Marvell® PXA3xx Processor Family

Vol.IV: Serial Controller Configuration Developers Manual

PXA30x Processor (88AP300, 88AP301, 88AP302, 88AP303)

PXA31x Processor (88AP310, 88AP311, 88AP312)

PXA320 Processor (88AP320, 88AP322)

Doc. No. MV-S301374-04, Revision 2.0
Version -

April 6, 2009 Released

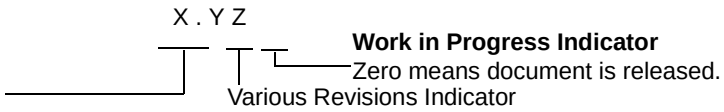
Document Conventions

	**Note:** Provides related information or information of special importance.
	**Caution:** Indicates potential damage to hardware or software, or loss of data.
	**Warning:** Indicates a risk of personal injury.

Document Status

Draft	For internal use. This document has not passed a complete technical review cycle and ECN signoff process.
Preliminary Tapeout (Advance)	This document contains design specifications for a product in its initial stage of design and development. A revision of this document or supplementary information may be published at a later date. Marvell may make changes to these specifications at any time without notice. Contact Marvell Field Application Engineers for more information.
Preliminary Information	This document contains preliminary specifications. A revision of this document or supplementary information may be published at a later date. Marvell may make changes to these specifications at any time without notice. . Contact Marvell Field Application Engineers for more information.
Complete Information	This document contains specifications for a product in its final qualification stages. Marvell may make changes to these specifications at any time without notice. Contact Marvell Field Application Engineers for more information.

Milestone Indicator:
Draft = 0.xx
Advance = 1.xx
Preliminary = 2.xx



Doc Status:	Technical Publication: 0.xx

For more information, visit our website at: www.marvell.com

Contents

Figures

Tables

1 Universal Serial Bus Device Controller

The universal serial bus (USB) device includes a USB device controller (UDC) with a number, type, and endpoints, interrupts to the processor, and Transmit/Receive FIFO interface. In this discussion of the USB, a working knowledge of the *Universal Serial Bus Specification* is assumed. The UDC is USB 1.1-compliant and supports all standard device requests issued by any certified USB host controller. For a full description of the USB protocol and its operation, refer not only to the specification but also to the following documents:

- *On-The-Go Supplement to Universal Serial Bus Specification,* Revision 2.0[1]
- *Pullup/Pulldown Resistors Engineering Change Notice* to the USB 2.0 Specification[2]

The UDC supports 24 endpoints (Endpoint 0 plus 23 programmable endpoints - refer to Section 1.2, "Features" for more details on programmable features). It is a full-speed device that operates half-duplex at a bit rate of 12 Mbps. The serial information transmitted and received by the UDC contains layers of communication protocols as defined in the USB specification, the most basic of which are fields.

1.1 PXA3xx Processor Differences

Table 1 shows the USB device controller differences among the PXA32x, PXA31x, and PXA30x processors. Refer to each individual register for other operating differences.

Table 1: PXA3xx Processors Feature Differences

Feature	PXA30x	PXA31x	PXA32x
USB Device Controller	Supported	Not Supported	Supported

1.2 Features

USB device features are as follows:

- USB Revision 1.1, full-speed compliant device (does not support low-speed operation)
- Supports 24 endpoints, including Endpoint 0 for control in and out and 23 programmable endpoints (A–X). Note that there is no Endpoint "O". Features of the programmable endpoints are as follows:
  - Endpoint type: bulk, isochronous, or interrupt
  - Endpoint direction: in or out
  - Endpoint maximum packet size
  - Programmable configuration, interface and alternate interface setting numbers
- 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
  - Endpoint 0 memory fixed at 32 bytes (16 for Receive and 16 for Transmit)

1. The latest revision of the *On-The-Go Supplement to Universal Serial Bus Specification Revision 2.0* can be accessed at the Internet site at: http://www.usb.org/
2. 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 the Internet site at: http://www.usb.org/

- Endpoint A - X memories configurable to be single-buffer or double-buffered

1.3 Signals

This section describes the UDC signals (see Table 2).

Table 2: USB Device Controller Interface Signals Summary

Name	Type	Description
USB Full Speed Transceiver Differential Signals - Port 2 Host/Device and OTG (PXA30x Processor only)		
USBOTG_P	Analog	**USB Full Speed Host/Device/OTG Port 2 D+** The positive signal of the differential pair that connects to the USB full-speed host, or the USB full-speed device, or the USB OTG interface for Port 2.
USBOTG_N	Analog	**USB Full Speed Host/Device/OTG Port 2 D-** The negative signal of the differential pair that connects to the USB full-speed host, or the USB full-speed device, or the USB OTG interface for Port 2.
USB Full Speed Single-Ended Signals - Port 2 Host/Device and OTG (Refer to Table 20 and Table 22 for more details on port configurations)		
USB_P2_1	Bidirectional	**USB Full Speed Host/Device and OTG Port 2 RCV/INT/SRP** • The Receive data from an external USB transceiver for port 2 • When configured for an external OTG transceiver, this signal is an interrupt input (UP2OCR = 0x4 or 0x5). • When configured for an external OTG power controller and the internal transceiver, this signal is the SRP detect input (UP2OCR = 0x6 or 0x7).
USB_P2_2	Bidirectional	**USB Full Speed Host/Device and OTG Port 2 OE/Valid** • Connects to the OE signal of an external USB transceiver for USB port 2. • This signal connects to the session valid output of an external OTG power controller.
USB_P2_3	Bidirectional	**USB Full Speed Host/Device and OTG Port 2 RXD-/SV** • The Receive negative data line from an external USB transceiver for port 2. • For an external OTG power controller, this signal is the session valid status input.
USB_P2_4	Bidirectional	**USB Full Speed Host/Device and OTG Port 2 TXD-/SE0_VM/SRP** • The Transmit negative data line to an external USB transceiver for port 2. • For an external OTG transceiver, this signal is the SE0_VM bidirectional data line. • For an external OTG power controller this signal is the SRP enable control output.
USB_P2_5	Bidirectional	**USB Full Speed Host/Device and OTG Port 2 RXD+/DAT_VP/VALID40** • The Receive positive data line from an external USB transceiver for port 2. • For an external OTG transceiver, this signal is the DAT_VP bidirectional data line. • For an external OTG power controller this signal is the 4.0 V Vbus valid status input.
USB_P2_6	Bidirectional	**USB Full Speed Host/Device and OTG Port 2 TXD+** The positive Transmit data line for an external USB transceiver for port 2.
USB_P2_7	Bidirectional	**USB Full Speed Host/Device and OTG Port 2 Speed/OTGID** Speed select signal for USB port 2 when configured for an external USB transceiver. Provides the OTG ID configuration using the internal transceiver.

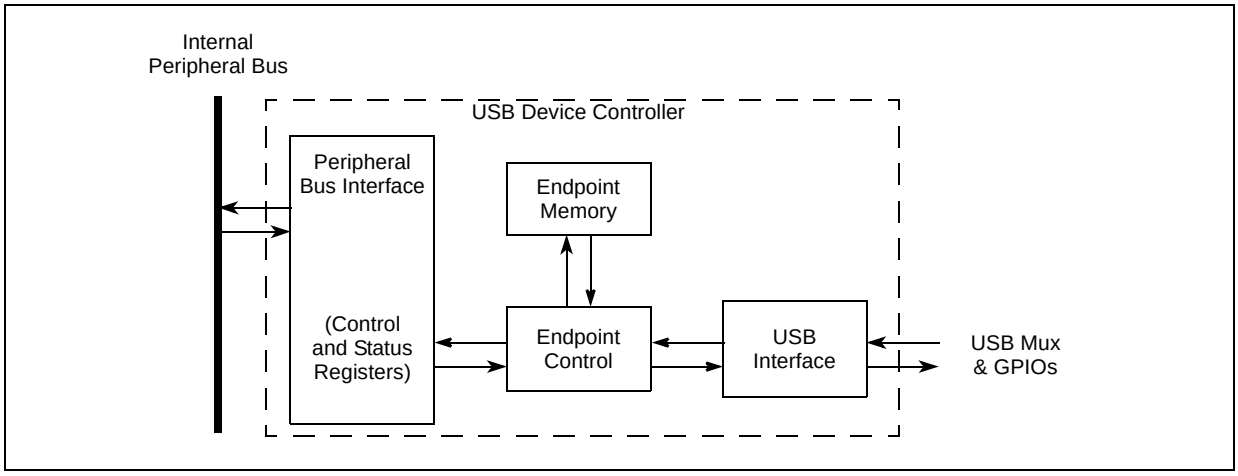
Table 2: USB Device Controller Interface Signals Summary (Continued)

Name	Type	Description
USB_P2_8	Input	**USB Full Speed Host/Device and OTG Port 2 Suspend, and Host Port 2 Power Control** • Suspend enable for USB port 2 when configured for an external USB OTG transceiver. • Controls power to the USB host port 2 (USBHPEN2).
NOTE: 1. Refer to the USB Port 2 Output Control Register (UP2OCR) for more information on configuring Host Port 2. 2. Refer to the USB Port 3 Output Control Register (UP3OCR) for more information on configuring Host Port 3.		

1.4 UDC Architecture

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 processor and the UDC. The endpoint memory is a 4-Kbyte SRAM for USB endpoint data storage. It has 32 bytes dedicated to Endpoint 0, and the remainder of its memory can be allocated to any of the 23 programmable endpoints. The endpoint control and USB interface blocks provide the USB functionality. Figure 1 is a block diagram of the UDC and UDC dedicated I/O.

Figure 1: UDC Block Diagram



1.4.1 Peripheral Bus Interface and Control/Status Registers

The UDC is a slave on the internal peripheral device connected to the internal peripheral bus. All software-initiated accesses to the UDC registers and endpoint memory are completed using the internal peripheral bus. The Control and Status registers provide 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 USB functions. It enables the UDC and indicates the configuration, interface, and alternate interface settings selected by the USB host controller on the USB. Either the processor interrupt mechanisms 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. The UDCCR contains 19 control

and status bits to determine UDC status and control global UDC functions. The UDCCR also contains a status bit to indicate whether 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 (see Section 1.5.1 for details on the UDCCR).

Although the UDC can generate only a single interrupt to the processor interrupt controller, there are 53 sources for its 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 an interrupt being generated. USB activity is 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 interrupt 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 to 1, 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 1.5.2 and Section 1.5.3 for 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:

- Contains 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 enumerates the UDC.

1.4.1.1 Endpoint 0

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 (UDCCSR0[RNE]) bit and Out-Packet-Complete (UDCCSR0[OPC]) bit are set when a complete data packet is received from the USB host controller. If the packet is part of a SETUP transaction, the setup active (UDCCSR0[SA]) bit is also set. The UDC flushes the Endpoint 0 Transmit FIFO after it receives an OUT data packet from the USB host controller. The Byte Count register (UDCBCR0) indicates the number of bytes of data 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 decrements to indicate the number of bytes remaining in the buffer. When all of the data is unloaded from the Receive buffer, the UDC clears UDCCSR0[RNE] to indicate that the Receive buffer is empty. After reading all of the data from the Endpoint 0 Receive buffer, software must clear UDCCSR0[OPC] so the buffer can receive another USB data packet. Software must also set UDCCSR0[IPR] to enable the UDC to ACK the status stage of the control transaction. Set UDCCSR0[IPR] for each packet transferred because the USB host controller can terminate a transaction with a premature status stage.

Loading a maximum packet size of 16 bytes into the Endpoint 0 Transmit FIFO automatically sets UDCCSR0[IPR]. UDCCSR0[IPR] must be set explicitly to indicate that a complete packet is loaded If fewer than 16 bytes are loaded into the Endpoint 0 Transmit FIFO. When the data is transmitted to the USB host controller, the UDC clears UDCCSR0[IPR] to indicate that the packet has been sent successfully (see Section 1.5.7, Section 1.5.9 and Section 1.5.10 for details on Endpoint 0 registers).

1.4.1.1.1 Back-to-Back Setup Packets

Logic is controlled by the OPC Disable FIFO Clear bit (UDCCSR0[ODFCLR] to prevent the inadvertent flushing of back-to-back Setup packets due to delays in clearing an OPC that was generated during a previous Setup status phase. The following describes the sequence when the bit is cleared (default reset value):

- Host sends a Setup packet (Get_descriptor) and it causes the following hardware changes to the UDCCSR0
  - UDCCSR0[OPC] = 1 - Out packet complete
  - UDCCSR0[FTF] = 1 for one system clock - Flush the Transmit FIFO
  - UDCCSR0[RNE] = 1 - Receive FIFO not empty
  - UDCCSR0[SA]= 1 - Setup Active
  - UDCBCR0 = 8 - Byte Count = 8 (size for a Get_descriptor packet)

1.4.2 If enabled, the UDC sends an interrupt

- Software processes the Get_descriptor packet
  - Software reads UDCBCR0 = 8
  - Software reads Receive FIFO data (UDCDR0) – UDCBCR0 decrements by 4 each time this register is read; the UDCCSR0[RNE] bit is cleared by hardware when UDCBCR0 decrements to 0
  - Software clears UDCCSR0[OPC] and UDCCSR0[SA] - The UDC NAKs any data packets if the UDCCSR0[OPC] is not cleared
- Software prepares the data to be sent to the host:
  - Software writes Transmit FIFO data (UDCDR0) with the appropriate data
  - Software sets UDCCSR0[IPR] = 1
- Host sends data packet request(s) (DATA_IN)
  - UDC transmits the data
  - UDCCSR0[IPR] is cleared by hardware
  - Multiple passes of setting UDCCSR0[IPR], data request, and data Transmit may be required depending on the amount of data that was requested by the host in the Setup packet
- If the data is received successfully by the host, the host sends a zero-length Status packet (STATUS_OUT)
  - UDCCSR0[OPC] = 1 - Out packet complete
  - UDCCSR0[FTF] = 1 for one system clock - Flush the Transmit FIFO
  - UDCBCR0 = 0 - Byte Count = 0 (size for a STATUS_OUT packet)
  - UDCCSR0[SA] = 0, UDCCSR0[RNE] = 0
  - If enabled, the UDC sends an interrupt

IF THE STATUS OPC IS CLEARED BEFORE NEXT SETUP PACKET

- Software processes the STATUS_OUT packet
  - Software reads UDCBCR0 = 0
  - Software clears OPC - The UDC cannot NAK any Setup packets

IF THE STATUS OPC IS CLEARED AFTER THE NEXT SETUP PACKET

- Software processes the STATUS_OUT packet
  - Software reads UDCBCR0 = 0

- Host sends a SETUP packet (that is, Get_descriptor) and it causes the following hardware changes to the UDCCSR0:
  - UDCCSR0[OPC] remains set to 1
  - UDCCSR0[FTF] = 1 for one system clock but the FIFO is not actually flushed
  - UDCCSR0[RNE] = 1
  - UDCCSR0[SA] = 1 - Setup Active
  - UDCBCR0 = 0 – Internal Byte Count = 8 but the actual Read value is 0
  - FIFO Overrun (UDCISR1[1]) is set if the interrupt is enabled
  - If enabled, the UDC sends an interrupt
- Software continues to processes the STATUS_OUT packet:
  - Software clears UDCCSR0[OPC]
  - Hardware:
  - -Clears UDCCSR0[OPC] for one system clock and then flops “shadow OPC” into UDCCSR0[OPC]
  - -Clears “shadow OPC”
  - -Unmasks the Read value of BC, indicating 8 bytes
- Software processes the Get_descriptor packet
  - Software reads UDCBCR0 = 8
  - Software reads Receive FIFO data (UDCDR0) – UDCBCR0 decrements by 4 each time this register is read; the UDCCSR0[RNE] bit is cleared by hardware when UDCBCR0 decrements to 0
  - Software clears UDCCSR0[OPC] and UDCCSR0[SA]

The following describes the sequence when the bit is written to a 1:

- Host sends a Setup packet (that is, Get_descriptor) and it causes the following hardware changes to the UDCCSR0:
  - UDCCSR0[OPC] = 1 - Out packet complete
  - UDCCSR0[FTF] = 1 for one system clock - Flush the Transmit FIFO
  - UDCCSR0[RNE] = 1 - Receive FIFO not empty
  - UDCCSR0[SA] = 1 - Setup Active
  - UDCBCR0 = 8 - Byte Count = 8 (size for a Get_descriptor packet)
  - If enabled, the UDC sends an interrupt
- Software processes the Get_descriptor packet:
  - Software reads UDCBCR0 = 8
  - Software reads Receive FIFO data (UDCDR0) – UDCBCR0 decrements by 4 each time this register is read; the UDCCSR0[RNE] bit is cleared by hardware when UDCBCR0 decrements to 0
  - Software clears UDCCSR0[OPC] and UDCCSR0[SA] - The UDC NAKs any data packets if the UDCCSR0[OPC] is not cleared
- Software prepares the data to be sent to the host
  - Software writes Transmit FIFO data (UDCDR0) with the appropriate data
  - Software sets UDCCSR0[IPR] = 1
- Host sends data packet request(s) (DATA_IN)
  - UDC transmits the data

- UDCCSR0[IPR] is cleared by hardware
- Multiple passes of setting UDCCSR0[IPR], data request, and data Transmit may be required depending on the amount of data that was requested by the Host in the Setup packet.

- If the data is received successfully by the host, the host sends a zero-length status packet (STATUS_OUT):
  - UDCCSR0[OPC](UDCCSR0[0] = 1 - Out packet complete
  - UDCCSR0[FTF] = 1 for one system clock - Flush the Transmit FIFO
  - UDCBCR0 = 0 - Byte Count = 0 (size for a STATUS_OUT packet)
  - UDCCSR0[SA] = 0, UDCCSR0[RNE] = 0
  - If enabled, the UDC sends an interrupt

STATUS UDCCSR0[OPC]CLEARED BEFORE NEXT SETUP PACKET

- Software processes the STATUS_OUT packet
  - Software reads UDCBCR0 = 0
  - Software clears UDCCSR0[OPC] - the UDC cannot NAK any Setup packets
- Host sends a Setup packet (Get_descriptor), which causes the following hardware changes to the UDCCSR0:
  - UDCCSR0[OPC] = 1 - Out packet complete
  - UDCCSR0[FTF] = 1 for one system clock - Flush the Transmit FIFO
  - UDCCSR0[RNE] = 1 - Receive FIFO not empty
  - UDCCSR0[SA] = 1 - Setup Active
  - UDCBCR0 = 8 - Byte Count = 8 (size for a Get_descriptor packet)
  - If enabled, the UDC sends an interrupt

STATUS UDCCSR0[OPC] NOT CLEARED BEFORE NEXT SETUP PACKET

- Software processes the STATUS_OUT packet
  - Software reads UDCBCR0 = 0
- Host sends a SETUP packet (that is, Get_descriptor), which causes the following hardware changes to the UDCCSR0:
  - UDCCSR0[OPC] = 1 - Out packet complete
  - UDCCSR0[FTF] = 1 for one system clock - Flush the Transmit FIFO
  - UDCCSR0[RNE] = 1 - Receive FIFO not empty
  - UDCCSR0[SA] = 1 - Setup Active
  - UDCBCR0 = 8 - Byte Count = 8 (size for a Get_descriptor packet)
  - FIFO Overrun (UDCISR1[1]) is set if the interrupt is enabled
- Software clears UDCCSR0[OPC]
  - UDCCSR0[SA] is cleared and FIFO is flushed
  - Causes UDCBCR0 and UDCCSR0[RNE] to clear
  - Second setup packet is missed

1.4.2.1 Endpoints A through X

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 1.4.3 for 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 is received from the USB host controller.

The Byte Count register of each endpoint indicates the number of data bytes 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 data is unloaded from the FIFO memory, the UDC clears the FS and Buffer-Not-Empty/ Buffer-Not-Full (BNE/BNF) bits in the Control/Status register and the Byte Count (BC) bit in the Byte Count register to indicate that 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.

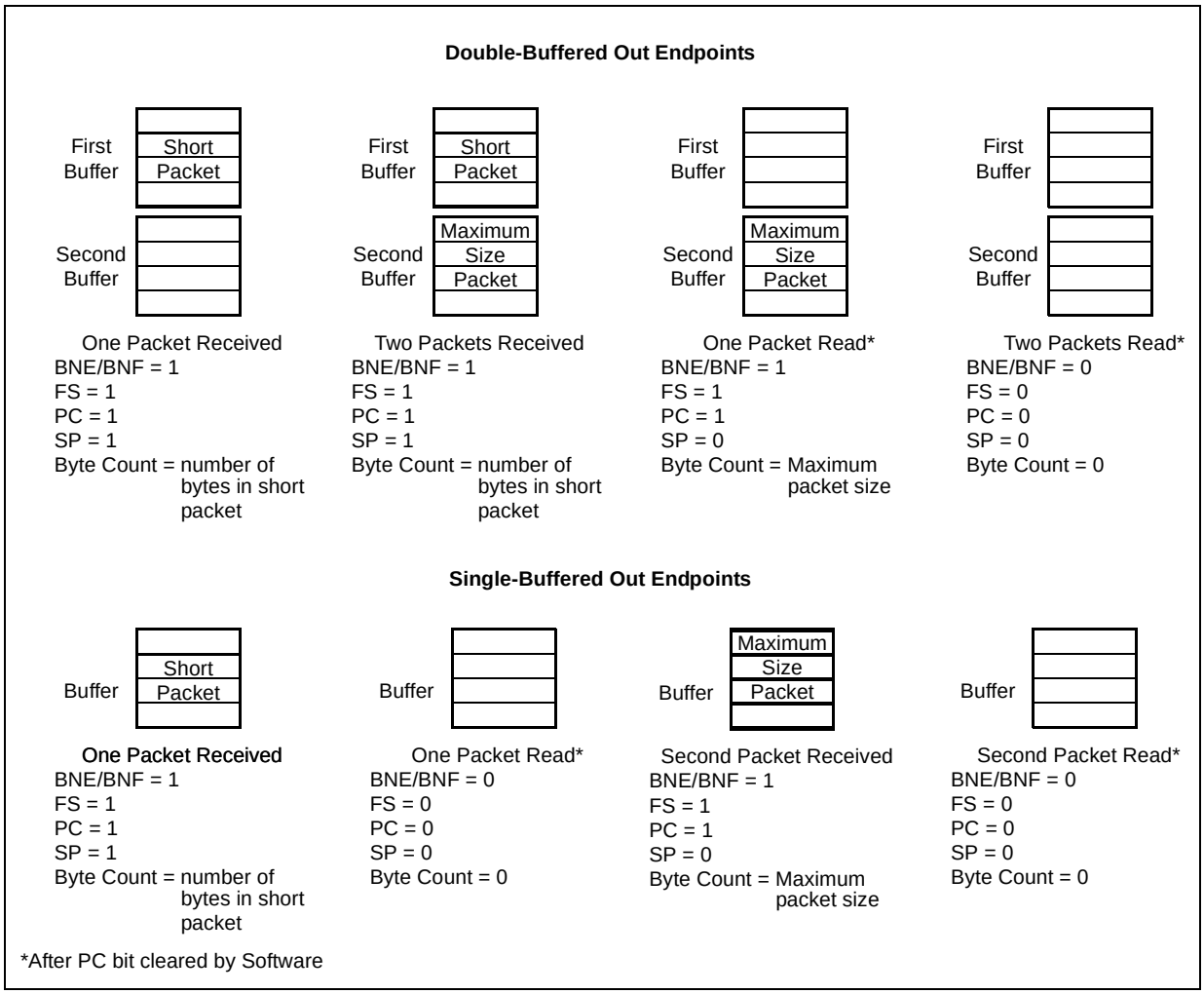
If an Out endpoint has double-buffering enabled, the Control/Status and Byte Count registers provide the status of the Endpoint buffer that is currently active. The FS, PC, and BNE/BNF bits in the Endpoint Control/Status register are set when the first buffer receives a complete data packet. The Data register can be used to unload 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 still to be read from the first buffer in the FIFO memory. The Byte Count register and the BNE/BNF bit in the Control/Status register are cleared when all data in the first buffer is read. If a second packet is received before all of the data is read from the first buffer, the FS bit continues to be set after the first buffer is read. However, the second packet of data does not set the PC bit until after software clears the PC bit.

The PC bit must be cleared to update the Control/Status and Byte Count registers with the status of the second buffer. The data in the buffer is lost if the PC bit is cleared before all the data in the first Receive buffer is read.

If the second buffer receives a complete data packet, the PC bit is again set to indicate that the Endpoint FIFO has data ready to be unloaded. The BNE/BNF and Byte Count registers indicate the amount of data in the second buffer. At this point, the Data register is used to unload 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 is read from the second buffer and software clears the PC bit can the PC bit be safely set again. Clearing the PC bit again updates the Control/Status and Byte Count registers to reflect the status of the first buffer (See Figure 2 for the relationship between the data in the Endpoint buffers and the status bits in the Endpoint Control/Status register).

Figure 2: Status Bits for Out Endpoints

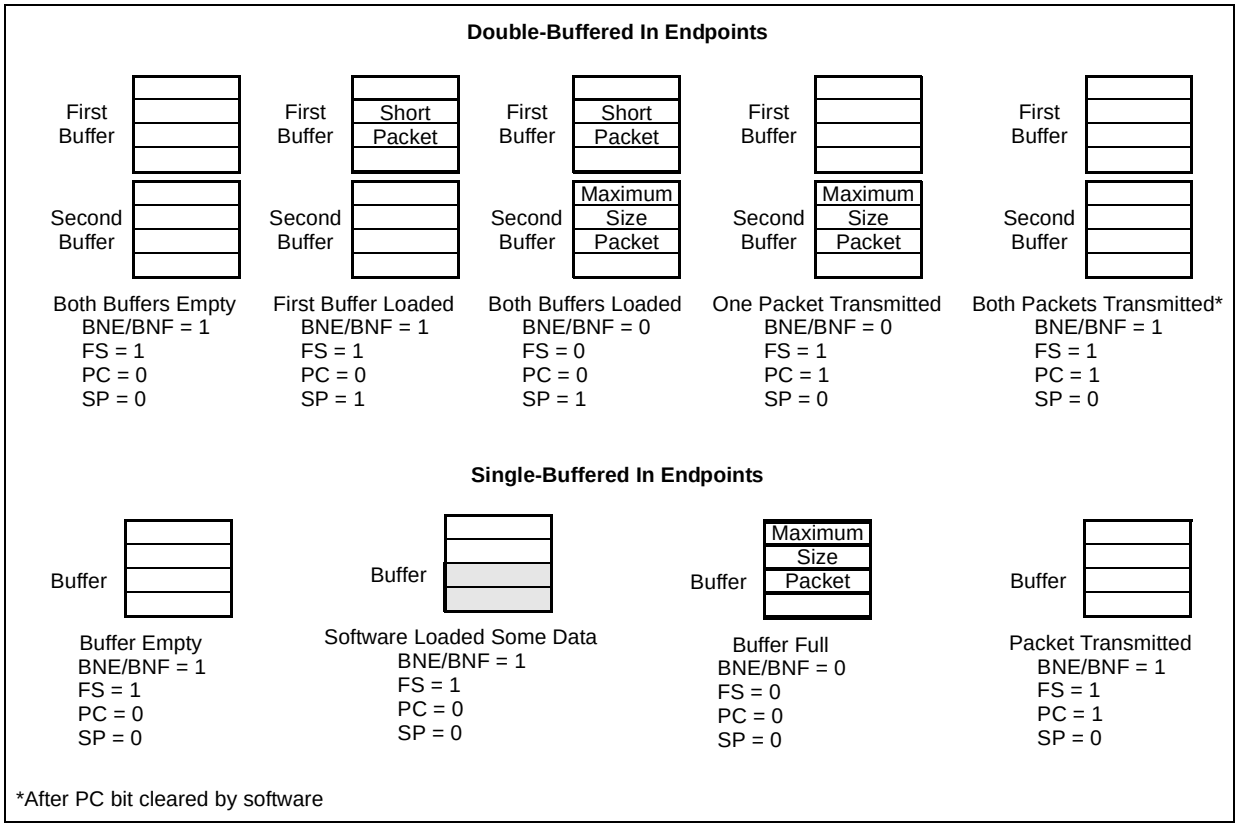


The Control/Status register provides the status of the Endpoint buffer if an endpoint is configured as an In endpoint and has double-buffering disabled. The Byte Count register is not used for In endpoints and is reserved. If data is loaded into the Endpoint buffer, but has not been transmitted to the USB host controller, the UDC clears the BNE/BNF and FS bits when a complete packet is loaded into the FIFO memory and is ready for transmission. Either write the maximum packet size to the buffer or set the Short-Packet (SP) bit to indicate that a complete packet is loaded. When the data is successfully transmitted to the USB host controller, the UDC sets the PC bit to indicate that 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 more data is loaded 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 requires service. Data is loaded into the first buffer by writing to the Endpoint Data register. The UDC clears the BNE/BNF bit when the current Endpoint buffer is not transmitted but has a complete packet loaded. The FS bit is cleared when both buffers are 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 is transmitted to the USB host controller, the PC and FS bits are set to indicate that 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 (see Figure 3 for the relationship between data in the endpoint buffers and the status bits in the Endpoint Control/Status register).

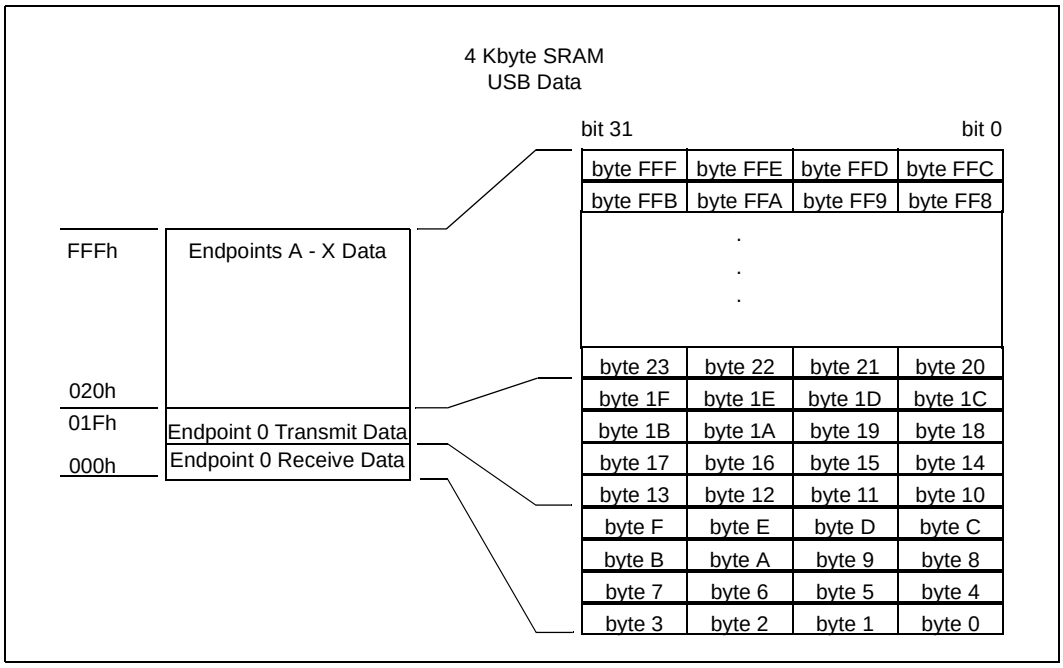
Figure 3: Status Bits for In Endpoints



1.4.3 Endpoint Memory Configuration

The Endpoint memory consists of 4 Kbytes of SRAM, arranged as a 1024 × 32-bit block and referred to as the Receive FIFO, Transmit FIFO, or FIFO memory throughout this document (Figure 4 shows the unconfigured memory map of the 4 Kbyte SRAM). The Endpoint memory is used to store USB data received from the USB host controller or loaded for transmission to the USB host controller. Thirty-two bytes of the endpoint memory are dedicated for Endpoint 0 USB data storage. Software can allocate the remaining 4064 bytes to Endpoints A–X. The FIFO memory space is flushed and the memory reallocated when software 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 FIFO SRAM is accessed through the Endpoint Data registers (UDCDR0 and UDCDRA–UDCDRX).

Figure 4: 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–UDCCRX (see Section 1.5.11 for details).

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 is not used and cannot be accessed. If this endpoint were programmed to be double buffered, it would use a total of 640 bytes of FIFO memory space (320 bytes $\times$ 2). In addition, any packet that is less than the programmed maximum packet size is considered to occupy 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 an endpoint with a packet that is not 32-bit aligned is loaded, software 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, software must write the first 12 bytes as three 32-bit accesses. The last three bytes can be written as either 3 byte writes, 1 byte Write and one half-word Write or as one 24-bit access (if it is loaded by the DMA controller). No other accesses to the Endpoint memory can be less than 32-bit accesses. All reads must be 32-bit accesses.

Table 3 shows the types of endpoints, the direction that can be programmed for each type, and the maximum packet sizes available. The Bulk, Isochronous, and Interrupt endpoints can be programmed as double buffered so that one packet can be processed while the next is assembled. If an In endpoint is double buffered, software can load the endpoint data into the second Transmit buffer while data from the first buffer is transmitted. Similarly, when an Out endpoint is unloaded, the

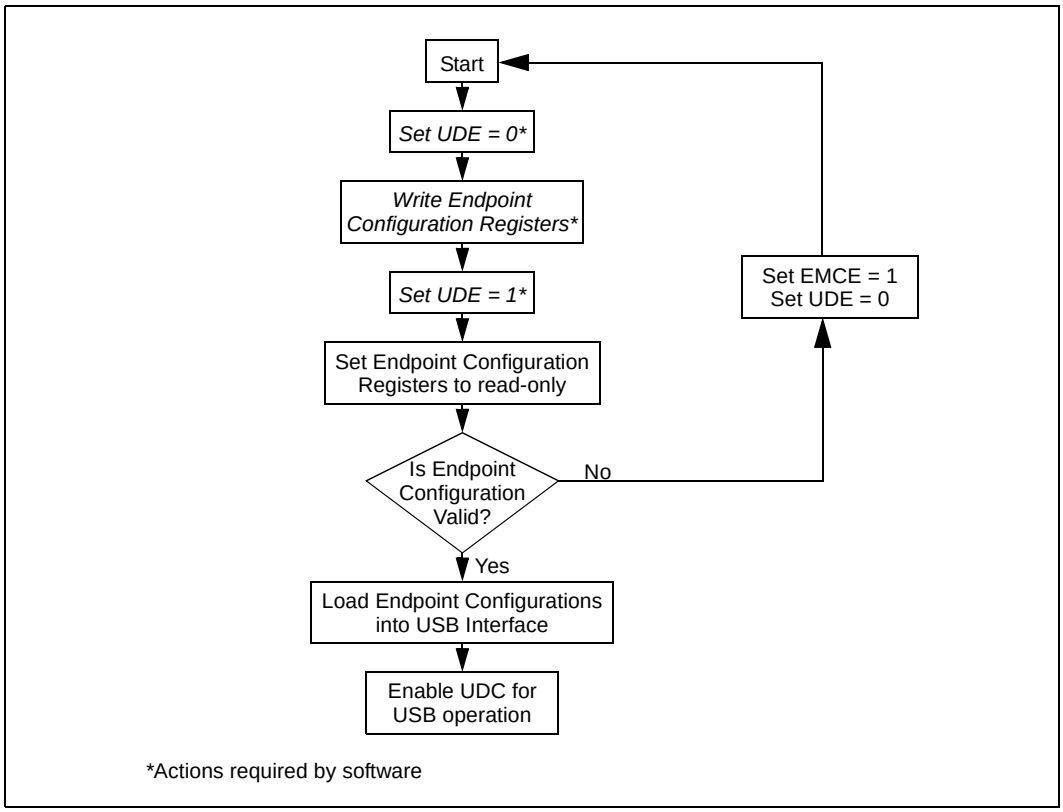
UDC can continue to process the next incoming packet to that endpoint. Endpoint 0 has 32 bytes of FIFO memory space, 16 bytes for IN data, and 16 bytes for OUT data, and it is not double buffered.

Table 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)	N/A
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

The total number of bytes of FIFO memory space allocated to all enabled endpoints for each configuration and interface cannot exceed 4064 bytes. The Endpoint configuration is checked and the error is detected if more than 4064 bytes of FIFO memory space is allocated to a single configuration and interface when UDCCR[UDE] is set. As a result, 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 the UDC clears UDCCR[UDE]. Figure 5 shows the sequence for allocating the FIFO memory, setting Endpoint configuration, and enabling the UDC for USB operation. Actions required by software are *italicized*.

Figure 5: FIFO Memory and Endpoint Configuration Sequence



If the DMA controller is used to load and unload 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 programmed I/O (PIO) method (processor transfers to/from memory directly) is used to load and unload the Endpoint memory, it must be accessed as a 32-bit read or 32-bit Write. When using the PIO method to load data for transmitting, four bytes must be loaded at a time unless the packet size is not 4-byte aligned; if so, the final word Write can contain less than four data bytes.

The memory space in the FIFO that is allocated to an endpoint (outside the space needed for the maximum packet size) is unused if an In endpoint maximum packet size is not 32-bit aligned. Similarly, four bytes at a time must be unloaded when the core is used to unload received (out) data. 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 (outside of the maximum packet size) produces unknown results and the Byte Count register decrements by the number of valid bytes read. For example, in Figure 6 FIFO memory space is unloaded by reading the Endpoint Data register and decrementing the Byte Count register for an Out endpoint with a maximum packet size of 13 bytes. In Figure 7, FIFO memory space is loaded by writing the Endpoint Data register for an In endpoint with a maximum packet size of 13 bytes.

Figure 6: Example Data Ordering and FIFO Organization for Out Endpoints

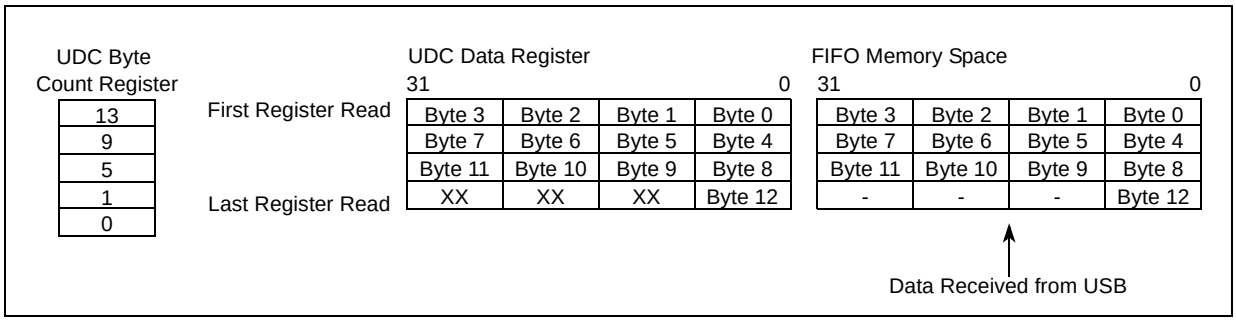
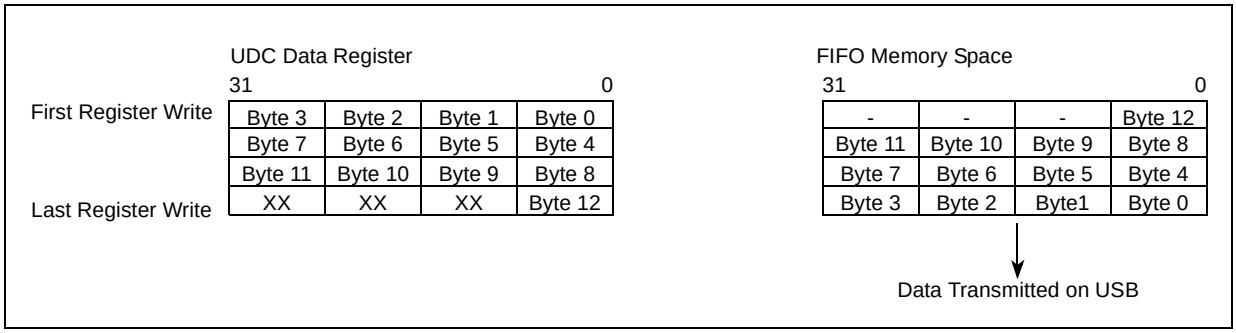


Figure 7: Example Data Ordering and FIFO Organization for In Endpoints



If the USB host controller requests IN data and no packet is loaded into the Endpoint FIFO memory, the Transmit/Receive NAK (TRN) bit in the Endpoint Control/Status register (UDCCSRx[TRN]) is set and one of the following actions is taken:

- The UDC issues a NAK handshake to the host controller if the endpoint is configured as a Bulk or Interrupt endpoint.
- The UDC sends a zero-size packet if the endpoint is configured as an Isochronous endpoint, and if enabled, the FIFO error interrupt for the affected endpoint is generated.

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 and one of the following actions is taken:

- The UDC sends a NAK handshake to the host if the endpoint is configured as a Bulk or Interrupt endpoint.
- The isochronous data packet is dropped if the endpoint is configured as an Isochronous endpoint, and the FIFO Error interrupt for the affected endpoint is generated, if enabled.

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].

1.4.3.1 DMA Access to Endpoint Memory

The DMA controller (DMAC) can be used to load and unload the 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. The UDC can operate without causing interrupts when the UDC and DMA are set for DMA loading and unloading of Endpoint memory.

Note

Disable the DMA-Enable (DME) bit only when the DMA channel is stopped.

When an Out endpoint receives any packet, except a Zero-size packet, the UDC asserts a DMA request for that endpoint. The DMA controller receives the request and responds to it by unloading the data from the Endpoint buffer. Then, 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 is reached (see Figure 8 for a DMA Descriptor for Out endpoint FIFO servicing). If the DMAC stops reading data from the Endpoint buffer because it reaches the end of the Descriptor, more data can still reside in the Endpoint buffer. The processor must read any data remaining in the Endpoint buffer, or the data may 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 DMAC 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 DMAC receives the request and responds to the request by unloading the data from the Endpoint buffer. When all data in the Endpoint buffer is unloaded, the UDC generates an interrupt to the processor, indicating that the SP bit must be cleared. The processor must clear the PC bit so that the Endpoint buffer can receive more data. The SP bit is cleared automatically when PC is cleared.

If the SP bit in the Endpoint Control register is set, indicating that 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 must be cleared. The processor must clear the PC bit so that the Endpoint buffer can receive more data. The SP bit is cleared automatically when the PC is cleared.

Figure 8: 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 = software selected, Width =
3, Len = transfer length (may be multiple packets)
```

When an In endpoint with the DME bit set has an empty buffer, the UDC asserts a DMA request for that endpoint. The DMA controller receives the request and responds to it by loading data into the Endpoint FIFO. When the DMA controller responds to the request, the UDC negates the DMA request. The DMA controller must load either a Maximum size packet or 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 9 illustrates the use of DMA chained Descriptors to load multiple packets of USB data for transmission to the USB host controller. The first Descriptor is used to load a maximum size packet, and the second Descriptor is used to set the SP bit. For In endpoints, the Descriptor length must be set to the maximum packet size and multiple Descriptors used to send multiple packets except when the maximum packet size is a multiple of the DMA burst size. In this instance, the Descriptor length can be set to the entire transfer length. For details on setting the DMA channel registers and Descriptors, refer to the DMA controller chapter.

Figure 9: 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 = Internal or External memory
desc[0].dtadr = UDCDRx
desc[0].dcmds = IncSrcAddr = 1, FlowTrg = 1, Size = software 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
desc[0].dcmds = Width = 3, Len = 4, FlowSrc = 0, FlowTrg = 0
```

1.4.3.2 USB Configurations and Interfaces

Figure 10 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 USB host controller enumerates the UDC. The endpoint memory allocations for configurations 1, 2, and 3 are programmable.

Figure 10: Configurations, Interfaces and Alternate Interface Settings for UDC

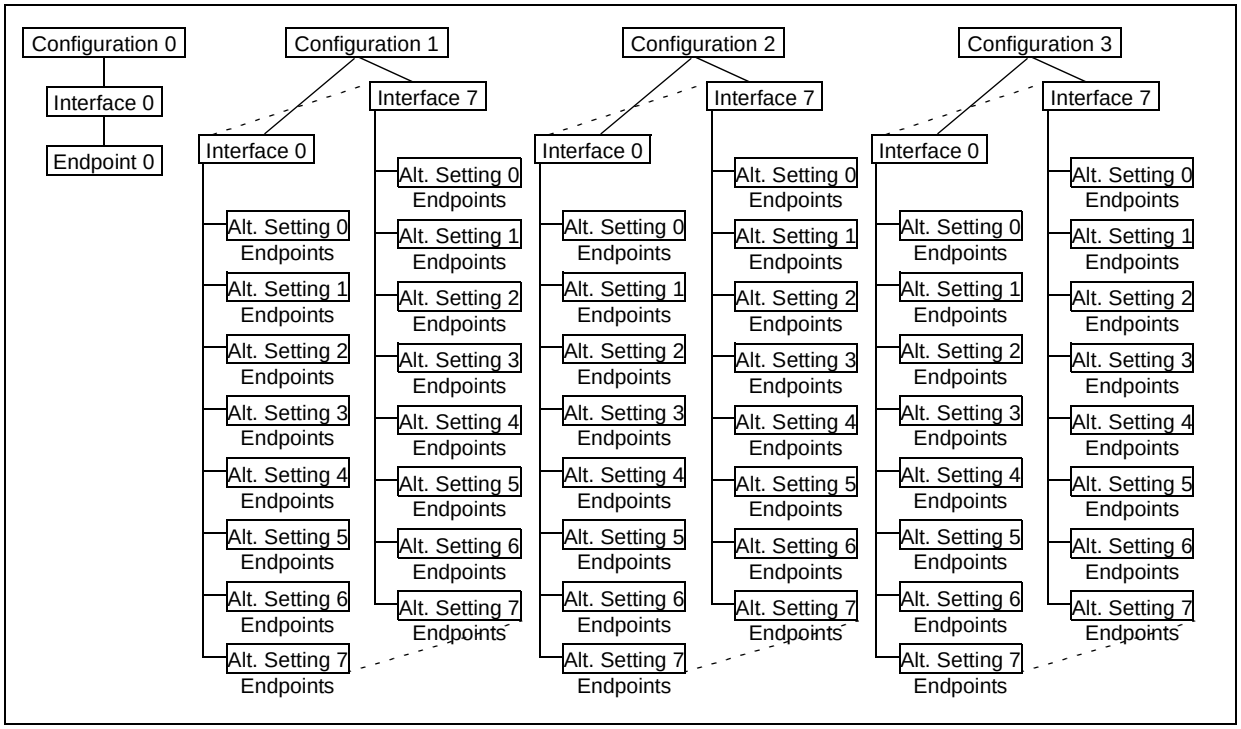
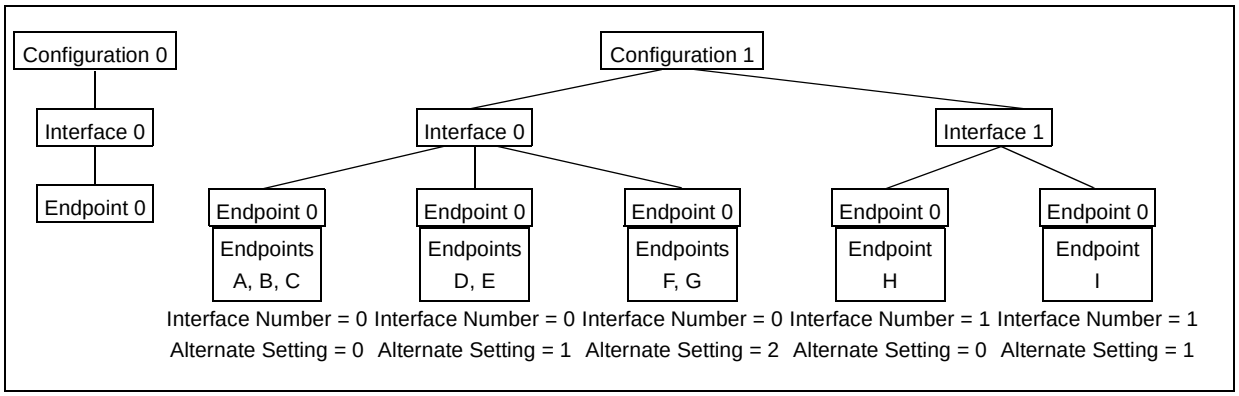


Figure 11 shows an example of two configurations (0 and 1). Configuration 1 has two interfaces and a total of five alternate interface settings that are each assigned a unique set of endpoints from endpoints A–X. Each programmable endpoint A–X can be assigned only one configuration,

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 11: Example of Two UDC Configurations



When the USB host controller executes a Set_configuration or Set_interface command and the Configuration-Change interrupt is enabled, an interrupt is generated. Any data residing 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 UDC clears UDCCR[SMAC] after the Endpoint memory reallocation completes. The UDC completes the status stage of the Set_configuration and Set_interface commands regardless of the UDCCR[SMAC] value.

Table 4 shows an example of the memory allocated for endpoints defined by a configuration, interface, and alternate interface setting.

This configuration has only six of the 23 programmable endpoints enabled, but uses 2968 bytes of the 4064 bytes available.

Table 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

Each configuration programmed for each interface must use the same amount of Endpoint memory across its alternate settings. Dummy endpoints must be programmed to equalize the endpoint memory storage, if necessary. The physical endpoints must be chosen in ascending order.

1.4.3.3 Example Endpoint Configuration

The programmable Endpoints A–X can be configured to respond as any USB endpoint number from 1 through 15; as any type: Isochronous, Interrupt or Bulk; and either direction: in or out. All programmable Endpoints A–X must be configured before the UDC is enabled 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.

Figure 12 shows an example configuration setting that can be used for the UDC. In this example, the UDC setup is defined as having two configurations: the default Configuration 0 and a software-programmed Configuration 1. Configuration 1 is defined to have two interfaces (0 and 1). Interface 0 has two alternate interface settings (0 and 1), and Interface 1 has three alternate interface settings (0, 1, and 2). Each alternate interface setting has a set of endpoints to implement that interface. The Endpoint Configuration registers are used to configure each UDC Endpoint A–X for its use within the configuration.

Figure 12: Example USB Configurations for UDC

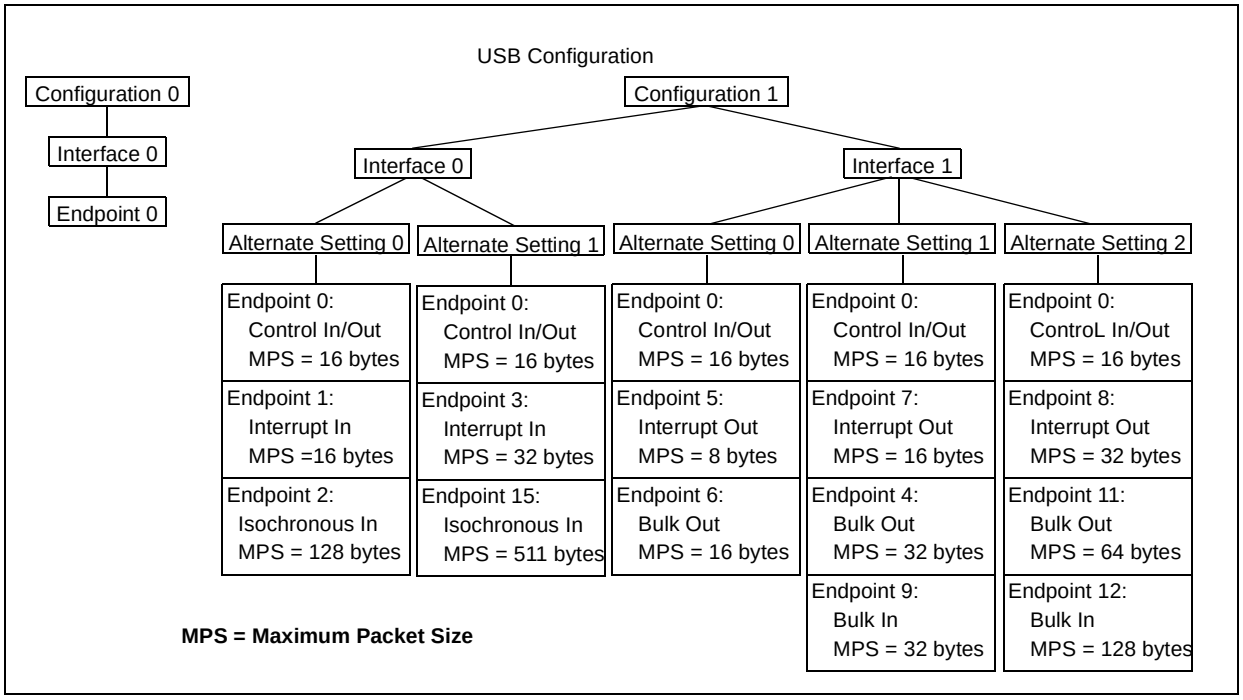


Table 5 lists each of the USB physical endpoints and the UDC programmable endpoint assigned to implement it. In Table 5, each UDC endpoint has a unique USB endpoint number. Although endpoint numbers can be duplicated across configurations, they cannot be duplicated within a single configuration; even 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 5: UDC Endpoint Configuration for Example USB Configuration

USB				UDC				
Config Number	IF Number	Alternate Interface Setting	EP Number	EP	Config Register	Double Buffered	Config 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
—	—	—	—	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

The 23 endpoints can be programmed to support USB endpoints throughout the three possible configurations, eight possible interfaces, and eight possible alternate interface settings. Clear UDCCR[UDE] before loading the Configuration registers. When UDCCR[UDE] is clear, the UDC Endpoint Configuration registers allow write accesses, the USB I/O signals are three-stated, and the

UDC cannot respond to USB host controller commands. Program the USB Endpoint Configuration registers and, when finished, load the Configuration registers into the USB interface block by setting UDCCR[UDE]. The USB Configuration registers are read-only when UDCCR[UDE] is set; the UDC memory allocation is checked. 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.

1.4.4 UDC Device Requests

The UDC Endpoint Control, Status, and Data registers are used to control and monitor the Transmit and Receive FIFOs for all UDC endpoints. The USB host controller controls all other UDC configuration and status reporting through the USB using device requests. Device requests are sent as control transfers to Endpoint 0. Each control-transfer Setup packet to Endpoint 0 is 8 bytes long and specifies the following:

- 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: used to pass a variable-sized data parameter
- Type of request.

The UDC decodes and responds to all but three of the standard-device requests with no software intervention. The UDC accepts the data and updates the appropriate internal UDC registers for the standard device requests Set_address, Set_feature, and Clear_feature. The data for these device requests is not forwarded to software through the Endpoint FIFO memory.

For Get_configuration, Get_interface, and Get_status, the UDC sends the appropriate response to the USB host controller with no software intervention and the device request is not forwarded to the software through the Endpoint FIFO memory.

Get_desciptor, Set_descriptor, and Synch_frame commands are passed through the Endpoint FIFO memory to software in response to the:

- Get_descriptor command, software must return a description of the UDC configuration.
- Synch_frame command, software must return the frame number for the requested isochronous endpoint or Stall Endpoint 0 if the Synch_frame command is sent for an endpoint that is not configured as isochronous.

Software must also handle vendor and class-specific requests. The UDC automatically manages the status stage of the control transfer and requires no software intervention.

If the host sends a Set_configuration or Set_interface transaction, the UDC accepts the data and updates the appropriate internal UDC registers. The data from these device requests is not forwarded to software through the Endpoint FIFO memory. The configuration and interface numbers are stored in the UDCCR. If UDCCSR0[ACM] is set, the UDC sends a NAK packet in the handshake phase of the SETUP transaction until the software sets UDCCSR0[AREN]. When AREN is set, the UDC sends an ACK packet for the handshake from the UDC to the host. If ACM is cleared, the UDC sends the appropriate response to the USB host controller with no software intervention.

The UDC sends an interrupt to the core when the USB host controller sends a Set_configuration or Set_interface command and the Interrupt-Enable bit for configuration change is set. Unload any data in the Endpoint memory and set UDCCR[SMAC] 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 the USB host controller sends a Set_configuration or Set_interface command, the UDC does not receive or transmit any Bulk, Interrupt, or Isochronous data until software sets UDCCR[SMAC]. When the UDC allocates the Endpoint memory for the new configuration, it is again ready for USB operation.

When UDCCR[UDE] is cleared, the configuration, interface, address, and features assigned or enabled by the USB host controller to the UDC are reset.

When the USB host controller sends a Set_feature command to enable the Device_remote_wakeup feature, the UDCCR[DRWE] (device-remote-wakeup-enable status) is set to indicate that the feature is enabled. When the USB host controller sends a Clear_feature command to disable the Device_remote_wakeupt feature, DRWE is cleared to indicate that the host has disabled the feature. Software can read the UDCCR at any time to determine whether the USB host controller has enabled the Device_remote_wakeup feature.

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 features, the UDC does not decode the command and responds with a stall unless UDCCR[OEN] (on-the-go enable) is set. When the USB host controller sends a Set_feature command to enable on-the-go specific features and UDCCR[OEN] is set, the On-the-Go status bits in the UDCCR are set to indicate that the feature is enabled. See Section 1.4.8 for details on the On-the-Go operation.

1.4.5 Endpoint Configuration

The UDC can physically support more data channel bandwidth than the USB allows. When responding to the USB host controller, software must specify a legal USB configuration. For example, if software specifies a configuration of six Isochronous endpoints of 256 bytes each, the USB host controller cannot schedule the proper bandwidth and does not take the UDC out of Configuration 0. Software must determine which endpoints to enable for each configuration and interface to ensure that the total number of endpoints allocated FIFO memory space does not exceed the available memory space 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.

Software must configure and enable all endpoints for each USB configuration and interface after a reset of the processor and before the UDC is enabled for USB operation. The USB interface block is held in reset until the UDC is enabled. The UDC I/O signals (USBOTG_P and USBOTG_N) are three-stated and unable to receive any USB host controller commands while the USB interface block is in reset. Program the Endpoint Configuration registers and then set UDCCR[UDE].

After the UDC is enabled, the Endpoint control block checks the programmed Endpoint memory allocations and if they are valid, loads the configuration for all enabled endpoints into the USB interface. It 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 1.4.3 for details on configuring endpoints, and Section 1.5.11 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.

1.4.6 Cable Attach and Detach

The USB host controller provides +5 V on the USB cable to comply with *Universal Serial Bus Specification*, *Revision 1.1*. Because the processor balls are not 5V tolerant, the power signal must be gated by an external level-shifting device, and the output routed to a GPIO ball, which sends an interrupt to software. When the GPIO indicates an attach event, software must enable the UDC by first configuring the Control and Configuration registers and then setting UDCCR[UDE]. Unload all data from the Endpoint memory first and then disable the UDC by clearing UDE when a Detach event is detected.

1.4.7 Suspend and Resume

The UDC detects the Suspend state if idle persists on the USB for more than 3 ms. An interrupt is sent to the processor if the Suspend interrupt is enabled. When the UDC enters the Suspend state, the processor stops the 48 MHz clock to the UDC and enables the UDC pins USBOTG_P and USBOTG_N to detect the Resume state. If the processor does not enter S2/D3/C4 mode, the state of the UDC is preserved and is ready for Resume detection.

The UDC can exit Suspend in three ways: Resume initiated by the UDC, Resume initiated by the USB host controller, or USB reset. If the USB host controller executes the Set_feature command and enables the device remote-wakeup feature of the UDC, after the UDC enters the Suspend state, it sends a wakeup signal to the USB host controller by setting UDCCR[UDR]. This signal forces the UDC to drive a non-idle state (K state) onto the USB for 3 ms without additional software intervention. The UDC hardware then clears UDR. The UDC waits for the USB host controller to reflect the Resume signal. An interrupt is sent to the processor when the Resume state is detected on the USB and the Resume interrupt is enabled.

The USB host controller can wake up the UDC by driving the non-idle state (K state) as either a Resume or USB reset onto the USB. The UDC pads signal the processor clock manager module to start the 48 MHz clock to the UDC when they detect this non-idle state on the USB. An interrupt is sent to the processor if the appropriate Interrupt Enable is set. Software must take the appropriate action to resume activity.

Refer to the Services module and PMU unit for information on programming for UDC wakeup events from Suspend.

1.4.8 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*[1]. The internal OTG transceivers provide on-chip pullup and pulldown resistors as specified in the *Pullup/Pulldown Engineering Change Notice to the USB 2.0 Specification*[2]. OTG operation requires software intervention but interrupts are provided to notify the software of OTG activities including Vbus changes, session detection and OTG ID changes. Use these interrupts along with the OTG

1. The latest revision of the *On-The-Go Supplement to the USB 2.0 Specification* can be accessed through the World Wide Web Internet site at: http://www.usb.org/
2. The latest revision of the *Pullup/Pulldown Resistors Engineering Change Notice to the Universal Serial Bus 2.0 Specification* can be accessed through the World Wide Web Internet site at: http://www.usb.org/

Control and Status registers to operate as an OTG device. The UDC OTG support includes the following:

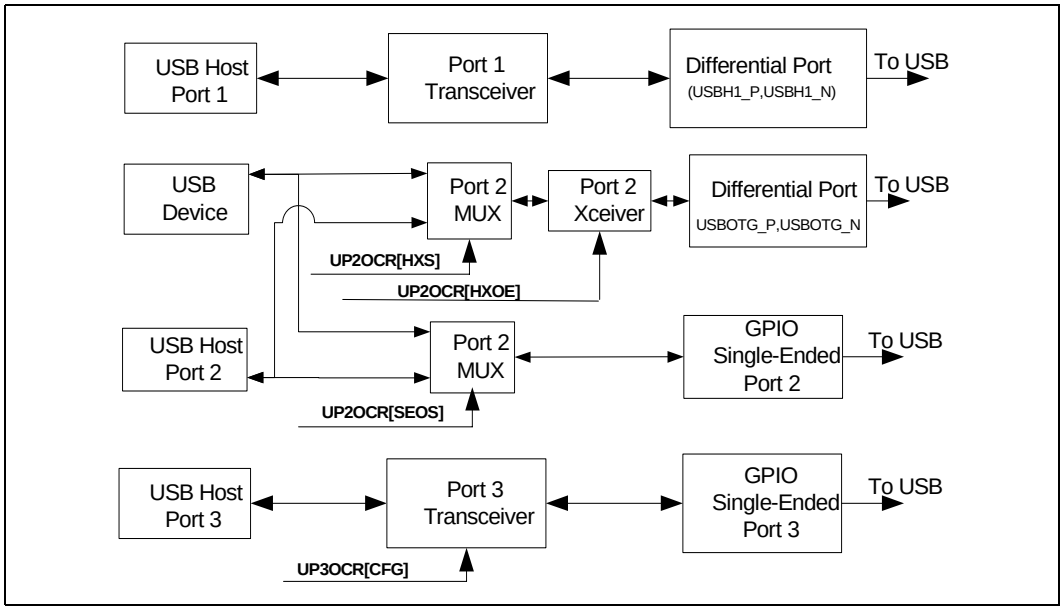
- Decode of Set_feature commands with OTG specific selector values
- Control for internal OTG transceiver with multiplexing between UDC and USB host controller (UHC) port 2
- Control for multiplexing between UDC, and UHC Port 2 through GPIOs
- Control, status and interrupt registers for interfacing to external OTG transceivers
- Control, status and interrupt registers for interfacing to external charge pump devices
- OTG ID support

Figure 13 shows the configurations that support OTG operation. Each configuration is discussed in detail.

Note

The PXA32x processor and PXA30x processor do not provide direct connection or control of the USB Vbus.

Figure 13: USB OTG Configurations



1.4.8.1 UDC OTG SET_FEATURE Commands

The UDC can selectively decode Set_feature commands with selector values of 3, 4, or 5 to enable the UDC for use in OTG and non-OTG modes. If a USB host controller sends a Set_feature command to enable OTG features and the OEN bit in the UDC Control Register (UDCCR) is set:

- The UDC decodes the command
- Responds with an ACK on the USB
- The corresponding OTG status bit in the UDC Control Register (UDCCR) is set to indicate that the feature is enabled.

If the USB host controller sends an OTG Set_feature command and UDCCR[OEN] is cleared:

- The UDC does not decode the command
- Responds with a stall on the USB
- Does not set any OTG status bits in the UDCCR.

The reset value for OEN is zero, so set OEN for OTG Set_feature commands to be decoded. Table 6 lists the OTG features and the UDCCR status bits assigned to each. When OEN is set, read the UDCCR at any time to determine whether the USB host controller has enabled the OTG features.

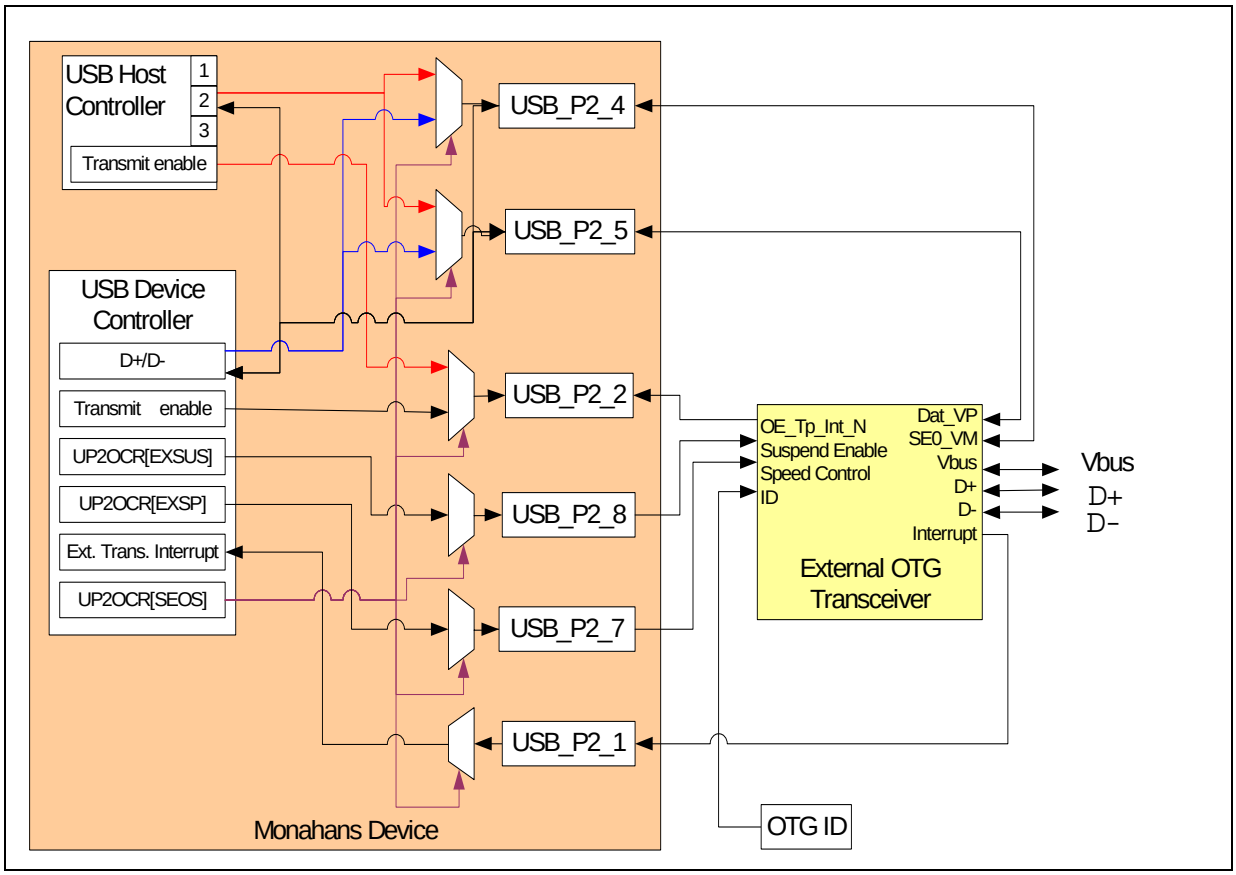
Table 6: On-The-Go Feature Selectors

Feature Selector	Value	UDCCR Status Bit	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 capable

1.4.8.2 Interface to External OTG Transceiver

The UDC contains control, status and interrupt registers to provide seamless interfacing to external transceivers where the internal OTG transceiver (Port 2 transceiver) is not used. External transceivers can be used to provide D+, D– and Vbus driver to the USB. The USB D+ and D– signals are output through GPIO pads, and UP2OCR[SEOS] is used to control multiplexers to select between UDC and USB host controller D+, D– and Transmit Enable signals. In addition, 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 14 illustrates the OTG connection to an external transceiver.

Figure 14: Connection to External OTG Transceiver



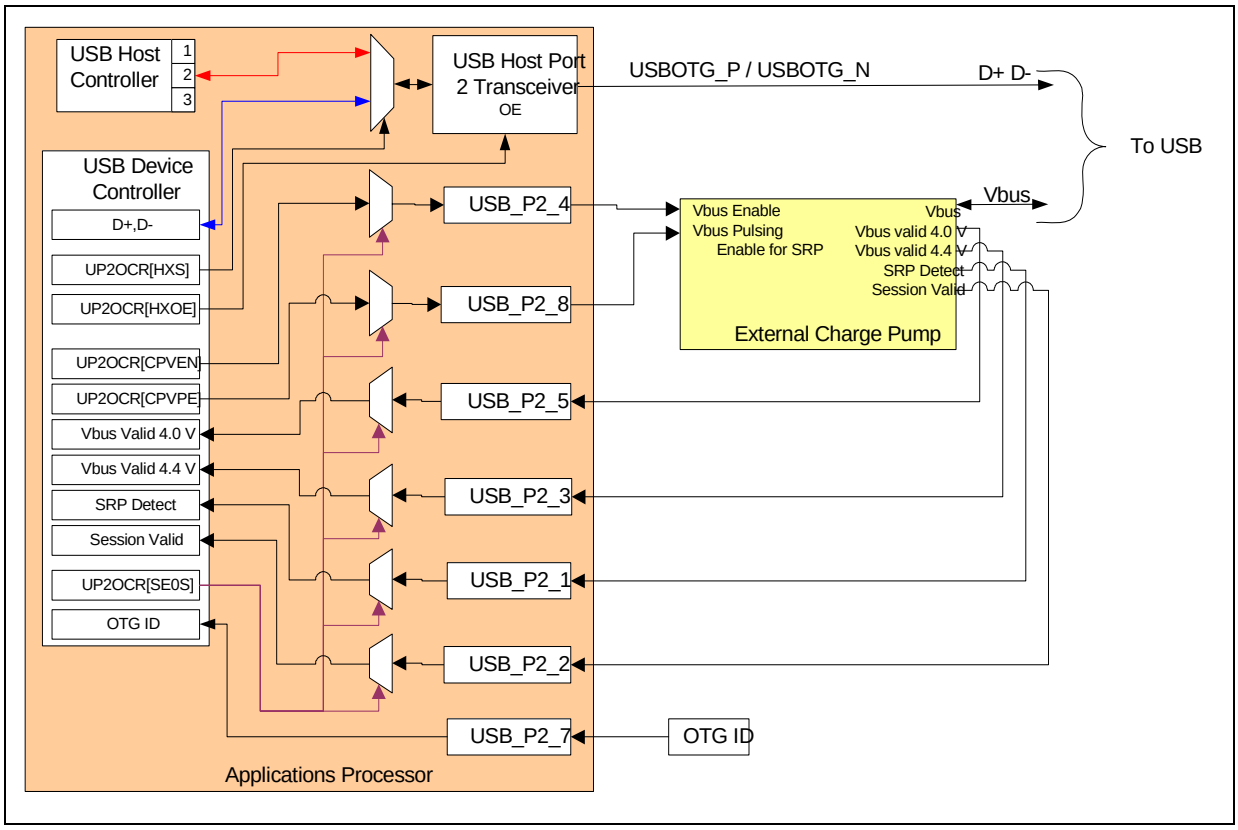
Note

For Figure 14, UP2OCR[SEOS] = 4 for the USB device and UP2OCR[SEOS] = 5 for the USB host.

1.4.8.3 Interface to External Charge Pump Device

The UDC provides control outputs and interrupt inputs to drive and monitor an external charge pump. The USB D+ and D– signals can be output using the internal OTG transceiver (Port 2 transceiver) and the Vbus interface provided by an external charge pump. UP2OCR[HXS] is used to control multiplexers 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 to enable the driving of Vbus and to enable the driving of pulses on Vbus, respectively. Additionally, UP2OCR provides the Vbus-Valid 4.0, Vbus-Valid 4.4, Session-Valid and Session-Request-Protocol-Detected interrupt inputs to interface to the external charge pump. Figure 15 illustrates the OTG connections to an external charge pump.

Figure 15: Connection to External OTG Charge Pump



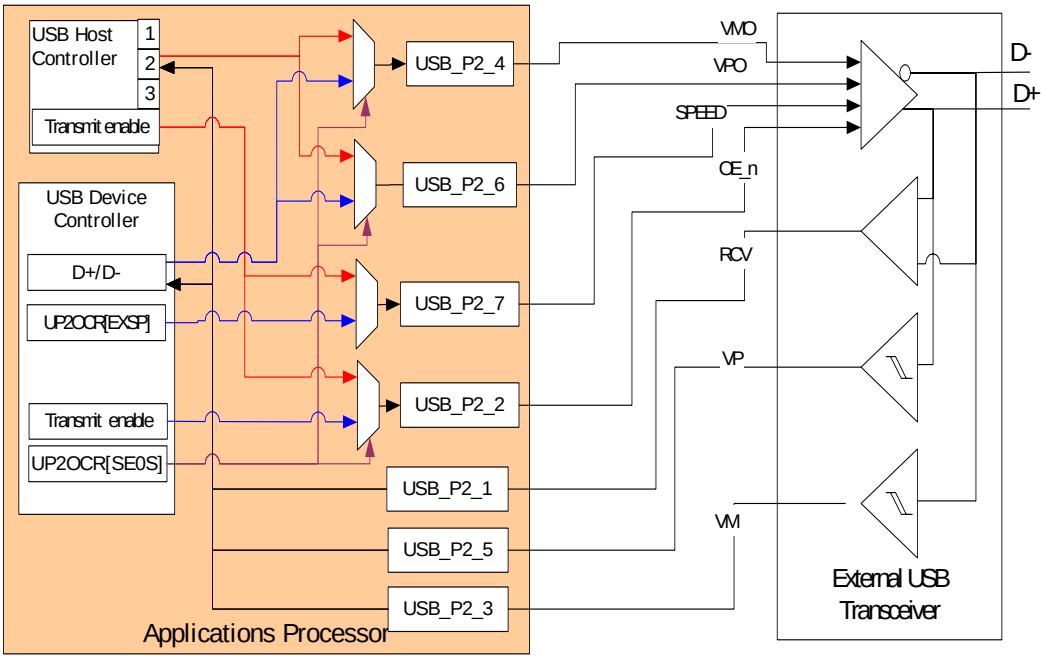
Note

For Figure 15, UP2OCR[SEOS] = 6 for USB device and UP2OCR[SEOS] = 7 for USB host.

1.4.8.4 Interface to External USB Transceiver

The UDC and USB host controller can interface to a non-OTG external USB transceiver through the single-ended interface with the GPIOs. 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 the SEOS field in the USB Port 2 Output Control Register (UP2OCR) is configured to a value of 2 or 3. Figure 16 illustrates the connection to an external USB transceiver.

Figure 16: Connection to External USB Transceiver



Note

For Figure 16, UP2OCR[SEOS] = 2 for USB Device and UP2OCR[SEOS] = 3 for USB host.

Table 7 and Table 8 list the definitions associated with the possible combinations of data in the external USB Transceiver mode.

Table 7: 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

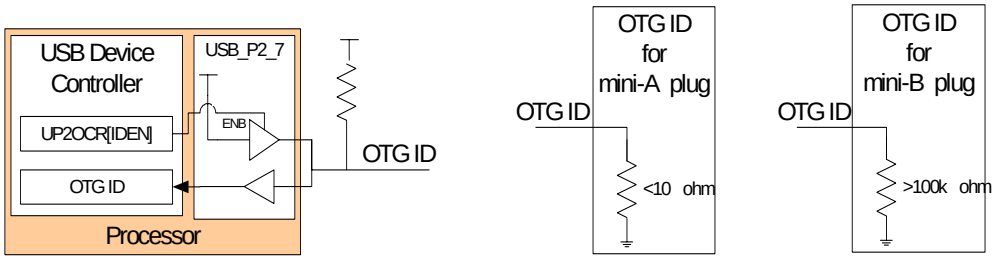
Table 8: Inputs from External USB Transceiver

P2_5/P3_5	P2_3/P3_3	Result
0	0	Single-Ended Zero (SE0)
0	1	Logic 0
1	0	Logic 1
1	1	Error

1.4.8.5 OTG ID

The UDC provides OTG ID interface support through USB_P2_7. The UDC provides ID output enable (UP2OCR[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 is enabled and driven high with a weak output driver so when the OTG ID pin on the USB connector is connected to a 100 k Ω resistor, the OTG ID input is a 1. When the OTG ID pin is connected to a 10 Ω resistor to ground, the USB_P2_7 output driver cannot drive OTG ID to a 1, resulting in an OTG ID input of zero. Figure 17 illustrates the interface to OTG ID.

