memory map shown in Figure 28-2 and Figure 28-3 results in a data-abort exception. Accessing reserved portions of a particular peripheral's address space does not cause a data-abort exception, but the data returned is undefined.

The PC Card interface is divided into Socket 0 and Socket 1 space. These two partitions are each subdivided into I/O, memory, and attribute space. Each socket is allocated 256 Mbytes of memory space.

Figure 28-1. Physical Address Map Decode Regions

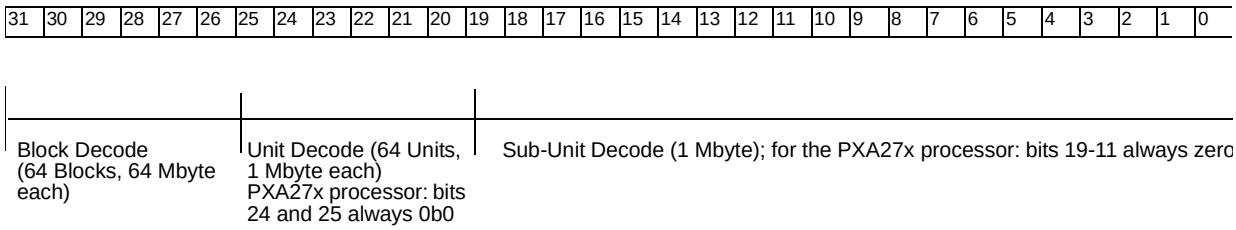


Figure 28-2. Memory Map—0x0000_0000–0x7FFF_FFFF

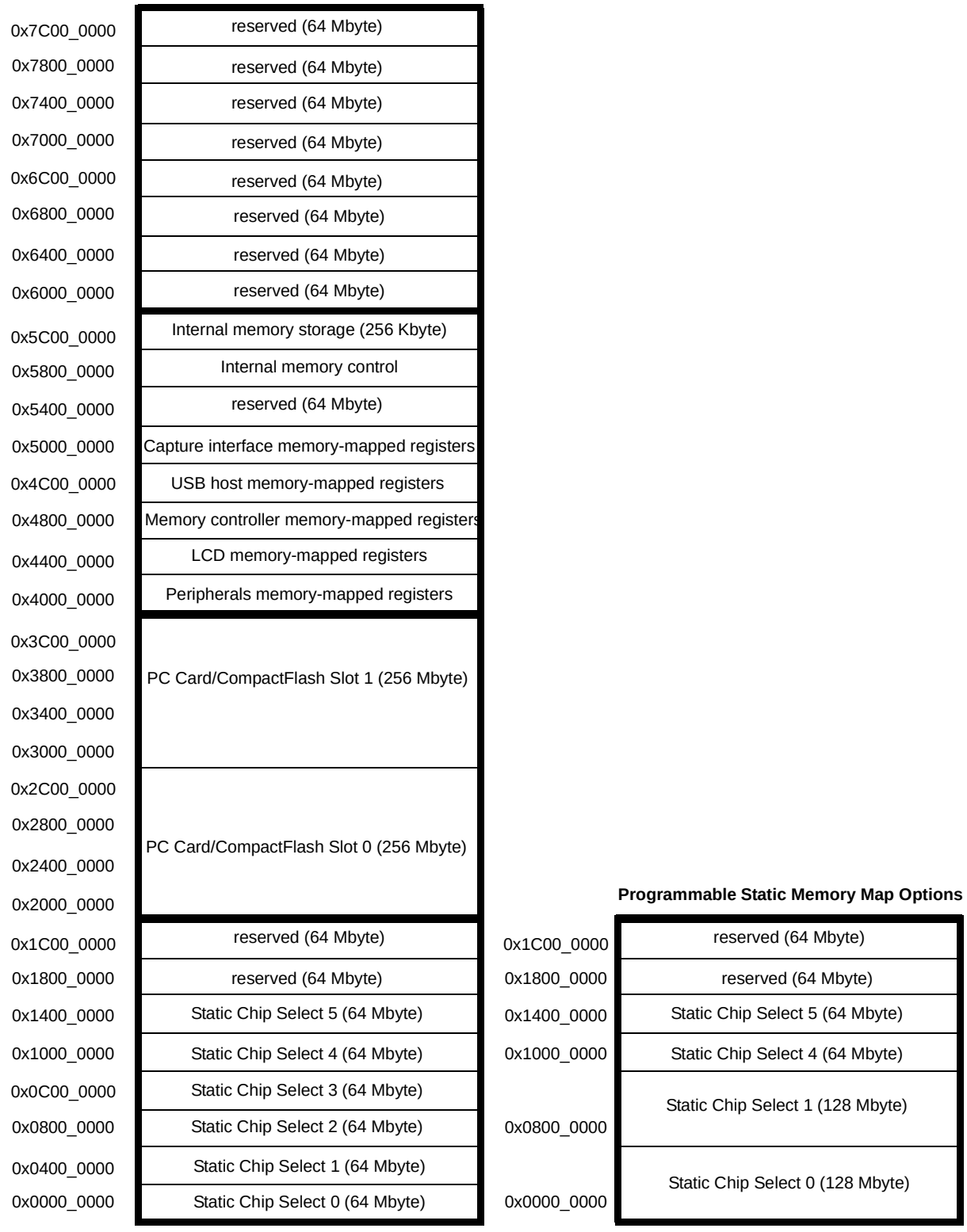
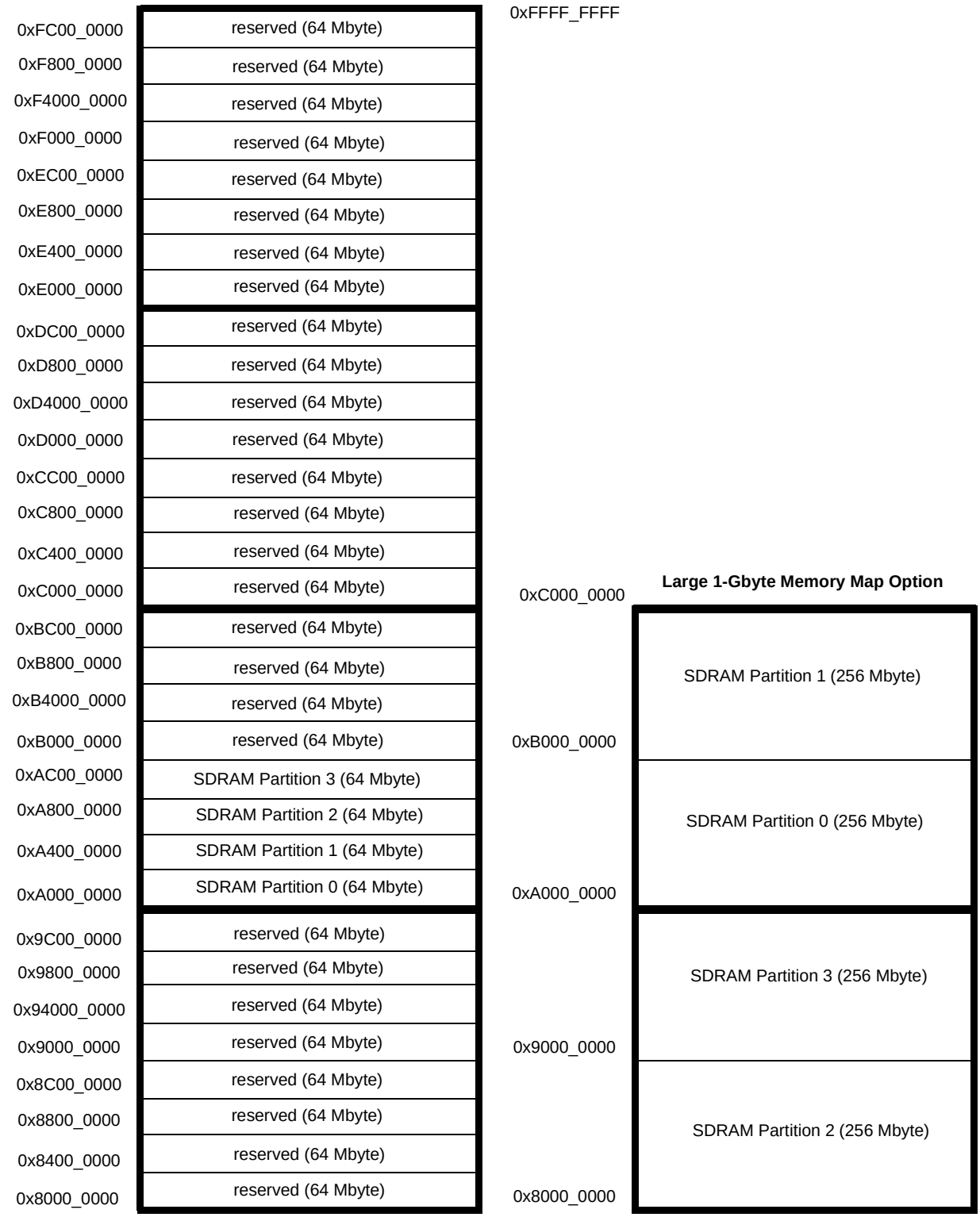


Figure 28-3. Memory Map—0x8000_0000–0xFFFF_FFFF



28.2 System Bus Unit Registers

This section summarizes the memory maps for the following modules on the main system bus of the on-chip system:

- Section 28.2.1 — Intel XScale® Microarchitecture Core Registers
- Section 28.2.2 — Memory Controller Registers (see also Chapter 6, “Memory Controller”)
- Section 28.2.3 — LCD Controller Registers (see also Chapter 7, “LCD Controller”)
- Section 28.2.4 — USB Host Controller Registers (see also Chapter 20, “USB Host Controller”)
- Section 28.2.5 — Internal Memory Registers (see also Chapter 4, “Internal Memory”)
- Section 28.2.6 — Quick Capture Interface Registers (see also Chapter 27, “Quick Capture Interface”)

The addresses of these blocks lie within the range 0x4000_0000–0x5FFF_FFFF.

28.2.1 Intel XScale® Microarchitecture Core Registers

Table 28-1 summarizes the registers within the independent coprocessors in the processor core. For more information, refer to the *Intel XScale® Core Developer’s Manual.*

These registers are accessible using coprocessor instructions. Some, as noted in Table 28-1, are also accessible through memory-map addressing.

Table 28-1. Coprocessor Register Summary (Sheet 1 of 3)

CRn	CRm	Opcode2	Register Symbol	Register Description
Coprocessor 6—Interrupt Controller				
0	0	0	ICIP†	Interrupt Controller IRQ Pending register
1	0	0	ICMR†	Interrupt Controller Mask register
2	0	0	ICLR†	Interrupt Controller Level register
3	0	0	ICFR†	Interrupt Controller FIQ Pending register
4	0	0	ICPR†	Interrupt Pending register
5	0	0	ICHP†	Interrupt Highest Priority register
Coprocessor 14—Clock, Power Management, Debug, and Trace Registers				
0	0	0	PMNC	Performance Monitoring Control register
1	0	0	CCNT	Performance Monitoring Clock Counter
2	0	0	PMN0	Performance Monitoring Event Counter 0
3	0	0	PMN1	Performance Monitoring Event Counter 1
4	—	—	—	reserved
5	—	—	—	reserved
6	0	0	CCLKCFG	Core Clock Configuration register
7	0	0	PWRMODE	Power Management register

Table 28-1. Coprocessor Register Summary (Sheet 2 of 3)

CRn	CRm	Opcode2	Register Symbol	Register Description
8	0	0	TX	Access Transmit Debug register
9	0	0	RX	Access Receive Debug register
10	0	0	DBGCSR	Access Debug Control and Status register
11	0	0	TBREG	Access Trace Buffer register
12	0	0	CHKPT0	Access Checkpoint0 register
13	0	0	CHKPT1	Access Checkpoint1 register
14	0	0	TXRXCTRL	Access Transmit and Receive Debug Control
15	—	—	—	reserved
Coprocessor 15—Intel XScale® Core System Control				
	ID and Cache Type Registers			
0	0	0	ID	Identification register
0	0	1		Cache Type
	Control and Auxiliary Registers			
1	0	0		ARM* Control register
1	0	1		Auxiliary Control register
2	0	0		Translation Table Base register
3	0	0		Domain Access Control register
4	—	—	—	reserved
5	0	0		Fault Status register
6	0	0		Fault Address register
	Cache Operations			
7	7	0		Invalidate I&D cache and BTB
7	5	0		Invalidate I cache and BTB
7	5	1		Invalidate I-cache Line
7	6	0		Invalidate D-cache
7	6	1		Invalidate D-cache Line
7	10	1		Clean D-cache Line
7	10	4		Drain Write (&Fill) Buffer
7	5	6		Invalidate Branch Target Buffer
7	2	5		Allocate Line in the Data Cache
8	7	0		Invalidate I&D TLB
8	5	0		Invalidate I TLB
8	5	1		Invalidate I TLB Entry
8	6	0		Invalidate D TLB
8	6	1		Invalidate D TLB Entry
	Cache Lock Down			
9	1	0		Fetch and Lock I-Cache Line

Table 28-1. Coprocessor Register Summary (Sheet 3 of 3)

CRn	CRm	Opcode2	Register Symbol	Register Description
	TLB Operations			
9	1	1		Unlock I-Cache
9	2	0		Read Data Cache Lock register
9	2	0		Write Data Cache Lock register
9	2	1		Unlock Data Cache
	TLB Lock Down			
10	4	0		Translate and Lock Instruction TLB Entry
10	8	0		Translate and Lock Data TLB Entry
10	4	1		Unlock Instruction TLB
10	8	1		Unlock Data TLB
11	—	—	—	reserved
12	—	—	—	reserved
13	0	0	PID	Processor ID
	Breakpoint Registers			
14	0	0	DBCR0	Data Breakpoint register 0
14	3	0	DBCR1	Data Breakpoint register 1
14	4	0	DBCON	Data Breakpoint Control register
14	8	0	IBCR0	Instruction Breakpoint register 0
14	9	0	IBCR1	Instruction Breakpoint register 1
15	1	0	CPAR	Coprocessor Access
15	1	0	CPAR	Coprocessor Access
† These registers are also accessible through memory-mapped addressing.				

28.2.2 Memory Controller Registers

Table 28-2. Memory Controller Register Summary (Sheet 1 of 2)

Address	Name	Description	Page
0x4800_0000	MDCNFG	SDRAM Configuration register	6-43
0x4800_0004	MDREFR	SDRAM Refresh Control register	6-52
0x4800_0008	MSC0	Static Memory Control register 0	6-62
0x4800_000C	MSC1	Static Memory Control register 1	6-62
0x4800_0010	MSC2	Static Memory Control register 2	6-62
0x4800_0014	MECR	Expansion Memory (PC Card/CompactFlash) Bus Configuration register	6-78
0x4800_0018	—	reserved	—
0x4800_001C	SXCNFG	Synchronous Static Memory Configuration register	6-57

Table 28-2. Memory Controller Register Summary (Sheet 2 of 2)

Address	Name	Description	Page
0x4800_0020	FLYCNFG	Fly-by DMA DVAL<1:0> polarities	5-39
0x4800_0024	—	reserved	—
0x4800_0028	MCMEM0	PC Card Interface Common Memory Space Socket 0 Timing Configuration register	6-76
0x4800_002C	MCMEM1	PC Card Interface Common Memory Space Socket 1 Timing Configuration register	6-76
0x4800_0030	MCATT0	PC Card Interface Attribute Space Socket 0 Timing Configuration register	6-76
0x4800_0034	MCATT1	PC Card Interface Attribute Space Socket 1 Timing Configuration register	6-76
0x4800_0038	MCIO0	PC Card Interface I/o Space Socket 0 Timing Configuration register	6-77
0x4800_003C	MCIO1	PC Card Interface I/o Space Socket 1 Timing Configuration register	6-77
0x4800_0040	MDMRS	SDRAM Mode Register Set Configuration register	6-48
0x4800_0044	BOOT_DEF	Boot Time Default Configuration register	6-74
0x4800_0048	ARB_CNTL	Arbiter Control register	29-2
0x4800_004C	BSCNTR0	System Memory Buffer Strength Control register 0	6-80
0x4800_0050	BSCNTR1	System Memory Buffer Strength Control register 1	6-81
0x4800_0054	LCDBSCNTR	LCD Buffer Strength Control register	7-104
0x4800_0058	MDMRSLP	Special Low Power SDRAM Mode Register Set Configuration register	6-50
0x4800_005C	BSCNTR2	System Memory Buffer Strength Control register 2	6-82
0x4800_0060	BSCNTR3	System Memory Buffer Strength Control register 3	6-83
0x4800_0064	SA1110	SA-1110 Compatibility Mode for Static Memory register	6-69
0x4800_0068–0x4800_FFFC	—	reserved	—

28.2.3 LCD Controller Registers

Table 28-3. LCD Controller Register Summary (Sheet 1 of 3)