Figure 17: Connection to OTG ID



1.4.9 Low-Power Modes

Refer to the “Clock Controllers and Power Management” chapter *in PXA3xx Processor Family: Vol. I System and Timer Configuration Developers Manual* for details on power modes. The UDC can be awakened from Low-power mode through a USB reset, USB resume, or generic wakeup event on one of the GPIO signals.

Note

The UDC retains internal state during S0/D1/C2 and S0/D2/C2 power modes and does not require programming or enumeration after exiting and returning to S0/D0/C0. However, the FIFO is not state retentive during any Low-power mode, so the data in the FIFO must be serviced prior to entering Low-power mode.

1.4.9.1 S0/D1/C2 and S0/D2/C2 Low-Power Modes

The following programming sequence is required for the UDC unit to enter and exit S0/D1/C2 Low-power mode.

1. UDC enters Suspend mode due to lack of USB traffic and informs the Application Subsystem Power Management Unit (APMU) of its status.
2. If UDC is Suspend-Enabled, program the GPIO bits to retain their current state (read the current GPIO Multi-Function Pin Register (MFPR) signal value and write that value to the GPIO MFPR) during S0/D1/C2 and S0/D2/C2, except for the V- IN pin, which is programmed for Edge-Detect and is the Wakeup mechanism. Otherwise, in Differential mode, the D+/D– transition is the Wakeup mechanism.
3. Write to the Application Subsystem Wakeup from S0/D1/C2 to S0/D0/C0 State Enable register (AD1D0ER) and Application Subsystem Wakeup from S0/D2/C2 to S0/D0/C0 State Enable register (AD2D0ER) to specify the UDC Wakeup mechanism (SE or differential).
4. Enter Low-power mode via Writes to the core PWRMODE register (CP14 register).
5. The APMU unit waits on the appropriate Wakeup event and then turns on the UDC clock.
6. With the return of the clock, the UDC unit detects when a Wakeup event occurred and resumes itself.

1.4.9.2 S2/D3/C4 Low-Power Mode

If the UDC enters the Suspend state before the processor is in S2/D3/C4 mode, the UDC USBOTG_P and USBOTG_N signals are used to detect the Resume state on the USB and resume operation. If the USB host controller tries to access the UDC when the processor is in S2/D3/C4 mode, the USBOTG_P and USBOTG_N signals detect the Resume state and signal the processor to begin the Wakeup sequence. The UDC has lost all state information and the UDC Configuration registers must be reloaded before UDCCR[UDE] is set. The USB host controller must issue a USB reset and re-enumerate the UDC (see the Peripheral module PMU unit for details on the Wakeup sequence and USB operation during S2/D3/C4 mode).

If the UDC is disconnected from the USB and the processor is in S2/D3/C4 mode, a GPIO signal must be programmed to detect connection to the USB and to signal the processor to begin the Wakeup sequence. The GPIO signal must be connected through a Level Shifter to the USB power signal to detect connect/disconnect to the USB. The UDC has lost all state information and software must load the UDC Configuration register and enable the UDC before the UDC can be ready for USB operation.

The following programming sequence is required for the UDC unit to enter and exit S2/D3/C4 Low-power mode.

1. UDC enters Suspend mode due to lack of USB traffic and informs the APMU of its status.
2. If UDC is Suspend-Enabled, program the GPIO Multi-Function Pin Registers (MFPR) bits to retain their current state (read the current GPIO MFPR signal value and write that value to the GPIO MFPR) during S2/D3/C4 except for the V- IN pin, which is programmed for Edge-Detect and is the Wakeup mechanism. Otherwise, in Differential mode, the D+/D– transition is the Wakeup mechanism.
3. Write to the Application Subsystem Wake-up from D3 Enable Register (AD3ER) register to specify the UDC Wakeup mechanism (Suspend-Enable or Differential).
4. To enter Low-power mode, write to the Core PWRMODE Register (CP14 Register 7).
5. APMU unit waits on the appropriate Wakeup event and starts the process to exit D3.
6. UDC has lost all state information, requiring a reload of the UDC Configuration registers and an enable of the UDC before the UDC is ready for USB operation. Expect the host to issue a USB reset and re-enumerate the UDC.

1.5 Registers

Table 9 summarizes the UDC registers and the memory-mapped addresses to access them. The summary table includes the page number of the detailed description for each register for easy reference. The remainder of this section describes the individual registers in detail.

Note

All registers except data registers (UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)) are 32 bits and cannot be addressed as partial bytes.

Table 9: UDC Register Summary

Address	Description	Page	Notes
0x4060_0000	UDC Control Register (UDCCR)	58	
0x4060_0004	UDC Interrupt Control Register 0 (UDCICR0)	65	
0x4060_0008	UDC Interrupt Control Register 1(UDCICR1)	67	
0x4060_000C	UDC Interrupt Status Register 0 (UDCISR0)	72	
0x4060_0010	UDC Interrupt Status Register 1(UDCISR1)	74	
0x4060_0014	UDC Frame Number Register (UDCFNR)	76	
0x4060_0018	UDC OTG Interrupt Control Register (UDCOTGICR)	75	
0x4060_001C	UDC Interrupt Status Register 1(UDCISR1)	71	
0x4060_0020	USB Port 2 Output Control Register (UP2OCR)	77	
0x4060_0024	USB Port 3 Output Control Register (UP3OCR)	83	
0x4060_0028 – 0x4060_00FC	Reserved		
0x4060_0100	UDC Endpoint 0 Control Status Register (UDCCSR0)	85	Endpoint 0
0x4060_0104	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint A
0x4060_0108	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint **B**
0x4060_010C	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint C
0x4060_0110	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint D
0x4060_0114	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint E
0x4060_0118	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint F
0x4060_011C	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint G
0x4060_0120	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint H
0x4060_0124	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint I

Table 9: UDC Register Summary (Continued)

Address	Description	Page	Notes
0x4060_0128	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint J
0x4060_012C	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint K
0x4060_0130	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint L
0x4060_0134	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint M
0x4060_0138	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint N
0x4060_013C	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint P
0x4060_0140	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint Q
0x4060_0144	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint R
0x4060_0148	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint S
0x4060_014C	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint T
0x4060_0150	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint U
0x4060_0154	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint V
0x4060_0158	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint W
0x4060_015C	UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)	88	Endpoint X
0x4060_0160 – 0x4060_01FC	Reserved		
0x4060_0200	UDC Endpoint 0 Byte Count Register (UDCBCR0)	98	Endpoint 0
0x4060_0204	UDC Endpoint A Byte Count Register (UDCBCRA)	98	Endpoint A
0x4060_0208	UDC Endpoint B Byte Count Register (UDCBCRB)	98	Endpoint **B**
0x4060_020C	UDC Endpoint C Byte Count Register (UDCBCRC)	98	Endpoint C
0x4060_0210	UDC Endpoint D Byte Count Register (UDCBCRD)	98	Endpoint D
0x4060_0214	UDC Endpoint E Byte Count Register (UDCBCRE)	98	Endpoint E
0x4060_0218	UDC Endpoint F Byte Count Register (UDCBCRF)	98	Endpoint F
0x4060_021C	UDC Endpoint G Byte Count Register (UDCBCRG)	98	Endpoint G
0x4060_0220	UDC Endpoint H Byte Count Register (UDCBCRH)	98	Endpoint H
0x4060_0224	UDC Endpoint I Byte Count Register (UDCBCRI)	98	Endpoint I
0x4060_0228	UDC Endpoint J Byte Count Register (UDCBCRJ)	98	Endpoint J
0x4060_022C	UDC Endpoint K Byte Count Register (UDCBCRK)	98	Endpoint K
0x4060_0230	UDC Endpoint L Byte Count Register (UDCBCRL)	98	Endpoint L
0x4060_0234	UDC Endpoint M Byte Count Register (UDCBCRM)	98	Endpoint M

Table 9: UDC Register Summary (Continued)

Address	Description	Page	Notes
0x4060_0238	UDC Endpoint N Byte Count Register (UDCBCRN)	98	Endpoint N
0x4060_023C	UDC Endpoint P Byte Count Register (UDCBCRP)	98	Endpoint P
0x4060_0240	UDC Endpoint Q Byte Count Register (UDCBCRQ)	98	Endpoint Q
0x4060_0244	UDC Endpoint R Byte Count Register (UDCBCRR)	98	Endpoint R
0x4060_0248	UDC Endpoint S Byte Count Register (UDCBCRS)	98	Endpoint S
0x4060_024C	UDC Endpoint T Byte Count Register (UDCBCRT)	98	Endpoint T
0x4060_0250	UDC Endpoint U Byte Count Register (UDCBCRU)	98	Endpoint U
0x4060_0254	UDC Endpoint V Byte Count Register (UDCBCRV)	98	Endpoint V
0x4060_0258	UDC Endpoint W Byte Count Register (UDCBCRW)	98	Endpoint W
0x4060_025C	UDC Endpoint X Byte Count Register (UDCBCRX)	98	Endpoint X
0x4060_0260 – 0x4060_02FC	Reserved		
0x4060_0300	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint 0
0x4060_0304	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint A
0x4060_0308	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint **B**
0x4060_030C	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint C
0x4060_0310	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint D
0x4060_0314	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint E
0x4060_0318	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint F
0x4060_031C	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint G
0x4060_0320	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint H
0x4060_0324	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint I
0x4060_0328	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint J
0x4060_032C	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint K
0x4060_0330	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint L
0x4060_0334	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint M
0x4060_0338	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint N
0x4060_033C	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint P
0x4060_0340	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint Q
0x4060_0344	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint R

Table 9: UDC Register Summary (Continued)

Address	Description	Page	Notes
0x4060_0348	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint S
0x4060_034C	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint T
0x4060_0350	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint U
0x4060_0354	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint V
0x4060_0358	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint W
0x4060_035C	UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)	99	Endpoint X
0x4060_0360 – 0x4060_03FC	Reserved		
0x4060_0400	Reserved		
0x4060_0404	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint A
0x4060_0408	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint **B**
0x4060_040C	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint C
0x4060_0410	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint D
0x4060_0414	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint E
0x4060_0418	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint F
0x4060_041C	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint G
0x4060_0420	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint H
0x4060_0424	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint I
0x4060_0428	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint J
0x4060_042C	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint K
0x4060_0430	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint L
0x4060_0434	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint M
0x4060_0438	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint N
0x4060_043C	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint P
0x4060_0440	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint Q
0x4060_0444	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint R
0x4060_0448	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint S
0x4060_044C	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint T
0x4060_0450	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint U
0x4060_0454	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint V

Table 9: UDC Register Summary (Continued)

Address	Description	Page	Notes
0x4060_0458	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint W
0x4060_045C	UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)	100	Endpoint X
0x4060_0460 – 0x406F_FFFC	Reserved		

1.5.1 UDC Control Register (UDCCR)

UDCCR, shown in Table 10, contains control and status bits. 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 for use by the UDC. The ACN is set when the USB host controller issues a Set_configuration command to the UDC. The AIN and AAISN numbers are set when the USB host controller issues a Set_interface command to the UDC. See Section 1.4.4 for details on the execution of USB device requests.

If the Interrupt Enable Configuration Change bit (UDCICR1[IECC]) is set, the UDC generates an interrupt to the processor to indicate that the Set_configuration or Set_interface command has completed. Set SMAC to change the Endpoint memory allocation to the Active Configuration, Active Interface, and Active Alternate Interface Setting.

Table 10: UDCCR Bit Definitions

Physical Address 0x40600000 — UDCCR — UDC

	31	30	29	28	27	26	25	24	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		PWRMD	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	0

Bits	Access	Name	Description
31	R/W	OEN	On-the-Go Enable Enables/disables hardware control of the decoding of OTG Set_feature commands. When OEN is cleared, 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.

Table 10: UDCCR Bit Definitions (Continued)

Physical Address 0x40600000 **UDCCR** **UDC**

	31	30	29	28	27	26	25	24	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		PWRMD	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	0

Bits	Access	Name	Description
30	R	AALTHNP	A-device Alternate Host Negotiation Protocol Port Support Indicates whether the USB host controller has enabled the A-device alternate HNP feature. AALTHNP is set when the USB host controller executes a SET_feature command and indicates that 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 whether the USB host controller has enabled the A-device HNP feature. The AHNP bit is set when the USB host controller executes a Set_feature command and indicates that 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 directly connected to an A-device port that supports HNP.
28	R	BHNP	B-device Host Negotiation Protocol Enable Indicates whether the USB host controller has enabled the B-device HNP feature. BHNP is set when the USB host controller executes a Set_feature command and indicates that the b_hnp_enable feature is enabled and the OEN bit is set. 0 = B-device HNP enable feature not enabled. 1 = B-device HNP enable feature is enabled.
27:17	—	Reserved	Reserved
16	R	DWRE	Device Remote Wakeup Feature Indicates whether the USB host controller has enabled the device remote wakeup feature. DWRE is set when the USB host controller executes a Set_feature command and indicates that the device remote wakeup feature is to be enabled. 0 = Device remote wakeup feature not enabled. 1 = Device remote wakeup feature is enabled.
15:14	—	Reserved	Reserved
13	Read/Write	PWRMD	Power Mode 0 = Self powered (Default reset value) 1 = Bus powered

Table 10: UDCCR Bit Definitions (Continued)

Physical Address 0x40600000	UDCCR	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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		PWRMD	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	0

Bits	Access	Name	Description
12:11	Read	ACN	Active UDC Configuration Number (0–3) Selected by the USB host controller for use by the UDC. ACN is set when the USB host controller issues a Set_configuration command to the UDC.
10:8	Read	AIN	Active UDC Interface Number (0–7) Selected by the USB host controller for use by the UDC. AIN is set when the USB host controller issues a Set_interface command to the UDC.
7:5	Read	AAISN	Active UDC Alternate Interface Setting Number (0–7) Selected by the USB host controller for use by the UDC. AAISN 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 Controls 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 SMAC causes the UDC to flush the Endpoint memory and change the Endpoint memory allocation to reflect the Active Configuration and interface set by the USB host controller. The UDC clears SMAC when the memory allocation completes. See Section 1.4.4 for details on the execution of USB device requests. The UDC NAKs all Bulk and Interrupt transactions received after the Set_configuration or Set_interface command executes but before the UDC changes the Endpoint memory to the new configuration. The UDC indicates an Overflow condition if an isochronous OUT data is received between the execution of a Set_configuration or Set_interface command. Changing of the Endpoint memory allocation sends Zero-size packets if an isochronous in token is received in the same circumstance. 0 = No change to Active Configuration. 1 = Change the Endpoint memory allocation to the Active Configuration, Active Interface, and Active Alternate Interface settings.

Table 10: UDCCR Bit Definitions (Continued)

Physical Address 0x40600000 **UDCCR** **UDC**

User Settings																															
Bit	31	30	29	28	27	26	25	24	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		PWRMD	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	0

Bits	Access	Name	Description
3	Read/Write 1 to Clear	EMCE	Endpoint Memory Configuration Error Indicates that the Endpoint configuration cannot be loaded and the maximum packet sizes indicated for the configuration are in error. When UDE is set, the UDC checks the endpoint configurations to verify that 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 EMCE and clears UDE. EMCE must be cleared before an attempt to re-enable the UDC. 0 = No Endpoint memory configuration error. 1 = The Endpoint memory configuration has an error and cannot be loaded.
2	Read/Write 1 to Set	UDR	UDC Resume If the USB host controller executes the Set_feature command and enables the device Remote Wakeup feature of the UDC, setting UDR when the UDC is in a Suspend state forces the UDC into a non-idle state (K state) to perform the Remote Wakeup operation. When UDR is set, the UDC drives resume signaling on the USBOTG_P and USBOTG_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 returns to Suspend state. The UDC clears UDR when it enters the non-idle state. If the USB host controller has not enabled the device Remote Wakeup feature of the UDC, UDR is ignored. 0 = Maintain UDC Suspend state. 1 = Force UDC out of Suspend.
1	Read	UDA	UDC Active Indicates whether the UDC is currently involved in a USB transaction. 0 = UDC currently inactive on the USB. 1 = UDC currently active on the USB.

Table 10: UDCCR Bit Definitions (Continued)

Physical Address 0x40600000	UDCCR	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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		PWRMD	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	0

Bits	Access	Name	Description
0	Read/Write	UDE	UDC Enable Enables/disables the UDC. UDE is cleared after a reset of the processor, which disables the UDC, three-states USBOTG_P and USBOTG_N, and disables monitoring of the USB. Any USB host controller commands or USB reset issued while UDE is cleared are ignored. The Endpoint Configuration registers must be programmed before UDE is set. When 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 UDE is cleared, the entire UDC is disabled and reset. If UDE is cleared 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 UDE is cleared, the UDC Endpoint Configuration registers are set to read/write access and can be programmed for another configuration. The configuration, interface, address, and features that the USB host controller assigns or enables to the UDC are reset. 0 = UDC disabled, USBOTG_P and USBOTG_N three-stated. 1 = UDC enabled, USBOTG_P and USBOTG_N used for transmission/reception.

1.5.2 UDC Interrupt Control Registers (UDCICR0, UDCICR1, and UDCOTGICR)

UDCICR0 contains 32 Read/Write control bits to enable interrupt-service requests from Data Endpoint 0–Endpoint 0. UDCICR1 contains 21 control bits to enable interrupt-service requests for Endpoints Q –X and specific USB events. UDCOTGICR 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.

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 blocks only future 0-to-1 transitions of the Interrupt request and the future setting of the corresponding Interrupt-Request bit in the Interrupt-Status register.

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—Generated when a FIFO underrun or overrun occurs. For an In endpoint, the 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 are 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). The interrupt is generated when a FIFO overrun occurs for an Out endpoint. 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. Because Endpoint 0 is a bidirectional endpoint, a FIFO error can occur on either a FIFO-overrun or FIFO-underrun condition. Refer to Section 1.4.1.1.1, Section 1.4.1 and Section 1.4.3 for details on FIFO error conditions.

Note

Endpoint A - X FIFO Error Interrupts occur only if the endpoint is configured as Isochronous.

- Packet Complete Interrupt— The packet-complete interrupt indicates that an Out packet is received and is ready to be read for an Out endpoint. For an In endpoint, the interrupt is generated when an In packet is transmitted. The Endpoint 0 Packet-Complete interrupt is also generated by the DME bit in the corresponding Endpoint 0 Control/Status register.

Table 11 shows the USB events interrupts, enable and status bits, and USB event that cause the UDC to generate each interrupt. For the Interrupt Enables for Reset (IERS), Suspend (IESU), Resume (IERU), SOF (IESOF) and Configuration-Change (IECC) conditions, each bit in the UDCICR1 is used to enable the 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 is cleared, 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; if the Interrupt-Enable bit is set, an interrupt is also generated and the bit in the Interrupt Status register is set.

Table 11: USB Events 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 shows the USB On-the-Go Event interrupts, the enable and status bits, and the USB event that causes each interrupt to be generated by the UDC. Each bit in the UDCOTGICR enables a respective interrupt request for the On-the-Go interrupt enables. When the Interrupt-Enable bit is set, the interrupt is enabled and is generated when the USB on-the-go event occurs. The interrupt is disabled and is not generated when the Interrupt-Enable bit is cleared. 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; if the Interrupt-Enable bit is set, an interrupt is also generated, and the bit in the Interrupt-Status register is set.

Table 12: USB On-The-Go Events Interrupts

Interrupt Enable Bit (UDCOTGICR)	Interrupt Status Bit (UDCOTGISR)	USB OTG Condition that Generates 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 Rising Edge
IEVV40F	IRVV40F	OTG Vbus Valid 4.0 Falling Edge
IEVV44R	IRVV44R	OTG Vbus Valid 4.4 Rising Edge
IEVV44F	IRVV44F	OTG Vbus Valid 4.4 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

The OTG interrupts notify software when an OTG Set_feature command is received and are 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 is decoded and the OTG Feature status bit is set in the UDC Control Register (UDCCR). The External Transceiver Interrupt Rising Edge interrupt is set when the interrupt input from an external USB transceiver transitions from 0 to 1. Similarly, the External Transceiver Interrupt Falling Edge interrupt is set when the interrupt input from an external USB transceiver transitions from 1 to 0. Select the appropriate transition for the external USB transceiver in the system. See Section 1.4.8, USB On-The-Go Operation.

The OTG Vbus-Valid interrupts are used to interface to an external charge pump device and to detect the Vbus voltage levels. OTG Vbus Valid 4.0V 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 4.4V interrupts detect that 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 4.0V Falling Edge interrupt indicates the A-device output voltage has fallen below the $V_{A\_VBUS\_VLD}$ minimum specified value. The OTG Vbus-Valid 4.4V Rising Edge interrupt indicates the A-device output voltage has reached the valid threshold of 4.4 V.

The OTG Session-Valid interrupts are used to 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:

1. An active OTG session.
2. The Vbus is not ready for SRP.

If the OTG charge pump Session-Valid output signal is active high, the OTG Session-Valid Falling Edge interrupt indicates:

1. An OTG session is not valid.
2. The bus is ready for SRP.

The OTG A-device SRP-Detect interrupts are used to 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. The OTG A-device SRP Falling-Edge interrupt indicates the end of the SRP.

The register organization and individual bit definitions for UDCICR0, UDCICR1, and UDCOTGICR are shown in Table 13, Table 14, and Table 15, respectively.

1.5.2.1 UDC Interrupt Control Register 0 (UDCICR0)

UDCICR0, defined in Table 13, contains 32 read/write control bits to enable interrupt-service requests from Data Endpoint 0–Endpoint 0.

Table 13: UDCICR0 Bit Definitions

Physical Address 0x4060_0004 **UDCICR0** **UDC**

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	Read/Write	IEP	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
29:28	Read/Write	IEN	Endpoint N 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error Interrupt enabled 0b11 = Both Packet Complete and FIFO Error Interrupts enabled
27:26	Read/Write	IEM	Endpoint M 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error Interrupt enabled 0b11 = Both Packet Complete and FIFO Error Interrupts enabled

Table 13: UDCICR0 Bit Definitions (Continued)

Physical Address 0x4060_0004	UDCICR0	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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
25:24	Read/Write	IEL	Endpoint L 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
23:22	Read/Write	IEK	Endpoint K 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
21:20	Read/Write	IEJ	Endpoint J 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
19:18	Read/Write	IEI	Endpoint I 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
17:16	Read/Write	IEH	Endpoint H 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
15:14	Read/Write	IEG	Endpoint G 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
13:12	Read/Write	IEF	Endpoint F 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled\
11:10	Read/Write	IEE	Endpoint E 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled

Table 13: UDCICR0 Bit Definitions (Continued)

Physical Address 0x4060_0004	UDCICR0	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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
9:8	Read/Write	IED	Endpoint D 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
7:6	Read/Write	IEC	Endpoint C 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
5:4	Read/Write	IEB	Endpoint B 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
3:2	Read/Write	IEA	Endpoint A 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
1:0	Read/Write	IE0	Endpoint 0 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled

1.5.2.2 UDC Interrupt Control Register 1(UDCICR1)

UDCICR1, defined in Table 14 contains 21 control bits to enable interrupt-service requests for endpoints Q –X and specific USB events.

Table 14: UDCICR1 Bit Definitions

Physical Address 0x4060_0008 **UDCICR1** **UDC**

	31	30	29	28	27	26	25	24	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	Read/Write	IECC	Configuration Change 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
30	Read/Write	IESOF	Start of Frame 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
29	Read/Write	IERU	Resume
28	Read/Write	IESU	Suspend 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
27	Read/Write	IERS	Reset 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
26:16	—	Reserved	Reserved
15:14	Read/Write	IEX	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
13:12	Read/Write	IEW	Endpoint W 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled

Table 14: UDCICR1 Bit Definitions (Continued)

Physical Address 0x4060_0008 **UDCICR1** **UDC**

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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
11:10	Read/Write	IEV	Endpoint V 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
9:8	Read/Write	IEU	Endpoint U 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
7:6	Read/Write	IET	Endpoint T 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
5:4	Read/Write	IES	Endpoint S 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled
3:2	Read/Write	IER	Endpoint R 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled\
1:0	Read/Write	IEQ	Endpoint Q 0b00 = No interrupts enabled 0b01 = Packet Complete interrupt enabled 0b10 = FIFO Error interrupt enabled 0b11 = Both Packet Complete and FIFO Error interrupts enabled

1.5.2.3 UDC OTG Interrupt Control Register (UDCOTGICR)

UDCOTGICR, defined in Table 15, 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.

Table 15: UDCOTGICR Bit Definitions

Physical Address 0x4060_0018 UDCOTGICR UDC

Bit	31	30	29	28	27	26	25	24	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	Reserved
24	Read/Write	IESF	OTG Set_feature command received 0 = Interrupt disabled 1 = Interrupt enabled
23:18	—	Reserved	Reserved
17	Read/Write	IEXR	External Transceiver Interrupt Rising Edge 0 = Interrupt disabled 1 = Interrupt enabled
16	Read/Write	IEXF	External Transceiver Interrupt Falling Edge 0 = Interrupt disabled 1 = Interrupt enabled
15:10	—	Reserved	Reserved
9	Read/Write	IEVV40R	OTG Vbus Valid 4.0 V Rising Edge 0 = Interrupt disabled 1 = Interrupt enabled
8	Read/Write	IEVV40F	OTG Vbus Valid 4.0 V Falling Edge 0 = Interrupt disabled 1 = Interrupt enabled
7	Read/Write	IEVV44R	OTG Vbus Valid 4.4 V Rising Edge 0 = Interrupt disabled 1 = Interrupt enabled
6	Read/Write	IEVV44F	OTG Vbus Valid 4.4 V Falling Edge 0 = Interrupt disabled 1 = Interrupt enabled
5	Read/Write	IESVR	OTG Session Valid Rising Edge 0 = Interrupt disabled 1 = Interrupt enabled
4	Read/Write	IESVF	OTG Session Valid Falling Edge 0 = Interrupt disabled 1 = Interrupt enabled
3	Read/Write	IESDR	OTG A-Device SRP Detect Rising Edge 0 = Interrupt disabled 1 = Interrupt enabled

Table 15: UDCOTGICR Bit Definitions (Continued)

Physical Address 0x4060_0018 **UDCOTGICR** **UDC**

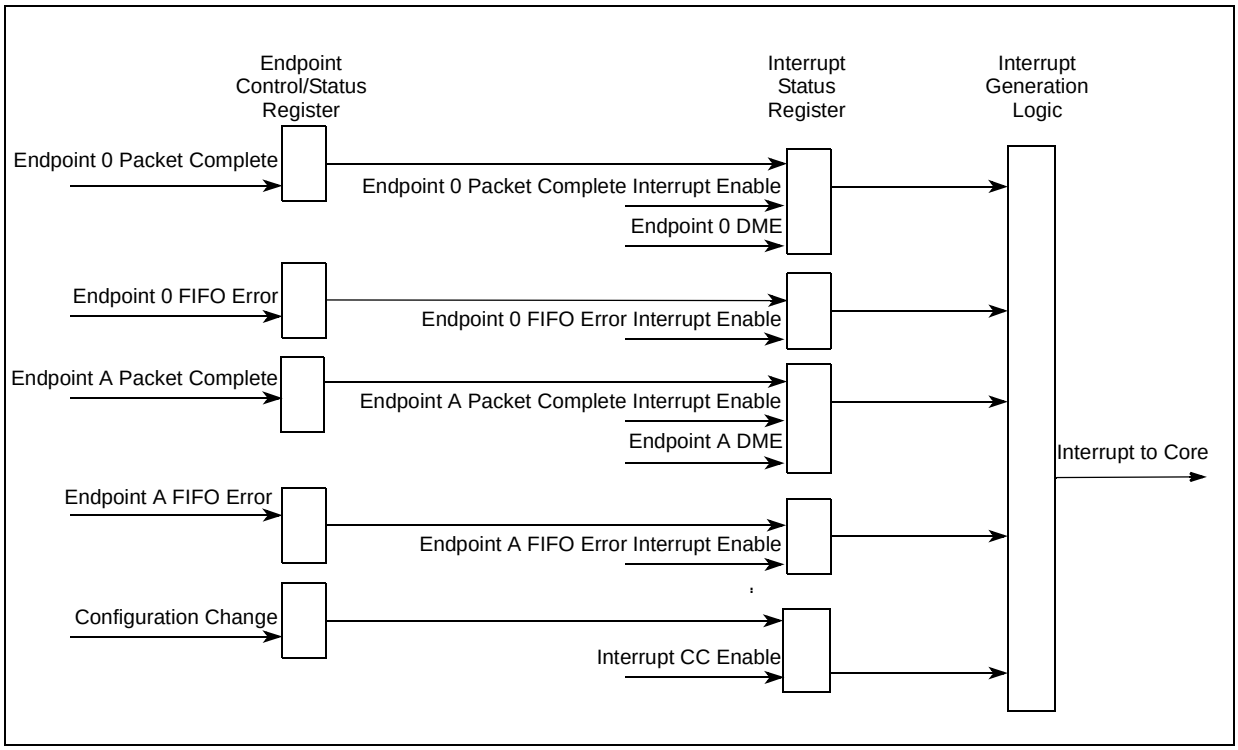
User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved							IESF	Reserved						IEXR	IEXF	Reserved						IEVV40R	IEVV40F	IEVV44R	IEVV44F	IESVR	IESVF	IESDR	IESDF	IEIDR	IEIDF
Reset	?	?	?	?	?	?	?	0	?	?	?	?	?	?	0	0	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	Read/Write	IESDF	OTG A-Device SRP Detect Falling Edge 0 = Interrupt disabled 1 = Interrupt enabled
1	Read/Write	IEIDR	OTG ID Change Rising Edge 0 = Interrupt disabled 1 = Interrupt enabled
0	Read/Write	IEIDF	OTG ID Change Falling Edge 0 = Interrupt disabled 1 = Interrupt enabled

1.5.3 UDC Interrupt Status Registers (UDCISR0, UDCISR1, and UDCOTGISR)

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 18 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 18: 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). The enable bits, when cleared, prevent a status bit in the corresponding UDC Interrupt Status register from generating an interrupt. The Interrupt-Status bit is set and a UDC interrupt is generated when the enable bit for a particular status bit is set and an interruptible condition occurs. To clear Interrupt-Status bits, write a 1 to the bit position to be cleared.

UDCISR0 contains Interrupt-Status bits for Endpoint 0 through Endpoint P; UDCISR1 contains Interrupt-Status bits for Endpoint Q through Endpoint X. Each endpoint has the potential of two interrupt sources: FIFO Error and Packet Complete. UDCOTGISR contains Interrupt-Status bits for 12 USB On-the-Go events. UDCISR1 also contains Interrupt-Status bits for five USB events (see Section 1.5.2 for details on the use of these interrupts).

The register organization and individual bit definitions are shown in Table 16, Table 17, and Table 18.

1.5.3.1 UDC Interrupt Status Register 0 (UDCISR0)

UDCISR0, defined in Table 16, 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.

Table 16: UDCISR0 Bit Definitions

Physical Address 0x4060_000C **UDCISR0** **UDC**

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	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	Read/Write 1 to clear	IRP	Endpoint P
29:28	Read/Write 1 to clear	IRN	Endpoint N
27:26	Read/Write 1 to clear	IRM	Endpoint M
25:24	Read/Write 1 to clear	IRL	Endpoint L
23:22	Read/Write 1 to clear	IRK	Endpoint K
21:20	Read/Write 1 to clear	IRJ	Endpoint J
19:18	Read/Write 1 to clear	IRI	Endpoint I
17:16	Read/Write 1 to clear	IRH	Endpoint H
15:14	Read/Write 1 to clear	IRG	Endpoint G
13:12	Read/Write 1 to clear	IMF	Endpoint F
11:10	Read/Write 1 to clear	IRE	Endpoint E
9:8	Read/Write 1 to clear	IRD	Endpoint D
7:6	Read/Write 1 to clear	IRC	Endpoint C
5:4	Read/Write 1 to clear	IRB	Endpoint B
3:2	Read/Write 1 to clear	IRA	Endpoint A
1:0	Read/Write 1 to clear	IR0	Endpoint 0

1.5.3.2 UDC Interrupt Status Register 1(UDCISR1)

UDCISR1, defined in Table 17, contains bits to generate the UDC interrupt request. Each bit in the UDC Interrupt Status registers is logically ORed together to produce one interrupt request.

Table 17: UDCISR1 Bit Definitions

Physical Address 0x4060_0010																UDC Interrupt Status Register 1															
User Settings																															
Bit	31	30	29	28	27	26	25	24	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	Read/Write 1 to clear	IRCC	Configuration change
30	Read/Write 1 to clear	IRSOF	Start of frame
29	Read/Write 1 to clear	IRRU	Resume
28	Read/Write 1 to clear	IRSU	Suspend
27	Read/Write 1 to clear	IRRS	Reset
26:16	—	—	Reserved
15:14	Read/Write 1 to clear	IRX	Endpoint X
13:12	Read/Write 1 to clear	IRW	Endpoint W
11:10	Read/Write 1 to clear	IRV	Endpoint V
9:8	Read/Write 1 to clear	IRU	Endpoint U
7:6	Read/Write 1 to clear	IRT	Endpoint T
5:4	Read/Write 1 to clear	IRS	Endpoint S
3:2	Read/Write 1 to clear	IRR	Endpoint R

Table 17: UDCISR1 Bit Definitions (Continued)

Physical Address 0x4060_0010	UDC Interrupt Status Register 1	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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
1:0	Read/Write 1 to clear	IRQ	Endpoint Q

1.5.3.3 UDC OTG Interrupt Status Register (UDCOTGISR)

UDCOTGISR, defined in Table 18, contains bits to generate the UDC OTG interrupt request. Each bit in the UDC Interrupt Status registers is logically ORed together to produce one interrupt request.

Table 18: UDCOTGISR Bit Definitions

Physical Address 0x4060_001C	UDCOTGISR	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	Reserved
24	Read/Write 1 to Clear	IRSF	OTG Set_feature command received
23:17	—	Reserved	Reserved
16	Read/Write 1 to Clear	IRXR	External Transceiver Interrupt Rising Edge
15	Read/Write 1 to Clear	IRXF	External Transceiver Interrupt Falling Edge
14:10	—	Reserved	Reserved
9	Read/Write 1 to Clear	IRVV40R	OTG Vbus Valid 4.0 V Rising Edge
8	Read/Write 1 to Clear	IRVV40F	OTG Vbus Valid 4.0 V Falling Edge

Table 18: UDCOTGISR Bit Definitions (Continued)

Physical Address 0x4060_001C	UDCOTGISR	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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
7	Read/Write 1 to Clear	IRVV44R	OTG Vbus Valid 4.4 V Rising Edge
6	Read/Write 1 to Clear	IRVV44F	OTG Vbus Valid 4.4 V Falling Edge
5	Read/Write 1 to Clear	IRSVR	OTG Session Valid Rising Edge
4	Read/Write 1 to Clear	IRSVF	OTG Session Valid Falling Edge
3	Read/Write 1 to Clear	IRSDR	OTG A-Device SRP Detect Rising Edge
2	Read/Write 1 to Clear	IRSDF	OTG A-Device SRP Detect Falling Edge
1	Read/Write 1 to Clear	IRIDR	OTG ID Change Rising Edge
0	Read/Write 1 to Clear	IRIDF	OTG ID Change Falling Edge

1.5.4 UDC Frame Number Register (UDCFNR)

UDCFNR, shown in Table 19, holds the 11-bit frame number contained in the last received SOF packet. If the SOF interrupt is enabled, a SOF interrupt is generated when the frame-number bits are updated.

Table 19: UDCFNR Bit Definitions (Continued)

Physical Address 0x4060_0014	UDCFNR	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	0

Bits	Access	Name	Description
31:11	—	Reserved	Reserved

Table 19: UDCFNR Bit Definitions (Continued)

Physical Address 0x4060_0014	UDCFNR	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	0

Bits	Access	Name	Description
10:0	R	FN	Frame number associated with last received SOF.

1.5.5 USB Port 2 Output Control Register (UP2OCR)

UP2OCR, shown in Table 22, 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. These bits select several interface options. Only one connection is allowed to each of the ports at a time. Unpredictable behavior occurs if more than one set of I/O is specified for a port. Refer to Section 1.4.8 for more information on OTG interface options. All reserved bits are read as unknown values and must be written with only a 0. A question mark indicates that the value is unknown at reset.

Table 20 shows the legal combinations of USB Port 2 control bit settings.

Table 20: 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	Non-OTG Device
0	0 or 1	3	Off	Non-OTG Host
0	0 or 1	4	Off	External OTG Transceiver Device
0	0 or 1	5	Off	External OTG Transceiver Host
1	0	3	Non-OTG Device	Non-OTG Host
1	0	6	Internal OTG Transceiver Device	External OTG Charge Pump Device
1	1	2	Non-OTG Host	Non-OTG Device

Table 20: Legal Combinations of USB Port 2 Control Bit Settings (Continued)

UP2OCR Control Bits			Differential Port	Single-Ended Port 2
HXOE	HXS	SEOS		
1	1	7	Internal OTG Transceiver Host	External OTG Charge Pump Host
1	0	0	Non-OTG Device	Off
1	1	0	Non-OTG Host	Off

1.5.5.1 USB Host Controller Single-Ended Output Select

Table 21 lists the signals selected for output on the USB alternate function ports for each value of SEOS. A value of SEOS = 0 does not select any outputs. Refer to Section 1.4.8 for more information on the operation of each selection value.

Table 21: 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)	Ext. 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	

Table 22: UP2OCR Bit Definitions

Physical Address 0x4060_0020 **UP2OCR** **UDC**

	31	30	29	28	27	26	25	24	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					SEOS			Reserved						HXOE	HXS	Reserved					IDON	EXSUS	EXSP	DMSTAT	VPMBlockEnbN	DPSTAT	DPPUE	DMPDE	DPPDE	CPVPE	CPVEN
Reset	?	?	?	?	?	0	0	0	?	?	?	0	?	?	0	0	?	?	?	?	?	0	0	0	0	0	1	0	0	0	0	0

Bits	Access	Name	Description
31:27	—	Reserved	Reserved
26:24	Read/Write	SEOS	Single-Ended Output Select Selects the USB data output on the single-ended signals through GPIOs. The GPIO controller must be programmed to select the USB signals. The USB host controller single-ended outputs can be used to provide Wakeups when a Connect/Disconnect or Resume signalling is detected. For information on programming UDC Wakeup events, refer to the Clock Controllers and Power Management chapter in *PXA3xx Processor Family: Vol. I System and Timer Configuration Developers Manual.* 000 = Off. No output selected. 001 = Not used 010 = Device controller single-ended operation using GPIO pads with an external transceiver 011 = Host controller single-ended operation using GPIO pads with an external transceiver 100 = Device controller with external OTG transceiver 101 = Host controller with external OTG transceiver 110 = Device controller with external charge pump 111 = Host controller with external charge pump
23:20	—	Reserved	Reserved
19:18	—	Reserved	Reserved
17	Read/Write	HXOE	Host Port 2 Transceiver Output Enable Enables/disables the USB host controller Port 2 Internal OTG transceiver. When HXOE is cleared, 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. 0 = Internal transceiver disabled. 1 = Internal transceiver enabled

Table 22: UP2OCR Bit Definitions (Continued)

Physical Address 0x4060_0020	UP2OCR	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	DMSTAT	VPMBlockEnbN	DPSTAT	DPPUE	DMPDE	DPPDE	CPVPE	CPVEN
Reset	?	?	?	?	?	0	0	0	?	?	?	0	?	?	0	0	?	?	?	?	?	0	0	0	0	0	1	0	0	0	0	0

Bits	Access	Name	Description
16	Read/Write	HXS	Host Port 2 Transceiver Output Select Selects the internal USB controller that is to use the USB host controller Port 2 as a transceiver. When HXS is cleared, the USB device controller I/O is assigned to the USB host Port 2 transceiver. 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. 0 = Output signals from USB device controller. 1 = Output signals from USB host controller.
15:11	—	Reserved	Reserved
10	Read/Write	IDON	OTG ID Read Enable Allows software 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), then set IDON and sample IRIDR and IRIDF in the UDC Interrupt Status Registers (UDCISR0, UDCISR1, and UDCOTGISR). After reading the OTG ID value, clear IDON to reduce power consumption. Refer to Section 1.4.8.5.
9	Read/Write	EXSUS	External Transceiver Suspend Enable Sets the Suspend-Enable bit that an external transceiver can use to indicate that the USB should enter Suspend.
8	Read/Write	EXSP	External Transceiver Speed Control Sets the Speed-Control signal that can be used by an external transceiver to indicate the speed of the USB connection.
7	Read	DMSTAT	Host Port 2 Transceiver Status of D pin Reads the status of the Port 2 D pin.

Table 22: UP2OCR Bit Definitions (Continued)

Physical Address 0x4060_0020 **UP2OCR** **UDC**

User Settings																														
Bit	31	30	29	28	27	26	25	24	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	DMSTAT	VPMBlockEnbN	DPSTAT	DPPUE	DMPDE	DPPDE	CPVPE	CPVEN
Reset	?	?	?	?	?	0	0	0	?	?	?	0	?	?	0	0	?	?	?	?	?	0	0	0	0	0	1	0	0	0	0	0

Bits	Access	Name	Description
6	Read/Write	VPMBlockEnbN	Host Port 3 Vp/Vm Block Enable_N (control bit for Single-Ended mode) Controls the operation of the USB_P3_6 (VPO) and USB_P3_4 (VMO) ports when the USB is in Transceiverless mode (UP3OCR[CFG] = 0x2). When VPMBlockEnbN is cleared, the USB host Port 3 ignores the VPO and VMO ports when USB_P3_2 (OE_n) is de-asserted, creating the following restrictions: • The USB host Port 3 in Transceiverless mode can be connected only to a full speed (12 Mbps) device • The host cannot recognize a device connect or disconnect through the VMO or VPO ports. Either the device is always connected or an alternative method of determining the connect/disconnect sequence should be used such as a separate GPIO or a read to the external device status register. When VPMBlockEnbN is set, the USB host Port 3 recognizes the VPO and VMO ports when OE_n is de-asserted. The port can be connected to a full-speed (12 Mbps) or slow-speed (1.5 Mbps) device and the host recognizes a connect/disconnect sequence. The host port may not operate correctly if the transceiverless device does not drive a steady state on the VMO and VPO ports when OE_n is de-asserted.
5	Read	DPSTAT	Host Port 2 Transceiver Status of D+ pin Reads the status of the Port 2 D+ pin.

Table 22: UP2OCR Bit Definitions (Continued)

Physical Address 0x4060_0020	UP2OCR	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	DMSTAT	VPMBlockEnbN	DPSTAT	DPPUE	DMPDE	DPPDE	CPVPE	CPVEN
Reset	?	?	?	?	?	0	0	0	?	?	?	0	?	?	0	0	?	?	?	?	?	0	0	0	0	0	1	0	0	0	0	0

Bits	Access	Name	Description
4	Read/Write	DPPUE	Host Port 2 Transceiver D+ Pullup Enable Enables/disables the pullup on D+ pin on the host controller port 2 transceiver as shown in Figure 16, “Connection to External USB Transceiver. When DPPUE is set, SW1 is closed and the D+ pullup is enabled; when DPPUE is cleared, SW1 is open and the D+ pullup is disabled. Set DPPUE when the USB device controller is selected as the OTG I/O for host port 2 output. With this selection, the D+ pin pullups are enabled when the USB OTG is either an A-device or B-device but is acting as the USB device. Clear DPPUE when the USB host controller is selected as the OTG I/O for host port 2 output. With this selection, the D+ pin pullups are disabled when the USB OTG is either an A-device or a B-device but is acting as the USB host controller. Clear DPPUE when OTG is not used. Refer to Table 20, Legal Combinations of USB Port 2 Control Bit Settings, on page 77 for a summary of the pullup-enable programming values.
3	Read/Write	DMPDE	Host Port 2 Transceiver D– Pulldown Enable Enables/disables the pulldown on the D– pin on the host controller Port 2 transceiver as shown in Figure 16, “Connection to External USB Transceiver. When DMPDE is set, SW3 is closed and D-pulldown is enabled; when DMPDE is cleared, SW3 is closed and the D-pulldown is disabled. Set DMPDE when the host port 2 is used to implement an USB OTG interface. With this setting, the D-pulldown is enabled when the USB OTG ports are either an A- or B-device. Clear PDE when the host Port 2 is used as the transceiver for the USB device controller in a non-OTG configuration. With this setting, the D-pulldown is disabled as required for UDC transceiver I/O.

Table 22: UP2OCR Bit Definitions (Continued)

Physical Address 0x4060_0020	UP2OCR	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	DMSTAT	VPMBlockEnbN	DPSTAT	DPPUE	DMPDE	DPPDE	CPVPE	CPVEN
Reset	?	?	?	?	?	0	0	0	?	?	?	0	?	?	0	0	?	?	?	?	?	0	0	0	0	0	1	0	0	0	0	0

Bits	Access	Name	Description
2	Read/Write	DPPDE	Host Port 2 Transceiver D+ Pulldown Enable Enable/disables the pulldown on the D+ pins on the host controller Port 2 transceiver as shown in Figure 16, “Connection to External USB Transceiver. When DPPDE is set, SW3 is closed and the D+ pulldown is enabled; when DPPDE is cleared, SW3 is open and the D+ pulldown is disabled. Set DPPDE when the host Port 2 is used to implement an 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+ pulldowns are enabled when the USB OTG ports are the A-device and are to pulldown D+. Clear PDE when the host Port 2 is used to implement an 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+ pulldown is disabled when the processor USB OTG ports are the 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- pulldown is disabled as required for UDC transceiver I/O.
1	Read/Write	CPVPE	Charge Pump Vbus Pulse Enable Enables the Vbus charge circuitry in an external charge pump device to provide Vbus using a 10 mA current source for the pulsing required by the OTG Session Request Protocol.
0	Read/Write	CPVEN	Charge Pump Vbus Enable Enables an external charge pump device to provide voltage to the USB Vbus.

1.5.6 USB Port 3 Output Control Register (UP3OCR)

UP3OCR contains two bits that control the USB transceivers for Port 3 on the USB host controller to enable use with an external transceiver. The USB host Port 3 Configuration (CFG) bits set the operating configuration for Port 3 of the USB host controller. This port can be used to connect to an external device and/or transceiver in several different configurations. The USB host controller Port 3 inputs provide a Wakeup when resume signalling is detected. USB host controller Port 3 inputs cannot provide a Wakeup when a connect/disconnect is detected. Refer to the “Clocks Controller and Power Management” chapter in *PXA3xx Processor Family Vol. I: System and Timer Configuration Developers Manual* for information on programming UDC Wakeup events.

When CFG is 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 0x1, the USB host controller Port 3 is configured to connect an external device controller directly to Port 2 outputs. This is a special mode of operation called *Passthrough mode* in which the ports are internally routed together and no intervention from either the UHC or UDC occurs. UP2OCR[HXOE] should be cleared for this mode of operation.

When CFG is 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 24 lists the USB host controller Port 3 Configuration selection values and the signals assigned to each GPIO for each setting.

Table 23: USB Host Controller Port 3 Configuration Selection Values

GPIO Alternate Function Port	Port 3 Configuration - UP3OCR[CFG]		
	0x0	0x1	0x2
USB_P3_1	UHC Rx Data (RCV - in)	USB_P2_1	UHC Rx Data (RCV - out)
USB_P3_2	UHC OE (OE_n - out)	USB_P2_2	UHC OE (OE_n - in)
USB_P3_3	UHC Rx D– (VM - in)	USB_P2_3	UHC Rx D- (VM - out)
USB_P3_4	UHC Tx D– (VMO - out)	USB_P2_4	UHC Tx D- (VMO - in)
USB_P3_5	UHC Rx D+ (VP - in)	USB_P2_5	UHC Rx D+ (VP - out)
USB_P3_6	UHC Tx D+ (VPO - out)	USB_P2_6	UHC Tx D+ (VPO - in)

Table 24: USB Host Controller Port 3 Configuration Selection Values

GPIO Alternate Function Port	Port 3 Configuration - UP3OCR[CFG]------			
	0x0	0x1	0x2	0x3
USB_P3_1	UHC Rx Data (RCV - in)	USB_P2_1	UHC Rx Data (RCV - out)	—
USB_P3_2	UHC OE (OE_n - out)	USB_P2_2	UHC OE (OE_n - in)	—
USB_P3_3	UHC Rx D– (VM - in)	USB_P2_3	UHC Rx D- (VM - out)	—
USB_P3_4	UHC Tx D– (VMO - out)	USB_P2_4	UHC Tx D- (VMO - in)	—

Table 24: USB Host Controller Port 3 Configuration Selection Values (Continued)

GPIO Alternate Function Port	Port 3 Configuration - UP3OCR[CFG]------			
	0x0	0x1	0x2	0x3
USB_P3_5	UHC Rx D+ (VP - in)	USB_P2_5	UHC Rx D+ (VP - out)	—
USB_P3_6	UHC Tx D+ (VPO - out)	USB_P2_6	UHC Tx D+ (VPO - in)	—

The register organization and individual bit definitions are shown in Table 25. All reserved bits are read as unknown values and must be written with only a 0. A question mark indicates that the value is unknown at reset.

Table 25: UP3OCR Bit Definitions

Physical Address 0x4060_0024 **UP3OCR** **UDC**

	31	30	29	28	27	26	25	24	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																														CFG	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
31:2	—	Reserved	Reserved
1:0	Read/Write	CFG	Host Port 3 Configuration: 0x0 – Host controller Port 3 transceiver with external host controller 0x1 – Passthrough mode when Client Port 2 connected to host Port 3 0x2 – Host controller Port 3 transceiver with external device controller 0x3 – **Reserved**

1.5.7 UDC Endpoint 0 Control Status Register (UDCCSR0)

UDCCSR0, shown in Table 26, contains 11 bits to operate Endpoint 0 (Control endpoint).

Table 26: UDCCSR0 Bit Definitions

Physical Address 0x4060_0100 **UDCCSR0** **UDC**

	31	30	29	28	27	26	25	24	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																Reserved					ODFCLR	ACM	AREN	SA	RNE	FST	SST	DME	FTF	IPR	OPC
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	Reserved	Reserved
15:11	—	Reserved	Reserved
10	Read/Write	ODFCLR	OPC Disable FIFO Clear 0 = OPC FIFO clear logic is enabled. 1 = OPC FIFO Clear logic is disabled. Controls the OPC FIFO clear logic outlined in Section 1.4.1.1.1, “Back-to-Back Setup Packets”.
9	Read/Write	ACM	ACK Control Mode The ACM bit enables software control of the ACK response to the status in requests of Set_configuration and Set_interface commands. When ACM is set to 0, the UDC automatically responds to the status in request following a Set_configuration and Set_interface with an ACK. When ACM is set to 1, the UDC responds to the status in request following a Set_configuration and Set_interface command with a NAK until AREN is set to 1. When software set 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 software 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, AREN enables software 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 the request following a Set_configuration and Set_interface command with a NAK until AREN is set. When software sets AREN, the UDC responds with an ACK to the next status in request. The UDC clears AREN 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.

Table 26: UDCCSR0 Bit Definitions (Continued)

Physical Address 0x4060_0100 **UDCCSR0** **UDC**

	31	30	29	28	27	26	25	24	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																Reserved					ODFCLR	ACM	AREN	SA	RNE	FST	SST	DME	FTF	IPR	OPC
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
7	Read/Write 1 to Clear	SA	Setup Active Indicates that the current packet in the Endpoint 0 Receive FIFO is part of a USB Setup command. This bit becomes active at the same time as Out-Packet-Complete (OPC). 0 = Setup command is inactive on the USB 1 = Setup command is active on the USB **NOTE:** SA must be cleared when OPC is cleared.
6	R	RNE	Receive FIFO Not Empty: Indicates unread data in the Endpoint 0 Receive FIFO. This bit must be polled when the OPC bit is set to determine whether 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 Forces the UDC to issue a STALL handshake. After the UDC issues a STALL handshake for the current Control transfer, the UDC clears the bit because Endpoint 0 cannot remain in a stalled condition. 0 = UDC cannot remain in stalled condition 1 = Force stall handshake
4	Read/Write 1 to Clear	SST	Sent Stall Set by the UDC when it issues a stall handshake as a result of the FST bit being set. When SST is set, UDCISR0[IR0] is set if the Packet-Complete Endpoint 0 interrupt is enabled. 1 = UDC sent stall handshake
3	Read/Write	DME	DMA Enable For Endpoint 0 Out transactions only, to request DMA reading of Endpoint 0 FIFO memory. The UDC uses DME to control the Endpoint Packet-Complete interrupt and DMA request. If the bit is cleared, 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. It is assumed the data is unloaded using the core, 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 controller of the received data. DMA enable can be disabled only when the DMA channel is stopped. 0 = Send data received interrupt after EOP received. Send data received DMA request after EOP received.

Table 26: UDCCSR0 Bit Definitions (Continued)

Physical Address 0x4060_0100	UDCCSR0	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																Reserved					ODFCLR	ACM	AREN	SA	RNE	FST	SST	DME	FTF	IPR	OPC
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	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 Resets the Endpoint 0 Transmit FIFO. The UDC resets FTF after the FIFO contents are deleted. Also, the UDC flushes the Endpoint 0 Transmit FIFO after it receives 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 is loaded and ready for transmission. Set IPR only when a packet smaller than 16 bytes is 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 is successfully transmitted or FTF is set. When the UDC clears IPR, the IR0[0] bit in UDCISR0 is set if the Packet-Complete Endpoint 0 interrupt is enabled. Software cannot clear IPR. 1 = In packet has been loaded and is ready for transmission
0	Read/Write 1 to Clear	OPC	Out Packet Complete 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 should not be cleared until all the Receive data is 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 is in the data phase of an Endpoint 0 transaction until the OPC bit is cleared. **Note:** OPC is set when DME 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.

1.5.8 UDC Endpoints A–X Control Status Registers (UDCCSRA–UDCCSRX)

UDCCRSA–UDCCRSX, shown in Table 28, are used by each of the 23 programmable Endpoints A–X to control the behavior of the endpoint and to report status for that endpoint after USB enumeration. The UDC registers for each endpoint contain 10 bits to operate their respective endpoint.

Table 27 summarizes how the UDCCRSA–UDCCRSX bits behave according to endpoint direction.

Table 27: UDCCRSA–UDCCRSX Bit Definition by Endpoint Direction

Register Bit	In Endpoint	Out Endpoint
HBNE/HBNF	Hidden Transmit buffer full/not full	Hidden Receive buffer empty/not empty
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

Table 28: UDCCRSA–UDCCRSX Bit Definitions

Physical Addresses **UDCCRSA–UDCCRSX** **UDC**
0x4060_0104–0x4060_015C

User Settings																														
Bit	31	30	29	28	27	26	25	24	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																					HBNE/HBNF	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:11	—	Reserved	Reserved

Table 28: UDCCRSA–UDCCRSX Bit Definitions (Continued)

Physical Addresses **UDCCRSA–UDCCRSX** **UDC**
0x4060_0104–0x4060_015C

	31	30	29	28	27	26	25	24	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																					HBNE/HBNF	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
10	Read	HBNE/HBNF	Hidden Buffer Not Empty/Hidden Buffer Not Full Indicates unread or untransmitted data in the second Receive or Transmit buffer of a double-buffered endpoint. For a double-buffered In endpoint, HBNE/HBNF means the second buffer not full; for a double-buffered Out endpoint, HBNE/HBNF means the second buffer not empty. For a single-buffered endpoint, HBNE/HBNF are reserved. For a double-buffered In endpoint, HBNE/HBNF clears when the endpoint first buffer space is filled with packet data and software has access to the second buffer space only. HBNE/HBNF is set when the contents of the endpoint first buffer space are sent. BNE/BNF indicates the status of the buffer being loaded. If both buffers are loaded with complete packets of data, the BNE/BNF, HBNE/HBNF and FS bits are cleared. A complete packet of data is signified by either loading a maximum size packet or loading a short packet and setting the SP bit. HBNE/HBNF and BNE/BNF can be used to determine the current status of both buffers. HBNE/HBNF indicates the status of the first buffer filled and BNE/BNF indicates the status of the current buffer. For a double-buffered Out endpoint, HBNE/HBNF is set when the endpoint second buffer space is filled with packet data received from the host. The UDC sets BNE/BNF when the Receive FIFO contains one complete data packet and HBNE/HBNF when the Receive FIFO contains two complete data packets. For a Bulk or Interrupt endpoint, HBNE/HBNF is not set until the UDC error-checks the Out packet and finds it to be error-free. For an Isochronous endpoint, HBNE/HBNF is not set until the UDC error-checks the Out packet data and DPE is set to indicate any errors found. A complete packet is defined as a Maximum Size packet, a Short packet, or a Zero packet. HBNE/HBNF clears when the first packet of received data is read and the second packet of data is transferred to the first packet. For double-buffered In endpoints (Buffer Not Full): 0 = Hidden Transmit buffer is full 1 = Hidden Transmit buffer is not full For double-buffered Out endpoints (Buffer Not Empty): 0 = Hidden Receive buffer is empty 1 = Hidden Receive buffer is not empty

Table 28: UDCCRSA–UDCCRSX Bit Definitions (Continued)

Physical Addresses UDCCRSA–UDCCRSX UDC
0x4060_0104–0x4060_015C

	31	30	29	28	27	26	25	24	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																					HBNE/HBNF	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
9	Read	DPE	Data Packet Error Used only for endpoints configured as Isochronous Out endpoints. DPE indicates the detection of a PID, bit-stuffing, or CRC error on the active buffer of Isochronous OUT data. 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 Resets the Endpoint Transmit or Receive FIFO by setting it. The UDC clears FEF after the FIFO contents are deleted. For a double-buffered IN endpoint, setting FEF only flushes the data in the active Transmit buffer. The data in the USB buffer remains unchanged and continues to be treated as valid. The next bulk IN command from the host initiates the transmission of data that is supposed to be flushed through the previous FEF bit set. This is not the preferred behavior when FEF is set on a particular endpoint. There is no clean workaround for this issue. The available workarounds are as follows: • Use only single-buffered endpoints, which results in additional NAKs by the UDC. • Direct the host to perform a dummy Bulk In transfer to empty the old data from the UDC USB Tx buffer. • Do not use the FEF bit to flush the UDC Endpoint FIFOs. None of these workarounds is ideal, especially because the USB host has no clear way to know when the UDC is issued a FIFO Flush request and therefore does not know when Bulk IN data may need to be discarded. For In endpoints: 1 = Flush the contents of the Transmit FIFO For Out endpoints: 1 = Flush the contents of the Receive FIFO

Table 28: UDCCRSA–UDCCRSX Bit Definitions (Continued)

Physical Addresses **UDCCRSA–UDCCRSX** **UDC**
0x4060_0104–0x4060_015C

	31	30	29	28	27	26	25	24	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																					HBNE/HBNF	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
7	In endpoints: Read/Write 1 to Set Out Endpoints: Read	SP	Short Packet Control/Status Indicates that a packet smaller than the maximum packet size is received and ready for unloading, or it is loaded and is ready for transmission. For an In endpoint, SP must be set when the last byte of a transfer that is shorter than the maximum packet size is loaded into the Transmit buffer. This notifies the UDC that the packet is ready for transmitting. For Maximum Size packets, SP does not need to be set. When the data packet successfully transmits, the UDC clears SP. For an Out endpoint, when the UDC sets SP, it indicates the last byte of a Receive-Data transfer that is smaller than the maximum size packet is loaded into the Receive FIFO. If SP is set and BNE/BNF is clear, a Zero-Length packet is received. The SP bit is read-only for all Out endpoints and is valid while the PC bit is set. For In endpoints: 1 = Short packet ready for transmission For Out endpoints: 1 = Short packet received and ready for reading

Table 28: UDCCRSA–UDCCRSX Bit Definitions (Continued)

Physical Addresses **UDCCRSA–UDCCRSX** **UDC**
0x4060_0104–0x4060_015C

	31	30	29	28	27	26	25	24	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																					HBNE/HBNF	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
6	Read	BNE/BNF	Buffer Not Empty/Buffer Not Full Indicates there is unread or untransmitted data in the current Receive or Transmit buffer. For an In endpoint, BNE/BNF means the buffer is not full; for an Out endpoint, BNE/BNF means the buffer is not empty. For an In endpoint, BNE/BNF clears when the Endpoint buffer space is filled with packet data. If the endpoint does not have double-buffering enabled, BNE/BNF clears when one complete packet of data is loaded into the FIFO memory space. For an In endpoint with double-buffering enabled, BNE/BNF indicates the status of the buffer being loaded. If both buffers are loaded with complete packets of data, both the BNE/BNF and FS bits are cleared. A complete packet of data is signified by either loading a maximum size packet or loading a short packet and setting the SP bit. For an Out endpoint, BNE/BNF must be polled when the PC bit is set to determine whether there is data in the FIFO that must be unloaded. 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 set until the UDC error-checks the Out packet and finds it to be error-free. For an Isochronous endpoint, BNE/BNF is set until the UDC error-checks the Out packet data and DPE is set to indicate any errors 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 completes. If the endpoint has double-buffering enabled, BNE/BNF clears when the current buffer is empty; the FS bit does not clear until neither buffer has data. 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

Table 28: UDCCRSA–UDCCRSX Bit Definitions (Continued)

Physical Addresses 0x4060_0104–0x4060_015C **UDCCRSA–UDCCRSX** **UDC**

	31	30	29	28	27	26	25	24	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																					HBNE/HBNF	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
5	Read/Write	FST	Force Stall Used by endpoints configured as Bulk or Interrupt endpoints only. FST is considered reserved for all endpoints configured as Isochronous endpoints.Set the FST bit to force the UDC to issue a Stall handshake in response to all In token requests or all Out tokens. FST is cleared and the SST bit is set when the Stall handshake is actually 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 USB host controller clears the actual stall condition in the UDC using the Clear_feature command. 1 = Issue Stall handshakes to all in tokens
4	Read/Write 1 to Clear	SST	Sent Stall Used by endpoints configured as Bulk or Interrupt endpoints only. SST is 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. SST is cleared by writing a 1 to it. For an In endpoint, the Transmit FIFO is flushed when SST is cleared. For 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:** Clear SST before the USB host controller requests more data from In endpoints or invalid or corrupted data may be sent to the host. Set the FST bit to force the UDC to issue a Stall handshake in response to all In token requests or all Out tokens. The FST bit is cleared and the SST bit is set when the Stall handshake is actually 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 USB host controller clears the actual stall condition in the UDC using the Clear_feature command. 1 = Stall handshake was sent

Table 28: UDCCRSA–UDCCRSX Bit Definitions (Continued)

Physical Addresses **UDCCRSA–UDCCRSX** **UDC**
0x4060_0104–0x4060_015C

	31	30	29	28	27	26	25	24	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																					HBNE/HBNF	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
3	Read/Write	DME	DMA Enable Set to request DMA reading or writing of Endpoint FIFO memory. For all Out endpoints, the UDC uses DME to control the output of the Endpoint Packet-Complete interrupt. If DME is cleared, the Packet-Complete interrupt is asserted when the End-of-Packet is received, and all received data is still in the Receive FIFO; data must be unloaded using the processor. If DME is set, the Packet Complete interrupt is not asserted, but a DMA request is generated to notify the DMA controller of the received data. For an In endpoint, the UDC uses DME to enable the output of the DMA request. If DME is cleared, the DMA request is not generated, and software must use BNE/BNF and FS to determine when to load data into the Endpoint FIFO memory for transmission. If DME 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 is filled or the SP bit is set. When DME is set, the PC bit is set at the end of the transmission of the packet; however, it is not necessary to clear PC before loading more data into the Endpoint FIFO. **Note:** DME should be disabled only when the DMA channel is stopped. 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 28: UDCCRSA–UDCCRSX Bit Definitions (Continued)

Physical Addresses **UDCCRSA–UDCCRSX** **UDC**
0x4060_0104–0x4060_015C

	31	30	29	28	27	26	25	24	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																					HBNE/HBNF	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	Read/Write 1 to Clear	TRN	Tx/Rx NAK 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. TRN is cleared by writing a 1 to it. For In endpoints, the UDC sets TRN when the UDC tries to transmit data but a complete packet of data is not 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. If the FIFO-Error interrupt is enabled, the UDC generates a FIFO-Error interrupt for the endpoint. For Out endpoints, the UDC sets TRN when the host sends OUT data while the Endpoint buffer 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 remains full. If the Out endpoint is configured as an Isochronous endpoint, isochronous data packets are sent from the host while the full Endpoint buffers are dropped. If the FIFO-Error interrupt is enabled, a FIFO-Error interrupt is generated for the endpoint. 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.

Table 28: UDCCRSA–UDCCRSX Bit Definitions (Continued)

Physical Addresses **UDCCRSA–UDCCRSX** **UDC**
0x4060_0104–0x4060_015C

	31	30	29	28	27	26	25	24	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																					HBNE/HBNF	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
1	Read/Write 1 to Clear	PC	Packet Complete Set by the UDC when an entire packet is sent to or received from the USB host controller. PC can be used to validate the other status/error bits in UDCCRSA-UDCCRSX. The PC bit is cleared by writing a 1 to it. For an Out endpoint, the UDC sets PC when an entire packet is received from the USB host controller. The UDC generates a Packet-Complete interrupt or DMA request for the endpoint according to the Packet-Complete Interrupt-Enable and DME bits. After all the received data is read 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 buffer(s) are full. For an In endpoint, the UDC sets the PC bit after an entire packet is sent to the USB host controller. Software can load more data into the Transmit FIFO regardless of the state of PC. When the UDC receives a packet, it sets PC, 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 is set when the UDC receives a packet, PC remains set and the corresponding PC interrupt bit in the UDCISR0 is set (if it is not already set) and an interrupt is generated if the interrupt is enabled and DME is clear. If the PC bit is set when the UDC receives a packet, PC remains set and if DME is set, a DMA request is generated. 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

Table 28: UDCCRSA–UDCCRSX Bit Definitions (Continued)

Physical Addresses 0x4060_0104–0x4060_015C	UDCCRSA–UDCCRSX	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																					HBNE/HBNF	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
0	Read	FS	FIFO Needs Service Indicates that there is room in the Endpoint FIFO for more data to be loaded or more data in the Endpoint FIFO to be unloaded. For an In endpoint with 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 is cleared 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 is cleared when one complete data packet is loaded into the FIFO. A complete packet of data is signified by either loading in a maximum size packet or by setting SP. For an Out endpoint with double-buffering enabled, the UDC sets the FS bit 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. If both buffers are empty, FS is not set until the UDC checks the out packet for errors. If double-buffering is disabled, the UDC sets FS when the Receive FIFO has one complete data packet in it. FS does not clear until all data is read from the FIFO. A complete packet can be the Maximum Size packet, a Short packet, or a Zero packet. 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

1.5.9 UDC Byte Count Registers (UDCBCR0 and UDCBCRA–UDCBCRX)

UDCBCR0 and UDCBCRA–UDCBCRX, shown in Table 29, 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.

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 UDC clears the BNE/BNF bit in the Endpoint Control/Status register 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 data is still in the Endpoint FIFO space. The BNE/BNF bit is set when the second buffer is the active Endpoint buffer, but it 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 unknown data, and the second buffer is not read. The PC bit in the Endpoint Control/Status register must be cleared so that the following tasks can be completed:

- 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.
- Enable the second buffer data for reading.

When the UDC unit receives a Status packet followed by a Setup packet, the Endpoint0 Byte Count register does *not* read 0 if the register is read before the OPC bit for the Status packet is cleared. For this instance, the Byte Count register contains the number of bytes for the subsequent Setup packet.

See Section 1.4.1.1.1, “Back-to-Back Setup Packets” for Byte Count values during back-to-back setup scenarios.

Note

The Byte Count registers can be ignored when DMA is used to move data (DME = 1)

Table 29: UDCBCR0 and UDCBCRA–UDCBCRX Bit Definitions

	Physical Addresses 0x4060_0200–0x4060_025C																															
	UDCBCR0, UDCBCRA–UDCBCRX																															
	UDC																															
User Settings																																
Bit	31	30	29	28	27	26	25	24	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	Reserved
9:0	Read	BC	Byte Count Updated by the UDC after each byte is read by software. Upon receiving an interrupt that indicates the endpoint has data or after the PC bit in the corresponding UDCCSR is 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.

1.5.10 UDC Endpoint Data Registers (UDCDR0 and UDCDRA–UDCDRX)

UDCDR0 and UDCDRA–UDCDRX are 32-bit by maximum packet size-entry bidirectional FIFOs. 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 sends data to the USB host controller, the data must be placed into the appropriate UDC Endpoint register. Although Read and Write operations can be performed on a single FIFO at various points in a control sequence, the FIFO cannot be read and written at the same time. The UDC controls the direction that the FIFO is flowing. Refer to Section 1.4.3 for details on accessing the Endpoint FIFO memory.

For Endpoint 0, the UDC is normally in an Idle state, waiting for the USB host controller to send commands. When a command is sent, the UDC fills the Endpoint 0 Receive FIFO with the command from the host, and the command is read from the FIFO when it arrives. The only time software can write to the Endpoint 0 Transmit FIFO is when it receives a Get_descriptor, vendor, or class-specific command from the host that requires a transmission in response.

With programmable Endpoints A–X, for a Bulk, Interrupt, or Isochronous In endpoint, data can be loaded through DMA or direct processor Writes. Up to two data packets can be loaded for transmission If double-buffering is enabled.

Note

Data registers can be ignored when DMA is used to move data (DME = 1).

Table 30: UDCDR0 and UDCDRA–UDCDRX Bit Definitions

Physical Addresses 0x4060_0300–0x4060_035C	UDCDR0 and UDCDRA–UDCDRX	UDC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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: Write Only Out Endpoints: R Endpoint 0: Read/Write	ED	Endpoint Data

1.5.11 UDC Endpoint A–X Configuration Registers (UDCCRA–UDCCRX)

UDCCRA–UDCCRX, shown in Table 31, 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. Software determines the maximum packet size, endpoint number, type, direction, and buffering for each endpoint. Software selects which endpoints are used for each configuration and interface, and it allocates the FIFO space as needed.

The UDC is disabled at powerup and all configurations, interfaces, and alternate interface settings are disabled. Software must program the Endpoint Configuration registers before enabling the UDC. When the UDC is enabled, the endpoint configuration information is checked and, if valid, is loaded into the USB interface and enabled for USB operation. The UDC can then be enumerated and configured by the USB host controller. When the UDC is 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.

Table 31: UDCCRA–UDCCRX Bit Definitions

Physical Addresses UDCCRA–UDCCRX UDC
0x4060_0404–0x4060_045C

	31	30	29	28	27	26	25	24	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					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	Reserved
26:25	If UDE=*0*: Read/Write If UDE=*1* Read	CN	Configuration Number Must be set before the UDC is enabled. Each Endpoint A–X can be assigned to Configuration 1, 2, or 3. Each can be assigned to only one configuration/interface/alternate interface setting combination, and it is active only when the USB host controller configures the UDC for that configuration, interface, and alternate-interface setting.
24:22	If UDE=*0*: Read/Write If UDE=*1* Read	IN	Interface Number Must be set before the UDC is enabled. Each Endpoint A–X can be assigned to Interface 1–7. Each E can be assigned to only one configuration/interface/alternate interface setting combination, and it is active only when the USB host controller configures the UDC for that configuration, interface, and alternate-interface setting.
21:19	If UDE=*0*: Read/Write If UDE=*1* Read	AISN	Alternate Interface Setting Number Must be set before the UCD is enabled. Each Endpoint A–X can be assigned to an alternate Interface setting 1–7. Each Endpoint A–X can be assigned only to one configuration/interface/alternate interface setting combination, and it is active only when the USB host controller configures the UDC for that configuration, interface, and alternate-interface setting.
18:15	If UDE=*0*: Read/Write If UDE=*1* Read	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, 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.
14:13	If UDE=*0*: Read/Write If UDE=*1* Read	ET	USB Endpoint Type 11 = Interrupt 10 = Bulk 01 = Isochronous 00 = Not used

Table 31: UDCCRA–UDCCRX Bit Definitions (Continued)

Physical Addresses UDCCRA–UDCCRX UDC
0x4060_0404–0x4060_045C

	31	30	29	28	27	26	25	24	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					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*: Read/Write If UDE=*1* Read	ED	USB Endpoint Direction Determines whether the endpoint is 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*: Read/Write If UDE=*1* Read	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*: Read/Write If UDE=*1* Read	DE	Double-buffering Enable Allocates two Transmit or Receive buffers in the Endpoint memory. If an endpoint has DE set, it 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 completes. If an Out endpoint is double-buffered, endpoint data can be unloaded from the first buffer while the UDC receives data and loads 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 1.4.3 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*: Read/Write If UDE=*1* Read	EE	Endpoint Enable Enables 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 it is not used during USB operation. 0 = Endpoint is disabled for USB operation 1 = Endpoint is enabled for USB operation

2 Universal Serial Bus 2.0 Device Controller

This chapter describes the Universal Serial Bus 2.0 Device Controller, or U2DC, found in the processor. This chapter describes the endpoints, interrupts to the Intel XScale® core, the dedicated DMA engine, and the Transmit/Receive FIFO interface of the U2DC. Additional information on various modes of operation are also explained. Software should have a working knowledge of the USB standard.

The U2DC supports both high-speed and full-speed modes. Throughout this chapter, features that pertain only to high-speed are followed by the addition of an “(HS)” tag and do not apply to full-speed operation.

The PXA32x processor and PXA30x processor uses the industry standard Universal Transceiver Macrocell Interface (UTMI), Version 1.05, as the interface between the U2DC and the transceiver. The PXA31x processor uses the industry standard UTMI+ Low Pincount Interface (ULPI) as the interface between the U2DC and the transceiver. Most of the following discussion is kept as generic as possible to cover both processors. A section of this chapter is dedicated for a description of the ULPI interface and any additional features or limitations associated with it.

2.1 PXA3xx Processor Differences

Table 32 shows the USB 2.0 Device Controller differences among the PXA32x, PXA31x, and PXA30x processors. Refer to each individual register for other operating differences.

Table 32: PXA3xx Processors Feature Differences

Feature	PXA30x	PXA31x	PXA32x
Phy interface	UTMI	ULPI	UTMI
Endpoints	7 (uni-directional)[1]	15 (uni-directional)[2]	7 (uni-directional)[1]
SRAM size	8KB	16KB	8KB
DMA Channels	8 (4 sets of 2)	16 (4 sets of 4)	8 (4 sets of 2)
OTG Capable	No (Only with UDC)	Yes	No (Only with UDC)
Carkit mode	Not Supported	Supported	Not Supported

1. Endpoints A - G
2. Endpoints A - P

2.2 Features

- Adheres to current USB industry specification
- Supports both high-speed and full-speed modes
- Automatically decodes most standard device requests without software intervention
- Supports programmable Bulk, Isochronous, and Interrupt endpoint types
- Programmable maximum packet size
- Up to 15 programmable configurations, including 1 default configuration for Endpoint 0
- Automatic token generation
- Automatic CRC checking

- USB OTG support (PXA31x processor only)
- Software access to the immediate ULPI PHY register space (PXA31x processor only)
- Supports 3-pin or 6-pin ULPI serial modes (OTG FS/LS Host only) (PXA31x processor only)
- Support for Carkit mode (UART data) (PXA31x processor only)
- OTG Interrupts for Vbus Valid, Session End, Session Valid, and ID change (PXA31x processor only)

2.3 Limitations

- No support for additional control endpoints.
- UTMI interface supports 8-bit mode only .
- The USB specification allows for physical endpoints to be addressed by both the 4-bit endpoint address and the endpoint direction. For example, Endpoint 3 can be used as an ISO Out endpoint as well as an unrelated Bulk In endpoint. There is no requirement for this use, but it is allowed. In U2DC, this option is under software control but the U2DC currently does not decode based on direction, only on endpoint number. If software were to enable two endpoints with the same endpoint number but different directions, internal logic for both endpoints would respond, resulting in corrupted data.
- A zero-length Isochronous or Control Out transfer with a CRC error is ignored. It is not stored in the FIFO, nor do the U2DCSR bits reflect that a packet was received.
- The ULPI interface does not support extended register access to ULPI PHY registers nor does it support “Guaranteed Link transactions” as described in the ULPI specification (PXA31x processor only).

2.4 Overview

The U2DC consists of six major components: the system bus interface, a dedicated DMA controller, configuration controller, endpoint memory, endpoint control, and USB interface. The system bus interface provides the interface between the processor and the U2DC. The DMA controller coordinates and controls memory accesses through the system bus interface. The configuration controller contains the U2DC Control and Status registers for the Endpoint Configuration data.

The U2DC uses single-ported memory to support FIFO operations. Bulk, Isochronous, and Interrupt Endpoint FIFO structures enable the endpoint to process one packet while assembling another. Depending on the amount of memory programmed to the FIFO, a single packet or many packets of data may be buffered for each endpoint in an effort to compensate for the amount of latency encountered on the system bus. Control endpoints use only a single packet at a time and complete the controls required by one packet before implementing the control required from the next packet. The dedicated DMA is used to transfer data to and from the Data Endpoint FIFOs, while the core is used to transfer data to and from the Control endpoint. Depending on the transfer method, an interrupt, DMA service request, or polling can be used to detect packet receipt.