Address	Name	Description	Page
0x4400_0000	LCCR0	LCD Controller Control register 0	7-56
0x4400_0004	LCCR1	LCD Controller Control register 1	7-64
0x4400_0008	LCCR2	LCD Controller Control register 2	7-66
0x4400_000C	LCCR3	LCD Controller Control register 3	7-69
0x4400_0010	LCCR4	LCD Controller Control register 4	7-74
0x4400_0014	LCCR5	LCD Controller Control register 5	7-79
0x4400_0018–0x4400_001C	—	reserved	—

Table 28-3. LCD Controller Register Summary (Sheet 2 of 3)

Address	Name	Description	Page
0x4400_0020	FBR0	DMA Channel 0 Frame Branch register	7-103
0x4400_0024	FBR1	DMA Channel 1 Frame Branch register	7-103
0x4400_0028	FBR2	DMA Channel 2 Frame Branch register	7-103
0x4400_002C	FBR3	DMA Channel 3 Frame Branch register	7-103
0x4400_0030	FBR4	DMA Channel 4 Frame Branch register	7-103
0x4400_0034	LCSR1	LCD Controller Status register 1	7-111
0x4400_0038	LCSR0	LCD Controller Status register 0	7-106
0x4400_003C	LIIDR	LCD Controller Interrupt ID register	7-118
0x4400_0040	TRGBR	TMED RGB Seed register	7-99
0x4400_0044	TCR	TMED Control register	7-100
0x4400_0048– 0x4400_004C	—	reserved	—
0x4400_0050	OVL1C1	Overlay 1 Control register 1	7-92
0x4400_0054– 0x4400_005C	—	reserved	—
0x4400_0060	OVL1C2	Overlay 1 Control register 2	7-93
0x4400_0064– 0x4400_006C	—	reserved	—
0x4400_0070	OVL2C1	Overlay 2 Control register 1	7-94
0x4400_0074– 0x4400_007C	—	reserved	—
0x4400_0080	OVL2C2	Overlay 2 Control register 2	7-96
0x4400_0084– 0x4400_008C	—	reserved	—
0x4400_0090	CCR	Cursor Control register	7-97
0x4400_0094– 0x4400_009C	—	reserved	—
0x4400_0100	CMDCR	Command Control register	7-98
0x4400_0104	PRSR	Panel Read Status register	7-105
0x4400_0108– 0x4400_010C	—	reserved	—
0x4400_0110	FBR5	DMA Channel 5 Frame Branch register	7-103
0x4400_0114	FBR6	DMA Channel 6 Frame Branch register	7-103
0x4400_0118– 0x4400_01FF	—	reserved	—
0x4400_0200	FDADR0	DMA Channel 0 Frame Descriptor Address register	7-102
0x4400_0204	FSADR0	DMA Channel 0 Frame Source Address register	7-119
0x4400_0208	FIDR0	DMA Channel 0 Frame ID register	7-119
0x4400_020C	LDCMD0	LCD DMA Channel 0 Command register	7-120
0x4400_0210	FDADR1	DMA Channel 1 Frame Descriptor Address register	7-102
0x4400_0214	FSADR1	DMA Channel 1 Frame Source Address register	7-119

Table 28-3. LCD Controller Register Summary (Sheet 3 of 3)

Address	Name	Description	Page
0x4400_0218	FIDR1	DMA Channel 1 Frame ID register	7-119
0x4400_021C	LDCMD1	LCD DMA Channel 1 Command register	7-120
0x4400_0220	FDADR2	DMA Channel 2 Frame Descriptor Address register	7-102
0x4400_0224	FSADR2	DMA Channel 2 Frame Source Address register	7-119
0x4400_0228	FIDR2	DMA Channel 2 Frame ID register	7-119
0x4400_022C	LDCMD2	LCD DMA Channel 2 Command register	7-120
0x4400_0230	FDADR3	DMA Channel 3 Frame Descriptor Address register	7-102
0x4400_0234	FSADR3	DMA Channel 3 Frame Source Address register	7-119
0x4400_0238	FIDR3	DMA Channel 3 Frame ID register	7-119
0x4400_023C	LDCMD3	LCD DMA Channel 3 Command register	7-120
0x4400_0240	FDADR4	DMA Channel 4 Frame Descriptor Address register	7-102
0x4400_0244	FSADR4	DMA Channel 4 Frame Source Address register	7-119
0x4400_0248	FIDR4	DMA Channel 4 Frame ID register	7-119
0x4400_024C	LDCMD4	LCD DMA Channel 4 Command register	7-120
0x4400_0250	FDADR5	DMA Channel 5 Frame Descriptor Address register	7-102
0x4400_0254	FSADR5	DMA Channel 5 Frame Source Address register	7-119
0x4400_0258	FIDR5	DMA Channel 5 Frame ID register	7-119
0x4400_025C	LDCMD5	LCD DMA Channel 5 Command register	7-120
0x4400_0260	FDADR6	DMA Channel 6 Frame Descriptor Address register	7-102
0x4400_0264	FSADR6	DMA Channel 6 Frame Source Address register	7-119
0x4400_0268	FIDR6	DMA Channel 6 Frame ID register	7-119
0x4400_026C	LDCMD6	LCD DMA Channel 6 Command register	7-120
0x4400_0270– 0x47FF_FFFC	—	reserved	—
0x4800_0054	LCDBSCNTR	LCD Buffer Strength Control register	7-104

28.2.4 USB Host Controller Registers

Table 28-4. USB Host Controller Register Summary (Sheet 1 of 2)

Address	Name	Description	Page
0x4C00 0000	UHCREV	UHC HCI Spec Revision register	20-10
0x4C00 0004	UHCHCON	UHC Host Control register	20-10
0x4C00 0008	UHCCOMS	UHC Command Status register	20-14
0x4C00 000C	UHCINTS	UHC Interrupt Status register	20-16
0x4C00 0010	UHCINTE	UHC Interrupt Enable register	20-18
0x4C00 0014	UHCINTD	UHC Interrupt Disable register	20-20
0x4C00 0018	UHCHCCA	UHC Host Controller Communication Area register	20-21

Table 28-4. USB Host Controller Register Summary (Sheet 2 of 2)

Address	Name	Description	Page
0x4C00 001C	UHCPCED	UHC Period Current Endpoint Descriptor register	20-21
0x4C00 0020	UHCCHED	UHC Control Head Endpoint Descriptor register	20-22
0x4C00 0024	UHCCCED	UHC Control Current Endpoint Descriptor register	20-22
0x4C00 0028	UHCBHED	UHC Bulk Head Endpoint Descriptor register	20-23
0x4C00 002C	UHCBCED	UHC Bulk Current Endpoint Descriptor register	20-24
0x4C00 0030	UHCDHEAD	UHC Done Head register	20-25
0x4C00 0034	UHCFMI	UHC Frame Interval register	20-26
0x4C00 0038	UHCFMR	UHC Frame Remaining register	20-27
0x4C00 003C	UHCFMN	UHC Frame Number register	20-28
0x4C00 0040	UHCPERS	UHC Periodic Start register	20-29
0x4C00 0044	UHCLST	UHC Low-Speed Threshold register	20-30
0x4C00 0048	UHCRHDA	UHC Root Hub Descriptor A register	20-31
0x4C00 004C	UHCRHDB	UHC Root Hub Descriptor B register	20-33
0x4C00 0050	UHCRHS	UHC Root Hub Status register	20-34
0x4C00 0054	UHCRHPS1	UHC Root Hub Port 1 Status register	20-35
0x4C00 0058	UHCRHPS2	UHC Root Hub Port 2 Status register	20-35
0x4C00 005C	UHCRHPS3	UHC Root Hub Port 3 Status register	20-35
0x4C00 0060	UHCSTAT	UHC Status register	20-39
0x4C00 0064	UHCHR	UHC Reset register	20-41
0x4C00 0068	UHCHIE	UHC Interrupt Enable register	20-44
0x4C00 006C	UHCHIT	UHC Interrupt Test register	20-45
0x4C00 0070– 0x4FFF FFFF	—	reserved	—

28.2.5 Internal Memory Registers

Table 28-5. Internal Memory Register Summary (Sheet 1 of 2)

Address	Name	Description	Page
0x5800_0000– 0x5BFF_FFFC	—	reserved	—
0x5C00_0000– 0x5C00_FFFC	Memory Bank 0	64-Kbyte SRAM	—
0x5C01_0000– 0x5C01_FFFC	Memory Bank 1	6-4Kbyte SRAM	—
0x5C02_0000– 0x5C02_FFFC	Memory Bank 2	64-Kbyte SRAM	—

Table 28-5. Internal Memory Register Summary (Sheet 2 of 2)

Address	Name	Description	Page
0x5C03_0000–0x5C03_FFFC	Memory Bank 3	64-Kbyte SRAM	—
0x5C04_0000–0x5C7F_FFFC	—	reserved	—
0x5C80_0000–0x5FFF_FFFC	—	reserved	—

28.2.6 Quick Capture Interface Registers

Table 28-6. Quick Capture Interface Register Summary

Physical Address	Name	Description	Page
0x5000_0000	CICR0	Quick Capture Interface Control register 0	27-24
0x5000_0004	CICR1	Quick Capture Interface Control register 1	27-28
0x5000_0008	CICR2	Quick Capture Interface Control register 2	27-32
0x5000_000C	CICR3	Quick Capture Interface Control register 3	27-33
0x5000_0010	CICR4	Quick Capture Interface Control register 4	27-34
0x5000_0014	CISR	Quick Capture Interface Status register	27-37
0x5000_0018	CIFR	Quick Capture Interface FIFO Control register	27-40
0x5000_001C	CITOR	Quick Capture Interface Time-Out register	27-37
0x5000_0020 – 0x5000_0024	—	reserved	—
0x5000_0028	CIBR0	Quick Capture Interface Receive Buffer register 0 (Channel 0)	27-42
0x5000_002C	—	reserved	—
0x5000_0030	CIBR1	Quick Capture Interface Receive Buffer register 1 (Channel 1)	27-42
0x5000_0034	—	reserved	—
0x5000_0038	CIBR2	Quick Capture Interface Receive Buffer register 2 (Channel 2)	27-42
0x5000_003C–0x53FF_FFFC	—	reserved	—

28.3 Peripheral Module Registers

Table 28-7 summarizes the memory areas for the modules connected to the processor peripheral bus. Table 28-8 details the register addresses within each peripheral.

Table 28-7. Peripheral Module Address Summary

Unit	Address
DMA Controller	0x4000_0000
UART1—Full Function UART	0x4010_0000
UART2—Bluetooth UART	0x4020_0000
Standard I^2C Bus Interface Unit	0x4030_0000
I^2S Controller	0x4040_0000
AC '97 Controller	0x4050_0000
USB Client Controller	0x4060_0000
UART 3—Standard UART	0x4070_0000
Fast Infrared Communications Port	0x4080_0000
RTC	0x4090_0000
OS Timers	0x40A0_0000
PWM0 and 2	0x40B0_0000
PWM1 and 3	0x40C0_0000
Interrupt Controller	0x40D0_0000
GPIO Controller	0x40E0_0000
Power Manager	0x40F0_0000
Reset Controller	0x40F0_0000
Power Manager I^2C	0x40F0_0180
Synchronous Serial Port 1	0x4100_0000
MultiMediaCard/SD/SDIO Controller	0x4110_0000
reserved	0x4120_0000
Clocks Manager	0x4130_0000
Mobile Scalable Link (MSL)	0x4140_0000
Keypad Interface	0x4150_0000
Universal Subscriber ID (USIM) Interface	0x4160_0000
Synchronous Serial Port 2	0x4170_0000
Memory Stick Host Controller	0x4180_0000
Synchronous Serial Port 3	0x4190_0000

Table 28-8. Register Address Summary—Peripherals (Sheet 1 of 24)

Address	Name	Description	Page
DMA Controller			
0x4000_0000	DCSR0	DMA Control/Status register for Channel 0	5-41
0x4000_0004	DCSR1	DMA Control/Status register for Channel 1	5-41
0x4000_0008	DCSR2	DMA Control/Status register for Channel 2	5-41
0x4000_000C	DCSR3	DMA Control/Status register for Channel 3	5-41
0x4000_0010	DCSR4	DMA Control/Status register for Channel 4	5-41
0x4000_0014	DCSR5	DMA Control/Status register for Channel 5	5-41
0x4000_0018	DCSR6	DMA Control/Status register for Channel 6	5-41
0x4000_001C	DCSR7	DMA Control/Status register for Channel 7	5-41
0x4000_0020	DCSR8	DMA Control/Status register for Channel 8	5-41
0x4000_0024	DCSR9	DMA Control/Status register for Channel 9	5-41
0x4000_0028	DCSR10	DMA Control/Status register for Channel 10	5-41
0x4000_002C	DCSR11	DMA Control/Status register for Channel 11	5-41
0x4000_0030	DCSR12	DMA Control/Status register for Channel 12	5-41
0x4000_0034	DCSR13	DMA Control/Status register for Channel 13	5-41
0x4000_0038	DCSR14	DMA Control/Status register for Channel 14	5-41
0x4000_003C	DCSR15	DMA Control/Status register for Channel 15	5-41
0x4000_0040	DCSR16	DMA Control/Status register for Channel 16	5-41
0x4000_0044	DCSR17	DMA Control/Status register for Channel 17	5-41
0x4000_0048	DCSR18	DMA Control/Status register for Channel 18	5-41
0x4000_004C	DCSR19	DMA Control/Status register for Channel 19	5-41
0x4000_0050	DCSR20	DMA Control/Status register for Channel 20	5-41
0x4000_0054	DCSR21	DMA Control/Status register for Channel 21	5-41
0x4000_0058	DCSR22	DMA Control/Status register for Channel 22	5-41
0x4000_005C	DCSR23	DMA Control/Status register for Channel 23	5-41
0x4000_0060	DCSR24	DMA Control/Status register for Channel 24	5-41
0x4000_0064	DCSR25	DMA Control/Status register for Channel 25	5-41
0x4000_0068	DCSR26	DMA Control/Status register for Channel 26	5-41
0x4000_006C	DCSR27	DMA Control/Status register for Channel 27	5-41
0x4000_0070	DCSR28	DMA Control/Status register for Channel 28	5-41
0x4000_0074	DCSR29	DMA Control/Status register for Channel 29	5-41
0x4000_0078	DCSR30	DMA Control/Status register for Channel 30	5-41
0x4000_007C	DCSR31	DMA Control/Status register for Channel 31	5-41
0x4000_0080–0x4000_009C	—	reserved	—
0x4000_00A0	DALGN	DMA Alignment register	5-49
0x4000_00A4	DPCSR	DMA Programmed I/O Control Status register	5-51

Table 28-8. Register Address Summary—Peripherals (Sheet 2 of 24)

Address	Name	Description	Page
0x4000_00A8–0x4000_00DC	—	reserved	—
0x4000_00E0	DRQSR0	DMA DREQ<0> Status register	5-40
0x4000_00E4	DRQSR1	DMA DREQ<1> Status register	5-40
0x4000_00E8	DRQSR2	DMA DREQ<2> Status register	5-40
0x4000_00EC	—	reserved	—
0x4000_00F0	DINT	DMA Interrupt register	5-48
0x4000_00F4–0x4000_00FC	—	reserved	—
0x4000_0100	DRCMR0	Request to Channel Map register for DREQ<0> (companion chip request 0)	5-31
0x4000_0104	DRCMR1	Request to Channel Map register for DREQ<1> (companion chip request 1)	5-31
0x4000_0108	DRCMR2	Request to Channel Map register for I^2S receive request	5-31
0x4000_010C	DRCMR3	Request to Channel Map register for I^2S transmit request	5-31
0x4000_0110	DRCMR4	Request to Channel Map register for BTUART receive request	5-31
0x4000_0114	DRCMR5	Request to Channel Map register for BTUART transmit request.	5-31
0x4000_0118	DRCMR6	Request to Channel Map register for FFUART receive request	5-31
0x4000_011C	DRCMR7	Request to Channel Map register for FFUART transmit request	5-31
0x4000_0120	DRCMR8	Request to Channel Map register for AC ’97 microphone request	5-31
0x4000_0124	DRCMR9	Request to Channel Map register for AC ’97 modem receive request	5-31
0x4000_0128	DRCMR10	Request to Channel Map register for AC ’97 modem transmit request	5-31
0x4000_012C	DRCMR11	Request to Channel Map register for AC ’97 audio receive request	5-31
0x4000_0130	DRCMR12	Request to Channel Map register for AC ’97 audio transmit request	5-31
0x4000_0134	DRCMR13	Request to Channel Map register for SSP1 receive request	5-31
0x4000_0138	DRCMR14	Request to Channel Map register for SSP1 transmit request	5-31
0x4000_013C	DRCMR15	Request to Channel Map register for SSP2 receive request	5-31
0x4000_0140	DRCMR16	Request to Channel Map register for SSP2 transmit request	5-31
0x4000_0144	DRCMR17	Request to Channel Map register for ICP receive request	5-31
0x4000_0148	DRCMR18	Request to Channel Map register for ICP transmit request	5-31
0x4000_014C	DRCMR19	Request to Channel Map register for STUART receive request	5-31
0x4000_0150	DRCMR20	Request to Channel Map register for STUART transmit request	5-31
0x4000_0154	DRCMR21	Request to Channel Map register for MMC/SDIO receive request	5-31
0x4000_0158	DRCMR22	Request to Channel Map register for MMC/SDIO transmit request	5-31
0x4000_015C	—	reserved	—
0x4000_0160	DRCMR24	Request to Channel Map register for USB endpoint 0 request	5-31
0x4000_0164	DRCMR25	Request to Channel Map register for USB endpoint A request	5-31
0x4000_0168	DRCMR26	Request to Channel Map register for USB endpoint B request	5-31
0x4000_016C	DRCMR27	Request to Channel Map register for USB endpoint C request	5-31

Table 28-8. Register Address Summary—Peripherals (Sheet 3 of 24)

Address	Name	Description	Page
0x4000_0170	DRCMR28	Request to Channel Map register for USB endpoint D request	5-31
0x4000_0174	DRCMR29	Request to Channel Map register for USB endpoint E request	5-31
0x4000_0178	DRCMR30	Request to Channel Map register for USB endpoint F request	5-31
0x4000_017C	DRCMR31	Request to Channel Map register for USB endpoint G request	5-31
0x4000_0180	DRCMR32	Request to Channel Map register for USB endpoint H request	5-31
0x4000_0184	DRCMR33	Request to Channel Map register for USB endpoint I request	5-31
0x4000_0188	DRCMR34	Request to Channel Map register for USB endpoint J request	5-31
0x4000_018C	DRCMR35	Request to Channel Map register for USB endpoint K request	5-31
0x4000_0190	DRCMR36	Request to Channel Map register for USB endpoint L request	5-31
0x4000_0194	DRCMR37	Request to Channel Map register for USB endpoint M request	5-31
0x4000_0198	DRCMR38	Request to Channel Map register for USB endpoint N request	5-31
0x4000_019C	DRCMR39	Request to Channel Map register for USB endpoint P request	5-31
0x4000_01A0	DRCMR40	Request to Channel Map register for USB endpoint Q request	5-31
0x4000_01A4	DRCMR41	Request to Channel Map register for USB endpoint R request	5-31
0x4000_01A8	DRCMR42	Request to Channel Map register for USB endpoint S request	5-31
0x4000_01AC	DRCMR43	Request to Channel Map register for USB endpoint T request	5-31
0x4000_01B0	DRCMR44	Request to Channel Map register for USB endpoint U request	5-31
0x4000_01B4	DRCMR45	Request to Channel Map register for USB endpoint V request	5-31
0x4000_01B8	DRCMR46	Request to Channel Map register for USB endpoint W request	5-31
0x4000_01BC	DRCMR47	Request to Channel Map register for USB endpoint X request	5-31
0x4000_01C0	DRCMR48	Request to Channel Map register for MSL receive request 1	5-31
0x4000_01C4	DRCMR49	Request to Channel Map register for MSL transmit request 1	5-31
0x4000_01C8	DRCMR50	Request to Channel Map register for MSL receive request 2	5-31
0x4000_01CC	DRCMR51	Request to Channel Map register for MSL transmit request 2	5-31
0x4000_01D0	DRCMR52	Request to Channel Map register for MSL receive request 3	5-31
0x4000_01D4	DRCMR53	Request to Channel Map register for MSL transmit request 3	5-31
0x4000_01D8	DRCMR54	Request to Channel Map register for MSL receive request 4	5-31
0x4000_01DC	DRCMR55	Request to Channel Map register for MSL transmit request 4	5-31
0x4000_01E0	DRCMR56	Request to Channel Map register for MSL receive request 5	5-31
0x4000_01E4	DRCMR57	Request to Channel Map register for MSL transmit request 5	5-31
0x4000_01E8	DRCMR58	Request to Channel Map register for MSL receive request 6	5-31
0x4000_01EC	DRCMR59	Request to Channel Map register for MSL transmit request 6	5-31
0x4000_01F0	DRCMR60	Request to Channel Map register for MSL receive request 7	5-31
0x4000_01F4	DRCMR61	Request to Channel Map register for MSL transmit request 7	5-31
0x4000_01F8	DRCMR62	Request to Channel Map register for USIM receive request	5-31
0x4000_01FC	DRCMR63	Request to Channel Map register for USIM transmit request	5-31
0x4000_0200	DDADR0	DMA Descriptor Address register for Channel 0	5-32
0x4000_0204	DSADR0	DMA Source Address register for Channel 0	5-33

Table 28-8. Register Address Summary—Peripherals (Sheet 4 of 24)

Address	Name	Description	Page
0x4000_0208	DTADR0	DMA Target Address register for Channel 0	5-34
0x4000_020C	DCMD0	DMA Command Address register for Channel 0	5-35
0x4000_0210	DDADR1	DMA Descriptor Address register for Channel 1	5-32
0x4000_0214	DSADR1	DMA Source Address register for Channel 1	5-33
0x4000_0218	DTADR1	DMA Target Address register for Channel 1	5-34
0x4000_021C	DCMD1	DMA Command Address register for Channel 1	5-35
0x4000_0220	DDADR2	DMA Descriptor Address register for Channel 2	5-32
0x4000_0224	DSADR2	DMA Source Address register for Channel 2	5-33
0x4000_0228	DTADR2	DMA Target Address register for Channel 2	5-34
0x4000_022C	DCMD2	DMA Command Address register for Channel 2	5-35
0x4000_0230	DDADR3	DMA Descriptor Address register for Channel 3	5-32
0x4000_0234	DSADR3	DMA Source Address register for Channel 3	5-33
0x4000_0238	DTADR3	DMA Target Address register for Channel 3	5-34
0x4000_023C	DCMD3	DMA Command Address register for Channel 3	5-35
0x4000_0240	DDADR4	DMA Descriptor Address register for Channel 4	5-32
0x4000_0244	DSADR4	DMA Source Address register for Channel 4	5-33
0x4000_0248	DTADR4	DMA Target Address register for Channel 4	5-34
0x4000_024C	DCMD4	DMA Command Address register for Channel 4	5-35
0x4000_0250	DDADR5	DMA Descriptor Address register for Channel 5	5-32
0x4000_0254	DSADR5	DMA Source Address register for Channel 5	5-33
0x4000_0258	DTADR5	DMA Target Address register for Channel 5	5-34
0x4000_025C	DCMD5	DMA Command Address register for Channel 5	5-35
0x4000_0260	DDADR6	DMA Descriptor Address register for Channel 6	5-32
0x4000_0264	DSADR6	DMA Source Address register for Channel 6	5-33
0x4000_0268	DTADR6	DMA Target Address register for Channel 6	5-34
0x4000_026C	DCMD6	DMA Command Address register for Channel 6	5-35
0x4000_0270	DDADR7	DMA Descriptor Address register for Channel 7	5-32
0x4000_0274	DSADR7	DMA Source Address register for Channel 7	5-33
0x4000_0278	DTADR7	DMA Target Address register for Channel 7	5-34
0x4000_027C	DCMD7	DMA Command Address register for Channel 7	5-35
0x4000_0280	DDADR8	DMA Descriptor Address register for Channel 8	5-32
0x4000_0284	DSADR8	DMA Source Address register for Channel 8	5-33
0x4000_0288	DTADR8	DMA Target Address register for Channel 8	5-34
0x4000_028C	DCMD8	DMA Command Address register for Channel 8	5-35
0x4000_0290	DDADR9	DMA Descriptor Address register for Channel 9	5-32
0x4000_0294	DSADR9	DMA Source Address register for Channel 9	5-33
0x4000_0298	DTADR9	DMA Target Address register for Channel 9	5-34
0x4000_029C	DCMD9	DMA Command Address register for Channel 9	5-35

Table 28-8. Register Address Summary—Peripherals (Sheet 5 of 24)

Address	Name	Description	Page
0x4000_02A0	DDADR10	DMA Descriptor Address register for Channel 10	5-32
0x4000_02A4	DSADR10	DMA Source Address register for Channel 10	5-33
0x4000_02A8	DTADR10	DMA Target Address register for Channel 10	5-34
0x4000_02AC	DCMD10	DMA Command Address register for Channel 10	5-35
0x4000_02B0	DDADR11	DMA Descriptor Address register for Channel 11	5-32
0x4000_02B4	DSADR11	DMA Source Address register for Channel 11	5-33
0x4000_02B8	DTADR11	DMA Target Address register for Channel 11	5-34
0x4000_02BC	DCMD11	DMA Command Address register for Channel 11	5-35
0x4000_02C0	DDADR12	DMA Descriptor Address register for Channel 12	5-32
0x4000_02C4	DSADR12	DMA Source Address register for Channel 12	5-33
0x4000_02C8	DTADR12	DMA Target Address register for Channel 12	5-34
0x4000_02CC	DCMD12	DMA Command Address register for Channel 12	5-35
0x4000_02D0	DDADR13	DMA Descriptor Address register for Channel 13	5-32
0x4000_02D4	DSADR13	DMA Source Address register for Channel 13	5-33
0x4000_02D8	DTADR13	DMA Target Address register for Channel 13	5-34
0x4000_02DC	DCMD13	DMA Command Address register for Channel 13	5-35
0x4000_02E0	DDADR14	DMA Descriptor Address register for Channel 14	5-32
0x4000_02E4	DSADR14	DMA Source Address register for Channel 14	5-33
0x4000_02E8	DTADR14	DMA Target Address register for Channel 14	5-34
0x4000_02EC	DCMD14	DMA Command Address register for Channel 14	5-35
0x4000_02F0	DDADR15	DMA Descriptor Address register for Channel 15	5-32
0x4000_02F4	DSADR15	DMA Source Address register for Channel 15	5-33
0x4000_02F8	DTADR15	DMA Target Address register for Channel 15	5-34
0x4000_02FC	DCMD15	DMA Command Address register for Channel 15	5-35
0x4000_0300	DDADR16	DMA Descriptor Address register for Channel 16	5-32
0x4000_0304	DSADR16	DMA Source Address register for Channel 16	5-33
0x4000_0308	DTADR16	DMA Target Address register for Channel 16	5-34
0x4000_030C	DCMD16	DMA Command Address register for Channel 16	5-35
0x4000_0310	DDADR17	DMA Descriptor Address register for Channel 17	5-32
0x4000_0314	DSADR17	DMA Source Address register for Channel 17	5-33
0x4000_0318	DTADR17	DMA Target Address register for Channel 17	5-34
0x4000_031C	DCMD17	DMA Command Address register for Channel 17	5-35
0x4000_0320	DDADR18	DMA Descriptor Address register for Channel 18	5-32
0x4000_0324	DSADR18	DMA Source Address register for Channel 18	5-33
0x4000_0328	DTADR18	DMA Target Address register for Channel 18	5-34
0x4000_032C	DCMD18	DMA Command Address register for Channel 18	5-35
0x4000_0330	DDADR19	DMA Descriptor Address register for Channel 19	5-32
0x4000_0334	DSADR19	DMA Source Address register for Channel 19	5-33

Table 28-8. Register Address Summary—Peripherals (Sheet 6 of 24)

Address	Name	Description	Page
0x4000_0338	DTADR19	DMA Target Address register for Channel 19	5-34
0x4000_033C	DCMD19	DMA Command Address register for Channel 19	5-35
0x4000_0340	DDADR20	DMA Descriptor Address register for Channel 20	5-32
0x4000_0344	DSADR20	DMA Source Address register for Channel 20	5-33
0x4000_0348	DTADR20	DMA Target Address register for Channel 20	5-34
0x4000_034C	DCMD20	DMA Command Address register for Channel 20	5-35
0x4000_0350	DDADR21	DMA Descriptor Address register for Channel 21	5-32
0x4000_0354	DSADR21	DMA Source Address register for Channel 21	5-33
0x4000_0358	DTADR21	DMA Target Address register for Channel 21	5-34
0x4000_035C	DCMD21	DMA Command Address register for Channel 21	5-35
0x4000_0360	DDADR22	DMA Descriptor Address register for Channel 22	5-32
0x4000_0364	DSADR22	DMA Source Address register for Channel 22	5-33
0x4000_0368	DTADR22	DMA Target Address register for Channel 22	5-34
0x4000_036C	DCMD22	DMA Command Address register for Channel 22	5-35
0x4000_0370	DDADR23	DMA Descriptor Address register for Channel 23	5-32
0x4000_0374	DSADR23	DMA Source Address register for Channel 23	5-33
0x4000_0378	DTADR23	DMA Target Address register for Channel 23	5-34
0x4000_037C	DCMD23	DMA Command Address register for Channel 23	5-35
0x4000_0380	DDADR24	DMA Descriptor Address register for Channel 24	5-32
0x4000_0384	DSADR24	DMA Source Address register for Channel 24	5-33
0x4000_0388	DTADR24	DMA Target Address register for Channel 24	5-34
0x4000_038C	DCMD24	DMA Command Address register for Channel 24	5-35
0x4000_0390	DDADR25	DMA Descriptor Address register for Channel 25	5-32
0x4000_0394	DSADR25	DMA Source Address register for Channel 25	5-33
0x4000_0398	DTADR25	DMA Target Address register for Channel 25	5-34
0x4000_039C	DCMD25	DMA Command Address register for Channel 25	5-35
0x4000_03A0	DDADR26	DMA Descriptor Address register for Channel 26	5-32
0x4000_03A4	DSADR26	DMA Source Address register for Channel 26	5-33
0x4000_03A8	DTADR26	DMA Target Address register for Channel 26	5-34
0x4000_03AC	DCMD26	DMA Command Address register for Channel 26	5-35
0x4000_03B0	DDADR27	DMA Descriptor Address register for Channel 27	5-32
0x4000_03B4	DSADR27	DMA Source Address register for Channel 27	5-33
0x4000_03B8	DTADR27	DMA Target Address register for Channel 27	5-34
0x4000_03BC	DCMD27	DMA Command Address register for Channel 27	5-35
0x4000_03C0	DDADR28	DMA Descriptor Address register for Channel 28	5-32
0x4000_03C4	DSADR28	DMA Source Address register for Channel 28	5-33
0x4000_03C8	DTADR28	DMA Target Address register for Channel 28	5-34
0x4000_03CC	DCMD28	DMA Command Address register for Channel 28	5-35

Table 28-8. Register Address Summary—Peripherals (Sheet 7 of 24)

Address	Name	Description	Page
0x4000_03D0	DDADR29	DMA Descriptor Address register for Channel 29	5-32
0x4000_03D4	DSADR29	DMA Source Address register for Channel 29	5-33
0x4000_03D8	DTADR29	DMA Target Address register for Channel 29	5-34
0x4000_03DC	DCMD29	DMA Command Address register for Channel 29	5-35
0x4000_03E0	DDADR30	DMA Descriptor Address register for Channel 30	5-32
0x4000_03E4	DSADR30	DMA Source Address register for Channel 30	5-33
0x4000_03E8	DTADR30	DMA Target Address register for Channel 30	5-34
0x4000_03EC	DCMD30	DMA Command Address register for Channel 30	5-35
0x4000_03F0	DDADR31	DMA Descriptor Address register for Channel 31	5-32
0x4000_03F4	DSADR31	DMA Source Address register for Channel 31	5-33
0x4000_03F8	DTADR31	DMA Target Address register for Channel 31	5-34
0x4000_03FC	DCMD31	DMA Command Address register for Channel 31	5-35
0x4000_0400–0x4000_10FC	—	reserved	—
0x4000_1100	DRCMR64	Request to Channel Map register for Memory Stick receive request	5-31
0x4000_1104	DRCMR65	Request to Channel Map register for Memory Stick transmit request	5-31
0x4000_1108	DRCMR66	Request to Channel Map register for SSP3 receive request	5-31
0x4000_110C	DRCMR67	Request to Channel Map register for SSP3 transmit request	5-31
0x4000_1110	DRCMR68	Request to Channel Map register for Quick Capture Interface Receive Request 0	5-31
0x4000_1114	DRCMR69	Request to Channel Map register for Quick Capture Interface Receive Request 1	5-31
0x4000_1118	DRCMR70	Request to Channel Map register for Quick Capture Interface Receive Request 2	5-31
0x4000_111C–0x4000_1124	—	reserved	—
0x4000_1128	DRCMR74	Request to Channel Map register for DREQ<2> (companion chip request 2)	5-31
0x4000_112C–0x400F_FFFC	—	reserved	—
0x4800_0020	FLYCNFG	Fly-by DMA DVAL<1:0> polarities	5-39
Full-Function UART			
0x4010_0000	FFRBR	Receive Buffer register	10-13
0x4010_0000	FFTHR	Transmit Holding register	10-14
0x4010_0000	FFDLL	Divisor Latch register, low byte	10-14
0x4010_0004	FFIER	Interrupt Enable register	10-15
0x4010_0004	FFDLH	Divisor Latch register, high byte	10-14
0x4010_0008	FFIIR	Interrupt ID register	10-17
0x4010_0008	FFFCR	FIFO Control register	10-19
0x4010_000C	FFLCR	Line Control register	10-25

Table 28-8. Register Address Summary—Peripherals (Sheet 8 of 24)