Figure 19 is a block diagram of the U2DC for the PXA32x processor and PXA30x processor. Figure 20 is a block diagram of the U2DC for the PXA31x processor.

Figure 19: PXA32x Processor and PXA30x Processor U2DC Block Diagram

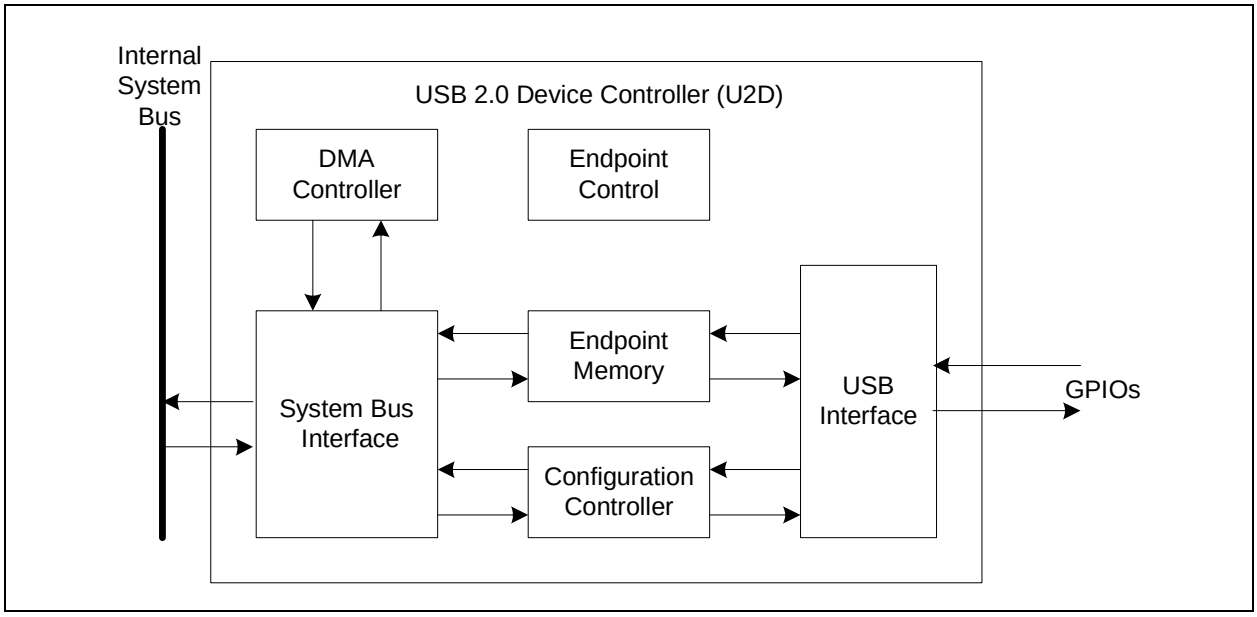
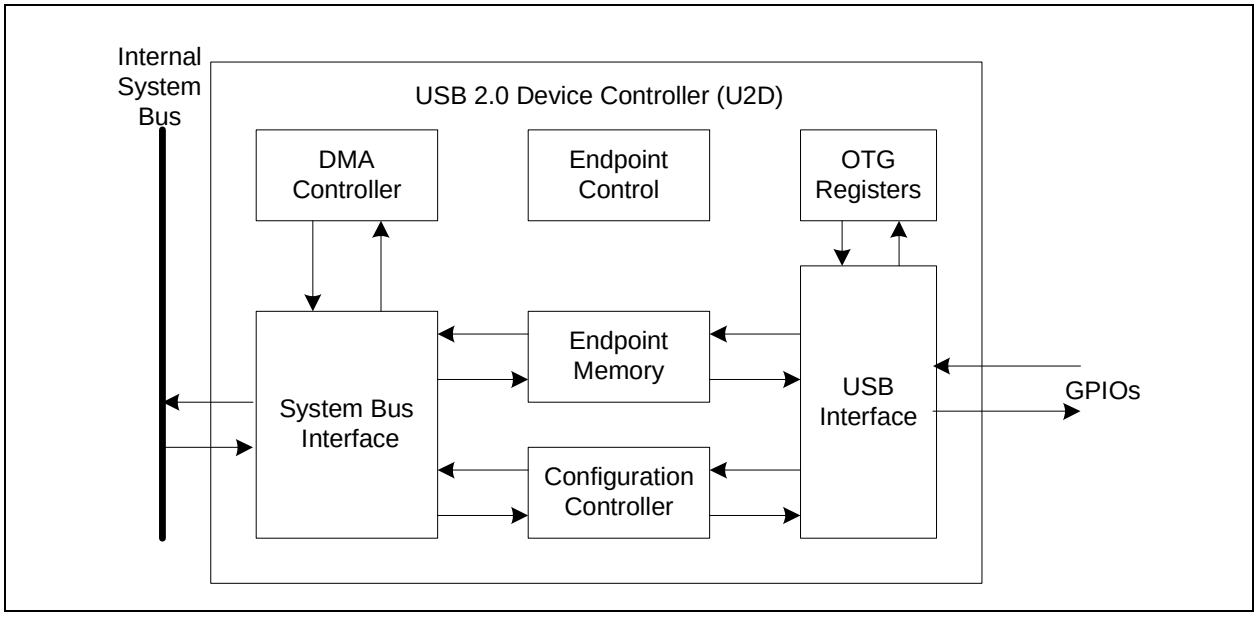


Figure 20: PXA31x Processor U2DC Block Diagram



2.5 PXA32x Processor and PXA30x Processor UTMI Signal Descriptions

This section describes the UTMI signals (See Table 33). All signals are synchronous to the transceiver clock. The UTMI data bus is a bi-directional bus controlled by the state of the Tx_valid signal.

Table 33: UTMI Signals Summary

Name	Type	Width	Description
GPIO			
Any GPIO that can be programmed for wakeup events	input	1	GPIO pin is required for detecting the USB cable attach/detach event. See Section 2.7.4, "PXA32x processor and PXA30x processor Cable Attach and Detach" for more details
utm_clk	input	1	Clock. This input from the PHY is used for clocking Receive and Transmit parallel data. 60 MHz HS/FS, with 8-bit interface
utm_reset	output	1	Reset. Reset all state machines in the UTM.
utm_xcvr_select	output	1	Transceiver Select. This signal selects between the FS and HS transceivers: 0 : HS transceiver enabled 1 : FS transceiver enabled
utm_term_select	output	1	Termination Select. This signal selects between the FS and HS terminations: 0 : HS termination enabled 1 : FS termination enabled
utm_suspendm_x	output	1	Suspend. Places the external PHY in a mode that draws minimal power from supplies. Shuts down all blocks not necessary for Suspend/Resume operation. While suspended, TermSelect must always be in FS mode to ensure that the 1.5 KOhm pullup on DP remains powered. 0 : PHY circuitry drawing Suspend current 1: PHY circuitry drawing normal current

Table 33: UTMI Signals Summary (Continued)

Name	Type	Width	Description
utm_linestate	input	2	Line State. These signals reflect the current state of the single ended receivers. They are combinational until a "usable" CLK is available, then they are synchronized to CLK. They directly reflect the current state of the DP (LineState[0]) and DM (LineState[1]) signals: DM DP Description 0 0 0: SE0 0 1 1: 'J' State 1 0 2: 'K' State 1 1 3: SE1
utm_opmode	output	2	Operational Mode. These signals select between various operational modes: [1] [0] Description 0 0 0: Normal Operation 0 1 1: Non-driving 1 0 2: Disable Bit Stuffing and NRZI encoding 1 1 3: **reserved**
utm_data<7:0>	Bidirectional	8	PHY Data. 8-bit parallel USB data bus to the external PHY.
utm_txvalid	output	1	Transmit Valid. Indicates that the DataIn bus is valid.
utm_txready	input	1	Transmit Data Ready. PHY signal indicating transmit data ready
utm_rxvalid	input	1	Receive Data Valid. Indicates that the DataOut bus has valid data.
utm_rxactive	input	1	Receive Active. Indicates that the receive state machine has detected SYNC and is active. RXActive is negated after a Bit Stuff Error or an EOP is detected.
utm_rxerror	input	1	Receive Error 0 = no error 1 = receive error has been detected This output is clocked with the same timing as the DataOut lines and can occur at anytime during a transfer. If asserted, it forces the negation of RXValid on the next rising edge of CLK.

2.6 PXA31x Processor ULPI Signal Descriptions

This section describes the interface signals of the ULPI interface (See Table 34).

Table 34: ULPI Signals Summary

Name	Type	Width	Description
ulpi_clk	input	1	Clock. This input from the PHY is used for clocking Receive and Transmit parallel data. 60 MHz HS/FS, with 8-bit interface (*Note that the ulpi_clk and the utm_clk are the same)
ulpi_dir	input	1	Controls the direction of the data bus. When the PHY has data to transfer to the link, it drives ulpi_dir high to take ownership of the bus. When the PHY has no data to transfer it drives ulpi_dir low and monitors the bus for commands from the link. The PHY pulls ulpi_dir high whenever the interface cannot accept data from the link, such as during PLL startup.
ulpi_nxt	input	1	The PHY asserts ulpi_nxt to throttle the data. When the link is sending data to the PHY, ulpi_nxt indicates when the current byte has been accepted by the PHY. The link places the next byte on the data bus in the following clock cycle.
ulpi_data<7:0>	Bidirectional	8	PHY Data. 8-bit data bus for the external PHY
ulpi_stp	output	1	The link asserts ulpi_stp for one clock cycle to stop the data stream currently on the bus. If the link is sending data to the PHY, ulpi_stp indicates the last byte of data was on the bus in the previous cycle.

2.7 Operation

The U2DC consists of six major components: the System Bus interface, dedicated DMA controller, configuration controller, Endpoint memory, Endpoint control, and USB interface. The section covers these components.

2.7.1 System Bus Interface

The U2DC is a peripheral device that is connected to the internal system bus. All software-initiated accesses to the U2DC registers and Endpoint memory are completed using the internal system bus. Also, all DMA transactions are completed through the internal system bus.

2.7.2 DMA Controller

The U2DC contains a Direct-Memory Access controller (U2DMA) that transfers data to and from memory in response to requests generated by the internal Endpoint FIFOs of the U2DC. The internal Endpoint FIFOs do not directly supply addresses and commands to the memory controller. Instead, the states required to manage a data stream are maintained in DMA channels in the U2DMA. Every DMA request from an Endpoint FIFO generates a system-bus transaction.

The U2DMA provides compatibility with the main processor DMA controller with respect to Descriptor Fetch operation for memory-to-peripheral (Endpoint FIFO) and peripheral (Endpoint FIFO)-to-memory transfers. The U2DMA does not implement no-Descriptor Fetch operation, memory-to-memory transfers, external requests, Descriptor Compare modes, or programmable data burst sizes and memory widths.

- Supports data transfers for U2DC Endpoint FIFOs. Supported types are FIFO-to-memory and memory-to-FIFO transfers.
- Supports aligned and unaligned transfers to and from system memory.
- Supports 8 channels and 8 Endpoint FIFO requests (PXA32x processor and PXA30x processor only)
- Supports 16 channels and 16 Endpoint FIFO requests (PXA31x processor Only)
- Supports up to 4 outstanding memory accesses simultaneously. These accesses can be from a single channel or multiple channels.
- Employs a priority mechanism to process active channels (four channels with outstanding DMA requests, at any given time).
- Allows each of the channels to operate with Descriptor-Fetch transfers (see Section 2.7.2.2).
- Retrieves trailing bytes in the Receive Endpoint FIFOs.
- Supports 1 KB transfer size to support maximum packet size for USB transfers. DMA transfer size cannot be greater than the USB packet size.
- Descriptor setup compatible with the processor DMA controller.

Note

Core accesses to the FIFO associated with Endpoint 0 (U2DC End Point 0 Data Register (U2DDR0)) are allowed. Core accesses to FIFOs associated with (U2DC Endpoint Configuration Registers (U2DCRx) are not allowed. Use the DMA to access U2DDRx registers.

2.7.2.1 U2DMA Operation

The U2DMA transfers data using flowthrough transactions to the U2DC Endpoint FIFOs. The U2DMA has up to 16 configurable channels. Figure 21 is a block diagram of the U2DMA for the PXA32x processor and PXA30x processor. Figure 22 is a block diagram of the U2DMA for the PXA31x processor. Refer to Section 2.10 for a description of the registers referenced in the block diagram and accompanying text.

Figure 21: PXA32x Processor and PXA30x Processor U2DMA Block Diagram

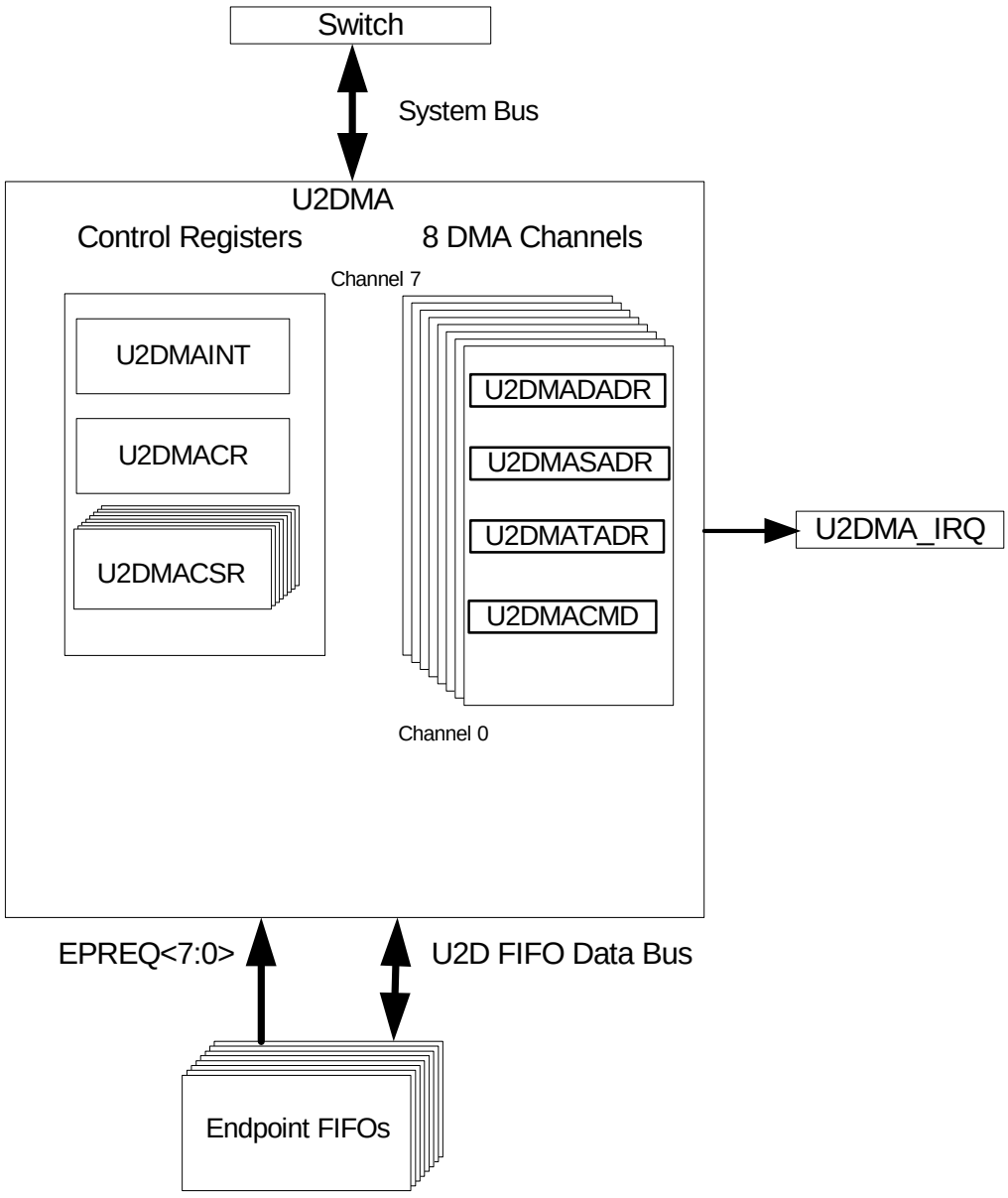
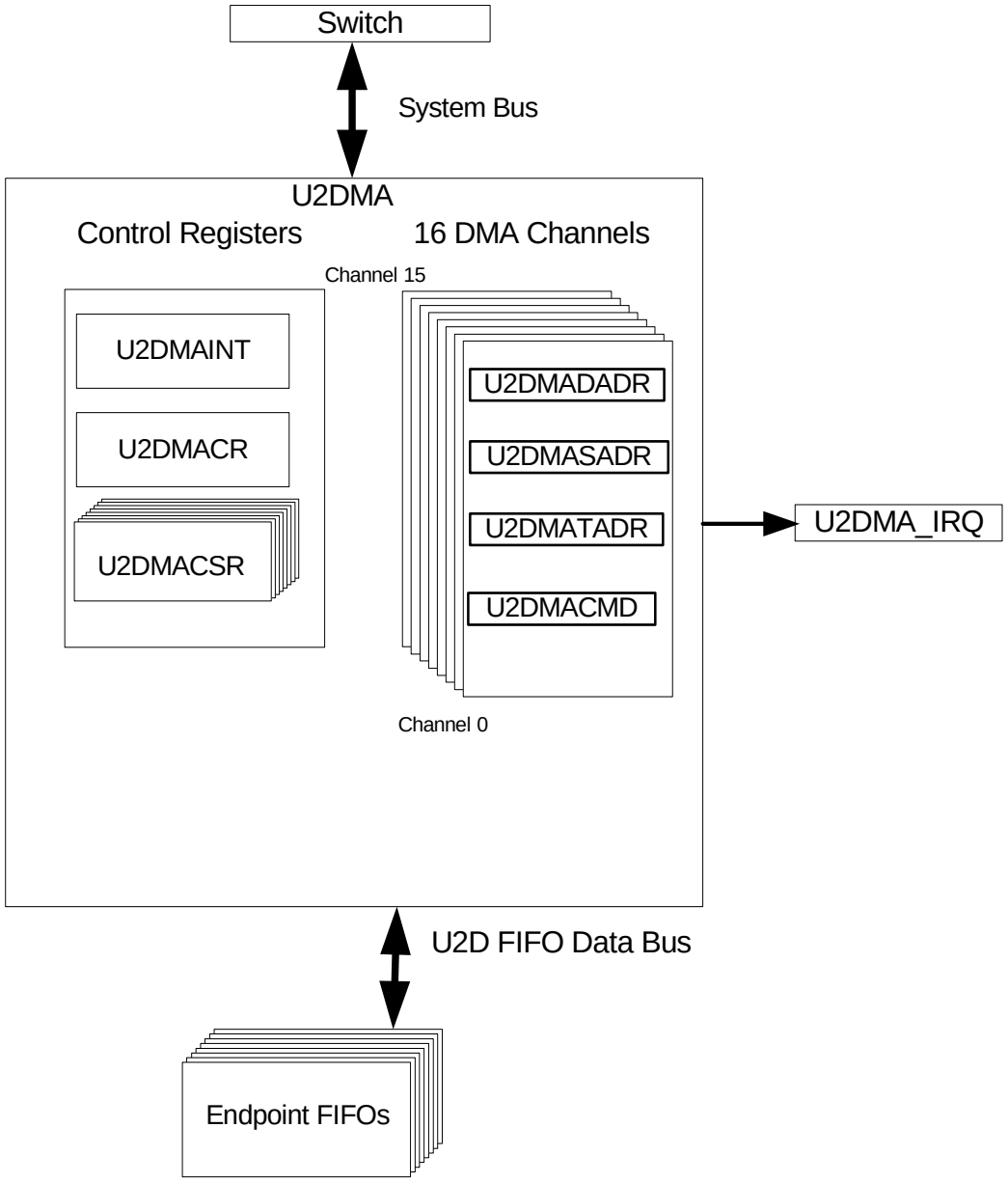


Figure 22: PXA31x Processor U2DMA Block Diagram



2.7.2.1.1 DMA Channels

Each channel of the DMA is controlled by four 32-bit Descriptor registers. In addition, the DMA contains a 32-bit Control/Status register for each channel. Each channel can be configured to service either a memory-to-FIFO or FIFO-to-memory transfer. Each channel is serviced in increments of the burst size (32 bytes) and delivered in the granularity of the FIFO port width (64

bits). Each channel is serviced with a burst of data when multiple channels are actively executing. The DMA controller performs a context switch to another active channel after each burst of data. The DMA controller performs context switches based on whether a channel is active, whether its target FIFO is currently requesting service, and the channel priority.

The U2DMA differs from the processor system DMA in that it cannot perform memory-to-memory data transfers. The only possible data transfers supported by the U2DMA are from system memory to Endpoint FIFOs or from Endpoint FIFOs to system memory. The Descriptor registers, Interrupt register (U2DMAINT), and Control/Status registers of the U2DMA are compatible with the system DMA.

2.7.2.1.2 DMA Channel Priority Scheme

The DMA channel priority scheme helps to ensure that Endpoint FIFOs are to be serviced according to their bandwidth requirements. Assign a high priority to critical endpoints (such as Isochronous endpoints) or endpoints with high bandwidth requirements and a lower priority to endpoints with lower criticality and bandwidth requirements. This approach ensures that critical and high bandwidth endpoints are serviced more often than low bandwidth endpoints and have a reduced risk of FIFO overflow/underflow.

The DMA channels have been permanently assigned to the endpoints in the following manner: Channel 0 is assigned to Endpoint 0, Channel 1 is assigned to Endpoint A, Channel 2 is assigned to Endpoint B, and so forth, See Table 35 for details.

The DMA channels are internally divided into four sets channels each. The channels in each set get a round-robin priority. Set 0 has the highest priority and Set 3 has the lowest. Program endpoints with the most strenuous latency requirements in Set 0. Sets 2 and 3 are low-priority sets. Refer to Table 35 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 two times, and Sets 2 and 3 are each serviced one time.

For example, if all the channels are active and all are requesting data transfers, the sets are prioritized in the 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, therefore, if only Channels 6 and 7 are active and both are requesting data transfers, then Channel 6 gets priority over Channel 7.

Table 35: Channel Priority

Set	Channels	Endpoints	Priority	Number of Times Served
For PXA32x Processor and PXA30x Processor Only				
0	6, 7	F, G	Highest Priority Set	4/8
1	4, 5	D, E	Higher than Set 2 and Set 3. Lower than Set 0.	2/8
2	2, 3	B, C	Higher than Set 3. Lower than Set 0 and Set 1.	1/8
3	0, 1	0, A	Lowest Priority Set	1/8
For PXA31x processor Only				
0	6, 7, 14, 15	F, G, N, P	Highest Priority Set	4/8

Table 35: Channel Priority (Continued)

Set	Channels	Endpoints	Priority	Number of Times Served
1	4, 5, 12, 13	D, E, L, M	Higher than Set 2 and Set 3. Lower than Set 0.	2/8
2	2, 3, 10, 11	B, C, J, K	Higher than Set 3. Lower than Set 0 and Set 1.	1/8
3	0, 1, 8, 9	0, A, H, I	Lowest Priority Set	1/8

For example, Endpoint G can be configured as an Isochronous endpoint, which is permanently assigned to Channel 7, which has the highest priority. Also, Endpoint A can be configured as a Bulk endpoint, which is permanently assigned to Channel 1, which has the lowest priority. Refer to Table 35 for details.

2.7.2.1.3 Concurrent Active Channels

The U2DMA can have up to four outstanding (active) DMA requests at a time. The U2DMA has four 32-byte internal buffers to hold Descriptor information or data fetched from memory.

The system bus can be allocated successively to four active channels or to a single channel for four successive requests, depending on the order and number of channel requests received by the U2DMA. Any combination of number of channels and number of requests per channel is supported for up to a total of four outstanding concurrent transfers on the system bus.

The total number of outstanding concurrent transfers can be reduced by programming the U2DMACR[MAXOCT] field to a non-zero value.

2.7.2.1.4 Channel States

The following states apply to the DMA channels. Refer to Table 36 for the valid software configuration settings to enable the channel states.

- Un-initialized — occurs after a reset. U2DMA Control Register (U2DMACR)U2DMACSRx[STOPINTR] is set when un-initialized.
- Valid Descriptor, not running — occurs when a valid Descriptor has been loaded in the U2DMADADRx register during a Descriptor-Fetch transfer, but the corresponding run bit, U2DMACSRx[RUN], is not set. For a Descriptor-Fetch transfer, U2DMACSRx[STOPINTR] is cleared when the DMA controller updates the U2DMADADRx register.
- Descriptor Fetch, running — after programming U2DMADADRx and setting U2DMACSRx[RUN], four words of Descriptors are fetched from the memory and U2DMACSRx[STOPINTR] continues to be clear. The channel then enters the “Wait for Request” state.
- Wait for Request — occurs as the channel waits for a request before it starts to transfer data; U2DMACSRx[STOPINTR] is clear.
- Transfer data — transferring data between the source and target; U2DMACSRx[STOPINTR] is clear.
- Channel error — the channel in error remains in the Stopped state until software clears the error condition, re-initializes the channel, and sets the U2DMACSRx[RUN] bit and the U2DMACSRx[BUSERRINTR] bit. See Section 2.11.26 and Section 2.11.25 for details.
- Stopped — the channel is stopped. U2DMACSRx[STOPINTR] is set. A stopped channel is re-initialized by updating the U2DMADADRx register and setting U2DMACSRx[RUN].

Table 36: Channel States Based on Software Configuration

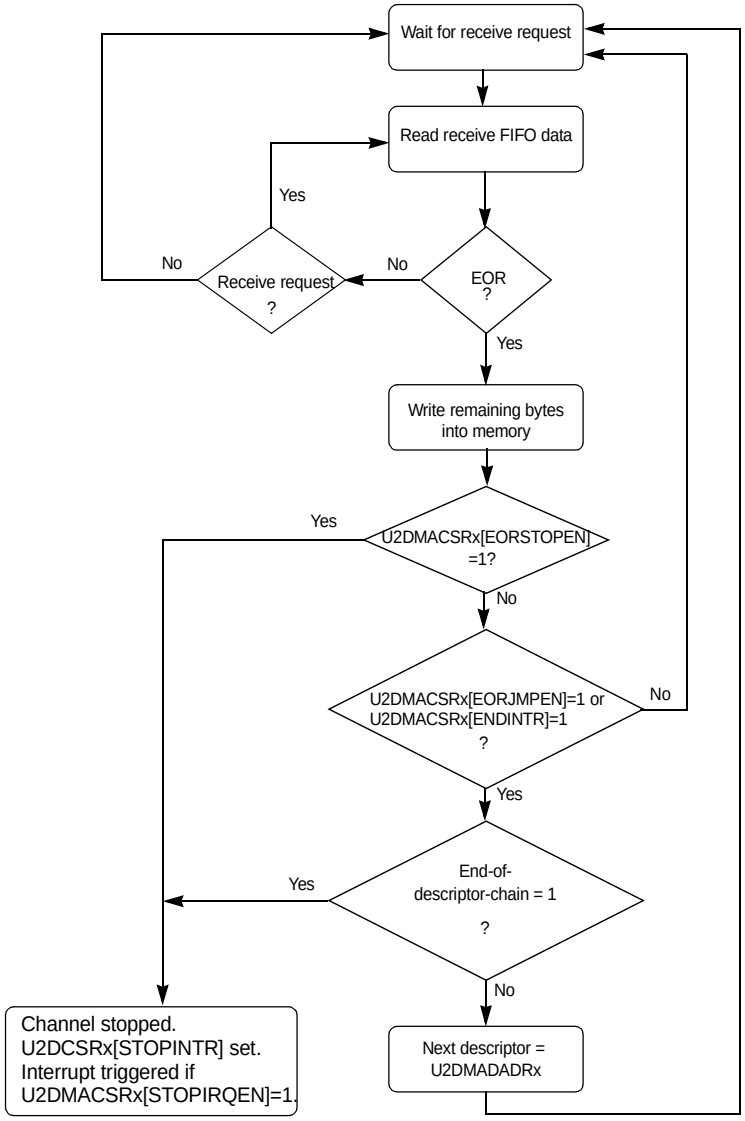
Software Configuration	U2DMACSRx [RUN]	U2DMACSRx [StopIntr]	Resulting channel state
Power-up	0	1	Uninitialized
Write to U2DMADADRx before U2DMACSRx[Run] is set. The descriptor must previously be set up in memory. The descriptor address is the value written to U2DMADADRx.	0	0	Valid descriptor, not running.
Set U2DMACSRx[Run] after writing to U2DMADADRx.	1	0	• Descriptor fetch, running. • Wait for request, running. (Occurs after descriptor fetch.) • Transfer data, running. (Occurs after Endpoint FIFO asserts its request.)
Stop running channel by clearing U2DMACSRx[Run] and U2DMACSRx[MaskRun]	0	0 -> 1	Channel eventually if not immediately switches to a stopped state (identified by U2DMACSRx[StopIntr] toggling from low to high).

2.7.2.2 DMA Descriptors

The U2DMA Descriptors are fetched from four-word-aligned memory addresses when the channel is started. The U2DMA Descriptor registers (U2DMADADRx, U2DMASADRx, U2DMATADRx, and U2DMACMDx) are not directly writable through software. See Section 2.11.20, U2DMA Control Register (U2DMACR) , Section 2.11.21, U2DMA Descriptor Address Registers (U2DMADADRx) , Section 2.11.22, U2DMA Source Address Registers (U2DMASADRx) , and Section 2.11.22, U2DMA Source Address Registers (U2DMASADRx) for more details on these regsisters.

Figure 23 shows the flow of the Descriptor logic.

Figure 23: Descriptor Behavior on End-of-Receive (EOR)



2.7.2.2.1 Descriptor-Fetch Transfer Operation

Descriptor-Fetch transfers operate in the following manner:

Software must first clear the U2DMACSRx[RUN] bit. Software must write a valid Descriptor address to the U2DMADADRx register and then set U2DMACSRx[RUN]. Doing so in this order enables the U2DMA to fetch the four-word Descriptor (if the memory is already set up with the Descriptor chain) from the memory that the U2DMADADRx register indicates. The channel waits for a request from

the Endpoint FIFO and begins to transfer data after the request is received. After the channel transfers a number of bytes equal to the smaller of 32 bytes and U2DMACMDx[LEN], it waits for the next request before it continues with the data transfer until the U2DMACMDx[LEN] reaches zero. The channel stops or continues with a new Descriptor Fetch from the memory, as determined by the U2DMADADRx[STOP] bit. Figure 24 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 U2DMACSRx[RUN] register.

For a Descriptor-Fetch transfer, the software must load the U2DMADADRx register and set the U2DMACSRx[RUN] bit. The channel Descriptor Fetch does not occur unless the U2DMACSRx[RUN] bit is set.

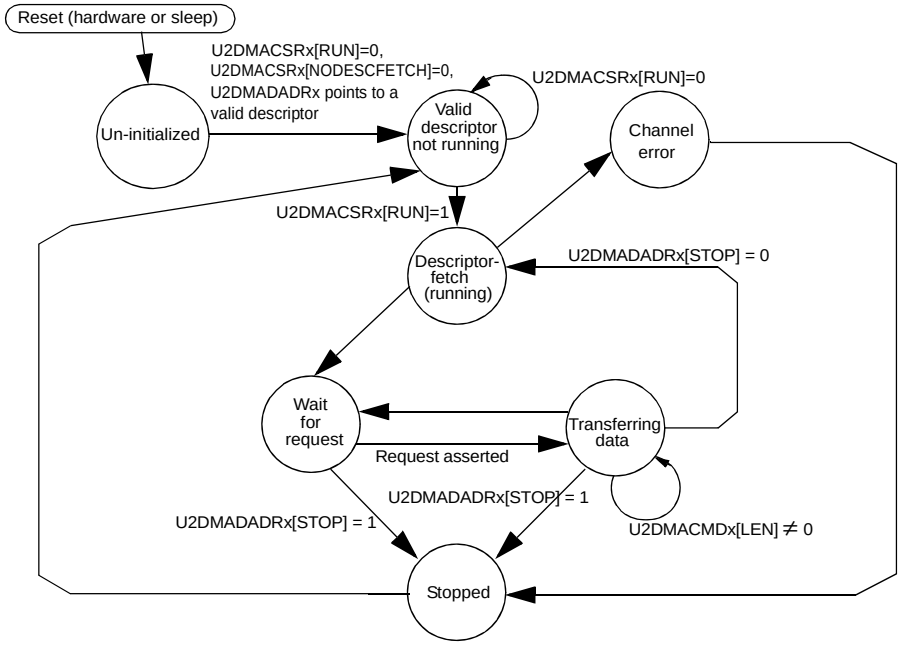
Although software loads the U2DMADADRx register, the U2DMASADRx, U2DMATADRx, and U2DMACMDx registers must be loaded indirectly from DMA Descriptors. The DMA Descriptor indicated in the U2DMADADRx register is loaded into the registers of the associated DMA channel after all of the data described by a Descriptor has been successfully transferred and when a Write to the U2DMACSRx[RUN] register switches the channel from “Stopped” to “Running.”

U2DMADADRx[STOP] of word [0] of a DMA Descriptor (the low bit of the U2DMADADRx field) is used to mark 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 Descriptor.

A DMA Descriptor is a four-word (32-bit) block, aligned on a 16-byte boundary in memory:

- Word [0] of the block contains a value for register U2DMADADRx and a single flag bit (STOP).
- Word [1] contains a value for the U2DMASADRx register. If the source for the DMA transfer is a memory location, this is a 32-bit system memory address. If the source for the transfer is an Endpoint FIFO, this word is ignored, and the channel number determines the Endpoint FIFO accessed as the source of the transfer.
- Word [2] contains a value for the U2DMATADRx register. If the target for the DMA transfer is a memory location, this is a 32-bit system memory address. If the target for the transfer is an Endpoint FIFO, this word is ignored, and the channel number determines the Endpoint FIFO accessed as the target of the transfer.
- Word [3] contains a value for the U2DMACMDx register.

Figure 24: Descriptor-Fetch Transfer Channel State Diagram



2.7.2.3 Transferring Data

The Endpoint FIFOs connected to the DMA operate in flowthrough transfers (refer to Section 2.7.2.3.2 for details). The source or destination of a DMA transfer is always an Endpoint FIFO memory intended to be used as a source or sink of DMA data. An Out endpoint requires a DMA channel setup to transfer data from the Endpoint FIFO to system memory, and an In endpoint requires a DMA channel setup to transfer data from system memory to the Endpoint FIFO. Refer to Section 2.10 for details of the register referenced in the following text.

2.7.2.3.1 Servicing U2DC Endpoint FIFOs

The U2DMA can transfer data either from memory-to-Endpoint FIFO or from Endpoint FIFO to memory. The possible Endpoint FIFO requests are mapped to the available channels on a one-to-one basis. (The Endpoint 0 FIFO request is mapped to U2DMA Channel 0, the Endpoint A FIFO request is mapped to U2DMA Channel 1, the Endpoint B FIFO request is mapped to U2DMA Channel 2, and so on.) The Endpoint FIFOs assert the appropriate endpoint request signal (EPREQx) to signal the FIFO threshold has been exceeded (either rising or falling, depending on the direction of transfer) and that a DMA transfer is requested. The EPREQ signals are directly sampled on every system bus clock. If the U2DMACMDx[XFRDIR] bit is cleared, the accessed channel number indicates which Endpoint FIFO is the source for the transfer. (U2DMACMD0 refers to Endpoint 0, U2DMACMD1 refers to Endpoint A, and so on.) This configuration corresponds to an Out endpoint, and the U2DMASADRx register is ignored in this instance. IF the U2DMACMDx[XFRDIR] bit is set, the accessed channel number indicates which Endpoint FIFO is the target for the transfer. (U2DMACMD0 refers to Endpoint 0, U2DMACMD1 refers to Endpoint A, and so on.) This configuration corresponds to an In endpoint, and the U2DMATADRx register is ignored in this instance. In both instances, the DMA processor waits for the request before it initiates the transfer.

If U2DMACMDx[ENDIRQEN] is set, a U2DMA interrupt is requested at the end of the last cycle associated with the byte that caused U2DMACMDx[LEN] to decrement to zero. The U2DMA interrupt is logically OR'd with the U2DC interrupt to generate a single interrupt signal.

2.7.2.3.2 Servicing In Endpoint FIFOs Using DMA Read Cycles

A DMA Read begins when an Endpoint FIFO sends a request via the EPREQ bus to a channel in the U2DMA while the channel is running and is configured for Reads. The number of bytes to be transferred is normally specified to be 32 bytes, but the transfer length can be less when a short packet is indicated. The short packet is communicated either when the U2DMACMDx[LEN] field shows less than 32 bytes for IN transfers, or by the Endpoint FIFO setting the End of Packet (EOP) signal for Out transfers. The following process begins when the request is recognized:

1. The U2DMA prompts the memory controller to read the required number of bytes addressed by U2DMASADRx into a 32-byte buffer in the U2DMA.
2. The U2DMA transfers the data to the Endpoint FIFO addressed by the channel number. The width of data transfer to the Endpoint FIFO is always 64 bits. For Endpoint 0, Byte Enables determine if all 8 bytes or a subset of the 64-bit transfer is written to the FIFO.
3. At the end of the transfer, U2DMASADRx is increased and U2DMACMDx[LEN] is decreased by the smaller of U2DMACMDx[LEN] and 32 bytes.

Use the following settings for the U2DMA register bits for a DMA Read from memory to an Endpoint FIFO:

- U2DMASADRx[SRCADDR] = memory address
- U2DMATADRx[TGTADDR] not used
- U2DMACMDx[XFRDIR] = 1
- DMA channel number (x) = Endpoint FIFO target address

Refer to Section 2.7.2.5 for a description of how trailing bytes are handled when servicing Endpoint FIFOs with DMA Read cycles.

Note

If multiple DMA Read transfers are chained together to transfer a packet, only the final transfer can be allowed to have the U2DMACMDx[LEN] field set to anything less than 32 bytes. If an intermediate DMA transfer has U2DMACMDx[LEN] set to less than 32 bytes, this action is interpreted by the FIFO as the end of the packet (a Short packet.) Because the U2DMACMDx[LEN] field is decremented by each 32-byte access, the value loaded must be a factor of 32 bytes for a Short packet not to be indicated.

2.7.2.3.3 Servicing Out Endpoint FIFOs Using DMA Write Cycles

A DMA Write begins when an Endpoint FIFO sends a request via the EPREQ bus to a channel in the U2DMA while the channel is running and is configured for Writes. The number of bytes to be transferred is normally specified to be 32 bytes, but the transfer length can be less when a Short packet is indicated. The Short packet is communicated either when the U2DMACMDx[LEN] field shows less than 32 bytes for IN transfers, or by the Endpoint FIFO setting the End of Packet (EOP) signal for OUT transfers. The following process begins when the request is recognized:

1. The U2DMA processes the request by transferring the required number of bytes from the Endpoint FIFO addressed by the channel number into a U2DMA buffer.

2. The U2DMA transfers the data to the memory controller via the system bus. The width of the data transfer from the Endpoint FIFO is always 64 bits. For Endpoint 0, Byte Enables determine if all 8 bytes or a subset of the 64-bit transfer is written to the system memory.
3. At the end of the transfer, U2DMATADRx is increased and U2DMACMDx[LEN] is decreased by the smaller of U2DMACMDx[LEN] and 32 bytes.

Use the following settings for the U2DMA register bits for a DMA Write to memory from an Endpoint FIFO:

- U2DMASADRx[SRCADDR] = not used
- U2DMATADRx[TRGADDR] = memory address
- U2DMACMDx[XFRDIR] = 0
- DMA channel number (x) = Endpoint FIFO source address

Refer to Section 2.7.2.5 for a description of how trailing bytes are handled when servicing Endpoint FIFOs with DMA Write cycles.

Note

If multiple DMA transfers are chained together to transfer a packet, only the final transfer can be allowed to have the U2DMACMDx[LEN] field set to anything less than 32 bytes. If an intermediate DMA transfer has U2DMACMDx[LEN] set to less than 32 bytes, this action is interpreted by the FIFO as the end of the packet (a Short packet). Because the U2DMACMDx[LEN] field is decremented by each 32 byte access, the value loaded must be a factor of 32 bytes for a Short packet not to be indicated.

If multiple DMA Write transfers are chained together to form a single packet, all Write transfers, except for the final one, must be a multiple of 8 bytes (U2DMACMDx[LEN] = 8x). The final Write transfer is not restricted to be a multiple of 8 bytes and can have the U2DMACMDx[LEN] field set to any value (U2DMACMDx[LEN] = x).

2.7.2.4 Programming Tips

This section provides information concerning software requirements, instruction ordering, and misaligned memory accesses.

2.7.2.4.1 Software Management Requirements

The information that must be maintained on a per-stream basis (for example, the memory address, the Endpoint FIFO address, the transfer count, and the implied direction of data flow), is maintained in Descriptor registers in the U2DMA. The Descriptor registers are loaded from memory locations specified by the software. Multiple DMA Descriptors can be chained together in a list, which 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.

Each demand for data that an endpoint generates is a memory data Read or Write.

2.7.2.4.2 Instruction Ordering

The U2DMA completes programmed I/O instructions in the order received.

References to internal addresses generally complete more quickly than those issued to external addresses. This action means that memory accesses can be sent in one order and completed in a different order.

The U2DMA 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).

2.7.2.4.3 Misaligned Memory Accesses

The U2DMA is a 64-bit device that can access memory on byte-aligned boundaries. The U2DMA may encounter misaligned (not aligned to 64-bit boundary) addresses while it accesses memory.

The U2DMA employs channel-specific Alignment buffers that hold either the leading or the lagging misaligned data. When the U2DMA 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 U2DMA encounters overhead while it works with misaligned data. However, the U2DMA performs the alignment to optimize performance when memory addresses cannot be restricted in this manner.

2.7.2.5 How DMA Handles Trailing Bytes

DMA normally transfers bytes equal to the transaction size of 32 bytes. But the number of trailing bytes in the U2DMACMDx[LEN] field could be smaller than the transfer size only when the final transfer is a Short packet. The DMA can transfer the exact number of trailing bytes if it receives a corresponding request from the Endpoint FIFO. Two instances are possible:

- Memory-to-Endpoint FIFO transfers: The Endpoint FIFO sends a normal request to the U2DMA to receive the trailing bytes. The DMA transfers a number of bytes equal to the smaller of U2DMACMDx[LEN] and 32 bytes. No special handshaking is necessary in this instance.
- Endpoint FIFO-to-memory transfers: Special handshaking signals and interrupts are employed for transferring trailing bytes from an Endpoint FIFO to memory. The conditions that use the handshaking signals and interrupts are explained below:
  - End of Packet (EOP): The endpoint receives its last data sample from the external USB host controller and detects an EOP based on its receive protocol. Any remaining data samples in the Endpoint Receive FIFO are treated as trailing bytes. The endpoint initiates 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. The endpoint logic asserts its EOP handshake signal to the DMA concurrently with the transfer of the last data byte to the DMA. CPU intervention is not required as long as the Descriptor chain has not ended.

Note

When an endpoint signals an EOP, the U2DMA sets the End-of-Receive (EOR) status bit in the corresponding channel Control Status register (U2DMACSRx). See Table 77 for details.

  - End-of-Descriptor Chain (EOC): Signal indicates that a DMA channel is at the end of its last Descriptor. After the current transfer, U2DMACMDx[LEN]=0 and U2DMADADRx[STOP] = 1. For non-0 endpoints, the software must add a new Descriptor and set the Run bit again. For Endpoint 0, the software can either add a new Descriptor and set the Run bit again or use programmed I/O to retrieve any trailing bytes. EOC is the only trailing-bytes case that can potentially require programmed I/O to retrieve data for Endpoint 0.
  - Request after Channel Stops (RAS): This signal is a status bit in the U2DMA Control Status register (U2DMACSRx). This bit is set when an Endpoint FIFO has generated a request for a channel that has completed its last Descriptor and has no new Descriptors in its Descriptor chain. This error condition must be handled either by setting up a new Descriptor or through programmed I/O. A new Descriptor must be loaded and enabled prior to clearing this bit by writing a 1 to it, or the bit clears for a single clock only. See Table 77 for details.

The following examples illustrate the handling of various trailing bytes using EOR, EOC, and RAS.

Example 1: The endpoint signals a DMA request to service trailing bytes in its Receive FIFO (RxFIFO). The current Descriptor U2DMACMDx[Len] is equal to or greater than the trailing-bytes count.

1. The endpoint signals a receive DMA request.
2. The U2DMA responds and reads out all trailing bytes including the last byte.
3. The endpoint signals an EOR.
4. The U2DMA transfers all trailing bytes to the channel target and then updates U2DMACSRx[EORInt].
5. The DMA channel can be configured to stop, jump, or just wait for another request after receiving EOR, depending on U2DMACSRx[EORStopEn] and U2DMACSRx[EORJmpEn].
6. U2DCSRx[EORInt] set indicates that all trailing bytes were read and transferred to the required target.

Example 2: The endpoint signals a DMA request to service trailing bytes in its RxFIFO. The current Descriptor U2DMACMDx[Len] is less than the trailing-bytes count. More Descriptors are available in the Descriptor chain.

1. The endpoint signals a receive DMA request.
2. The U2DMA responds and reads only the number of trailing bytes programmed by U2DMACMDx[Len].
3. The current Descriptor U2DMACMDx[Len] decrements to zero. This condition forces the U2DMA to transfer all the data read from the peripheral's RxFIFO to the channel target.
4. The U2DMA fetches the next Descriptor.
5. Since the endpoint still has trailing bytes in its RxFIFO, it must make another request. The endpoint continues to issue such requests until the U2DMA reads out all the trailing bytes and until an EOR is signalled.
6. U2DCSRx[EORInt] set indicates that all trailing bytes were read and transferred to the required target.

Example 3: The endpoint signals a DMA request to service trailing bytes in its RxFIFO. The current Descriptor U2DMACMDx[Len] is less than the trailing-bytes count. No more Descriptors are available in the Descriptor chain.

1. The endpoint signals a receive DMA request.
2. The U2DMA responds and reads only the number of trailing bytes programmed by U2DMACMDx[Len].
3. The current Descriptor U2DMACMDx[Len] decrements to zero. This condition forces the U2DMA 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. The DMA sets U2DMACSRx[STOPINTR] if the U2DMA reached an end-of-chain (EOC) condition while reading the last byte. This bit can cause an interrupt to the CPU if programmed to do so by setting the U2DMACSRx[STOPIRQEN] bit. The interrupt indicates for software to handle the remainder of the trailing bytes in the endpoint RxFIFO either through setting up a new Descriptor or through programmed I/O.
6. The U2DMA sets U2DMACSRx[RAS] if the endpoint signals a DMA request before software sets up a new Descriptor. This setting indicates for software to handle the remainder of the trailing bytes in the endpoint RxFIFO.

2.7.2.6 Quick Reference to DMA Programming

Table 37 provides a quick reference for programming the U2DMA for the U2DC endpoints.

Table 37: DMA Quick Reference for Endpoint FIFOs

Endpoint Accessed	U2DMASADRx	U2DMATADRx	Burst Size (bytes)	XFRDIR	U2DMA Channel Request
endpoint 0	unused	U2DMATADR0	32 or trailing	0	0
endpoint 0	U2DMASADR0	unused	32 or trailing	1	0
endpoint A	unused	U2DMATADR1	32 or trailing	0	1
endpoint A	U2DMASADR1	unused	32 or trailing	1	1
endpoint B	unused	U2DMATADR2	32 or trailing	0	2
endpoint B	U2DMASADR2	unused	32 or trailing	1	2
endpoint C	unused	U2DMATADR3	32 or trailing	0	3
endpoint C	U2DMASADR3	unused	32 or trailing	1	3
endpoint D	unused	U2DMATADR4	32 or trailing	0	4
endpoint D	U2DMASADR4	unused	32 or trailing	1	4
endpoint E	unused	U2DMATADR5	32 or trailing	0	5
endpoint E	U2DMASADR5	unused	32 or trailing	1	5
endpoint F	unused	U2DMATADR6	32 or trailing	0	6
endpoint F	U2DMASADR6	unused	32 or trailing	1	6
endpoint G	unused	U2DMATADR7	32 or trailing	0	7
endpoint G	U2DMASADR7	unused	32 or trailing	1	7
For PXA31x Processor:					
	unused	U2DMATADR8	32 or trailing	0	8
endpoint H	U2DMASADR8	unused	32 or trailing	1	8
	unused	U2DMATADR9	32 or trailing	0	9
endpoint I	U2DMASADR9	unused	32 or trailing	1	9
	unused	U2DMATADR10	32 or trailing	0	10
endpoint J	U2DMASADR10	unused	32 or trailing	1	10
	unused	U2DMATADR11	32 or trailing	0	11
endpoint K	U2DMASADR11	unused	32 or trailing	1	11
	unused	U2DMATADR12	32 or trailing	0	12
endpoint L	U2DMASADR12	unused	32 or trailing	1	12

Table 37: DMA Quick Reference for Endpoint FIFOs (Continued)

Endpoint Accessed	U2DMASADRx	U2DMATADRx	Burst Size (bytes)	XFRDIR	U2DMA Channel Request
	unused	U2DMATADR13	32 or trailing	0	13
endpoint M	U2DMASADR13	unused	32 or trailing	1	13
	unused	U2DMATADR14	32 or trailing	0	14
endpoint N	U2DMASADR14	unused	32 or trailing	1	14
	unused	U2DMATADR15	32 or trailing	0	15
endpoint P	U2DMASADR15	unused	32 or trailing	1	15