Address	Name	Description	Page
0x4010_0010	FFMCR	Modem Control register	10-29
0x4010_0014	FFLSR	Line Status register	10-26
0x4010_0018	FFMSR	Modem Status register	10-31
0x4010_001C	FFSPR	Scratch Pad register	10-33
0x4010_0020	FFISR	Infrared Select register	10-33
0x4010_0024	FFFOR	Receive FIFO Occupancy register	10-22
0x4010_0028	FFABR	Auto-baud Control register	10-23
0x4010_002C	FFACR	Auto-baud Count register	10-24
0x4010_0030– 0x401F_FFFC	—	reserved	—
Bluetooth UART			
0x4020_0000	BTRBR	Receive Buffer register	10-13
0x4020_0000	BTTHR	Transmit Holding register	10-14
0x4020_0000	BTDLL	Divisor Latch register, low byte	10-14
0x4020_0004	BTIER	Interrupt Enable register	10-15
0x4020_0004	BTDLH	Divisor Latch register, high byte	10-14
0x4020_0008	BTIIR	Interrupt ID register	10-17
0x4020_0008	BTFCR	FIFO Control register	10-19
0x4020_000C	BTLCR	Line Control register	10-25
0x4020_0010	BTMCR	Modem Control register	10-29
0x4020_0014	BTLSR	Line Status register	10-26
0x4020_0018	BTMSR	Modem Status register	10-31
0x4020_001C	BTSPR	Scratch Pad register	10-33
0x4020_0020	BTISR	Infrared Select register	10-33
0x4020_0024	BTFOR	Receive FIFO Occupancy register	10-22
0x4020_0028	BTABR	Auto-Baud Control register	10-23
0x4020_002C	BTACR	Auto-Baud Count register	10-24
0x4020_0030– 0x402F_FFFC	—	reserved	—
Standard I^2C			
0x4030_1680	IBMR	I^2C Bus Monitor register	9-30
0x4030_1684	—	reserved	—
0x4030_1688	IDBR	I^2C Data Buffer register	9-29
0x4030_168C	—	reserved	—
0x4030_1690	ICR	I^2C Control register	9-23
0x4030_1694	—	reserved	—
0x4030_1698	ISR	I^2C Status register	9-26

Table 28-8. Register Address Summary—Peripherals (Sheet 9 of 24)

Address	Name	Description	Page
0x4030_169C	—	reserved	—
0x4030_16A0	ISAR	I^2C Slave Address register	9-28
0x4030_16A4–0x403F_FFFC	—	reserved	—
I^2S Controller			
0x4040_0000	SACR0	Serial Audio Global Control register	14-10
0x4040_0004	SACR1	Serial Audio I^2S/MSB-Justified Control register	14-13
0x4040_0008	—	reserved	—
0x4040_000C	SASR0	Serial Audio I^2S/MSB-Justified Interface and FIFO Status register	14-14
0x4040_0010	—	reserved	—
0x4040_0014	SAIMR	Serial Audio Interrupt Mask register	14-18
0x4040_0018	SAICR	Serial Audio Interrupt Clear register	14-17
0x4040_001C–0x4040_005C	—	reserved	—
0x4040_0060	SADIV	Audio Clock Divider register	14-16
0x4040_0064–0x4040_007C	—	reserved	—
0x4040-0080	SADR	Serial Audio Data register (TX and RX FIFO access register).	14-18
0x4040-0084–0x404F_FFFC	—	reserved	—
AC '97 Controller			
0x4050_0000	POCR	PCM Out Control register	13-27
0x4050_0004	PCMICR	PCM In Control register	13-28
0x4050_0008	MCCR	Microphone In Control register	13-33
0x4050_000C	GCR	Global Control register	13-22
0x4050_0010	POSR	PCM Out Status register	13-29
0x4050_0014	PCMISR	PCM In Status register	13-30
0x4050_0018	MCSR	MIC In Status register	13-34
0x4050_001C	GSR	Global Status register	13-24
0x4050_0020	CAR	Codec Access register	13-31
0x4050_0024–0x4050_003C	—	reserved	—
0x4050_0040	PCDR	PCM Data register	13-32
0x4050_0044–0x4050_005C	—	reserved	—
0x4050_0060	MCDR	MIC In Data register	13-35
0x4050_0064–0x4050_00FC	—	reserved	—
0x4050_0100	MOCR	Modem Out Control register	13-36
0x4050_0104	—	reserved	—

Table 28-8. Register Address Summary—Peripherals (Sheet 10 of 24)

Address	Name	Description	Page
0x4050_0108	MICR	Modem In Control register	13-37
0x4050_010C	—	reserved	—
0x4050_0110	MOSR	Modem Out Status register	13-38
0x4050_0114	—	reserved	—
0x4050_0118	MISR	Modem In Status register	13-39
0x4050_011C–0x4050_013C	—	reserved	—
0x4050_0140	MODR	Modem Data register	13-40
0x4050_0144–0x4050_01FC	—	reserved	—
(0x4050_0200–0x4050_02FC) with all in increments of 0x00004	—	Primary Audio Codec registers	13-41
(0x4050_0300–0x4050_03FC) with all in increments of 0x00004	—	Secondary Audio Codec registers	13-41
(0x4050_0400–0x4050_04FC) with all in increments of 0x0000_0004	—	Primary Modem Codec registers	13-41
(0x4050_0500–0x4050_05FC) with all in increments of 0x00004	—	Secondary Modem Codec registers	13-41
0x4050_0600–0x405F_FFFC	—	reserved	13-41
USB Client Controller			
0x4060_0000	UDCCR	UDC Control register	12-31
0x4060_0004	UDCICR0	UDC Interrupt Control register 0	12-35
0x4060_0008	UDCCIR1	UDC Interrupt Control register 1	12-35
0x4060_000C	UDCISR0	UDC Interrupt Status register 0	12-50
0x4060_0010	UDCSIR1	UDC Interrupt Status register 1	12-50
0x4060_0014	UDCFNR	UDC Frame Number register	12-53
0x4060_0018	UDCOTGICR	UDC OTG Interrupt Control register	12-35
0x4060_001C	UDCOTGISR	UDC OTG Interrupt Status register	12-50
0x4060_0020	UP2OCR	USB Port 2 Output Control register	12-41
0x4060_0024	UP3OCR	USB Port 3 Output Control register	12-47
0x4060_0028–0x4060_00FC	—	reserved	—

Table 28-8. Register Address Summary—Peripherals (Sheet 11 of 24)

Address	Name	Description	Page
0x4060_0100	UDCCSR0	UDC Control/Status register—Endpoint 0	12-54
0x4060_0104	UDCCSRA	UDC Control/Status register—Endpoint A	12-57
0x4060_0108	UDCCSRB	UDC Control/Status register—Endpoint B	12-57
0x4060_010C	UDCCSRC	UDC Control/Status register—Endpoint C	12-57
0x4060_0110	UDCCSRD	UDC Control/Status register—Endpoint D	12-57
0x4060_0114	UDCCSRE	UDC Control/Status register—Endpoint E	12-57
0x4060_0118	UDCCSRF	UDC Control/Status register—Endpoint F	12-57
0x4060_011C	UDCCSRG	UDC Control/Status register—Endpoint G	12-57
0x4060_0120	UDCCSRH	UDC Control/Status register—Endpoint H	12-57
0x4060_0124	UDCCSRI	UDC Control/Status register—Endpoint I	12-57
0x4060_0128	UDCCSRJ	UDC Control/Status register—Endpoint J	12-57
0x4060_012C	UDCCSRK	UDC Control/Status register—Endpoint K	12-57
0x4060_0130	UDCCSRL	UDC Control/Status register—Endpoint L	12-57
0x4060_0134	UDCCSRM	UDC Control/Status register—Endpoint M	12-57
0x4060_0138	UDCCSRN	UDC Control/Status register—Endpoint N	12-57
0x4060_013C	UDCCSRP	UDC Control/Status register—Endpoint P	12-57
0x4060_0140	UDCCSRQ	UDC Control/Status register—Endpoint Q	12-57
0x4060_0144	UDCCSRR	UDC Control/Status register—Endpoint R	12-57
0x4060_0148	UDCCSRS	UDC Control/Status register—Endpoint S	12-57
0x4060_014C	UDCCSRT	UDC Control/Status register—Endpoint T	12-57
0x4060_0150	UDCCSRU	UDC Control/Status register—Endpoint U	12-57
0x4060_0154	UDCCSRV	UDC Control/Status register—Endpoint V	12-57
0x4060_0158	UDCCSRW	UDC Control/Status register—Endpoint W	12-57
0x4060_015C	UDCCSRX	UDC Control/Status register—Endpoint X	12-57
0x4060_0160– 0x4060_01FC	—	reserved	—
0x4060_0200	UDCBCR0	UDC Byte Count register—Endpoint 0	12-63
0x4060_0204	UDCBCRA	UDC Byte Count register—Endpoint A	12-63
0x4060_0208	UDCBCRB	UDC Byte Count register—Endpoint B	12-63
0x4060_020C	UDCBCRC	UDC Byte Count register—Endpoint C	12-63
0x4060_0210	UDCBCRD	UDC Byte Count register—Endpoint D	12-63
0x4060_0214	UDCBCRE	UDC Byte Count register—Endpoint E	12-63
0x4060_0218	UDCBCRF	UDC Byte Count register—Endpoint F	12-63
0x4060_021C	UDCBCRG	UDC Byte Count register—Endpoint G	12-63
0x4060_0220	UDCBCRH	UDC Byte Count register—Endpoint H	12-63
0x4060_0224	UDCBCRI	UDC Byte Count register—Endpoint I	12-63
0x4060_0228	UDCBCRJ	UDC Byte Count register—Endpoint J	12-63
0x4060_022C	UDCBCRK	UDC Byte Count register—Endpoint K	12-63

Table 28-8. Register Address Summary—Peripherals (Sheet 12 of 24)

Address	Name	Description	Page
0x4060_0230	UDCBCRL	UDC Byte Count register—Endpoint L	12-63
0x4060_0234	UDCBCRM	UDC Byte Count register—Endpoint M	12-63
0x4060_0238	UDCBCRN	UDC Byte Count register—Endpoint N	12-63
0x4060_023C	UDCBCRP	UDC Byte Count register—Endpoint P	12-63
0x4060_0240	UDCBCRQ	UDC Byte Count register—Endpoint Q	12-63
0x4060_0244	UDCBCRR	UDC Byte Count register—Endpoint R	12-63
0x4060_0248	UDCBCRS	UDC Byte Count register—Endpoint S	12-63
0x4060_024C	UDCBCRT	UDC Byte Count register—Endpoint T	12-63
0x4060_0250	UDCBCRU	UDC Byte Count register—Endpoint U	12-63
0x4060_0254	UDCBCRV	UDC Byte Count register—Endpoint V	12-63
0x4060_0258	UDCBCRW	UDC Byte Count register—Endpoint W	12-63
0x4060_025C	UDCBCRX	UDC Byte Count register—Endpoint X	12-63
0x4060_0260–0x4060_02FC	—	reserved	—
0x4060_0300	UDCDR0	UDC Data register—Endpoint 0	12-63
0x4060_0304	UDCDRA	UDC Data register—Endpoint A	12-63
0x4060_0308	UDCDRB	UDC Data register—Endpoint B	12-63
0x4060_030C	UDCDRC	UDC Data register—Endpoint C	12-63
0x4060_0310	UDCDRD	UDC Data register—Endpoint D	12-63
0x4060_0314	UDCDRE	UDC Data register—Endpoint E	12-63
0x4060_0318	UDCDRF	UDC Data register—Endpoint F	12-63
0x4060_031C	UDCDRG	UDC Data register—Endpoint G	12-63
0x4060_0320	UDCDRH	UDC Data register—Endpoint H	12-63
0x4060_0324	UDCDRI	UDC Data register—Endpoint I	12-63
0x4060_0328	UDCDRJ	UDC Data register—Endpoint J	12-63
0x4060_032C	UDCDRK	UDC Data register—Endpoint K	12-63
0x4060_0330	UDCDRL	UDC Data register—Endpoint L	12-63
0x4060_0334	UDCDRM	UDC Data register—Endpoint M	12-63
0x4060_0338	UDCDRN	UDC Data register—Endpoint N	12-63
0x4060_033C	UDCDRP	UDC Data register—Endpoint P	12-63
0x4060_0340	UDCDRQ	UDC Data register—Endpoint Q	12-63
0x4060_0344	UDCDRR	UDC Data register—Endpoint R	12-63
0x4060_0348	UDCDRS	UDC Data register—Endpoint S	12-63
0x4060_034C	UDCDRT	UDC Data register—Endpoint T	12-63
0x4060_0350	UDCDRU	UDC Data register—Endpoint U	12-63
0x4060_0354	UDCDRV	UDC Data register—Endpoint V	12-63
0x4060_0358	UDCDRW	UDC Data register—Endpoint W	12-63
0x4060_035C	UDCDRX	UDC Data register—Endpoint X	12-63

Table 28-8. Register Address Summary—Peripherals (Sheet 13 of 24)

Address	Name	Description	Page
0x4060_0360–0x4060_03FC	—	reserved	—
0x4060_0400	—	reserved	—
0x4060_0404	UDCCRA	UDC Configuration register—Endpoint A	12-65
0x4060_0408	UDCCRB	UDC Configuration register—Endpoint B	12-65
0x4060_040C	UDCCRC	UDC Configuration register—Endpoint C	12-65
0x4060_0410	UDCCRD	UDC Configuration register—Endpoint D	12-65
0x4060_0414	UDCCRE	UDC Configuration register—Endpoint E	12-65
0x4060_0418	UDCCRF	UDC Configuration register—Endpoint F	12-65
0x4060_041C	UDCCRG	UDC Configuration register—Endpoint G	12-65
0x4060_0420	UDCCRH	UDC Configuration register—Endpoint H	12-65
0x4060_0424	UDCCRI	UDC Configuration register—Endpoint I	12-65
0x4060_0428	UDCCRJ	UDC Configuration register—Endpoint J	12-65
0x4060_042C	UDCCRK	UDC Configuration register—Endpoint K	12-65
0x4060_0430	UDCCRL	UDC Configuration register—Endpoint L	12-65
0x4060_0434	UDCCRM	UDC Configuration register—Endpoint M	12-65
0x4060_0438	UDCCRN	UDC Configuration register—Endpoint N	12-65
0x4060_043C	UDCCRP	UDC Configuration register—Endpoint P	12-65
0x4060_0440	UDCCRQ	UDC Configuration register—Endpoint Q	12-65
0x4060_0444	UDCCRR	UDC Configuration register—Endpoint R	12-65
0x4060_0448	UDCCRS	UDC Configuration register—Endpoint S	12-65
0x4060_044C	UDCCRT	UDC Configuration register—Endpoint T	12-65
0x4060_0450	UDCCRU	UDC Configuration register—Endpoint U	12-65
0x4060_0454	UDCCRV	UDC Configuration register—Endpoint V	12-65
0x4060_0458	UDCCRW	UDC Configuration register—Endpoint W	12-65
0x4060_045C	UDCCRX	UDC Configuration register—Endpoint X	12-65
0x4060_0460–0x406F_FFFC	—	reserved	—
Standard UART			
0x4070_0000	STRBR	Receive Buffer register	10-13
0x4070_0000	STTHR	Transmit Holding register	10-14
0x4070_0000	STDLL	Divisor Latch register, low byte	10-14
0x4070_0004	STIER	Interrupt Enable register	10-15
0x4070_0004	STDLH	Divisor Latch register, high byte	10-14
0x4070_0008	STIIR	Interrupt ID register	10-17
0x4070_0008	STFCR	FIFO Control register	10-19
0x4070_000C	STLCR	Line Control register	10-25
0x4070_0010	STMCR	Modem Control register	10-29

Table 28-8. Register Address Summary—Peripherals (Sheet 14 of 24)

Address	Name	Description	Page
0x4070_0014	STLSR	Line Status register	10-26
0x4070_0018	STMSR	Modem Status register	10-31
0x4070_001C	STSPR	Scratch Pad register	10-33
0x4070_0020	STISR	Infrared Select register	10-33
0x4070_0024	STFOR	Receive FIFO Occupancy register	10-22
0x4070_0028	STABR	Auto-Baud Control register	10-23
0x4070_002C	STACR	Auto-Baud Count register	10-24
0x4070_0030– 0x407F_FFFC	—	reserved	—
Infrared Communications Port			
0x4080_0000	ICCR0	FICP Control register 0	11-10
0x4080_0004	ICCR1	FICP Control register 1	11-13
0x4080_0008	ICCR2	FICP Control register 2	11-14
0x4080_000C	ICDR	FICP Data register	11-15
0x4080_0010	—	reserved	—
0x4080_0014	ICSR0	FICP Status register 0	11-16
0x4080_0018	ICSR1	FICP Status register 1	11-18
0x 4080 001C	ICFOR	FICP FIFO Occupancy Status register	11-19
0x 4080 0020– 0x 4080 FFFC	—	reserved	—
Real-Time Clock			
0x4090_0000	RCNR	RTC Counter register	21-24
0x4090_0004	RTAR	RTC Alarm register	21-19
0x4090_0008	RTSR	RTC Status register	21-17
0x4090_000C	RTTR	RTC Timer Trim register	21-16
0x4090_0010	RDCR	RTC Day Counter register	21-24
0x4090_0014	RYCR	RTC Year Counter register	21-25
0x4090_0018	RDAR1	RTC Wristwatch Day Alarm register 1	21-20
0x4090_001C	RYAR1	RTC Wristwatch Year Alarm register 1	21-21
0x4090_0020	RDAR2	RTC Wristwatch Day Alarm register 2	21-20
0x4090_0024	RYAR2	RTC Wristwatch Year Alarm register 2	21-21
0x4090_0028	SWCR	RTC Stopwatch Counter register	21-26
0x4090_002C	SWAR1	RTC Stopwatch Alarm register 1	21-22
0x4090_0030	SWAR2	RTC Stopwatch Alarm register 2	21-22
0x4090_0034	RTCPICR	RTC Periodic Interrupt Counter register	21-27
0x4090_0038	PIAR	RTC Periodic Interrupt Alarm register	21-23
0x4090_003C– 0x409F_FFFC	—	Reserved	—

Table 28-8. Register Address Summary—Peripherals (Sheet 15 of 24)

Address	Name	Description	Page
OS Timers			
0x40A0_0000	OSMR0	OS Timer Match 0 register	22-15
0x40A0_0004	OSMR1	OS Timer Match 1 register	22-15
0x40A0_0008	OSMR2	OS Timer Match 2 register	22-15
0x40A0_000C	OSMR3	OS Timer Match 3 register	22-15
0x40A0_0010	OSCR0	OS Timer Counter 0 register	22-17
0x40A0_0014	OSSR	OS Timer Status register (used for all counters)	22-18
0x40A0_0018	OWER	OS Timer Watchdog Enable register	22-16
0x40A0_001C	OIER	OS Timer Interrupt Enable register (used for all counters)	22-16
0x40A0_0020	OSNR	OS Timer Snapshot register	22-19
0x40A0_0024– 0x40A0_003C	—	reserved	—
0x40A0_0040	OSCR4	OS Timer Counter 4–11 registers	22-17
0x40A0_0044	OSCR5		
0x40A0_0048	OSCR6		
0x40A0_004C	OSCR7		
0x40A0_0050	OSCR8		
0x40A0_0054	OSCR9		
0x40A0_0058	OSCR10		
0x40A0_005C	OSCR11		
0x40A0_0060– 0x40A0_007C	—	reserved	—
0x40A0_0080	OSMR4	OS Timer Match 4–11 registers	22-15
0x40A0_0084	OSMR5		
0x40A0_0088	OSMR6		
0x40A0_008C	OSMR7		
0x40A0_0090	OSMR8		
0x40A0_0094	OSMR9		
0x40A0_0098	OSMR10		
0x40A0_009C	OSMR11		
0x40A0_00A0– 0x40A0_00BC	—	reserved	—
0x40A0_00C0	OMCR4	OS Match Control 4–7 registers	22-9
0x40A0_00C4	OMCR5		
0x40A0_00C8	OMCR6		
0x40A0_00CC	OMCR7		
0x40A0_00D0	OMCR8	OS Match Control 8 register	22-11
0x40A0_00D4	OMCR9	OS Match Control 9 register	22-13

Table 28-8. Register Address Summary—Peripherals (Sheet 16 of 24)

Address	Name	Description	Page
0x40A0_00D8	OMCR10	OS Match Control 10 register	22-11
0x40A0_00DC	OMCR11	OS Match Control 11 register	22-13
0x40A0_00E0–0x40AF_FFFC	—	reserved	—
Pulse-Width Modulation			
0x40B0_0000	PWMCR0	PWM 0 Control register	23-7
0x40B0_0004	PWMDCR0	PWM 0 Duty Cycle register	23-8
0x40B0_0008	PWMPCR0	PWM 0 Period register	23-9
0x40B0_000C	—	reserved	—
0x40B0_0010	PWMCR2	PWM 2 Control register	23-7
0x40B0_0014	PWMDCR2	PWM 2 Duty Cycle register	23-8
0x40B0_0018	PWMPCR2	PWM 2 Period register	23-9
0x40B0_001C–0x40BF_FFFC	—	reserved	—
0x40C0_0000	PWMCR1	PWM 1 Control register	23-7
0x40C0_0004	PWMDCR1	PWM 1 Duty Cycle register	23-8
0x40C0_0008	PWMPCR1	PWM 1 Period register	23-9
0x40C0_000C	—	reserved	—
0x40C0_0010	PWMCR3	PWM 3 Control register	23-7
0x40C0_0014	PWMDCR3	PWM 3 Duty Cycle register	23-8
0x40C0_0018	PWMPCR3	PWM 3 Period register	23-9
0x40C0_001C–0x40CF_FFFC	—	reserved	—
Interrupt Controller			
0x40D0_0000	ICIP	Interrupt Controller IRQ Pending register	25-11
0x40D0_0004	ICMR	Interrupt Controller Mask register	25-20
0x40D0_0008	ICLR	Interrupt Controller Level register	25-24
0x40D0_000C	ICFP	Interrupt Controller FIQ Pending register	25-15
0x40D0_0010	ICPR	Interrupt Controller Pending register	25-6
0x40D0_0014	ICCR	Interrupt Controller Control register	25-27
0x40D0_0018	ICHP	Interrupt Controller Highest Priority register	25-30
0x40D0_001C–0x40D0_0098	IPR0–IPR31	Interrupt Priority registers for Priorities 0–31	25-29
0x40D0_009C	ICIP2	Interrupt Controller IRQ Pending register 2	25-10
0x40D0_00A0	ICMR2	Interrupt Controller Mask register 2	25-23
0x40D0_00A4	ICLR2	Interrupt Controller Level register 2	25-27
0x40D0_00A8	ICFP2	Interrupt Controller FIQ Pending register 2	25-19
0x40D0_00AC	ICPR2	Interrupt Controller Pending register 2	25-6

Table 28-8. Register Address Summary—Peripherals (Sheet 17 of 24)

Address	Name	Description	Page
0x40D0_00B0– 0x40D0_00CC	IPR32–IPR39	Interrupt Priority registers for Priorities 32–39	25-29
0x40D0_00D0– 0x40DF_FFFC	—	reserved	—
General-Purpose I/O (GPIO) Controller			
0x40E0_ 0000	GPLR0	GPIO Pin-Level register GPIO<31:0>	24-28
0x40E0_ 0004	GPLR1	GPIO Pin-Level register GPIO<63:32>	24-28
0x40E0_ 0008	GPLR2	GPIO Pin-Level register GPIO<95:64>	24-28
0x40E0_ 000C	GPDR0	GPIO Pin Direction register GPIO<31:0>	24-11
0x40E0_ 0010	GPDR1	GPIO Pin Direction register GPIO<63:32>	24-11
0x40E0_ 0014	GPDR2	GPIO Pin Direction register GPIO<95:64>	24-11
0x40E0_ 0018	GPSR0	GPIO Pin Output Set register GPIO<31:0>	24-14
0x40E0_ 001C	GPSR1	GPIO Pin Output Set register GPIO<63:32>	24-14
0x40E0_ 0020	GPSR2	GPIO Pin Output Set register GPIO<95:64>	24-14
0x40E0_ 0024	GPCR0	GPIO Pin Output Clear register GPIO<31:0>	24-14
0x40E0_ 0028	GPCR1	GPIO Pin Output Clear register GPIO <63:32>	24-14
0x40E0_ 002C	GPCR2	GPIO pin Output Clear register GPIO <95:64>	24-14
0x40E0_ 0030	GRER0	GPIO Rising-Edge Detect Enable register GPIO<31:0>	24-18
0x40E0_ 0034	GRER1	GPIO Rising-Edge Detect Enable register GPIO<63:32>	24-18
0x40E0_ 0038	GRER2	GPIO Rising-Edge Detect Enable register GPIO<95:64>	24-18
0x40E0_ 003C	GFER0	GPIO Falling-Edge Detect Enable register GPIO<31:0>	24-18
0x40E0_ 0040	GFER1	GPIO Falling-Edge Detect Enable register GPIO<63:32>	24-18
0x40E0_ 0044	GFER2	GPIO Falling-Edge Detect Enable register GPIO<95:64>	24-18
0x40E0_ 0048	GEDR0	GPIO Edge Detect Status register GPIO<31:0>	24-30
0x40E0_ 004C	GEDR1	GPIO Edge Detect Status register GPIO<63:32>	24-30
0x40E0_ 0050	GEDR2	GPIO Edge Detect Status register GPIO<95:64>	24-30
0x40E0_ 0054	GAFR0_L	GPIO Alternate Function register GPIO<15:0>	24-23
0x40E0_ 0058	GAFR0_U	GPIO Alternate Function register GPIO<31:16>	24-23
0x40E0_ 005C	GAFR1_L	GPIO Alternate Function register GPIO<47:32>	24-23
0x40E0_ 0060	GAFR1_U	GPIO Alternate Function register GPIO<63:48>	24-23
0x40E0_ 0064	GAFR2_L	GPIO Alternate Function register GPIO<79:64>	24-23
0x40E0_ 0068	GAFR2_U	GPIO Alternate Function register GPIO <95:80>	24-23
0x40E0_ 006C	GAFR3_L	GPIO Alternate Function register GPIO<111:96>	24-23
0x40E0_ 0070	GAFR3_U	GPIO Alternate Function register GPIO<120:112>	24-23
0x40E0_ 0074– 0x40E0_ 00FC	—	reserved	—
0x40E0_ 0100	GPLR3	GPIO Pin-Level register GPIO<120:96>	24-28
0x40E0_ 0104– 0x40E0_ 0108	—	reserved	—

Table 28-8. Register Address Summary—Peripherals (Sheet 18 of 24)

Address	Name	Description	Page
0x40E0_ 010C	GPDR3	GPIO Pin Direction register GPIO<120:96>	24-11
0x40E0_ 0110–0x40E0_ 0114	—	reserved	—
0x40E0_ 0118	GPSR3	GPIO Pin Output Set register GPIO<120:96>	24-14
0x40E0_ 011C–0x40E0_ 0120	—	reserved	—
0x40E0_ 0124	GPCR3	GPIO Pin Output Clear register GPIO<120:96>	24-14
0x40E0_ 0128–0x40E0_ 012C	—	reserved	—
0x40E0_ 0130	GRER3	GPIO Rising-Edge Detect Enable register GPIO<120:96>	24-18
0x40E0_ 0134–0x40E0_ 0138	—	reserved	—
0x40E0_ 013C	GFER3	GPIO Falling-Edge Detect Enable register GPIO<120:96>	24-18
0x40E0_ 0140–0x40E0_ 0144	—	reserved	—
0x40E0_ 0148	GEDR3	GPIO Edge Detect Status register GPIO<120:96>	24-18
0x40E0_ 014C–0x40EF_FFFC	—	reserved	—
Power Manager and Reset Control			
0x40F0_0000	PMCR	Power Manager Control register	3-68
0x40F0_0004	PSSR	Power Manager Sleep Status register	3-70
0x40F0_0008	PSPR	Power Manager Scratch Pad register	3-73
0x40F0_000C	PWER	Power Manager Wake-Up Enable register	3-74
0x40F0_0010	PRER	Power Manager Rising-Edge Detect Enable register	3-77
0x40F0_0014	PFER	Power Manager Falling-Edge Detect Enable register	3-78
0x40F0_0018	PEDR	Power Manager Edge-Detect Status register	3-79
0x40F0_001C	PCFR	Power Manager General Configuration register	3-80
0x40F0_0020	PGSR0	Power Manager GPIO Sleep State register for GPIO<31:0>	3-83
0x40F0_0024	PGSR1	Power Manager GPIO Sleep State register for GPIO<63:32>	3-83
0x40F0_0028	PGSR2	Power Manager GPIO Sleep State register for GPIO<95:64>	3-83
0x40F0_002C	PGSR3	Power Manager GPIO Sleep State register for GPIO<120:96>	3-83
0x40F0_0030	RCSR	Reset Controller Status register	3-84
0x40F0_0034	PSLR	Power Manager Sleep Configuration register	3-85
0x40F0_0038	PSTR	Power Manager Standby Configuration register	3-88
0x40F0_003C	—	reserved	—
0x40F0_0040	PVCR	Power Manager Voltage Change Control register	3-89
0x40F0_0044–0x40F0_0048	—	reserved	—
0x40F0_004C	PUCR	Power Manager USIM Card Control/Status register	3-90
0x40F0_0050	PKWR	Power Manager Keyboard Wake-Up Enable register	3-92

Table 28-8. Register Address Summary—Peripherals (Sheet 19 of 24)

Address	Name	Description	Page
0x40F0_0054	PKSR	Power Manager Keyboard Level-Detect Status register	3-93
0x40F0_0058–0x40F0_007C	—	reserved	—
0x40F0_0080–0x40F0_00FC	PCMD0–PCMD31	Power Manager I2C Command register File	3-94
0x40F0_0100–0x40F0_017C	—	reserved	—
Power Manager I2C			
0x40F0_0180	PIBMR	Power Manager I2C Bus Monitor register	9-30
0x40F0_0184	—	reserved	—
0x40F0_0188	PIDBR	Power Manager I2C Data Buffer register	9-29
0x40F0_018C	—	reserved	—
0x40F0_0190	PICR	Power Manager I2C Control register	9-23
0x40F0_0194	—	reserved	—
0x40F0_0198	PISR	Power Manager I2C Status register	9-26
0x40F0_019C	—	reserved	—
0x40F0_01A0	PISAR	Power Manager I2C Slave Address register	9-28
0x40F0_01A4–0x40FF_FFFC	—	reserved	—
Synchronous Serial Port 1			
0x4100_0000	SSCR0_1	SSP 1 Control register 0	8-25
0x4100_0004	SSCR1_1	SSP 1 Control register 1	8-30
0x4100_0008	SSSR_1	SSP 1 Status register	8-43
0x4100_000C	SSITR_1	SSP 1 Interrupt Test register	8-42
0x4100_0010	SSDR_1	SSP 1 Data Write register/Data Read register	8-48
0x4100_0014–0x4100_0024	—	reserved	—
0x4100_0028	SSTO_1	SSP 1 Time-Out register	8-41
0x4100_002C	SSPSP_1	SSP 1 Programmable Serial Protocol	8-39
0x4100_0030	SSTSA_1	SSP1 TX Timeslot Active register	8-48
0x4100_0034	SSRSA_1	SSP1 RX Timeslot Active register	8-49
0x4100_0038	SSTSS_1	SSP1 Timeslot Status register	8-50
0x4100_003C	SSACD_1	SSP1 Audio Clock Divider register	8-51
0x4100_0040–0x416F_FFFC	—	reserved	—
MultiMediaCard/SD/SDIO Controller			
0x4110_0000	MMC_STRPCL	MMC Clock Start/Stop register	15-29
0x4110_0004	MMC_STAT	MMC Status register	15-29
0x4110_0008	MMC_CLKRT	MMC Clock Rate register	15-31

Table 28-8. Register Address Summary—Peripherals (Sheet 20 of 24)

Address	Name	Description	Page
0x4110_000C	MMC_SPI	MMC SPI Mode register	15-31
0x4110_0010	MMC_CMDAT	MMC Command/Data register	15-32
0x4110_0014	MMC_RESTO	MMC Response Time-Out register	15-34
0x4110_0018	MMC_RDTO	MMC Read Time-Out register	15-34
0x4110_001C	MMC_BLKLEN	MMC Block Length register	15-35
0x4110_0020	MMC_NUMBLK	MMC Number of Blocks register	15-35
0x4110_0024	MMC_PRTBUF	MMC Buffer Partly Full register	15-36
0x4110_0028	MMC_I_MASK	MMC Interrupt Mask register	15-36
0x4110_002C	MMC_I_REG	MMC Interrupt Request register	15-38
0x4110_0030	MMC_CMD	MMC Command register	15-41
0x4110_0034	MMC_ARGH	MMC Argument High register	15-41
0x4110 _0038	MMC_ARGL	MMC Argument Low register	15-42
0x4110_003C	MMC_RES	MMC Response FIFO	15-42
0x4110_0040	MMC_RXFIFO	MMC Receive FIFO	15-42
0x4110_0044	MMC_TXFIFO	MMC Transmit FIFO	15-43
0x4110_0048	MMC_RDWAIT	MMC RD_WAIT register	15-43
0x4110_004C	MMC_BLKS_REM	MMC Blocks Remaining register	15-44
Clocks Manager			
0x4130_0000	CCCR	Core Clock Configuration register	3-95
0x4130_0004	CKEN	Clock Enable register	3-98
0x4130_0008	OSCC	Oscillator Configuration register	3-99
0x4130_000C	CCSR	Core Clock Status register	3-101
0x4130_0010–0x413F_FFFC	—	reserved	—
Mobile Scalable Link (MSL) Interface			
0x4140_0004	BBFIFO1	MSL Channel 1 Receive/Transmit FIFO register	16-13
0x4140_0008	BBFIFO2	MSL Channel 2 Receive/Transmit FIFO register	16-13
0x4140_000C	BBFIFO3	MSL Channel 3 Receive/Transmit FIFO register	16-13
0x4140_0010	BBFIFO4	MSL Channel 4 Receive/Transmit FIFO register	16-13
0x4140_0014	BBFIFO5	MSL Channel 5 Receive/Transmit FIFO register	16-13
0x4140_0018	BBFIFO6	MSL Channel 6 Receive/Transmit FIFO register	16-13
0x4140_001C	BBFIFO7	MSL Channel 7 Receive/Transmit FIFO register	16-13
0x4140_0020–0x4140_0040	—	reserved	—
0x4140_0044	BBCFG1	MSL Channel 1 Configuration register	16-15
0x4140_0048	BBCFG2	MSL Channel 2 Configuration register	16-15
0x4140_004C	BBCFG3	MSL Channel 3 Configuration register	16-15
0x4140_0050	BBCFG4	MSL Channel 4 Configuration register	16-15