2.7.2.7 Examples

Example 1: Setting Up and Starting a Channel

The following example of code shows how to set up and start a channel to transfer LEN words that start at the address in the U2DMASADRx register to the endpoint address referenced by the channel number. In this example, the stop bit in the U2DMADADRx 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].u2dmadadr = STOP;
desc[0].u2dmasadr = U2DMASADR;
desc[0].u2dmatadr = don't care;
desc[0].u2dmacmd = U2DMACMD;
// start the channel
DMANEXT[CHAN] = &desc[0];
DRUN = 1;
```

Example 2: Adding a Descriptor to End-of-Descriptor List (channel running)

Note

This example assumes that a Descriptor-fetch transfer is active.

DMA Descriptor lists are used as queues of full buffers for transmitted or received data. 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 U2DMACSRx[RUN] = 0.
2. Wait for the channel to stop. The stop status is indicated in U2DMACSRx[STOPINTR].
3. Create the end Descriptor in the memory with the stop bit set.

4. Manipulate the U2DMADADRx register of the last Descriptor of the current chain in the memory to ensure that the U2DMADADRx register points to the newly created end Descriptor in step 3.
5. Create a new Descriptor with values in the U2DMADADRx, U2DMASADRx, U2DMATADRx, and U2DMACMDx 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 U2DMADADRx register of this Descriptor so that it points to the newly created end Descriptor.
7. Program the channel U2DMADADRx register with the next Descriptor created in Step 5.
8. Set U2DMACSRx[RUN] = 1.

2.7.3 Endpoint Memory Configurations

2.7.3.1 PXA32x Processor and PXA30x Processor Memory Configurations

The Endpoint memory consists of 8-Kbytes of SRAM, arranged as a 1K x 64-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. One-hundred-twenty-eight bytes of the Endpoint memory are dedicated for Endpoint 0 USB data storage, leaving 8064 bytes that can be allocated by software to Endpoints A - G. The entire memory space for Endpoints A - G is flushed and the memory reallocated when software set the *S*witch Endpoint Memory to Active Configuration (SMAC) bit in U2DC Control register (U2DCR) to 1. Each configuration setting allocates the entire 8064 bytes of Endpoint memory for use when that configuration and its given set of alternate interfaces are active. The USB data stored in the 8-Kbytes SRAM is accessed by software using the U2DMA.

2.7.3.2 PXA31x Processor Memory Configurations

The Endpoint memory consists of 16-Kbytes of SRAM, arranged as a 1K x 64-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. 128 bytes of the Endpoint memory are dedicated for Endpoint 0 USB data storage, leaving 16256 bytes that can be allocated by users to Endpoints A - P. The entire memory space for Endpoints A - P is flushed and the memory reallocated when users set the *S*witch Endpoint Memory to Active Configuration (SMAC) bit in U2DC Control register (U2DCR) to 1. Each configuration setting allocates the entire 16256 bytes of Endpoint memory for use when that configuration and its given set of alternate interfaces are active. The USB data stored in the 16-Kbytes SRAM is accessed by users using the U2DMA.

2.7.3.3 FIFO Memory Allocation

Each enabled endpoint is allocated space in the SRAM. See Section 2.11.17 for details.

Warning

For the PXA32x processor and PXA30x processor, this configuration has only 5 of the 7 programmable endpoints enabled, and uses a total FIFO space of 4448 bytes of the 8064 bytes available. For the PXA31x processor, this configuration has only 5 of the 15 programmable endpoints enabled, and uses total FIFO space of 4448 bytes of the16256 bytes available.

The Endpoint memory is 64 bits wide, and is allocated only on 64-bit boundaries. Although Isochronous and Interrupt endpoints can be programmed for any maximum packet size from 1 - 1024 bytes, if an endpoint is programmed for a maximum packet size that is not 64-bit aligned, its minimum FIFO memory space is allocated up to the next valid 8-byte boundary.

In addition to FIFO memory allocated for data storage, one 64-bit word of FIFO memory is also consumed for each packet stored in endpoints. These 8-bytes are used to store implementation-specific information about the packet such as packet size. This memory space is referred to as a *Packet Delimiter*. The data within the Packet Delimiter cannot be accessed by users, and no knowledge of the Packet Delimiter is required for operation other than for allocation of FIFO memory space.

The minimum example of a max packet size of 1 byte would require FIFO memory space of one 8-byte word for a Packet Delimiter, and an additional 8-byte word to hold the single byte, with the other 7 bytes unused. To hold three minimum packets would require allocation of three of these sets as shown in Table 38.

Table 38: Example Memory Allocation for 1-Byte Packets

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
FIFO Memory for Previous Endpoint							
Packet #1 Delimiter							
Unused	Unused	Unused	Unused	Unused	Unused	Unused	Data Byte
Packet #2 Delimiter							
Unused	Unused	Unused	Unused	Unused	Unused	Unused	Data Byte
Packet #3 Delimiter							
Unused	Unused	Unused	Unused	Unused	Unused	Unused	Data Byte
FIFO Memory for Next Endpoint							

In another example, if an Isochronous endpoint is configured for a maximum packet size of 317 bytes, the endpoint must be allocated a minimum of 320 bytes of FIFO memory space for data storage, and eight additional bytes for a Packet Delimiter for a total of 328 bytes. The additional three bytes of allocated FIFO memory space is not used and cannot be accessed. If this endpoint was programmed to have a FIFO depth to support two complete packets, it would use a total of 656 bytes of FIFO memory space (328 bytes x 2). FIFO memory is *not* required to be allocated as multiples of packet size, and arbitrary number of bytes can be allocated for a given endpoint as long as it is larger than the storage required for a single packet. For example, allocating 1-1/2 packets of FIFO memory would allow the system or USB to work ahead by 1/2-packet of data to reduce the risk of FIFO overrun/underrun.

However, beyond the 8-byte alignment, any packet that is less than the programmed maximum packet size is *not* considered as occupying the entire packet size in the FIFO, and it only requires the number of bytes to align it to an 8-byte boundary. For example, in a max packet sized Out FIFO, a Short or Zero packet occupies less than one packet, leaving the rest of the FIFO space open to begin loading the next packet. Consider the example of an endpoint with max packet size of 16 bytes. If 48 total bytes of FIFO memory were allocated, this would normally support storage of two full packets, which includes two sets of eight bytes each for 16 bytes of Packet Delimiters, and two sets of 16 bytes each for 32 bytes of data storage.

However, consider the example where the first packet transferred is a Zero packet, the second packet transferred is a 7-byte Short packet, and these are then followed by a full packet. In this instance, all three packets are accommodated in the 48 bytes of FIFO memory allocated as illustrated in Table 39.

Table 39: Example Memory Allocation for Zero and Short Packets

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
FIFO Memory for Previous Endpoint							
Packet #1 Delimiter (Zero Packet, so no data follows)							
Packet #2 Delimiter (7-byte Short Packet follows)							
Unused	Data Byte 6	Data Byte 5	Data Byte 4	Data Byte 3	Data Byte 2	Data Byte 1	Data Byte 0
Packet #3 Delimiter (16-byte full packet follows)							
Data Byte 7	Data Byte 6	Data Byte 5	Data Byte 4	Data Byte 3	Data Byte 2	Data Byte 1	Data Byte 0
Data Byte 15	Data Byte 14	Data Byte 13	Data Byte 12	Data Byte 11	Data Byte 19	Data Byte 9	Data Byte 8
FIFO Memory for Next Endpoint							

When loading an Endpoint 0 with a packet that is not 32-bit aligned when using direct processor access, write all data as 32-bit accesses except for the final, partial word Write. The U2DMA takes care of alignment itself when it is used to accomplish data transfers.

Table 40 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 with FIFO depth > Max Packet Size, and so one packet can be processed while the next is being assembled. If an In endpoint has FIFO depth > Max Packet Size, the next packet can be loading into the Transmit FIFO while the U2DC is transmitting the first packet. Likewise, while unloading an Out endpoint, the U2DC can continue to process the next incoming packet to that endpoint. Endpoint 0 has 128 bytes of FIFO memory space, 64 bytes for IN data, and 64 bytes for Out data and the FIFO depth matches the max packet size. No Endpoint Memory Delimiter is required, allocated, or utilized for Endpoint 0.

Table 40: Endpoint Configuration

Endpoint Type	U2DC Endpoint	Endpoint Direction	Full-Speed Maximum Packet Size (bytes)	High-Speed Maximum Packet Size (bytes)	Minimum Memory Allocated for Data (bytes)	Minimum Total Allocated Memory (bytes)	Notes
Control	0	In/Out	64	64	128 (64 for In, 64 for Out)	128 (64 for In, 64 for Out)	
Bulk	A - G	In or Out	8,16, 32 or 64	512	8, 16, 32, 64 or 512 (HS)	16, 24, 40, 72 or 520 (HS)	
Bulk	H - P	In or Out	8,16, 32 or 64	512	8, 16, 32, 64 or 512 (HS)	16, 24, 40, 72 or 520 (HS)	PXA31x Only

Table 40: Endpoint Configuration (Continued)

Endpoint Type	U2DC Endpoint	Endpoint Direction	Full-Speed Maximum Packet Size (bytes)	High-Speed Maximum Packet Size (bytes)	Minimum Memory Allocated for Data (bytes)	Minimum Total Allocated Memory (bytes)	Notes
Isochronous	A - G	In or Out	1 - 1023	1-1024	8 - 1024	16-1032	
Isochronous	H - P	In or Out	1 - 1023	1-1024	8 - 1024	16-1032	PXA31x Only
Interrupt	A - G	In or Out	1 - 64	1-1024	8 - 64 (FS)/ 8-1024 (HS)	16-72 (FS)/ 16-1032 (HS)	
Interrupt	H - P	In or Out	1 - 64	1-1024	8 - 64 (FS)/ 8-1024 (HS)	16-72 (FS)/ 16-1032 (HS)	PXA31x Only

Note

Total allocated memory includes eight additional bytes for each non-control packet for a Packet Delimiter. Control packets do not require delimiters because only a single packet is stored starting at Address 0.

Programmable endpoints can be selected as Bulk, Isochronous, or Interrupt endpoints. All Bulk endpoints have maximum packet sizes of eight, 16, 32, or 64 bytes (full speed) or 512 bytes (high speed), and must be allocated at least this much FIFO memory space plus eight bytes for a Packet Delimiter. Similarly, all Isochronous endpoints and Interrupt endpoints have maximum packet sizes of 1 - 1024 bytes, and would require a minimum allocated FIFO memory space of 8-1024 bytes plus eight bytes for a Packet Delimiter. The total number of bytes of FIFO memory space allocated to all enabled endpoints (besides Endpoint 0) for each configuration cannot exceed the maximum number of bytes.

The error will be detected when more than the maximum allowable bytes of FIFO memory space are allocated to a single configuration with its currently active alternate interface settings, when the Endpoint configuration is checked after setting SMAC. This error sets the Endpoint Memory Configuration Error (EMCE) bit in the U2DC Control Register (U2DCR) to 1 and clears the U2DC Enable bit (UDE) in the U2DC Control Register (U2DCR) to 0. The clearing of UDE also disables the U2DC. If maximum number of bytes or less of FIFO memory space is allocated to the active configuration and its alternate interfaces, when the Endpoint configuration is checked, the U2DC Active (UDA) bit is set to 1 and the UDE bit remains set to 1. Figure 32 shows the sequence for allocating the FIFO memory, setting Endpoint configuration, and enabling the U2DC for USB operation. Required user actions are shown in *italics*.

The data currently in the FIFO is lost when re-allocating the memory due to a set configuration or set interface command after operation has begun. All active DMA channels or direct processor accesses to U2DC must be stopped by software before overwriting control registers that results in re-allocation of memory. Data residing in the memory FIFOs is lost, even for packets that had a successful handshake indication to the USB host.

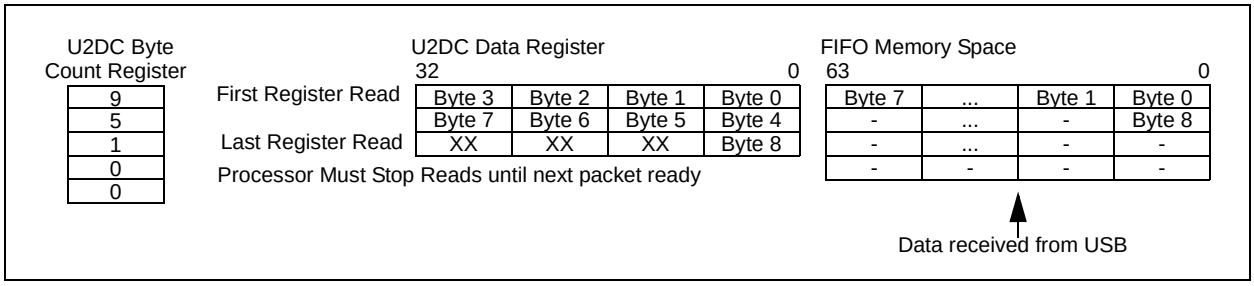
2.7.3.4 FIFO Organization and Data Ordering

Note

Core accesses to the FIFO associated with Endpoint 0 (U2DC End Point 0 Data Register (U2DDR0)) are allowed. Core accesses to FIFOs associated with (U2DC Endpoint Configuration Registers (U2DCRx) are not allowed. Use the DMA to access U2DCRx registers.

Four-byte word accesses must be used if the XScale® core is used to unload the Endpoint memory for Out endpoints. All processor reads from Endpoint Receive FIFOs must be 32-bit word accesses which alternate between access to the lower and upper halves of the 64-bit data word stored in the FIFO memory. The Byte Count register should be read once prior to reading the first Receive FIFO data of the packet to determine the number of valid bytes to read from the FIFO, and then the appropriate number of Reads must be initiated. If an Out endpoint data is not 32-bit aligned, the Read of the final entry in the Receive FIFO contains unknown data in the unused byte locations. If the upper half of the 64-bit word in the FIFO memory contains all unused data, the processor must not read the final unused 4-byte word; for example, the processor must never read more 32-bit words that exist in the packet or make an additional read after the byte count is 0. Figure 25 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 9 bytes.

Figure 25: Data Ordering and FIFO Organization for Out Endpoints



If the DMA is used to load Endpoint memory for In endpoints, the DMA channel must be programmed with the length set to less than or equal to the maximum packet size. If the maximum packet size is not 64-bit aligned, the DMA writes the last entry of data into the Transmit FIFO using the appropriate byte enables. Similarly, when loading a short packet, the DMA descriptor must be programmed with the length of the short packet. If the short packet size is not 64-bit aligned, the DMA loads the final entry of data using byte enables. The byte enables are handled automatically by the DMA and no additional user action is required other than setting the transfer length in the descriptor.

If the maximum packet size for an In endpoint is 32-bit aligned and the XScale® core is used to load the Transmit FIFO, all data must be loaded using word writes. If the maximum packet size for an In endpoint is not 32-bit aligned and the XScale® core is used to load the Transmit FIFO, all data except for the last entry in the FIFO must be loaded using word writes. The last entry of the Transmit FIFO can be loaded using byte enables by having the processor request the correct number of bytes on the last access.

Since the processor core cannot issue writes of size three bytes, adjust the size of the last Data Register 0 Write via setting IPA bit (in packet adjust) for byte counts equal to 4n+3, where n = 0,1,...,15 (3, 7,..., 63). Software should write the final word as a full 4-byte transfer instead of the impossible 3-byte access (the MSB can hold any data.). Then software writes the IPA bit along with IPR (to signal the end of the packet), which results in adjusting the internal count for number of bytes

to transfer over the USB bus as if the last PIO transfer was three bytes instead of four bytes. If the number of bytes loaded through PIO was not four, the resulting byte total to be transferred will be corrupted, resulting in incorrect size data being transferred. IPA should be set only when IPR is set, and be cleared at the same time as IPR when the packet is transferred over the USB bus or the packet is flushed.

Figure 27 shows the loading of FIFO memory space by writing the Endpoint Data register for an In endpoint with a 32-bit aligned packet size of 16 bytes. Figure 26 shows the loading of FIFO memory space by writing the Endpoint Data register for an In endpoint with a non-32-bit aligned packet size of 17 bytes. Here, once the final byte is written no additional Write is required or allowed, despite that the final unused bytes of the 64-bit FIFO word are not written by the processor.

Figure 26: Data Ordering and FIFO Organization for In Endpoints- 32-Bit Aligned

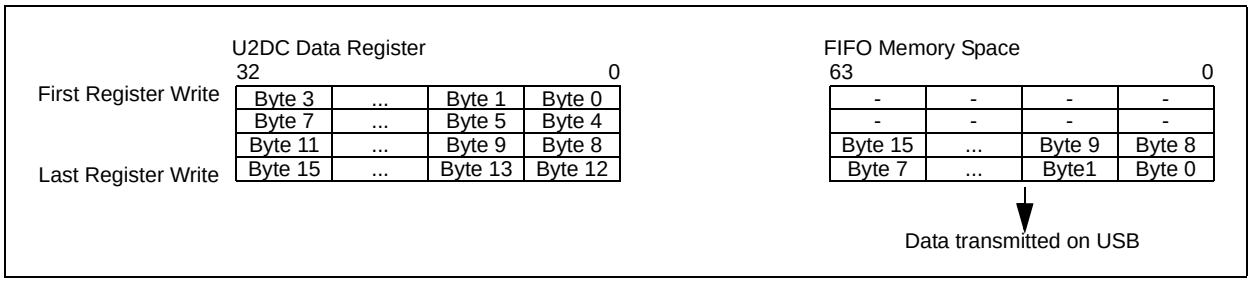
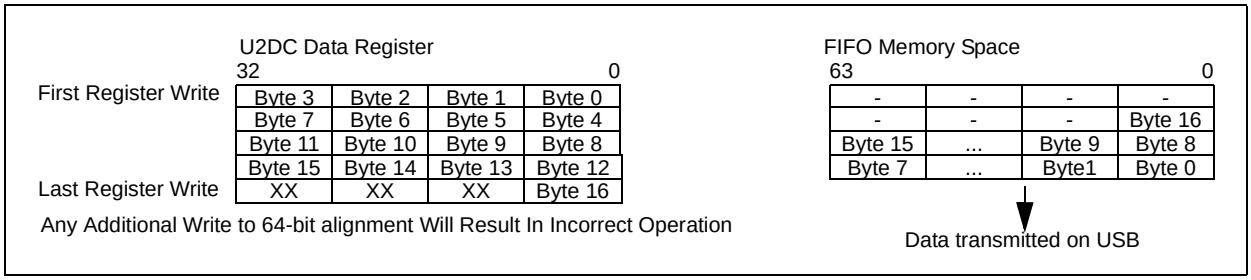
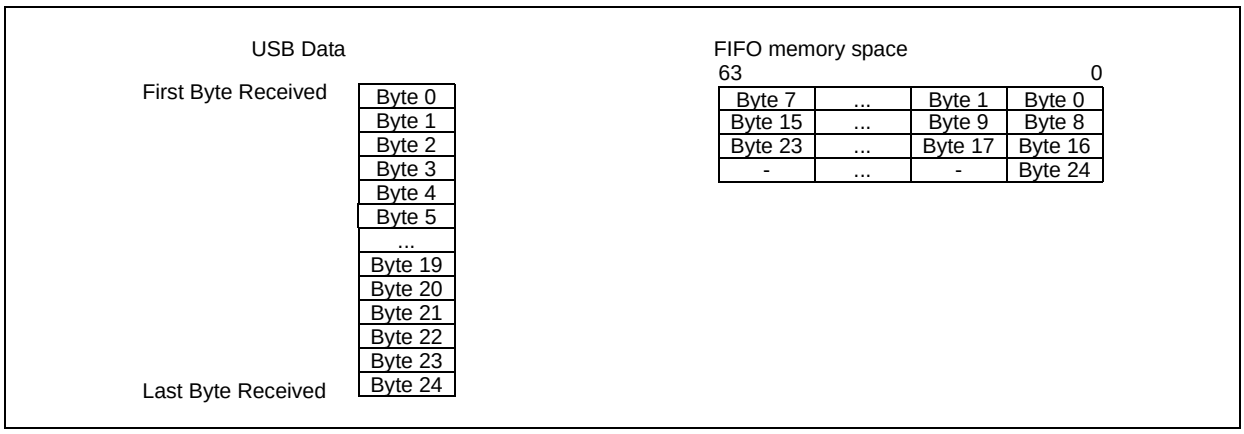


Figure 27: Data Ordering and FIFO Organization for In Endpoints- NON-32-Bit Aligned



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 28 shows an example of how the data received on the USB is organized in the Endpoint memory. In this example, 25 bytes of data is received, the Receive data is not 64-bit aligned, and seven bytes of the data in the last register are left as unknown.

Figure 28: Data Ordering in Endpoint Memory



2.7.3.5 Access via DMA

The U2DC internal DMA can be used to load and unload the Endpoint memory. When the DMA is used to load or unload 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 1 to enable the DMA request for the endpoint. When the U2DC and DMA are set for DMA loading and unloading of Endpoint memory, it is possible for the U2DC to remain operational without requiring the generation of any interrupts to the core for endpoints. Endpoint 0 operates differently and does require generation of interrupts to the core because it communicates control and setup information, which must be handshaked with software.

When an Out endpoint that has DME set to 1 receives a maximum size packet or a short packet (except a zero size packet) from the USB Host controller, the U2DC 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 has responded to the request, the U2DC negates the DMA request, but re-asserts the request if the DMA burst does not unload the entire packet. The U2DC continues to request DMA service until the Receive FIFO has been emptied of data in any completed packets. When the DMA reads the last entry in the Receive buffer for each packet, the U2DC generates an end-of-packet (EOR) to the DMA controller.

The DMA continues to service the U2DC request until either the end of Descriptor has been reached or the end of packet has been received from the U2DC. If the DMA stops reading data from the Endpoint buffer due to reaching the end of the Descriptor, more data may still be in the Endpoint buffer. In this instance, the DMA must fetch another Descriptor to continue unloading data from the Endpoint buffer. The U2DC does not generate an interrupt to the XScale® core if the U2DMA reaches the end of the Descriptor chain and has not unloaded all of the data from the Receive buffer. The U2DMA must be programmed to generate an interrupt when it reaches the end of the Descriptor chain and the core must be programmed to check the Endpoint control/status register to determine if more data is pending.

Multiple packets can not be chained together into a single Descriptor. Each packet completes the DMA access in a Descriptor. For In endpoints, the U2DMACMD[PACKCOMP] bit in the Descriptor must be set along with the last Descriptor in the packet. For Out endpoints, the U2DC initiates an EOR to the U2DMA, which signifies that the end of a packet causes the remainder of the DMA transfer to abort.

If the Short Packet (SP) bit in the Endpoint Control register is set, indicating the packet in the Endpoint buffer is a short packet, the U2DC asserts a DMA request for the endpoint and the DMA receives the request and responds to the request by unloading the data from the Endpoint buffer.

With the final data an EOR is issued to the U2DMA, which signifies the end of the packet, and to end the current DMA access even if the U2DMACMD[LEN] shows the DMA is attempting to transfer more data. The SP bit is cleared automatically when the PC is cleared.

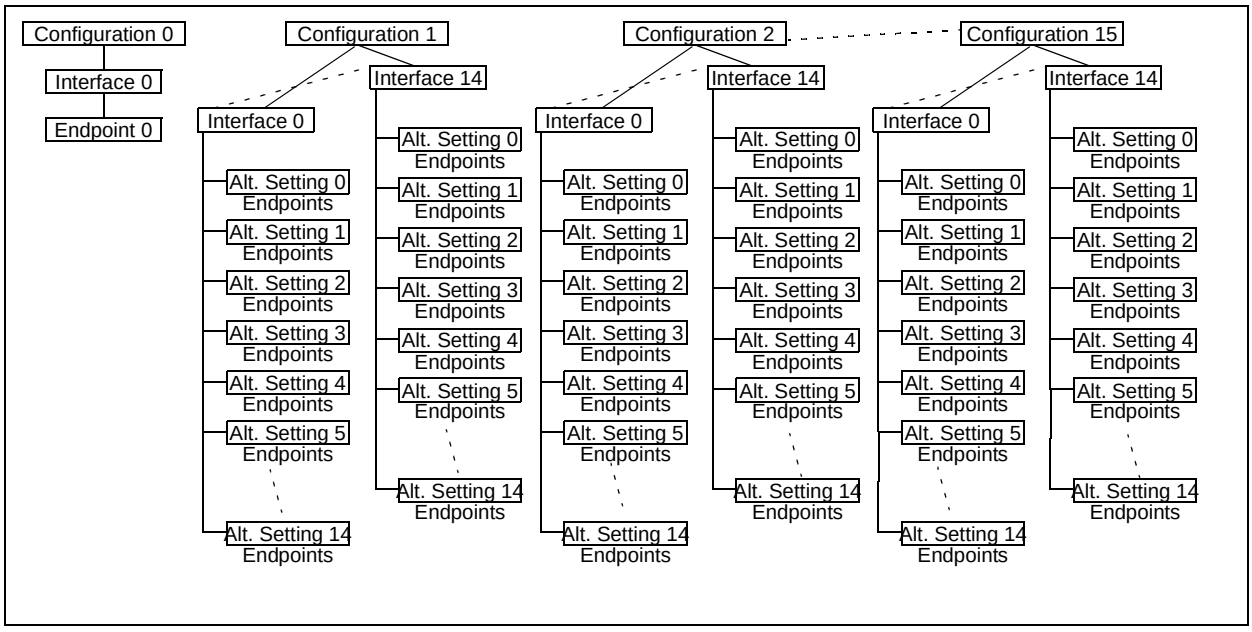
For zero-packets, if the Short Packet (SP) bit in the Endpoint Control register is set, indicating the packet in the Endpoint buffer is a short packet and also no data was received with the packet (a zero-size packet), the U2DC does not assert a DMA request for the endpoint, but only generates an interrupt to the XScale® core, if the interrupt is enabled, indicating that a short packet was received

The U2DC asserts a DMA request for an In endpoint when DME is set to 1 and there is enough space in the FIFO memory to store the amount of data specified in the DMA Descriptor length field. This ensures that there is always room in the FIFO memory to store the amount of data which the DMA tries to transfer, so the DMA does not overrun the FIFO memory. The DMA receives the request and responds to the request by loading data into the Endpoint memory using a series of 32-byte bursts. Only a single request is required to transfer all the data specified in the length field of the DMA descriptor, a request for each 32-byte burst is not required since the bursts are controlled automatically by the U2DMA. When the DMA has responded to the request, the U2DC negates the DMA request. Over the course of several DMA transfers, the DMA must load either a maximum size packet or load a short packet, and have its packet complete (U2DMACMD[PACKCOMP]) bit in the DMA descriptor set on the final transfer to enable the packet for transmission. For more information on setting the DMA channel registers and the DMA descriptors, refer to Section 2.7.2.

2.7.3.6 USB Configurations and Interfaces

Figure 29 shows the configurations, interfaces, and alternate interface settings that are available in the U2DC. Configuration 0 has a fixed FIFO memory space allocation of 128 bytes used for Endpoint 0 and is the only configuration available on the USB until the U2DC has been enumerated by the USB host controller. The Endpoint memory allocations for configurations 1-15 are programmable.

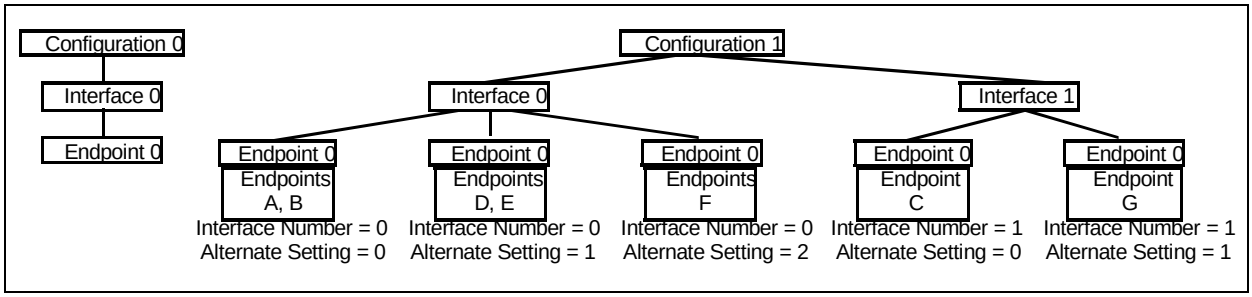
Figure 29: Configurations, Interfaces and Alternate Interface Settings for U2DC



NOTE: At any given time, only *one* configuration, and only one alternate setting per interface may be active. Interfaces within a configuration are all always simultaneously active with *one* and only one of their alternate settings.

Figure 30 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 physical endpoints from Endpoints. Each programmable Endpoint can be assigned *one* configuration, interface, and alternate interface setting at a time. These endpoints can be assigned any USB endpoint number, type, direction, and maximum packet size with the total allocated FIFO memory space for each active configuration and its active alternate interface settings not to exceed the maximum number of bytes. Configurations that are not ever selected, or combinations of alternate interfaces which are never active at the same time are not checked for FIFO memory space, and do not cause any type of error if they were to exceed the maximum number of bytes, as long as they are never activated by the USB host.

Figure 30: Example of Two U2DC Configurations



2.7.3.7 USB Configuration and Interface Changes

When the USB host controller executes a Set_configuration or Set_interface command, and if the Configuration Change Interrupt (IECC) is enabled, an interrupt is generated (IRCC bit in the U2DISR) to notify users of the change. If the interrupt is not enabled, poll the Configuration Change (CC) bit in the U2DCR. Perform the following steps to service the configuration change:

1. Read the ACN, AIN, and AAISN fields in the U2DCR register to determine the new configuration or interface changes requested by the USB host. Save these value for later comparison.
2. Stop the DMA channels for any endpoints that are going to affected.
3. Unload any data still in the Endpoint memory that software needs for any affected endpoints -- if the Endpoint memory is reallocated later in these steps, it is possible that user access may be lost to data currently in the Endpoint memory.

Note

All endpoints are affected by a Set_configuration command; however, for a Set_Interface command some of the existing endpoints may not be affected and care must be taken to keep their Endpoint memory intact. If memory endpoint re-allocation is required, the software should set up the endpoint buffer sizes in the U2DCRx registers so that the Endpoint memory of the unaffected endpoints is re-allocated to the the exact same location as before. This setup and re-allocation may not always be possible, but the chances are increased with prior planning of the potential configurations.

4. Reconfigure the U2DC endpoint and DMA registers to match the new configuration received from the USB Host. This reconfiguration includes programming the U2DC Endpoint Information Registers (U2DENx), U2DCSRx, U2DCRx, and DMA channel registers. At a minimum, clear the EE bit in the U2DCRx registers for any endpoints that are no longer valid for the current interface or configuration settings. Do not set the DMA Run bit until after the configuration change is complete.

5. The Endpoint memory must be re-allocated if the buffer size (BS) field in any of the U2DCRx registers was changed. This re-allocation is done by setting the SMAC bit to 1 and should be completed after all of the register changes have been made as the last step prior to completing the configuration change.
6. After setting the SMAC bit to 1, read the U2DCR register and ensure that the UDE bit is still 1 and that the EMCE bit is 0. In addition, check that the ACN, AIN, and AAISN fields match the field values read in Step #1 above.
   - If any of the ACN, AIN, or AAISN fields do not match the values read in Step #1, the USB Host has sent another Set_configuration or Set_interface command before the current one was completed. Use these current ACN, AIN, and AAISN values and return to Step #1 and repeat the steps to update the endpoint settings.
7. To complete the configuration change, the Config Change Interrupt bit (IRCC) must be cleared in the U2DISR register. Clear the Configuration Change bit (CC) bit in the U2DCR register if polling is being used instead of interrupts. Once this is done, the USB Interface responds to the USB Host Get Status command with an ACK instead of a NAK.
8. Finally, the U2DCR register must be read one more time and the ACN, AIN, and AAISN fields must be compared again to ensure they match the values read in Step #6.
   - There is a small window between Steps #6 and #7 where another Set_configuration or Set_interface command could arrive from the USB host change before the first configuration change is complete. Minimize as much as possible the delay between comparing the ACN, AIN, and AAISN values in step #6 and clearing the IRCC bit (or the CC bit if polling is used) in Step #7.
   - If any of the ACN, AIN, or AAISN fields do not match, read the U2DISR register and check if the IRCC bit is set again. If polling is used, check if the CC bit is set in the U2DCR and use it instead for the next comparisons. If the IRCC bit is set, return to Step #1 and repeat the steps to update the endpoint settings. If the IRCC bit is not set, the new Set_configuration or Set_interface command arrived during the small window between steps #6 and #7. The recommended way to recover from this rare situation is to clear the UDE bit in the U2DCR register and all other U2DCRx, U2DCSRx, and DMA registers and set the UDE bit back to 1. This resets the USB Interface and causes the USB Host to see a device disconnect followed by a device connect. The host then re-enumerates the U2DC.
9. If using the Configuration Change interrupt (IRCC) instead of polling, Marvell recommends clearing the Configuration Change (CC) bit in the U2DISR once all previous steps are completed successfully. Both bits will be reset when the next Set_configuration or Set_interface command arrives from the USB host.

2.7.3.8 USB Configuration Example

Table 41 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, with a maximum packet size of 1023 bytes. Endpoint B is programmed to be a bulk Out endpoint, assigned Endpoint number 2, and a maximum packet size of 512 bytes. Endpoint C is programmed to be a bulk In endpoint with Endpoint number 3, and a maximum packet size of 512 bytes. Endpoint D is programmed to be an isochronous In endpoint with an Endpoint number 4, a maximum packet size of 387 bytes. Finally, Endpoint E is programmed to be an interrupt In endpoint with an Endpoint number 5 and a maximum packet size of 16 bytes.

The actual FIFO space allocated includes overhead of eight bytes per packet of storage, which accounts for the 64 bytes difference between the FIFO memory space allocated for data and the total.

Program the total FIFO depth for each endpoint. The U2DC automatically inserts the overhead of the Packet Delimiters between each packet; however, be aware of this inherent overhead when budgeting the memory between endpoints. Allocate twice the max packet size *plus an additional 16*

bytes for Packet Delimiters to provide storage for two full packets of data storage. Also any packet that is not 64-bit aligned consumes data space rounded up to the next eight bytes. So, in this example, the 387-byte packet of Endpoint D would be allocated 392 bytes of FIFO memory to store the data, and the extra five bytes would be unused. An additional eight bytes would be needed to store the Packet Delimiter, so the minimum FIFO space allocated for this endpoint would have to be at least 392+8= 400 bytes. This minimum value is required because for In endpoints a full packet must be stored before starting the transfer over the DMA since the packet may be corrupted and the full packet must be received before it is checked for errors and may have to be re-transmitted. For Out endpoints, a full packet must be in local memory before starting the USB transfer so that it can be guaranteed that the full packet is available for transfer.

In this example, space was allocated for exactly two packets, so 800 bytes were allocated; however memory allocation is *not* required to be a multiple of max packet size. It is acceptable to specify any FIFO depth greater than the minimum size for any endpoint. For Endpoint D, if 784 bytes is allocated for the total FIFO depth instead of 800 byes, this FIFO depth would provide buffering for 1.96 packets of data and is an acceptable setting. For In endpoints, it is preferable (but not required) to have at least two packets of buffering in high-speed mode to allow the U2DC to receive one full packet and potentially have additional space to store a second packet. If storage for a second packet is available after completing the first, the U2DC can avoid having to generate a not-yet (NYET) handshake on the USB bus. A NYET handshake would require the host to verify that buffer space is available before the next transfer, causing additional overhead of a ping handshake before the next transfer.

Table 41: Maximum Packet Size Example

Endpoint	Endpoint Number	Endpoint Type	Endpoint Direction	MPS (Bytes)	FIFO Memory Space Allocated for Data (bytes)	Total Memory Space Allocated (Bytes)
A	1	Isochronous	Out	1023	2048	2064
B	2	Bulk	Out	512	512	520
C	3	Bulk	In	512	1024	1040
D	4	Isochronous	In	387	784	800
E	5	Interrupt	In	16	16	24
TOTAL					4384	4448

NOTE: Memory allocation is determined by settings in the endpoint configuration registers (U2DCRx) and must have compatible settings in the DMA descriptors when referenced by the DMA.

2.7.3.9 Configuring the Endpoints

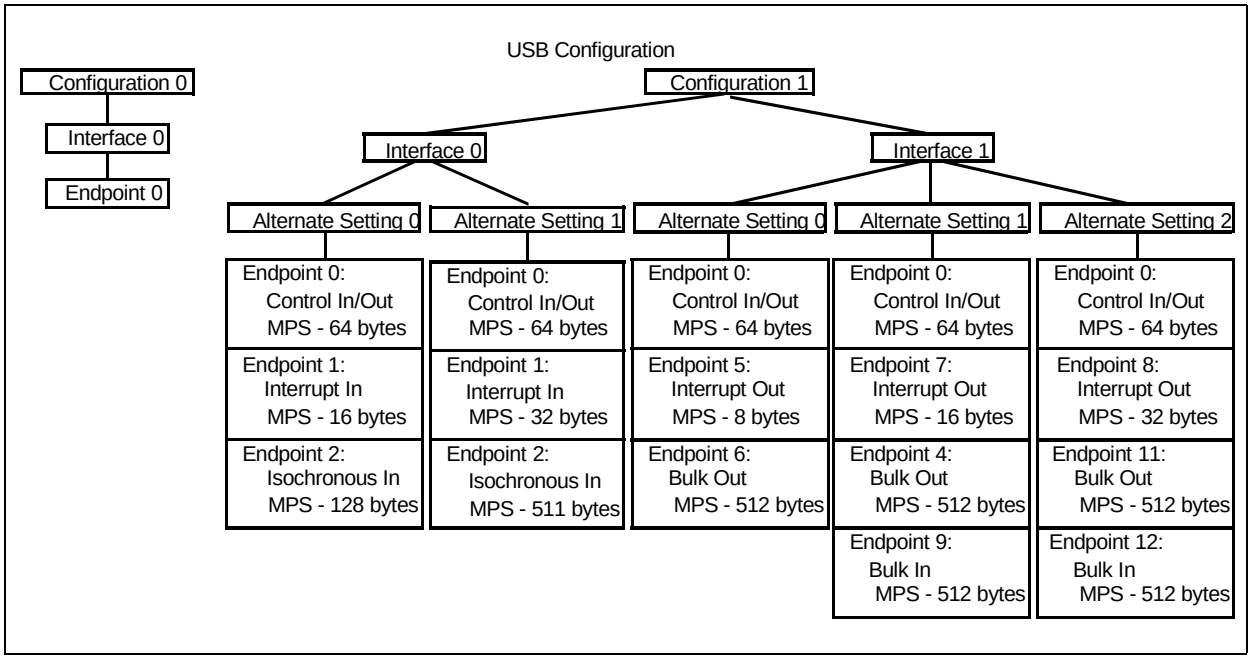
The programmable endpoints can be configured to respond as any USB Endpoint number: 1 -15, type: isochronous, interrupt or bulk, and direction: In or Out. All programmable endpoints must be configured after enabling the U2DC for USB operation. Configuring a programmable endpoint defines the configuration and interface where the endpoint is active, the USB endpoint number, type and direction, and the maximum packet size and amount of buffering to be used for storing the endpoint data. The U2DC endpoint configuration can be changed after the U2DC has been enabled for USB operation, but only after a Set_configuration or Set_interface command from the USB host is active and before SMAC is set by a user. This operation allows the physical endpoints to be

reassigned to the current configuration and alternate interface settings to limit the total number of physical endpoints required.

Configuration settings written must conform to the device descriptors reported to the USB host during the last Get_descriptor command. This simply means that the configuration settings for the endpoints should not be changed arbitrarily, but only be updated to assign a mapping between physical channels and the newly selected alternate interface or configuration. After the configuration registers are updated, the DMA must also be set up for DMA enabled endpoints before traffic can begin. This setup does not have to happen before SMAC is set, but Marvell recommends it to prevent a FIFO overflow/underflow from occurring before the DMA is enabled.

Figure 31 is an example configuration setting that could be used for the U2DC. In this example, the U2DC 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 that are used to implement that interface. The Endpoint Configuration and Information registers are used to configure each U2DC Endpoint for its use within the configuration.

Figure 31: Example USB Configurations for U2DC



NOTE: At any given time, only *one* configuration, and only *one* alternate setting per interface may be active. Interfaces within a configuration are all always simultaneously active with one and only one of their alternate settings.

Table 42 lists each of the USB physical endpoints and the U2DC programmable endpoint assigned to implement it, as well as the U2DC programmable endpoints that are not yet bound to physical endpoints in the default (alternate interfaces = 0) setup. This example offers only one possible mapping; other mappings are equally valid. Each physical U2DC endpoint does not require a unique USB endpoint number (USB Endpoint numbers 1 and 2 have been assigned to two different physical sets of endpoints) as long as duplicated USB endpoints exist in mutually exclusive alternate interface settings. Within each configuration, each USB endpoint number may have one In or one Out endpoint assigned to it. For example, Configuration 1 may have one endpoint assigned to

Endpoint 1 as an In endpoint or one endpoint assigned to Endpoint 1 as an Out endpoint. Having both In and Out endpoints within the same configuration assigned to the same endpoint number is not allowed.

Table 42: U2DC Endpoint Configuration for Example USB Default Configuration

USB				U2DC			
Config Number	IF Number	Alt. IF Setting	EP Number	EP	Config Register	Max Packet Size	FIFO Memory Allocated
All	All	All	0	0	-	64	128
1	0	0	1	A	U2DCRA	16	24
1	0	0	2	B	U2DCRB	128	264
1	0	1	1	C	U2DCRC	32	40
1	0	1	2	D	U2DCRD	511	1032
1	1	0	5	E	U2DCRE	8	16
1	1	0	6	F	U2DCRF	512	1032
1	1	1	7	-	-	16	0
1	1	1	4	-	-	512	0
1	1	1	9	-	-	512	0
1	1	2	8	-	-	32	0
1	1	2	11	-	-	512	0
1	1	2	12	-	-	512	0
-	-	-	-	G	U2DCRG	-	-

Note

During the current configuration, endpoints not bound to a physical endpoint have no FIFO memory allocated to them since the endpoints are not active. While physical Endpoints C and D are not active because their alternate interface is not selected, they are still allocated FIFO memory since they are enabled and bound to a USB endpoint.

Table 43 shows the Endpoint memory allocation for each set of possible configuration and combination of alternate interface settings. Set vii with Configuration 1, Alternate Interface 1 of Interface 0, and Alternate Interface 2 of Interface 1 allocates the most Endpoint memory space, using a total of 3336 bytes. In this case, all sets of possible configuration/alternate interfaces had less than the total memory allocated, but this is not required. If some sets of combinations of alternate interfaces would result in larger than the memory space being allocated, the only requirement is that those set are never active.

Table 43: Endpoint Memory Allocation by Configuration

Set	USB				U2DC		
	Config Number	Interface Number	Alternate Interface Setting	Endpoint	Active Endpoint	Endpoint Allocated Memory (Bytes)	Total EP Allocated Memory (Bytes)
i	0	0	0	0	0	128	128
ii	1	-	-	0	0	128	1464
	1	0	0	1	A	24	
				2	B	264	
	1	1	0	5	E	16	
				6	F	1032	
iii	1	-	-	0	0	128	2248
	1	0	1	1	C	40	
				2	D	1032	
	1	1	0	5	E	16	
				6	F	1032	
iv	1	-	-	0	0	128	2505
	1	0	0	1	A	24	
				2	B	264	
	1	1	1	7	E	24	
				4	F	1032	
				9	G	1032	
v	1	-	-	0	0	128	3288
	1	0	1	1	C	40	
				2	D	1032	
	1	1	1	7	E	24	
				4	F	1032	
				9	G	1032	

Table 43: Endpoint Memory Allocation by Configuration (Continued)

Set	USB				U2DC		
	Config Number	Interface Number	Alternate Interface Setting	Endpoint	Active Endpoint	Endpoint Allocated Memory (Bytes)	Total EP Allocated Memory (Bytes)
vi	1	-	-	0	0	128	2552
	1	0	0	1	A	24	
				2	B	264	
	1	1	2	8	E	72	
				11	F	1032	
				12	G	1032	
vii	1	-	-	0	0	128	3336
	1	0	1	1	C	40	
				2	D	1032	
	1	1	2	8	E	72	
				11	F	1032	
				12	G	1032	

The endpoints can be programmed to support USB endpoints throughout the 15 possible configurations, 15 possible interfaces, and 15 possible alternate interface settings. The USB Endpoint Configuration registers can also be programmed and, when finished, must have the SMAC bit set. Then, the U2DC memory allocation is checked for the active configuration and alternate interfaces and if the allocated memory space is valid the U2DC is enabled for USB operation. If the allocated memory space is not valid, an interrupt is generated to the core if the interrupt is enabled, and the U2DC continues to NAK any accesses on endpoints other than Endpoint 0 on the USB bus. No error or stall is signalled to the USB host.

2.7.4 PXA32x processor and PXA30x processor Cable Attach and Detach

The USB host controller provides +5 V on the USB cable to be compliant with the *Universal Serial Bus Specification, Revision 2.0*. Since the 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 wake-up enabled GPIO pin which will send an interrupt to software. When the GPIO indicates an Attach event, software must enable the U2DC by first setting the Control and Configuration registers, and then setting U2DCR[UDE]. When a Detach event is detected, data is unloaded from Endpoint memory first and U2DC is disabled by clearing U2DCR[UDE].

Note

The software must always clear all Endpoint Enable (EE) bits in the U2DC Endpoint Configuration registers (U2DCRx) whenever UDE is set to 0. The Endpoint Enable bit for each endpoint should be set only to 1 again after setting UDE to 1 and re-programming that endpoint's U2DC Endpoint Information Registers (U2DENx).

2.7.5 Suspend and Resume

The U2DC starts the process of entering the Suspend state when the U2DC detects an idle bus condition for more than 3ms. If the processor does not enter S2/D3/C4 power mode, the state of the U2DC is preserved, and ready for resume detection.

The U2DC can exit Suspend in three ways: Resume initiated by the U2DC, resume initiated by the USB host controller, or USB reset. If the USB host controller has (1) executed the Set_feature command and (2) enabled the device remote wake-up feature of the U2DC, once the U2DC has entered the Suspend state, software can signal a wake-up to the USB host controller by setting the U2DCR U2DC Resume (UDR) bit to 1. The U2DC then forces the PHY to drive a K-state onto the USB for 3 ms without additional software intervention. Additionally, the U2DC clears the UDR bit. The U2DC waits for the Resume signal to be reflected to it by the USB host controller (1) when the Resume state is detected on the USB, if the Resume Interrupt is enabled, and (2) an Interrupt is sent to the core. Software should take the appropriate action to resume activity.

The USB host controller can wake up the U2DC by driving the K-state either by forcing a Resume or USB Reset onto the USB. When the PHY detects this K-state on the USB, it starts the clock to the U2DC and if the Resume Interrupt is enabled, an interrupt is sent to the core. Software should take the appropriate action to enter resumed activity.

Note

The U2DC does not grant the PMU low-power mode request until it has gone to suspend mode. The U2DC DMA relies on a long idle time that is required on the USB bus before Suspend mode is entered, and it does not interact with the low-power mode grant mechanism. As a result, no U2DC DMA transactions should be running when the U2DC is in Suspend mode, and the DMA must be stopped. The software must always clear all Endpoint Enable (EE) bits in the U2DC Endpoint Configuration Registers (U2DCRx) whenever UDE is set to 0. The Endpoint Enable bit for each endpoint should be set to 1 again after the software has set UDE to 1 and re-programmed that endpoint's U2DC Endpoint Information Registers (U2DENx).

2.7.5.1 Low Power Mode Operation

Refer to the "Clocks Controller and Power Management" chapter in *PXA3xx Processor Family Vol. I: System and Timer Configuration Developers Manual* for more details on power modes. The processor may be awakened from low-power mode via a USB reset, USB resume, OTG event (PXA31x processor only), or generic wakeup event on one of the generic GPIOs.

Note

The U2DC retains internal state during S0/D1/C2 and S0/D2/C2 power mode and does not require programming or enumeration after exiting and returning to S0/D0/C0. However, the FIFO is not state retentive during any low power mode so software must service the data in the FIFO prior to entering low power mode.

2.7.5.1.1 S2/D3/C4 Mode Operation

If the U2DC has entered the Suspend state before the processor is in S2/D3/C4 power mode, the PHY detects the Resume state on the USB and resumes operation while the processor is in S2/D3/C4 mode. If the USB host controller tries to access the U2DC when the processor is in S2/D3/C4 power mode, the PHY detects the Resume state and signals the processor wakeup controller to begin the wakeup sequence. If the U2DC has been disconnected from the USB and the processor is in S2/D3/C4 mode, the PHY detects connection to the USB and signal the processor wakeup controller to begin the wakeup sequence.The U2DC will have lost all state information and software must reload the U2DC Configuration registers and enable the U2DC before the U2DC can be ready for USB operation. Software should expect the host to issue a USB reset and re-enumerate the U2DC.

Always enable all wakeup sources for S2/D3/C4 in S0/D0/C0 prior to any low-power mode entry.

2.7.6 Initialization of U2DC through Software

The following is an ordered list of the steps required to prepare the U2DC for operation.

1. System reset (must be asserted for at least 200 microsecond when using a PHY with oscillator and asserted at least 1.9 millisecond when using a PHY with a crystal).
2. When using the ULPI interface: (PXA31x processor only).
3. Program the U2DC OTG Control Register (U2DOTGCR) (PXA31x Processor Only) with the following values:
   - U2DOTGCR[ULAF] = 1 to select the ULPI alternate function in the padring
   - U2DOTGCR[SMAF] = 0 to de-select the serial mode alternate function in the padring
   - U2DOTGCR[CKAF] = 0 to de-select the serial mode alternate function in the padring
   - U2DOTGCR[UTMID] = 1 to disable the U2DC ULPI interface
4. Set the ULE bit in the U2DC OTG Control Register (U2DOTGCR) (PXA31x Processor Only) to enable the ULPI interface. Do this in a separate register Write to the U2DOTCR than the previous writes to ensure that the GPIO pads are configured correctly before the ULPI interface is enabled.
5. Enable all rise/fall interrupts in the U2DOTGICR register.
6. Wait for one or more OTG interrupts to fire and then determine if the ULPI interface should be configured for device or host operation. For USB OTG host operation, see Section 2.8.2.2; otherwise, complete the following steps for USB OTG device operation:
7. Clear the UTMID bit in the U2DOTGCR register to enable the U2DC ULPI interface.
8. Write preferred configuration into U2DC Interrupt Control Register (U2DICR), U2DC Interrupt Control Register 2 (U2DICR2) (PXA31x Processor Only) (PXA31x processor only), and U2DC Endpoint Configuration Registers (U2DCRx)
9. Program U2DC Control Register (U2DCR) with these following bit values:
   - U2DCR[SMAC] = 1 to perform FIFO memory allocation
   - U2DCR[DRWE] as required for device remote wakeup capability

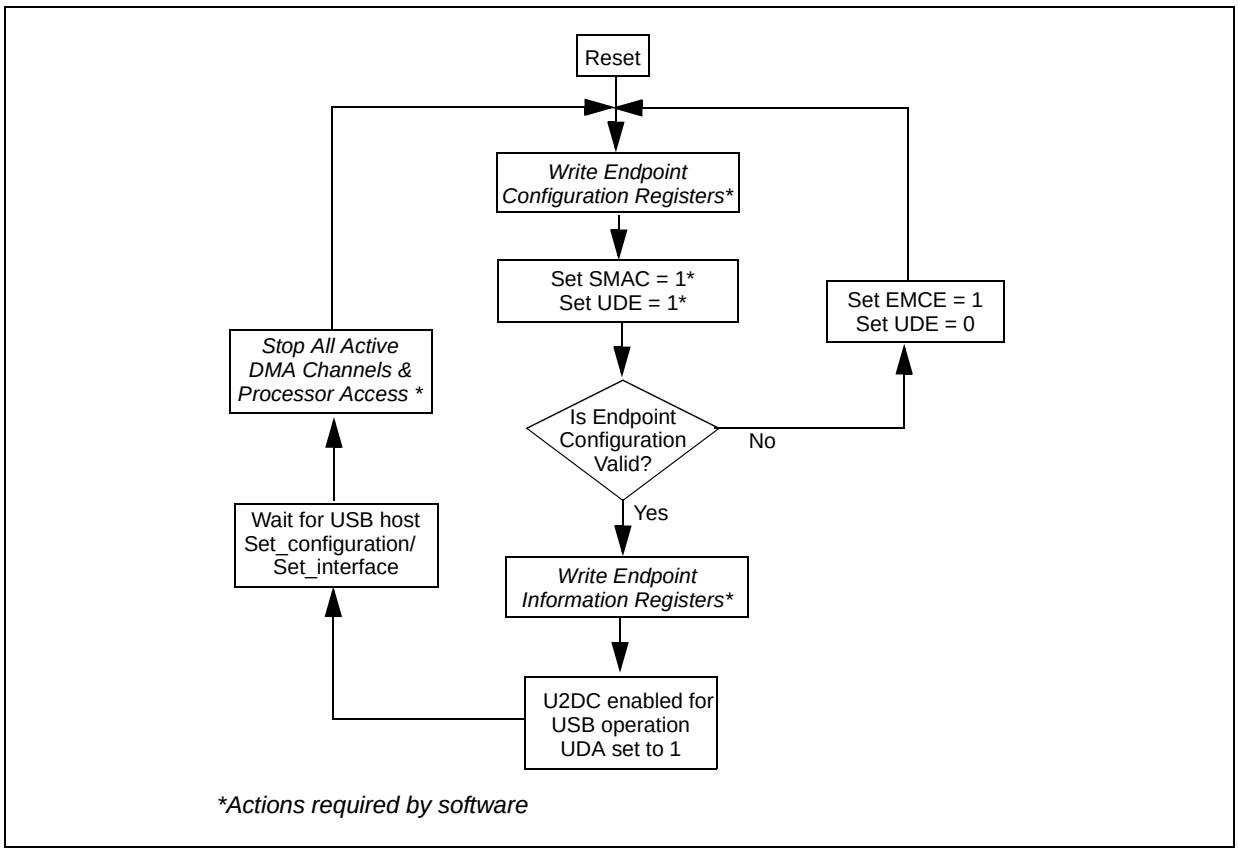
- U2DCR[UDE] = 1 to enable communication with the USB bus. Initially, only the default configuration with Endpoint 0 can be used because endpoint information registers could not be set while U2DCR[UDE]=0.

10. Write required configuration into U2DC Endpoint Information Registers (U2DENx)
11. USB traffic may or may not start enumeration on Endpoint 0; this is handled without need for software interaction.
12. The USB host may or may not apply the Get Descriptor command to Endpoint 0. This command is routed into the Endpoint 0 space in the FIFO without the need to write the Endpoint 0 information register.
13. The USB host may or may not attempt additional Endpoint 0 transactions, which are NAK'd by the U2DC core until the Endpoint 0 FIFO has been read out by the system.
14. Write required configuration into U2DC Endpoint Information Registers (U2DENx).
15. The U2DC is now initialized, potentially with the Get Configuration command waiting in the FIFO in Endpoint 0 space, which can be checked by viewing the appropriate bits in the U2DCSR0 register.

Note

Endpoint Information registers, which hold the characteristics of the endpoints, cannot be written until the USB device is enabled. But as soon as the device is enabled, transactions can start to occur on the USB. Following this sequence results in reliable startup of operation.

Figure 32: FIFO Memory and Endpoint Configuration Sequence



2.7.6.1 Programming for Low Power Mode Setup

1. The GPIOs must also be programmed with pullup/pulldown, drive strength, sleep control and data, edge type detecting and alternate function. To ensure that the USB_RESET is asserted to the PHY during any power mode, software must set bit 8 (sleep_data) to 0 in the Multi-Function Pin register configured as USB_RESET.
2. The appropriate U2DC wakeup sources must be enabled as defined in the Application Subsystem Wake-up from D3 Enable Register (AD3ER), Application Subsystem Wake-Up from D2 to D0 State Enable Register (AD2D0ER), and Application Subsystem Wake-Up from D1 to D0 State Enable Register (AD1D0ER).
3. Software should enable the Suspend Interrupt Request bit setting the IESU in the U2DC Interrupt Control Register (U2DICR) to generate an interrupt when the U2DC has been suspended.

Refer to the “Clocks Controller and Power Management” chapter in *PXA3xx Processor Family Vol. I: System and Timer Configuration Developers Manual* for a more complete detailed description on requirements for wakeups.

2.8 PXA31x Processor USB On-The-Go Operation

The PXA31x 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*[1]. The host can operate at low speed (LS) or full speed (FS) while the device can operate at full speed (FS) or high speed (HS). OTG operation requires software intervention to

program registers and monitor interrupts from the off-chip OTG PHY. The interface to the off-chip PHY is the 12-pin ULPI standard as specified in the *UTMI+ Low Pin Interface (ULPI) Specification*[1]. All host and device OTG traffic passes through the same ULPI interface to the off-chip ULPI OTG PHY. Support for the OTG features has been added to the U2DC and includes the following:

- Decoding of Set_Feature commands in the U2DC with OTG specific selector values
- Control register bits for controlling the GPIO pad multiplexing between the U2DC and USB host controller (UHC) Port 2 to the off-chip ULPI PHY
- Control, status, and interrupt registers for interfacing to and programming the ULPI registers in off-chip ULPI PHY

2.8.1 OTG Overview

The PXA31x processor OTG system consists of a USB FS/LS host controller (UHC), a USB 2.0 FS/HS device controller (U2DC), a ULPI interface, OTG control and interrupt registers, I/O mux and padring logic to switch between the U2DC and UHC paths, and an off-chip ULPI OTG transceiver (PHY). These each are described below:

Host Controller (UHC) - The UHC is a full host controller with FS/LS capability and two root hub ports. One host port is connected to a dedicated USB 1.1 port while the other is muxed with the U2DC path to create an OTG system. Both ports use the single-ended (serial) protocol.

Device Controller (U2DC) - The U2DC is HS/FS-capable which uses a UTMI interface that is converted to ULPI by the ULPI interface. It can operate as a standalone HS device or be muxed with the UHC path to create an OTG system.

OTG Registers - Located in the U2DC, these registers allow the software to monitor and control the OTG features of the off-chip PHY. The registers can also be used for indirect access of the ULPI registers in the off-chip PHY.

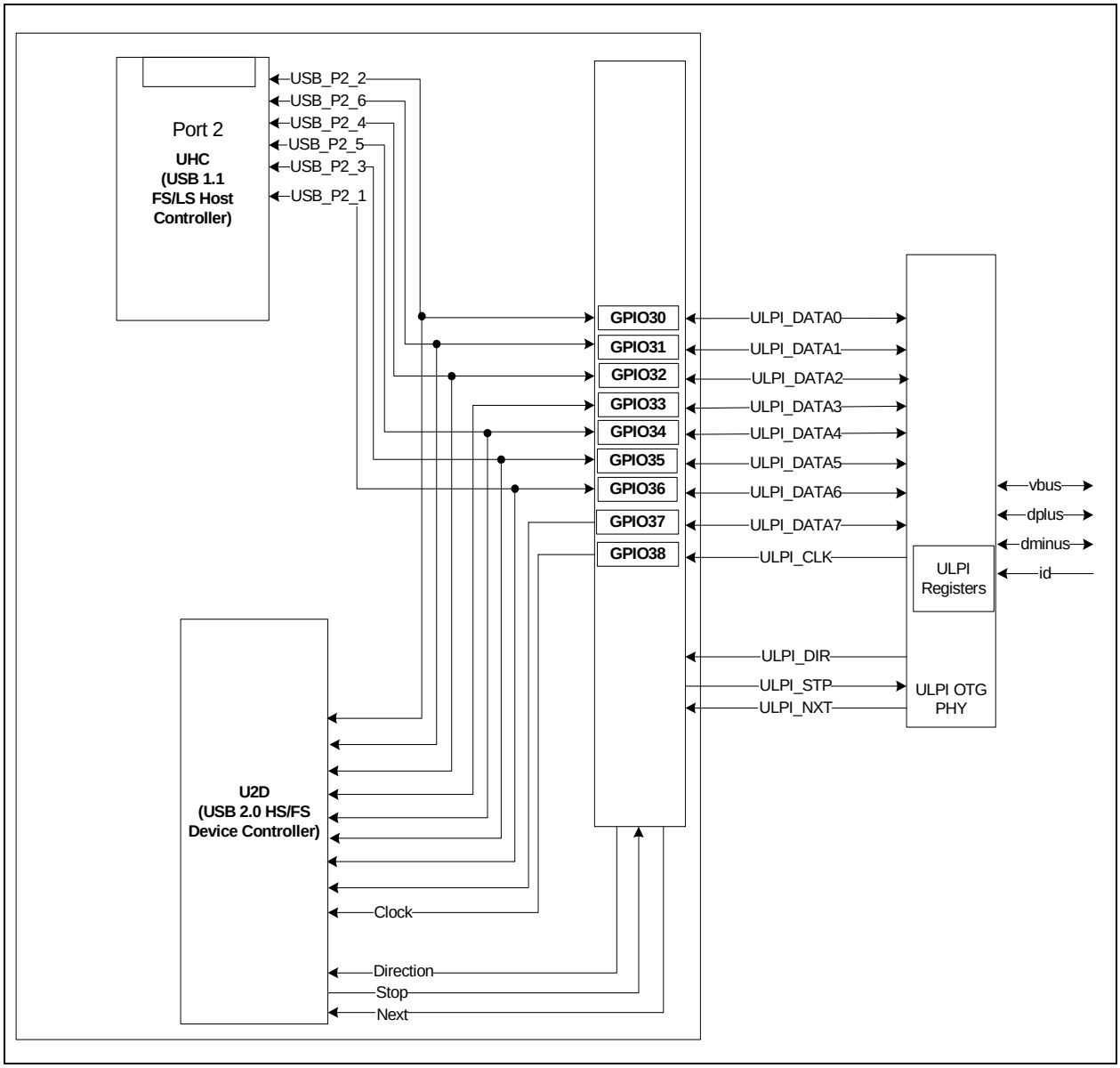
I/O Mux and Padring - Location of the muxing between the UHC and U2DC data paths for the OTG system. The muxing is controlled by alternate function select bits located in the U2DOTGCR register in the U2DC. Since the UHC uses the optional serial mode supported by the ULPI protocol, the PHY must detect ahead of time which interface mode to be in before switching between the UHC and U2DC data paths.

Off-Chip ULPI OTG PHY - Contains the ULPI registers and must support the optional 6-pin serial mode. Implements the OTG features for detecting events such as vbus_valid, session_valid, HNP, SRP, etc. The PHY provides status back to the U2DC OTG registers.

Figure 33 shows a block diagram the USB OTG System.

1. The latest revision of the *Universal Serial Bus Specification Revision 2.0* can be accessed via the World Wide Web Internet site at: http://www.usb.org/
1. The latest revision of the *UTMI+ Low Pin Interface (ULPI) Specification Revision 1.1* can be accessed via the World Wide Web Internet site at: http://www.ulpi.org/

Figure 33: ULPI OTG System Block Diagram



2.8.2 OTG Configuration

The PXA31x processor can be set up to operate in one of the following OTG conditions:

- OTG device (HS or FS)
- OTG host (FS or LS)

The following sections describe how to configure the PXA31x processor for the each of the OTG situations. In all cases, the software must monitor the OTG interrupts, determine what action to take, and then update the appropriate configuration registers.

Software must enable the following OTG interrupts in the U2DOTGICRTo detect a change in whether a mini-A or mini-B cable is plugged in:

- IFEID and IREID (ID rise and fall interrupt enables)
- IFESE and IRESE (Session_end rise and fall interrupt enables)
- IFESV and IRESV (Session_valid rise and fall interrupt enables)
- IFEVV and IREVV (Vbus_valid rise and fall interrupt enables)

2.8.2.1 OTG Device (HS/FS)

Software follows these steps to configure the PXA31x processor as an OTG device:

1. Return the ULPI PHY to synchronous mode if it is not already (see Section 2.8.4.1).
2. Ensure the OTG interrupts are enabled in the U2DOTGICR.
3. Write the U2DOTGUCR to turn off the DP/DM pulldowns in the ULPI PHY (that is, write a value of 0x030A0000 to this register). Wait for U2DOTGUCR[RUN] == 0.
4. Write the U2DOTGCR[UTMID] = 0 to enable the UTMI interface of the U2DC.
5. Write the U2DCR[UDE] = 1 to enable the U2DC and then program the U2DC registers for device operation as described in Section 2.7.6.
6. Watch for any OTG interrupts that indicate a change in the cable or the arrival of a Set_feature command that enables the U2DC to do HNP.

2.8.2.2 OTG Host (HS/FS)

Software follows these steps to configure the PXA31x processor as an OTG host:

1. Return the ULPI PHY to synchronous mode if not already (see Section 2.8.4.1)
   - Write the U2DCR[UDE] = 0 to disable the U2DC.
   - Write the U2DOTGCR[UTMID] = 1 to disable the UTMI interface of the U2DC.
   - Make sure the OTG interrupts including the serial interrupt (ISSI) are enabled in the U2DOTGICR.
2. Write the U2DOTGUCR to enable the following bits in the ULPI OTG control register in the PHY. Wait for U2DOTGUCR[RUN] == 0.
   - Bit 1 (DpPulldown) = 1 to enable the 15K Ohm pull-down resistor on D+.
   - Bit 2 (DmPulldown) = 1 to enable the 15K Ohm pull-down resistor on D-.
   - Bit 5 (DrvVbus) = 1 to cause 5V to be driven on Vbus.
3. Write the U2DP3CFG fields with the following:
   - P2SS = 0 to configure the internal UHC interface for 6-pin serial mode.
   - P2SS = 1 to configure the internal UHC interface for 3-pin serial mode.
4. Write the U2DOTGUCR to enable either 6-pin or 3-pin serial mode in ULPI PHY Interface. control register. Wait for U2DOTGUCR[RUN] == 0.
   - Bit 0 (6-pin FsLsSerialMode) = 1 to set the ULPI PHY to 6-pin serial mode.
   - Bit 1 (3-pin FsLsSerialMode) = 1 to set the ULPI PHY to 3-pin serial mode.
5. Write the U2DOTGCR fields with the following values:
   - ULAF = 0 to disable the ULPI padring interface to the internal ULPI interface.
   - SMAF = 1 to enable the ULPI padring interface to the internal UHC port 2 connection.
   - CKAF = 0 to disable the ULPI padring interface to the internal UART carkit connection.
6. Enable the UHC Port 2 interface and commence initializing the USB connection.
7. Watch for a serial mode interrupt (U2DOTGISR[ISSI]) to indicate a change in Vbus_valid, Session_valid, Session_end for catastrophic events such as overcurrent or a disconnect. To determine the cause of the interrupt, the software returns the PHY to synchronous mode.

2.8.3 Off-Chip ULPI OTG PHY Operation

The PXA31x processor requires the use of an off-chip USB OTG ULPI PHY with the following features:

- ULPI 12-pin interface (8-bit data) capable of OTG for both host and device operation
- 1.8V pin voltage
- Support for Synchronous (ULPI), 3-pin or 6-pin serial, and low-power modes

Control of the off-chip OTG Transceiver is managed by the ULPI interface.

At startup (power-on reset), the PXA31x processor holds the STP output pin high to protect the PHY from interpreting glitches on the data pins as commands or data. Once the power-on reset is complete and the ULPI data pins are stable, the STP pin is driven low when the software writes the ULE (or UDE) bit to 1. This process allows allow communication to begin, and the PHY starts sending RXCMDs.

2.8.4 ULPI Operation Modes

The USB OTG system can operate in more than one mode and relies on the software to detect when to switch and then perform the switch between each mode.

2.8.4.1 Synchronous (ULPI) Mode

Synchronous mode is the default mode of the off-chip ULPI PHY when it powers up. It is the mode that is used when the U2DC is enabled and active and it is also the only mode in which the software can access the ULPI registers in the PHY.

The ULPI PHY continuously sends RXCMDs to the PXA31x processor in this mode. The OTG related contents of these RXCMDs are reflected in the U2DOTGUSR register and can result in interrupts being set in the U2DOTGISR.

Software follows these steps to return to synchronous mode from one of the other modes (serial, low-power, carkit):

1. Ensure the U2DOTGCR[UTMID] bit = 1 to disable the UTMI interface from the U2DC.
2. Enable the OTG interrupts in the U2DOTGICR.
3. Write the U2DOTGCR[RTSM] bit = 1 to signal the ULPI PHY to return to synchronous mode.
4. Set the U2DOTGCR with these following values:
   - U2DOTGCR[ULAF] = 1 to select the ULPI alternate function in the padring
   - U2DOTGCR[SMAF] = 0 to de-select the serial mode alternate function in the padring
   - U2DOTGCR[CKAF] = 0 to de-select the Carkit mode alternate function in the padring

One of two scenarios exist while in synchronous mode:

1. The U2DC being enabled and active as an OTG device (default).
2. The U2DC being disabled.

Note

To enable the U2DC in synchronous mode, the software must set the U2DOTGCR[UTMID] bit to a zero so that the U2DC UTMI interface can control the ULPI interface and set the correct device conditions in the off-chip ULPI PHY.

Software monitors both the normal U2DC interrupts as well as the OTG interrupts when the U2DC is enabled and active. Software ensures the U2DOTGCR[UTMID] bit = 1 to prevent the U2DC from inadvertently configuring the off-chip ULPI PHY when the U2DC is disabled.

If the PXA31x processor is going to a suspend state (D1, D2, etc.), software follows these steps prior to switching to the suspend state:

1. Enables all interrupts in the U2DOTGICR
2. Enables the USB GPIO wakeups on DAT0, DAT1, and DAT3
3. Places the off-chip ULPI PHY into low-power mode as described in Section 2.8.4.3.

2.8.4.2 Serial Mode

Serial mode is used only when the UHC Port 2 is active as an OTG host. The PXA31x processor supports both 3-pin and 6-pin serial modes.

By default, the off-chip ULPI PHY shuts off its 60 MHz clock source to conserve power during Serial mode. In Serial mode, the eight ULPI data pins are re-mapped to accommodate the 6-pin or 3-pin serial interface. The mapping for the 6-pin and 3-pin serial modes is shown below in Table 44 and Table 45, respectively. An interrupt pin is mapped onto one of the ULPI data pins as a way for the PHY to indicate to the link that an unmasked interrupt occurred (host disconnect, vbus_valid, session_end, etc.). These pins can be enabled to generate a wakeup event to the PXA31x processor when it is in a non-D0 state.

Table 44: 6-pin ULPI Serial Mode Pin Mappings

Signal	ULPI	GPIO	PHY Direction	Description
tx_enable	data[0]	GPIO30	IN	Active high transmit enable
tx_dat	data[1]	GPIO31	IN	Transmit differential data on D+/D-
tx_se0	data[2]	GPIO32	IN	Transmit single-ended zero on D+/D-
interrupt	data[3]	GPIO33*	OUT	Active high interrupt indication. Must be asserted whenever any unmasked interrupt occurs.
rx_dp	data[4]	GPIO34*	OUT	Single-ended receive data from D+
rx_dm	data[5]	GPIO35*	OUT	Single-ended receive data from D-
rx_rcv	data[6]	GPIO36*	OUT	Differential receive data from D+/D-
reserved	data[7]	GPIO37	OUT	Reserved. The PHY must drive this pin to low.
**wakeup capable GPIO pins*				

Table 45: 3-pin ULPI Serial Mode Pin Mappings

Signal	ULPI	GPIO	PHY Direction	Description
tx_enable	data[0]	GPIO30	IN	Active high transmit enable
dat	data[1]	GPIO31*	IN/OUT	Transmit differential data on D+/D- when tx_enable is high Receive differential data from D+/D- when tx_enable is low

Table 45: 3-pin ULPI Serial Mode Pin Mappings (Continued)

Signal	ULPI	GPIO	PHY Direction	Description
se0	data[2]	GPIO32*	IN/OUT	Transmit single-ended zero on D+/D- when tx_enable is high Receive single-ended zero from D+/D- when tx_enable is low
interrupt	data[3]	GPIO33*	OUT	Active high interrupt indication. Must be asserted whenever any unmasked interrupt occurs.
*wakeup capable GPIO pins				

Software follows these steps to switch to serial mode:

1. Return the ULPI PHY to synchronous mode if it is not already in this mode (see Section 2.8.4.1).
2. Write the U2DP3CFG[P2SS] to select either 3-pin or 6-pin serial mode.
3. Write the U2DOTGICR[ISSI] bit = 1 to enable serial mode interrupts.
4. Write the U2DOTGUCR register with the required value to program the PHY to switch to 3-pin or 6-pin Serial mode. Wait for the U2DOTGUCR[RUN] bit = 0.
5. Set the U2DOTGCR with these following values:
   - U2DOTGCR[ULAF] = 0 to de-select the ULPI alternate function in the padring
   - U2DOTGCR[SMAF] = 1 to select the serial mode alternate function in the padring
   - U2DOTGCR[CKAF] = 0 to de-select the Carkit mode alternate function in the padring

Software follows the steps to return to Synchronous mode from Serial mode as described in Section 2.8.4.1.

The following scenarios can occur while in Serial mode:

1. **The UHC Port 2 detects a disconnect** -- the software enables OTG interrupts in the U2DOTGICR, causes the PXA31x processor to exit Serial mode and return to Synchronous mode, and then wait for a new connect event or change in the OTG interrupts.
2. **The U2DOTGISR[ISSI] interrupt occurs** -- the software enables OTG interrupts in the U2DOTGICR and causes the PXA31x processor to exit Serial mode and return to Synchronous mode. At this point, the PHY starts sending RXCMDs to the PXA31x processor with the current OTG status that is reflected in the U2DOTGUSR and U2DOTGISR. From these, the software can determine the cause of the Serial mode interrupt. Optionally, the software can read the ULPI Interrupt Latch register in the PHY to confirm the cause of the interrupt.
3. **The PXA31x processor switches to a suspend state** (D1, D2, etc.) -- prior to switching to the suspend state, the software completes all UHC Port 2 USB traffic and suspends all devices. Next, the software enables all OTG interrupts in the U2DOTGICR and then exits Serial mode and returns to synchronous mode (a USB connect event could occur during this transition). The software then places the PHY in low-power mode and, if necessary, enables GPIO wakeups on GPIO30, 31, and 33 of the ULPI interface.

2.8.4.3 Low-Power Mode

Low-power mode is used for the off-chip ULPI PHY when it is preferable to conserve power when the USB bus is USB Suspend or disconnected. In this mode, the ULPI PHY can generate a single generic interrupt whenever an unmasked OTG interrupt occurs (vbus_valid, session_end, session_valid, or id change). The software can further reduce the power consumption in this mode by disabling interrupt enables in the off-chip ULPI PHY registers to disable the comparators (for more details about this refer to the reference manual for the particular ULPI PHY being used).

By default, the off-chip ULPI PHY shuts off its 60 MHz clock source in Low-power mode. In Low-power mode, the eight ULPI data pins are remapped as shown below in Table 46. This remapping gives a real-time view of the linestate values and an interrupt pin for any unmasked interrupts. These pins can be enabled to generate a wakeup event to the PXA31x processor when it is in a non-D0 state.

Table 46: Low-Power Mode Pin Mappings

Signal	ULPI	GPIO	PHY Direction	Description
linestate(0)	data[0]	GPIO30*	OUT	Combinatorial LineState(0) driven directly by FS analog receiver.
linestate(1)	data[1]	GPIO31*	OUT	Combinatorial LineState(1) driven directly by FS analog receiver.
reserved	data[2]	GPIO32	OUT	Reserved. The PHY must drive this pin to low.
interrupt	data[3]	GPIO33*	OUT	Active high interrupt indication. Must be asserted whenever any unmasked interrupt occurs.
*wakeup capable GPIO pins				

Software follows these steps to switch to low-power mode:

1. Returns the ULPI PHY to synchronous mode if it is not already in this mode (see Section 2.8.4.1).
2. Writes the U2DOTGICR[ISSI] bit = 1 to enable low-power mode interrupts.
3. Writes the U2DOTGUCR register with the value to set the SuspendM bit in the PHY to zero. Waits for the U2DOTGUCR[RUN] bit to be zero.

Software follows the steps to return to synchronous mode from Low-power mode as described in Section 2.8.4.1.

The PXA31x processor can be in a D0-D4 state while the off-chip ULPI PHY is in low-power mode. If the PXA31x processor is in a non-D0 state, the software first enables the USB wakeups for GPIO30, 31, and 33 before going to that state. If a USB wakeup event occurs, the software must bring the PXA31x processor back to a D0 state, enable the OTG Interrupts in the U2DOTGICR, and cause the ULPI PHY to return to Synchronous mode from Low-power mode as described in Section 2.8.4.1. The software monitors the U2DOTGISR to determine the cause of the interrupt or optionally reads the ULPI Interrupt Latch register in the off-chip ULPI PHY.

The following scenarios can occur if the PXA31x processor is in D0 while the ULPI PHY is in low-power mode:

1. **The U2DOTGISR[ISSI] interrupt occurs** -- the software enables OTG interrupts in the U2DOTGICR and causes the PXA31x processor to exit Serial mode and return to Synchronous mode. At this point, the PHY starts sending RXCMDs to the PXA31x processor with the current OTG status that is reflected in the U2DOTGUSR and U2DOTGISR. From these, software can determine the cause of the low-power mode interrupt. Optionally, the software can read the ULPI Interrupt Latch register in the PHY to confirm the cause of the interrupt.
2. **The PXA31x processor switches to a Low-power mode (S0/D1/C2, S0/D2/C2, S2/D3/C4, etc.)** -- Software enables the USB wakeups for GPIO30, 31, and 33 prior to switching to Low-power mode.

2.8.5 ULPI Power States

The ULPI interface controls the off-chip PHY and controls when it enters and exits Low-power mode. When the USB OTG system is in Device mode, the U2DC automatically determines when to tell the off-chip PHY to enter and exit Low-power mode via the UTMI SuspendM signal (See Section 2.7.5.1). However, when the U2DC is disabled or the USB OTG system is configured as an OTG host, software must program the SuspendM bit in the ULPI Function Control register in the off-chip PHY using the U2DOTGUCR register (see Section 2.8.4.3).

The Table 47 shows the possible modes of the off-chip ULPI PHY depending on the power state of the PXA31x processor.

Table 47: Possible Modes of Off-Chip ULPI PHY

Power Mode	PXA3xx	ULPI PHY
S0/D0/C0	On	Synchronous (ULPI), CarKit, Serial, or Low-Power
S0/D1/C2, S0/D2/C2	Suspended (drowsy)	Low-Power, Carkit, Serial
S2/D3/C4	Sleep	Low-Power, Carkit, Serial
S3/D4/C4, S3/D4/C4	Deep Sleep, Off	Off

Before the processor goes to a D1, D2, or D3 power mode, Marvell recommends that the off-chip ULPI PHY be first be put into Low-power mode (although it can also remain in Carkit or Serial mode). Also, the appropriate GPIO wakeups must be enabled as described in the Serial, Carkit, and Low-power mode sections (see Section 2.8.4).

Software must write the RTSM bit to a 1 in the U2DOTGCR (see Section 2.8.4.3) to bring the off-chip ULPI PHY out of Low-power mode upon returning from a D1 or D2 state to D0.

A system reset is required when coming out of a D3 or D4 state, which causes the ULPI_STP pin to be reset to 1 and which automatically brings the ULPI PHY out of Low-power mode. Communication with the ULPI PHY cannot begin again until the ULE and ULAF bits are written to a 1 in the U2DOTGCR.

2.8.6 OTG Interrupts

When software receives the global U2DC interrupt, it must also access the U2DOTGISR to determine if it was an OTG interrupt and which one caused the interrupt. Where the interrupt was an ISSI interrupt (Serial, Carkit, or Low-power mode interrupt), the software must return the ULPI PHY to synchronous mode before it can either re-read the U2DOTGISR or use the U2DOTGUCR register to access the ULPI Interrupt Latch register in the ULPI PHY and determine the cause. Software must rely upon the wakeup-enabled GPIO pads to generate a wakeup to the PMU If the processor is powered down.

The software can enable and disable both rising and falling OTG interrupts in both the U2DOTGICR as well as the by using the U2DOTGUCR to program the ULPI Interrupt Enable Rising/Falling registers in the off-chip PHY. However, in general, the interrupt enable registers in the ULPI should be left with everything enabled.

If an OTG-related interrupt (change in ID or Vbus_valid) is detected during Suspend, the ULPI PHY signals an interrupt that causes the ISSI bit to be set in the U2DOTGISR register. If detected, the U2DC to ULPI interface is disabled by asserting the UTIMD bit in the U2DOTGCR register. Then the

PHY is brought out of Low-power (Suspend) mode by writing the RTSM bit to a 1 in the U2DOTGCR register. This action causes the PHY to restart the clock and send an RXCMD with the state of ID, Vbus_valid, Session_Valid, and Session_End to the ULPI interface. The ULPI interface posts the current RXCMD status to the U2DOTGUSR register, which generates rise/fall interrupts in the U2DOTGISR. The USB Interrupt Latch register in the ULPI PHY can also be read to determine why the interrupt occurred.

2.8.7 U2DC OTG Set_Feature Commands

The U2DC can selectively decode Set_Feature commands with selector values of 3, 4, or 5 to enable the U2DC to be used in OTG and non-OTG modes. 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 OTGEN bit in the U2DOTG Control Register (U2DOTGCR) is set to 1, the U2DC decodes the command, responds with an ACK on the USB, and the corresponding OTG status bit in the U2DOTGCR is set to 1 to indicate the feature has been enabled. If the USB host controller sends an OTG specific Set_Feature command and OTGEN is cleared to 0, the U2DC does not decode the command, responds with a Stall on the USB, and does not set any OTG status bits in the U2DOTGCR. The reset value for OTGEN is 0, so software must set OTGEN to 1 for OTG Set_Feature commands to be decoded. Table 48 lists the OTG features and the U2DOTGCR status bits assigned to each. When OTGEN is set to 1, read the U2DOTGCR at any time to determine if the OTG features have been enabled by the USB host controller.

Table 48: On-The-Go Feature Selectors

Feature Selector	Value	U2DOTGCR 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

2.8.8 Host Negotiation Protocol (HNP)

The Host Negotiation Protocol is used to allow dual-role devices to switch back and forth between being a host and device.

The ID pin in the cable determines the default roles of each end when an OTG cable is plugged in. The host end may enable the device to become the host if required by sending a SetFeature command to the device. The software must monitor the status of the SetFeature status bits in the U2DOTGCR as well as the OTG interrupts in the U2DOTGISR to correctly implement the HNP exchange. Refer to the OTG specification for the exact details on the exchange required for HNP.

The default roles of host and device are restored whenever a session ends (Vbus voltage falls below the session threshold). The default host always supplies Vbus regardless of whether it has relinquished its role as host to another device.

2.8.9 Session Request Protocol (SRP)

The OTG supplement allows a host to leave Vbus turned off when the bus is not being used to conserve power. The Session Request Protocol (SRP) allows a device to request that the host supply power on Vbus.

A session is defined as the period of time that Vbus is above the session valid threshold of a given device.

As defined in the OTG supplement, there are two methods to signal SRP -- (1) data-line pulsing, and (2) Vbus pulsing. The OTG supplement requires a host to respond to only one of the methods, but a device must use both methods when initiating SRP to insure that the host responds.

When acting as a device, SRP is controlled by software programming the Charge_Vbus bit in the OTG registers inside the off-chip PHY. The Session_valid interrupt is monitored for SRP when acting as a host.

When the PXA31x processor is configured as an OTG host, the software must monitor the ISSI interrupt in the U2DOTGISR to determine if an SRP condition exists. If configured as an OTG device, the software must monitor the OTG interrupts in the U2DOTGISR and use the U2DOTGUCR register to control the ULPI PHY registers to signal the SRP (Vbus and data line pulsing).

2.9 PXA31x Processor Carkit Mode (Transparent UART mode)

The PXA31x processor provides support for USB Carkit mode[1]. The USB Carkit mode or transparent UART mode is used to implement an automotive speakerphone, either through a self-contained carkit speakerphone adapter, or through a carkit-enabled car stereo. The PXA31x processor supports the Carkit mode of operation using either a standalone external carkit IC, or using a carkit-enabled ULPI PHY.

Carkit mode support requires the ability to multiplex a UART peripheral interface and an audio codec over the USB D+ and D- signal pins. The PXA31x processor UART peripheral interface can be used for this function. The UART transmit (TXD) and receive (RXD) data pins are multiplexed over the USB D- and D+ pins during Carkit UART signaling mode, respectively. The audio codec is an external component to the PXA31x processor, and its SPKR_L and SPKR_R/MIC analog signals are multiplexed over the USB D- and D+ pins during the Carkit audio modes. (Both mono and stereo audio modes are supported by the carkit standard.)

2.9.1 UART Requirements for Carkit Mode

The UART requirements to support the Carkit UART signaling mode are shown below.

- UART data rate: 9600 baud
- 8 bit format
- No parity
- One stop bit
- Assertion high polarity, i.e., logic 1 = high, logic 0 = low

2.9.2 Carkit Operation using a Standalone Carkit IC

When using a standalone external carkit IC, all of the multiplexing and control of the carkit interface is handled in the carkit interface IC. The carkit IC uses the USB D+ and D- signals (from an external ULPI PHY), the TXD and RXD UART signals from the PXA31x processor, and the SPKR_L and SPKR_R/MIC signals (from an external audio codec) to provide the carkit interface on the USB port.

2.9.3 Carkit Operation using a Carkit-enabled ULPI PHY

When using a carkit-enabled ULPI PHY, the multiplexing and control of the carkit interface between USB/UART signaling modes and the audio modes is controlled by the external ULPI PHY. The PXA31x processor controls the multiplexing between the USB and UART signaling modes based on the Carkit mode status. Table 49 shows this multiplexing required within the PXA31x processor

1. See the CEA Standard CEA-936A, USB Carkit Specification for details of the USB Carkit mode.

when using a carkit-enabled ULPI PHY. Figure 17 is an example of the implementation of carkit support using a carkit-enabled ULPY PHY and the PXA31x processor.

Table 49: Carkit Multiplexing with a Carkit-enabled ULPI PHY

Signal	ULPI	GPIO	PHY Direction	Description
txd	data[0]	GPIO30	IN	UART TXD that is routed to the D- pin
rxd	data[1]	GPIO31*	OUT	UART RXD that is routed from the D+ pin
reserved	data[2]	GPIO32	OUT	Reserved.
interrupt	data[3]	GPIO33*	OUT	Active high interrupt indication. Must be asserted whenever any unmasked interrupt occurs.
**wakeup capable GPIO pins*				

In Carkit UART signaling mode, the UART TXD signal from the PXA31x processor is routed to the ULPI PHY using the ULPI_DATA[0] pin, and the UART RXD signal is routed from the ULPI PHY using the ULPI_DATA[1] pin. Also, the carkit interrupt output (INT) from the ULPI PHY is routed to the PXA31x processor using the ULPI_DATA[3] pin. In Carkit UART signaling mode, the ULPI_DATA[3] pin is configured as an interrupt-generating GPIO. Refer to the *CEA Standard CEA-936-A, USB Carkit Specification* for details of the Carkit signaling modes.

2.9.4 Steps to Enable/Disable Carkit Mode with Carkit-enabled ULPI PHY

Below are the steps required to enter Carkit mode when the PXA31x processor is used with a carkit-enabled ULPI PHY.

1. Set the U2DOTGCR with these following values:
   - U2DOTGCR[ULAF] = 1 to select the ULPI alternate function in the padring