Table 28-8. Register Address Summary—Peripherals (Sheet 21 of 24)

Address	Name	Description	Page
0x4140_0054	BBCFG5	MSL Channel 5 Configuration register	16-15
0x4140_0058	BBCFG6	MSL Channel 6 Configuration register	16-15
0x4140_005C	BBCFG7	MSL Channel 7 Configuration register	16-15
0x4140_0060–0x4140_0080	—	reserved	—
0x4140_0084	BBSTAT1	MSL Channel 1 Status register	16-19
0x4140_0088	BBSTAT2	MSL Channel 2 Status register	16-19
0x4140_008C	BBSTAT3	MSL Channel 3 Status register	16-19
0x4140_0090	BBSTAT4	MSL Channel 4 Status register	16-19
0x4140_0094	BBSTAT5	MSL Channel 5 Status register	16-19
0x4140_0098	BBSTAT6	MSL Channel 6 Status register	16-19
0x4140_009C	BBSTAT7	MSL Channel 7 Status register	16-19
0x4140_00A0–0x4140_00C0	—	reserved	—
0x4140_00C4	BBEOM1	MSL Channel 1 EOM register	16-22
0x4140_00C8	BBEOM2	MSL Channel 2 EOM register	16-22
0x4140_00CC	BBEOM3	MSL Channel 3 EOM register	16-22
0x4140_00D0	BBEOM4	MSL Channel 4 EOM register	16-22
0x4140_00D4	BBEOM5	MSL Channel 5 EOM register	16-22
0x4140_00D8	BBEOM6	MSL Channel 6 EOM register	16-22
0x4140_00DC	BBEOM7	MSL Channel 7 EOM register	16-22
0x4140_00E0–0x4140_00FC	—	reserved	—
0x4140_0100–0x4140_0104	—	reserved	—
0x4140_0108	BBIID	MSL Interrupt ID register	16-23
0x4140_010C	—	reserved	
0x4140_0110	BBFREQ	MSL Transmit Frequency Select register	10-6
0x4140_0114	BBWAIT	MSL Wait Count register	16-24
0x4140_0118	BBCST	MSL Clock Stop Time register	16-25
0x4140_011C–0x4140_0138	—	reserved	—
0x4140_013C	—	reserved	—
0x4140_0140	BBWAKE	MSL Wake-Up register	16-26
0x4140_0144	BBITFC	MSL Interface Width register	10-6
0x4140_0148–0x414F_FFFC	—	reserved	—

Table 28-8. Register Address Summary—Peripherals (Sheet 22 of 24)

Address	Name	Description	Page
Keypad Interface			
0x4150_0000	KPC	Keypad Interface Control register	18-12
0x4150_0004	—	reserved	—
0x4150_0008	KPDK	Keypad Interface Direct Key register	18-16
0x4150_000C	—	reserved	—
0x4150_0010	KPREC	Keypad Interface Rotary Encoder Count register	18-17
0x4150_0014	—	reserved	—
0x4150_0018	KPMK	Keypad Interface Matrix Key register	18-18
0x4150_001C	—	reserved	—
0x4150_0020	KPAS	Keypad Interface Automatic Scan register	18-18
0x4150_0024	—	reserved	—
0x4150_0028	KPASMKP0	Keypad Interface Automatic Scan Multiple Keypress register 0	18-20
0x4150_002C	—	reserved	—
0x4150_0030	KPASMKP1	Keypad Interface Automatic Scan Multiple Keypress register 1	18-20
0x4150_0034	—	reserved	—
0x4150_0038	KPASMKP2	Keypad Interface Automatic Scan Multiple Keypress register 2	18-20
0x4150_003C	—	reserved	—
0x4150_0040	KPASMKP3	Keypad Interface Automatic Scan Multiple Keypress register 3	18-20
0x4150_0044	—	reserved	—
0x4150_0048	KPKDI	Keypad Interface Key Debounce Interval register	18-23
0x4150_004C– 0x415F_FFFC	—	reserved	—
Universal Subscriber ID (USIM) Interface			
0x4160_0000	RBR	USIM Receive Buffer register	19-18
0x4160_0004	THR	USIM Transmit Holding register	19-19
0x4160_0008	IER	USIM Interrupt Enable register	19-20
0x4160_000C	IIR	USIM Interrupt Identification register	19-22
0x4160_0010	FCR	USIM FIFO Control register	19-24
0x4160_0014	FSR	USIM FIFO Status register	19-26
0x4160_0018	ECR	USIM Error Control register	19-27
0x4160_001C	LCR	USIM Line Control register	19-29
0x4160_0020	USCCR	USIM Card Control register	19-31
0x4160_0024	LSR	USIM Line Status register	19-32
0x4160_0028	EGTR	USIM Extra Guard Time register	19-34
0x4160_002C	BGTR	USIM Block Guard Time register	19-34
0x4160_0030	TOR	USIM Time-Out register	19-35
0x4160_0034	CLKR	USIM Clock register	19-36

Table 28-8. Register Address Summary—Peripherals (Sheet 23 of 24)

Address	Name	Description	Page
0x4160_0038	DLR	USIM Divisor Latch register	19-37
0x4160_003C	FLR	USIM Factor Latch register	19-37
0x4160_0040	CWTR	USIM Character Waiting Time register	19-38
0x4160_0044	BWTR	USIM Block Waiting Time register	19-39
0x4160_0048–0x4160_FFFC	—	reserved	—
Synchronous Serial Port 2			
0x4170_0000	SSCR0_2	SSP2 Control register 0	8-25
0x4170_0004	SSCR1_2	SSP 2 Control register 1	8-30
0x4170_0008	SSSR_2	SSP 2 Status register	8-43
0x4170_000C	SSITR_2	SSP 2 Interrupt Test register	8-42
0x4170_0010	SSDR_2	SSP 2 Data Write register/Data Read register	8-48
0x4170_0014–0x4170_0024	—	reserved	—
0x4170_0028	SSTO_2	SSP 2 Time-Out register	8-41
0x4170_002C	SSPSP_2	SSP 2 Programmable Serial Protocol	8-39
0x4170_0030	SSTSA_2	SSP2 TX Timeslot Active register	8-48
0x4170_0034	SSRSA_2	SSP2 RX Timeslot Active register	8-49
0x4170_0038	SSTSS_2	SSP2 Timeslot Status register	8-50
0x4170_003C	SSACD_2	SSP2 Audio Clock Divider register	8-51
0x4170_0040–0x418F_FFFC	—	reserved	—
Memory Stick Host Controller			
0x4180_0000	MSCMR	MSHC Command register	17-8
0x4180_0004	MSCRSR	MSHC Control and Status register	17-9
0x4180_0008	MSINT	MSHC Interrupt and Status register	17-10
0x4180_000C	MSINTEN	MSHC Interrupt Enable register	17-11
0x4180_0010	MSCR2	MSHC Control register 2	17-12
0x4180_0014	MSACD	MSHC ACD Command register	17-13
0x4180_0018	MSRXFIFO	MSHC Receive FIFO register	17-14
0x4180_001C	MSTXFIFO	MSHC Transmit FIFO register	17-15
0x4180_0020–0x418F_FFFC	—	reserved	—
Synchronous Serial Port 3			
0x4190_0000	SSCR0_3	SSP 3 Control register 0	8-25
0x4190_0004	SSCR1_3	SSP 3 Control register 1	8-30
0x4190_0008	SSSR_3	SSP 3 Status register	8-43
0x4190_000C	SSITR_3	SSP 3 Interrupt Test register	8-42

Table 28-8. Register Address Summary—Peripherals (Sheet 24 of 24)

Address	Name	Description	Page
0x4190_0010	SSDR_3	SSP 3 Data Write register/Data Read register	8-48
0x4190_0014– 0x4190_0024	—	reserved	—
0x4190_0028	SSTO_3	SSP 3 Time-Out register	8-41
0x4190_002C	SSPSP_3	SSP 3 Programmable Serial Protocol	8-39
0x4190_0030	SSTSA_3	SSP TX Timeslot Active register	8-48
0x4190_0034	SSRSA_3	SSP RX Timeslot Active register	8-49
0x4190_0038	SSTSS_3	SSP Timeslot Status register	8-50
0x4190_003C	SSACD_3	SSP Audio Clock Divider register	8-51
0x4190_0040– 0x419f_FFFC	—	reserved	—

System Bus Arbiter 29

This chapter describes the internal system bus arbitration mechanism included in the PXA27x processor.

29.1 Overview

The PXA27x processor system bus supports six clients — the core, the DMA controller, the LCD controller, the USB host controller, and the two memory controllers (internal and external). The bus is multiplexed (instead of a three-state approach), and clients can request the bus without any limitations. Arbitration for bus access is performed by the arbiter, which is programmable through the ARB_CNTRL register.

29.2 Features

The internal system bus arbiter includes the following features:

- Programmable client weights
- Software-selectable bus parking
- Bus locking

29.3 Signal Descriptions

There are no external signals associated with the bus arbiter.

29.4 Operation

29.4.1 Programmable Weights

The lower 12 bits of the Arbiter Control register (shown in Table 29-1) determine the arbitration priority of the clients on the bus. A detailed description of the arbitration policy of the scalable programmable system bus arbiter is beyond the scope of this document. However, the values programmed in the three weight fields of the ARB_CNTRL register denote the relative importance of the three programmable clients on the bus—core, DMA controller, and LCD controller. Between them, they attempt to capture the “spread” of the transactions on the internal bus.

For example, if the core is expected to need twice the bandwidth as the LCD controller, which needs the same as that of the DMA controller, a good value to program would be 0x77E. This ensures that the “weight” of the core is twice that of the other clients on the bus. This effect could also have been achieved by 0x224 or 0x448 or any similar combination. The absolute number to chose is a trade-off between the accuracy of the arbiter in implementing the desired weight and the

maximum guaranteed time to grant for any client. Thus, 0x77E could force the LCD controller to wait for 8 transactions (7 core and 1 DMA) before getting on the bus, even as it attempts to more accurately implement the programmer's intended importance to the core over its challengers.

In general, chose smaller numbers when max_time_to_grant must be kept to a minimum, while larger numbers provide improved accuracy.

The reset value of this field is 0x234, but this is largely a formality, since only the core is expected to be executing transactions during boot-up time.

29.4.2 Bus Parking

Parking reduces the latency of a transaction by speculatively granting the bus to a client in anticipation of a request from that client. The penalty for a mis-speculation could be two to three cycles in the PXA27x processor. The bus is parked with a particular client by setting the park bit corresponding to that client (bits 31:24). Once the register is set, the bus is granted to that client if no other client asks for the bus.

An exception to this rule occurs when there are overriding circumstances. For instance, if Lock_flag (bit 23) is set and the DMA is designated as the park-client (by setting ARB_CNTRL[DMA_park]) and there is a SWAP operation in progress from the core, then the park directive is overridden. Other conditions in the system such as sleep and retries may also prevent park from taking effect. In all such conditions, the bus is not parked with any client.

If no park bits are set, the bus is not given to any client.

If more than one park bit is set, the priority is from bit 31 to bit 24 (descending).

29.5 Register Descriptions

29.5.1 Arbiter Control Register (ARB_CNTRL)

ARB_CNTRL, defined in Table 29-1, physically resides in the external memory controller. Writes to this register are immediately communicated to the arbiter.

There are three types of fields in the register—weights, parking, and a lock flag. These occupy a total of 21 bits, and the rest are reserved.

This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 29-1. ARB_CNTRL Bit Definitions

Physical Address 0x4800_0048 — **ARB_CNTRL** — **Internal Bus Arbiter**

	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
User Settings																																
Bit	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
	DMA_SLV_park	CI_park	EX_MEM_park	INT_MEM_park	USB_park	LCD_park	DMA_park	Core_park	LOCK_FLAG	reserved											LCD_wt				DMA_wt				Core_wt			
Reset	0	0	0	0	0	0	0	0	1	?	?	?	?	?	?	?	?	?	?	?	0	0	1	0	0	0	1	1	0	1	0	0

Bits	Access	Name	Description
31	R/W	DMA_SLV_park	1 = Bus is parked with DMA slave when idle
30	R/W	CI_park	1 = Bus is parked with quick capture interface when idle
29	R/W	EX_MEM_park	1 = Bus is parked with external memory controller when idle
28	R/W	INT_MEM_park	1 = Bus is parked with internal memory controller when idle
27	R/W	USB_park	1 = Bus is parked with USB host controller when idle
26	R/W	LCD_park	1 = Bus is parked with LCD controller when idle
25	R/W	DMA_park	1 = Bus is parked with DMA controller when idle
24	R/W	Core_park	1 = Bus is parked with core when idle
23	R/W	LOCK_FLAG	1 = Only locking masters gain access to the bus
22:12	R/W	reserved	reserved
11:8	R/W	LCD_Wt	LCD Priority Value Values in this field determine the relative priority of LCD requests for the bus with core and DMA requests.
7:4	R/W	DMA_Wt	DMA Priority Value Values in this field determine the relative priority of DMA requests for the bus with core and LCD requests.
3:0	R/W	Core_Wt	Core Priority Value Values in this field determine the relative priority of core requests for the bus with DMA and LCD requests.

29.6 System Considerations

29.6.1 Access Latency on System Bus

There are four masters (core, DMA, USB host, and LCD controller) on the system bus. All accesses to the external memory from any of these masters flow through the memory controller. The memory controller has four internal buffers that work as a FIFO. All requests are executed in order.

The programmer can program the system bus arbitration priority enabling one of the four masters to access the system bus with higher priority than others.

To compute the worst-case latency for an external-memory transfer in a very busy system, where all the internal bus masters are continuously trying to access the system bus (assuming that the master has highest priority programmed in the arbiter), programmers must account for the following transfers ahead of the current master accessing the external bus:

- One current external bus transfer
- Four pending transfers in the memory controller queue
- Two transfers on the system bus
- Any other transfers pending within the master
- An SDRAM refresh.

To compute the worst-case latency for an internal-memory transfer in a very busy system (assuming that the master has highest priority programmed in the arbiter and the internal memory bank is not in standby or sleep mode and the queue is empty), programmers must account for the following transfers ahead of the current master acquiring the system bus:

- Two transfers on the system bus
- Any other transfers pending within the master.

As any of the internal memory accesses is generally lower in latency, any accesses having a critical latency requirement, such as USB host isochronous buffer data, must be placed in internal memory.

29.6.2 I/O Ordering

The PXA27x processor memory controller contains queues that accept memory requests from the four internal masters (core, DMA, USB host, and LCD controller). All external memory accesses that are issued by any one master are guaranteed to complete in order. However, there is no guarantee of strict ordering of internal or external memory transactions between different masters.

Because of buffering in the memory controller, loads and stores to internal addresses (for example, on-chip memory) generally complete in a shorter time than those issued to external (memory) addresses, which makes it possible for a pair of operations to complete in the reverse of their program order. For example, in the following sequence, the store to address R4 completes before the store to address R2 because the external-memory transfer could be buffered waiting for memory, while the internal-memory transfer completes with no delay.

```
str r1, [r2]    ; store to external memory address [r2]
str r3, [r4]    ; store to internal (on-chip) memory address [r4]
```

If the two stores happen to be control operations that absolutely must complete in program order, insert a load-to-external memory followed by an operation dependent on the data from that load, as shown:

```
str r1, [r2]    ; First store is issued
ldr r5, [r6]    ; load from some external-memory address ([r2] if possible)
mov r5, r5      ; nop stalls until previous load completes
str r3, [r4]    ; second store completes in program order
```

29.6.3 Flushing the Memory Controller Buffers

Memory controller buffers work as a FIFO and are flushed if the processor reads any memory controller internal registers.

29.6.4 Semaphores

The Swap (SWP) and Swap Byte (SWPB) instructions, as described in the ARM* V5TE architecture reference, can be used for semaphore manipulation. The PXA27x processor guarantees that no other on-chip master (or process) accesses a memory location between the load-and-store portion of a SWP or SWPB to the same location.

It is not possible for an external companion chip (through the use of the MBREQ/MBGNT handshake) to take ownership of the bus in the middle of a Swap sequence.