   - U2DOTGCR[SMAF] = 0 to de-select the serial mode alternate function in the padring
   - U2DOTGCR[CKAF] = 0 to de-select the Carkit mode alternate function in the padring
   - U2DOTGCR[UTMID] = 1 to disable the UTMI interface from the U2DC
2. Write the U2DOTGICR[ISSI] bit to a one to enable Carkit mode interrupts.
3. Write the U2DOTGCR[ULE] bit to a one to enable the ULPI interface.
4. Write the ULPI interface control register in the PHY to tell the PHY to go to Carkit mode. This is accomplished by writing the value 0x02080400 to the U2DOTGUCR register. Wait for the U2DOTGUCR[RUN] bit to be zero.
5. Set the U2DOTGCR with these following values:
   - U2DOTGCR[ULAF] = 0 to de-select the ULPI alternate function in the padring
   - U2DOTGCR[SMAF] = 0 to de-select the serial mode alternate function in the padring
   - U2DOTGCR[CKAF] = 1 to select the Carkit mode alternate function in the padring

Follow the steps for returning to synchronous mode from Carkit mode as described in Section 2.8.4.1.

Note

Carkit mode uses an internal UART in the PXA31x processor without flow control so there is no need to configure the UART for flow control.

2.10 Register Descriptions

The USB host controller controls and communicates all configuration, request/service, and status reporting to the U2DC via the USB. Several registers are available to software to control the U2DC interface to software. A Control register is used to enable the U2DC and monitor USB activity. Control registers are used to enable the various interrupt sources that exist within the U2DC. Status registers are used to 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 - G and Endpoints A - P each have one Control/Status register, one Configuration Control register, one Byte Count register.

2.11 Register Summary

Table 50 shows the registers associated with the U2DC and the memory-mapped addresses used to access them.

Table 50: U2DC Register Addresses

Physical Address	Description	Page	Notes
0x5410_0000	U2DC Control Register (U2DCR)	167	
0x5410_0004	U2DC Interrupt Control Register (U2DICR)	174	Endpoints 0, A - G
0x5410_0008	U2DC Interrupt Control Register 2 (U2DICR2) (PXA31x Processor Only)	174	Endpoints H - P
0x5410_0008	Reserved		PXA32x and PXA30x Only
0x5410_000C	U2DC Interrupt Status Register (U2DISR)	178	Endpoints 0, A - G
0x5410_0010	U2DC Interrupt Status Register 2 (U2DISR2) (PXA31x Processor)	178	Endpoints H - P

Table 50: U2DC Register Addresses (Continued)

Physical Address	Description	Page	Notes
0x5410_0014	The U2DC Interrupt Status register U2DISR contain bits that are used to generate the U2DC interrupt request. The U2DISR2 register contains interrupt status for Endpoints H - P. Each bit in the U2DC Interrupt Status registers is logically ANDed with its Interrupt Enable and then logically ORed together to produce one signal, which is logically ORed with the U2DMA interrupt and the U2DOTG interrupt to then generate a single interrupt request. Figure 34 shows the U2DC interrupt generation. When the Interrupt Service Routine (ISR) for the U2DC is executed, software must read the U2DC Interrupt Status registers and U2DMA Interrupt register, and U2DOTG Interrupt register to determine why the U2DC interrupt occurred.	182	
0x5410_0018	Reserved		
0x5410_001C	Reserved		
0x5410_0020	U2DC OTG Control Register (U2DOTGCR) (PXA31x Processor Only)	183	
0x5410_0020	Reserved		PXA32x and PXA30x Only
0x5410_0024	U2DC OTG Interrupt Control/Enable Register (U2DOTGICR) (PXA31x Processor Only)	186	
0x5410_0024	Reserved		PXA32x and PXA30x Only
0x5410_0028	U2DC OTG Interrupt Status Register (U2DOTGISR) (PXA31x Processor Only)	189	
0x5410_0028	Reserved		PXA32x and PXA30x Only
0x5410_002C	U2DC OTG ULPI Status Register (U2DOTGUSR) (PXA31x Processor Only)	190	
0x5410_002C	Reserved		PXA32x and PXA30x Only
0x5410_0030	U2DC OTG ULPI Control Register (U2DOTGUCR) (PXA31x Processor Only)	191	
0x5410_0030	Reserved		PXA32x and PXA30x Only
0x5410_0034	U2DC Host Port 3 Control Register (U2DP3CR) (PXA31x Processor Only)	192	
0x5410_0034	Reserved		PXA32x and PXA30x Only

Table 50: U2DC Register Addresses (Continued)

Physical Address	Description	Page	Notes
0x5410_0038 – 0x5410_00FF	Reserved		
0x5410_0100	U2DC Endpoint 0 Control/Status Register (U2DCSR0)	193	
0x5410_0104	U2DC Endpoint A - G Control Status Registers (U2DCSRA-G)	198	Endpoint A
0x5410_0108	U2DC Endpoint A - G Control Status Registers (U2DCSRA-G)	198	Endpoint B
0x5410_010C	U2DC Endpoint A - G Control Status Registers (U2DCSRA-G)	198	Endpoint C
0x5410_0110	U2DC Endpoint A - G Control Status Registers (U2DCSRA-G)	198	Endpoint D
0x5410_0114	U2DC Endpoint A - G Control Status Registers (U2DCSRA-G)	198	Endpoint E
0x5410_0118	U2DC Endpoint A - G Control Status Registers (U2DCSRA-G)	198	Endpoint F
0x5410_011C	U2DC Endpoint A - G Control Status Registers (U2DCSRA-G)	198	Endpoint G
0x5410_0120	U2DC Endpoint H - P Control Status Registers (U2DCSRH-P) (PXA31x Processor Only)	198	Endpoint H (PXA31x Only)
0x5410_0120	Reserved		PXA32x and PXA30x Only
0x5410_0124	U2DC Endpoint H - P Control Status Registers (U2DCSRH-P) (PXA31x Processor Only)	198	Endpoint I
0x5410_0124	Reserved		PXA32x and PXA30x Only
0x5410_0128	U2DC Endpoint H - P Control Status Registers (U2DCSRH-P) (PXA31x Processor Only)	198	Endpoint J
0x5410_0128	Reserved		PXA32x and PXA30x Only
0x5410_012C	U2DC Endpoint H - P Control Status Registers (U2DCSRH-P) (PXA31x Processor Only)	198	Endpoint K
0x5410_012C	Reserved		PXA32x and PXA30x Only
0x5410_0130	U2DC Endpoint H - P Control Status Registers (U2DCSRH-P) (PXA31x Processor Only)	198	Endpoint L

Table 50: U2DC Register Addresses (Continued)

Physical Address	Description	Page	Notes
0x5410_0130	Reserved		PXA32x and PXA30x Only
0x5410_0134	U2DC Endpoint H - P Control Status Registers (U2DCSRH-P) (PXA31x Processor Only)	198	Endpoint M
0x5410_0134	Reserved		PXA32x and PXA30x Only
0x5410_0138	U2DC Endpoint H - P Control Status Registers (U2DCSRH-P) (PXA31x Processor Only)	198	Endpoint N
0x5410_0138	Reserved		PXA32x and PXA30x Only
0x5410_013C	U2DC Endpoint H - P Control Status Registers (U2DCSRH-P) (PXA31x Processor Only)	198	Endpoint P
0x5410_013C	Reserved		PXA32x and PXA30x Only
0x5410_0140 – 0x5410_01FF	Reserved		
0x5410_0200	U2DC Byte Count register - Endpoint 0	206	
0x5410_0204 – 0x5410_02FF	Reserved		
0x5410_0300	U2DC End Point 0 Data Register (U2DDR0)	207	
0x5410_0304 – 0x5410_0403	Reserved		
0x5410_0404	U2DC Endpoints A - G Configuration Registers (U2DCRA-G)	208	Endpoint A
0x5410_0408	U2DC Endpoints A - G Configuration Registers (U2DCRA-G)	208	Endpoint B
0x5410_040C	U2DC Endpoints A - G Configuration Registers (U2DCRA-G)	208	Endpoint C
0x5410_0410	U2DC Endpoints A - G Configuration Registers (U2DCRA-G)	208	Endpoint D
0x5410_0414	U2DC Endpoints A - G Configuration Registers (U2DCRA-G)	208	Endpoint E
0x5410_0418	U2DC Endpoints A - G Configuration Registers (U2DCRA-G)	208	Endpoint F

Table 50: U2DC Register Addresses (Continued)

Physical Address	Description	Page	Notes
0x5410_041C	U2DC Endpoints A - G Configuration Registers (U2DCRA-G)	208	Endpoint G
0x5410_0420	U2DC Endpoints H - P Configuration Registers (U2DCRH-P) (PXA31x Processor Only)	208	Endpoint H
0x5410_0420	Reserved		PXA32x and PXA30x Only
0x5410_0424	U2DC Endpoints H - P Configuration Registers (U2DCRH-P) (PXA31x Processor Only)	208	Endpoint I
0x5410_0424	Reserved		PXA32x and PXA30x Only
0x5410_0428	U2DC Endpoints H - P Configuration Registers (U2DCRH-P) (PXA31x Processor Only)	208	Endpoint J
0x5410_0428	Reserved		PXA32x and PXA30x Only
0x5410_042C	U2DC Endpoints H - P Configuration Registers (U2DCRH-P) (PXA31x Processor Only)	208	Endpoint K
0x5410_042C	Reserved		PXA32x and PXA30x Only
0x5410_0430	U2DC Endpoints H - P Configuration Registers (U2DCRH-P) (PXA31x Processor Only)	208	Endpoint L
0x5410_0430	Reserved		PXA32x and PXA30x Only
0x5410_0434	U2DC Endpoints H - P Configuration Registers (U2DCRH-P) (PXA31x Processor Only)	208	Endpoint M
0x5410_0434	Reserved		PXA32x and PXA30x Only
0x5410_0438	U2DC Endpoints H - P Configuration Registers (U2DCRH-P) (PXA31x Processor Only)	208	Endpoint N
0x5410_0438	Reserved		PXA32x and PXA30x Only
0x5410_043C	U2DC Endpoints H - P Configuration Registers (U2DCRH-P) (PXA31x Processor Only)	208	Endpoint P
0x5410_043C	Reserved		PXA32x and PXA30x Only
0x5410_0440 – 0x5410_0500	Reserved		

Table 50: U2DC Register Addresses (Continued)

Physical Address	Description	Page	Notes
0x5410_0504	U2DC Endpoint 0 Information Register (U2DEN0)	209	
0x5410_0508	U2DC Endpoint A - G Information Registers (U2DENA - G)	210	Endpoint A
0x5410_050C	U2DC Endpoint A - G Information Registers (U2DENA - G)	210	Endpoint B
0x5410_0510	U2DC Endpoint A - G Information Registers (U2DENA - G)	210	Endpoint C
0x5410_0514	U2DC Endpoint A - G Information Registers (U2DENA - G)	210	Endpoint D
0x5410_0518	U2DC Endpoint A - G Information Registers (U2DENA - G)	210	Endpoint E
0x5410_051C	U2DC Endpoint A - G Information Registers (U2DENA - G)	210	Endpoint F
0x5410_0520	U2DC Endpoint A - G Information Registers (U2DENA - G)	210	Endpoint G
0x5410_0524	U2DC Endpoint H - P Information Registers (U2DENH - P) (PXA31x Processor Only)	209	Endpoint H
0x5410_0524	Reserved		PXA32x and PXA30x Only
0x5410_0528	U2DC Endpoint H - P Information Registers (U2DENH - P) (PXA31x Processor Only)	210	Endpoint I
0x5410_0528	Reserved		PXA32x and PXA30x Only
0x5410_052C	U2DC Endpoint H - P Information Registers (U2DENH - P) (PXA31x Processor Only)	210	Endpoint J
0x5410_052C	Reserved		PXA32x and PXA30x Only
0x5410_0530	U2DC Endpoint H - P Information Registers (U2DENH - P) (PXA31x Processor Only)	210	Endpoint K
0x5410_0530	Reserved		PXA32x and PXA30x Only
0x5410_0534	U2DC Endpoint H - P Information Registers (U2DENH - P) (PXA31x Processor Only)	210	Endpoint L
0x5410_0534	Reserved		PXA32x and PXA30x Only
0x5410_0538	U2DC Endpoint H - P Information Registers (U2DENH - P) (PXA31x Processor Only)	210	Endpoint M
0x5410_0538	Reserved		PXA32x and PXA30x Only
0x5410_053C	U2DC Endpoint H - P Information Registers (U2DENH - P) (PXA31x Processor Only)	210	Endpoint N

Table 50: U2DC Register Addresses (Continued)

Physical Address	Description	Page	Notes
0x5410_053C	Reserved		PXA32x and PXA30x Only
0x5410_0540	U2DC Endpoint H - P Information Registers (U2DENH - P) (PXA31x Processor Only)	210	Endpoint P
0x5410_0540	Reserved		PXA32x and PXA30x Only
0x5410_0544 – 0x5410_0FFF	Reserved		

Table 51 summarizes the U2DMA Controller registers.

Table 51: U2DMA Controller Registers

Physical Address	Description	Page	Notes
0x5410_1000	U2DMA Channel 0 - 7 Control/Status Registers (U2DMACSR0-7)	217	Channel 0
0x5410_1004	U2DMA Channel 0 - 7 Control/Status Registers (U2DMACSR0-7)	217	Channel 1
0x5410_1008	U2DMA Channel 0 - 7 Control/Status Registers (U2DMACSR0-7)	217	Channel 2
0x5410_100C	U2DMA Channel 0 - 7 Control/Status Registers (U2DMACSR0-7)	217	Channel 3
0x5410_1010	U2DMA Channel 0 - 7 Control/Status Registers (U2DMACSR0-7)	217	Channel 4
0x5410_1014	U2DMA Channel 0 - 7 Control/Status Registers (U2DMACSR0-7)	217	Channel 5
0x5410_1018	U2DMA Channel 0 - 7 Control/Status Registers (U2DMACSR0-7)	217	Channel 6
0x5410_101C	U2DMA Channel 0 - 7 Control/Status Registers (U2DMACSR0-7)	217	Channel 7
0x5410_1020	U2DMA Channel 8 - 15 Control/Status Registers (U2DMACSR8-15) (PXA31x Processor Only)	217	Channel 8
0x5410_1020	Reserved		PXA32x and PXA30x Only
0x5410_1024	U2DMA Channel 8 - 15 Control/Status Registers (U2DMACSR8-15) (PXA31x Processor Only)	217	Channel 9
0x5410_1024	Reserved		PXA32x and PXA30x Only
0x5410_1028	U2DMA Channel 8 - 15 Control/Status Registers (U2DMACSR8-15) (PXA31x Processor Only)	217	Channel 10

Table 51: U2DMA Controller Registers (Continued)

Physical Address	Description	Page	Notes
0x5410_1028	Reserved		PXA32x and PXA30x Only
0x5410_102C	U2DMA Channel 8 - 15 Control/Status Registers (U2DMACSR8-15) (PXA31x Processor Only)	217	Channel 11
0x5410_102C	Reserved		PXA32x and PXA30x Only
0x5410_1030	U2DMA Channel 8 - 15 Control/Status Registers (U2DMACSR8-15) (PXA31x Processor Only)	217	Channel 12
0x5410_1030	Reserved		PXA32x and PXA30x Only
0x5410_1034	U2DMA Channel 8 - 15 Control/Status Registers (U2DMACSR8-15) (PXA31x Processor Only)	217	Channel 13
0x5410_1034	Reserved		PXA32x and PXA30x Only
0x5410_1038	U2DMA Channel 8 - 15 Control/Status Registers (U2DMACSR8-15) (PXA31x Processor Only)	217	Channel 14
0x5410_1038	Reserved		PXA32x and PXA30x Only
0x5410_103C	U2DMA Channel 8 - 15 Control/Status Registers (U2DMACSR8-15) (PXA31x Processor Only)	217	Channel 15
0x5410_103C	Reserved		PXA32x and PXA30x Only
0x5410_1040 – 0x5410_107F	Reserved		
0x5410_1080	U2DMA Control Register (U2DMACR)	212	
0x5410_1084 – 0x5410_10DF	Reserved		
0x5410_10F0	U2DMA Interrupt Register (U2DMAINT)	223	
0x5410_10F4 – 0x5410_11FF	Reserved		
0x5410_1200	U2DMA Channel 0 - 7 Descriptor Address Registers (U2DMADADR0-7)	213	Channel 0
0x5410_1204	U2DMA Channel 0 - 7 Source Address Registers (U2DMASADR0 - 7)	214	Channel 0
0x5410_1208	U2DMA Channel 0 - 7 Target Address Registers (U2DMATADR0-7)	215	Channel 0
0x5410_120C	U2DMA Channel 0 - 7 Command Registers (U2DMACMD0-7)	216	Channel 0
0x5410_1210	U2DMA Channel 0 - 7 Descriptor Address Registers (U2DMADADR0-7)	213	Channel 1

Table 51: U2DMA Controller Registers (Continued)

Physical Address	Description	Page	Notes
0x5410_1214	U2DMA Channel 0 - 7 Source Address Registers (U2DMASADR0 - 7)	214	Channel 1
0x5410_1218	U2DMA Channel 0 - 7 Target Address Registers (U2DMATADR0-7)	215	Channel 1
0x5410_121C	U2DMA Channel 0 - 7 Command Registers (U2DMACMD0-7)	216	Channel 1
0x5410_1220	U2DMA Channel 0 - 7 Descriptor Address Registers (U2DMADADR0-7)	213	Channel 2
0x5410_1224	U2DMA Channel 0 - 7 Source Address Registers (U2DMASADR0 - 7)	214	Channel 2
0x5410_1228	U2DMA Channel 0 - 7 Target Address Registers (U2DMATADR0-7)	215	Channel 2
0x5410_122C	U2DMA Channel 0 - 7 Command Registers (U2DMACMD0-7)	216	Channel 2
0x5410_1230	U2DMA Channel 0 - 7 Descriptor Address Registers (U2DMADADR0-7)	213	Channel 3
0x5410_1234	U2DMA Channel 0 - 7 Source Address Registers (U2DMASADR0 - 7)	214	Channel 3
0x5410_1238	U2DMA Channel 0 - 7 Target Address Registers (U2DMATADR0-7)	215	Channel 3
0x5410_123C	U2DMA Channel 0 - 7 Command Registers (U2DMACMD0-7)	216	Channel 3
0x5410_1240	U2DMA Channel 0 - 7 Descriptor Address Registers (U2DMADADR0-7)	213	Channel 4
0x5410_1244	U2DMA Channel 0 - 7 Source Address Registers (U2DMASADR0 - 7)	214	Channel 4
0x5410_1248	U2DMA Channel 0 - 7 Target Address Registers (U2DMATADR0-7)	215	Channel 4
0x5410_124C	U2DMA Channel 0 - 7 Command Registers (U2DMACMD0-7)	216	Channel 4
0x5410_1250	U2DMA Channel 0 - 7 Descriptor Address Registers (U2DMADADR0-7)	213	Channel 5
0x5410_1254	U2DMA Channel 0 - 7 Source Address Registers (U2DMASADR0 - 7)	214	Channel 5
0x5410_1258	U2DMA Channel 0 - 7 Target Address Registers (U2DMATADR0-7)	215	Channel 5
0x5410_125C	U2DMA Channel 0 - 7 Command Registers (U2DMACMD0-7)	216	Channel 5
0x5410_1260	U2DMA Channel 0 - 7 Descriptor Address Registers (U2DMADADR0-7)	213	Channel 6
0x5410_1264	U2DMA Channel 0 - 7 Source Address Registers (U2DMASADR0 - 7)	214	Channel 6

Table 51: U2DMA Controller Registers (Continued)

Physical Address	Description	Page	Notes
0x5410_1268	U2DMA Channel 0 - 7 Target Address Registers (U2DMATADR0-7)	215	Channel 6
0x5410_126C	U2DMA Channel 0 - 7 Command Registers (U2DMACMD0-7)	216	Channel 6
0x5410_1270	U2DMA Channel 0 - 7 Descriptor Address Registers (U2DMADADR0-7)	213	Channel 7
0x5410_1274	U2DMA Channel 0 - 7 Source Address Registers (U2DMASADR0 - 7)	214	Channel 7
0x5410_1278	U2DMA Channel 0 - 7 Target Address Registers (U2DMATADR0-7)	215	Channel 7
0x5410_127C	U2DMA Channel 0 - 7 Command Registers (U2DMACMD0-7)	216	Channel 7
0x5410_1280	U2DMA Channel 8 - 15 Descriptor Address Registers U2DMADADR8-15) (PXA31x Processor Only)	213	Channel 8
0x5410_1280	Reserved		PXA32x and PXA30x Only
0x5410_1284	U2DMA Channel 8 - 15 Source Address Registers (U2DMASADR8 - 15) (PXA31x Processor Only)	214	Channel 8
0x5410_1284	Reserved		PXA32x and PXA30x Only
0x5410_1288	U2DMA Channel 8 - 15 Target Address Registers (U2DMATADR8-15) (PXA31x Processor Only)	215	Channel 8
0x5410_1288	Reserved		PXA32x and PXA30x Only
0x5410_128C	U2DMA Channel 8 - 15 Command Registers (U2DMACMD8-15) (PXA31x Processor Only)	216	Channel 8
0x5410_128C	Reserved		PXA32x and PXA30x Only
0x5410_1290	U2DMA Channel 8 - 15 Descriptor Address Registers U2DMADADR8-15) (PXA31x Processor Only)	213	Channel 9
0x5410_1290	Reserved		PXA32x and PXA30x Only
0x5410_1294	U2DMA Channel 8 - 15 Source Address Registers (U2DMASADR8 - 15) (PXA31x Processor Only)	214	Channel 9
0x5410_1294	Reserved		PXA32x and PXA30x Only
0x5410_1298	U2DMA Channel 8 - 15 Target Address Registers (U2DMATADR8-15) (PXA31x Processor Only)	215	Channel 9
0x5410_1298	Reserved		PXA32x and PXA30x Only

Table 51: U2DMA Controller Registers (Continued)

Physical Address	Description	Page	Notes
0x5410_129C	U2DMA Channel 8 - 15 Command Registers (U2DMACMD8-15) (PXA31x Processor Only)	216	Channel 9
0x5410_129C	Reserved		PXA32x and PXA30x Only
0x5410_12A0	U2DMA Channel 8 - 15 Descriptor Address Registers U2DMADADR8-15) (PXA31x Processor Only)	213	Channel 10
0x5410_12A0	Reserved		PXA32x and PXA30x Only
0x5410_12A4	U2DMA Channel 8 - 15 Source Address Registers (U2DMASADR8 - 15) (PXA31x Processor Only)	214	Channel 10
0x5410_12A4	Reserved		PXA32x and PXA30x Only
0x5410_12A8	U2DMA Channel 8 - 15 Target Address Registers (U2DMATADR8-15) (PXA31x Processor Only)	215	Channel 10
0x5410_12A8	Reserved		PXA32x and PXA30x Only
0x5410_12AC	U2DMA Channel 8 - 15 Command Registers (U2DMACMD8-15) (PXA31x Processor Only)	216	Channel 10
0x5410_12AC	Reserved		PXA32x and PXA30x Only
0x5410_12B0	U2DMA Channel 8 - 15 Descriptor Address Registers U2DMADADR8-15) (PXA31x Processor Only)	213	Channel 11
0x5410_12B0	Reserved		PXA32x and PXA30x Only
0x5410_12B4	U2DMA Channel 8 - 15 Source Address Registers (U2DMASADR8 - 15) (PXA31x Processor Only)	214	Channel 11
0x5410_12B4	Reserved		PXA32x and PXA30x Only
0x5410_12B8	U2DMA Channel 8 - 15 Target Address Registers (U2DMATADR8-15) (PXA31x Processor Only)	215	Channel 11
0x5410_12B8	Reserved		PXA32x and PXA30x Only
0x5410_12BC	U2DMA Channel 8 - 15 Command Registers (U2DMACMD8-15) (PXA31x Processor Only)	216	Channel 11
0x5410_12BC	Reserved		PXA32x and PXA30x Only
0x5410_12C0	U2DMA Channel 8 - 15 Descriptor Address Registers U2DMADADR8-15) (PXA31x Processor Only)	213	Channel 12
0x5410_12C0	Reserved		PXA32x and PXA30x Only

Table 51: U2DMA Controller Registers (Continued)

Physical Address	Description	Page	Notes
0x5410_12C4	U2DMA Channel 8 - 15 Source Address Registers (U2DMASADR8 - 15) (PXA31x Processor Only)	214	Channel 12
0x5410_12C4	Reserved		PXA32x and PXA30x Only
0x5410_12C8	U2DMA Channel 8 - 15 Target Address Registers (U2DMATADR8-15) (PXA31x Processor Only)	215	Channel 12
0x5410_12C8	Reserved		PXA32x and PXA30x Only
0x5410_12CC	U2DMA Channel 8 - 15 Command Registers (U2DMACMD8-15) (PXA31x Processor Only)	216	Channel 12
0x5410_12CC	Reserved		PXA32x and PXA30x Only
0x5410_12D0	U2DMA Channel 8 - 15 Descriptor Address Registers U2DMADADR8-15) (PXA31x Processor Only)	213	Channel 13
0x5410_12D0	Reserved		PXA32x and PXA30x Only
0x5410_12D4	U2DMA Channel 8 - 15 Source Address Registers (U2DMASADR8 - 15) (PXA31x Processor Only)	214	Channel 13
0x5410_12D4	Reserved		PXA32x and PXA30x Only
0x5410_12D8	U2DMA Channel 8 - 15 Target Address Registers (U2DMATADR8-15) (PXA31x Processor Only)	215	Channel 13
0x5410_12D8	Reserved		PXA32x and PXA30x Only
0x5410_12DC	U2DMA Channel 8 - 15 Command Registers (U2DMACMD8-15) (PXA31x Processor Only)	216	Channel 13
0x5410_12DC	Reserved		PXA32x and PXA30x Only
0x5410_12E0	U2DMA Channel 8 - 15 Descriptor Address Registers U2DMADADR8-15) (PXA31x Processor Only)	213	Channel 14
0x5410_12E0	Reserved		PXA32x and PXA30x Only
0x5410_12E4	U2DMA Channel 8 - 15 Source Address Registers (U2DMASADR8 - 15) (PXA31x Processor Only)	214	Channel 14
0x5410_12E4	Reserved		PXA32x and PXA30x Only
0x5410_12E8	U2DMA Channel 8 - 15 Target Address Registers (U2DMATADR8-15) (PXA31x Processor Only)	215	Channel 14
0x5410_12E8	Reserved		PXA32x and PXA30x Only

Table 51: U2DMA Controller Registers (Continued)

Physical Address	Description	Page	Notes
0x5410_12EC	U2DMA Channel 8 - 15 Command Registers (U2DMACMD8-15) (PXA31x Processor Only)	216	Channel 14
0x5410_12EC	Reserved		PXA32x and PXA30x Only
0x5410_12F0	U2DMA Channel 8 - 15 Descriptor Address Registers U2DMADADR8-15) (PXA31x Processor Only)	213	Channel 15
0x5410_12F0	Reserved		PXA32x and PXA30x Only
0x5410_12F4	U2DMA Channel 8 - 15 Source Address Registers (U2DMASADR8 - 15) (PXA31x Processor Only)	214	Channel 15
0x5410_12F4	Reserved		PXA32x and PXA30x Only
0x5410_12F8	U2DMA Channel 8 - 15 Target Address Registers (U2DMATADR8-15) (PXA31x Processor Only)	215	Channel 15
0x5410_12F8	Reserved		PXA32x and PXA30x Only
0x5410_12FC	U2DMA Channel 8 - 15 Command Registers (U2DMACMD8-15) (PXA31x Processor Only)	216	Channel 15
0x5410_12FC	Reserved		PXA32x and PXA30x Only
0x5410_1300–0 x5410_FFFF	Reserved		

2.11.1 U2DC Control Register (U2DCR)

The register organization and individual bit definitions are shown in Table 52.

Table 52: U2DCR Bit Definitions

Physical Address 0x5410_0000 **U2DCR** **U2DC**

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	NDC	HSTC			SPEOREN	FSTC			Reserved	UCLKOVR	ABP	ADD	CC	HS	SMAC	DWRE	ACN				AIN				AAISN				EMCE	UDR	UDA	UDE
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	Read/Write	NDC	**NAK during Config** If an endpoint is access by the Host that is currently affected by an active set interface or set configuration, the U2DC will return a NAK response instead of STALL response. During a configuration change, the U2DC will respond to all non-control accesses from the host with Stall, unless the NDC bit is set and then the U2DC will respond with a Nak. During an interface change, the U2DC will respond to all non-control accesses from the host with Stall to all endpoints affected by the interface change, unless the U2DCR[NDC] bit is set and then the U2DC will respond with a Nak. 0 = U2DC returns a STALL response 1 = U2DC returns a NAK response
30:28	Read/Write	HSTC	**High Speed Timeout Calibration:** Software can vary the high-speed timeout value by adding 0 to 7 PHY clock periods, and can increase the High Speed Timeout value from 736 to 848 bit times
27	Read/Write	SPEOREN	**Short Packet EOR INTR generation Enable** Used to control the setting of EORINTR for short packets.. 0 = U2DMACSRx[EORINTR] is set upon End-of-Receive for all packets, short or otherwise 0 = U2DMACSRx[EORINTR] is only set upon End-of-Receive for short packets **NOTE:** SPEOREN has no effect on endpoint0. EoR for endpoint0 always results in EORINTR, regardless of packet size.
26:24	Read/Write	FSTC	**Full Speed Timeout Calibration:** Software can vary the high-speed timeout value by adding 0 to 7 PHY clock periods, and can increase the Full Speed Timeout value from 16 to 18 bit times
23	—	Reserved	**Reserved**

Table 52: U2DCR Bit Definitions (Continued)

Physical Address 0x5410_0000 **U2DCR** **U2DC**

	31	30	29	28	27	26	25	24	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	NDC	HSTC			SPEOREN	FSTC			Reserved	UCLKOVR	ABP	ADD	CC	HS	SMAC	DWRE	ACN				AIN				AAISN				EMCE	UDR	UDA	UDE
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	Read/Write	UCLKOVR	**UTM Clock Override** This bit is used to override clock muxing during Synchronous Test Mode (STM). To prevent clock glitches during STM, this bit should be set prior to the enabling of the U2DC (U2DCR[UDE] = 1). This bit should not be setduring normal operation. 0 = do not override clock muxing 1 = override clock muxing with external utm_clk
21	Read/Write	ABP	Application Bus Power: Determines if the device draws its power from the USB or if the device provides its own power. 0 = Device is Self-powered 1 = Device is Bus powered
20	Read/Write	ADD	**Application Device Disconnect:** Causes a soft disconnect of the device which is done by the PHY disabling its D+ pullup. 0 = Device is not soft disconnected 1 = Device is soft disconnected

Table 52: U2DCR Bit Definitions (Continued)

Physical Address **U2DCR** **U2DC**
0x5410_0000

	31	30	29	28	27	26	25	24	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	NDC	HSTC			SPEOREN	FSTC			Reserved	UCLKOVR	ABP	ADD	CC	HS	SMAC	DWRE	ACN				AIN				AAISN				EMCE	UDR	UDA	UDE
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	Read/Write 1 to Clear	CC	**Configuration Change** Indicates that a set configuration or set interface command has been received on Endpoint 0 by the U2DC. This gives the software the opportunity to modify the endpoint information registers to support the new configuration or alternate interface. While the software is potentially modifying the endpoint information registers, during a configuration change, the U2DC will respond to all non-control accesses from the host with Stall, unless the U2DCR[NDC] bit is set and then the U2DC will respond with a Nak. However, during an interface change, the U2DC will respond to all non-control accesses from the host with Stall to all endpoints affected by the interface change, unless the U2DCR[NDC] bit is set and then the U2DC will respond with a Nak. In order to proceed with subsequent transfers, either the configuration change interrupt (U2DISR[CC]) must be cleared from a set state, or this register must be cleared from a set state. If the U2DICR[CC] interrupt enable is not asserted, then the only way to handshake that the configuration change is completed and allow transfers to continue is to clear this bit. If the interrupt enable is asserted, and the interrupt is cleared to accomplish the handshake then this bit can be left asserted indefinitely without negatively affecting operation. Clearing this bit along with U2DISR[CC] is also acceptable. 1 = Indicates when Set_configuration or Set_interface device request has been received by U2DC.
18	Read	HS	**High Speed USB Detection:** Indicates whether the PHY's high-speed chirp was recognized by the host controller and the U2DC is operating in high-speed mode. 0 = U2DC Operating at Full Speed 1 = U2DC Operating at High Speed

Table 52: U2DCR Bit Definitions (Continued)

Physical Address 0x5410_0000 **U2DCR** **U2DC**

	31	30	29	28	27	26	25	24	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	NDC	HSTC			SPEOREN	FSTC			Reserved	UCLKOVR	ABP	ADD	CC	HS	SMAC	DWRE	ACN				AIN				AAISN				EMCE	UDR	UDA	UDE
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
17	Read/Write 1 to Set	SMAC	**Switch Endpoint Memory to Active Configuration**: Used to control the Endpoint Memory allocation. When the USB Host controller sends a Set_configuration or Set_interface command to the U2DC (if the Configuration Change interrupt is enabled), the U2DC sends an interrupt to the processor. After any data remaining in the Endpoint memory is read, setting SMAC causes the U2DC to flush the Endpoint memory for Endpoints and changes the Endpoint memory allocation to reflect the active configuration and alternate interfaces set by the USB Host controller. Endpoint memory for Endpoint 0 is unaffected by SMAC. The U2DC clears SMACwhen the memory allocation has completed. See Section 2.7 for more information on the execution of USB device requests. The U2DC will indicate an overflow condition if an isochronous Out data is received between the execution of a Set_configuration or Set_interface command. Changing of the Endpoint memory allocation and will send zero-size packets if an isochronous In token is received in the same circumstance. 0 = No change to active configuration 1 = Change the Endpoint memory allocation to the Active Configuration, and active alternate interface setting
16	Read	DWRE	**Device Remote Wake-up Feature:** When the USB Host controller sends a Set_feature command to enable the Device_remote_wakeup feature, DRWE Status bit is set to 1 to indicate the feature has been enabled. When the USB Host controller sends a Clear_feature command to disable the Device_remote_wakeup feature, DRWE bit is set to 0 to indicate the feature has been disabled by the host. 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

Table 52: U2DCR Bit Definitions (Continued)

Physical Address 0x5410_0000 — **U2DCR** — **U2DC**

	31	30	29	28	27	26	25	24	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	NDC	HSTC			SPEOREN	FSTC			Reserved	UCLKOVR	ABP	ADD	CC	HS	SMAC	DWRE	ACN				AIN				AAISN				EMCE	UDR	UDA	UDE
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
15:12	Read	ACN	Active Configuration Number Indicates the current configuration with which the last selected alternate interface was associated. This is selected by the USB Host controller to be used by the U2DC. The active configuration number is set when the USB Host controller issues a Set_configuration command to the U2DC. If the U2DICR[IECC] is set, the U2DC will generate an interrupt to the processor to indicate the Set_configuration or Set_interface command has been completed. Software must set U2DCR[SMAC] to set the Endpoint memory allocation to the new configuration or alternate interface setting. 1 - 15 – Active U2DC Configuration Number selected by the USB Host controller to be used by the U2DC. The active configuration number is set when the USB Host controller issues a SET_CONFIGURATION command to the U2DC.
11:8	Read	AIN	**Associated interface number** Indicates (respectively) the last selected alternate interface with which the last selected alternate interface was associated. This is selected by the USB Host controller to be used by the U2DC. 0 - 15 – Associated U2DC Interface Number selected by the USB Host controller to be used by the U2DC. The associated interface setting number is set when the USB Host controller issues a SET_INTERFACE command to the U2DC, they show to which interface the new alternate interface setting is associated.
7:4	Read	AAISN	**Active alternate interface setting number** Indicates the interface number with which the last selected alternate interface was associated. This is selected by the USB Host controller to be used by the U2DC. Only the last selected alternate interface is stored—not all active alternate interfaces. Software must maintain a map of active alternate interfaces selected by the USB host if a configuration with more than one interface is selected. 0 - 15 – Active U2DC Alternate Interface Setting Number selected by the USB Host controller to be used by the U2DC. The active alternate interface setting number is set when the USB Host controller issues a SET_INTERFACE command to the U2DC.

Table 52: U2DCR Bit Definitions (Continued)

Physical Address 0x5410_0000 **U2DCR** **U2DC**

	31	30	29	28	27	26	25	24	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	NDC	HSTC			SPEOREN	FSTC			Reserved	UCLKOVR	ABP	ADD	CC	HS	SMAC	DWRE	ACN				AIN				AAISN				EMCE	UDR	UDA	UDE
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	Read/Write 1 to Clear	EMCE	**Endpoint Memory Configuration Error** Indicates the endpoint configuration could not be loaded and the FIFO depth sizes indicated for the configuration are in error. When a Set_configuration or Set_interface command is executed, the U2DC checks the endpoint configurations to verify the Endpoint FIFO depth sizes are valid and does not allocate more memory than the available. If the FIFO memory is allocated incorrectly, the U2DC will set EMCE and clear U2DCR[UDE]. EMCE must be cleared before attempting to re-enable the U2DC. **NOTE:** Software must always clear all Endpoint Enable (EE) bits in the U2DC Endpoint configuration registers (U2DCRx) whenever UDE is cleared to 0. The Endpoint Enable bit for each endpoint should only be set to 1 again after the software has set UDE to 1 and re-programmed that endpoint's U2DC Endpoint Information Registers (U2DENx). 0 = No Endpoint memory configuration error 1 = The Endpoint memory configuration has an error and cannot be loaded
2	Read/Write 1 to Set	UDR	**U2DC Resume:** If the USB Host controller has executed the Set_feature command and enabled the device remote wake-up feature of the U2DC, software can set UDR when the U2DC is in a Suspend state to force the U2DC to perform the remote wake-up operation. When UDR is set, the U2DC will cause the PHY to drive Resume signaling on the USBC_P and USBC_N signals for 3 ms, and will then float the bus. If the USB Host controller does not drive Resume signaling on the USB within 3 ms, the U2DC goes back into Suspend. The UDR bit is cleared by the U2DC upon entering the K-state. If the USB Host controller has not enabled the Device Remote Wake-up feature of the U2DC, UDR is ignored. 0 = Maintain U2DC suspend state 1 = Force U2DC out of suspend
1	Read	UDA	**U2DC Active:** Indicates the U2DC is ready for active communication on the USB 2.0 bus. When 0, this bit signifies that the configuration still needs to be loaded before the U2DC can be ready for communications on the USB bus. 0 = U2DC currently inactive on the USB 1 = U2DC currently active on the USB

Table 52: U2DCR Bit Definitions (Continued)

Physical Address 0x5410_0000 **U2DCR** **U2DC**

	31	30	29	28	27	26	25	24	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	NDC	HSTC			SPEOREN	FSTC			Reserved	UCLKOVR	ABP	ADD	CC	HS	SMAC	DWRE	ACN				AIN				AAISN				EMCE	UDR	UDA	UDE
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	Read/Write	UDE	**U2DC Enable:** Enables and disables the U2DC. UDE is cleared following a reset of the processor, which disables the U2DC, and disables the monitoring of the USB. Any USB Host controller commands or USB reset issued while UDE is clear are ignored. The Endpoint Configuration registers may be programmed prior to setting UDE to minimize the amount of time the U2DC will NAK accesses after the first Set_configuration command is issued by the USB host. Alternatively the registers may be set up after the first Set_configuration command just as it is done for later Set_configuration or SET_INTERFACE commands as shown in the flowchart in Figure 32. When UDE is set, the U2DC is then enabled for USB transmission and reception. Software must always clear all Endpoint Enable (EE) bits in the U2DC Endpoint Configuration Registers (U2DCRx) whenever UDE is cleared to 0. The Endpoint Enable bit for each endpoint should only be set to 1 again after the software has set UDE to 1 and re-programmed that endpoint's U2DC Endpoint Information Registers (U2DENx). When UDE is cleared, the entire U2DC is disabled and reset. If this bit is cleared while the U2DC is actively transmitting or receiving data, the U2DC 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 flushed. While UDE is clear, the U2DC Endpoint Configuration registers can be programmed for another configuration, however the U2DC Endpoint Information Registers (U2DENx) are held in reset and cannot be successfully written. If the UDE bit is 0 and the EMCE bit is 1, there was an error while re-allocating the Endpoint memory. When this condition is detected, the hardware automatically clears the UDE bit. This means that the U2DC is in reset and all previous U2DC Endpoint Information Registers (U2DENx) programming including states and enumeration values of the USB Interface are set back to their default values. The USB Host will see a Device Disconnect event and the only way to recover from this is for the software to clear the U2DC Endpoints Control Status Registers (U2DCSRx), and DMA register for each endpoint and then set the UDE bit back to 1. The USB Host will have to detect the device connect and re-enumerate the U2DC all over again. 0 = U2DC disabled 1 = U2DC enabled **PXA31x Processor Only:** The UDE bit also enables the ULPI interface since all U2DC traffic passes through the ULPI interface. The ULPI interface can be enabled separately without enabling the U2DC by setting the U2DOTGCR[ULE] bit. To disable the ULPI interface, both the U2DCR[UDE] and U2DOTGCR[ULE] bits must not be set. When the ULPI interface is disabled, it is held in reset

2.11.2 U2DC Interrupt Control Register (U2DICR)

The U2DC Interrupt Control register (U2DICR) contain read/write Control bits to enable interrupt service requests from data Endpoint 0 - Endpoint G and specific USB events. The U2DICR register contains the interrupt control bits for Endpoints 0, A-G. Setting any Interrupt Enable bit enables the interrupt, while clearing any Interrupt Enable bit disables that interrupt. All of the U2DICR enable bits are cleared on system reset.

Each endpoint has the potential of three interrupt sources, FIFO Error, Packet Complete and Short Packet Complete.

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. However; programming any interrupt enable bit to 0 clears an already set U2DC interrupt signal to the core. See Figure 34 for more detail. This behavior is different from UDC (USB 1.1) interrupt functionality. In the UDC, clearing of interrupt enable does not affect the state of its single Interrupt signal, but in U2DC the interrupt signal is cleared.

When the U2DC is operating in High-speed mode, interrupts for both the SOF and µSOF are generated. The SOF generation occurs concurrently with every eighth µSOF when the frame number is updated.

Note

To receive notification of the Endpoint 0 Receive FIFO overflow, software must enable IE0 bit in the U2DICR as the Endpoint 0 control/status register does not contain an overflow indication bit

The register organization and individual bit definitions for U2DICR are shown in Table 53.

Table 53: U2DICR Bit Definitions

Physical Address 0x5410_0004	U2DICR	U2DC

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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**	**IEUSOF**	**IERU**	**IESU**	**IERS**	**IEDPE**	**Reserved**	**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**

Bits	Access	Name	Description
31	Read/Write	IECC	Configuration Change 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
30	Read/Write	IESOF	Start of Frame 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
29	Read/Write	IEUSOF	Micro Start of Frame (HS only) 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
28	Read/Write	IERU	Resume 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
27	Read/Write	IESU	Suspend 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
26	Read/Write	IERS	Reset 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
25	Read/Write	IEDPE	Data Packet Error 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
24	—	Reserved	**Reserved**

Table 53: U2DICR Bit Definitions (Continued)

Physical Address 0x5410_0004 **U2DICR** **U2DC**

User Settings																														
Bit	31	30	29	28	27	26	25	24	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	IEUSOF	IERU	IESU	IERS	IEDPE	Reserved	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

Bits	Access	Name	Description
23:21	Read/Write	IEG	Endpoint G 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled.
20:18	Read/Write	IEF	Endpoint F 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
17:15	Read/Write	IEE	Endpoint E 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
14:12	Read/Write	IED	Endpoint D 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
11:9	Read/Write	IEC	Endpoint C 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
8:6	Read/Write	IEB	Endpoint B 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
5:3	Read/Write	IEA	Endpoint A 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
2:0	Read/Write	IE0	Endpoint 0 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled

2.11.3 U2DC Interrupt Control Register 2 (U2DICR2) (PXA31x Processor Only)

The U2DC Interrupt Control register (U2DICR2) contain read/write Control bits to enable interrupt service requests from data Endpoint H - Endpoint P. Setting any Interrupt Enable bit enables the interrupt, while clearing any Interrupt Enable bit disables that interrupt. On system reset all of the U2DICR2 enable bits are cleared.

Each Endpoint has the potential of three interrupt sources, FIFO Error, Packet Complete and Short Packet Complete.

Table 54: U2DICR2 Bit Definitions

Physical Address 0x5410_0008 **U2DICR2** **U2DC**

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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								**IEP**			**IEN**			**IEM**			**IEL**			**IEK**			**IEJ**			**IEI**			**IEH**		
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	**Reserved**
23:21	Read/Write	IEP	Endpoint P 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
20:18	Read/Write	IEN	Endpoint N 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
17:15	Read/Write	IEM	Endpoint M 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
14:12	Read/Write	IEL	Endpoint L 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
11:9	Read/Write	IEK	Endpoint K 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
8:6	Read/Write	IEJ	Endpoint J 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled

Table 54: U2DICR2 Bit Definitions (Continued)

Physical Address 0x5410_0008	U2DICR2	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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								IEP			IEN			IEM			IEL			IEK			IEJ			IEI			IEH		
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
5:3	Read/Write	IEI	Endpoint I 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled
2:0	Read/Write	IEH	Endpoint H 0b000 = No interrupts enabled 0bxx1 = Short Packet Complete Interrupt enabled 0bx1x = Packet Complete Interrupt enabled 0b1xx = FIFO Error Interrupt enabled

2.11.4 U2DC Interrupt Status Register (U2DISR)

The U2DC Interrupt Status register U2DISR contains bits that are used to generate the U2DC interrupt request. The U2DISR register contains interrupt status for Endpoints 0, A-G.Each bit in the U2DC Interrupt Status registers is logically ANDed with its Interrupt Enable and then logically ORed together to produce one signal, which is logically ORed with the U2DMA interrupt to then generate a single interrupt request. Figure 34 shows the U2DC interrupt generation. When the Interrupt Service Routine (ISR) for the U2DC is executed, it must read the U2DC Interrupt Status register and U2DMA Interrupt register to determine why the U2DC interrupt occurred.

- **FIFO Error Interrupt** — The FIFO Error 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 U2DC tries to transmit more bytes of data from the Transmit buffer than what was loaded into the buffer, including when the U2DC tries to transmit data when the Transmit buffer is empty and no zero packet was identified. Properly identified zero packets do not cause a FIFO Error Interrupt. Properly identifying zero packets to the U2DC involve using the DMA Descriptor command register with U2DMACMDx[LEN]=0, and U2DMACMDx[PACK_COMP] set if DMA is enabled, or using the Control register bits U2DCSRx[SP] or U2DCSR0[IPR] for processor accesses. 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 U2DC tries to load more bytes of data into the Receive buffer than the maximum packet size of the buffer, or if the Receive FIFO does contain enough space for the packet based on the size specified in U2DCRx[MPS]. Since Endpoint 0 is a bidirectional endpoint, a FIFO error can occur on either a FIFO overrun or FIFO underrun condition. Refer to Section 2.7.3 for more information on FIFO error conditions.
- **Packet Complete Interrupt** — The second interrupt for each endpoint is Packet Complete. 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 successfully transmitted. Since Endpoint 0 is neither an In endpoint nor an Out endpoint, but instead is bi-directional and can move data in either direction, a Packet Complete Interrupt is set for In transfers and Out transfers to Endpoint 0. The Packet Complete Interrupt is generated whether a full packet or a

short packet is encountered. The Packet Complete Interrupt is masked by the DME bit in the corresponding Control/Status register such that the interrupt may only occur if DME is enabled.

- **Short Packet Complete Interrupt** — The third interrupt for each endpoint is Short Packet Complete. This interrupt *only* applies to Out endpoints. If the endpoint is configured as an Out endpoint, the Short Packet Complete Interrupt indicates that an Out packet has been received for the packet currently being accessed by the system, but the amount of data in the transfer did not match the endpoint's max packet size. It is provided as a separate interrupt so that software may mask the normal packet completion, which can be handled directly by the U2DMA, but still be notified of the existence of a short packet, so that the software can take appropriate steps, depending on the endpoint usage. The U2DMA handles short packets automatically by using the EOR methodology (See Figure 23); however, the existence of a short packet may have special meaning to software and special actions maybe required apart from the U2DMA control. When a short packet is encountered for an Out packet, both the Packet Complete Interrupt and the Short Packet Complete Interrupt bits are set. No interrupt is generated if the endpoint is configured as an In endpoint.

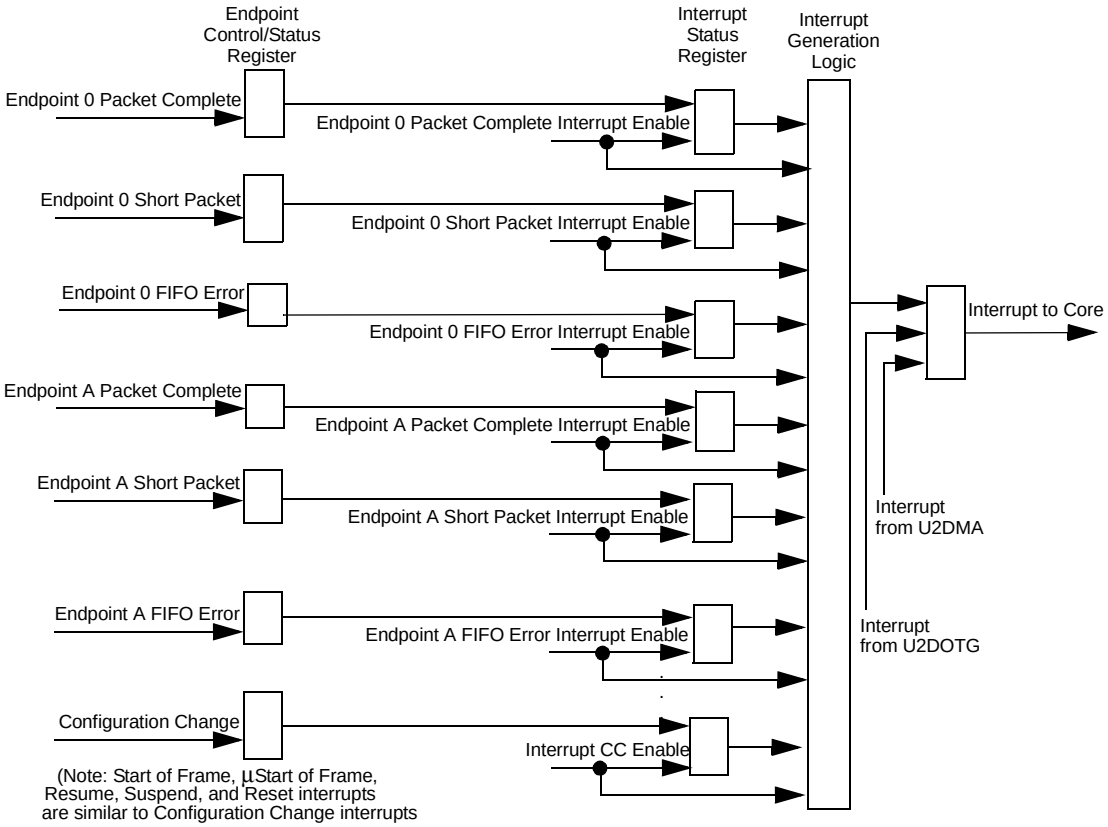
Table 55 shows the USB events interrupts, the Enable and Status bits, and the USB event that causes each interrupt to be generated by the U2DC. For the interrupt enables for Reset (IERS), Suspend (IESU), Resume (IERU), SOF (IESOF), µSOF (IEUSOF - HS) and Configuration Change (IECC) conditions, each bit in the U2DICR is used to enable their 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 is clear, the interrupt is disabled and is not generated. The setting of the Enable bit does not affect the setting of the U2DC 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 55: USB Events Interrupts

Interrupt Enable Bit (U2DICR)	Interrupt Status Bit (U2DISR)	USB Condition That Generates Interrupt
IECC	IRCC	Set_configuration or Set_interface command received
IESOF	IRSOF	Start of frame received (FS *and* HS)
IEUSOF	IRUSOF	Micro Start of Frame received (HS only)
IERU	IRRU	Resume detected
IESU	IRSU	Suspend detected
IERS	IRRS	USB Reset detected

In addition to the endpoint interrupts and the USB event interrupts, a U2DISR[IRDPE] interrupt can also be generated when a data packet error occurs on an ISO endpoint and the U2DICR[IEDPE]enable is asserted. This interrupt is used to identify that one of the endpoints had its control/status register (U2DCSRx) DPE bit asserted. It is provided as an interrupt so that software does not have to poll each endpoint Control Status register to determine if there is an error in the packet before each packet transfer. Instead when the interrupt is triggered, software can poll the U2DCSRx registers. This permits software to only poll these registers on the rare times there is an actual error.

Figure 34: U2DC Interrupt Generation



Every bit in U2DISR is controlled by an enable bit in the U2DC Interrupt Control registers (U2DICR). When the enable bits are clear, they prevent a status bit in the corresponding U2DC 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 U2DC interrupt is generated. Software must write a 1 to the bit position to be cleared to clear interrupt status bits. The interrupt request for the U2DC remains active as long as the value of the U2DC Interrupt Status register bits ANDed with the enable bit is non-zero.

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 56.

Table 56: U2DISR Bit Definitions

Physical Address 0x5410_000C **U2DISR** **U2DC**

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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**	**IRUSOF**	**IRRU**	**IRSU**	**IRRS**	**IRDPE**	**reserved**	**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**

Bits	Access	Name	Description
31	Read/Write 1 to clear	IRCC	Configuration Change
30	Read/Write 1 to clear	IRSOF	Start of Frame
29	Read/Write 1 to clear	IRUSOF	Micro Start of Frame (HS only)
28	Read/Write 1 to clear	IRRU	Resume
27	Read/Write 1 to clear	IRSU	Suspend
26	Read/Write 1 to clear	IRRS	Reset
25:	Read/Write 1 to clear	IRDPE	Data Packet Error
24	—	Reserved	**Reserved**
23:21	Read/Write 1 to clear	IRG	Endpoint G
20:18	Read/Write 1 to clear	IRF	Endpoint F
17:15	Read/Write 1 to clear	IRE	Endpoint E
14:12	Read/Write 1 to clear	IRD	Endpoint D
11:9	Read/Write 1 to clear	IRC	Endpoint C
8:6	Read/Write 1 to clear	IRB	Endpoint B
5:3	Read/Write 1 to clear	IRA	Endpoint A
2:0	Read/Write 1 to clear	IR0	Endpoint 0

2.11.5 U2DC Interrupt Status Register 2 (U2DISR2) (PXA31x Processor)

The U2DC Interrupt Status register U2DISR contain bits that are used to generate the U2DC interrupt request. The U2DISR2 register contains interrupt status for Endpoints H - P. Each bit in the U2DC Interrupt Status registers is logically ANDed with its Interrupt Enable and then logically ORed together to produce one signal, which is logically ORed with the U2DMA interrupt and the U2DOTG interrupt to then generate a single interrupt request. Figure 34 shows the U2DC interrupt generation. When the Interrupt Service Routine (ISR) for the U2DC is executed, software must read the U2DC Interrupt Status registers and U2DMA Interrupt register, and U2DOTG Interrupt register to determine why the U2DC interrupt occurred.

Table 57: U2DISR2 Bit Definitions

Physical Address 0x5410_0010 **U2DISR2** **U2DC**

	31	30	29	28	27	26	25	24	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								IRP			IRN			IRM			IRL			IRK			IRJ			IRI			IRH		
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	**Reserved**
23:21	Read/Write 1 to clear	IRP	Endpoint P
20:18	Read/Write 1 to clear	IRN	Endpoint N
17:15	Read/Write 1 to clear	IRM	Endpoint M
14:12	Read/Write 1 to clear	IRL	Endpoint L
11:9	Read/Write 1 to clear	IRK	Endpoint K
8:6	Read/Write 1 to clear	IRJ	Endpoint J
5:3	Read/Write 1 to clear	IRI	Endpoint I
2:0	Read/Write 1 to clear	IRH	Endpoint H

2.11.6 U2DC Frame Number Register (U2DFNR)

The U2DC Frame Number register (U2DFNR), shown in Table 58, holds the 11-bit frame number contained in the last received SOF/µSOF (HS) packet, and can be monitored by software when performing isochronous transfers. If the SOF interrupt is enabled, a SOF interrupt is generated when the frame number bits are updated. For high-speed operation, this register holds the 11 bits sent during eight consecutive microforms by the host such that this field is updated at the same rate whether operating in full-speed or high-speed mode.

Table 58: U2DFNR Bit Definitions

Physical Address 0x5410_0014	U2DFNR	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	0

Bits	Access	Name	Description
31:11	—	Reserved	**Reserved**
10:0	Read	FN	Frame number associated with last received SOF/µSOF (HS).

2.11.7 U2DC OTG Control Register (U2DOTGCR) (PXA31x Processor Only)

This register contains the control bits for the OTG system. These bits interface to the ULPI interface and when a change is detected by the interface, it automatically generates a Write command to the ULPI registers in the PHY.

The register organization and individual bit definitions are shown in Table 59.

Table 59: U2DOTGCR Bit Definitions

Physical Address 0x5410_0020	U2DOTGCR	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	OTGEN	AALTHNP	AHNP	BHNP	Reserved																						CKAF	UTMID	ULAF	SMAF	RTSM	ULE
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
31	Read/Write	OTGEN	**On-the-Go Enable:** The OTG enable (OTGEN) bit enables software control of the decoding of OTG Set_feature commands. When OTGEN is set to 0, the U2DC 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. The AALTHNP, AHNP, and BHNP status bits are cleared when the OTGEN bit is set to 0. When OTGEN is set to 1, the U2DC 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

Table 59: U2DOTGCR Bit Definitions (Continued)

Physical Address 0x5410_0020 **U2DOTGCR** **U2DC**

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	OTGEN	AALTHNP	AHNP	BHNP	Reserved																						CKAF	UTMID	ULAF	SMAF	RTSM	ULE
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
30	Read	AALTHNP	**A-device Alternate Host Negotiation Protocol Port Support:** The A-device alternate host negotiation protocol port (AALTHNP) indicates if the A-device alternate HNP feature has been enabled by the USB Host controller. The AALTHNP bit is set to 1 when the USB Host controller executes a Set_feature command and indicates the a_alt_hnp_support feature is enabled and the OTGEN bit is set to 1. The AALTHNP bit is set to 0 when either a USB reset occurs or a session end is detected. It cannot be cleared using a Clear_feature command. It is also cleared when the OTGEN bit is set to 0. 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	Read	AHNP	**A-device Host Negotiation Protocol Support:** The A-device host negotiation protocol support (AHNP) indicates if the A-device HNP feature has been enabled by the USB Host controller. The AHNP bit is set to 1 when the USB Host controller executes a Set_feature command and indicates the a_hnp_support feature is enabled and the OTGEN bit is set to 1. The AHNP bit is set to 0 when either a USB reset occurs or a session end is detected. It cannot be cleared using a Clear_feature command. It is also cleared when the OTGEN bit is set to 0. 0 = B-device is connected to an A-device port that does not support HNP 1 = B-device is directly connected to an A-device port that supports HNP
28	R	BHNP	**B-device Host Negotiation Protocol Enable:** The B-device host negotiation protocol enable (BHNP) indicates if the B-device HNP feature has been enabled by the USB Host controller. The BHNP bit is set to 1 when the USB Host controller executes a Set_feature command and indicates the b_hnp_enable feature is enabled and the OTGEN bit is set to 1. The BHNP bit is set to 0 when either a USB reset occurs or a session end is detected. It cannot be cleared using a Clear_feature command. It is also cleared when the OTGEN bit is set to 0. 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
27:6	—	Reserved	**Reserved**

Table 59: U2DOTGCR Bit Definitions (Continued)

Physical Address 0x5410_0020 — **U2DOTGCR** — **U2DC**

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	OTGEN	**AALTHNP**	**AHNP**	**BHNP**	**Reserved**																						**CKAF**	**UTMID**	**ULAF**	**SMAF**	**RTSM**	**ULE**
Reset	**0**	**0**	**0**	**0**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**0**	**0**	**0**	**0**	**0**

Bits	Access	Name	Description
5	Read/Write	CKAF	**Carkit Mode Alternate Function Select** Selects the I/O Mux path for Carkit mode at the ULPI pad interface. This bit cannot be set at the same time as the ULAF and SMAF bits. It can only be written to a 1 if the SMAF and ULAF bits are also written to a 0 at the same time. 0 = Do not enable the Carkit mode path 1 = Enable the Carkit mode path
4	Read/Write	UTMID	**UTMI Interface Disable** Setting this bit prevents the UTMI interface of the U2DC from causing register writes to be generated to the “ULPI Function Control” register in the off-chip ULPI PHY. The U2DC UTMI interface should be disabled whenever the U2DC is disabled (U2DCR[UDE]=0) or the OTG system is configured as a Host (ULPI serial mode) or ULPI Carkit is active (ULPI Carkit mode). 0 = U2DC UTMI interface active 1 = U2DC UTMI interface disabled
3	Read/Write	ULAF	**ULPI Mode Alternate Function Select** Selects the I/O Mux path for Synchronous (ULPI) mode to the ULPI pad interface. This bit cannot be set at the same time as the SMAF and CKAF bits. It can only be written to a 1 if the SMAF and CKAF bits are also written to a 0 at the same time. 0 = Do not enable the ULPI mode path 1 = Enable the ULPI mode path
2	Read/Write	SMAF	**Serial Mode Alternate Function Select** Selects the I/O Mux path to serial mode for use with the USB Host -- This bit cannot be set at the same time as the ULAF and CKAF bits. It can only be written to a 1 if the ULAF and CKAF bits are also written to a 0 at the same time. 0 = Do not enable the serial mode path 1 = Enable the serial mode path
1	Read/Write 1 to Set	RTSM	**Return to Synchronous Mode (ULPI Mode)** This bit should only be written after the off-chip ULPI PHY is in serial mode, low-power mode, or Carkit mode. Writing this bit to a 1 causes the ULPI_STP output to the off-chip PHY to be asserted which signals it to return to Synchronous mode. This bit is self clearing and cannot be cleared by writing a 0 to it. 0 = Default 1 = Causes ULPI_STP to be asserted to the off-chip ULPI PHY

Table 59: U2DOTGCR Bit Definitions (Continued)

Physical Address 0x5410_0020	U2DOTGCR	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	OTGEN	AALTHNP	AHNP	BHNP	Reserved																						CKAF	UTMID	ULAF	SMAF	RTSM	ULE
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
0	Read/Write	ULE	**ULPI Interface Enable** This bit is used to enable the ULPI interface when the U2DC is not enabled (U2DCR[UDE]=0). When neither the UDE or ULE bits are set, the ULPI interface is held in reset. If the U2DC is disabled, the ULE bit must be set to 1 in order to communicate with the off-chip ULPI PHY. 0 = ULPI interface is disabled (only if U2DCR[UDE] is also set to 0) 1 = ULPI interface is enabled

2.11.8 U2DC OTG Interrupt Control/Enable Register (U2DOTGICR) (PXA31x Processor Only)

This register contains the interrupt enable bits for the OTG system. These bits determine whether the Interrupt Status bit is set in the U2DOTGISR for each interrupt from the PHY.

The register organization and individual bit definitions are shown in Table 60.

Table 60: U2DOTGICR Bit Definitions

Physical Address 0x5410_0024	U2DOTGICR	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	IESI	Reserved	IRELS1	IRELS0	IREID	IRESE	IRESV	IREVV	IRECK	Reserved	IFELS1	IFELS0	IFEID	IFESE	IFESV	IFEVV	IFECK
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:18	—	Reserved	**Reserved**
17	Read/Write	IESF	**OTG Set Feature Command Received Enable** 0 = No interrupt generated 1 = Generate an interrupt when the U2DC (device) receives a SetFeature command that enables it for HNP.
16	Read/Write	IESI	**OTG Interrupt Enable** 0 = No interrupt generated 1 = Generate an interrupt when the interrupt signal (ulpi_data[3]) is asserted during low-power, serial, or Carkit modes only.

Table 60: U2DOTGICR Bit Definitions (Continued)

Physical Address 0x5410_0024 **U2DOTGICR** **U2DC**

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	IESI	Reserved	IRELS1	IRELS0	IREID	IRESE	IRESV	IREVV	IRECK	Reserved	IFELS1	IFELS0	IFEID	IFESE	IFESV	IFEVV	IFECK
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
15	—	Reserved	**Reserved**
14	Read/Write	IRELS1	**RXCMD Linestate[1] Change Interrupt Rise Enable** 0 = No interrupt generated 1 = Generate an interrupt when the Linestate[1] bit in the ULPI RXCMD changes from low to high.
13	Read/Write	IRELS0	**RXCMD Linestate[0] Change Interrupt Rise Enable** 0 = No interrupt generated 1 = Generate an interrupt when the **Linestate[0]** bit in the ULPI RXCMD changes from low to high.
12	Read/Write	IREID	**RXCMD OTG ID Change Interrupt Rise Enable** 0 = No interrupt generated 1 = Generate an interrupt when the IDGnd bit in the ULPI RXCMD changes from low to high.
11	Read/Write	IRESE	**RXCMD OTG Session End Interrupt Rise Enable** 0 = No interrupt generated 1 = Generate an interrupt when the SessionEnd bit in the ULPI RXCMD changes from low to high.
10	Read/Write	IRESV	**RXCMD OTG Session Valid Interrupt Rise Enable** 0 = No interrupt generated 1 = Generate an interrupt when the SessionValid bit in the ULPI RXCMD changes from low to high.
9	Read/Write	IREVV	**RXCMD OTG Vbus Valid Interrupt Rise Enable** 0 = No interrupt generated 1 = Generate an interrupt when the VbusValid bit in the ULPI RXCMD changes from low to high.
8	Read/Write	IRECK	**RXCMD Carkit Interrupt Rise Enable** 0 = No interrupt generated 1 = Generate an interrupt when the CarkitInt bit is the ULPI RXCMD changes from low to high
7	—	Reserved	**Reserved**
6	Read/Write	IFELS1	**RXCMD Linestate[1] Change Interrupt Fall Enable** 0 = No interrupt generated 1 = Generate an interrupt when the Linestate[1] bit in the ULPI RXCMD changes from high to low. Linestate[1] maps to the D- of the PHY.

Table 60: U2DOTGICR Bit Definitions (Continued)

Physical Address 0x5410_0024	U2DOTGICR	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	IESI	Reserved	IRELS1	IRELS0	IREID	IRESE	IRESV	IREVV	IRECK	Reserved	IFELS1	IFELS0	IFEID	IFESE	IFESV	IFEVV	IFECK
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
5	Read/Write	IFELS0	**RXCMD Linestate[0] Change Interrupt Fall Enable** 0 = No interrupt generated 1 = Generate an interrupt when the Linestate[0] bit in the ULPI RXCMD changes from high to low. Linestate[0] maps to the D+ of the PHY.
4	Read/Write	IFEID	**RXCMD OTG ID Change Interrupt Fall Enable** 0 = No interrupt generated 1 = Generate an interrupt when the IDGnd bit in the ULPI RXCMD changes from high to low.
3	Read/Write	IFESE	**RXCMD OTG Session End Interrupt Fall Enable** 0 = No interrupt generated 1 = Generate an interrupt when the SessionEnd bit in the ULPI RXCMD changes from high to low.
2	Read/Write	IFESV	**RXCMD OTG Session Valid Interrupt Fall Enable** 0 = No interrupt generated 1 = Generate an interrupt when the SessionValid bit in the ULPI RXCMD changes from high to low.
1	Read/Write	IFEVV	**RXCMD OTG Vbus Valid Interrupt Fall Enable** 0 = No interrupt generated 1 = Generate an interrupt when the VbusValid bit in the ULPI RXCMD changes from high to low.
0	Read/Write	IFECK	**RXCMD Carkit Interrupt Fall Enable** 0 = No interrupt generated 1 = Generate an interrupt when the CarkitInt bit is the ULPI RXCMD changes from high to low

2.11.9 U2DC OTG Interrupt Status Register (U2DOTGISR) (PXA31x Processor Only)

This register contains the interrupt status bits for the OTG system. These bits are set only if the corresponding Interrupt Enable bit is set in the U2DOTGICR.

The ULPI interface provides the OTG interrupts from the off-chip PHY to the U2DOTGISR if they are enabled in the U2DOTGICR. Each bit in the U2DOTG Interrupt Status register is logically ANDed with its Interrupt Enable and then logically ORed together to produce one OTG interrupt signal which is logically ORed with the U2DMA interrupt and the U2DC interrupt to then generate the global U2DC interrupt signal (See Figure 34).

The register organization and individual bit definitions are shown in Table 61.

Table 61: U2DOTGISR Bit Definitions

Physical Address 0x5410_0028 **U2DOTGISR** **U2DC**

User Settings																															
Bit	31	30	29	28	27	26	25	24	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														ISSF	ISSI	Reserved	IRSLS1	IRSLS0	IRSID	IRSSE	IRSSV	IRVV	IRSCK	Reserved	IFSLS1	IFSLS0	IFSID	FISSE	IFSSV	IFSVV	IFSCK
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:18	—	Reserved	**Reserved**
17	Read/Write 1 to Clear	ISSF	**OTG Set Feature command received Status**
16	Read/Write 1 to Clear	ISSI	**OTG Interrupt Status** 1 = Generate an interrupt when the interrupt signal (ulpi_data[3]) is asserted during low-power, serial, or Carkit modes only.
15	—	Reserved	**Reserved**
14	Read/Write 1 to Clear	IRSLS1	**RXCMD Linetstate[1] Rise Interrupt Status** 1 = Linestate[1] maps to the D- of the PHY.
13	Read/Write 1 to Clear	IRSLS0	**RXCMD Linetstate[0] Rise Interrupt Status** 1 = Linestate[0] maps to the D+ of the PHY.
12	Read/Write 1 to Clear	IRSID	**RXCMD OTG ID Change Rise Interrupt Status**
11	Read/Write 1 to Clear	IRSSE	**RXCMD OTG Session End Rise Interrupt Status**
10	Read/Write 1 to Clear	IRSSV	**RXCMD OTG Session Valid Rise Interrupt Status**
9	Read/Write 1 to Clear	IRSVV	**RXCMD OTG Vbus Valid Rise Interrupt Status**
8	Read/Write 1 to Clear	IRSCK	**RXCMD Carkit Interrupt Rise Interrupt Status**
7	—	Reserved	**Reserved**
6	Read/Write 1 to Clear	IFSLS1	**RXCMD Linetstate[1] Fall Interrupt Status** 1 = Linestate[1] maps to the D- of the PHY.
5	Read/Write 1 to Clear	IFSLS0	**RXCMD Linetstate[0] Fall Interrupt Status** 1 = Linestate[0] maps to the D+ of the PHY.
4	Read/Write 1 to Clear	IFSID	**RXCMD OTG ID Change Fall Interrupt Status**
3	Read/Write 1 to Clear	IFSSE	**RXCMD OTG Session End Fall Interrupt Status**
2	Read/Write 1 to Clear	IFSSV	**RXCMD OTG Session Valid Fall Interrupt Status**

Table 61: U2DOTGISR Bit Definitions (Continued)

Physical Address 0x5410_0028	U2DOTGISR	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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														ISSF	ISSI	Reserved	IRSLS1	IRSLS0	IRSID	IRSSE	IRSSV	IRSVV	IRSCK	Reserved	IFSLS1	IFSLS0	IFSID	FISSE	IFSSV	IFSVV	IFSCK
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
1	Read/Write 1 to Clear	IFSVV	**RXCMD OTG Vbus Valid Fall Interrupt Status**
0	Read/Write 1 to Clear	IFSCK	**RXCMD Carkit Interrupt Fall Interrupt Status**

2.11.10 U2DC OTG ULPI Status Register (U2DOTGUSR) (PXA31x Processor Only)

This read-only register contains various ULPI related status bits. The register organization and individual bit definitions are shown below in Table 63.

Table 62: U2DOTGUSR Bit Definitions

Physical Address 0x5410_002C	U2DOTGUSR	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LPA	S6A	S3A	CKA	Reserved																					LS1	LS0	ID	SE	SV	VV	CK
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	Read	LPA	**ULPI Low Power Mode Active** 1 = ULPI PHY is programmed to low-power mode
30	Read	S6A	**ULPI Serial Mode (6-pin) Active** 1 = ULPI PHY is programmed to 6-pin serial mode
29	Read	S3A	**ULPI Serial Mode (3-pin) Active** 1 = ULPI PHY is programmed to 3-pin serial mode
28	Read	CKA	**ULPI Car Kit Mode Active** 1 = ULPI PHY is programmed to car kit mode
27:7	—	Reserved	**Reserved**
6	Read	LS1	**RXCMD Linestate 1 Status** Value of the **Linestate[1]** bit in the last ULPI RXCMD received from the PHY. Linestate[1] maps to the D- of the PHY.
5	Read	LS0	**RXCMD Linestate 0 Status** Value of the **Linestate[0]** bit in the last ULPI RXCMD received from the PHY. Linestate[0] maps to the D+ of the PHY.

Table 62: U2DOTGUSR Bit Definitions (Continued)

Physical Address 0x5410_002C	U2DOTGUSR	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LPA	S6A	S3A	CKA	Reserved																					LS1	LS0	ID	SE	SV	VV	CK
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
4	Read	ID	**OTG IDGnd Status** Value of the **IDGnd** bit in the last ULPI RXCMD received from the PHY
3	Read	SE	**OTG Session End Status** Value of the **SessionEnd** bit in the last ULPI RXCMD received from the PHY
2	Read	SV	**OTG Session Valid Status** Value of the **SessionValid** bit in the last ULPI RXCMD received from the PHY
1	Read	VV	**OTG Vbus Valid Status** Value of the **VbusValid** bit in the last ULPI RXCMD received from the PHY
0	Read	CK	**Carkit Interrupt Status** Value of the **CarkitInt** bit in the last ULPI RXCMD received from the PHY

2.11.11 U2DC OTG ULPI Control Register (U2DOTGUCR) (PXA31x Processor Only)

This register is how software accesses the ULPI PHY registers in the off-chip PHY. Many registers can be accessed as read, write, set (OR'd), and clear (Masked) depending on the address that is accessed as described in the specification. This is not used during "normal" operation, but is instead used for switching between ULPI and serial and other modes.

The register organization and individual bit definitions are shown below in Table 63.

Table 63: U2DOTGUCR Bit Definitions

Physical Address 0x5410_0030	U2DOTGUCR	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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						RUN	RNW	Reserved		ADDR						WDATA								RDATA							
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
31:26	—	Reserved	**Reserved**
25	Read/Write 1 to Set	RUN	**Run** Writing this bit to a 1 will start the Read/Write operation. The contents of the other fields in this register cannot be written to while this bit is set to a 1.This bit is automatically cleared by the hardware when the operation is complete. The software should poll this bit until is reads 0.

Table 63: U2DOTGUCR Bit Definitions (Continued)

Physical Address 0x5410_0030	U2DOTGUCR	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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						RUN	RNW	Reserved		ADDR						WDATA								RDATA							
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
24	Read/Write	RNW	**Read or Write operation** This field cannot be written to unless the RUN bit is zero 0 = Write 1 = Read
23:22	—	Reserved	**Reserved**
21:16	Read/Write	ADDR	**Address of the ULPI PHY register to be accessed** This field cannot be written to unless the RUN bit is zero
15:8	Read/Write	WDATA	**The data for a WRITE command** This field cannot be written to unless the RUN bit is zero
7:0	Read	RDATA	**The data returned from a READ command** This READ data is valid when the RUN bit is zero (after the READ command is complete)

2.11.12 U2DC Host Port 3 Control Register (U2DP3CR) (PXA31x Processor Only)

The U2DC Host Port 3 Output Control register (U2DP3CR) contains control bits that are used to control the USB_MUX for the Host Controller (UHC) Port 3 and the Serial mode selection for the UHC Port 2 and Port 3.

The register organization and individual bit definitions are shown in Table 64. All reserved bits are read as unknown values and must be written with only a 0. A question mark indicates that the value is unknown at reset.

Table 64: U2DP3CR Bit Definitions

Physical Address 0x5410_0034	U2DP3CR	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																						P2SS		Reserved	P3SS			Reserved	VPVMBEN	CFG	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	0	?	0	0	0

Bits	Access	Name	Description
31:3	—	Reserved	**Reserved**

Table 64: U2DP3CR Bit Definitions (Continued)

Physical Address 0x5410_0034	U2DP3CR	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																						P2SS		Reserved	P3SS			Reserved	VPVMBEN	CFG	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	0	?	0	0	0

Bits	Access	Name	Description
9:8	Read/Write	P2SS	**Host Port 2 Serial Mode Select** 0x0 – ULPI 6-pin serial mode 0x1 – ULPI 3-pin serial mode 0x2 – Reserved 0x3 – Reserved
7	—	Reserved	**Reserved**
6:4	Read/Write	P3SS	**Host Port 3 Serial Mode Select** 0x0 – 6-wire single ended mode 0x1 – Reserved 0x2 – 6-wire differential mode 0x3 – Reserved 0x4 – 3-wire CEA-2011 mode 0x5 – Reserved 0x6 – 4-wire CEA-2011 mode 0x7 – Reserved
3	—	Reserved	**Reserved**
2	Read/Write	VPVMBEN	**Host Port 3 Vp/Vm Block Enable** Control bit for single ended mode (applies to CFG 0x2 and 0x3 only) 0 - Enable 1 - Disable
1:0	Read/Write	CFG	**Host Port 3 Configuration:** 0x0 – Host controller Port 3 transceiver with external host controller 0x1 – Pass through mode when Client port2 connected to Host port3 0x2 – Host controller Port 3 transceiver with external device controller 0x3 – Host controller Port 3 transceiver with external device controller with VPout and VMout replaced by Tx_SE0 and Tx_SE0_N

2.11.13 U2DC Endpoint 0 Control/Status Register (U2DCSR0)

The U2DC Endpoint 0 Control/Status register, shown in Table 65, contains eight bits that are used to operate Endpoint 0 (control endpoint).

2.11.13.1 Control/Status Register Usage for Out Endpoint 0

The Byte Count register is only used and valid for Out Endpoint 0. It indicates the number of data bytes that need to be unloaded from the buffer for the packet being currently read by the system. As data is read from the FIFO memory, the Byte Count register value is decremented to indicate the number of bytes remaining in the packet. When all of the data has been unloaded from the FIFO

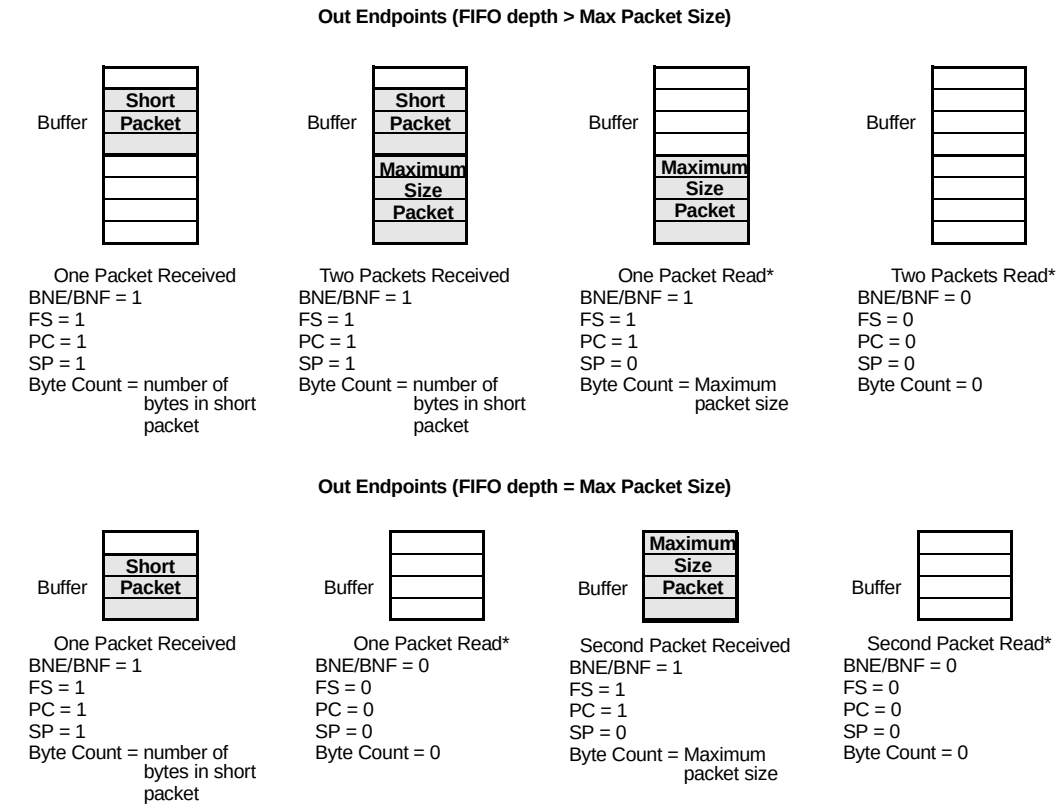
memory, the FIFI Service (FS) and Buffer Not Empty/ Buffer Not Full (BNE/BNF) bits in the Control/Status register (U2DCSRx) and the Byte Count (BC) in the U2DC Byte Count Register (U2DBCR0) are cleared to 0 by the U2DC to indicate that the current buffer is empty.

The Control/Status and Byte Count registers provide the status of the Endpoint FIFO. The FIFO Service (FS), PC, and BNE/BNF bits in the U2DC Endpoint 0 Control/Status Register (U2DCSR0) are set to 1 when the FIFO contains a complete data packet. The SP bit is set to 1 if the packet received is smaller than the maximum packet size for the endpoint. The communication of a packet completion and the requirement for software intervention depends on whether DMA access is enabled for an endpoint, and whether the automatic packet-completion control bit is enabled. As the data is read from the first Receive buffer, the value in the U2DC Byte Count Register (U2DBCR0) is decremented and indicates the number of data bytes that still need to be read from the current packet in the FIFO memory. When all of the data in the current packet has been read, the U2DC Byte Count Register (U2DBCR0) and the BNE/BNF bit is 0. If a second packet is received before all of the data has been read from the first packet, the FS bit continues to be set after the first packet has been read, but the second packet of data does not set the PC bit until after the processor clears the PC bit.

The full status of a subsequent packet cannot be determined until after the PC bit has been cleared. The PC bit must be cleared at which time the Control/Status and Byte Count registers are changed to reflect the status of the next packet.

If the second complete data packet has been read, the PC bit again is set to1 to indicate the Endpoint FIFO has data that is ready to be unloaded and the BNE/BNF and U2DC Byte Count Register (U2DBCR0) indicate the amount of data present in the second packet. At this point, the DMA can be used to unload data from the second packet. The Control/Status and Byte Count registers continue to hold the status of the second packet until the PC bit is again cleared. Once all of the data has been read from the second packet, write a 1 to the PC bit at which time the Control/Status and Byte Count registers are changed to reflect the status of the next packet. Figure 35 shows the relationship between data in the Endpoint buffers and the Status bits in the Endpoint Control/Status register.

Figure 35: Status Bits for Out Endpoints



*After PC bit cleared by Software/DMA

2.11.13.2 Control/Status Register Usage for In Endpoints

If an endpoint is configured as an In endpoint and has FIFO depth = Max Packet Size, the U2DC Endpoint 0 Control/Status Register (U2DCSR0) provides the status of the packet in the FIFO. The U2DC Byte Count Register (U2DBCR0) is not used for In endpoints, and is considered 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 U2DC when a complete packet has been loaded into the FIFO memory and is ready for transmission. Indicate a complete packet has been loaded by either writing the maximum packet size to the FIFO or by setting the packet complete bit in the U2DMA Descriptor (U2DMACMDx[PACKCOMP]) if the U2DMA is enabled (U2DCSRx[DME]=1). Once the data has been successfully transmitted to the USB host controller, the U2DC sets the PC bit to 1 to indicate the packet has been sent and the Error/Status bits in the Control/Status register are valid for the completed transaction.

Table 65: U2DCSR0 Bit Definitions

Physical Address 0x5410_0100 **U2DCSR0** **U2DC**

	31	30	29	28	27	26	25	24	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																							IPA	SA	RNE	FST	SST	DME	FTF	IPR	OPC
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:9	—	Reserved	**Reserved**
8	Read/Write	IPA	**In Packet Adjusted:** Since the processor core cannot issue writes of size 3 bytes, when this bit is set the U2DC subtracts 1 byte from a processor Write to the Endpoint 0 Data Register (U2DDR0). This bit should be written at the same time as IPR to adjust the processor Write of 4 bytes down to 3 bytes.If the number of bytes loaded through PIO was not 4, the resulting byte total to be transferred will be corrupted resulting in incorrect size data being transferred. IPA should only be set when IPR is set, and will be cleared at the same time as IPR when the packet is transferred over the USB bus or the packet is flushed. 0 = U2DDR0 is not adjusted 1 = Last PIO Write of 4 bytes to U2DDR0 is adjusted to 3 bytes
7	Read/Write 1 to Clear	SA	**Setup Active:** The Setup-active bit indicates the current packet in the endpoint 0 Receive FIFO is part of a USB Setup command. This bit will become active at the same time as out-packet-complete (OPC). Software must clear this bit by writing a 1 to it. SA must be cleared when OPC is cleared. 0 = Set-up command inactive 1 = Set-up command is active on the USB
6	Read	RNE	Receive Fifo Not Empty: The Receive-FIFO-Not-empty (RNE) bit 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 will be lost. 0 = Receive FIFO empty 1 = Receive FIFO not empty
5	Read/Write 1 to Set	FST	**Force Stall:** Set the Force Stall (FST) bit to 1 to force the U2DC to issue a Stall handshake. After the U2DC issues a Stall handshake for the current Control transfer, the bit is cleared by the U2DC. Though FST will be cleared, the U2DC will continue to respond to subsequent accesses to Endpoint 0 with a Stall handshake until an access to Endpoint 0 occurs with a SETUP packet id (that is, until a new set-up transfer is initiated.) 0 = No handshaking 1 = Force stall handshake

Table 65: U2DCSR0 Bit Definitions (Continued)

Physical Address 0x5410_0100 **U2DCSR0** **U2DC**

	31	30	29	28	27	26	25	24	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																							IPA	SA	RNE	FST	SST	DME	FTF	IPR	OPC
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
4	Read	SST	**Sent Stall:** The Sent Stall bit is set by the U2DC when it issues a Stall handshake as a result of the FST being set. Once SST is set, the U2DC will continue to respond to subsequent accesses to Endpoint 0 with a Stall handshake until an access to Endpoint 0 occurs with a Set-up packet id (that is, until a new setup transfer is initiated.). The Send Stall bit will be cleared after the new setup packet has been received successfully. 0 = No stall handshakes 1 = U2DC sent stall handshake and will continue to issue stall handshakes until the next setup command
3	Read/Write	DME	**DMA Enable:** The DMA Enable (DME) bit is used for Endpoint 0 Out transactions only, to request DMA reading of Endpoint 0 FIFO memory. DME is used by the U2DC to control the Endpoint Packet Complete interrupt and DMA request. If the bit is set to 0, 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 will be unloaded using the XScale® core, so an interrupt is generated instead of a DMA request. If DME is set to 1, the Packet Complete interrupt is not asserted, but a DMA request is generated to notify the DMA of the received data. DMA Enable should 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
2	Read 0/Write 1 to Set	FTF	**Flush Transmit FIFO:** Set the Flush Transmit FIFO (FTF) bit to reset the Endpoint 0 Transmit FIFO. FTF will be reset by the U2DC after the FIFO contents have been deleted.The Endpoint 0 Transmit FIFO is also flushed by the U2DC after the U2DC has received an Out packet from the USB Host controller. 0 = Outpacket received from UHC 1 = Flush the contents of Transmit FIFO
1	Read/Write 1 to Set	IPR	**In Packet Ready:** The IPR bit indicates a packet has been loaded and is ready for transmission when DME is not enabled (the DMA descriptor indicates this condition if DME is enabled.) Software must set IPR regardless of the number of bytes that have been written to the Endpoint 0 Transmit FIFO. There is no need to set IPR when using the U2DMA. The U2DC automatically clears IPR when the packet has been successfully transmitted or when the FIFO has been flushed. When IPR is cleared by the U2DC, the IR0[1] bit in U2DISR0 will be set if the Packet Complete Endpoint 0 interrupt is enabled. IPR cannot be cleared by software. 0 = Packet successfully transmitted or FIFO has been flushed 1 = IN packet has been loaded and is ready for transmission **NOTE:** The design was never intended to handle the clearing of OPC and setting of IPR at the same time. Programmers must clear OPC first, by writing it to 1, then set IPR in a separate register Write.

Table 65: U2DCSR0 Bit Definitions (Continued)

Physical Address 0x5410_0100	U2DCSR0																						U2DC									
User Settings																																
Bit	31	30	29	28	27	26	25	24	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																							IPA	SA	RNE	FST	SST	DME	FTF	IPR	OPC
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
0	Read/Write 1 to Clear	OPC	**Out Packet Complete:** The OPC bit is set by the U2DC when it receives a valid token to Endpoint 0. When OPC is set, the IR0[1] bit in the U2DC Interrupt Status register (U2DISR) is set if the Packet Complete Endpoint 0 Interrupt is enabled. OPC is cleared by writing a 1 to it, and it should 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 will be flushed and any data remaining in the FIFO is lost. The U2DC will not enter the Data phase of an Endpoint 0 transaction until the OPC bit is cleared. **NOTE:** OPC is set even if DME is set and must be cleared by software to enter the Data phase of the Endpoint 0 transaction. The U2DC will 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. **NOTE:** The design was never intended to handle the clearing of OPC and setting of IPR at the same time. Programmers must clear OPC first, by writing it to 1, then set IPR in a separate register Write. 0 = Out packet not received. 1 = Out packet received and ready for reading.

2.11.14 U2DC Endpoints Control Status Registers (U2DCSRx)

2.11.14.1 U2DC Endpoint A - G Control Status Registers (U2DCSRA-G)

Each of the 7 programmable Endpoints A–G use their Control/Status registers to control the behavior of the endpoint and to report status for that endpoint after USB enumeration. The U2DC Control Status registers are used to operate their respective endpoint.

2.11.14.2 U2DC Endpoint H - P Control Status Registers (U2DCSRH-P) (PXA31x Processor Only)

Each of the 7 programmable Endpoints H - P use their Control/Status registers (U2DCSRH–U2DCSRP) to control the behavior of the endpoint and to report status for that endpoint after USB enumeration. The U2DC Control Status registers are used to operate their respective endpoint.

Note

The DMA Enable (DME) bit should be disabled only when the DMA channel is stopped and if the DMA is not enabled, DMA should not be started (u2dmacsr#[run] should not be set to 1). Doing so causes unwanted behavior.

The register organization and individual bit definitions are shown in Table 66.

Table 66: U2DCSRx Bit Definitions

Physical Address **U2DC**
0x5410_0104–0x5410_011C — U2DCSRA–U2DCSRG
0x5410_0120–0x5410_013C — U2DCSRH–U2DCSRP (PXA31x Only)

User Settings																													
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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																					**BF/BE**	**DPE**	**FEF**	**SP**	**BNE/BNF**	**FST**	**SST**	**DME**	**TRN**	**PC**	**FS**
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:11	—	Reserved	**Reserved**
10	Read	BF/BE	**Buffer Full/Buffer Empty** The Buffer-full/Buffer-empty (BF/BE) bit indicates that the Receive buffer is full or that the Transmit buffer is empty. If an endpoint is configured as an In endpoint, the BF/BE means transmit buffer empty, and if an endpoint is configured as an Out Endpoint, the BF/BE means receive buffer full. If the endpoint is configured as an In endpoint, BF/BE will clear when the endpoint buffer space has at least one word of data loaded, or when a zero packet has been written to the FIFO and made ready for transmission. When set, it shows that the USB 2.0 has unloaded all the data or zero packets made ready by the system, and the system must place more data in the FIFO to accommodate additional In requests by the USB 2.0 host. If an endpoint is configured as an Out endpoint, BF/BE will clear when the endpoint buffer is emptied of at least one word of data. When set, it shows that the U2DC has fully loaded the receive FIFO, and that the system must remove data from the FIFO in order to accommodate additional Out requests by the UHC. For In endpoints (Buffer Empty): 0 = Current transmit buffer is not empty 1 = Current transmit buffer is empty For Out endpoints (Buffer Full): 0 = Current receive buffer is not full 1 = Current receive buffer is full
9	Read	DPE	**Data Packet Error (Isochronous Endpoints Only)** The data packet error (DPE) bit is used for endpoints configured as isochronous Out endpoints only. The DPE bit indicates bit-stuffing or CRC error has been detected on the active packet of isochronous Out data. DPE is valid while PC is set to 1. If an endpoint is configured as any type other than isochronous or as an In endpoint, DPE is not used. DPE is cleared when PC is cleared. 0 = No error detected on Out data packet 1 = Bit Stuffing or CRC error was detected on Out packet

Table 66: U2DCSRx Bit Definitions (Continued)

Physical Address **U2DC**
0x5410_0104–0x5410_011C — U2DCSRA–U2DCSRG
0x5410_0120–0x5410_013C — U2DCSRH–U2DCSRP (PXA31x Only)

	31	30	29	28	27	26	25	24	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																					BF/BE	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
8	Read/Write 1 to Set	FEF	**Flush Endpoint FIFO:** The FEF bit is used to reset the Endpoint Transmit or Receive FIFO by setting it. FEF will be cleared by the U2DC after the FIFO contents have been deleted. The Endpoint FIFO is flushed of *all* data stored within it, potentially including multiple packets of data, not just the current packet. If the U2DMA is in use, the DMA channel must be stopped by software before clearing the FIFO. Stopping the DMA channel is necessary because after the FIFO is flushed it will continue to receive the data in the middle of the packet, leading to corrupted data. If the USB is currently receiving or transmitting data, the transfer will be aborted with an error which will cause the USB to retry bulk or interrupt transfers. 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: **reserved** Out Endpoints: R	SP	**Short Packet Control/Status:** The short packet (SP) bit indicates a packet smaller than the maximum packet size has been received and is ready for unloading. If an endpoint is configured as an Out endpoint, when the U2DC sets the SP bit it indicates 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. The SP bit is not used if DME is set. If DME is set, a different method is used to signal short packets. The DMA descriptor indicates when the packet is complete for In endpoints (from which short packets are deduced.) A descriptor which the packet complete bit set which has a length field of 0 indicates that a zero packet is to be transmitted. For Out endpoints short packets are signalled through the short packet complete bit in the interrupt status register (U2DISR[IRx]). For Out endpoints: 1 = Short packet received and ready for reading

Table 66: U2DCSRx Bit Definitions (Continued)

Physical Address **U2DC**
0x5410_0104–0x5410_011C — U2DCSRA–U2DCSRG
0x5410_0120–0x5410_013C — U2DCSRH–U2DCSRP (PXA31x Only)

User Settings																													
Bit	31	30	29	28	27	26	25	24	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																					BF/BE	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
6	Read	BNE/BNF	**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 buffer not full, and if an endpoint is configured as an Out endpoint, the BNE/BNF means buffer not empty. These are valid for both when the U2DMA is enabled and when it is not. If the U2DMA is utilized to transfer data, the U2DC will automatically control transfer in both directions, asserting requests when the U2DMA has room to write data into the transmit FIFO for In endpoints, or when the U2DMA has data which can be read from the receive FIFO for Out endpoints. In the U2DMA enabled mode of operation this bit supplies information, but does not need to be accessed for proper operation. If the endpoint is configured as an In endpoint, BNE/BNF will clear when the endpoint buffer space has been filled with packet data and there is no room to accommodate data in the size of the LEN field in the DMA Descriptor if DME is set, or max packet size if DME is disabled. If the FIFO is loaded completely with packets of data, both the BNE/BNF and FS bits will be 0. A complete packet of data is signified by either loading a maximum size packet. If an endpoint is configured as an Out endpoint, BNE/BNF must be polled when the PC bit is set to 1 to determine if there is any data in the FIFO that needs to be unloaded. The U2DC 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 will not be set until the Out packet has been error-checked by the U2DC and found to be error-free. If the endpoint is an isochronous endpoint, BNE/BNF will not be set until the Out packet data has been error-checked by the U2DC 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 will not clear until the Read of the current buffer is complete. BNE/BNF will clear when the current packet is emptied and FS bit will not clear until the FIFO has been emptied of data. 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

Table 66: U2DCSRx Bit Definitions (Continued)

Physical Address **U2DC**
0x5410_0104–0x5410_011C — U2DCSRA–U2DCSRG
0x5410_0120–0x5410_013C — U2DCSRH–U2DCSRP (PXA31x Only)

	31	30	29	28	27	26	25	24	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																					BF/BE	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
5	Read/Write 1 to Set	FST	**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. To force the U2DC to issue a Stall handshake in response to all In token requests or all Out tokens, set the FST bit. The U2DC will clear the FST bit and set the SST bit when the Stall handshake has actually been sent to the USB Host controller and the Stall feature has been set for the particular endpoint (this may be delayed if the U2DC is active with another token when the FST bit is set). If the endpoint is configured as an In endpoint, the Transmit FIFO is flushed when SST is set. If the endpoint is configured as an Out endpoint, no action is taken on the Receive FIFO when SST is set. Any data in the endpoint Receive FIFO remains valid and must be unloaded by software. The Stall bits affect the behavior on the USB side of the FIFO, but do not block access on the system side of the FIFO. DMA access to the FIFO can continue as normal or until the FIFO is empty or full. Because the Transmit FIFO is flushed when SST is set, software must stop any DMA transfers on that channel before setting the FST bit on the endpoint that corresponds to the DMA channel in order to not lose or corrupt packet data. The Stall handshakes will continue to be sent until the USB Host controller clears the Stall feature for the particular endpoint. For In endpoints, the U2DC will clear the SST bit when a new packet has been successfully transmitted from the Transmit FIFO. For Out endpoints, the U2DC will clear the SST bit when a new packet has been successfully received into the Receive FIFO. 1 = Issue Stall handshakes to all bulk or interrupt tokens, (In or Out depending upon endpoint direction)
4	Read	SST	**Sent Stall:** 1 = Stall handshake was sent

Table 66: U2DCSRx Bit Definitions (Continued)

Physical Address **U2DC**
0x5410_0104–0x5410_011C — U2DCSRA–U2DCSRG
0x5410_0120–0x5410_013C — U2DCSRH–U2DCSRP (PXA31x Only)

	31	30	29	28	27	26	25	24	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																					BF/BE	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
3	Read/Write	DME	**DMA Enable:** The DMA Enable (DME) bit is set to request DMA reading or writing of Endpoint FIFO memory. If the DME bit is clear, the Packet Complete interrupt is asserted (if enabled) 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 processor. If DME is set, the Packet Complete interrupt is also asserted (if the interrupt is enabled), but a DMA request is also generated. This notifies the DMA of the received data for either a full packet or a short packet. If an endpoint is configured as an In endpoint, DME is used by the U2DC to enable the output of the DMA request. If the DME bit is clear, the DMA request is not generated, and software 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 U2DC will generate a DMA request when there is space in the Endpoint FIFO memory for the amount of data specified in the DMA Descriptor U2DMACMDx[LEN] field. The U2DC will continue to issue a DMA request until the complete packet has been filled or the U2DMACMDx[PACKCOMP] bit in the DMA descriptor has been set to 1. When DME is set, the PC bit will be 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. For In endpoints: 1 = Send DMA request when transmit FIFO has room for DMA Descriptor LEN worth of data For Out endpoints: 1 = Send data received DMA request after EOP received

Table 66: U2DCSRx Bit Definitions (Continued)

Physical Address **U2DC**
0x5410_0104–0x5410_011C — U2DCSRA–U2DCSRG
0x5410_0120–0x5410_013C — U2DCSRH–U2DCSRP (PXA31x Only)

	31	30	29	28	27	26	25	24	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																					BF/BE	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
2	Read/Write 1 to Clear	TRN	**Tx/Rx NAK:** The Transmit/ Receive NAK (TRN) bit is set by the U2DC 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 is full. The TRN bit is cleared by writing a 1 to it. For In endpoints, the TRN bit will be set by the U2DC when the U2DC 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 U2DC will issue NAK handshakes to In tokens while the endpoint buffer continues to be empty. If the endpoint is configured as an isochronous endpoint, the U2DC will send zero-size packets while the endpoint buffer is empty and if the FIFO Error interrupt is enabled, the U2DC will generate a FIFO Error interrupt for the endpoint. For Out endpoints, the TRN bit will be set by the U2DC when the host sends Out data while the endpoint buffer does not contain enough FIFO space for the packet based on the size specified in U2DCRx[MPS]. If the endpoint is configured as a bulk or interrupt endpoint, the U2DC will issue NAK handshakes to Out tokens while the endpoint buffer 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 will be dropped. If the FIFO Error interrupt is enabled, the U2DC also generates a FIFO Error interrupt for the endpoint. If U2DMA access occurs while a Transmit/Receive NAK is occurring, the U2DMA access is unaffected. When the U2DMA has finished loading a packet into the Transmit FIFO for In endpoints, or when the U2DMA has finished reading some data from the Receive FIFO for Out endpoints, this will enable the host to successfully complete the transaction on the next subsequent retry and avoid a 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 (endpoint buffer does not contain enough FIFO space for the packet based on the size specified in U2DCRx[MPS])

Table 66: U2DCSRx Bit Definitions (Continued)

Physical Address **U2DC**
0x5410_0104–0x5410_011C — U2DCSRA–U2DCSRG
0x5410_0120–0x5410_013C — U2DCSRH–U2DCSRP (PXA31x Only)

	31	30	29	28	27	26	25	24	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																					BF/BE	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
1	Read/Write 1 to Clear	PC	**Packet Complete:** The U2DC 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 U2DC when an entire packet has been received from the USB Host controller, and will generate a Packet Complete interrupt and/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 by the processor.The PC bit is cleared automatically by the U2DC if DME is enabled, but must be done by a Write to this register bit for processor accesses. Any data remaining in the Endpoint Receive buffer will be deleted when the PC bit is cleared. If an endpoint is configured as an In endpoint, the PC bit is set by the U2DC after an entire packet has been sent to the USB Host controller. DMA may load more data into the Transmit FIFO, regardless of the state of PC. When a packet is received by the U2DC, the PC is set by the U2DC, and the corresponding PC interrupt bit in the U2DISR will be set and an interrupt generated, if the interrupt is enabled. If the PC bit was set when a packet is received by the U2DC, the PC remains set. The corresponding PC interrupt bit in the U2DISR will be set, if it has not been set already, and an interrupt generated, if the interrupt is enabled. If the PC bit was set when a packet is received by the U2DC, the PC remains set and if DME is set, a DMA request is generated. 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

Table 66: U2DCSRx Bit Definitions (Continued)

Physical Address **U2DC**
0x5410_0104–0x5410_011C — U2DCSRA–U2DCSRG
0x5410_0120–0x5410_013C — U2DCSRH–U2DCSRP (PXA31x Only)

User Settings																													
Bit	31	30	29	28	27	26	25	24	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																					BF/BE	DPE	FEF	SP	BNE/BNF	FST	SST	DME	TRN	PC	FS
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
0	Read	FS	**FIFO Needs 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. The bit is valid during U2DMA access, however because the U2DMA is automated, there is no need for software to monitor this bit when DME is enabled. Instead the U2DMA relies on DMA requests when the FIFO is ready for access. If an endpoint is configured as an In endpoint, the U2DC sets FS when there is space for one or more Data packets (DME = 0) or data the size of the DMA Descriptor U2DMACMDx[LEN] field (DME=1) to be loaded into the Transmit FIFO. FS will clear when there is no more room for data in the Transmit FIFO. If an endpoint is configured as an Out endpoint, the FS bit will be set by the U2DC when the Receive FIFO has at least one complete data packet in it. FS will not clear until the Read of the FIFO is complete and there are no more pending packets. In the case of an uncompleted packet resident in the FIFO, FS will not be set until the U2DC has checked the Out packet for errors. 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

2.11.15 U2DC Byte Count Register (U2DBCR0)

The Byte Count register (U2DBCR0) maintains the remaining byte count in the packet currently being read by the system (DMA or processor) of Endpoint 0.

2.11.15.1 Byte Count

The register organizations and the individual bit definitions are shown in Table 67.

Table 67: U2DBCR0 Bit Definitions

Physical Address 0x5410_0200	U2DBCR0	U2DC

User Settings																														
Bit	31	30	29	28	27	26	25	24	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	0

Bits	Access	Name	Description
31:11	—	Reserved	**Reserved**
10:0	Read	BC	**Byte Count** For U2DMA access, the byte count is informational only. There is no need to reference the byte count register when using the U2DMA. The Byte Count bits are updated by the U2DC after each byte is read by software. Upon receiving an interrupt that indicates the endpoint has data or after the OPC bit in the corresponding U2DCSR0 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 0 Receive FIFO. The Byte Count register is decremented by the number of valid data bytes read (usually eight when accessed by the dedicated DMA, and four when accessed through the U2DC DATA register), each time data is read from the Endpoint 0 FIFO memory. The byte count cannot decrement to less than 0. The Byte Count register will indicate the number of bytes that need to be read by the system (U2DMA or processor) from the packet currently active for system access (the active packet.) The Byte Count register is decremented each time data is read from the buffer, and will be set to 0 when the current packet is emptied. The BNE/BNF bit in the Endpoint Control/Status register will be cleared by the U2DC when the byte count of the current packet is 0. If there is another completed packet in the FIFO, the FS bit will continue to be set to indicate there is still data in the Endpoint FIFO space. The BNE/BNF bit will be set when the next packet becomes the active packet, but will remain 0 while the first packet is the active packet. Continuing to read the Endpoint FIFO memory after the byte count is 0 will result in reading unknown data.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. The PC bit is cleared automatically by the U2DC when DME is active, and must be accomplished by the software for processor accesses. See Section 2.7.3 for more information on the Byte Count register.

2.11.16 U2DC End Point 0 Data Register (U2DDR0)

U2DC Endpoint 0 Data register is a 32 bits wide by 2*FIFO depth deep bidirectional FIFO. It appears half as wide and twice as deep as the actual FIFO memory to accommodate the 32-bit system bus access while still providing access to the full memory space. The U2DC automatically translates the 32-bit system bus accesses into 64-bit accesses to the FIFO memory. It can be used as an alternative to the internal dedicated DMA for accessing the USB data. When the USB host controller transmits data to U2DC Endpoint 0, the U2DC Endpoint 0 register can be read to access the data.

When the U2DC is sending data to the USB host controller, the data to be sent may be placed into the appropriate U2DC Endpoint register. Although Read and Write operations can be performed on a single FIFO during various points in a control sequence, the FIFO may not be read and written at the same time. The U2DC controls the direction that the FIFO is flowing. Refer to Section 2.7.3 for more details on accessing the Endpoint FIFO memory.

For Endpoint 0, normally the U2DC is in an Idle state, waiting for the USB host controller to send commands. When this occurs, the U2DC 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 writes 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.

The register organization and individual bit definitions are shown in Table 68.

Table 68: U2DDR0 Bit Definitions

Physical Address 0x5410_0300	U2DDR0	U2DC

Bit	31	30	29	28	27	26	25	24	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: Write Out endpoints: Read Endpoint 0: Read/Write	ED	Endpoint Data

2.11.17 U2DC Endpoint Configuration Registers (U2DCRx)

Upon power up, the U2DC is disabled and all configurations, interfaces, and alternate interface settings are disabled. Software must program the Endpoint Configuration and Endpoint Information registers before enabling the U2DC.

2.11.17.1 U2DC Endpoints A - G Configuration Registers (U2DCRA-G)

The U2DC Configuration registers U2DCRA-U2DCRG, shown in Table 69 are used to define FIFO depth and enable the programmable endpoints that are active for each particular configuration/interface/alternate interface setting combination.

2.11.17.2 U2DC Endpoints H - P Configuration Registers (U2DCRH-P) (PXA31x Processor Only)

The U2DC Configuration registers U2DCRH-U2DCRP, shown in Table 69 are used to define FIFO depth and enable the programmable endpoints that are active for each particular configuration/interface/alternate interface setting combination.

After updating these registers, the U2DCR[SMAC] must be set before the new settings take effect.

Table 69: U2DCRx Bit Definitions

Physical Address **U2DC**
0x5410_0404–0x5410_041C — U2DCRA–U2DCRG
0x5410_0420–0x5410_043C — U2DCRH–U2DCRP (PXA31x Only)

User Settings																														
Bit	31	30	29	28	27	26	25	24	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																					BS										EE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:11	—	Reserved	**Reserved**
10:1	Read/Write	BS	**Buffer Size:** Size of the FIFO for endpoint data. Must be set larger than the endpoint's max packet size, but is not restricted to an integer of max packet size. Endpoints with small max packet sizes may require a FIFO depth several times their max packet size in order to not be limited by system latency, while endpoints with large max packet sizes may tolerate a FIFO depth not even twice their max packet size and not be limited by system latency. Buffer size = BS * 8 bytes (for example. the minimum BS setting of 1 represents 8 bytes and the maximum BS setting of 0x3FF represents 8184 bytes, and all buffer sizes are a multiple of 8 bytes) **NOTE:** Minimum value allowed for Buffer size is 2 (buffer size = 16 bytes). See Section 2.7.3.3 for more information on memory allocation.
0	Read/Write	EE	**Endpoint Enable:** Set the Endpoint Enable (EE) bit to enable the endpoint and reserve FIFO memory space for the endpoint. If an endpoint is not enabled, it will not be allocated memory during USB operation. When EE is cleared, this action only affects the memory allocation. If the corresponding endpoint information register keeps a valid configuration, the U2DC will respond to host accesses to those endpoints even though there is no memory allocated for them. This will result in incorrect operation (Stall response). Any time an endpoint has EE cleared, the corresponding U2DC Endpoint Information Registers (U2DENx) should first be cleared. 0 = Endpoint is disabled and no FIFO memory is allocated 1 = Endpoint is enabled and allocated FIFO memory

2.11.18 U2DC Endpoint 0 Information Register (U2DEN0)

The U2DC Endpoint 0 Information register (U2DEN0, shown in Table 70) is used to define max packet size for Endpoint 0. Other fields for Endpoint Type, Interface Number, and so on, are not needed because these characteristics are fixed for Endpoint 0 in the IP. Max Packet size reset value is 64 bytes* and must never be written to any other value to ensure proper operation, both in high-speed and full-speed modes. Therefore, if software writes to U2DEN0, it must write 0x0200_0000.

*The reset value for the Max Packet Size (MPS) field is 64; however, whenever U2DCR[UDE]==0, the MPS field always reads back with a value of 0.

Table 70: U2DEN0 Bit Definitions

Physical Address 0x5410_0504	U2DEN0	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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		MPS											Reserved																		
Reset	?	?	0	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:30	—	Reserved	**Reserved**
29:19	Read/Write	MPS	**Maximum Packet Size,** 64 is the only valid value for this implementation, therefore U2DEN0 = 0x0200_0000. **NOTE:** After setting UDCCR[UDE] to 1 U2DEN0[25] is set to 1.
18:0	—	Reserved	**Reserved**

2.11.19 U2DC Endpoint Information Registers (U2DENx)

These registers are active even if the U2DCRx[EE] Endpoint Enable bit is cleared. This status means that the U2DC responds to host requests to the endpoint, even though no memory is allocated for it in the FIFO. Because this action causes incorrect behavior (stall response), the Endpoint Information registers must be cleared before clearing the U2DCRx[EE] bit in the Configuration register.

When updating these registers, the corresponding Endpoint FIFO must be flushed and associated ConfigurationStatus registers cleared; otherwise, the FIFO and U2DCSRx registers retains data and status from the outdated setup.

Upon power up, the U2DC is disabled and all configurations, interfaces, and alternate interface settings are disabled. Software must program the Endpoint Information registers before enabling the U2DC.

The U2DC Endpoint Information registers can be changed after the U2DC has been enabled for USB operation, but these should only be updated after a Set_configuration or Set_interface command has been submitted by the USB host.

Note

Software must always clear all Endpoint Enable (EE) bits in the U2DC Endpoint Configuration Registers (U2DCRx) whenever UDE is cleared to 0. The Endpoint Enable bit for each endpoint should only be set to 1 again after the software has set UDE to 1 and re-programmed that endpoint U2DC Endpoint Information Registers (U2DENx).

2.11.19.1 U2DC Endpoint A - G Information Registers (U2DENA - G)

The U2DC Endpoint Information registers U2DENA–U2DENG, shown in Table 71) are used to define endpoint number, interface and alternate interface for the programmable endpoints that are active for each particular configuration/interface/alternate interface setting combination. They can be

written only while U2DCR[UDE] is set. This register is held in its reset state while U2DCR[UDE] is clear.

2.11.19.2 U2DC Endpoint H - P Information Registers (U2DENH - P) (PXA31x Processor Only)

The U2DC Endpoint Information registers U2DENH–U2DENP, shown in Table 71, are used to define endpoint number, interface and alternate interface for the programmable endpoints that are active for each particular configuration/interface/alternate interface setting combination. They can be written only while U2DCR[UDE] is set. This register is held in its reset state while U2DCR[UDE] is clear.

Table 71: U2DENx Bit Definitions

Physical Address **U2DC**
0x5410_0508–0x5410_0520 U2DENA–U2DENG
0x5410_0524–0x5410_0540 U2DENH–U2DENP(PXA31x Only)

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HBW		MPS											AISN				IN				CN				ET		ED	EN			
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	Read/Write when UDE=1	HBW	**High Bandwidth Commands per Micro-frame:** For high-speed isochronous and interrupt In endpoints only this field represents how many accesses per micro-frame is issued. Values of 0 or 1 represent a normal bandwidth of a single access per micro-frame. Values of 2 or 3 represent a high bandwidth endpoint with the corresponding number of accesses each micro-frame. This field is ignored during full speed mode, for Out endpoints, and for bulk and control endpoints.
29:19	Read/Write when UDE=1	MPS	**Maximum Packet Size:** Valid values are defined by endpoint type: For interrupt endpoint: 1 - 64 (1-1024 for high speed) For bulk endpoint: 8, 16, 32, 64 (512 for high speed) For isochronous endpoint: 1 - 1023 (1-1024 for high speed)
18:15	Read/Write when UDE=1	AISN	**Alternate Interface Number:** AIN alternate interface setting numbers for Endpoints must be set before the UCD is enabled. Refer to Figure 32 for more details on the Configuration register programming sequence. Each Endpoint can be assigned to alternate interface setting 0 - 15. Each Endpoint can be assigned to only one configuration/interface/alternate interface setting combination, and will be active only when the USB Host controller configures the U2DC for that configuration, interface, and alternate interface setting. 0 - 15 – USB Interface Alternate Interface Setting Number

Table 71: U2DENx Bit Definitions (Continued)

Physical Address **U2DC**
0x5410_0508–0x5410_0520 U2DENA–U2DENG
0x5410_0524–0x5410_0540 U2DENH–U2DENP(PXA31x Only)

	31	30	29	28	27	26	25	24	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
	HBW		MPS											AISN				IN				CN				ET		ED	EN			
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
14:11	Read/Write when UDE=1	IN	**Interface Number:** Interface number must be set before the U2DC is enabled. Refer to Figure 32 for more details on the Configuration register programming sequence.Each Endpoint can be assigned to Interface 0 - 15. Each Endpoint can be assigned to only one configuration/interface/alternate interface setting combination, and will be active only when the USB Host controller configures the U2DC for that configuration, interface, and alternate interface setting. 0 - 15 – USB Interface Number
10:7	Read/Write when UDE=1	CN	**Configuration Number:** Configuration number must be set before the U2DC is enabled. Refer to Figure 43 for more details on the Configuration register programming sequence. Each Endpoint can be assigned to Configuration 1-15. Each Endpoint can be assigned to only one configuration/interface/alternate interface setting combination, and will be active only when the USB Host controller configures the U2DC for that configuration, interface, and alternate interface setting. 1 - 15 – USB Configuration Number
6:5	Read/Write when UDE=1	ET	**USB Endpoint Type:** 11 – Interrupt 10 – Bulk 01 – Isochronous 00 – Not used
4	Read/Write when UDE=1	ED	**USB Endpoint Direction:** The Endpoint Direction bit determines whether the endpoint is a an In endpoint, which sends data from the U2DC to the USB Host controller, or an Out endpoint, which receives data from the USB Host controller. 0 = Out 1 = In
3:0	Read/Write when UDE=1	EN	**Endpoint Number:** Endpoint number is used to define the endpoint number. Each Endpoint 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 must not 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

2.11.20 U2DMA Control Register (U2DMACR)

The U2DMA Control Register contains general system bus configuration settings for the U2DMA. The MAXOCT field determines the maximum number of outstanding concurrent transactions

allowed on the system bus. The RETRYTOEN bit enables/disables the system bus retry time-out checking for all U2DMA-initiated system bus transactions.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 72: U2DMACR Bit Definitions

Physical Address 0x5410_1080												U2DMACR																U2DC		
User Settings																														
Bit	31	30	29	28	27	26	25	24	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																													RETRYTOEN	MAXOCT	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0

Bits	Access	Name	Description
31:3	—	Reserved	**Reserved**
2	Read/Write	RETRYTOEN	**Internal System Bus Retry Time-out Enable:** 0 = Disable the system bus retry time-out counter, and do not assert channel interrupt bits for retries on the bus. 1 = Enable the system bus retry time-out counter, and assert an interrupt bit for the channel in the U2DMAINT[CHLINTR] when greater than 64 retries of a single transaction occur for the channel.
1:0	Read/Write	MAXOCT	**Maximum Number Of Outstanding Concurrent Transactions On The System Bus For All Channels**: 00 = 4 outstanding concurrent transactions allowed 01 = 1 outstanding concurrent transaction allowed 10 = 2 outstanding concurrent transactions allowed 11 = 3 outstanding concurrent transactions allowed

2.11.21 U2DMA Descriptor Address Registers (U2DMADADRx)

The bits in this register, except U2DMADADRx[STOP], are undefined on power up. U2DMADADRx[STOP] is cleared on power up. Even though the Stop bit is cleared on power up, this does not imply that the channel is running. The run/stop status of the channel is controlled by U2DMACSRx[RUN]. The U2DMADADRx[STOP] controls whether the channel is stopped after the Descriptor is completely processed. The address must be aligned to a 128-bit (4-word) boundary. U2DMADADRx must not contain the address of any other internal peripheral register or DMA register, as this causes a bus error.

2.11.21.1 U2DMA Channel 0 - 7 Descriptor Address Registers (U2DMADADR0-7)

The registers shown in Table 73 contain the memory address of the next Descriptor for Channels 0-7.

2.11.21.2 U2DMA Channel 8 - 15 Descriptor Address Registers U2DMADADR8-15) (PXA31x Processor Only)

The registers shown in Table 73 contain the memory address of the next Descriptor for Channels 8-15.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 73: U2DMADADRx Bit Definitions

Physical Address **U2DC**
0x5410_1200–0x5410_1270 — U2DMADADR0–U2DMADADR7
0x5410_1280–0x5410_12F0 — U2DMADADR8–U2DMADADR15 (PXA31x Only)

	31	30	29	28	27	26	25	24	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	DESCRIPA																												reserved			STOP
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
31:4	Read/Write	DESCRIPA	Descriptor Address Contains Address of Next Descriptor: Address must be aligned to a 128-bit (4-word) boundary.
3:1	—	Reserved	**Reserved**
0	Read/Write	STOP	If U2DMADADRx[STOP] is cleared, a new descriptor fetch based on register U2DMADADRx 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 (U2DMACMDx[LEN] = 0).

2.11.22 U2DMA Source Address Registers (U2DMASADRx)

If the DMA transfer is set up for an Out endpoint (U2DMACMDx[XFRDIR] = 0), the U2DMASADRx register is not used and has no effect on the transfer. In this instance, the channel number defines the Endpoint FIFO that is accessed as the source for the transfer. U2DMASADR cannot contain addresses of any other internal DMA registers for In endpoint transfers as this causes a bus error.

The address can be aligned to a byte boundary if the source address is the address of a memory location.

On power-up, the bits in this register are reset to 0.

2.11.22.1 U2DMA Channel 0 - 7 Source Address Registers (U2DMASADR0 - 7)

The registers shown in Table 74 contain the source address of the current Descriptor for Channels 0-7. The source address is the address of a memory location for In endpoint transfers.

2.11.22.2 U2DMA Channel 8 - 15 Source Address Registers (U2DMASADR8 - 15) (PXA31x Processor Only)

The registers shown in Table 74 contain the source address of the current Descriptor for Channels 8-15. The source address is the address of a memory location for In endpoint transfers.

These are read-only registers.

Table 74: U2DMASADRx Bit Definitions

Physical Address **U2DC**
0x5410_1204–0x5410_1274 — U2DMASADR0–U2DMASADR7
0x5410_1284–0x5410_12F4 — U2DMASADR8–U2DMASADR15 (PXA31x Only)

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	0	0	0	0	0	0	0	0	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	Read	SRCADDR	**Source Address of a Memory Location for In Endpoint Transfers:** If the transfer is for an Out endpoint, this register is not used.

2.11.23 U2DMA Target Address Registers (U2DMATADRx)

If the DMA transfer is set up for an In endpoint (U2DMACMDx[XFRDIR] = 1), the U2DMATADRx register is not used and has no effect on the transfer. In this instance, the channel number defines the Endpoint FIFO that is accessed as the target for the transfer. U2DMATADR cannot contain addresses of any other internal DMA registers for Out endpoint transfers as this causes a bus error.

The address can be aligned to a byte boundary if the target address is the address of a memory location.

On power up, the bits in this register are reset to 0

2.11.23.1 U2DMA Channel 0 - 7 Target Address Registers (U2DMATADR0-7)

The registers in Table 75 contain the target address of the current Descriptor for Channels 0 - 7. The target address is the address of a memory location for Out endpoint transfers.

2.11.23.2 U2DMA Channel 8 - 15 Target Address Registers (U2DMATADR8-15) (PXA31x Processor Only)

The registers in Table 75 contain the target address of the current Descriptor for Channels 8 - 15. The target address is the address of a memory location for Out endpoint transfers.

These registers are read-only registers.

Table 75: U2DMATADRX Bit Definitions

U2DC

Physical Address
0x5410_1208–0x5410_1278 — U2DMATADR0–U2DMATADR7
0x5410_1288–0x5410_12F8 — U2DMATADR8–U2DMATADR15 (PXA31x Only)

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	0	0	0	0	0	0	0	0	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	Read	TRGADDR	**Target Address of a Memory Location for Out Endpoint Transfers:** If the transfer is for an In endpoint, this register is not used.

2.11.24 U2DMA Command Registers (U2DMACMDx)

The bits in these registers are cleared to 0 on power up.

2.11.24.1 U2DMA Channel 0 - 7 Command Registers (U2DMACMD0-7)

The registers shown in Table 76 contain the command and length of the current Descriptor for Channels 0-7.

2.11.24.2 U2DMA Channel 8 - 15 Command Registers (U2DMACMD8-15) (PXA31x Processor Only)

The registers shown in Table 76 contain the command and length of the current Descriptor for Channels 8-15.

These registers are read-only registers.

Table 76: U2DMACMDx Bit Definitions

Physical Address
0x5410_1210–0x5410_127C — U2DMACMD0–U2DMACMD7 **U2DC**
0x5410_128C–0x5410_12FC — U2DMACMD8–U2DMACMD15 (PXA31x Only)

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	XFRDIR	Reserved								STARTIRQEN	EndIrqEn	Reserved							PACKCOMP	Reserved		LEN										
Reset	0	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	?	0	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	Read	XFRDIR	**Transfer Direction:** If the source for the DMA transfer is an Endpoint FIFO address and the target for the transfer is system memory, this bit must be cleared. If the source for the DMA transfer is system memory and the target for the transfer is an Endpoint FIFO, this bit must be set to one. 0 = DMA transfer from Endpoint FIFO to system memory. 1 = DMA transfer from system memory to Endpoint FIFO.
30:23	—	—	**Reserved**
22	Read	STARTIRQEN	**Start Interrupt Enable:** Indicates that the interrupt is enabled as soon as the descriptor is loaded. 0 = Interrupt not triggered after descriptor is loaded. 1 = Set interrupt bit for that channel in the U2DINT[CHLINTR] when the descriptor (4 words) for the channel is loaded.
21	Read	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.
20:14	—	Reserved	**Reserved**

Table 76: U2DMACMDx Bit Definitions (Continued)

Physical Address
0x5410_1210–0x5410_127C — U2DMACMD0–U2DMACMD7 **U2DC**
0x5410_128C–0x5410_12FC — U2DMACMD8–U2DMACMD15 (PXA31x Only)

	31	30	29	28	27	26	25	24	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	XFRDIR	Reserved								STARTIRQEN	EndIrqEn	Reserved							PACKCOMP	Reserved		LEN										
Reset	0	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	?	0	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
13	Read	PACKCOMP	**Packet Complete:** Provides communication to the Endpoint FIFO logic that at the end of the current descriptor, the present packet is complete. The Endpoint control logic should take the necessary steps to conclude the packet on the USB interface when this bit is set. This bit is only valid for In endpoints (DMA transfers from memory to Endpoint FIFO). 0 = DMA should not assert packet complete status to the Endpoint control logic at the end of this descriptor. (Valid for In endpoints only.) 1 = DMA should assert packet complete status to the Endpoint control logic at the end of this descriptor. (Valid for In endpoints only.)
12:11	—	—	**Reserved**
10:0	Read	LEN	**Length (LEN):** Provides the length of transfer in bytes. LEN = 0 signifies a zero length packet, and is only valid if the PACKCOMP bit is also set. The maximum transfer length is equal to the endpoint's maximum packet size. The largest possible maximum packet size for USB 2.0 is 1024 bytes.

2.11.25 U2DMA Channel Control/Status Registers (U2DMACSRx)

2.11.25.1 U2DMA Channel 0 - 7 Control/Status Registers (U2DMACSR0-7)

The registers shown in Table 77 contain the control and status bits for the Channels 0-7.

2.11.25.2 U2DMA Channel 8 - 15 Control/Status Registers (U2DMACSR8-15) (PXA31x Processor Only)

The registers shown in Table 77 contain the control and status bits for the Channels 8-15.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 77: U2DMACSRx Bit Definitions

Physical Address **U2DC**
0x5410_1000–0x5410_101C — U2DMACSR0–U2DMACSR7
0x5410_1020–0x5410_103C — U2DMACSR8–U2DMACSR15 (PXA31x Only)

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	reserved	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	Reserved		RASIrqEn	MaskRun	Reserved		SCEMC		SCEMI					BUSERRTYPE			EORINTR	REQPEND	Reserved			RASIntr	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	?	0	0	0	0	?	?	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
31	Read/Write	RUN	**Run:** Bit 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 U2DMADADRx, no descriptor fetch occurs. This bit is reset as soon as it is cleared and when the channel stops normally. U2DMACSRx[STOPINTR] must be polled to see the status of the channel or to set U2DMACSRx[STOPIRQEN] and expect an interrupt after the channel stops. After the channel stops, U2DMACSRx[STOPINTR]is set. STOPINTR must be polled to read the channel status or to set U2DMACSRx[STOPIRQEN] and expect an interrupt after the channel stops. 0 = Stops the channel 1 = Starts the channel **Warning** If the DMA is not enabled (U2DCSRx[DME] = 1), the DMA should not be started (U2DMACSRx[RUN] should not be set to 1). Doing so will cause unwanted behavior.
30	—	—	**Reserved**
29	Read/Write	STOPIRQEN	**Stop Interrupt Enabled:** Bit enables the interrupt when U2DMACSRx[STOPINTR] is set. If STOPINTEN is cleared, no interrupt is generated after the channel stops. U2DMACSRx[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	Read/Write	EORIRQEN	**End-of-Receive Interrupt Enable:** Setting bit triggers an interrupt on an EOR. Clearing this bit does not generate an EOR-related interrupt. 0 = Interrupt not triggered even if U2DMACSRx[EORINT] is set. 1 = Enable Interrupt if U2DMACSRx[EORINT] is set.

Table 77: U2DMACSRx Bit Definitions (Continued)

Physical Address **U2DC**
0x5410_1000–0x5410_101C — U2DMACSR0–U2DMACSR7
0x5410_1020–0x5410_103C — U2DMACSR8–U2DMACSR15 (PXA31x Only)

User Settings																														
Bit	31	30	29	28	27	26	25	24	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	reserved	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	Reserved		RASIrqEn	MaskRun	Reserved		SCEMC		SCEMI					BUSERRTYPE			EORINTR	REQPEND	Reserved			RASIntr	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	?	0	0	0	0	?	?	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
27	Read/Write	EORJMPEN	**Jump to the Next Descriptor:** EORJMPEN on EOR bit controls the descriptor flow when the endpoint signals an EOR to the U2DMA. Setting EORJMPEN causes the U2DMA to jump to the next descriptor on an EOR. Clearing EORJMPEN causes the U2DMA to wait for the endpoint's next receive request. See Figure 23 for the behavior of the descriptor during this condition. 0 = U2DMA continues to hold the current descriptor and simply waits until the mapped endpoint makes another receive request. 1 = U2DMA jumps to the channel's next descriptor on receiving an EOR from the mapped endpoint.
26	Read/Write	EORSTOPEN	**Stop the channel on an EOR:** Setting EORSTOPEN causes the U2DMA to stop the channel on an EOR and set the corresponding U2DMACSRx[STOPINTR] bit. If U2DMACSRx[STOPIRQEN] is set when EORSTOPEN is set, an interrupt occurs. See Figure 23 for the behavior of the descriptor. 0 = U2DMA continues to hold the current descriptor and waits until the mapped endpoint makes another receive request. 1 = U2DMA stops the channel that received an EOR from the mapped endpoint.
25:24	—	—	**Reserved**
23	Read/Write	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 U2DMAINT[CHLINTR] when a peripheral asserts a DMA request after the channel has stopped.
22	Write	MaskRun	**Mask Run:** Mask U2DMACSRx[Run] during a Programmed IO Write to the U2DMACSRx register. 0 = Software (programmed IO Write) will be able to modify U2DMACSRx[Run] during a Write transaction in which U2DMACSRx[MaskRun] is 0. 1 = Software (programmed IO Write) will not be able to modify U2DMACSRx[Run] during a Write transaction in which U2DMACSRx[MaskRun] is 1.
21:20	—	—	**Reserved**

Table 77: U2DMACSRx Bit Definitions (Continued)

Physical Address **U2DC**

0x5410_1000–0x5410_101C — U2DMACSR0–U2DMACSR7

0x5410_1020–0x5410_103C — U2DMACSR8–U2DMACSR15 (PXA31x Only)

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	reserved	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	Reserved		RASIrqEn	MaskRun	Reserved		SCEMC		SCEMI					BUSERRTYPE			EORINTR	REQPEND	Reserved			RASIntr	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	?	0	0	0	0	?	?	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
19:18	Read	SCEMC	**System Bus Split Completion Error Message Class.** This bit field is only valid when the BUSERRINTR bit is set and the BUSERRTYPE = 0b011. 00 = Write Completion 01 = System Bus Bridge Error 10 = System Bus Completer Error 11 = **reserved**
17:13	Read	SCEMI	**System Bus Split Completion Error Message Index.** This bit field is only valid when the BUSERRINTR bit is set and the BUSERRTYPE = 0b011. If BUSSCEMC = 0b00 00000 = Normal completion All other values **reserved** If BUSSCEMC = 0b01 00000 = System bus bridge received a master abort on the destination 00001 = System bus bridge received a target abort on the destination All other values **reserved** If BUSSCEMC = 0b10 00000 = System bus completer byte count out of range 1XXXX = System bus completer device-specific error. SCEMI[3:0] is used to encode a device-specific error code. All other values **reserved**

Table 77: U2DMACSRx Bit Definitions (Continued)

Physical Address **U2DC**
0x5410_1000–0x5410_101C — U2DMACSR0–U2DMACSR7
0x5410_1020–0x5410_103C — U2DMACSR8–U2DMACSR15 (PXA31x Only)

User Settings																														
Bit	31	30	29	28	27	26	25	24	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	reserved	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	Reserved		RASIrqEn	MaskRun	Reserved		SCEMC		SCEMI					BUSERRTYPE			EORINTR	REQPEND	Reserved			RASIntr	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	?	0	0	0	0	?	?	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
12:10	Read	BUSERRTYPE	**System Bus Error Type.** This bit field is only valid when the BUSERRINTR bit is set, and it applies to channel-related, U2DMA-initiated data transfer errors on the System Bus. BUSSERRTYPE is cleared by clearing BUSERRINTR. 000 = No System Bus Error exists 001 = System Bus Master Abort on a U2DMA-initiated transaction 010 = System Bus Target Abort on a U2DMA-initiated transaction 011 = System Bus Split Completion Error on a U2DMA-initiated transaction. (See SCEMC and SCEMI for details of error.) 100 = System Bus Retry Timeout on a U2DMA-initiated transaction. (Must be enabled by setting U2DMACR[RETRYTOEN] = 1.) All other values **reserved**
9	Read/Write 1 to Clear	EORINTR	**End of Receive:** The End of Receive bit indicates the status of the mapped endpoint's receive data. EORINT is set after the U2DMA reads out the last trailing sample from the peripheral's receive FIFO. Figure 23 illustrates the behavior of the descriptor during this condition. To clear EORINT, write 0b1 to the bit. 0 = DMA continues with current descriptor because the endpoint is still actively receiving data. 1 = Channel mapped endpoint has no data remaining in its receive FIFO and has completed all receive transactions. Refer to the description of U2DMACSRx[EORJMPEN] for the behavior of the U2DMA during this condition. The behavior of this bit is controlled by the U2DCR[SPEOREN] bit. **Note: This bit should be ignored for IN endpoints and must be cleared when changing the direction of the endpoint for the current channel.*

Table 77: U2DMACSRx Bit Definitions (Continued)

Physical Address **U2DC**

0x5410_1000–0x5410_101C — U2DMACSR0–U2DMACSR7

0x5410_1020–0x5410_103C — U2DMACSR8–U2DMACSR15 (PXA31x Only)

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	reserved	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	Reserved		RASIrqEn	MaskRun	Reserved		SCEMC		SCEMI					BUSERRTYPE			EORINTR	REQPEND	Reserved			RASIntr	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	?	0	0	0	0	?	?	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
8	Read	REQPEND	**Request Pending:** The Request Pending bit 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 U2DMACSRx[RUN] is cleared to stop that channel, REQPEND and the internal registers that hold the DREQ assertion information, will not remain set. If the channel is restarted, REQPEND must be reset by a descriptor that transfers dummy data. 0 = No request is pending for the channel. 1 = A request is pending for the channel.
7:5	—	Reserved	**Reserved**
4	Read/Write 1 to Clear	RASIntr	**Request After Channel Stopped:** The Request After Channel Stopped bit indicates that an Endpoint FIFO has generated a request for a channel that has completed its last descriptor and has no new descriptors in its descriptor chain. This error condition must be handled either by setting up a new descriptor or through programmed I/O. A new descriptor must be loaded and enabled prior to clearing this bit by writing a 1 to it, or the bit will only clear for a single clock. 0 = No interrupt 1 = Interrupt caused due to a request made by the Endpoint FIFO after the respective channel stopped.
3	Read	STOPINTR	**Stop Interrupt:** Stop Interrupt is a read-only bit that reflects the channel's state. If the channel is un-initialized or stopped, STOPINTR is set. If U2DMACSRx[STOPIRQEN] is set, the U2DMA generates an interrupt. Software must clear U2DMACSRx[STOPIRQEN] to reset the interrupt. Reprogramming U2DMADADRx and setting U2DMACSRx[RUN] restarts the channel. 0 = The channel is running 1 = The channel is in uninitialized or stopped state.

Table 77: U2DMACSRx Bit Definitions (Continued)

Physical Address **U2DC**
0x5410_1000–0x5410_101C — U2DMACSR0–U2DMACSR7
0x5410_1020–0x5410_103C — U2DMACSR8–U2DMACSR15 (PXA31x Only)

User Settings																														
Bit	31	30	29	28	27	26	25	24	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	reserved	STOPIRQEN	EORIRQEN	EORJMPEN	EORSTOPEN	Reserved		RASIrqEn	MaskRun	Reserved		SCEMC		SCEMI					BUSERRTYPE			EORINTR	REQPEND	Reserved			RASIntr	STOPINTR	ENDINTR	STARTINTR	BUSERRINTR
Reset	0	?	0	0	0	0	?	?	0	0	?	?	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	0	1	0	0	0

Bits	Access	Name	Description
2	Read/Write 1 to Clear	ENDINTR	**End Interrupt:** The End Interrupt bit indicates that the current descriptor finished successfully and that U2DMACMDx[ENDIRQEN] is set. Write a 1 to clear this bit and reset the corresponding interrupt. Writing a 0 has no effect on the value of this bit or the corresponding interrupt. 0 = No interrupt 1 = Interrupt was caused due to successful completion of the current transaction and U2DMACMDx[LEN] = 0.
1	Read/Write 1 to Clear	STARTINTR	**Start Interrupt:** The Start Interrupt bit indicates that the current descriptor was loaded successfully and that U2DMACMDx[STARTIRQEN] is set. Write a 1 to clear this bit and reset the corresponding interrupt. Writing a 0 has no effect on the value of this bit or the corresponding interrupt. 0 = No interrupt 1 = Interrupt was caused due to successful descriptor fetch.
0	Read/Write 1 to Clear	BUSERRINTR	**Bus Error Interrupt:** The Bus Error Interrupt bit indicates that an error occurred during a data transfer on the System 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.

2.11.26 U2DMA Interrupt Register (U2DMAINT)

U2DMAINT show in Table 78 logs the interrupt information for each channel. An interrupt is generated if any of the following conditions occurs:

- Any transaction error occurs on the system bus associated with the relevant channel. These transaction errors are defined in the U2DMACSRx[BUSERRTYPE] field.
- The current transfer finishes successfully and the U2DMACMDx[ENDIRQEN] bit is set.
- The current Descriptor is loaded successfully and the U2DMACMDx[STARTIRQEN] bit is set.
- U2DMACSRx[STOPERQEN] is set and the channel is in an un-initialized or stopped state.

- U2DMACSRx[EORIRQEN] is set and U2DMACSRx[EORINT] is set (EOR signalled by an endpoint).
- U2DMACSRx[RASIrqEn] is set and the endpoint makes a DMA request after the channel has stopped.

All U2DMA interrupts, except the one that corresponds to U2DMACSRx[STOPINTR], are cleared when software sets the respective error bit in U2DMACSRx[STOPINTR].

This is a read-only register.

Table 78: U2DMAINT Bit Definitions

Physical Address 0x5410_10F0	U2DMAINT	U2DC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																								CHLINTR7	CHLINTR6	CHLINTR5	CHLINTR4	CHLINTR3	CHLINTR2	CHLINTR1	CHLINTR0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	Reserved	**Reserved**
7:0	Read	CHLINTRx	**Channel X Interrupt:** CHLINTRx indicates that DMA channel *x* has been interrupted: 0 = No interrupt 1 = Interrupt

3 Universal Serial Bus Host Controller

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. This chapter describes the PXA30x processor or the PXA31x processor USB host controller. Familiarity with the *Universal Serial Bus Specification*, Revision 1.1[1] and the *Open Host Controller Specification for USB (OHCI)*[2], also known as Open HCI, is assumed.

3.1 PXA3xx Processor Differences

Table 79 shows the clocks and power management differences among the PXA32x, PXA31x and PXA30x processors. Refer to each individual register for other operating differences.

Table 79: PXA3xx Processors Feature Differences

Feature	PXA30x	PXA31x	PXA32x
Host Port 1	Supported	Not Supported	Supported
Host Port 2	6 - wire	3 and 6	6 - wire
Host Port 3 Serial Modes	6 wire Single Ended	3, 4 and 6 wire Single Ended	6 wire Single Ended
VBus Control	Supported	Not Supported	Supported
Root Hub downstream ports	3	2	3
Full Speed Intergrated Transceiver	Supported	Not Supported	Supported

3.2 Features

- Adheres to current USB specification
- Manages both low-speed and full-speed USB devices
- Open Host Controller Interface (OHCI) Rev 1.0a compatible
- Root hub supports up to three downstream ports
- Integrated transceiver option

3.3 Limitations

- Only one overcurrent indicator input; no per-port overcurrent sense support
- V_{bus} enable signal for Port 1 & Port 2 only
- No integrated transceiver option on the PXA31x processor

1. The latest revision of the *Universal Serial Bus Specification Revision 1.1* can be accessed on the World Wide Web Internet site at: http://www.usb.org/
2. *Open Host Controller Interface Specification for USB*, Release 1.0a, loc cit.

- Port 2 connection limited to ULPI transceiver on the PXA31x processor

3.4 Signals

This section describes the signals that are inputs or outputs of the USB host controller (see Figure 1).

Table 80: USB Host Controller Signals

External Name	Type	Description	Notes
USB Full-Speed Transceiver Differential Signals, Host Port 1 and On-the-Go (OTG) Port 2			
USBH1_P	I/O	USB Full-Speed Host Port 1 D+ The positive pin of the differential pair that connects to the USB full-speed host for Port 1.	PXA32x and PXA30x Only
USBH1_N	I/O	USB Full-Speed Host Port 1 D– The negative pin of the differential pair that connects to the USB full-speed host for Port 1.	PXA32x and PXA30x Only
USBHPWR	I	USB Full-Speed Host Port Power Indicator Over-current indicator from the USB host port.	PXA32x and PXA30x Only
USBHPEN	O	USB Full-Speed Host Port 1 Power Control Controls power to USB host Port 1.	PXA32x and PXA30x Only
USBOTG_P	I/O	USB Full Speed Host/Device/OTG Port 2 D+ The positive pin of the differential pair that connects to the USB full-speed host, the USB full-speed device, or the USB OTG interface for Port 2.	PXA32x and PXA30x Only
USBOTG_N	I/O	USB Full-Speed Host/Device/OTG Port 2 D– The negative pin of the differential pair that connects to the USB full-speed host, the USB full-speed device, or the USB OTG interface for Port 2.	PXA32x and PXA30x Only
USBHPEN2	O	USB Full-Speed Host Port 2 Power Control Controls power to the USB Host Port 2. This signal is multiplexed with USB_P2_8 but depends on the value of UP2OCR[SEOS]. Refer to Chapter 1, “Universal Serial Bus Client Controller”.	PXA32x and PXA30x Only
USB Full-Speed Single-Ended Signals, Port 2 Host/Device/OTG			
USB_P2_1	I/O	USB Full-Speed Host/OTG Port 2 RCV/INT/SRP The receive data from an external USB transceiver for Port 2. When configured for an external OTG transceiver, this signal is an interrupt input. When configured for an external OTG power controller and the internal transceiver, this signal is the SRP detect input.	
USB_P2_2	I/O	USB Full-Speed Host/Client and OTG Port 2 OE/Valid Connects to the OE signal of an external USB transceiver for USB Port 2. This signal connects to the session valid output of an external OTG power controller.	

Table 80: USB Host Controller Signals (Continued)

External Name	Type	Description	Notes
USB_P2_3	I/O	USB Full Speed Host/Client and OTG Port 2 RXD-/SV The receive negative data line from an external USB transceiver for Port 2. For an external OTG power controller, this signal is the session valid status input.	
USB_P2_4	I/O	USB Full Speed Host/Client and OTG Port 2 TXD-/SE0_VM/SRP The transmit negative data line to an external USB transceiver for Port 2. For an external OTG transceiver, this signal is the SE0_VM bidirectional data line. For an external OTG power controller this signal is the SRP enable control output.	
USB_P2_5	I/O	USB Full-Speed Host/Client and OTG Port 2 RXD+/DAT_VP/VALID40 The receive positive data line from an external USB transceiver for Port 2. For an external OTG transceiver, this signal is the DAT_VP bidirectional data line. For an external OTG power controller, this signal is the 4.0 V Vbus valid status input.	
USB_P2_6	I/O	USB Full-Speed Host/Client and OTG Port 2 TXD+ The positive transmit data line for an external USB transceiver for Port 2.	
USB_P2_7	I/O	USB Full-Speed Host/Client and OTG Port 2 Speed/OTGID Speed select signal for USB Port 2 when configured for an external USB transceiver. Provides the OTG ID configuration using the internal transceiver.	PXA32x and PXA30x Only
USB_P2_8	I/O	USB Full-Speed Host/Client and OTG Port 2 Suspend Suspend enable for USB Port 2 when configured for an external USB OTG transceiver.	PXA32x and PXA30x Only
USB Full-Speed Single-Ended Signals, Port 3 Host			
USB_P3_1	I/O	USB Full-Speed Host Port 3 RCV Receive data signal that connects to an external transceiver or the transceiver interface of a USB client controller as defined by the CFG bits in the USB Port 3 Output Control Register (UP3OCR) or the U2D Host Port 3 Control Register (U2DP3CR).	1, 2
USB_P3_2	I/O	USB Full-Speed Host Port 3 OE Output enable signal that connects to an external transceiver or the transceiver interface of a USB client controller as defined by the CFG bits in the USB Port 3 Output Control Register (UP3OCR) or the U2D Host Port 3 Control Register (U2DP3CR).	1, 2
USB_P3_3	I/O	USB Full-Speed Host Port 3 RXD Receive data (–) signal that connects to an external transceiver or the transceiver interface of a USB client controller as defined by the CFG bits in the USB Port 3 Output Control Register (UP3OCR) or the U2D Host Port 3 Control Register (U2DP3CR).	1, 2
USB_P3_4	I/O	USB Full-Speed Host Port 3 TXD Transmit data (–) signal that connects to an external transceiver or the transceiver interface of a USB client controller as defined by the CFG bits in the USB Port 3 Output Control Register (UP3OCR) or the U2D Host Port 3 Control Register (U2DP3CR).	1, 2

Table 80: USB Host Controller Signals (Continued)

External Name	Type	Description	Notes
USB_P3_5	I/O	USB Full-Speed Host Port 3 RXD+ Receive data (+) signal that connects to an external transceiver or the transceiver interface of a USB client controller as defined by the CFG bits in the USB Port 3 Output Control Register (UP3OCR) or the U2D Host Port 3 Control Register (U2DP3CR).	1, 2
USB_P3_6	I/O	USB Full-Speed Host Port 3 TXD+ Transmit data (+) signal that connects to an external transceiver or the transceiver interface of a USB client controller as defined by the CFG bits in the USB Port 3 Output Control Register (UP3OCR) or the U2D Host Port 3 Control Register (U2DP3CR).	1, 2

NOTE:
1. The USB Port 3 Output Control Register (UP3OCR) is defined in the USB Device Controller chatper for the PXA32x processor and PXA30x processor.
2. The U2D Host Port 3 Control Register (U2DP3CR) is defined in the USB 2.0 Device Controller chapter for the PXA31x processor.

A USB host must supply 5.0 V as required by the USB specification. However, the processor does not have 5.0-Volt-tolerant pads, so system designers must provide an external device to conntect the USBHPEN and USBHPWR pins to the power supply and over-current detection circuits.

Table 81: USB Host Controller Interface Signals Summary (PXA31x Processor Only)

External Name	Type	Description
USB Single Ended Interface Signals, Host Port 3		
P3_2	O	USB OE
P3_6	O	USB TxD+
P3_4	O	USB TxSE0
P3_5	I	USB RxD+
P3_3	I	USB RxD-
P3_1	I	USB RxD (Differential output from PHY)
USBHPEN	O	USB Power Control Controls power to the external USB host PHY. USBHPEN reflects the status of the UHCRHPSx[PPS] bits and can be used to 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 active low. These signals are multiplexed with GPIO pins. Refer to the GPIO section for details on configuration.

Table 81: USB Host Controller Interface Signals Summary (PXA31x Processor Only) (Continued)

External Name	Type	Description
USBHPWR	I	USB Power Indicator Over-current indicator from the external USB host PHY. The USB host can be programmed to change the polarity of this signal to active low. This signal is multiplexed with GPIO pins. See the GPIO chapter for details on configuring GPIO signals.
USB ULPI 6-pin Serial Mode Interface Signals, OTG Host Port 2		
ULPI_DATA[0]	O	USB OE (Active Low Internal → Active High External)
ULPI_DATA[1]	O	**USB TxD+**
ULPI_DATA[2]	O	**USB TxSE0**
ULPI_DATA[4]	I	**USB RxD+**
ULPI_DATA[5]	I	**USB RxD-**
ULPI_DATA[6]	I	**USB RxD (Differential output from PHY)**

3.5 Operation

Although the PXA32x processor and PXA30x processor provides three USB host ports, the first port is configured by the USB host. The second and third USB host ports must be configured through both the USB host and the USB Device Controller (UDC) registers. The PXA30x supports one full differential port (Port 1) with an integrated transceiver. A second port (Port 2) can be configured in one of the following ways:

- a USB OTG interface (see the UDC chapter for more information)
- a second differential host port with integrated transceiver
- a host port with a digital single-ended interface.

A third port provides a single-ended digital interface for connection to an external Transceiver or to operate in Transceiverless mode.

Only one over-current input is accessible through the GPIO pins. All over-current events are reported globally regardless of which port has an over-current condition. For multiple ports, over-current can be monitored externally and logically combined with the GPIO over-current input to the processor. Alternatively, user-defined GPIO inputs and system software can monitor over-current conditions on host ports 2 and 3. The over-current GPIO does not have to be enabled, but the GPIO must be pulled to the disabled state so that false over-currents are not registered.

The PXA31x provides two USB host ports. Port 2 provides an OTG or a stand-alone host interface. Port 3 provides a digital single-ended interface to either an external transceiver or a means for communicating in Transceiverless mode. The ports must be configured using both the UHC and the USB 2.0 device controller (U2D) Configuration registers.

Also, the PXA31x has only one over-current sense input. Over-current is reported at a global level. If Port 2 and Port 3 require over-current sense detection, the signals can be logically combined externally and brought in through the over-current GPIO.

Host Port 2 over-current conditions can be detected in the the external ULPI PHY and communicated as a Serial mode interrupt to the U2D interrupt registers. Alternatively, user-defined

GPIO inputs and system software can be utilized to monitor over-current conditions on Host Ports 2 and 3.??????????????

The UHC consists of an OHCI core, a bus interface unit to connect to the system bus, two small FIFOs for buffering data in and out, two input pins, two 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 via DMA. The registers must be accessed as 32-bit entities on 32-bit aligned addresses.

Figure 36: USB Host Controller Block Diagram (PXA32x and PXA30x Processors Only)

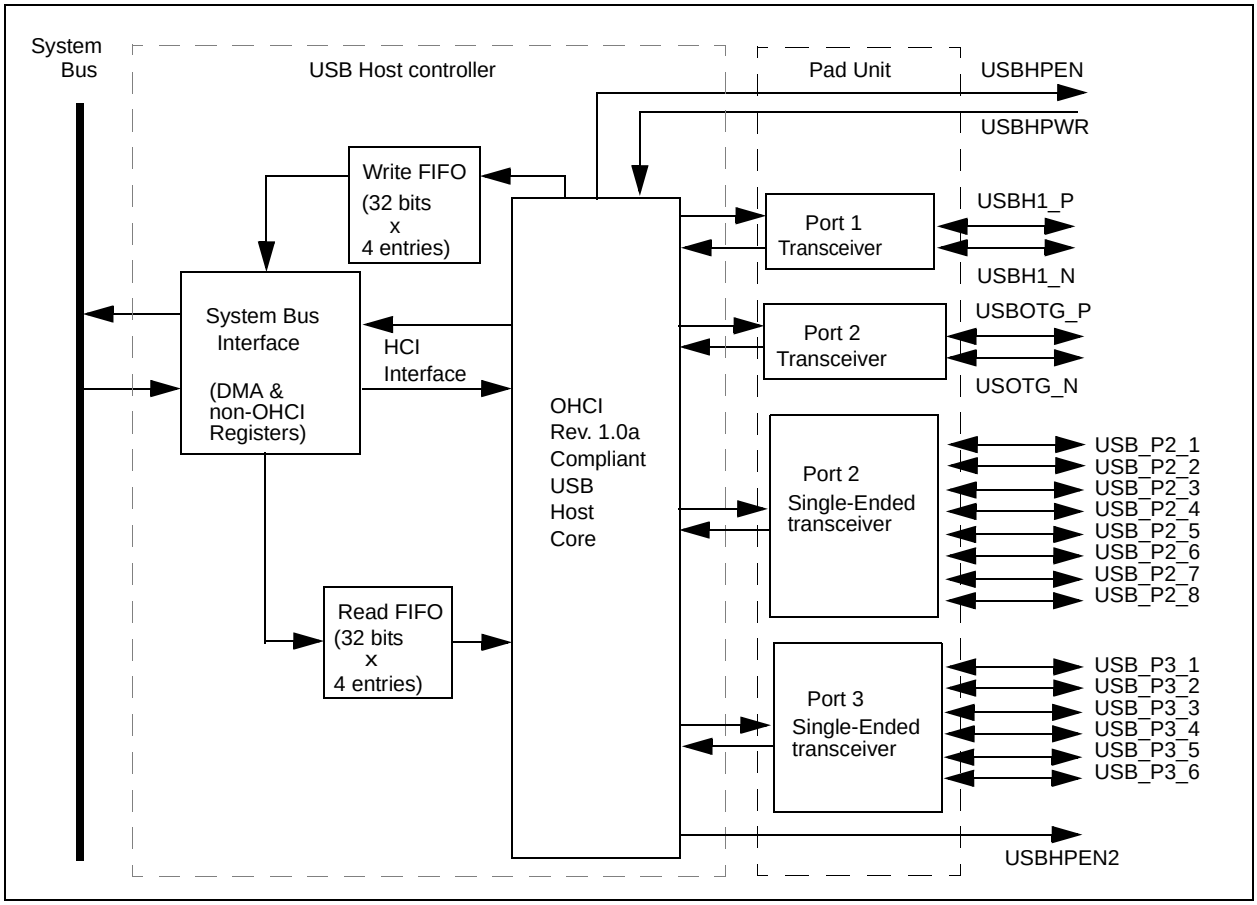
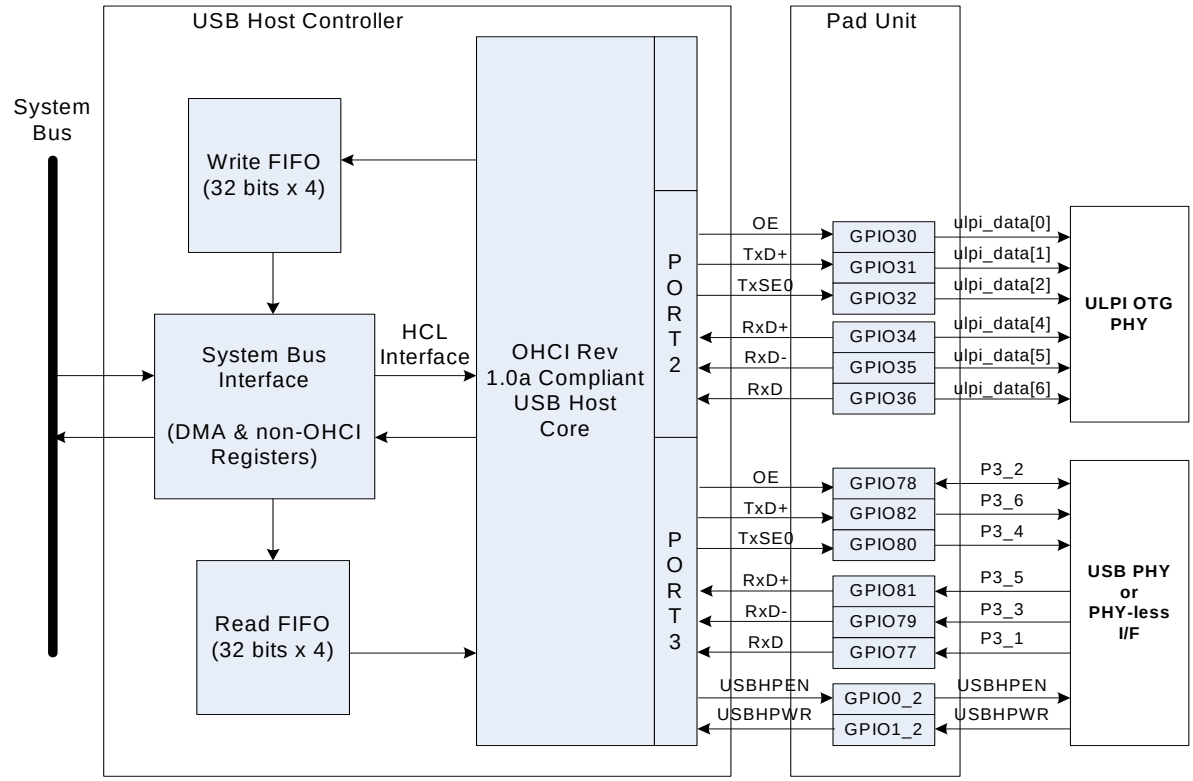


Figure 37: USB Host Controller Block Diagram (PXA31x only)



The USB host controller functions as a DMA operating on linked lists, called Endpoint Descriptors & Transfer Descriptors (EDs and TDs), created by the Host Controller Driver (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 and then passes control to the UHC for processing. 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.

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 (UHCHCON[BLE], UHCHCON[CLE], UHCHCON[IE], UHCHCON[PLE]), and interrupt enables (UHCINTE).

Note

In Isochronous Transfer mode, the latency associated with using VLIO and PCMCIA 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.

1. See the OHCI Rev.1.0a specification, page 9, section 3.2.2 for details.

3.6 USB Host Port Serial Configurations

3.6.1 USB Host Port 2 Serial Selection

The USB host (UHC) Port 2 serial selection (P2SS) bits in the U2D Host Port 3 Control Register (U2DP3CR) are used to select between 3-pin or 6-pin ULPI modes in the USB Mux. Internally the UHC Port 2 has a 6-pin single-ended interface and this field allows the software to convert the internal UHC Port 2 interface to one that matches the external off-chip ULPI PHY.

Note

The TXEN_n of the UHC Port 2 is inverted to the ULPI PHY so that all zeros is the idle condition on the ULPI interface.

3.6.2 USB Host Port 3 Serial Selection

The USB host (UHC) Port 3 serial selection (P3SS) bits in the U2D Host Port 3 Control Register (U2DP3CR) are used to select between the CEA-2011 3-pin and 4-pin modes as well as the legacy 6-pin mode at the GPIOs of the UHC Port 3 interface. Internally the UHC Port 3 is a 6-pin single-ended interface and this field allows the software to convert the internal interface to the 3-pin, 4-pin, and 6-pin modes.

Note

Refer to the CEA-2011 specification for more details on these 3-pin and 4-pin interfaces.

The U2DP3CR[P3SS] field and the UHC Port 3 configuration U2DP3CR[CFG] field can be combined together to select between host, client, or passthru modes and 3, 4, or 6-pin modes of the external connection on the GPIO pads for the UHC Port 3 interface

The following table describes each of the supported serial modes available at the GPIOs for the UHC Port 3 interface:

Pin	Type	Description
6-wire single ended		
OE_n/INT_n	I/O	Transmit Enable (active low) or Interrupt (active low) when suspended
DAT_VP	output	Transmit Data
SE0_VM	output	Transmit Single Ended 0 (SE0)
DP	input	Receive D+
DM	input	Receive D-
RCV	input	Differential receiver output from PHY
6-wire differential		
OE_n/INT_n	I/O	Transmit Enable (active low) or Interrupt (active low) when suspended
DAT_VP	output	Transmit D+

Pin	Type	Description
SE0_VM	output	Transmit D-
DP	input	Receive D+
DM	input	Receive D-
RCV	input	Differential receiver output from PHY
3-wire CEA-2011		
OE_n/INT_n	I/O	Transmit Enable (active low) or Interrupt (active low) when suspended
DAT_VP	I/O	Transmit Differntial Data when TXEN_n is low Receive Differential Data when TXEN_n is high
SE0_VM	I/O	Transmit Single Ended Zero (SE0) when TXEN_n is low Receive Single Ended Zero (SE0) when TXEN_n is high