29.6.5 Interrupts

If an interrupt is taken, software can read a single Interrupt Pending register to identify the interrupt source. Refer to Chapter 25, “Interrupt Controller” for details. It is then the responsibility of software to service the interrupt and clear it (in the source unit) before exiting the service routine.

A delay exists associated with clearing interrupts; the interrupt-service routine (ISR) must clear the interrupt early in the routine to allow time for the status bit to clear before returning from that routine.

29.7 Register Summary

Table 29-2 summarizes the register used for bus arbitration.

Table 29-2. System Bus Arbiter Register Summary

Address	Name	Description	Page
0x4800_0048	ARB_CNTRL	Arbiter Control register	29-2

Glossary

3G—An industry term that describe the next, still-to-come generation of wireless applications. It represents a move from circuit-switched communications (where a device user has to dial in to a network) to broadband, high-speed, packet-based wireless networks (which are always on). The first generation of wireless communications relied on analog technology, followed by digital wireless communications. The third generation expands the digital capabilities by including high-speed connections and increased reliability.

802.11—Wireless specifications developed by the IEEE, outlining the means to manage packet traffic over a network and ensure that packets do not collide, which could result in the loss of data, when travelling from device to device.

8PSK—8-phase shift key modulation scheme. Used in the EDGE standard.

AC ’97—AC-link standard serial interface for modem and audio.

ACK—Handshake packet indicating a positive acknowledgment.

Active Device—A device that is powered and is not in the suspended state.

Air Interface—The RF interface between a mobile cellular handset and the base station.

AMPS—Advanced Mobile Phone Service. A term used for analog technologies, the first generation of wireless technologies.

Analog—Radio signals that are converted into a format that allows them to carry data. Cellular phones and other wireless devices use analog in geographic areas with insufficient digital networks.

ARM* V5te—An ARM* architecture designation indicating the processor is conforms to ARM* architecture version 5, including “Thumb” mode and the “El Segundo” DSP extensions.

Asynchronous Data—Data transferred at irregular intervals with relaxed latency requirements.

Asynchronous RA—The incoming data rate, Fs i, and the outgoing data rate, Fs o, of the RA process are independent (in other words, there is no shared master clock). *See also* Rate Adaptation.

Asynchronous SRC—The incoming sample rate, Fs i, and outgoing sample rate, Fs o, of the SRC process are independent (in other words, there is no shared master clock). *See also* Sample Rate Conversion (SRC).

Audio device—A device that sources or sinks sampled analog data.

AWG#—The measurement of a wire’s cross-section, as defined by the American Wire Gauge standard.

b/s—Transmission rate expressed in bits per second.

B/s—Transmission rate expressed in bytes per second.

Babble—Unexpected bus activity that persists beyond a specified point in a (micro)frame.

Backlight Inverter—A device to drive cold cathode fluorescent lamps that illuminate LCD panels.

Bandwidth—The amount of data transmitted per unit of time, typically bits per second (b/s) or bytes per second (B/s). The size of a network “pipe” or channel for communication in wired networks. In wireless, it refers to the range of available frequencies that carry a signal.

Base Station—The telephone company’s interface to the Mobile Station.

BFSK—Binary Frequency Shift Keying. A coding scheme for digital data.

BGA—Ball Grid Array.

Bit—A unit of information used by digital computers. Represents the smallest piece of addressable memory within a computer. A bit expresses the choice between two possibilities and is typically represented by a logical one (1) or zero (0).

Bit Stuffing—Insertion of a “0” bit into a data stream to cause an electrical transition on the data wires, allowing a PLL to remain locked.

Blackberry—A two-way wireless device (pager) made by Research In Motion (RIM) that allows users to check e-mail and voice mail translated into text, as well as page other users of a wireless network service. It has a miniature “qwerty” keyboard that can be used by your thumbs, and uses SMS protocol. A Blackberry user must subscribe to the proprietary wireless service that allows for data transmission.

Bluetooth—A short-range wireless specification that allows for radio connections between devices within a 30-foot range of each other. The name comes from 10th-century Danish King Harald Blatand (Bluetooth), who unified Denmark and Norway.

BPSK—Binary Phase Shift Keying. A means of encoding digital data into a signal using phase-modulated communication.

BTB—Branch Target Buffer.

BTS—Base Transmitter Station.

Buffer—Storage used to compensate for a difference in data rates or time of occurrence of events, when transmitting data from one device to another.

Bulk Transfer—One of the four USB transfer types. Bulk transfers are non-periodic, large bursty communication typically used for a transfer that can use any available bandwidth and can also be delayed until bandwidth is available. *See also* Transfer Type.

Bus Enumeration—Detecting and identifying USB devices.

Byte—A data element that is eight bits in size.

Capabilities—Those attributes of a USB device that are administrated by the host.

CAS—Cycle Accurate Simulator.

CAS-B4-RAS—*See* CBR.

CBR—CAS Before RAS. Column Address Strobe Before Row Address Strobe. A fast refresh technique in which the DRAM keeps track of the next row it needs to refresh, thus simplifying what a system would have to do to refresh the part.

CDMA—Code Division Multiple Access. U.S. wireless carriers Sprint PCD and Verizon use CDMA to allocate bandwidth for users of digital wireless devices. CDMA distinguishes between multiple transmissions carried simultaneously on a single wireless signal. It carries the transmissions on that signal, freeing network room for the wireless carrier and providing interference-free calls for the user. Several versions of the standard are still under development. CDMA may increase network capacity for wireless carriers and improve the quality of wireless messaging. CDMA is an alternative to GSM.

CDPD—Cellular Digital Packet Data Telecommunications. Companies can use DCPD to transfer data on unused cellular networks to other users. If one section, or "cell" of the network is overtaxed, DCPD automatically allows for the reallocation of services.

Cellular—Technology that senses analog or digital transmissions from transmitters that have areas of coverage called cells. As a user of a cellular phone moves between transmitters from one cell to another, the users' call travels from transmitter to transmitter uninterrupted.

Characteristics—Those qualities of a USB device that are unchangeable; for example, the device class is a device characteristic.

Circuit Switched—Used by wireless carriers, this method lets a user connect to a network or the Internet by dialing in, such as with a traditional phone line. Circuit switched connections are typically slower and less reliable than packet-switched networks, but are currently the primary method of network access for wireless users in the U.S.

Client—Software resident on the host that interacts with the USB System Software to arrange data transfer between a function and the host. The client is often the data provider and consumer for transferred data.

CML—Current mode logic.

Configuring Software—Software resident on the host software that is responsible for configuring a USB device. This may be a system configuration or software specific to the device.

Control Endpoint—A pair of device endpoints with the same endpoint number that are used by a control pipe. Control endpoints transfer data in both directions and, therefore, use both endpoint directions of a device address and endpoint number combination. Thus, each control endpoint consumes two endpoint addresses.

Control Pipe—Same as a message pipe.

Control Transfer—One of the four USB transfer types. Control transfers support configuration/command/status type communication between client and function. *See also* Transfer Type.

CRC—*See* Cyclic Redundancy Check (CRC).

CSP—Chip Scale Package.

CTE—Coefficient of thermal expansion.

CTI—Computer Telephony Integration.

Cyclic Redundancy Check (CRC)—A check performed on data to see if an error has occurred in transmitting, reading, or writing the data. The result of a CRC is typically stored or transmitted with the checked data. The stored or transmitted result is compared to a CRC calculated for the data to determine if an error has occurred.

D-cache—Data cache.

DECT—Digital European Cordless Telecommunications standard.

Default Address—An address defined by the USB Specification and used by a USB device when it is first powered or reset. The default address is 00h.

Default Pipe—The message pipe created by the USB system software to pass control and status information between the host and a USB device's endpoint zero.

Device—A logical or physical entity that performs a function. The actual entity described depends on the context of the reference. At the lowest level, *device* may refer to a single hardware component, as in a memory device. At a higher level, it may refer to a collection of hardware components that perform a particular function, such as a USB interface device. At an even higher level, device may refer to the function performed by an entity attached to the USB; for example, a data/FAX modem device. Devices may be physical, electrical, addressable, and logical. When used as a non-specific reference, a USB device is either a hub or a function.

Device Address—A seven-bit value representing the address of a device on the USB. The device address is the default address (00H) when the USB device is first powered or the device is reset. Devices are assigned a unique device address by the USB system software.

Device Endpoint—A uniquely addressable portion of a USB device that is the source or sink of information in a communication flow between the host and device. *See also* Endpoint Address.

Device Resources—Resources provided by USB devices, such as buffer space and endpoints. *See also* Host Resources and Universal Serial Bus Resources.

Device Software—Software that is responsible for using a USB device. This software may or may not also be responsible for configuring the device for use.

DMA—Direct Memory Access.

Downstream—The direction of data flow from the host or away from the host. A downstream port is the port on a hub electrically farthest from the host that generates downstream data traffic from the hub. Downstream ports receive upstream data traffic.

DQPSK—Differential Quadrature Phase Shift Keying. A modulation technique used in TDMA.

Driver—When referring to hardware, an I/O pad that drives an external load. When referring to software, a program responsible for interfacing to a hardware device, that is, a device driver.

DSP—Digital Signal Processing.

DSTN—Double-layer Supertwist Nematic. A passive LCD panel that uses two display layers to counteract the color shifting that occurs with conventional supertwist displays. *See* STN.

Dual Band Mobile Phone—A phone that supports both analog and digital technologies by picking up analog signals when digital signals fade. Most mobile phones are not dual-band.

DWORD—Double word. A data element that is two words (four bytes or 32 bits) in size.

Dynamic Insertion and Removal—The ability to attach and remove devices while the host is in operation.

E2PROM—See Electrically Erasable Programmable Read-Only Memory (EEPROM).

EAV—End of Active Video.

EDGE—Enhanced Data GSM Environment. A faster version of the GSM standard. It is faster because it can carry messages using broadband networks that employ more bandwidth than standard GSM networks.

EEPROM—*See* Electrically Erasable Programmable Read-Only Memory (EEPROM).

Electrically Erasable Programmable Read-Only Memory (EEPROM)—Non-volatile re-writable memory storage technology.

End User—The user of a host.

Endpoint—*See* Device Endpoint.

Endpoint Address—The combination of an endpoint number and an endpoint direction on a USB device. Each endpoint address supports data transfer in one direction.

Endpoint Direction—The direction of data transfer on the USB. The direction can be either IN or OUT. IN refers to transfers to the host; OUT refers to transfers from the host.

Endpoint Number—A four-bit value between 0H and FH, inclusive, associated with an endpoint on a USB device.

Envelope Detector—An electronic circuit inside a USB device that monitors the USB data lines and detects certain voltage related signal characteristics.

EOF—End-of-(micro)Frame.

EOP—End-of-Packet.

EOTD—Enhanced Observed Time Difference.

ETM—Embedded Trace Macrocell. The ARM* real-time trace capability.

External Port—*See* Port.

Eye Pattern—A representation of USB signaling that provides minimum and maximum voltage levels as well as signal jitter.

False EOP—A spurious, usually noise-induced event that is interpreted by a packet receiver as an EOP.

FAR—Fault Address Register. Part of the ARM* architecture.

FDD—The Mobile Station transmits on one frequency; the Base Station transmits on another frequency.

FDM—Frequency Division Multiplexing. Each mobile station transmits on a different frequency (within a cell).

FDMA—Frequency Division Multiple Access. An analog standard that lets multiple users access a group of radio frequency bands and eliminates interference of message traffic.

FHSS—*See* Frequency Hopping Spread Spectrum.

FIQ—Fast Interrupt Request. *See also* IMMU.

Flash Memory —A type of erasable, rewritable memory that holds its content when power is off. The low-cost, low-power and high-density memory chip, with high-speed architecture, is highly reliable.

Frame—A 1-millisecond time base established on full-/low-speed buses.

Frame Pattern—A sequence of frames that exhibit a repeating pattern in the number of samples transmitted per frame. For a 44.1 kHz audio transfer, the frame pattern could be nine frames containing 44 samples followed by one frame containing 45 samples.

Frequency Hopping Spread Spectrum—A method by which a carrier spreads out packets of information (voice or data) over different frequencies. For example, a phone call is carried on several different frequencies so that when one frequency is lost another picks up the call without breaking the connection.

Fs—*See* Sample Rate (Fs).

FSR—Fault Status Register. Part of the ARM* architecture.

Full-Duplex—Computer data transmission occurring in both directions simultaneously.

Full-Speed—USB operation at 12 Mb/s. *See also* Low-Speed and High-Speed.

Function—A USB device that provides a capability to the host, such as an ISDN connection, a digital microphone, or speakers.

GMSK—Gaussian Minimum Shift Keying. A modulation scheme used in GSM.

GPRS—General Packet Radio Service A technology that sends packets of data across a wireless network at speeds up to 114 Kbps. Unlike circuit-switched networks, wireless users do not have to dial in to networks to download information; GPRS wireless devices are “always on” in that they can send and receive data without dial-ins. GPRS works with GSM.

GPS—Global Positioning System.

GSM—Global System for Mobile Communication. A standard for how data is coded and transferred through the wireless spectrum. The European wireless standard, also used in parts of Asia, GSM is an alternative to CDMA. GSM digitizes and compresses data and sends it across a channel with two other streams of user data. GSM is based on TDMA technology.

Hamming Distance—The distance (number of bits) between encoded values that can change without causing a decode into the wrong value.

Handshake Packet—A packet that acknowledges or rejects a specific condition. For examples, see ACK and NAK.

HDML—Handheld Device Markup Language. HDML uses hypertext transfer protocol (HTTP) to display text versions of web pages on wireless devices. Unlike WML, HDML is not based on XML. HDML does not allow scripts, while WML uses a variant of JavaScript. Web site developers using HDML must re-code their web pages in HDML to be viewed on the smaller screen sizes of handheld devices.

HARP—Hardware Adaptation Reference Platform. Microsoft® Windows CE standard development platform specification.

High-Bandwidth Endpoint—A high-speed device endpoint that transfers more than 1024 bytes and less than 3073 bytes per microframe.

High-Speed—USB operation at 480 Mb/s. *See also* Low-Speed and Full-Speed.

Host—The host computer system where the USB host controller is installed. This includes the host hardware platform (CPU, bus, and so forth.) and the operating system in use.

Host Controller—The host’s USB interface.

Host Controller Driver (HCD)—The USB software layer that abstracts the host controller hardware. The host controller driver provides an SPI for interaction with a host controller. The host controller driver hides the specifics of the host controller hardware implementation.

Host Resources—Resources provided by the host, such as buffer space and interrupts. *See also* Device Resources and Universal Serial Bus Resources.

HSTL—High-Speed Transceiver Logic.

Hub—A USB device that provides additional connections to the USB.

Hub Tier—One plus the number of USB links in a communication path between the host and a function.

I/O Request Packet—An identifiable request by a software client to move data between itself (on the host) and an endpoint of a device in an appropriate direction.

IBIS—I/O Buffer Information Specification is a behavioral description of the I/O buffers and package characteristics of a semiconductor device. IBIS models use a standard format to make it easier to import data into circuit simulation software packages.

I-cache—Instruction cache.

iDEN—Integrated Digital Enhanced Network. A technology that allows users to access phone calls, two-way radio transmissions, paging and data transmissions from one wireless device. iDEN was developed by Motorola and based on TDMA.

IMMU—Instruction Memory Management Unit. Part of the Intel XScale® core.

I-Mode—A Japanese wireless service for transferring packet-based data to handheld devices created by NTT DoCoMo. I-Mode is based on a compact version of HTML and does not currently use WAP.

Interrupt Request (IRQ) —A hardware signal that allows a device to request attention from a host. The host typically invokes an interrupt service routine to handle the condition that caused the request.

Interrupt Transfer—One of the four USB transfer types. Interrupt transfer characteristics are small data, non-periodic, low-frequency, and bounded-latency. Interrupt transfers typically handle service needs. *See also* Transfer Type.

IrDA—Infrared Development Association.

IRP—*See* I/O Request Packet.

IRQ—*See* IMMU.

ISI—Inter-Signal Interference. Data ghosting caused when multi-path delay causes previous symbols to interfere with the one currently being processed.

ISM—Industrial, Scientific, and Medical band. Part of the wireless spectrum that is less regulated, such as 802.11.

Isochronous Data—A stream of data whose timing is implied by its delivery rate.

Isochronous Device—An entity with isochronous endpoints, as defined in the USB Specification, that sources or sinks sampled analog streams or synchronous data streams.

Isochronous Sink Endpoint—An endpoint that is capable of consuming an isochronous data stream that is sent by the host.

Isochronous Source Endpoint—An endpoint that is capable of producing an isochronous data stream and sending it to the host.

Isochronous Transfer—One of the four USB transfer types. Isochronous transfers are used when working with isochronous data. Isochronous transfers provide periodic, continuous communication between host and device. *See also* Transfer Type.

Jitter—A tendency toward lack of synchronization caused by mechanical or electrical changes. More specifically, the phase shift of digital pulses over a transmission medium.

kb/s—Transmission rate expressed in kilobits per second. A measurement of bandwidth in the U.S.

kB/s—Transmission rate expressed in kilobytes per second.