4-wire CEA-2011		
OE_n/INT_n	I/O	Transmit enable (active low) or Interrupt (active low) when suspended
VP_DP	I/O	Transmit D+ when TXEN_n is low Receive D+ when TXEN_n is high
VM_DM	I/O	Transmit D- when TXEN_n is low Receive D- when TXEN_n is high
RCV	input	Differential receiver output from the PHY

3.7 USB Host Port 3 Configuration

The USB host Port 3 configuration (CFG) bits in the U2D Host Port 3 Control Register (U2DP3CR) or USB Port 3 Output Control Register (UP3OCR) are used to set the operating configuration for Port 3 of the USB Host controller. This port can be used to connect to an external device and/or transceiver in several different configurations. The USB Host controller Port 3 inputs can be used to provide a wakeup when resume signalling is detected. USB Host controller Port 3 inputs can not be used to provide a wakeup when a connect/disconnect is detected. For information on programming wakeup events, refer to the Clocks Controller and Power Management Chapter in PXA3xx Processor Family Vol. I: System and Timer Configuration Developers Manual.

Table 82 shows the possible Port 3 configurations and register settings for these configurations.

Table 82: USB Host Controller Port 3 Configuration Selection Values

GPIO Alternate Function Port	Port 3 Configuration - UP3OCR[CFG] / U2DP3CR[CFG]		
	0x0	0x1	0x2
USB_P3_1	UHC Rx Data (RCV - in)	USB_P2_1	UHC Rx Data (RCV - out)
USB_P3_2	UHC OE (OE_n - out)	USB_P2_2	UHC OE (OE_n - in)

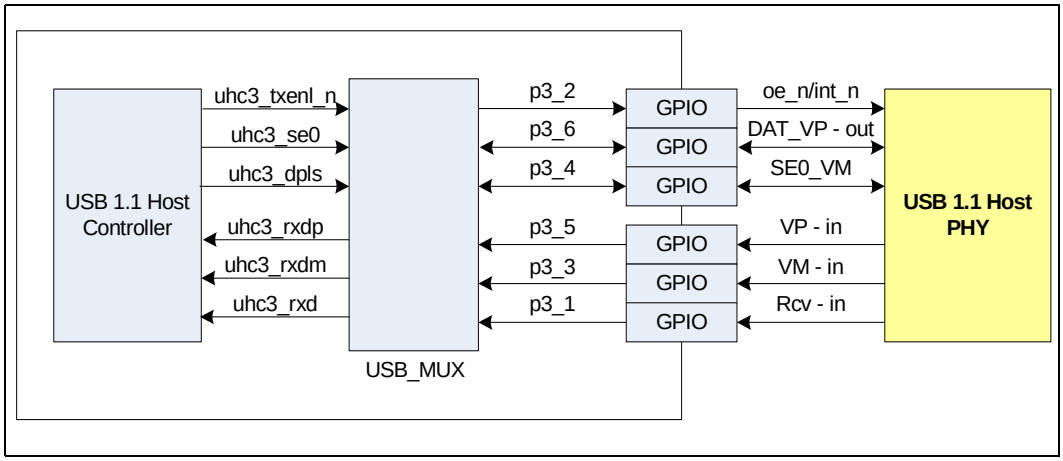
Table 82: USB Host Controller Port 3 Configuration Selection Values

GPIO Alternate Function Port	Port 3 Configuration - UP3OCR[CFG] / U2DP3CR[CFG]		
	0x0	0x1	0x2
USB_P3_3	UHC Rx D– (VM - in)	USB_P2_3	UHC Rx D- (VM - out)
USB_P3_4	UHC Tx D– (VMO - out)	USB_P2_4	UHC Tx D- (VMO - in)
USB_P3_5	UHC Rx D+ (VP - in)	USB_P2_5	UHC Rx D+ (VP - out)
USB_P3_6	UHC Tx D+ (VPO - out)	USB_P2_6	UHC Tx D+ (VPO - in)

3.7.1 Transciever Host Mode

When U2DP3CR[CFG] is cleared 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. See Figure 4 below.

Figure 38: Host Mode Block Diagram

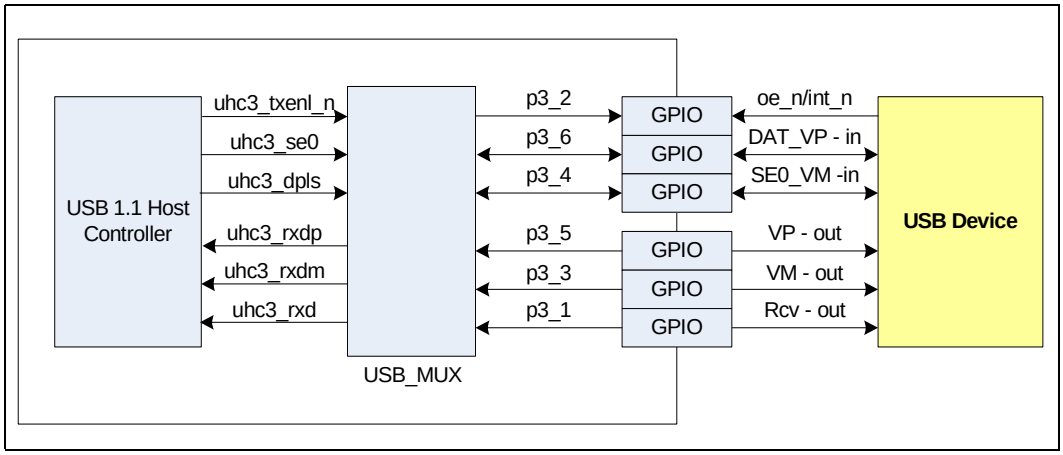


3.7.2 Transceiver-less Host Mode

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. See Figure 5 below.

An over-current detect input and a power control signal are also provided to enable control of a standard USB host transceiver. These are always provided on USBHPEN and USBHPWR regardless of other functions.

Figure 39: Transceiver-less Block Diagram



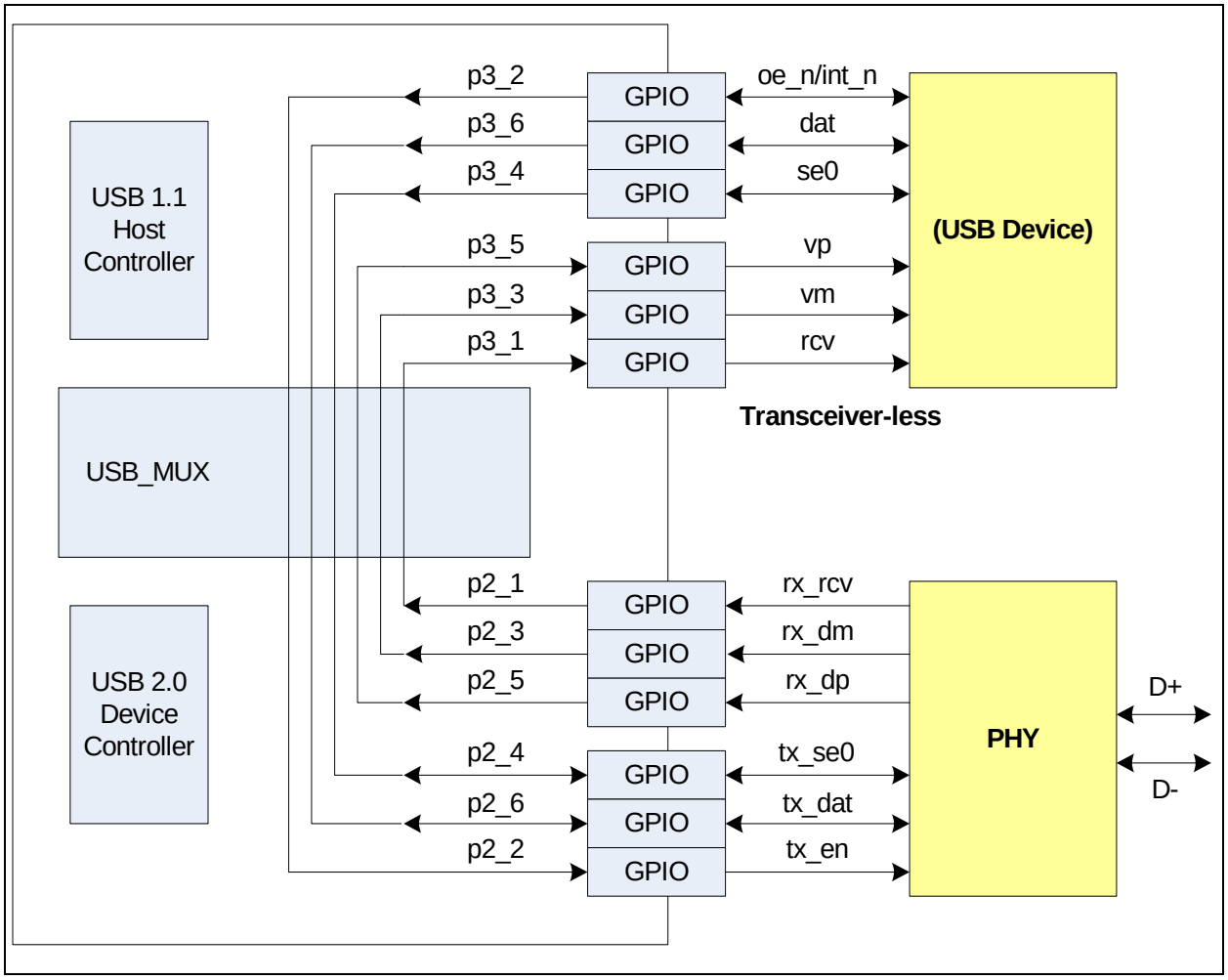
Note

The P2SS and P3SS fields in this register can be used to configure the size of the interface to the external chips (3-pin, 4-pin, or 6-pin).

3.7.3 Pass Through Mode

When CFG is set to 0x1, the USB Host controller Port 3 is configured to connect an external USB device controller to the USB Host controller Port 2 single-ended outputs. In this configuration, the external USB device controller USB signals are output directly to the processor USB Host controller Port 2 single-ended outputs with no control from the processor USB modules. For this mode to operate correctly, UP2OCR[HXOE] must be cleared to 0.See Figure 40 below.

Figure 40: Pass Through Mode Block Diagram



.

3.7.4 Interrupts

The USB host controller generates two interrupts to the interrupt controller (see the *Interrupt Controller* chapter in Vol. I of this document for details).

- An OHCI USB interrupt (generated from the Interrupt Status register; see Section 3.8.5)
- All other USB host controller event interrupts (see Section 3.8.24)
  - Buffer-access interrupt
  - Remote wakeup interrupt
  - Bus Master Abort interrupt

- Bus Target Abort interrupt
- Port-resume signal interrupt
- OHCI-initiated interface clear signal interrupt (transfer abort)
- USB port power over-current exception interrupts

3.7.5 Programming Considerations

Note

PXA31x Only: The USB Port 2 is only used as an OTG Host port connected to an external ULPI PHY. Before performing the following steps, the GPIOs and ULPI PHY need to be already configured and set up for host mode operation using the U2D configuration registers. See the U2D chapter for more information about how this is done.

Recommended software flow to enable a port is as follows:

1. Verify that ASCR[RDH] = 0 (refer to the Slave Power Management Unit for details)
2. Set the USBHPEN & USBHPWR GPIOs as necessary
3. Enable the appropriate ports by clearing the UHCHR[SSEx] bit
4. Clear UHCHR[SSE] bit
5. Set the PSPL & PCPL bits to the preferred polarity
6. Set UHCHIR[UPSx] over-current sense, and any other necessary non-OHCI interrupts
7. Set UHCHR[FHR], wait > 10us, then clear FHR to perform a hard reset on the UHC
8. Set the base address for the HC Communications Area (HCCA) via UHCHCCA
9. Place the UHC in the USB Operational mode via UHCHCON[HCFS]
10. Clear UHCRHDA[NOCP] to enable over-current protection
11. Verify UHCRHDA[OCPM] is cleared since the UHC only supports global over-current reporting
12. Clear UHCRHDA[NPS] to enable port power switching
13. Set UHCRHDA[PSM] for per-port power control
14. Set appropriate UHCRHPSx[PPS] bit to enable port power (V_{bus})
15. Wait for Device Connected detection - read until UHCRHPS2[CSC] is set
16. Enable Port 2 - set UHCRHS[PES]
17. Wait for Port 2 Enable Status - read until UHCRHPS2[PESC] is set
18. Optionally, poll CCS bit until a connection is established and then proceed per specification. For example, when CCS goes high, a port reset could be done, at which point UHCHPSx[PES] would get set at the end of the USB Reset; SOFs would automatically start being transmitted afterward

3.7.6 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. 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

3.7.6.1 USB Suspend

Note

Refer to the Clocks Controller and Power Management chapter in *PXA3xx Processor Family Vol. I: System and Timer Configuration Developers Manual* for system level Power mode details. The ASCR[RDH] bit in the Application Subsystem Power Management Unit (APMU) should be cleared after exiting any system reset and S2/D3/C4 and prior to enumeration. Clearing this bit too early or too late may cause undesired values on the GPIO pins.

The USB clocks can be stopped at any time. However, stopping the clocks is recommended only when the USB is in the USB Suspend state, which is reached when there has been no activity on the bus for more than 3 ms. In the suspend state, the ports' transmit circuits are turned off while the receivers are left on in order to detect Resume signaling. The asynchronous interrupt is then generated to wake up the PMU controller (refer to the power management chapters for details). The USB clock should be turned off to enable the wakeup event to occur; otherwise, a false wakeup event could be triggered. Note that an over-current condition on a port can also cause an interrupt to be sent asynchronously to the wakeup controller of the PMU unit (see the power management chapters) to restart clocks.

Full-speed transceivers have a Suspend pin input that enables the controller to place the PHY into a low-power mode. For standlone host applications, a GPIO could be used to control the pin on the transceiver and enable the low-power mode.

Note

PXA31x Only: For the UHC Port 2, the external ULPI PHY must be put into low-power mode manually, by bringing the ULPI PHY out of serial mode and putting it into low-power mode. Any wakeup conditions will be indicated via the U2D interrupt register while the PHY is in low-power mode (see the U2D chapter for more details about how to do this) or through GPIO wakeups if the U2D is powered off. The external PHY can also be put into low-power mode whenever a device is not connected via the U2D control registers (see the U2D chapter for more details). Any mode change requires U2D intervention in order to switch the state of the PHY; the UHC has no direct control over the ULPI PHY.

3.7.6.2 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 and the remote wakeup interrupt is enabled via UHCHIE[RWIE], then a port-resume event asynchronously sends an interrupt to the wakeup controller of the PMU 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.

Three events can cause a USB Port Resume interrupt:

- A device is connected (indicated by one of the USB port signals being pulled high)

- A device 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)

Note

PXA31x Only: For the UHC Port 2, the Port Resume detection must be handled by the software interrupt routine for ULPI interrupts in the U2D. This includes reading the ULPI PHY registers to determine the interrupt cause (see the U2D chapter for more details).

3.7.6.3 Suggested Power-Management Routines

The following power management routines are given as examples on how to map certain USB events to platform low-power modes. The following items apply to all of the suggested power-management routines in this section:

- The USB Port Resume interrupt is not generated when a device is connected unless UHCHR[SSE], the Sleep Standby Enable bit, is cleared.
- First start the USB clock before setting the Global Power Enable bit UHCRHS[LPSC].
- To enable the appropriate UHC wakeup sources in the respective Power mode, write BPMU Application Subsystem wakeup from Dx to D0 State Enable Registers for D1, D2, or D3 (AD1D0ER, AD2D0ER, or AD3ER). The UHC has wakeup sources on WEUSBH, WEDMUX3, WEDMUX2, WEOTG bits in the AD1D0ER, AD2D0ER, and AD3ER registers.

Note

To prevent false USB Host wakeups from occurring the Sleep Standby Enable bits for each port (UHCHR[SSEP1], UHCHR[SSEP2], UHCHR[SSEP3]) must be cleared and the main Sleep Standby Enable bit (UHCHR[SSE]) must be also be cleared before going into a Low-power mode.

Note

The USB Port Resume 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.

One of these routines should be followed before any low-power state is entered.

3.7.6.3.1 Low-Power Mode in Suspend State Sequence

Perform the following in sequence to put the USB system into Low-power mode:

1. Wait for the USB to enter either a global suspend state or a port suspend.
2. Set the appropriate Port Suspend bit (UHCRHPS1[PSS] or UHCRHPS2[PSS] or UHCRHPS3[PSS]). For a global USB Suspend, use UHCHCON[HCFS] to place the controller in a global USB Suspend state. This can be automatic after 3 ms of inactivity on the USB.
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. Verify the Sleep Standby Enable bit (UHCHR[SSE]) is cleared, enabling power to the USB single-ended receivers and the USB port power supplies.

5. Enter low power modeD3 by writing to the Intel XScale® core PWRMODE Register (CP14 Register 7).

Note

Software waits for the USB Port Resume interrupt to go active. Start the USB clock, and the host controller restarts.

Note

PXA31x Only: For UHC Port 2, the ULPI PHY should be put in Low-power mode after step 3 in the above sequence (see the U2D chapter for details).

3.7.6.3.2 Low-Power Mode after Device Disconnect Sequence

Follow this sequence to activate Low-power mode:

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. Verify the Sleep Standby Enable bit (UHCHR[SSE]) is cleared to enable power to the downstream ports.
3. Enter Low-power mode D3 by writing to the Intel XScale® core PWRMODE Register (CP14 Register 7).

Note

The port is now powered down. 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.

Note

PXA31x Only: For UHC Port 2, the ULPI PHY should be put in Low-power mode after step 1 in the above sequence (see the U2D chapters for details).

3.8 Register Descriptions

The USB host controller block contains control and status registers in addition to registers required by the OHCI specification (0x4C00_0000 – 0x4C00_006C).

3.8.1 Register Summary

Table 83 lists these registers and their memory-mapped locations. All registers must be accessed as 32-bit entities. The DMA controller should not be used to read these registers because read errors occur during an odd (non-8-byte-aligned address, that is, address[2]= 1) read.

Table 83: USB Host controller Register Addresses

Physical Address	Name	Description	Page
0x4C00_0000	UHCREV	UHC HCI Specification Revision (UHCREV)	3-242
0x4C00_0004	UHCHCON	UHC Host Control Register (UHCHCON)	3-242
0x4C00_0008	UHCCOMS	UHC Command Status (UHCCOMS)	3-246
0x4C00_000C	UHCINTS	UHC Interrupt Status (UHCINTS)	3-248
0x4C00_0010	UHCINTE	UHC Interrupt Enable (UHCINTE)	3-250
0x4C00_0014	UHCINTD	UHC Interrupt Disable (UHCINTD)	3-251
0x4C00_0018	UHCHCCA	UHC Host Controller Communication Area (UHCHCCA)	3-252
0x4C00_001C	UHCPCED	UHC Period Current Endpoint Descriptor (UHCPCED)	3-253
0x4C00_0020	UHCCHED	UHC Control Head Endpoint Descriptor (UHCCHED)	3-253
0x4C00_0024	UHCCCED	UHC Control Current Endpoint Descriptor (UHCCCED)	3-254
0x4C00_0028	UHCBHED	UHC Bulk Head Endpoint Descriptor (UHCBHED)	3-254
0x4C00_002C	UHCBCED	UHC Bulk Current Endpoint Descriptor (UHCBCED)	3-255
0x4C00_0030	UHCDHEAD	UHC Done Head (UHCDHEAD)	3-255
0x4C00_0034	UHCFMI	UHC Frame Interval (UHCFMI)	3-256
0x4C00_0038	UHCFMR	UHC Frame Remaining (UHCFMR)	3-257
0x4C00_003C	UHCFMN	UHC Frame Number (UHCFMN)	3-258
0x4C00_0040	UHCPERS	UHC Periodic Start (UHCPERS)	3-259
0x4C00_0044	UHCLST	UHC Low-Speed Threshold (UHCLST)	3-260
0x4C00_0048	UHCRHDA	UHC Root Hub Descriptor A (UHCRHDA)	3-260
0x4C00_004C	UHCRHDB	UHC Root Hub Descriptor B (UHCRHDB)	3-262
0x4C00_0050	UHCRHS	UHC Root Hub Status (UHCRHS)	3-263
0x4C00_0054	UHCRHPS1	UHC Root Hub Port Status 1 (UHCRHPS1)	3-265
0x4C00_0058	UHCRHPS2	UHC Root Hub Port Status 2 (UHCRHPS2)	3-265
0x4C00_005C	UHCRHPS3	UHC Root Hub Port Status 3 (UHCRHPS3)	3-265
0x4C00_0060	UHCSTAT	UHC Status Register (UHCSTAT)	3-269
0x4C00_0064	UHCHR	UHC Reset Register (UHCHR)	3-273

Table 83: USB Host controller Register Addresses (Continued)

Physical Address	Name	Description	Page
0x4C00_0068	UHCHIE	UHC Interrupt Enable Register (UHCHIE)	3-276
0x4C00_006C	UHCHIT	UHC Interrupt Test Register (UHCHIT)	3-278
0x4C00_0070–0x4FFF_FFFF	—	Reserved	

3.8.2 UHC HCI Specification Revision (UHCREV)

UHCREV, shown in Table 84, contains the binary-coded decimal (BCD) representation of the version of the OHCI specification for the processor. The UHC complies with the *OHCI Specification*, Rev. 1.0a so this register always has a value of 0x0000_0010.

Table 84: UHCREV Bit Definitions

Physical Address 0x4C00_0000 **UHCREV** **UHC**

	31	30	29	28	27	26	25	24	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
	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	Read	Rev	Revision Indicates the revision number of the OHCI specification. This USB host is compliant with OHCI Rev. 1.0a, so this register reads 0x0000_0010.

3.8.3 UHC Host Control Register (UHCHCON)

UHCHCON, shown in Table 85, defines the operating modes of the host controller. Most fields in this register are modified only by the host controller driver, except for the host controller functional state and remote wakeup connected fields. All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 85: UHCHCON Bit Definitions

Physical Address 0x4C00_0004 **UHCHCON** **UHC**

	31	30	29	28	27	26	25	24	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																					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	Reserved

Table 85: UHCHCON Bit Definitions (Continued)

Physical Address 0x4C00_0004 **UHCHCON** **UHC**

User Settings																														
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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																					**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
10	Read/Write	RWE	Remote Wakeup Enable The host controller driver uses this bit to enable or disable the remote wakeup feature when upstream resume signaling is detected. When this bit is set and the resume detected UHCINTS[RD] bit is set, a remote wakeup is signaled to the host system. Setting this bit has no affect on the generation of a hardware interrupt. 0 = Disable remote wakeup signalling to the host system. 0 = Enable remote wakeup signalling to the host system.
9	Read/Write	RWC	Remote Wakeup Connected Indicates whether the UHC supports remote wakeup signaling. If it is, system firmware must set this bit after reset. The UHC clears the bit upon a hardware reset, but it does not alter it upon a software reset. 0 = No wakeup signaling. 1 = UHC supports remote wakeup signaling.
8	Read/Write	IR	Interrupt Routing Determines the routing of interrupts generated by events registered in the UHC Interrupt Status register (UHCINTS). If IR is 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 IR as a tag to indicate the ownership of the UHC. This implementation of the OHCI host controller does not support SMI, so the host controller driver should never set this bit. 0 = All interrupts are routed to the normal host bus interrupt mechanism. 1 = Reserved.

Table 85: UHCHCON Bit Definitions (Continued)

Physical Address 0x4C00_0004	UHCHCON	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																					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	Read/Write	HCFS	Host Controller Functional State Displays the current functional state of the USB host controller. A transition to USBOPERATIONAL from another state causes an SOF generation to begin 1 ms later. The host controller driver can determine whether UHC is sending SOFs by reading the start-of-frame field, UHCINTS[SF]. The UHC and change this field only in the USBSUSPEND state. The 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. Software must be careful to avoid an illegal state transition because the UHC does not monitor the validity of the transition. For example, if the UHC is in USBOPERATIONAL, software can mistakenly transition to USBRESUME without the UHC detecting this error. 0b00 = USBRESET 0b01 = USBRESUME 0b10 = USBOPERATIONAL 0b11= USBSUSPEND
5	Read/Write	BLE	Bulk List Enable The host controller driver sets this bit to enable processing of the bulk list in the next frame. UHC checks this bit when it needs to process the list. When BLE is cleared, the host controller driver can modify the list. If Bulk Current ED points to an ED to be removed, the host controller driver must advance the pointer by updating Bulk Current ED before re-enabling list processing. 0 = Disables processing of the bulk list after the next SOF. 1 = Enables processing of the bulk list in the next frame.

Table 85: UHCHCON Bit Definitions (Continued)

Physical Address 0x4C00_0004 **UHCHCON** **UHC**

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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																					**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
4	Read/Write	CLE	Control List Enable: 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 Control Current ED is pointing to an ED to be removed, the host controller driver must advance the pointer by updating Control Current ED 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.
3	Read/Write	IE	Isochronous Enable: 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	Read/Write	PLE	Periodic List Enable: 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.

Table 85: UHCHCON Bit Definitions (Continued)

Physical Address 0x4C00_0004	UHCHCON	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																					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
1:0	Read/Write	CBSR	Control Bulk Service Ratio: The Control Bulk Service Ratio specifies the service ratio between control and bulk endpoint descriptors (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

3.8.4 UHC Command Status (UHCCOMS)

The USB host controller uses the UHC Command Status (UHCCOMS, shown in Table 86) register to receive commands that the host controller driver issues, as well as reflect 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.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 86: UHCCOMS Bit Definitions

Physical Address 0x4C00_0008	UHCCOMS	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	Reserved

Table 86: UHCCOMS Bit Definitions (Continued)

Physical Address 0x4C00_0008 **UHCCOMS** **UHC**

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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
17:16	Read	SOC	Scheduling Overrun Count: The Scheduling Overrun Count 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 UHCINTS[SO] (Scheduling Overrun) field in the UHC Interrupt Status register. 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 UHCINTS[SO] (Scheduling Overrun) 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	Reserved
3	Read	OCR	Ownership Change Request: This bit always returns 0 when read. This implementation of the OHCI host does not support SMI, therefore, software should never write a 1 to this bit. 0 = No HCD request for a change in control of the UHC is pending 1 = Reserved
2	Read/Write	BLF	Bulk List Filled: This bit indicates whether there are any transfer descriptors (TD) 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 this bit. As long as this bit is 0, UHC does not start processing the bulk list. If this bit is 1, UHC starts processing the bulk list and then clears this bit. If UHC finds a TD on the list, then UHC sets this bit, 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 this bit, then this bit is still 0 when UHC completes processing the bulk list and bulk list processing stops. 0 = Bulk list processing stops 1 = Enables processing of Bulk EDs.

Table 86: UHCCOMS Bit Definitions (Continued)

Physical Address 0x4C00_0008	UHCCOMS	UHC

Bit	31	30	29	28	27	26	25	24	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
1	Read/Write	CLF	Control List Filled: 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 this (CLF) bit. As long as CLF is 0, UHC does not start processing the control list. If CLF is set, UHC starts processing the control list and then clears CLF. If UHC finds a TD on the list, then UHC sets CLF 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 still is 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 clears ControlListFilled
0	Read/Write	HCR	Host Controller Reset: 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 UHCHCON[IR] field). 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 µsec.). 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

3.8.5 UHC Interrupt Status (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 UHC Interrupt Enable register (see Section 3.8.6, “UHC Interrupt Enable (UHCINTE)”) and the MasterInterruptEnable (UHCINTE[MIE]) bit is set. The host controller driver can clear specific bits in this register by setting the desired 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 this delay into consideration.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 87: UHCINTS Bit Definitions

Physical Address 0x4C00_000C **UHCINTS** **UHC**

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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																									**RHSC**	**FNO**	**UE**	**RD**	**SF**	**WDH**	**SO**
Reset	**?**	**0**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Bits	Access	Name	Description
31:7	—	Reserved	Reserved
6	Read/Write 1 to clear	RHSC	Root Hub Status Change: This bit is set when the content of UHCRHS or the content of any of UHCRHPSx registers has changed. 0 = The contents of these registers has not changed 1 = The contents of UHCRHS or UHCRHPSx has changed
5	Read/Write 1 to clear	FNO	Frame Number Overflow: This bit is set when the most significant bit of UHCFMN (bit 15) changes value, from 0b0 to 0b1 or from 0b1 to 0b0, and after HccaFrameNumber has been updated. 0 = Otherwise 1 = UHCFMN (bit 15) changed in value, from 0b0 to 0b1 or from 0b1 to 0b0, or HccaFrameNumber has been updated
4	Read/Write 1 to clear	UE	Unrecoverable Error: This bit is set when UHC detects a system error not related to USB. UHC should not proceed with any processing nor 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	Read/Write 1 to clear	RD	Resume Detected: This bit is set when UHC detects that a device on the USB is asserting resume signaling. It is the transition from bus idle to K-state signaling that sets this bit. This bit is not set when HCD sets the USB Resume state. 0 = No device is asserting resume signaling 1 = A device on the USB is asserting resume signaling
2	Read/Write 1 to clear	SF	Start of Frame: This bit is set by UHC at each start of a frame and after the update of Hcca FrameNumber. UHC also generates a SOF token at the same time. 1 = A start of a frame or an update of HccaFrameNumber has occurred

Table 87: UHCINTS Bit Definitions (Continued)

Physical Address 0x4C00_000C	UHCINTS	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																									RHSC	FNO	UE	RD	SF	WDH	SO
Reset	?	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
1	Read/Write 1 to clear	WDH	Writeback Done Head: This bit is set immediately after the UHC has written the value of the UHCDHEAD to an external memory location referred to (by the OHCI specification) as HccaDoneHead. Additional updates of the HccaDoneHead does not occur until this bit has been cleared. HCD should clear this bit only after it has saved the content of HccaDoneHead. 1 = UHC has written UHCDHEAD to HccaDoneHead
0	Read/Write 1 to clear	SO	Scheduling Overrun: 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 Scheduling Overrun Count of UHCCOMS to be incremented In order to avoid this condition, care should be taken to enable the periodic lists as early in the frame as possible. Processing of the periodic lists occurs with no TD's 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 to1 and the SOC counter is incremented. 1 = The USB schedule for the current Frame has overrun or the update of HccaFrameNumber has occurred

3.8.6 UHC Interrupt Enable (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. If the UHCINTE[MIE], the MasterInterruptEnable bit is set, then the OCHI interrupt request is sent to the processor interrupt controller.

Setting a bit in this register sets the corresponding bit, whereas clearing 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, set the corresponding bit in the Interrupt Disable register (UHCINTD), see Section 3.8.7, "UHC Interrupt Disable (UHCINTD)".

The register organization and individual bit definitions are shown in Table 88.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 88: UHCINTE Bit Definitions

Physical Address 0x4C00_0010	UHCINTE	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	Read/Write	MIE	Master Interrupt Enable: 0 = Ignored by UHC. 1 = Enables 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	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
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 UHCDHEAD: 0 = Ignore 1 = Enable interrupt generation due to UHCDHEAD Writeback
0	Read/Write 1 to set	SO	Scheduling Overrun: 0 = Ignore 1 = Enable interrupt generation due to scheduling overrun

3.8.7 UHC Interrupt Disable (UHCINTD)

Each disable bit in the UHC Interrupt Disable (UHCINTD, shown in Table 89 on page 252) 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, setting a bit in this register clears the corresponding bit in the UHC Interrupt Enable register, whereas clearing 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 89: UHCINTD Bit Definitions

Physical Address 0x4C00_0014	UHCINTD	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	Read/Write	MIE	Master Interrupt Enable: This field is set after a hardware or software reset. 0 = Ignored by UHC 1 = Disables interrupt generation due to events specified in the other bits of this register.
30	Read/Write	OC	Ownership Change: 0 = Ignore 1 = Disable interrupt generation due to ownership change
29:7	—	Reserved	Reserved
6	Read/Write	RHSC	Root Hub Status Change: 0 = Ignore 1 = Disable interrupt generation due to root hub status change
5	Read/Write	FNO	Frame Number Overflow: 0 = Ignore 1 = Disable interrupt generation due to frame number overflow
4	Read/Write	UE	Unrecoverable Error: 0 = Ignore 1 = Disable interrupt generation due to unrecoverable error.
3	Read/Write	RD	Resume Detect: 0 = Ignore 1 = Disable interrupt generation due to resume detect
2	Read/Write	SF	Start of Frame: 0 = Ignore 1 = Disable interrupt generation due to start of frame.
1	Read/Write	WDH	Writeback UHCDHEAD: 0 = Ignore 1 = Disable interrupt generation due to UHCDHEAD Writeback
0	Read/Write	SO	Scheduling Overrun: 0 = Ignore 1 = Disable interrupt generation due to scheduling overrun.

3.8.8 UHC Host Controller Communication Area (UHCHCCA)

The UHCHCCA register contains the exact physical address of the host controller communication area.

Table 90: UHCHCCA Bit Definitions

Physical Address 0x4C00_0018	UHCHCCA	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																								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
31:8	Read/Write	HCCA	**Host Controller Communication Area:** This is the base address of the host controller communication area. This register is a pointer to an area that is used to hold 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 returns 0 when read. Detailed descriptions are located in Chapter 4 of the *OHCI Rev. 1.0a Spec*.
7:0	-	Reserved	Reserved

3.8.9 UHC Period Current Endpoint Descriptor (UHCPCED)

The UHCPCED register contains the exact physical address of the current isochronous or interrupt endpoint descriptor (ED). The register organization and individual bit definitions are shown in Table 91. The lower 4 bits are read as 0 and are not affected by writes.

Table 91: UHCPCED Bit Definitions

Physical Address 0x4C00_001C	UHCPCED	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																												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
31:4	Read	PCED	**Period Current Endpoint Descriptor:** This register is used by the UHC to point to the head of one of the periodic lists that are 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.
3:0	-	Reserved	Reserved

3.8.10 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 92. The lower 4 bits are read as 0 and are not affected by writes.

Table 92: UHCCHED Bit Definitions

Physical Address 0x4C00_0020	UHCCHED	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																												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
31:4	Read/Write	CHED	Control Head Endpoint Descriptor: The UHC traverses the control using the pointer to the first endpoint descriptor; the content is loaded from the HCCA during the initialization of the UHC.
3:0	—	Reserved	Reserved

3.8.11 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 93. All reserved bits are read as 0 and are not affected by writes.

Table 93: UHCCCED Bit Definitions

Physical Address 0x4C00_0024	UHCCCED	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																												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
31:4	Read/Write	CCED	**Control Current 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 UHCCOMS[CLF] (ControlListFilled). If set, it copies the content of the UHCCHED register to 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 UHCHCON[CLE] (ControlListEnable) 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 bits 0 through 3 are always 0.
3:0	—	Reserved	Reserved

3.8.12 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 94. The lower 4 bits are read as 0 and are not affected by writes.

Table 94: UHCBHED Bit Definitions

Physical Address 0x4C00_0028 **Table 95: UHCBHED** **UHC**

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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																												**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
31:4	Read/Write	BHED	Bulk Head Endpoint Descriptor: The UHC traverses the bulk list starting with the pointer contained in this register; the content is loaded from the HCCA during the initialization of the UHC.
3:0	—	Reserved	Reserved

3.8.13 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 96. The lower 4 bits are read as 0 and are not affected by writes.

Table 96: UHCBCED Bit Definitions

Physical Address 0x4C00_002C **UHCBCED** **UHC**

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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																												**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
31:4	Read/Write	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 UHCHCON[CLF] bit. If set, the UHC copies the content of the UHCBHED register to this register 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 UHCHCON[BLE] bit 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.
3:0	—	Reserved	Reserved

† Bitmap and bit definitions

3.8.14 UHC Done Head (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 97. The lower 4 bits are read as 0 and are not affected by writes.

Table 97: UHCDHEAD Bit Definitions

Physical Address 0x4C00_0030 **UHCDHEAD** **UHC**

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																												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
31:4	R	DHED	**Done Head:** When a TD is completed, the UHC writes the content of this 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 UHCINTS[WBDH] (WritebackDoneHead) bit. In normal operation, the host controller driver should not need to read this register as its content is periodically written to the HCCA.
3:0	—	Reserved	Reserved

3.8.15 UHC Frame Interval (UHCFMI)

The UHC Frame Interval (UHCFMI) register contains a 14-bit value that indicates the bit time interval in a frame, that is, 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 a 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 98.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 98: UHCFMI Bit Definitions

Physical Address 0x4C00_0034	UHCFMI																UHC													
User Settings																														
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FIT	FSMPS															Reserved		FI													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	?	1	0	1	1	1	0	1	1	0	1	1	1	1	1

Bits	Access	Name	Description
31	Read/Write	FIT	**Frame Interval Toggle:** HCD toggles this bit whenever it loads a new value to Frame Interval
30:16	Read/Write	FSMPS	**FS Largest Data Packet:** 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	Reserved
13:0	Read/Write	FI	Frame Interval: This specifies the interval between two consecutive SOFs in bit times. The nominal value is set at 11,999. HCD should store the current value of this field before resetting UHC (by setting the UHCCOMS[HCR] bit, 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.

3.8.16 UHC Frame Remaining (UHCFMR)

The UHC Frame Remaining (UHCFMR) register is a 14-bit down counter showing the bit time remaining in the current frame.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 99: UHCFMR Bit Definitions

Physical Address 0x4C00_0038	UHCFMR	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FRT	Reserved																	FR													
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31	R	FRT	Frame Remaining Toggle: This bit is 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	Reserved
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.

3.8.17 UHC Frame Number (UHCFMN)

The UHC Frame Number (UHCFMN) register is a 16-bit counter which provides a timing reference.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 100: UHCFMN Bit Definitions

Physical Address 0x4C00_003C	UHCFMN	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	0	0	0	0	0	0

Bits	Access	Name	Description
31:16	—	Reserved	Reserved

Table 100: UHCFMN Bit Definitions (Continued)

Physical Address 0x4C00_003C	UHCFMN	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	0	0	0	0	0	0

Bits	Access	Name	Description
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 0x0000 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]).

3.8.18 UHC Periodic Start (UHCPERS)

The UHC PERiodic Start (UHCPERS) register has a 14-bit programmable value that determines the earliest time the UHC should start processing the periodic list.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 101: UHCPERS Bit Definitions

Physical Address 0x4C00_0040	UHCPERS	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																		PS													
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:14	—	Reserved	Reserved
13:0	Read/Write	PS	Periodic Start: 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 10% 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.

3.8.19 UHC Low-Speed Threshold (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 102.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 102: UHCLST Bit Definitions

Physical Address 0x4C00_0044	UHCLST	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	Reserved
11:0	Read/Write	LST	LS Threshold: This field contains an 12-bit value which 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.

3.8.20 UHC Root Hub Descriptor A (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 B register.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 103: UHCRHDA Bit Definitions

Physical Address 0x4C00_0048	UHCRHDA	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	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
31:24	Read/Write	POTPGT	Power onto Power Good Time: The duration that the HCD must wait before accessing a powered-on port (in 2 ms units).
23:13	—	Reserved	Reserved

Table 103: UHCRHDA Bit Definitions (Continued)

Physical Address 0x4C00_0048	UHCRHDA	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	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
12	Read/Write	NOCP	No Over-current Protection: This bit describes if over-current protection is supported. If this bit is set, then over-current protection is not supported. If clear, over-current protection is supported. 0 = Over-current protection is supported 1 = No over-current protection supported **NOTE:** If NOCP is set, a false over-current condition could still be triggered and registerd in the UHSTAT register.
11	Read/Write	OCPM	Over-Current Protection Mode: 0 = This bit must be cleared. This bit describes how the over-current status for the root hub ports are reported. At reset, this field reflects the same mode as Power-switching mode. This field is valid only if the no over current protection (NOCP) field is cleared.Over-current status is reported collectively for all downstream ports. Per port over-current protection is not supported. 1 = Reserved
10	Read	DT	**Device Type:** 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.
9	Read/Write	NPS	**No Power Switching:** This bit is used to specify if power switching is supported or if ports are always powered. When this bit is cleared, the Power-switching mode (UHCRHDA[PSM]) bit specifies global or per-port switching, that is, 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

Table 103: UHCRHDA Bit Definitions (Continued)

Physical Address 0x4C00_0048	UHCRHDA	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	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
8	Read/Write	PSM	Power Switching Mode: This bit is used to specify how the power switching of the root hub ports is controlled. This field is only valid if UHCRHDA[NPS] (No Power Switching) 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 UHCRHDB[PPCM] (port power control mask) is set, the port responds only to port power commands (Set/Clear Port Power: to Set Port Power, write a 1 to UHCRHPS1/2/3[PPS], to Clear Port Power, write a 1 to UHCRHPS1/2/3[LDA]). If the port mask is cleared, the port is controlled only by the global power switch (to Set/Clear Global Power: write 1 to UHCRHS[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	Read	NDP	**Number Downstream Ports:** These bits specify the number of downstream ports supported by the Root Hub. **Note:** Although only some of the USB host ports are available in the current package configuration, the USB host controller supports three downstream ports, so this field always contains 0b0000_0010.

3.8.21 UHC Root Hub Descriptor B (UHCRHDB)

The 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, 3, 17, 18, and 19 are writable for PXA30x; all other bits always read as 0. For PXA31x, bits 1 and 17 are read-only.

Table 104: UHCRHDB Bit Definitions

Physical Address 0x4C00_004C	UHCRHDB	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	Read/Write	PPCM	Port Power Control Mask: 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, 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 (PXA32x and PXA30x only) bit 17: Unused, always reads as 0 (PXA31x only) bit 18: Ganged-power mask on Port #2 bit 19: Ganged-power mask on Port #3 bits 20 - 31: Unused, l always reads as 0
15:0	Read/Write	DR	Device Removable: 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, and bit 2 refers to Port 2, and bit 3 refers to Port 3. Bits 15 through 4 and bit 0 are unused. If set to a 1, this bit forces the corresponding Port Status register current connection status bit (UHCRHPS1/2/3[CSC]) to be a 1. bit 0: Unused, always reads as 0 bit 1: 1 if device attached to Port #1 is not removable (PXA32x and PXA30x only) bit 1: Unused, always reads as 0 (PXA31x only) 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

3.8.22 UHC Root Hub Status (UHCRHS)

The Root Hub Status register is divided into two parts. The lower part of a word represents the hub status field and the upper part represents the hub status change field.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 105: UHCRHS Bit Definitions

Physical Address 0x4C00_0050	UHCRHS	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CRWE	Reserved													OCIC	LPSC	DRWE	Reserved													OCI	LPS
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
31	Read/Write	CRWE	Clear Remote Wakeup Enable: Setting this CRWEt clears the device remote wakeup enable (UHCRHS[DRWE]) bit. Clearing CRWE has no effect. This bit always reads as a 0. 0 = No effect 1 = Clears UHCRHS[DRWE]
30:18	—	Reserved	Reserved
17	Read/Write 1 to clear	OCIC	Over Current Indicator Change: 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) Local Power Status Change / (Write) Set Global Power: 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 port power status (UHCRHPS1/2/3[PPS]) only on ports whose port power control mask (UHCRHDB[PPCM]) bit 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) Device Remote Wakeup Enable / (Write) Set Remote Wakeup Enable: This bit enables a connect status change bit (UHCRHPS1/2/3[CSC]) as a resume event, causing a USB Suspend to USB Resume state transition and setting the Resume Detected interrupt. If it reads 0, a connect status change is not a remote wakeup event. If it reads 1, connect status change is a remote wakeup event. Writing a 1 sets the device remote wakeup enable (this bit). Writing a 0 has no effect. 0 = ConnectStatusChange is not a remote wakeup event 1 = ConnectStatusChange is a remote wakeup event.
14:2	—	Reserved	Reserved
1	R	OCI	Over-current Indicator: When set, an over-current condition exists. When cleared, all power operations are normal. 0 = Power operations are normal 1 = Over-current condition exists.

Table 105: UHCRHS Bit Definitions (Continued)

Physical Address 0x4C00_0050	UHCRHS	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CRWE	Reserved													OCIC	LPSC	DRWE	Reserved													OCI	LPS
Reset	0	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
0	Read/Write 1 to clear CGP	LPS	(Read) Local Power Status / (Write) Clear Global Power: 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), LPS is set to turn off power to all ports (clear Port Power Status bit). In per-port Power mode, it clears Port Power Status bit only on ports whose port power control mask bit is not set. Writing a 0 has no effect. 0 = No effect 1 = Turns off power to all ports

3.8.23 UHC Root Hub Port Status1/2/3 (UHCRHPS1, UHCRHPS2, and UHCRHPS3)

UHCRHOSx, shown in Table 106, control and report USB Port 1, 2, and 3 events on a per-port basis. The lower 16 bits of UHCRHPSx reflect the port status, and the upper 16 bits reflect the status change bits. Some status bits have special write behavior. 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 PPS bit is essentially the V_{bus} enable bit. If PPS is set, it enables the USBHPENx signal. Note that the single-ended receivers must also be on via the apprpriate SSEx bit. If cleared, V_{bus} can't be enabled via USBHPENx. The polarity of the PEN signal is controlled with the PCPL big in the UHCHR register.

When PES is set, the transmit lines, single-ended or differential, are fully enabled for communication with the bus.In order for this bit to remain set, a device must be connected and the CCS bit set, no over-current conditions may exist, and port power must be applied.

The CCS bit sets when a device connects to the controller. Several conditions must be true for CCS to be set: no over-current condition exists, the device removable bit is not set, a connect event is sensed on the bus, port is in a disconnected state prior to this connect event, and no babble activity is occurring.

Note

UHCRHPS1 should be considered reserved for PXA31x. Modifying this register has no effect.

The register organization and individual bit definitions are shown in Table 106.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown**.**

Table 106: UHCRHPS1/2/3 Bit Definitions

Physical Address	UHC
0x4C00_0054	UHCRHPS1 (PXA32x and PXA30x Only)
0x4C00_0058	UHCRHPS2
0x4C00_005C	UHCRHPS3

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	Reserved	PSSC	PESC	CSC	Reserved						LSDA	PPS	Reserved			PRS	CSS	PSS	PES	CCS
Reset	?	?	?	?	?	?	?	?	?	?	?	0	?	0	0	0	?	?	?	?	?	?	0	0	?	?	?	0	0	0	0	0

Bits	Access	Name	Description
31:21	—	Reserved	Reserved
20	Read/Write	PRSC	**Port Reset Status Change:** 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	—	Reserved	Reserved
18	Read/Write	PSSC	Port Suspend Status Change: 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 reset status change is set. 0 = Resume is not completed 1 = Resume completed
17	Read/Write	PESC	**Port Enable Status Change:** This bit is set when events (such as over-current condition, disconnect, switched-off power, or operational bus error-babble), cause the port enable status 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 Port Enable Status 1 = Change in Port Enable Status

Table 106: UHCRHPS1/2/3 Bit Definitions (Continued)

Physical Address	UHC
0x4C00_0054	UHCRHPS1 (PXA32x and PXA30x Only)
0x4C00_0058	UHCRHPS2
0x4C00_005C	UHCRHPS3

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	Reserved	PSSC	PESC	CSC	Reserved						LSDA	PPS	Reserved			PRS	CSS	PSS	PES	CCS
Reset	?	?	?	?	?	?	?	?	?	?	?	0	?	0	0	0	?	?	?	?	?	?	0	0	?	?	?	0	0	0	0	0

Bits	Access	Name	Description
16	Read/Write	CSC	Connect Status Change: 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 current connect status 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 should not occur if the port is disconnected. 0 = No change in Current Connect Status 1 = Change in Current Connect Status **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	Reserved
9	Read/Write 1 to clear CPP	LSDA	(Read) Low Speed Device Attached / (Write) Clear Port Power: 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 current connect status is set. When the HCD writes a 1 to this bit, the controller clears the port power status bit. Writing a 0 has no effect. 0 = Full speed device attached 1 = Low speed device attached
8	Read/Write	PPS	(Read) Port Power Status / (Write) Set Port Power: This bit reflects the port power status, 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 **NOTE:** The HCD writes a 1 to set the Port Power Status bit. Writing a 0 has no effect.
7:5	—	Reserved	Reserved

Table 106: UHCRHPS1/2/3 Bit Definitions (Continued)

Physical Address	UHC
0x4C00_0054	UHCRHPS1 (PXA32x and PXA30x Only)
0x4C00_0058	UHCRHPS2
0x4C00_005C	UHCRHPS3

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	Reserved	PSSC	PESC	CSC	Reserved						LSDA	PPS	Reserved			PRS	CSS	PSS	PES	CCS
Reset	?	?	?	?	?	?	?	?	?	?	?	0	?	0	0	0	?	?	?	?	?	?	0	0	?	?	?	0	0	0	0	0

Bits	Access	Name	Description
4	Read/Write	PRS	(Read) Port Reset Status / (Write) Set Port Reset: 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 port reset status change is set. This bit cannot be set if current connect status is cleared. The HCD sets the port reset signaling by writing a 1 to this bit. Writing a 0 has no effect. If current connect status 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	Write	CSS	(Write) Clear Suspend Status: The HCD writes a 1 to this bit to initiate a resume. Writing a 0 has no effect. A resume is initiated only if port suspend status is set. 0 = No effect 1 = Writes a 1 to this bit to initiate a resume
2	Read/Write	PSS	(Read) Port Suspend Status / (Write) Set Port Suspend: 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 port suspend status change is set at the end of the resume interval. This bit cannot be set if current connect status is cleared. This bit is also cleared when port reset status 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 should propagate to the HC. The HCD sets the port suspend status bit by writing a 1 to this bit. Writing a 0 has no effect. If current connect status is cleared, this write does not set port suspend status; instead it sets connect status change. This informs the driver that it attempted to suspend a disconnected port. 0 = Port is not suspended 1 = Port is suspended

Table 106: UHCRHPS1/2/3 Bit Definitions (Continued)

Physical Address	UHC
0x4C00_0054	UHCRHPS1 (PXA32x and PXA30x Only)
0x4C00_0058	UHCRHPS2
0x4C00_005C	UHCRHPS3

	31	30	29	28	27	26	25	24	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	Reserved	PSSC	PESC	CSC	Reserved						LSDA	PPS	Reserved			PRS	CSS	PSS	PES	CCS
Reset	?	?	?	?	?	?	?	?	?	?	?	0	?	0	0	0	?	?	?	?	?	?	0	0	?	?	?	0	0	0	0	0

Bits	Access	Name	Description
1	Read/Write	PES	(Read) Port Enable Status / (Write) Set Port Enable: Indicates whether the port is enabled or disabled. The root hub can clear this bit when an over-current condition, disconnect event, switched-off power, or operational bus error, such as babble (unexpected bus activity), is detected. This change also causes the Port Enabled Status Change bit (UHCRHPS1/2/3[PESC] to be set. The HCD sets the Port Enable Status 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 current connect status is cleared, this write does not set port enable status, but instead sets connect status change. This informs the driver that it attempted to enable a disconnected port. Writing a 0 to the Port Enable Status bit has no effect. If not already set, this bit is also set 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	Read/Write	CCS	(Read) Current Connect Status / (Write) Clear Current Connect Status: 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 Current Connect status bit. Writing a 0 has no effect. The current connect status 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

3.8.24 UHC Status Register (UHCSTAT)

UHCSTAT, shown in Table 107, lets users monitor the non-OHCI interrupt sources. When an event occurs, if the interrupt enable bit for that event is set in the UHCHIE register, 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 be seen in this register. Refer to the section on the interrupt controller for more details on the two USB host interrupts.

The register organization and individual bit definitions are shown in Table 107.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 107: UHCSTAT Bit Definitions

Physical Address 0x4C00_0060	UHCSTAT	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	Reserved																SBMAI	SBTAI	UPRI	Reserved	UPS1	HTA	Reserved	HBA	RWUE	Reserved						
PXA31x Only	Reserved															UPS3	SBMAI	SBTAI	UPRI	Reserved	Reserved	HTA	Reserved	HBA	RWUE	Reserved						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	?	0	0	?	0	0	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:17	—	Reserved	Reserved
16	Read/Write 1 to Clear	UPS3	USB Over-Current Sense The processors support only global over-current reporting. The available ports are tied together internally so that the USBHPWR signal reports an over-current condition for any associated ports. Externally, any ports in use that need over-current sense can be ORd together and connected to the USBHPWR pin. When an over-current condition exists on a port, its UHCRHS[OCI] bit is set. Additionally, the UPS3 bit is set, causing the appropriate interrupt to be generated if the over-current sense interrupt is enabled in the UHCHIE register. Even when the over-current sense interrupts are not enabled in the UHCIE register, false over-currents can still be triggered in the UHCSTAT register. For example, if the polarity of UHCHR[PSPL] bit is programmed too late in the controller initialization routine, a false over-current condition is triggered. Each time the PSPL bit is flipped, it causes an over-current to be reported by the UPS3 bits. This condition occurs whether over-current reporting is enabled or not. 0 = No port over-current 1 = Port over-current sense **NOTE:** PXA31x processor Only
	—	Reserved	Reserved (PXA32x and PXA30x Only)
15	Read/Write 1 to Clear	SBMAI	System Bus Master Abort Interrupt Indicates a master abort event on a USB host-initiated system bus transfer. This interrupt is non-maskable. 1 = Master abort event on a USB host-initiated system bus transfer

Table 107: UHCSTAT Bit Definitions (Continued)

Physical Address 0x4C00_0060 **UHCSTAT** **UHC**

	31	30	29	28	27	26	25	24	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																																
PXA32x and PXA30x Only	Reserved																SBMAI	SBTAI	UPRI	Reserved	UPS1	HTA	Reserved	HBA	RWUE	Reserved						
PXA31x Only	Reserved															UPS3	SBMAI	SBTAI	UPRI	Reserved	Reserved	HTA	Reserved	HBA	RWUE	Reserved						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	?	0	0	?	0	0	?	?	?	?	?	?	?

Bits	Access	Name	Description
14	Read/Write 1 to Clear	SBTAI	System Bus Target Abort Interrupt Indicates a target abort event on a USB host-initiated system bus transfer. This interrupt is non-maskable. 1 = Target abort event on a USB host-initiated system bus transfer
13	Read/Write 1 to Clear	UPRI	USB Port Resume Interrupt Indicates a resume, a connect, or a disconnect event on Port 1, Port 2, or Port 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 the clocks and power management chapters for details). An asynchronous signal is sent to the wakeup controller of the clock unit, which then begins sending USB clocks after which, the UHC sends this interrupt to the interrupt controller. 1 = Resume, connect, or disconnect event on Port 1, Port 2. or Port 3.
12	—	Reserved	Reserved

Table 107: UHCSTAT Bit Definitions (Continued)

Physical Address 0x4C00_0060	UHCSTAT	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	Reserved																SBMAI	SBTAI	UPRI	Reserved	UPS1	HTA	Reserved	HBA	RWUE	Reserved						
PXA31x Only	Reserved															UPS3	SBMAI	SBTAI	UPRI	Reserved	Reserved	HTA	Reserved	HBA	RWUE	Reserved						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	?	0	0	?	0	0	?	?	?	?	?	?	?

Bits	Access	Name	Description
11	Read/Write 1 to Clear	UPS1	USB Over-Current Sense The processors support only global over-current reporting. The available ports are tied together internally so that the USBHPWR signal reports an over-current condition for any associated ports. Externally, any ports in use that need over-current sense can be ORd together and connected to the USBHPWR pin. When an over-current condition exists on a port, its UHCRHS[OCI] bit is set. Additionally, the UPS1 bit is set, causing the appropriate interrupt to be generated if the over-current sense interrupt is enabled in the UHCHIE register. Even when the over-current sense interrupts are not enabled in the UHCIE register, false over-currents can still be triggered in the UHCSTAT register. For example, if the polarity of UHCHR[PSPL] bit is programmed too late in the controller initialization routine, a false over-current condition is triggered. Each time the PSPL bit is flipped, it causes an over-current to be reported by the UPS1 bit. This condition occurs whether over-current reporting is enabled or not. 0 = No port over-current 1 = Port over-current sense **NOTE:** PXA32x processor and PXA30x processor Only
	—	Reserved	Reserved (PXA31x Only)
10	Read/Write 1 to Clear	HTA	HCI Transfer Abort 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 set, HTA indicates that the application should terminate the transfer and that the FIFOs are cleared. This event is active for a minimum of 10 µs. The HCD should not try to force the host controller to re-enter the operational state from the suspend state until a minimum of 10 µsec after this interrupt. The HCD can clear this bit by setting it. 1 = HCI interface clear signal has gone active.
9	—	Reserved	Reserved

Table 107: UHCSTAT Bit Definitions (Continued)

Physical Address 0x4C00_0060	UHCSTAT	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	Reserved																SBMAI	SBTAI	UPRI	Reserved	UPS1	HTA	Reserved	HBA	RWUE	Reserved						
PXA31x Only	Reserved															UPS3	SBMAI	SBTAI	UPRI	Reserved	Reserved	HTA	Reserved	HBA	RWUE	Reserved						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	?	0	0	?	0	0	?	?	?	?	?	?	?

Bits	Access	Name	Description
8	Read/Write 1 to Clear	HBA	HCI Buffer Active Set while the host controller accesses the data buffer for the current transfer descriptor (TD) or when an ED is accessed (this informs the application whether the host controller is accessing the shared memory). HCD can clear this bit by setting it. 1 = Host controller is accessing the shared memory.
7	Read/Write 1 to Clear	RWUE	HCI Remote Wake Up Event If enabled (UHCHIE[RWIE] bit and the UHCHCON[RWE] bit are set), this bit is set when a remote wakeup event occurs on a downstream port 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 setting it. 1 = Remote Wake Up event on one of the downstream ports of the root hub
6:0	—	Reserved	Reserved

3.8.25 UHC Reset Register (UHCHR)

UHCHR 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 General Reset bit (CGR) is used only for simulation and test, and user software should never set this bit. This register should 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]).

The SSEx bits control the per-port single-ended receivers. These bits are also one of the bits that need to be configured to enable the V_{bus} enable signal, USBHPENx. The global SSE bit has to be enabled in order for any of the per-port single-ended receivers to be enabled.

The PSPL bit controls the polarity of the USBHPWR over-current sense input pin. Care must be taken to program the polarity of PSPL early in the inititalization of the UHC; otherwise, a false over-current can be triggered whether sensing is enabled or not.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown..

Table 108: UHCHR Bit Definitions

Physical Address 0x4C00_0064	UHCHR	UHC

User Settings																														
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	Reserved																				SSEP3	SSEP2	SSEP1	Reserved	PCPL	PSPL	SSE	UIT	SSDC	CGR	FHR	FSBIR
PXA31x Only	Reserved																				SSEP3	SSEP2	Reserved	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	Reserved
11	Read/Write	SSEP3	Sleep Standby Enable For Port 3: 0 = Enable power to the USB single-ended receivers and the USB Port 3 power supply 1 = Disables power to the USB single-ended receiver and the USB Port 3 power supply
10	Read/Write	SSEP2	Sleep Standby Enable For Port 2: 0 = Enable power to the USB single-ended receivers and the USB Port 2 power supply. 1 = Disables power to the USB single-ended receiver and the USB Port 2 power supply
9	Read/Write	SSEP1	Sleep Standby Enable For Port 1: 0 = Enable power to the USB single-ended receivers and the USB Port 1 power supply 1 = Disables power to the USB single-ended receiver and the USB Port 1 power supply. **NOTE:** PXA32x and PXA30x Only
	—	Reserved	Reserved (PXA31x Only)
8	—	Reserved	Reserved
7	Read/Write	PCPL	Power Control Polarity Low: This bit selects the polarity of the USBHPEN<2:1> output signals. When set to a 1, the output signals are considered active-low. 0 = The polarity of the USBHPEN<2:1> output signals is active high 1 = The output signals are considered active low
6	Read/Write	PSPL	Power Sense Polarity Low: This bit selects the polarity of the USBHPWR input signal. When set to a 1, the input signals are considered active-low. 0 = The polarity of the USBHPWR input signal is active high 1 = The input signals are considered active low

Table 108: UHCHR Bit Definitions (Continued)

Physical Address 0x4C00_0064 **UHCHR** **UHC**

	31	30	29	28	27	26	25	24	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
PXA32x and PXA30x Only	Reserved																				SSEP3	SSEP2	SSEP1	Reserved	PCPL	PSPL	SSE	UIT	SSDC	CGR	FHR	FSBIR
PXA31x Only	Reserved																				SSEP3	SSEP2	Reserved	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
5	Read/Write	SSE	Sleep Standby Enable: When set to 1, this bit disables the USB single-ended receivers and the USB port power supplies for all USB host downstream ports. 0 = Enable power to the USB single-ended receivers and the USB port power supply 1 = Disables power to the USB single-ended receiver and the USB port power supply
4	Read/Write	UIT	USB Interrupt Test: When set to 1 and Test mode is active, this bit enables the interrupt test bits that directly force the interrupt signals to the UHCSTAT register. 0 = Disable the interrupt test bits in the UHCHIT register 1 = Enable the interrupt test bits in the UHCHIT register
3	Read/Write	SSDC	Simulation Scale Down Clock: When set to 1, 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 = The 1 msec interval clock in the UHC is switched to 1 µsec.
2	Read/Write	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 = The reset to the DPLL circuit is active 1 = The reset to the DPLL is inactive

Table 108: UHCHR Bit Definitions (Continued)

Physical Address 0x4C00_0064	UHCHR	UHC

User Settings																														
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	Reserved																				SSEP3	SSEP2	SSEP1	Reserved	PCPL	PSPL	SSE	UIT	SSDC	CGR	FHR	FSBIR
PXA31x Only	Reserved																				SSEP3	SSEP2	Reserved	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
1	Read/Write	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 a 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 µsec. 0 = Normal operation 1 = OHCI core and 12 MHz logic is in reset
0	Read/Write	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 is needed if the system bus interface has already requested a bus cycle. This bit always reads as a 0.

3.8.26 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 109.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 109: UHCHIE Bit Definitions

Physical Address 0x4C00_0068	UHCHIE																UHC														
User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	Reserved																	Reserved	UPRIE	Reserved	UPS1IE	TAIE	Reserved	HBAIE	RWIE	Reserved						
PXA31x Only	Reserved																	UPS3IE	UPRIE	Reserved	Reserved	TAIE	Reserved	HBAIE	RWIE	Reserved						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	?	0	0	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:15	—	Reserved	Reserved
14	Read/Write	UPS3IE	USB Over-Current Sense Interrupt Enable: Enables/disables the Overcurrent Indicator pin to cause an USBHStatus[UPS3] type interrupt. 0 = Disable interrupt 1 = Enable interrupt **NOTE:** PXA31x Only
	—	Reserved	Reserved (PXA32x and PXA30x Only)
13	Read/Write	UPRIE	USB Port Resume Interrupt Enable: If enabled, an interrupt is generated if a resume condition is generated on either Port 1, Port 2 or Port 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 12- and 48-MHz clocks are turned off. 0 = Disable interrupt 1 = Enable interrupt
12	—	Reserved	Reserved
11	Read/Write	UPS1IE	USB Overcurrent Sense Interrupt Enable: Enables/disables the Overcurrent Indicator pin to cause an USBHStatus[UPS1] type interrupt. 0 = Disable interrupt 1 = Enable interrupt **NOTE:** PXA32x and PXA30x Only
	—	Reserved	Reserved (PXA31x Only)

Table 109: UHCHIE Bit Definitions (Continued)

Physical Address 0x4C00_0068	UHCHIE																UHC														
User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	Reserved																	Reserved	UPRIE	Reserved	UPS1IE	TAIE	Reserved	HBAIE	RWIE	Reserved						
PXA31x Only	Reserved																	UPS3IE	UPRIE	Reserved	Reserved	TAIE	Reserved	HBAIE	RWIE	Reserved						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	?	0	0	?	?	?	?	?	?	?

Bits	Access	Name	Description
10	Read/Write	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	Reserved
8	Read/Write	HBAIE	HCI Buffer Active Interrupt Enable: Enables/disables the HCI buffer active-type interrupt. 0 = Disable interrupt 1 = Enable interrupt
7	Read/Write	RWIE	HCI Remote wakeup Interrupt Enable: Enables/disables the remote wakeup interrupt. 0 = Disable interrupt 1 = Enable interrupt
6:0	—	Reserved	Reserved

3.8.27 UHC Interrupt Test Register (UHCHIT)

UHCHIT, shown in Table 110, 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 on 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 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 110.

All reserved bits are read as unknown. Reserved bits must be written with 0b0. A question mark indicates that the reset value is unknown.

Table 110: UHCHIT Bit Definitions

Physical Address 0x4C00_006C	UHCHIT	UHC

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	Reserved																SMAT	STAT	UPRT	Reserved	UPS1T	TAT	IRQT	BAT	RWUT	Reserved						
PXA31x Only	Reserved															UPS3T	SMAT	STAT	UPRT	Reserved	Reserved	TAT	IRQT	BAT	RWUT	Reserved						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:17	—	Reserved	Reserved
16	Read/Write	UPS3T	USB Overcurrent Sense interrupt Test: 0 = No effect 1 = Force interrupt strobe if UHCHIE[UPS3IE] bit is set **NOTE:** PXA31x Only
	—	Reserved	Reserved (PXA32x and PXA30x Only)
15	Read/Write	SMAT	System Bus Master Abort Interrupt Test: 0 = No effect 1 = Force interrupt strobe (non-maskable)
14	Read/Write	STAT	System Bus Target Abort Interrupt Test: 0 = No effect 1 = Force interrupt strobe (non-maskable)
13	Read/Write	UPRT	USB Port Resume Interrupt Test: 0 = No effect 1 = Force interrupt strobe if UHCHIE[UPRIE] bit is set
12	-	Reserved	Reserved
11	Read/Write	UPS1T	USB Overcurrent Sense Interrupt Test: 0 = No effect 1 = Force interrupt strobe if UHCHIE[UPS1IE] bit is set **NOTE:** (PXA32x and PXA30x Only)
	-	Reserved	Reserved (PXA31x Only)
10	Read/Write	TAT	HCI Interface Transfer Abort Interrupt Test: 0 = No effect 1 = Force interrupt strobe if UHCHIE[TAIE] bit is set
9	Read/Write	IRQT	Normal OHC Interrupt Test: 0 = No effect 1 = Force interrupt strobe

Table 110: UHCHIT Bit Definitions (Continued)

Physical Address 0x4C00_006C	UHCHIT																UHC													
User Settings																														
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	Reserved																SMAT	STAT	UPRT	Reserved	UPS1T	TAT	IRQT	BAT	RWUT	Reserved						
PXA31x Only	Reserved															UPS3T	SMAT	STAT	UPRT	Reserved	Reserved	TAT	IRQT	BAT	RWUT	Reserved						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?

Bits	Access	Name	Description
8	Read/Write	BAT	HCI Buffer Active Interrupt Test: 0 = No effect 1 = Force interrupt strobe if UHCHIE[BAIE] bit is set
7	Read/Write	RWUT	HCI Remote wakeup Interrupt Test: 0 = No effect 1 = Force interrupt strobe if UHCHIE[RWUIE] bit is set
6:0	—	Reserved	Reserved

4 SSP Serial Ports

This chapter describes the signal definitions and operation of the four Synchronous Serial Protocol (SSP) serial ports SSP1, SSP2, SSP3, and SSP4 for the processor. All four SSPs are identical in operation and differ only in the number of external interface signals for each SSP and in the memory-map base-address location for each SSP.

4.1 Overview

The SSP port is a synchronous serial controller that can be connected 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 controllers directly supports the Texas Instruments* Synchronous Serial Protocol (SSP) and the Motorola* Serial Peripheral Interface (SPI) protocol. Inter-IC Sound (I^2S) protocol is supported by programming the Programmable Serial Protocol (PSP) mode for I^2S operation.

Note

PSP format is supported only for emulation of the I^2S protocol. Operation in PSP mode except as described in *PXA3xx Processor Family I2S Emulation Using SSP/PSP Application Note* is not supported.

The SSP ports are configurable to operate in master mode (the attached peripheral functions as a slave) or slave mode (the attached peripheral functions as a master). The SSP ports supports serial bit rates from 6.3 Kbps (minimum recommended speed) up to 26 Mbps. Serial data sample size can be set to 8 bits, 16 bits, 18 bits or 32 bits in length. A FIFO is provided for transmit data and a second, independent FIFO is provided for receive data. The two FIFOs are both 16 samples deep x 32 bits wide or both 32 samples deep x 16 bits wide.

The FIFOs can be loaded or emptied by the processor using programmed I/O or by DMA burst transfers.

4.1.1 PXA3xx Processor Differences

Table 111 shows the SSP controller differences among the PXA32x, PXA31x, and PXA30x processors. Refer to each individual register for other operating differences.

Table 111: PXA3xx Processors Feature Differences

Feature	PXA30x	PXA31x	PXA32x
Maximum Master Mode Bit Rate	13 Mbps	26 Mbps	13 Mbps
Even/Odd TX indicator	Not Supported	Supported	Not Supported

3. The SSCR0_x[52MM] is used to select the bit rate.
4. The SSSR_x[TX_OSS] indicates whether there are an even or odd number of samples in the TX FIFO.

4.2 Features

The SSP port features are:

- Directly supports Texas Instruments* Synchronous Serial (SSP) and Motorola* Serial Peripheral Interface (SPI).
- The Inter-IC Sound (I^2S) protocol is supported by programming the Programmable Serial Protocol (PSP). PSP format is only supported for emulation of the I^2S protocol.
- Data sample sizes can be set to 8, 16, 18 or 32 bits
- One FIFO for transmit data (TXFIFO) and a 2nd, independent, FIFO for receive data (RXFIFO). For non-packed data mode, the two FIFOs are each 16 rows deep x 32 bits wide for a total of 16 samples.
- FIFO packed mode allows double depth FIFOs if the samples are 8 bits or 16 bits wide. For packed data mode, both FIFOs are 32 locations deep x 16 bits wide for a total of 32 samples.
- Master mode and slave mode operation supported
  - The PXA32x processor and PXA30x processor supports a maximum serial bit-rate of 13 Mbps. The PXA31x processor supports a maximum serial bit-rate of 26 Mbps in master mode, and 13 Mpbs in slave mode. (see Note in Section 4.4.1 and Section 4.4.1.1 regarding limitation)
- Receive-without-transmit operation
- Network mode with up to eight time slots for PSP formats, and independent Transmit/Receive in any/all/none of the time slots.
- Audio clock control to provide a 4x or 8x output clock to support most standard audio frequencies.

4.3 Signal Descriptions

This section lists the external interface signals for each of the SSPx ports.

4.3.1 External Interface to Synchronous Serial Peripherals

Table 112, Table 113, External Interface (SSP2),Table 114, External Interface (SSP3),and Table 115, External Interface (SSP4),list the external interface signals from SSP1, SSP2, SSP3 and SSP4, respectively.

Table 112: External Interface (SSP1)

Name	Direction	Description
SSPSCLK	Inout	Synchronous Serial Protocol Serial Clock: Controls the timing of a serial transfer. SSPSCLK can be generated internally (master mode) or taken from an external source (slave mode) See SSCR1_1[SCLKDIR] as defined in Table 122.
SSPSFRM	Inout	Synchronous Serial Protocol Serial Frame Indicator: Indicates the beginning and the end of a serialized data sample. The SSPSFRM can be generated internally (master mode) or taken from an external source (slave mode). See SSCR1_1[SFRMDIR] as defined in Table 122.
SSPTXD	Output	Synchronous Serial Protocol Transmit Data: Serial data out. Sample length is selected by SSCR0_1[EDSS,DSS] as defined in Table 121.

Table 112: External Interface (SSP1) (Continued)

Name	Direction	Description
SSPSYSCLK	Output	Synchronous Serial Protocol System Clock: An internal version of this signal is divided by 1, 4 or 8 to generate SSPSCLK. See SSACD_1[SCDX8] as defined in Table 132 and SSACD_x[SCDB].
SSPRXD	Input	Synchronous Serial Protocol Receive Data: Serial data in. Sample length is selected by SSCR0_1[EDSS,DSS] as defined in Table 121.
SSPEXTCLK	Input	Synchronous Serial Protocol External Clock: A selectable external clock that replaces the internal 13-MHz clock when SSPSCLK is an output. The SSPEXTCLK input is only used when SSCR0_1[ACS=0,NCS=0,ECS=1] as defined in Table 121 and SSCR1_1[SCLKDIR=0] as defined in Table 122.
SSPSCLKEN	Input	Synchronous Serial Protocol Serial Clock Enable: An asynchronous external enable for SSPSCLK that is only recognized when SSCR0_1[ECS=0] as defined in Table 121 and when the SSP port is the master with respect to the SSPSCLK, that is., SSCR1_1[SCLKDIR=0] as defined in Table 122. 0 = disables SSPSCLK 1 = enables SSPSCLK

Table 113: External Interface (SSP2)

Name	Direction	Description
SSPSCLK2	Inout	Synchronous Serial Protocol Serial Clock 2: Controls the timing of a serial transfer. SSPSCLK can be generated internally (master mode) or taken from an external source (slave mode): See SSCR1_2[SCLKDIR] as defined in Table 122.
SSPSFRM2	Inout	Synchronous Serial Protocol Serial Frame Indicator 2: Indicates the beginning and the end of a serialized data sample. The SSPSFRM can be generated internally (master mode) or taken from an external source (slave mode): See SSCR1_2[SFRMDIR] as defined in Table 122.
SSPTXD2	Output	Synchronous Serial Protocol Transmit Data 2: Serial data out. Sample length is selected by the SSCR0_2[EDSS,DSS] bits as defined in Table 121.
SSPSYSCLK2	Output	Synchronous Serial Protocol System Clock 2: An internal version of this signal is divided by 1, 4 or 8 to generate SSPSCLK. See SSACD_2[SCDX8] as defined in Table 132.
SSPRXD2	Input	Synchronous Serial Protocol Receive Data 2: Serial data in. Sample length is selected by the SSCR0_2[EDSS,DSS] bits as defined in Table 121.
SSPEXTCLK2	Input	Synchronous Serial Protocol External Clock 2: A selectable external clock that replaces the internal 13-MHz clock when SSPSCLK2 is an output. The SSPEXTCLK2 input is only used when SSCR0_2[ECS=1,NCS=0] as defined in Table 121 and SSCR1_2[SCLKDIR=0] as defined in Table 122.

Table 113: External Interface (SSP2) (Continued)

Name	Direction	Description
SSPSCLK2EN	Input	Synchronous Serial Protocol Serial Clock Enable 2: An asynchronous external enable for SSPSCLK that is only recognized when SSCR0_2[ECS=0] as defined in Table 121 and when the SSP2 port is the master with respect to the SSPSCLK, that is, SSCR1_2[SCLKDIR=0] as defined in Table 122. 0 = disables SSPSCLK2 1 = enables SSPSCLK2

Table 114: External Interface (SSP3)

Name	Direction	Description