Little Endian —Method of storing data that places the least significant byte of multiple-byte values at lower storage addresses. For example, a 16-bit integer stored in little endian format places the least significant byte at the lower address and the most significant byte at the next address.

LOA—Loss of bus activity characterized by an SOP without a corresponding EOP.

Low-Speed—USB operation at 1.5 Mb/s. *See also* Full-Speed and High-Speed.

LSb—Least Significant Bit.

LSB—Least Significant Byte.

LVDS—Low-Voltage Differential Signal.

MAC—Multiply Accumulate unit.

Mb/s—Transmission rate expressed in megabits per second.

MB/s—Transmission rate expressed in megabytes per second.

MC—Media Center. A combination digital set-top box, video and music jukebox, personal video recorder and an Internet gateway and firewall that hooks up to a broadband connection.

Message Pipe—A bidirectional pipe that transfers data using a request/data/status paradigm. The data has an imposed structure that allows requests to be reliably identified and communicated.

Microframe—A 125-microsecond time base established on high-speed buses.

MMC—MultiMediaCard. Small form factor memory and I/O card.

MMX Technology—The Intel® MMX™ technology comprises a set of instructions that are designed to greatly enhance the performance of advanced media and communications applications. See Chapter 10 of the *Intel Architecture Software Developers Manual, Volume 3: System Programming Guid*e, Order #245472.

Mobile Station—Cellular telephone handset

M-PSK—Multilevel Phase Shift Keying. A convention for encoding digital data in which there are multiple states.

MMU—Memory Management Unit. Part of the Intel XScale®core.

MSb—Most Significant Bit.

MSB—Most Significant Byte.

MSL—Mobile Scalable Link.

NAK—Handshake packet indicating a negative acknowledgment.

Non Return to Zero Invert (NRZI)—A method of encoding serial data in which ones and zeroes are represented by opposite and alternating high and low voltages where there is no return to zero (reference) voltage between encoded bits. Eliminates the need for clock pulses.

NTSC—National Television Standards Committee. The analog transmission standard used for televisions in the United States, Canada, Japan, and other countries. Delivers 525 lines of resolution at 60 half-frames per second, whereas PAL, used widely in the rest of the world, delivers 625 lines at 50 half-frames per second.

Object—Host software or data structure representing a USB entity.

Orthogonal Frequency Division Multiplexing (OFDM)—A special form of multi-carrier modulation. In a multi-path channel, most conventional modulation techniques are sensitive to inter-symbol interference unless the channel symbol rate is small compared to the delay spread of the channel. OFDM is significantly less sensitive to inter-symbol interference, because a special set of signals builds the composite transmitted signal. The basic idea is that each bit occupies a frequency-time window that ensures little or no distortion of the waveform. In practice, it means that bits are transmitted in parallel over a number of frequency-nonselective channels.

OTG—*See* USB OTG.

Packet—A bundle of data organized in a group for transmission. Packets typically contain three elements: control information (for example, source, destination, and length), the data to be transferred, and error detection and correction bits. Packet data is the basis for packet-switched networks, which eliminate the need to dial-in to send or receive information, because they are “always on.”

Packet Buffer—The logical buffer used by a USB device for sending or receiving a single packet. This determines the maximum packet size the device can send or receive.

Packet ID (PID)—A field in a USB packet that indicates the type of packet, and by inference, the format of the packet and the type of error detection applied to the packet.

Packet Switched Network—Networks that transfer packets of data.

PAL—Phase Alternating Line. Video standard for composite color encoding used in Europe, Australia, parts of Africa and the Middle East. PAL delivers 625 lines at 50 half-frames per second, whereas NTSC, the television standard used in the United States, delivers 525 lines of resolution at 60 half-frames per second.

PCMCIA—Personal Computer Memory Card Interface Association (PC Card).

PCS—Personal Communication Services. An alternative to cellular, PCS works like cellular technology because it sends calls from transmitter to transmitter as a caller moves. But PCS uses its own network, not a cellular network, and offers fewer “blind spots” than cellular, where calls are not available. PCS transmitters are generally closer together than their cellular counterparts.

PDA—Personal Digital Assistant. A mobile handheld device that gives users access to text-based information. Users can synchronize their PDAs with a PC or network; some models support wireless communication to retrieve and send e-mail and get information from the Internet.

Phase—A token, data, or handshake packet. A transaction has three phases.

Phase-Locked Loop (PLL)—A circuit that acts as a phase detector to keep an oscillator in phase with an incoming frequency.

Physical Device—A device that has a physical implementation; for example, speakers, microphones, and CD players.

PID—*See* Packet ID (PID) or Process ID.

PIO—Programmed Input/Output.

Pipe—A logical abstraction representing the association between an endpoint on a device and software on the host. A pipe has several attributes; for example, a pipe may transfer data as streams (stream pipe) or messages (message pipe). *See also* Stream Pipe and Message Pipe.

PLL—*See* Phase-Locked Loop (PLL).

PM—Phase Modulation.

PMIC—Power Management Integrated Circuit. A highly integrated device with both required and optional features to support the nine power domains on the processor, as well as dynamic voltage management features.

Polling—Asking multiple devices, one at a time, if they have any data to transmit.

POR—*See* Power-On Reset (POR).

Port—Point of access to or from a system or circuit. For the USB, the point where a USB device is attached.

Power-On Reset (POR)—Restoring a storage device, register, or memory to a predetermined state when power is applied.

Process ID—Process Identifier.

Programmable Data Rate—Either a fixed data rate (single-frequency endpoints), a limited number of data rates (32 kHz, 44.1 kHz, 48 kHz, …), or a continuously programmable data rate. The exact programming capabilities of an endpoint must be reported in the appropriate class-specific endpoint descriptors.

Protocol—A specific set of rules, procedures, or conventions relating to format and timing of data transmission between two devices.

PSP—Programmable Serial Protocol.

PWM—Pulse Width Modulator.

QAM—Quadrature Amplitude Modulation. A coding scheme for digital data.

QBS—Qualification By Similarity. A technique allowed by JEDEC for part qualification when target parameters are fully understood and data exist to warrant omitting a specific test.

QPSK—Quadrature Phase Shift Keying. A convention for encoding digital data into a signal using phase-modulated communication.

RA—*See* Rate Adaptation.

Radio Frequency Device—These devices use radio frequencies to transmit data. One typical use is for bar code scanning of products in a warehouse or distribution center, and sending that information to an ERP database.

Rate Adaptation—The process by which an incoming data stream, sampled at Fs i, is converted to an outgoing data stream, sampled at Fs o, with a certain loss of quality, determined by the rate adaptation algorithm. Error control mechanisms are required for the process. Fs i and Fs o can be different and asynchronous. Fs i is the input data rate of the RA; Fs o is the output data rate of the RA.

Request—A request made to a USB device contained within the data portion of a SETUP packet.

Retire—The action of completing service for a transfer and notifying the appropriate software client of the completion.

RGBT—Red, Green, Blue, Transparency color model.

ROM—Read-Only Memory.

Root Hub—A USB hub directly attached to the host controller. This hub (tier 1) is attached to the host.

Root Port—The downstream port on a root hub.

RTC—Real-Time Clock.

Intel® StrongARM* SA-1110—StrongARM* based applications processor for handheld products.

Intel® StrongARM* SA-1111—Companion chip for the Intel® SA-1110 processor.

SAD—Sum of absolute differences.

Sample—The smallest unit of data on which an endpoint operates; a property of an endpoint.

Sample Rate (Fs)—The number of samples per second, expressed in Hertz (Hz).

Sample Rate Conversion (SRC)—A dedicated implementation of the RA process for use on sampled analog data streams. The error control mechanism is replaced by interpolating techniques. Service A procedure provided by a System Programming Interface (SPI).

Satellite Phone—Phones that connect callers by satellite. Users have a world-wide alternative to terrestrial connections. Typical use is for isolated users, such as crews of deep-see oil rigs with phones configured to connect to a satellite service.

SAV—Start of Active Video.

SAW—Surface Acoustic Wave filter.

SD—Secure Digital. A non-volatile and very small memory card with high-storage capacity.

SDIO—*See* Secure Digital I/O (SDIO).

SDRAM—Synchronous Dynamic Random Access Memory.

Secure Digital I/O (SDIO)—Protocol provides high-speed data I/O with low-power consumption. The card supports multiple I/O functions, interrupts, and read/write operations.

SE0 (Single-Ended Zero)—The USB uses differential signals (D+ and D-) to transmit and receive data. The SE0 can be set to control the multiplexors.

Service Interval—The period between consecutive requests to a USB endpoint to send or receive data.

Service Jitter—The deviation of service delivery from its scheduled delivery time.

Service Rate—The number of services to a given endpoint per unit time.

SIMD—Single Instruction Multiple Data. A parallel processing architecture.

Smart Phone—A combination of a mobile phone and a PDA, which allow users to communicate as well as perform tasks; such as, accessing the Internet and storing contacts in a database. Smart phones have a PDA-like screen.

SMROM—Synchronous Mask ROM.

SMS—Short Messaging Service. A service through which users can send text-based messages from one device to another. The message can be up to 160 characters and appears on the screen of the receiving device. SMS works with GSM networks.

SOC—System On Chip.

SOF—*See* Start-of-Frame (SOF).

SOP—Start-of-Packet.

SPI—*See* System Programming Interface (SPI).

SPI—Serial Peripheral Interface. Also see the Serial Peripheral Interface protocol.

Split Transaction—A transaction type supported by host controllers and hubs. This transaction type allows full- and low-speed devices to be attached to hubs operating at high-speed.

Spread Spectrum—An encoding technique patented by actress Hedy Lamarr and composer George Antheil, which broadcasts a signal over a range of frequencies.

SRAM—Static Random Access Memory.

SRC—*See* Sample Rate Conversion (SRC).

SSE—Streaming SIMD Extensions.

SSE2—Streaming SIMD Extensions 2. For Intel Architecture machines, 144 new instructions, a 128-bit SIMD integer arithmetic, and 128-bit SIMD double precision floating point instructions, enabling enhanced multimedia experiences.

SSP—Synchronous Serial Port.

SSTL—Stub Series Terminated Logic.

Stage—One part of the sequence composing a control transfer; stages include the Setup stage, the Data stage, and the Status stage.

Start-of-Frame (SOF)—The first transaction in each (micro)frame. An SOF allows endpoints to identify the start of the (micro)frame and synchronize internal endpoint clocks to the host.

STN—Supertwist Nematic. A passive LCD display using the technique of twisting light rays to improve the quality of the LCD screens. Passive LCD displays apply either full-on voltage or full-off voltage (on or off) during each frame refresh. By intelligently turning the pixel on and off each frame a partial intensity is affected on each pixel. This process is known as dithering.

Stream Pipe—A pipe that transfers data as a stream of samples with no defined USB structure.

SWI—Software Interrupt.

Synchronization Type—A classification that characterizes an isochronous endpoint's capability to connect to other isochronous endpoints.

Synchronous RA—The incoming data rate, Fs i, and the outgoing data rate, Fs o, of the RA process are derived from the same master clock. There is a fixed relation between Fs i and Fs o.

Synchronous SRC—The incoming sample rate, Fs i, and outgoing sample rate, Fs o, of the SRC process are derived from the same master clock. There is a fixed relation between Fs i and Fs o.

System Programming Interface (SPI)—A defined interface to services provided by system software.

TAP —Test Access Port. The boundary-scan interface provides a means of driving and sampling the external pins of the processor: testing the processor's electrical connections to the circuit board and the integrity of the circuit board connections between devices. The interface is controlled through five dedicated TAP pins.

TC—Temperature Cycling.

TDD (Time Division Duplexing) —The Mobile Station and the Base Station transmit on same frequency at different times.

TDM—*See* Time Division Multiplexing (TDM).

TDMA—Time Division Multiple Access. TDMA protocol allows multiple users to access a single radio frequency by allocating time slots for use to multiple voice or data calls. TDMA breaks down data transmissions, such as a phone conversation, into fragments and transmits each fragment in a short burst, assigning each fragment a time slot. With a cell phone, the caller would not detect this fragmentation. TDMA works with GSM and digital cellular services.

TDR—*See* Time Domain Reflectometer (TDR).

Termination—Passive components attached at the end of cables to prevent signals from being reflected or echoed.

TFT —Thin Film Transistor. An active LCD panel in which each pixel is controlled by individual transistors that allows the voltage applied to each pixel to be precisely controlled. This permits faster response time and greater contrast compared to passive LCD panels.

Three-State—A high-impedance state in which the output is floating and is electrically isolated from the buffer's circuitry.

Time Division Multiplexing (TDM)—A method of transmitting multiple signals (data, voice, and/or video) simultaneously over one communication medium by interleaving a piece of each signal one after another.

Time Domain Reflectometer (TDR)—An instrument capable of measuring impedance characteristics of the USB signal lines.

Time-Out—The detection of a lack of bus activity for some predetermined interval.

Token Packet—A type of packet that identifies what transaction is to be performed on the bus.

TPV—Third-Party Vendor.

Transaction—The delivery of service to an endpoint; consists of a token packet, optional data packet, and optional handshake packet. Specific packets are allowed/required based on the transaction type.

Transaction Translator—A functional component of a USB hub. The transaction translator responds to special high-speed transactions and translates them to full/low-speed transactions with full/low-speed devices attached on downstream facing ports.

Transfer—One or more bus transactions to move information between a software client and its function.

Transfer Type—Determines the characteristics of the data flow between a software client and its function. Four standard transfer types are defined: control, interrupt, bulk, and isochronous.

TS—Thermal Shock.

Turn-Around Time—The time a device needs to wait to begin transmitting a packet after a packet has been received to prevent collisions on the USB. This time is based on the length and propagation delay characteristics of the cable and the location of the transmitting device in relation to other devices on the USB.

UART—Universal Asynchronous Receiver/Transmitter serial port.

UICC—Universal Integrated Circuit Card, formerly SIM card. A small electronic device about the size of a credit card that contains an embedded 8-bit microprocessor. The card stores a mathematical algorithm that encrypts voice and data transmissions. The card also identifies the caller to the mobile network as being a legitimate caller.

UDC—Universal Serial Bus Device Controller. The UDC, which consists of peripheral bus interface, endpoint memory, endpoint control, and USB interface, provides interface options to a host of peripheral devices at full speed.

UHC—Universal Serial Bus Host Controller. The UHC in conjunction with the UHC driver serially transfers data between a shared-memory data structure and the USB controller.

Universal Serial Bus Driver (USBD)—The host resident software entity responsible for providing common services to clients that are manipulating one or more functions on one or more host controllers.

Universal Serial Bus Resources—Resources provided by the USB, such as bandwidth and power. *See also* Device Resources and Host Resources.

Universal Subscriber Identity Module (USIM)—The USIM controller is an interface for a GSM mobile handset that supports communication with smart cards.

Upstream—The direction of data flow towards the host. An upstream port is the port on a device electrically closest to the host that generates upstream data traffic from the hub. Upstream ports receive downstream data traffic.

USBD—*See* Universal Serial Bus Driver (USBD).

USB-IF—USB Implementers Forum, Inc. is a nonprofit corporation formed to facilitate the development of USB-compliant products and promote the technology.

USB OTG—The USB On-The-Go provides “dual-role peripheral” capability: it can act as either host or peripheral depending on how users connect the cable to its unique mini-AB receptacle. When the dual-role device is connected to the mini-A plug, it turns into a host. When the mini-B plug is connected instead, the device becomes a peripheral.

USIM—*See* Universal Subscriber Identity Module (USIM).

VBI—Vertical Blanking Interval. Also known as the “backporch”.

Virtual Device—A device that is represented by a software interface layer. An example of a virtual device is a hard disk with its associated device driver and client software that makes it able to reproduce an audio WAV file.

VLIO—Variable Latency Input/Output interface.

WAP—Wireless Application Protocol. WAP is a set of protocols that lets users of mobile phones and other digital wireless devices access Internet content, check voice mail and e-mail, receive text of faxes and conduct transactions. WAP works with multiple standards, including CDMA and GSM. Not all mobile devices support WAP.

W-CDMA—Wideband CDMA. A third generation wireless technology under development that allows for high-speed, high-quality data transmission. Derived from CDMA, W-CDMA digitizes and transmits wireless data over a broad range of frequencies. It requires more bandwidth than CDMA, but offers faster transmission because it optimizes the use of multiple wireless signals, instead of one, as does CDMA.

Wireless LAN—A wireless LAN uses radio frequency technology to transmit network messages through the air for relatively short distances, like across an office building or a college campus. A wireless LAN can serve as a replacement for, or an extension to, a traditional wired LAN.

Wireless Spectrum—A band of frequencies where wireless signals travel carrying voice and data information.

Word—A data element that is four bytes (32 bits) in size.

WML—Wireless Markup Language. A version of HDML based on XML. Wireless applications developers use WML to re-target content for wireless devices.

YUV—A method of characterizing video signals typically used in digital cameras and PAL television specifying luminance and chrominance.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y