SSPSCLK3	Inout	Synchronous Serial Protocol Serial Clock 3: Controls the timing of a serial transfer. SSPSCLK can be generated internally (master mode) or taken from an external source (slave mode): See SSCR1_3[SCLKDIR] as defined in Table 122.
SSPSFRM3	Inout	Synchronous Serial Protocol Serial Frame Indicator 3: Indicates the beginning and the end of a serialized data sample. The SSPSFRM3 can be generated internally (master mode) or taken from an external source (slave mode): See SSCR1_3[SFRMDIR] as defined in Table 122.
SSPTXD3	Output	Synchronous Serial Protocol Transmit Data 3: Serial data out. Sample length is selected by the SSCR0_3[EDSS,DSS] bits as defined in Table 121.
SSPSYSCLK3	Output	Synchronous Serial Protocol System Clock 3: An internal version of this signal is divided by 1, 4 or 8 to generate SSPSCLK. See SSACD_3[SCDX8] as defined in Table 132.
SSPRXD3	Input	Synchronous Serial Protocol Receive Data 3: Serial data in. Sample length is selected by the SSCR0_3[EDSS,DSS] bits as defined in Table 121.

Note

SSP3 does not have an external clock input, SSPEXTCLK3, or the clock-enable input SSPSCLK3EN.

Table 115: External Interface (SSP4)

Name	Direction	Description
SSPSCLK4	Inout	Synchronous Serial Protocol Serial Clock 4: Controls the timing of a serial transfer. SSPSCLK4 can be generated internally (master mode) or taken from an external source (slave mode): See SSCR1_4[SCLKDIR] as defined in Table 122.

Table 115: External Interface (SSP4) (Continued)

Name	Direction	Description
SSPSFRM4	Inout	Synchronous Serial Protocol Serial Frame Indicator 4: Indicates the beginning and the end of a serialized data sample. The SSPSFRM4 can be generated internally (master mode) or taken from an external source (slave mode): See SSCR1_4[SFRMDIR] as defined in Table 122.
SSPTXD4	Output	Synchronous Serial Protocol Transmit Data 4: Serial data out. Sample length is selected by the SSCR0_4[EDSS,DSS] bits as defined in Table 121.
SSPSYSCLK4	Output	Synchronous Serial Protocol System Clock 4: An internal version of this signal is divided by 1, 4 or 8 to generate SSPSCLK. See SSACD_4[SCDX8] as defined in Table 132.
SSPRXD4	Input	Synchronous Serial Protocol Receive Data 4: Serial data in. Sample length is selected by the SSCR0_4[EDSS,DSS] bits as defined in Table 121.

Note

SSP4 does not have an external clock input, SSPEXTCLK4, or the clock-enable input SSPSCLK4EN.

4.4 Operation

Serial data is transferred between the processor and an external peripheral through FIFOs in the SSP port. Data transfers between an SSPx port and memory are initiated by the processor using programmed I/O or by DMA bursts. Separate transmit and receive FIFOs and serial data paths permit simultaneous transfers in both directions to and from the external peripheral, depending on the protocols chosen.

Programmed I/O can transfer data between:

- The processor and the FIFO Data register for the TXFIFO
- The processor and the FIFO Data register for the RXFIFO
- The processor and the SSP control or status registers

DMA bursts can transfer data between:

- The memory and the FIFO Data register for the TXFIFO
- The memory and the FIFO Data register for the RXFIFO

4.4.1 SSP FIFO Access

The data is accessed through the TXFIFO and RXFIFO. A processor access takes the form of programmed I/O, transferring one FIFO entry per access (see Section 4.4.1.1). Processor accesses are normally triggered by an interrupt caused by a Status register event (see SSP Status register, Table 124, SSSR_x Bit Definitions,) and must always be 32 bits wide. Processor Writes to the TXFIFO are 32-bits wide, but the serializing logic ignores all bits beyond the programmed FIFO data size (see SSCR0_x[EDSS] and SSCR0_x[DSS], Table 121, SSCR0_x Bit Definitions,). Processor reads from the RXFIFO are also 32 bits wide, but the data that is received by the RXD interface signal is written, with zeroes inserted in the MSBs down to the programmed data size, into the RXFIFO.

The TXFIFO and RXFIFO can also be accessed by DMA bursts, which must be 8 bytes, 16 bytes or 32 bytes in length, and must transfer one FIFO entry per access. When SSCR0[EDSS] is set, the SSPx port must be configured as a 32-bit peripheral. The DMA DCMD[WIDTH] register must be at least the data size programmed into SSCR0_x[EDSS] and SSCR0_x[DSS](see Table 121, SSCR0_x Bit Definitions,).

The TXFIFO and RXFIFO are each seen as one 32-bit location by the processor. For data transmission, the SSPx port takes the data from the TXFIFO, serializes it, and transmits it via the SSPTXD serial interface signal to the external peripheral. Data received from the external peripheral via the SSPRXD interface signal is converted to parallel words and written into the RXFIFO.

A programmable FIFO trigger threshold, when exceeded, generates an interrupt or DMA service request that, if enabled, signals the processor or DMA, respectively, to empty the RXFIFO or to refill the TXFIFO.

The TXFIFO and RXFIFOs are differentiated by whether the access is a Read or a Write transfer. Reads from the FIFO Data register automatically target the RXFIFO. Writes to the FIFO Data register automatically target the TXFIFO. From a memory-map perspective, the TXFIFO and the RXFIFO are at the same address. Each FIFO is 16 rows deep x 32 bits wide for a total of 16 data samples. Each sample can be 8 bits, 16 bits, 18 bits, or 32 bits in length.

Note

At serial bit rates approaching 13 MHz for continuous data transfers, the DMA might not be able to access the RXFIFO or TXFIFO fast enough to avoid overflow or underflow, respectively. Using packed mode improves performance.

4.4.1.1 FIFO Operation in Packed Mode

When the TXFIFO and RXFIFO are operating in packed mode, each FIFO is 32 rows deep x 16-bits wide for a total of 32 data samples. For packed mode, each sample can be 8 bits or 16 bits in length. When the data is serialized and transmitted, bits 15 to 0 are transmitted first, followed by bits 31 to 16.

When the TXFIFO and RXFIFO are operating in packed mode, they may best be thought of as a single entry of 32 bits holding two 8-bit or 16-bit samples. Thus, the processor or the DMA should write and read 32 bits of data at a time where each Write or Read transfers two samples. The entire FIFO width (32 bits) must be read/written in this mode. The SSPx port does not support writing two separate 16-bit samples in this mode. Calculate the thresholds based on the number of 32-bit Writes or Reads, not the number of 16-bit or less values.

4.4.2 Trailing Bytes in RXFIFO

When the number of samples in the RXFIFO is less than its trigger threshold level, and no additional data is received, the remaining bytes are called *RXFIFO trailing bytes*. RXFIFO Trailing bytes can be handled by either the processor or DMA, as indicated by the SSCR1_x[TRAIL] bit (see Table 122). RXFIFO Trailing bytes are identified by means of a time-out mechanism and the existence of data within the RXFIFO after timeout.

Note

When FIFO packed mode is used, the DMA can not be used to handle the RXFIFO trailing bytes. The RXFIFO trailing bytes must be handled by the processor.

4.4.2.1 Timeout

A timeout condition exists when the RXFIFO has been idle for a period of time defined by the value programmed within the SSP Timeout register, SSTO_x[TIMEOUT], see Table 127. When a timeout occurs, the receiver-timeout interrupt SSSR_x[TINT] bit is set to a 1, and if the timeout interrupt is enabled by the SSCR1_x[TINTE=1] bit (see Table 122), a timeout interrupt signals the processor that a timeout condition has occurred. The timeout timer is reset after a new data sample is received into the RXFIFO. Once SSSR_x[TINT] (see Table 124) is set, it must be cleared by writing 0b1 to SSSR_x[TINT]. Clearing SSSR_x[TINT] also causes the timeout interrupt, if enabled, to be de-asserted.

4.4.2.2 Peripheral Trailing Byte Interrupt

It is possible for the DMA to reach the end of its Descriptor chain while removing RXFIFO data. When this happens, the processor needs to take over because the DMA can no longer service the SSPx port until a new chain is linked. When the DMA has reached the end of its Descriptor chain with data in the RXFIFO, the SSPx port:

4. Sets the peripheral trailing-byte interrupt SSSR_x[PINT] (see Table 124).
5. Asserts the SSPx port interrupt to signal to the processor that a peripheral trailing-byte interrupt condition has occurred (if the interrupt is enabled by SSCR1_x[PINTE]=1, see Table 122).
6. Sets the SSSR_x[EOC] (End of DMA Chain bit - see Table 124). If more data is received after SSSR_x[EOC] was set (and SSSR_x[EOC] remains set), then the SSSR_x[PINT] is set. SSSR_x[EOC] must be cleared by writing 0b1 to SSSR_x[EOC].
7. Once SSSR_x[PINT] is set, the processor must clear the bit by writing 0b1 to SSSR_x[PINT]. Clearing SSSR_[PINT] also de-asserts the SSP interrupt if it has been enabled (SSCR1_x[PINTE]=1, see Table 122).

The remaining bytes must then be removed by the processor as described in the processor programmed I/O method in Section 4.4.2.3, or by reprogramming a new Descriptor chain and restarting the DMA. Programmers need to be aware of this possibility. Refer to the DMA chapter for details on Descriptor programming and “end-of-chain” events.

4.4.2.3 Removing FIFO Trailing Bytes

SSCR1_x[TRAIL]=0: Trailing bytes left in the RXFIFO are handled by the processor programmed I/O.

This is the default method indicated by SSCR1_x[TRAIL]=0 (see Table 122). If a timeout occurs, the processor is only interrupted by means of a timeout interrupt if it has been enabled by setting SSCR1_x[TINTE]. To read out the trailing bytes from the RXFIFO, software should wait for the timeout interrupt and then read all trailing bytes as indicated by SSSR_x[OSS], SSSR_x [RFL] and SSSR_x [RNE] (see Table 124). To remove trailing bytes using programmed I/O, enable the timeout interrupt by setting SSCR1_x[TINTE] (see Table 122).

Note

If FIFO packed mode is enabled (SSRC0_x[FPCKE]=1), trailing bytes must be removed using programmed I/O. If SSSR_x[OSS]=1, then the last FIFO line only contains 1 sample (see Table 124).

SSCR1_x[TRAIL]=1: Trailing bytes left in the RXFIFO are handle by the DMA controller.

A DMA service request is issued automatically after SSCR1_x[TRAIL] is set and a timeout occurs, SSSR_x[TINT]=1, see Table 122 and Table 124. The DMA empties the RXFIFO unless the DMA reaches the end of its Descriptor chain (see: Section 4.4.2.2). When handling trailing bytes using the

DMA, if a timeout occurs and the RXFIFO is empty (SSSR_x[RNE]=0, see Table 124), an end-of-receive (EOR) is sent to the DMA. Refer to the DMA chapter for details regarding EOR.

If a DMA End-of-Chain (EOC) occurs (SSSR_x[EOC]=1) at the time that the last sample is read out of the RXFIFO (the DMA Descriptor chain was just exactly long enough), and the timeout counter is still running (that is, a timeout has not occurred and SSTO_x[TIMEOUT] is non-zero, see Table 127), then, when the timeout does occur, the SSPx port generates a DMA request which creates an RAS interrupt from the DMA (see DMA chapter for details). When this occurs, re-initialize the DMA registers and re-enable the channel for the SSPx port to send its EOR to the DMA controller.

Note

When the SSPx port is running in network mode and the FPCKE is set, use the processor to handle trailing bytes in the RXFIFO with SSCR1_x[TRAIL] cleared and SSCR1_x[PINTE] set. After the PINT interrupt occurs, the interrupt-service routine must clear MOD and wait for NMBSY to go low before removing any extra or trailing samples from the RXFIFO, which can be discarded.

When MOD is set, the SSPx port continues transceiving data even after the TXFIFO is empty and until MOD is cleared. DMA does not have a way to clear MOD and, therefore, it is possible that extra samples may be received in the RXFIFO. Since software must clear MOD, the processor also must handle the trailing bytes.

4.4.3 Frame Counter

The SSPx port sends a start-of-frame signal that can increment a frame counter in the OS timer (see the operating-system timer chapter for details). This signal is two SSPSCLK cycles wide. In normal mode (SSCR0_x[MOD] cleared, see Table 121), this signal is asserted on every sample received. In network mode (SSCR0_x[MOD] set), this signal is asserted once every time that the SSPSFRMx interface signal is asserted.

4.4.4 Data Formats

This section describes the types of formats used to transfer serial data between the processor and external peripherals.

4.4.4.1 Serial Data Formats for Transfer to/from Peripherals

Four interface signals for each SSPx port transfer data between the processor and external peripherals. Although three serial-data formats exist, each has the same basic structure, and in all instance the interface signals are used as:

- SSPSCLKx — Defines the bit rate at which serial data is driven onto and sampled from the port.
- SSPSFRMx — Defines the boundaries of a basic data “unit,” comprised of multiple serial bits.
- SSPTXDx — The serial datapath for transmitted data, from the SSPx port to peripheral.
- SSPRXDx — The serial datapath for received data, from peripheral to the SSPx port.

A data frame can contain from 8, 16, 18, or 32 bits (see SSCR0_x[EDSS,DSS] in Table 121). Serial data is transmitted most significant bit first. Two formats are directly supported: Motorola* SPI, Texas Instruments* Synchronous Serial Protocol (SSP). Inter-IC Sound (I^2S) protocol is supported by programming the Programmable Serial Protocol (PSP) format.

The SSPSFRM function and use varies between each format. SSPSFRM is programmable in direction, delay, polarity, and width. Master and slave modes are supported.

- SPI format: SSPSFRM 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 SSPSFRM signal can be either held low or pulsed, depending upon the value of SSCR1[SPH]). Master and slave modes are supported. SPI is a full-duplex format.
- SSP format: SSPSFRM is pulsed high for one (serial) data period at the start of each frame. Master and slave modes are supported. SSP is a full-duplex format.
- PSP format (I^2S): SSPSFRM 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 format.

The SSPSCLK function and use varies between each format.

- SPI format: Programmers choose which edge of SSPSCLK to use for switching transmit data and for sampling receive data. In addition, the phase of SSPSCLK is movable, 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 in both, the SSPSCLK only toggles during active transfers (does not run continuously).
- SSP format: Data sources switch transmit data on the rising edge of SSPSCLK, and sample receive data on the falling edge. Master and slave modes are supported. When driven by the SSPx port, the SSPSCLK only toggles during active transfers (not continuously) unless the SSPSCLKEN or ECRA/ECRB functions are used. When the SSPSCLK is driven by another device, it is allowed to be either continuous or only driven during transfers.
- PSP format (I^2S): Programmers choose which edge of SSPSCLK to use for switching transmit data, and for sampling receive data. In addition, the Idle state for SSPSCLK and the number of active clocks that precede and follow the data transmission are all programmable. Master and slave modes are supported. When driven by the SSPx port, the SSPSCLK toggles only during active transfers, not continuously, unless the SSPSCLKEN or ECRA/ECRB functions are used. When the SSPSCLK is driven by another device, it can be either continuous or driven only during transfers, but certain restrictions on PSP parameters apply (see Section 4.4.4.4).

Normally, the serial clock (SSPSCLK) when driven by the SSPx port toggles only while an active data transfer is underway. There are several conditions, however, that can cause the clock to run continuously. If the Receive-Without-Transmit mode is enabled by setting SSCR1[RWOT], the SSPSCLK toggles regardless of whether transmit data exists within the Transmit FIFO. The SSPSCLK also toggles continuously if the SSPx port is in Network mode (see Section 4.5.2.1), or if SSPSCLKEN, ECRA, or ECRB is enabled (see Section 4.5.3.5 and Section 4.5.3.6). At other times, SSPSCLK is held in an inactive or Idle state, as defined by the specified protocol under which it operates.

4.4.4.2 TI-SSP Format Details

SSPSFRM asserts for one clock period when outgoing data in the SSP controller is ready to transmit. On the following clock, data to be transmitted is driven on SSPTXD one bit at a time, most significant bit first. For receive data, the peripheral similarly drives data on the SSPRXD pin. Word length can be 8, 16, 18 or 32 bits. All output transitions occur on the rising edge of SSPSCLK while data sampling occurs on the falling edge. The SSPTXD signal either retains the value of the last bit sent (Bit 0) or goes to three-state at the end of the transfer. If the SSPx port is disabled or reset, SSPTXD is forced to zero (unless SSCR1[TTE] is set, in which case it goes into a three-state condition).

The start of one frame immediately follows the completion of the previous for back-to-back transfers. The MSB of one transfer immediately follows the LSB of the preceding with no “dead” time between them.

When the enhanced SSPx port is a master to the frame synch (SSPSFRM) and a slave to the clock (SSPSCLK), then at least three extra clocks (SSPSCLKs) are needed at the beginning and end of each block of transfers to synchronize control signals from the ARM* peripheral bus (APB) clock domain into the SSP clock domain (a block of transfers is a group of back-to-back continuous transfers).

4.4.4.3 SPI Format Details

SPI format has four possible sub-modes, depending on the SSPSCLK edges selected for driving data and sampling received data, and on the selection of the phase mode of SSPSCLK (see Section 4.5.3.20 and Figure 50 for a complete description of each sub-mode).

SSPSCLK and SSPTXD are low and SSPSFRM is high for SPI format when the SSPx port is idle or when disabled. SSPSFRM goes low (at the rising edge of the SSPSCLK) and stays low for the remainder of the frame when transmit data is available. The most significant bit of the serial data is driven onto SSPTXD one half-cycle later, and halfway into the first bit period, SSPSCLK asserts high and continues toggling for the remaining data bits. Data transitions on the falling edge of SSPSCLK and is sampled on the rising edge of SSPSCLK. Eight, 16, 18, or 32 bits can be transferred per frame.

With the assertion of SSPSFRM, receive data is driven simultaneously from the peripheral on SSPRXD, most significant bit first. Data transitions on SSPSCLK falling edges, and is sampled by the controller on rising edges. At the end of the frame, SSPSFRM is de-asserted high one clock period (at the rising edge of the SSPSCLK) after the last bit has been latched at its destination, and the completed incoming word is shifted into the “incoming” FIFO. The peripheral can drive SSPRXD to a high-impedance state after sending the last bit of the frame. SSPTXD retains the last value transmitted when the controller goes into Idle mode, unless the enhanced SSPx port is disabled or reset (which forces SSPTXD to zero).

For back-to-back transfers, start and completion are like those of a single transfer, but SSPSFRM does not de-assert between words. Both transmitter and receiver know the word length, and keep track internally of the start and end of words (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 in Motorola SPI format, the enhanced SSPx port can be either a master or a slave device, but the clock and frame direction must be the same. For example, the SSCR1_x[SCLKDIR] and SSCR1_x[SFRMDIR] bits must both be set or both be cleared.

4.4.4.4 Programmable Serial Protocol (PSP) Format

Note

PSP format is supported only for emulation of the I^2S protocol. Operation in PSP mode except as described in *PXA3xx Processor Family I2S Emulation Using SSP/PSP Application Note* is not supported.

The PSP format defines programmable parameters that determine the transfer timings between data samples.

Four serial clock modes are defined in SSPSP_x[SCMODE] (see Table 128). These modes select the SSPSCLKx rising and falling edges for driving data, sampling received data, and the SSPSCLKx idle state.

As shown in Table 115, Table 116, and Table 118, the idle and disable modes of the SSPTXDx, SSPSCLKx, and SSPSFRMx interface signals are programmable using the SSPSP_x[ETDS,SFRMP,SCMODE] bits (see Table 128). When transmit data is ready, SSPSCLKx remains in its Idle state for the number of serial clock (SSPSCLK) periods programmed into the SSPSP_x[STRTDLY]) start delay field.

SSPSCLKx then starts toggling. SSPTXDx remains in the idle state for the number of serial clock periods programmed into the SSPSP_x[DMYSTRT]) dummy start field. SSPSFRMx is asserted after the number of half serial clock periods programmed into the SSPSP_x[SFRMDLY] serial frame-delay field. SSPSFRMx remains asserted for the number of serial clock periods programmed into SSPSP_x[SFRMWDTH] (serial frame-width field), then SSPDFRMx de-asserts.

Serial data of 8, 16, 18, or 32 bits can be transferred per frame by setting SSCR0_x[EDSS] and SSCR0_x[DSS] to the preferred data size select (see Table 121). Once the last bit (LSB) is transferred, SSPSCLKx continues toggling for the number of serial clock periods programmed into the SSPSP_x[DMYSTOP] dummy stop field. SSPTXDx either retains the last bit-value transmitted or is forced to 0, depending on the value programmed into the SSPSP_x[ETDS] end-of-transfer-data state field, when the SSPx port goes into Idle mode, unless the SSPx port is disabled or reset, which forces SSPTXDx to 0.

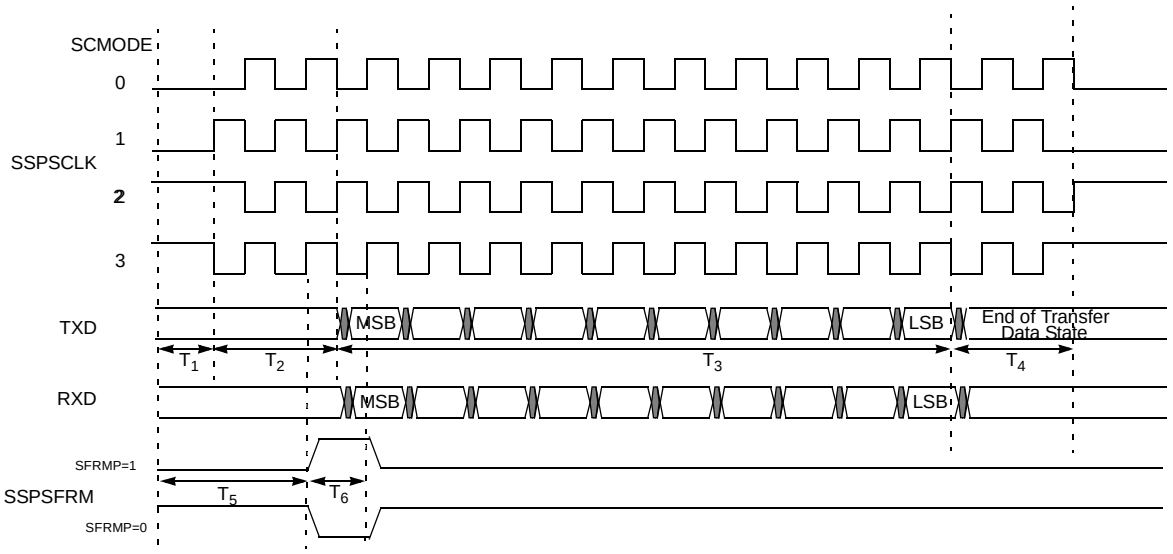
With the assertion of SSPSFRMx, receive data is driven simultaneously from the peripheral onto SSPRXDx, MSB first. Data transitions on the SSPSCLKx edge based on the serial-clock mode that is selected (see SSPSP_x[SCMODE] in Table 128) and is sampled by the SSPx port on the opposite clock edge. When the SSPx port is a master to SSPSFRMx and a slave to SSPSCLKx, at least three extra SSPSCLKs are needed at the beginning and end of each block of transfers to synchronize control signals from the APB clock domain into the SSP clock domain (a block of transfers is a group of back-to-back continuous transfers).

The PSP mode should not be configured as:

sfrmdly/2 + sfrmwidth = dmystrt + dmystop + frame_size & dmystop = 0

This is an obscure case. PSP mode yields an error if configured as mentioned above.

Figure 41: PSP Programmable Serial Protocol Format



Note: If SSPx port is the master of SSPSCLKx (output) and SSPSP_x[ETDS=0], the end-of-transfer-data state for the SSPTXD line is 0. If the SSP is the master of the clock, and the SSPSP[ETDS] bit is set, then the SSPTXD line remains at the last bit transmitted (LSB).

If the SSPx port is a slave to SSPSCLK (input), and modes 1 or 3 are used, then the End of Transfer data state (ETDS) can only change from the LSB if more SSPSCLKs are sent to the SSPx port (for example., dummy stop clocks or slave SSPSCLK is free-running).

Figure 42: PSP Programmable Protocol Format (Consecutive Transfers)

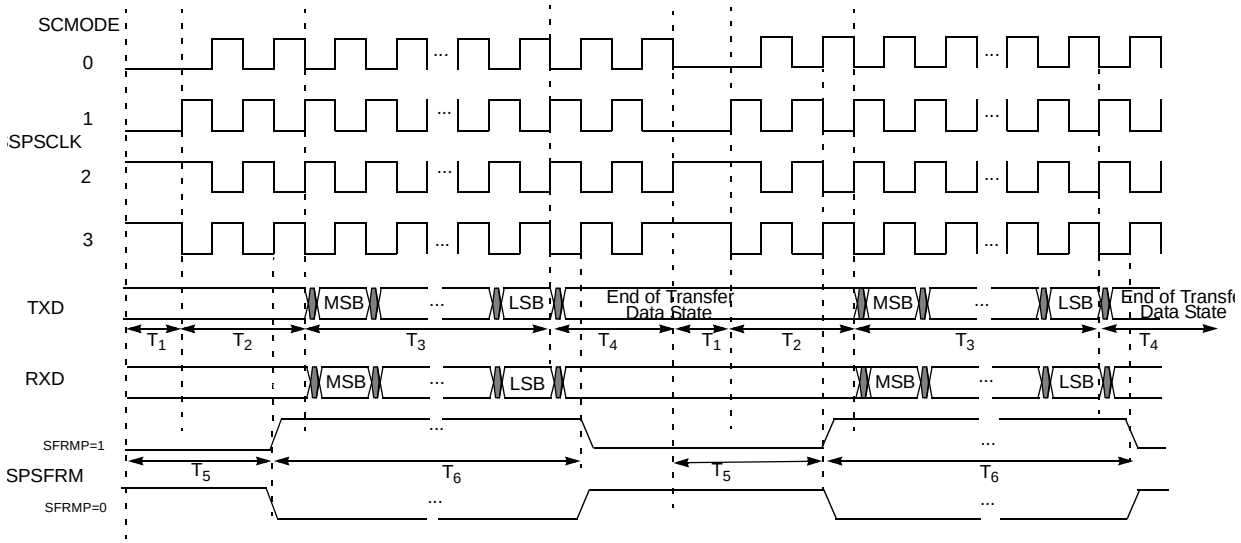


Table 116: Programmable Protocol Parameters

Symbol	Definition	Range	Units
-	Serial clock mode (SSPSP[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[SFRMP])	High or Low	-
T_1	Start delay (SSPSP[STRTDLY])	0 - 7	Clock period
T_2	Dummy start (SSPSP[EDMYSTRT] + SSPSP[DMYSTRT])	0 - 15	Clock period
T_3	Data size (SSCR0[EDSS] and SSCR0[DSS])	4 - 32	Clock period
T_4	Dummy stop (SSPSP[EDMYSTOP] + SSPSP[DMYSTOP])	0 - 31	Clock period
T_5	SSPSFRM delay (SSPSP_x[SFRMDLY])	0 - 127	Half-clock period
T_6	SSPSFRM width (SSPSP_x[SFRMWDTH])	1 - 63	Clock period

Table 116: Programmable Protocol Parameters (Continued)

Symbol	Definition	Range	Units
-	End of transfer data state (SSPSP_x[ETDS])	Low or [bit 0]	-

SSPSFRM delay (T_5) *must not* extend beyond the end of T_4. SSPSFRM width (T_6) must be asserted for at least one SSPSCLK period, and should be de-asserted before the end of T_4 (for example, in terms of time, not bit values, ($T_5 + T_6$) <= ($T_1 + T_2 + T_3 + T_4$), 1<= T_6 < ($T_2 + T_3 + T_4$), and ($T_5 + T_6$) >= (T_1 + 1) to ensure that SSPSFRM is asserted for at least two edges of SSPSCLKx). Program T_1 to 0b0 when SSPSCLKx is enabled by SSPSCLKxEN, or either of the SSCR1_x[ECRA,ECRB] bits. While the PSP can be programmed to generate the assertion of SSPSFRMx during the middle of the data transfer (for example, after the MSB has been sent), the SSPx port is unable to receive data in frame-slave mode (SSPSP_x[SFRMDIR] set, see Table 128) if the assertion of frame is not before the MSB is sent (for example, T_5 <= T_2 if SSCR1_x[SFRMDIR] is set, see Table 122). Transmit data transitions from the end-of-transfer-data state (ETDS) to the next MSB data value upon assertion of the internal version of SSPSFRM. Program SSPSP_x[STRTDLY] = 0x00 whenever SSPSCLKx or SSPSFRMx is configured as an input (for example, SSCR1_x[SCLKDIR] cleared and SSCR1_x[SFRMDIR] cleared, see Table 122).

Note

When the SSPx port is slave to the frame, the sum of T_1+T_2+T_3+T_4 can be less than the actual time from the beginning of the current frame to the beginning of the next frame. For example, when the rate of SSPSCLKx is 13 MHz and the data sample size is 16-bits, the beginning of the frame can occur at a rate of 8 KHz.

4.4.5 High Impedance on SSPTXDx

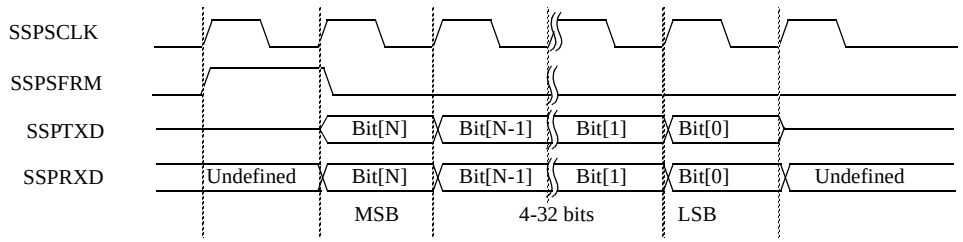
The SSP supports placing the SSPTXDx into high impedance during idle times instead of driving SSPTXDx as controlled SSCR1_x[TTE] and SSCR1_x[TTELP]. SSCR1_x[TTE] enables three-stating on SSPTXDx. The SSCR1_x[TTELP] determines on which SSPSCLK phase SSPTXDx becomes high impedance.

4.4.5.1 TI SSP Format

The enhanced SSPx port behaves as described in Section 4.4.5.3, PSP Format when SSCR1_x[TTE=0].

If SSCR1_x[TTE=1], see Figure 44 for SSPTXDx functionality when SSCR1_x[TTELP=1].

Figure 44: TI SSP with SSCR1_x[TTE=1] and SSCR1_x[TTELP=1]

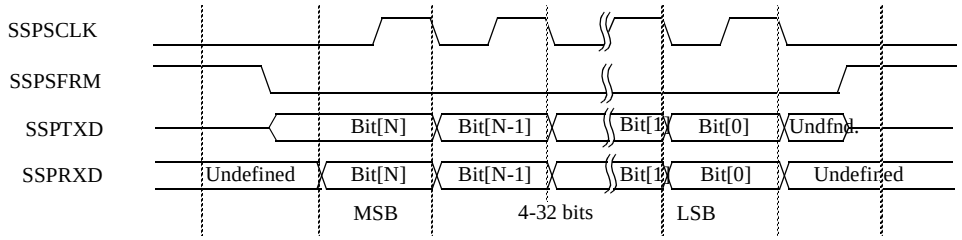


4.4.5.2 Motorola SPI Format

Refer Section 4.4.4.3 when SSCR1[TTE] is cleared.

SSCR1_x[TTELP] must be cleared for Motorola SPI format. See Figure 45 for SSPTXDx functionality if SSCR1_x[TTE=1]. SSPTXDx three-states whenever SSPSFRMx is not asserted for Motorola SPI format.

Figure 45: Motorola SPI with SSCR1_x[TTE=1] and SSCR1_x[TTELP=0]



4.4.5.3 PSP Format

Clear SSCR1_x[TTE] (refer to Section 4.4.4.3) for SSP functionality.

See Figure 46 and Figure 47 for SSPTXDx functionality when SSCR1_x[TTE=1].

Figure 46: PSP Format with SSCR1_x[TTE=1], SSCR1_x[TTELP=0] and SSCR1_x[SFRMDIR=1]

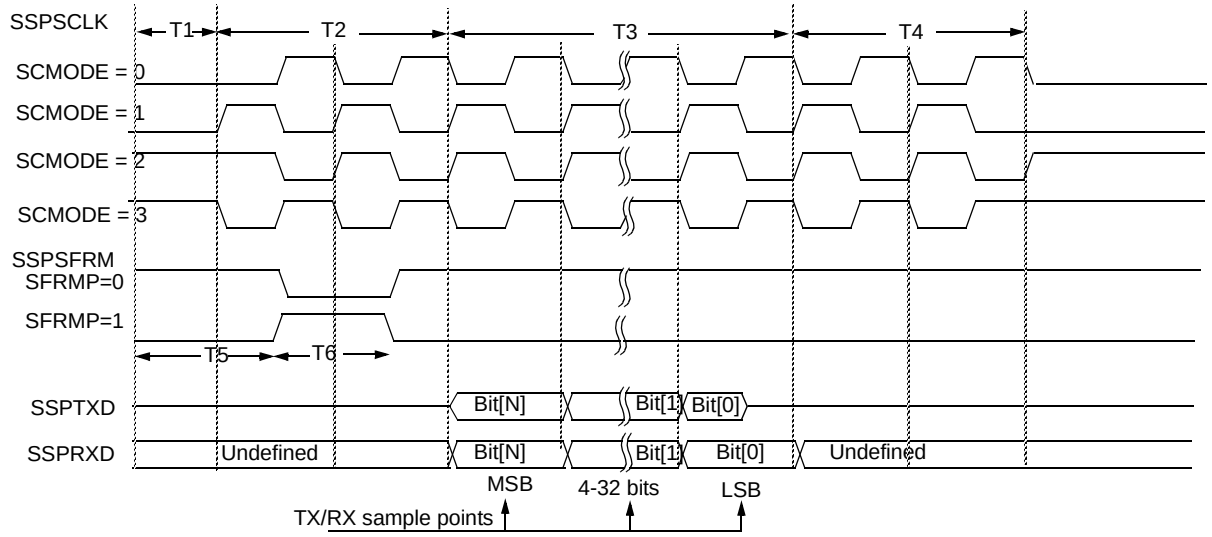
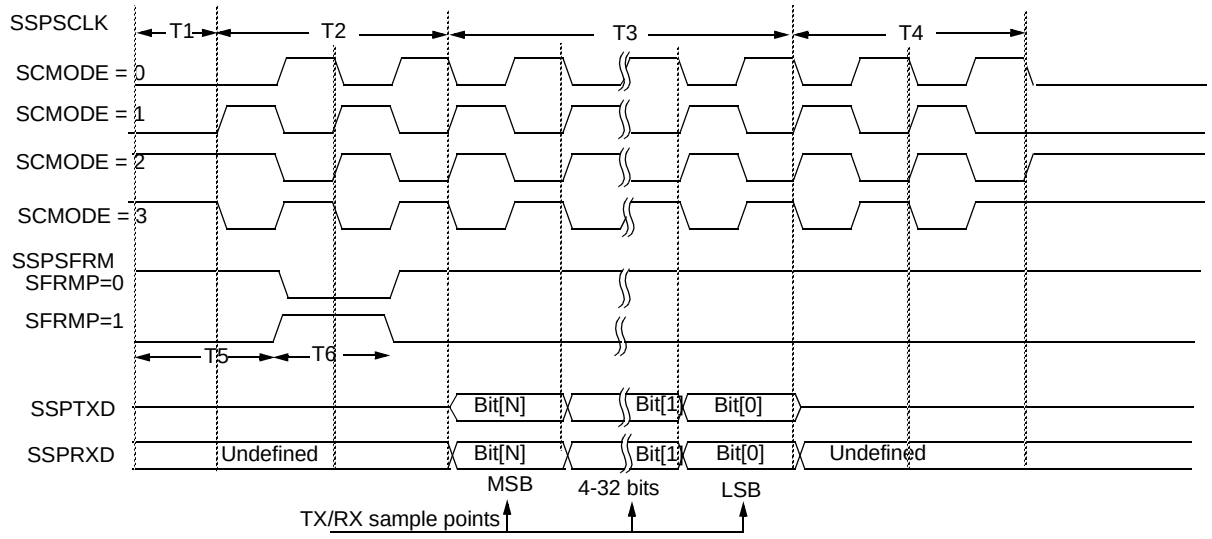


Figure 47: PSP Format with SSCR1_x[TTE=1], and either SSCR1_x[TTELP=1] or SSCR1_x[SFRMDIR=0]



4.4.6 Parallel Data Formats for FIFO Storage

All processor and DMA accesses transfer one FIFO entry per access. Data in the FIFOs is either stored with one 32-bit value per data sample (in non-packed or sample > 16 bits) or in a 16-bit value in packed mode when the data is 8 or 16 bits. Within each 32-bit or 16-bit field, the stored data sample is right-justified, with the least significant bit of the word in bit 0. In the receive FIFO, unused bits are packed as zeroes above the most significant bit. In the transmit FIFO, unused “don’t-care” bits are above the most significant bit. For example, DMA and processor accesses do not have to write to the unused bit locations. Logic in the SSP automatically formats data in the transmit FIFO so that the sample is properly transmitted on SSPTXD in the selected frame format.

4.4.7 FIFO Operation

This section describes the operation of transmit and receive FIFOs.

4.4.7.1 Tavor PV Processor-Initiated Data Transfers

Two separate and independent FIFOs are present for transmitting (TXFIFO to peripheral) and receiving (RXFIFO from peripheral) serial data. The FIFOs are filled or emptied by programmed I/O or DMA bursts.

4.4.7.2 Using Programmed I/O Data Transfers

FIFO filling and emptying can be performed by the processor in response to an interrupt from the FIFO logic. Each FIFO has a programmable FIFO trigger threshold that triggers an interrupt. When the number of entries in the RXFIFO exceeds SSCR1_x[RFT] (see Table 122), an interrupt is generated (if enabled) that signals the processor to empty the RXFIFO. When the number of entries in the TXFIFO is less than or equal to SSCR1_x[TFT] plus 1, an interrupt is generated (if enabled) that signals the processor to refill the TXFIFO.

The SSP Status register (see Section 4.5.3.1) can be polled to determine how many samples are in a FIFO, and whether the FIFO is full or empty. Software ensures that the proper SSCR1_x[RFT,TFT] values are chosen to prevent SSSR_x[ROR,TUR] error conditions (see Table 124).

4.4.7.3 Using DMA Data Transfers

The DMA controller can also be programmed to transfer data to and from the FIFOs. Be careful when setting the FIFO trigger threshold levels by setting the correct DMA burst sizes. Such attention prevents overruns of the TXFIFO or underruns of the RXFIFO when using the DMA. TXFIFO overruns and RXFIFO underruns are silent errors. There is no indication of the overrun or underrun condition other than missing data at the receiving end of the link. The DMA burst size must be smaller than the trigger threshold.

The programming model for using DMA is:

- Program the total number of Transmit/Receive byte lengths, burst sizes, and peripheral width.
- When not using the FIFO packed mode, program DCMD_x[Width] to 0b01 for FIFO data sizes of 8 bits, 0b10 for FIFO data sizes of 16 bits, and 0b11 for FIFO data sizes of more than 16 bits. The SSP stores one data sample per FIFO location where each FIFO has 16 locations when not using packed mode. For example, the DMA burst size (DCMD_x[Size]) must not exceed 16 bytes when DCMD_x[Width] is set to 0b01 (byte wide).
- When using FIFO packed mode (see Section 4.4.1.1), program DCMD_x[Width] to 0b11. The SSP stores two data samples per FIFO location where each FIFO has 16 locations when using packed mode. Therefore, the DMA burst size (DCMD_x[Size]) must not exceed 16 bytes when DCMD_x[Width] is set to 0b11 (more than 16 bits wide).
- Because the SSP is not flow-controlled and it has only 16 location FIFOs, software must program the TXFIFO threshold (SSCR1_x[TFT], see Table 122), RXFIFO threshold (SSCR1_x[RFT]), and the DMA burst size to ensure that a TXFIFO overrun or RXFIFO underrun does not occur. Software must also ensure that the SSP DMA requests are properly prioritized in the system to prevent overruns and underruns.
- Program the preferred values into the SSP Control registers (see Table 121 and Table 122).
- Enable the SSP by setting SSCR0_x[SSE].
- Set the run bits in the DMA Command register.
- The DMA waits for either the TXFIFO or RXFIFO service request.
- If the receive byte length is not an even multiple of the transfer burst size, a trailing- byte condition may occur as described in Section 4.4.2.

The DMA channel should be set up to transmit and receive the same number of bytes in full-duplex formats where the SSP always receives the same number of data samples that it transmits.

Table 117: SSCR1_x[TFT, RFT] Values with DMA Burst Sizes

SSP Sample Size (bits)	DMA Width DCMD[WIDTH]	Burst Size	TFT TX FIFO Trigger Threshold	RFT RX FIFO Trigger Threshold
8	1 byte WIDTH=0b01	8	7	8
16, 18 or 32	4 bytes WIDTH=0b11	16	3	4
16, 18 or 32	4 bytes WIDTH=0b11	32	7	8

The DMA transmit-burst size is limited because the smallest TFT value is zero, which equates to one sample left in the TXFIFO. Software must program the TXFIFO threshold and the DMA burst size (DCMD[Size]) such that an overflow of the TXFIFO does not occur. Software must program the RXFIFO threshold and the DMA burst size such that an underflow of the RX FIFO does not occur.

Note

When the FIFO packing-enable bit, SSCR0_x[FPCKE] is set to 0b1, the SSCR1_x[TFT,RFT] fields represent twice the number of FIFO entries as when the packing enable bit is 0b0. When in packed mode, the maximum allowed DMA burst size (8 or 16) in Table 118 could be doubled.

4.4.8 Baud-Rate Generation

When the SSP is configured as the master (output) of SSPSCLKx as determined by clearing SSCR1_x[SCLKDIR], see Table 122. The baud rate (or serial bit-rate clock SSPSCLKx) is generated internally by dividing the on-chip clock (13 MHz), the network clock (CLK_EXT, described in the GPIO chapter), or the SSP external clock (SSPEXTCLKx) by a programmable divider (SSCR0_x[SCR], see Table 121) to generate baud rates up to 13 Mbps. Additionally, if audio clock select SSCR0_x[ACS] is set, then the frequency of the clock sent to the SSP is determined by the Audio Clock Divider register (see Section 4.5.12.4 and the Audio Clock Dither Divider register (see Section 4.5.13). This audio clock can also be changed by the baud-rate divider in the SSP, but phase relationship with SSPSYSCLKx is lost.

Table 119 shows the clock selected to drive the baud-rate generator by the external clock select SSCR0_x[ECS], audio clock select SSCR0_x[ACS], and network clock select SSCR1_x[NCS] bits (see Section 4.5.2.1). When the clock source is to be changed, software must:

- Disable the SSPx port by writing 0b0 to SSCR0_x[SSE].
- Disable the SSP port internal clock by clearing the appropriate bit in the Clock Enable Register. See the Clocks and Power Management chapters for more information.
- Write the SSCR0_x[ECS,ACS,NCS] bits.
- Write 0b1 to the appropriate bit in the CKEN register to re-enable the SSPx port internal clock.
- Set SSCR0_x[SSE] to re-enable the SSPx port.

Whenever the baud rate is to be changed, software must:

- Disable the SSPx port by clearing SSCR0_x[SSE].
- Write a new value to SSCR0_x[SCR].
- Set SSCR0_x[SSE] to re-enable the SSPx port.
- Wait 2 SSPSCLK cycles before writing new data to the TXFIFO.

For more information, see the Clocks and Power Management chapters in *PXA3xx Processor Family Vol. I: System and Timer Configuration Developers Manual*.

Table 119: SSP Clock Selection

SSCR0_x [ACS]	SSCR0_x [ECS]	SSCR0_x [NCS]	SSCR0_x [52mm]	Selected Clock
0	0	0	X	On-chip clock (internal 13-MHz PLL)

Table 119: SSP Clock Selection (Continued)

SSCR0_x [ACS]	SSCR0_x [ECS]	SSCR0_x [NCS]	SSCR0_x [52mm]	Selected Clock
0	X	1	X	Network clock (CLK_EXT)
0	1	0	X	SSP external clock (SSPEXTCLKx)
1	X	X	X	On-chip audio clock (determined from internal PLL, Audio Clock Divider register and Audio Clock Dither Divider register values)
0	0	0	1	On-chip clock (internal 26-MHz PLL)

4.5 Register Descriptions

Each SSPx consists of twelve registers: seven control, one data, two status, one time-out, and one test.

- The SSP Control registers SSCR0_x (see Table 121) SSCR1_x (see Table 122) specify the baud rate, data length, frame format, data-transfer mechanism, and port enabling. In addition, they permit setting the FIFO trigger thresholds that trigger an interrupt.
- All registers must be accessed using aligned words.
- In SSCR0_x (see Table 121), DSS, FRF, EDSS, ECS, NCS, ACS and FPCKE must be modified before SSE is set to 0b1. All bits in SSCR1_x (see Table 122) must be written before the SSCR0-x[SSE] is set. Any writable bits in the SSSR_x (see Table 124), SSTO_x (see Table 127), and SSITR_x (see Table 125) registers can be modified at any time. All bits in the SSPSP_x (see Table 128) and SSACD_x (see Table 132) registers must be written before the SSCR0_x[SSE] is set. All bits in the SSRSA_x (see Table 130) and SSTSA_x (see Table 129) register should be written before the SSCR0_x[SSE] bit is set.
- The Timeout register SSTO_x (see Table 127) programs the timeout value that signals a specified period of RXFIFO inactivity.
- The SSPx port Programmable Serial Protocol SSPSP_x (see Table 128) register programs the parameters used to define PSP format data transfer.

Note

PSP format is only supported for emulation of the I^2S protocol. Operation in PSP mode except as described in *PXA3xx Processor Family I2S Emulation Using SSP/PSP Application Note* is not supported.

- The Data register SSDR_x (see Table 126) is mapped as one 32-bit location, which physically points to either of two 32-bit registers: one register is for Writes of data transfers to the TXFIFO, while the other register is for Reads that take data from the RXFIFO. A processor Write cycle or burst DMA Write cycle loads successive words into the SSDR_x Write register and then into the TXFIFO. A processor-Read cycle or burst-DMA Read cycle takes data from the SSDR_x Read register, and the RXFIFO reloads it with available data samples when available.

Read and Write DMA bursts should not increment the address of the SSDR_x register; all accesses to the SSDR_x register address target either the DDSR_x Read register or the SSDR_x Write register. The TXFIFO and RXFIFO are independent buffers that allow full-duplex operation.

- Besides showing the state of the FIFO buffers, the Status register SSSR_x (see Table 124) shows whether the programmable FIFO trigger thresholds has been passed and whether a transmit or receive FIFO service request is active. The SSSR_x register also shows the actual “fullness” of either FIFO. Flag bits in the SSSR_x register indicate if the SSPx port is actively transmitting data, if the TXFIFO is not full, and if the RXFIFO is not empty. The SSSR_x[ROR] bit signals an overrun of the RXFIFO, in which case newly received data would be discarded, see Section 4.5.4.10.

When programming registers, reserved bits must be written as 0b0, and they read as undefined.

4.5.1 Register Summary

Table 120 shows the SSPx Port registers and their physical addresses. The base addresses of the SSPx ports are as follows:

SSP1 = 0x4100_0000, SSP2 = 0x4170_0000, SSP3 = 0x4190_0000, SSP4 = 0x41A0_0000

Table 120: SSP Register Summary

Physical Address	Description	Page
SSP Controller 1 (SSP1)		
0x4100_0000	SSP Control Registers 0 (SSCR0_x)	page 301
0x4100_0004	Mode Select (MOD)	page 305
0x4100_0008	TXD Three-State Enable On Last Phase (TTELP)	page 312
0x4100_000C	SSP Interrupt Test Registers (SSITR_x)	page 327
0x4100_0010	Test RXFIFO Overrun (TROR)	page 328
0x4100_0014 – 0x4100_0024	Reserved	
0x4100_0028	SSP Time Out Registers (SSTO_x)	page 329
0x4100_002C	SSP Programmable Serial Protocol Registers (SSPSP_x)	page 330
0x4100_0030	Extended Dummy Stop (EDMYSTOP)	page 332
0x4100_0034	SSP RX Time Slot Active Registers (SSRSA_x)	page 335
0x4100_0038	SSP Time Slot Status Registers (SSTSS_x)	page 336
0x4100_003C	Network Mode Busy (NMBSY)	page 336
0x4100_0040	SSP Audio Clock Dither Divider Registers (SSACDD_x)	page 342
0x4100_0044 – 0x410F FFFF	Reserved	

Table 120: SSP Register Summary (Continued)

Physical Address	Description	Page
SSP Controller 2 (SSP2)		
0x4170_0000	SSP Control Registers 0 (SSCR0_x)	page 301
0x4170_0004	Mode Select (MOD)	page 305
0x4170_0008	TXD Three-State Enable On Last Phase (TTELP)	page 312
0x4170_000C	SSP Interrupt Test Registers (SSITR_x)	page 327
0x4170_0010	Test RXFIFO Overrun (TROR)	page 328
0x4170_0014 – 0x4170_0024	Reserved	
0x4170_0028	SSP Time Out Registers (SSTO_x)	page 329
0x4170_002C	SSP Programmable Serial Protocol Registers (SSPSP_x)	page 330
0x4170_0030	Extended Dummy Stop (EDMYSTOP)	page 332
0x4170_0034	SSP RX Time Slot Active Registers (SSRSA_x)	page 335
0x4170_0038	SSP Time Slot Status Registers (SSTSS_x)	page 336
0x4170_003C	Network Mode Busy (NMBSY)	page 336
0x4170_0040	SSP Audio Clock Dither Divider Registers (SSACDD_x)	page 342
0x4170_0044 – 0x417F FFFF	Reserved	
SSP Controller 3 (SSP3)		
0x4190_0000	SSP Control Registers 0 (SSCR0_x)	page 301
0x4190_0004	Mode Select (MOD)	page 305
0x4190_0008	TXD Three-State Enable On Last Phase (TTELP)	page 312
0x4190_000C	SSP Interrupt Test Registers (SSITR_x)	page 327
0x4190_0010	Test RXFIFO Overrun (TROR)	page 328
0x4190_0014 – 0x4190_0024	Reserved	
0x4190_0028	SSP Time Out Registers (SSTO_x)	page 329
0x4190_002C	SSP Programmable Serial Protocol Registers (SSPSP_x)	page 330
0x4190_0030	Extended Dummy Stop (EDMYSTOP)	page 332
0x4190_0034	SSP RX Time Slot Active Registers (SSRSA_x)	page 335

Table 120: SSP Register Summary (Continued)

Physical Address	Description	Page
0x4190_0038	SSP Time Slot Status Registers (SSTSS_x)	page 336
0x4190_003C	Network Mode Busy (NMBSY)	page 336
0x4190_0040	SSP Audio Clock Dither Divider Registers (SSACDD_x)	page 342
0x4190_0044 – 0x419F FFFF	Reserved	
SSP Controller 4 (SSP4)		
0x41A0_0000	SSP Control Registers 0 (SSCR0_x)	page 301
0x41A0_0004	Mode Select (MOD)	page 305
0x41A0_0008	TXD Three-State Enable On Last Phase (TTELP)	page 312
0x41A0_000C	SSP Interrupt Test Registers (SSITR_x)	page 327
0x41A0_0010	Test RXFIFO Overrun (TROR)	page 328
0x41A0_0014 – 0x41A0_0024	Reserved	
0x41A0_0028	SSP Time Out Registers (SSTO_x)	page 329
0x41A0_002C	SSP Programmable Serial Protocol Registers (SSPSP_x)	page 330
0x41A0_0030	Extended Dummy Stop (EDMYSTOP)	page 332
0x41A0_0034	SSP RX Time Slot Active Registers (SSRSA_x)	page 335
0x41A0_0038	SSP Time Slot Status Registers (SSTSS_x)	page 336
0x41A0_003C	Network Mode Busy (NMBSY)	page 336
0x41A0_0040	SSP Audio Clock Dither Divider Registers (SSACDD_x)	page 342
0x41A0_0044 – 0x41AF FFFF	Reserved	

4.5.2 SSP Control Registers 0 (SSCR0_x)

The SSP Port Control 0 registers (SSCR0_1, SSCR0_2, SSCR0_3 and SSCR0_4) contain 13 different bit fields that control various functions within the SSP port. Table 121 shows the bit locations corresponding to the different control bit fields within SSCR0_x. The reset state of all bits are as shown, but they must be programmed to their preferred values before enabling the SSP port.

Note

When the data size is programmed to be less than 32 bits, right-justify the data before loading it into the TXFIFO.

Writes to reserved bits must be 0b0, and read values of reserved bits are undefined.

Table 121: SSCR0_x Bit Definitions

Physical Address	SSPx
0x4100_0000 (SSP1)	SSCR0_1
0x4170_0000 (SSP2)	SSCR0_2
0x4190_0000 (SSP3)	SSCR0_3
0x41A0_0000 (SSP4)	SSCR0_4

User Settings																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	MOD	ACS	FPCKE	Reserved	Reserved	FRDC			TIM	RIM	NCS	EDSS	SCR												SSE	ECS	FRF		DSS			
PXA31x Only	MOD	ACS	FPCKE	Reserved	52MM	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	0	0

Bits	Access	Name	Description
31	R/W	MOD	Mode: 0 = Normal SSP mode 1 = Network mode
30	R/W	ACS	Audio Clock Select: 0 = Clock selection is determined by the NCS and ECS bits 1 = Audio clock (and audio clock divider) are used to create the SSP's serial bit clock (SSPSCLKx)
29	R/W	FPCKE	FIFO Packing Enable: 0 = FIFO packing mode disabled 1 = FIFO packing mode enabled
28	—	—	Reserved (must be written as 0)
27	R/W	52MM	52 MHz Mode 0 = 13 Mbps Mode 1 = 26 Mbps Mode **NOTE:** PXA31x Only
	—	—	Reserved (PXA32x and PXA30x Only)
26:24	R/W	FRDC	Frame Rate Divider Control: Value of 0b000-111 specifies the number of time slots per frame when in network mode (the actual number of time slots is FRDC +1, so 1 to 8 time slots can be specified).
23	R/W	TIM	Transmit FIFO Underrun Interrupt Mask: 0 = TUR events generate an SSP interrupt 1 = TUR events do NOT generate an SSP interrupt

Table 121: SSCR0_x Bit Definitions (Continued)

Physical Address		SSPx
0x4100_0000 (SSP1)	SSCR0_1	
0x4170_0000 (SSP2)	SSCR0_2	
0x4190_0000 (SSP3)	SSCR0_3	
0x41A0_0000 (SSP4)	SSCR0_4	

	31	30	29	28	27	26	25	24	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
PXA32x and PXA30x Only	MOD	ACS	FPCKE	Reserved	Reserved	FRDC			TIM	RIM	NCS	EDSS	SCR												SSE	ECS	FRF		DSS			
PXA31x Only	MOD	ACS	FPCKE	Reserved	52MM	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	0	0

Bits	Access	Name	Description
22	R/W	RIM	Receive FIFO Overrun Interrupt Mask: 0 = ROR events generate an SSP interrupt 1 = ROR events do NOT generate an SSP interrupt
21	R/W	NCS	Network Clock Select: 0 = Clock selection is determined by ECS bit. 1 = Network clock is used to create the SSP's serial clock (SSPSCLKx)
20	R/W	EDSS	Extended Data Size Select: 0 = A 0b0 is pre-appended to the DSS value to set the DSS range from 8 to 16 bits. 1 = A 0b1 is pre-appended to the DSS value to set the DSS range from 18 to 32 bits.
19:8	R/W	SCR	Serial Clock Rate: Values 0x000, 0x001, 0x00F, and 0x07F are used to generate the serial bit rate of SSPSCLKx. Serial bit rate = SSPx Clock / (SCR + 1), where SCR is one of the above hex values converted to decimal.
7	R/W	SSE	Synchronous Serial Port Enable: 0 = SSPx port is disabled 1 = SSPx port is enabled
6	R/W	ECS	External Clock Select: 0 = On-chip SSPx Clock is used in the serial clock rate formula (above) to produce the SSPx port's SSPSCLKx 1 = SSPEXTCLKx (GPIO pin) is used in the serial clock rate formula to produce the SSPx port's SSPSCLKx

Table 121: SSCR0_x Bit Definitions (Continued)

Physical Address		SSPx
0x4100_0000 (SSP1)	SSCR0_1	
0x4170_0000 (SSP2)	SSCR0_2	
0x4190_0000 (SSP3)	SSCR0_3	
0x41A0_0000 (SSP4)	SSCR0_4	

User Settings																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PXA32x and PXA30x Only	MOD	ACS	FPCKE	Reserved	Reserved	FRDC			TIM	RIM	NCS	EDSS	SCR												SSE	ECS	FRF		DSS			
PXA31x Only	MOD	ACS	FPCKE	Reserved	52MM	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	0	0

Bits	Access	Name	Description					
5:4	R/W	FRF	Frame Format: This field must be written with 0b11 to select the PSP format. 00 – Motorola* Serial Peripheral Interface (SPI) 01 – Texas Instruments* Synchronous Serial Protocol (SSP) 10 – **Reserved** 11 – Programmable Serial Protocol (PSP) PSP format is only supported for emulation of the I^2S protocol. Operation in PSP mode except as described in *PXA3xx Processor Family I2S Emulation Using SSP/PSP Application Note is not supported*					
3:0	R/W	DSS	Data Size Select:					
			EDSS	DSS	Data Size	EDSS	DSS	Data Size
			1	0000	**Reserved**, undefined	0	0000	**Reserved**, undefined
			1	0001	18-bit data	0	0001	**Reserved**, undefined
			1	0010	**Reserved**, undefined	0	0010	**Reserved**, undefined
			1	0011	**Reserved**, undefined	0	0011	**Reserved**, undefined
			1	0100	**Reserved**, undefined	0	0100	**Reserved**, undefined
			1	0101	**Reserved**, undefined	0	0101	**Reserved**, undefined
			1	0110	**Reserved**, undefined	0	0110	**Reserved**, undefined
			1	0111	**Reserved**, undefined	0	0111	8-bit data
			1	1000	**Reserved**, undefined	0	1000	**Reserved**, undefined
			1	1001	**Reserved**, undefined	0	1001	**Reserved**, undefined
			1	1010	**Reserved**, undefined	0	1010	**Reserved**, undefined
			1	1011	**Reserved**, undefined	0	1011	**Reserved**, undefined
			1	1100	**Reserved**, undefined	0	1100	**Reserved**, undefined
			1	1101	**Reserved**, undefined	0	1101	**Reserved**, undefined
			1	1110	**Reserved**, undefined	0	1110	**Reserved**, undefined
			1	1111	32-bit data	0	1111	16-bit data

4.5.2.1 Mode Select (MOD)

The MOD bit selects between normal and network modes. Use network mode in systems where several devices are connected to the same SSPTXDx, SSPRXDx, SSPSFRMx and SSPSCLKx interface signals. In network mode, from 1-8 time slots (see Section 4.5.2.5) can be specified. The SSPx port can transmit or receive in any of the time slots (see Section Table 131: and Section Table 131:). Software should set MOD only when using the PSP (see Figure 48) format. Setting MOD causes the SSPSCLKx to run continuously, if the SSP port is a master of the clock (SSCR1_x[SCLKDIR] cleared, see Table 122).

Note

In PSP format, the interface signal SSPSFRMx can be delayed by a programmable value SSPSP_x[SFRMDLY], see Table 128. The SPSFRMx interface signal can assert with internal frame sync or some SFRMDLY delay after assertion of internal frame sync. PSP format is supported only for emulation of the I^2S protocol. Operation in PSP mode except as described in *PXA3xx Processor Family I2S Emulation Using SSP/PSP Application Note* is not supported.

When in network mode, only one SSPSFRMx is sent (master mode) or received (slave mode) for the number of time slots programmed into the SSCR0_x[FRDC] field. When beginning in network mode, while the SSPx port is a master to the SSPSFRMx interface signal, the first SSPSFRMx signal does not occur until after data is in the TXFIFO. After assertion of the first SSPSFRMx signal, if the SSP is a master to SSPSFRMx, subsequent SSPSFRMx signals continue to assert regardless of whether data resides in the TXFIFO. Therefore, the transmit underrun bit SSSR_x[TUR] (see Table 124) is set to 0b1 if there is no data in the TXFIFO and the SSP port is programmed to drive SSPTXDx data in the current time slot, even if the SSPx port is master to SSPSFRMx. When using PSP format in network mode, the parameters SFRMDLY, STRTDLY, DMYSTP, DMYSTRT must all be 0b0. The other parameters SFRMP, SCMODE, FSRT, SFRMWDTH are programmable.

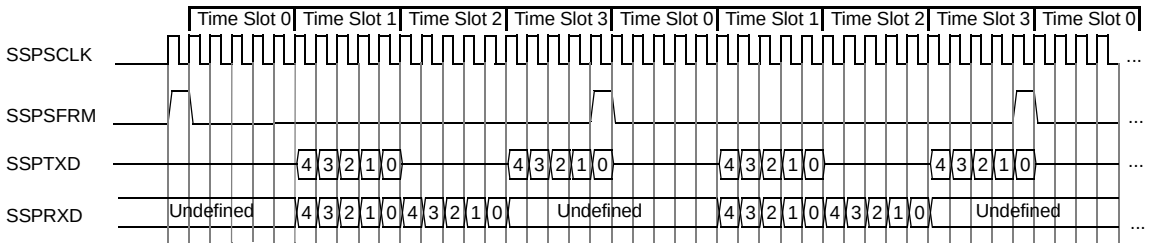
When the SSP port is a master to the SSPSFRMx signal and a need arises to exit from network mode, software should:

- Clear MOD. SSCR0_x[SSE] does not need to change.
- Wait until SSTSS_x[NMBSY] (see Table 131) is cleared.
- Disable the SSP port by clearing SSCR0_x[SSE]).
- Before exiting network mode, verify the TXFIFO is empty (SSSR_x[TFL]=0b0000 and SSSR_x[TNF]=0b1.)
- If data remains in the TXFIFO after the network mode is exited, a non-network mode frame will be sent.

Due to synchronization delay between the internal bus and the SSPx port clock domain, one extra frame may be transmitted after software clears MOD. The SSP port continues to drive SSPSCLKs (if SSCR1_x[SCLKDIR] is cleared, see Table 122) or SSPSFRMx (if SSCR1_x[SFMRDIR] is cleared) until the end of the last valid time slot.

If the SSP port is a slave to both SSPSCLKx (SSCR1_x[SCLKDIR] set) and SSPSFRMx (SSCR1_x[SFMRDIR] set), NMBSY remains asserted until MOD is cleared or until one SSPSCLKx after the end of the last valid time slot.

Figure 48: Network Mode (Example Using 4 Time Slots)



Note: This example has 5 bits of data per sample. The TXD three-state enable bit and the TXD three-state enable on last phase bit are both 0b1. The SSP is a master of SSPSCLK and SSPSFRM. The SSP has been programmed for 4 time slots (SSCR0_x[FRDC=0b11]), TX Time Slot Active register is programmed to 0x0000_000A (SSTSA[TTSA] = b00001010), and the RX Time Slot Active register has been set to 0x0000_0006 (SSRSA[RTSA] = b00000110).

4.5.2.2 Audio Clock Select (ACS)

When the ACS bit is set to 0b1, the rate of SSPSCLKx is determined by the audio clock divider (ACDS) and audio clock PLL select (ACPS) fields of the SSACD_x register (see Figure 132) and by the fields of the Audio Clock Dither Divider register (see Figure). If the ACS bit is 0b0, the NCS and ECS bits determine the clock source (see Figure 119) that is divided by the SCR field to determine the rate of SSPSCLKx. If ACS is set (and the GPIOs are properly configured), SSPSYSCLKx is output continually, even if the SSP port is disabled. SSPSCLKx is output as determined by the SSCR0_x[FRF] format, the value of the SSPSCLKxEN interface signal, and SSCR1_x[ECRA] and SSCR1_x[ECRB].

Note

When the processor is run in the Ring Oscillator mode, the 624 MHz oscillator is disabled and therefore the SSCR0_x[ACS] should be cleared because the proper audio clock frequencies cannot be produced.

4.5.2.3 FIFO Packing Enable (FPCKE)

When the FIFO packing is enabled, SSCR0_x[FPCKE=1], the sample size of the TXFIFO and RXFIFO doubles from 16 to 32 samples, which is 2 samples wide by 16 locations deep, with a maximum data sample size of 16 bits. If the extended data-size select (SSCR0_x[EDSS]) bit is set to 0b1, then the SSCR0_x[FPCKE] bit is ignored and the FIFO sizes remain 16 samples, which is one sample wide by 16 locations deep. If the SSCR0_x[FPCKE] bit is set to 0b1 while SSCR0_x[EDSS] is cleared, then the FIFO threshold values SSCR1_x[TFT] and SSCR1_x[RFT] (see Figure 122) are effectively doubled and the FIFO fill levels SSSR_x[TFL] and SSSR_x[RFL] (see Figure 124) become 2x to 2x - 2 times their previous values.

Note

Special conditions when FIFO packing is enabled:

- The SSSR_x[TNF] (TXFIFO Not Full) bit is not valid. The TXFIFO full condition occurs when SSSR_x[TFL=15], meaning there is one less usable line in the TXFIFO.
- The amount of data in the RXFIFO is read from SSSR_x[RFL] and SSSR_x[RNE], but must be qualfied by the SSSR_x[OSS] bit. SSSR_x[OSS] specifies whether there are one or two samples in the RXFIFO.
- The SSSR[OSS] bit indicates whether an even or odd number of samples have been stored in the RXFIFO. The software must poll this bit to determine if the last entry in the RXFIFO has one or two valid entries.
- DMA usage is recommended in Packed mode but the burst size must be less than or equal to the threshold.
- When using SSP Packed mode, the blocks of data transmitted must be of even length.

4.5.2.4 52MM (52 Mbps Mode)

Note

This mode is for use in the PXA31x processor only. This bit is used for software compatibility with the SSP controllers of earlier PXA3xx family products by configuring the baud rates supported of the SSPs. When this bit is set, the SSP operates in 52 Mbps mode - with a 52 MHz clock feeding the pre-scaler that generates the SSP clock. When this bit is set, the resulting SSP clock operates at 26 MHz.

When this bit is clear, the SSP operates in a 13 Mbps mode with the 13-MHz clock feeding the prescaler to generate the SSP clock, which is the same as previous PXA3xx family products.

For PXA30x processors, this bit is reserved and must be written as zero.

4.5.2.5 Frame Rate Divider Control (FRDC)

The FRDC field indicates the number of time slots used when in network mode (SSCR0[MOD] set). The SSPx port supports from one to eight time slots. A value of 0b000 indicates one time slot, and a value of 0b111 indicates eight time slots.

4.5.2.6 Transmit FIFO Underrun Interrupt Mask (TIM)

When set to 0b1, the TIM bit masks the TXFIFO underrun event (SSSR_x[TUR], see Figure 124) from generating an SSPx port interrupt even though SSSR_x[TUR] still indicates that a TUR event has occurred. The TIM bit can be written to at any time, before or after SSP port is enabled.

4.5.2.7 RXFIFO Overrun Interrupt Mask (RIM)

When set to 0b1, the RIM bit masks the RXFIFO overrun event (SSSR_x[ROR], see Figure 124) from generating an SSPx port interrupt even though SSSR_x[ROR] still indicates that an ROR event has occurred. The RIM bit can be written to at any time, before or after SSP is enabled.

4.5.2.8 Network Clock Select (NCS)

When ACS is cleared, use NCS in conjunction with ECS to select the source clock (see Table 119) that is divided by SCR + 1 to determine the rate of SSPSCLKx, when SSCR1_x[SCLKDIR] is cleared.

4.5.2.9 Extended Data Size Select (EDSS)

Use the EDSS bit in conjunction with the DSS field to select the size of the data sample that is transmitted and received by the SSPx port.

4.5.2.10 Serial Clock Rate (SCR)

The 12-bit serial SCR bit field is used in the formula (source clock)/(SCR + 1) to determine the bit rate of SSPSCLKx when SSPCLKx is an output (master) and controlled by SSCR1_x[SCLKDIR] cleared (see Table 122). The serial-clock generator uses one of three source clocks as selected by the SSCR0_x[ECS] and SSCR0_x[NCS]. Many different SSPSCLKx bit rates can be selected, where the maximum bit rate is 13 Mbps. To change the baud rate, software must:

- Disable the SSPx port by writing 0b0 to SSCR0_x[SSE].
- Write a new value to SSCR0_x[SCR].
- Write 0b1 to SSCR0_x[SSE] to re-enable the SSPx port.
- Wait 2 SSPSCLK cycles before writing new data to the TXFIFO.

SSPCLKx is used by the SSPx port transmit logic to drive data onto the SSPTXDx interface signal (output), and to latch data on the SSPRXDx interface signal (input). 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. Software should not change the SCR bits, when SSPSCLKx is enabled by the SSPSCLKxEN interface (input) signal, or change SSPCR1_x[ECRA] and SSPCR1_x[ECRB], as this causes an immediate change of the rate of SPSCLKx.

When the SSP port is a slave with respect to SSPSCLKx (SSPSCLKx is an input as defined by SSCR1_x[SCLKDIR] set), the SCR field is ignored because transmission and reception data rates are determined by an external clock driving the SSPSCLKx input.

4.5.2.11 Synchronous Serial Port Enable (SSE)

The SSE bit enables and disables all SSPx port operations:

- SSCR0_x[SSE] cleared—The SSPx port is disabled
- SSCR0_x[SSE] set—The SSPx port is enabled.

When the SSPx port is disabled, all of its clocks can be stopped to minimize power consumption (see the CKEN register in the Clocks and Power Management chapters). On reset, the SSPx port is disabled.

Clearing the SSCR0_x[SSE] during active operation has the following effects:

- SSPx port is disabled immediately
- Current transmit or receive frame is terminated
- Resets the SSPx port FIFOs and SSSR_x status bits (the SSPx ports control registers SSCR0_x and SSCR1_x are not reset).

Note

After reset or after clearing SSCR0_x[SSE], ensure the SSCR0_x, SSCR1_x, SSITR_x, SSTO_x, and SSPSP_x registers are properly re-configured, and reset the SSSR_x register before re-enabling the SSPx port by setting SSCR0_x[SSE]. Other bits in SSCR0_x can be written at the same time as SSCR0_x[SSE].

When any SSPx port is disabled, its interface pins can be configured for use as other alternate functions or used as GPIOs.

4.5.2.12 External Clock Select (ECS)

When ACS is cleared, use ECS in conjunction with NCS to select which source clock (see Table 119) is divided by SCR + 1 to determine the rate of SSPSCLKx (SSCR1_x[SCLKDIR] cleared).

When ECS and ACS are both cleared, the SSPEXTCLKx input is treated as SSPSCLKxEN to gate the SSPSCLKx output, when SSCR1_x[SCLKDIR] is cleared. See Figure 49. When the SSPSCLKxEN input signal changes, there is a 1-2 clock lag before the SSPSCLKx is started or stopped, due to internal synchronization to the SSPSCLKxEN signal.

The SSP port is a slave when SSCR1_x[SCLKDIR] is set. So, the SSP port receives SSPSCLKx from an external clock source. In this case, ECS is ignored and serial data rates are determined by the external clock source.

Note

Clear SSE to disable the SSP port before setting ECS or NCS. Then program ECS or NCS before, or at the same time, that SSE is re-enabled to prevent glitches on the SSPSCLKx and SSPTXDx interface signals.

4.5.2.13 Frame Format (FRF)

The FRF bit field selects which frame format to use: Motorola* SPI (FRF = 0b00), Texas Instruments* SSP (FRF = 0b01), or the Programmable Serial Protocol (PSP) format (FRF = 0b11).

Note

PSP format is only supported for emulation of the I^2S protocol. Operation in PSP mode, except as described in *PXA3xx Processor Family I2S Emulation Using SSP/PSP Application Note,* is not supported.

4.5.2.14 Data Size Select (DSS)

Use the DSS field 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 data sizes of 8, 16, 18, and 32 bits in length. Other values are reserved and undefined.

4.5.3 SSP Control Registers 1 (SSCR1_x)

The SSP Port Control 1 registers (SSCR1_1, SSCR1_2, SSCR1_3 and SSCR1_4) contain bit fields that control various SSP port functions. Table 122 shows bit locations corresponding to control bit fields in SSCR1_x. The reset state of all bits is shown, but must be set to the preferred value before enabling the SSP port by setting SSCR0_x[SSE].

Write 0b0 to reserved bits; the read values of reserved bits are undetermined.

Table 122: SSCR1_x Bit Definitions

Physical Address	SSPx
0x4100_0004 (SSP1) 0x4170_0004 (SSP2) 0x4190_0004 (SSP3) 0x41A0_0004 (SSP4)	SSCR1_1 SSCR1_2 SSCR1_3 SSCR1_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
	TTELP	TTE	EBCEI	SCFR	ECRA	ECRB	SCLKDIR	SFRMDIR	RWOT	TRAIL	TSRE	RSRE	TINTE	PINTE	Reserved	IFS	STRF	EFWR	RFT				TFT				Reserved	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 Three-state Enable On Last Phase: 0 = TXDx is three-stated 1/2 clock cycle after the beginning of the LSB. 1 = TXDx output signal is three-stated on the clock edge that ends the LSB.
30	R/W	TTE	TXD Three-State Enable: 0 = TXDx output signal is not three-stated. 1 = TXD is three-stated when not transmitting data.
29	R/W	EBCEI	Enable Bit Count Error Interrupt: 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: 0 = Clock input to SSPSCLKx is continuously running 1 = Clock input to SSPSCLKx is only active during data transfers. (ob1 Required for slave mode.)
27	R/W	ECRA	Enable Clock Request A: 0 = Clock request from other SSPx is disabled. 1 = Clock request from other SSPx is enabled.
26	R/W	ECRB	Enable Clock Request B: 0 = Clock request from other SSPx is disabled. 1 = Clock request from other SSPx is enabled.
25	R/W	SCLKDIR	SSP Serial Bit Rate Clock (SSPSCLKx) Direction: 0 = Master mode, SSPx port drives SSPSCLKx. 1 = Slave mode, SSPx port receives SSPSCLKx.
24	R/W	SFRMDIR	SSP Frame (SSPSFRMx) Direction: 0 = Master mode, SSPx port drives SSPSFRMx. 1 = Slave mode, SSPx port receives SSPSFRMx.
23	R/W	RWOT	Receive Without Transmit: 0 = Transmit/receive mode. 1 = Receive without transmit mode.
22	R/W	TRAIL	Trailing Byte: 0 = Trailing bytes are handled by the processor. 1 = Trailing bytes are handled by DMA bursts.

Table 122: SSCR1_x Bit Definitions (Continued)

Physical Address		SSPx
0x4100_0004 (SSP1) 0x4170_0004 (SSP2) 0x4190_0004 (SSP3) 0x41A0_0004 (SSP4)	SSCR1_1 SSCR1_2 SSCR1_3 SSCR1_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**
	TTELP	TTE	EBCEI	SCFR	ECRA	ECRB	SCLKDIR	SFRMDIR	RWOT	TRAIL	TSRE	RSRE	TINTE	PINTE	Reserved	IFS	STRF	EFWR	RFT				TFT				Reserved	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: 0 = DMA service request is disabled. 1 = DMA service request is enabled.
20	R/W	RSRE	Receive Service Request Enable: 0 = DMA service request is disabled. 1 = DMA service request is enabled.
19	R/W	TINTE	Receiver Time-out Interrupt Enable: 0 = Receiver time-out interrupt is disabled. 1 = Receiver time-out interrupt is enabled.
18	R/W	PINTE	Peripheral Trailing Byte Interrupt Enable: 0 = Peripheral trailing byte interrupt is disabled. 1 = Peripheral trailing byte interrupt is enabled.
17	—	—	**Reserved**
16	R/W	IFS	Invert Frame Signal 0 = SSPSFRMx polarity is determined by the PSP polarity bits. 1 = SSPSFRMx will be inverted from normal-SSPSFRMx (as defined by the PSP polarity bits). (Works in all frame formats: SPI, SSP, and PSP)
15	R/W	STRF	Select FIFO For Efwr (Test Mode Bit). Only when EFWR = 1: 0 = TXFIFO is selected for both Writes and Reads through the SSP Data register (SSDR_x). 1 = RXFIFO is selected for both Writes and Reads through the SSP Data register (SSDR_x).
14	R/W	EFWR	Enable FIFO Write/Read (Test Mode Bit): 0 = FIFO Write/Read special function is disabled (normal SSPx operational mode). 1 = FIFO Write/Read special function is enabled.
13:10	R/W	RFT	RXFIFO Trigger Threshold: Sets threshold level at which RXFIFO asserts interrupt. Level should be set to the preferred threshold value minus 1.
9:6	R/W	TFT	TXFIFO Trigger Threshold: Sets threshold level at which TXFIFO asserts interrupt. Level should be set to the preferred threshold value minus 1.

Table 122: SSCR1_x Bit Definitions (Continued)

Physical Address		SSPx
0x4100_0004 (SSP1) 0x4170_0004 (SSP2) 0x4190_0004 (SSP3) 0x41A0_0004 (SSP4)	SSCR1_1 SSCR1_2 SSCR1_3 SSCR1_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
	TTELP	TTE	EBCEI	SCFR	ECRA	ECRB	SCLKDIR	SFRMDIR	RWOT	TRAIL	TSRE	RSRE	TINTE	PINTE	Reserved	IFS	STRF	EFWR	RFT				TFT				Reserved	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	—	—	Reserved
4	R/W	SPH	Motorola SPI SSPSCLK phase setting: 0 = SSPSCLKx is inactive until one cycle after the start of a frame and active until 1/2 cycle before the end of a frame. 1 = SSPSCLKx is inactive until 1/2 cycle after the start of a frame and active until one cycle before the end of a frame.
3	R/W	SPO	Motorola SPI SSPSCLK Polarity Setting: 0 = The inactive or idle state of SSPSCLKx is low. 1 = The inactive or idle state of SSPSCLKx is high.
2	R/W	LBM	Loopback Mode (Test Mode Bit): 0 = Normal serial port operation is enabled. 1 = Output of TX serial shifter is internally connected to input of RX serial shifter.
1	R/W	TIE	Transmit FIFO Interrupt Enable: 0 = TXFIFO threshold-level-reached interrupt is disabled. 1 = TXFIFO threshold-level-reached interrupt is enabled.
0	R/W	RIE	Receive FIFO Interrupt Enable: 0 = RXFIFO threshold-level-reached interrupt is disabled. 1 = RXFIFO threshold-level-reached interrupt is enabled.

4.5.3.1 TXD Three-State Enable On Last Phase (TTELP)

Use TTELP in conjunction with TTE. When TTE is set, SSPTXDx is three-stated when not transmitting data. If TTELP is set, the SSPTXDx output signal goes to the high-impedance state 1/2 clock cycle earlier than that specified in the TTE bit description.

4.5.3.2 TXD Three-State Enable (TTE)

If TTE is cleared, the SSPTXDx output signal is always driven. When TTE is set, the SSPTXDx output signal goes into high-impedance during the time that the SSPx port is not transmitting data. The timing for the high-impedance enable/disable varies according to the different serial formats and frame direction. For Motorola SPI format, the SSPTXDx output signal goes into high-impedance if the SSPSFRMx interface signal is de-asserted (at 0b1).

For PSP format, if the SSPx port is a slave to frame (SSCR1_x[SFRMDIR] set), the SSPTXDx output signal goes into high impedance on the same clock edge that ends the SSPTXDx LSB. If the SSP is a master to frame (SSCR1_x[SFRMDIR] cleared), the SSPTXDx output signal goes into high

impedance on the clock edge after the beginning of the SSPTXDx LSB, even if the clock edge does not appear on SSPSCLKx.

Note

PSP format is only supported for emulation of the I^2S protocol. Operation in PSP mode, except as described in *PXA3xx Processor Family I2S Emulation Using SSP/PSP Application Note,* is not supported.

4.5.3.3 Enable Bit Count Error Interrupt (EBCEI)

When EBCEI is set, a bit-count error interrupt is generated. A bit-count error occurs when the SSPx port is a slave to clock (SCLKDIR set) or slave to frame (SFRMDIR set) and the SSPx port detects a new frame before the internal bit counter has reached zero, or before the SSPTXDx LSB is driven.

4.5.3.4 Slave Clock is Free Running (SCFR)

When in slave mode (SCLKDIR set), the SCFR bit informs the SSPx port that the external clock that is input to the SSPCLKx signal is running continuously (SCFR cleared) or not (SCFR set). When SCLKDIR is cleared, the SSPCLKx signal is an output and SCFR is ignored.

Note

When the enhanced SSPx port is configured for TI-SSP mode or PSP mode as slave to clock (SCLKDIR set), ensure SCFR is set (regardless of whether or not SSPSCLKx driven by external device is continuous). PSP format is supported only for emulation of the I^2S protocol. Operation in PSP mode, except as described in *PXA3xx Processor Family I2S Emulation Using SSP/PSP Application Note,* is not supported.

4.5.3.5 Enable Clock Request A (ECRA)

The clock output on this SSP port SSPSCLK signal can be configured to run continuously even if this SSP port is disabled (SSCR0_x[SSE] cleared, see Table 121). This configuration allows the SSPSCLK output from this SSP port to be used as the SSPSCLK input to other SSP ports. The SSPSCLKx output signal continues to run at the programmed bit rate when one or more of these three conditions are true:

- Another SSP port clock request is active and that SSP port clock request into this SSP port is enabled in this SSP port (ECRA=1 or ECRB =1 in this SSP).
- This SSP port is enabled and in the middle of a serial data transfer.
- This SSP has its SSPSCLKEN input signal set and this SSP has SSCR0_[ECS] cleared.

The other SSP port clock request is active when it has data to send with its TXFIFO, TXFIFO Holding register, and Shift register are not all empty (see the tx_not_empty signal in Figure 49). The ECRA bit is not functional for SSP4, so clear the ECRA bit to 0b0. See Table 123.

Table 123: SSP Clock Request Enable Selection

Enable Clock Request	SSP1	SSP2	SSP3
A	from SSP2	from SSP1	from SSP1

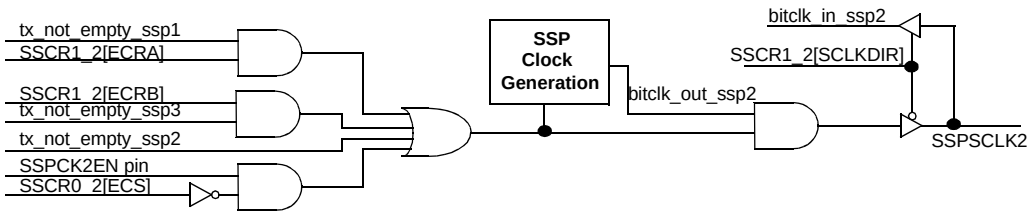
Table 123: SSP Clock Request Enable Selection (Continued)

Enable Clock Request	SSP1	SSP2	SSP3
B	from SSP3	from SSP3	from SSP2

4.5.3.6 Enable Clock Request B (ECRB)

ECRB enables the clock request from one of the other two SSP ports. See the description for the ECRA bit in Section 4.5.3.5. Refer to Table 123 and Figure 49 for an explanation on clock enabling (SSP2 is only shown as an example). The ECRB bit is not functional for SSP4, so clear ECRB to 0b0.

Figure 49: Clock Enabling on SSP2



NOTE: 1) The SSCR0 andSSCR1 control bits above are in SSP2.

2) Because SSP3 and SSP4 do not have an external clock source, the SSPSCLKEN signals for SSP3 and SSP4 are tied to ground.

3) The ECRA and ECRB inputs to SSP4 are also tied to ground.

4.5.3.7 SSP Serial Bit-Rate Clock Direction (SCLKDIR)

SCLKDIR is a Read-Write bit that determines whether the SSPx port is the master or slave with respect to SSPSCLKx:

- SCLKDIR cleared — The SSPx port generates SSPSCLKx internally, acts as the master, and drives the SSPSCLKx output signal.
- SCLKDIR set — The SSPx port acts as the slave, receives the SSPSCLKx input signal from an external clock source, and uses SSPSCLKx to determine when to drive transmit data on SSPTXDx and when to sample receive data on SSPRXDx.

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 SSPx port, the SCLKDIR bit has precedence over the GPIO direction bit. Therefore, Marvell recommends programming SCLKDIR and SFRMDIR before writing to the GPIO direction bits to prevent any possible contention on the SSPSCLKx or SSPSFRMx interface signals. Also, when SCLKDIR is set, SSCR0_x[ECS] and SSCR0_x[NCS] must both be cleared.

Normally, SCLKDIR and SFRMDIR are the same value, either both set or both cleared. However, this is not a requirement. When SFRMDIR and SCLKDIR are different values, the SSP is said to be configured in “mixed mode” - meaning that the frame signal and the clock signal are not sourced by the same device. The PXA31x processor cannot operate in mixed mode with serial bit rates greater than 13 MHz.

4.5.3.8 SSP Frame Direction (SFRMDIR)

The Read-Write SFRMDIR bit determines whether the SSPx port is the master or slave with respect to the SSPSFRMx interface signal:

- SFRMDIR cleared — The SSPx port internally generates frame, acts as the master, and drives the SSPSFRMx output signal.
- SFRMDIR set — The SSPx port acts as the slave and receives the SSPSFRMx input signal from an external device.

When the SSPx port is configured as a slave to frame, the SSPx port is enabled by setting SSCR0_x[SSE] (see Table 121.) Then the software must wait until SSRSR_X[CSS] is cleared automatically (see Table 124) before enabling the external device so it can begin asserting the SSPSFRMx input signal.

When the GPIO alternate function is selected for the SSPx port, SFRMDIR has precedence over the GPIO direction bit.

Normally SCLKDIR and SFRMDIR are the same value, either both set or both cleared; however, this is not a requirement. When SFRMDIR and SCLKDIR are different values, the SSP is said to be configured in “mixed mode” - meaning that the frame signal and the clock signal are not sourced by the same device.

Note

Marvell recommends programming SCLKDIR and SFRMDIR before programming the GPIO direction bits to prevent any possible contention on the SSPSCLKx or SSPSFRMx interface signals. When SCLKDIR is set, the SSCR0_x[NCS] and SSCR0_x[ECS] (see Table 121) must both be cleared.

4.5.3.9 Receive Without Transmit (RWOT)

The RWOT bit is a Read-Write bit used to place the SSPx port into a mode similar to half- duplex. When the SSPx port is in the Transmit/Receive mode determined by RWOT cleared, the SSPx port transmits and receives data simultaneously as supported by the individual formats (for example, normally all modes are full-duplex) and the SSPSCLKx interface signal only toggles while an active data transfer is underway. When in Receive-without-Transmit mode as determined by RWOT set, the SSPx port continues to clock in received data independent of whether data resides in the TXFIFO. Data is transmitted/received according to the selected format immediately after the SSPx port SSCR0_x[SSE]) is set (see Table 121), which allows the SSPx port to receive data without transmitting data (half-duplex only). If there is no data to transmit when RWOT is set, disable the DMA service requests and interrupts for the TXFIFO by clearing both TSRE and TIE. If the TXFIFO is empty, the SSPTXDx output signals is driven to 0b0. The TXFIFO underrun condition does not occur when RWOT is set. When RWOT is set, SSSR_x[BUSY] (see Table 124) remains active (set to 0b1) until software clears RWOT. If the SSPx port has been in RWOT mode and software disables RWOT mode by clearing RWOT, an extra frame cycle may occur due to the synchronization delay between the internal bus and SSPx port clock domains. Do not set RWOT when SSCR0_x[MOD] is set (see Table 121).

Note

In general, RWOT must be set for a full-duplex transfer where SSPx port is transmitting/receiving data simultaneously to allow for the SSPx port to receive data correctly even when there is no data in TxFIFO to transmit. If this is not done, then an underflow on TxFIFO can cause receive data corruption, which is an undesirable condition.

4.5.3.10 Trailing Byte (TRAIL)

Use the Read-Write TRAIL bit to configure how trailing bytes are handled:

- TRAIL cleared — Trailing bytes are handled by the processor.
- TRAIL set — Trailing bytes are handled by the DMA bursts.

See Section 4.4.2 for details.

4.5.3.11 Transmit Service Request Enable (TSRE)

Use TSRE to enable the DMA service request for the TXFIFO:

- TSRE cleared — The DMA service request for the TXFIFO is masked and the state of the service request (SSSR_x[TFS], see Table 124) for the TXFIFO is ignored.
- TSRE set — The DMA service request for the TXFIFO is enabled and whenever the SSSR_x[TFS] is set, a DMA service request is generated.

Clearing TSRE does not affect the current state of SSSR_x[TFS] or the ability of the TXFIFO logic to set and clear SSSR_x[TFS] as it only blocks generation of the DMA service request. The state of TSRE does not affect the interrupt request that is generated (if enabled by setting TIE) whenever SSSR_x[TFS] is set.

4.5.3.12 Receive Service Request Enable (RSRE)

Use RSRE to enable the DMA service request for the RXFIFO:

- RSRE cleared — The DMA service request for the RXFIFO is masked and the state of the service request (SSSR_x[RFS], see Table 124) bit for the RXFIFO is ignored.
- RSRE set — The DMA service request for the RXFIFO is enabled and whenever SSSR_x[RFS] is set, a DMA service request for the RXFIFO is generated.

Clearing RSRE does not affect the current state of SSSR_x[RFS] or the ability of the RXFIFO logic to set and clear SSSR_x[RFS] as it only blocks the generation of a DMA service request for the RXFIFO. The state RSRE does not affect the interrupt request that is generated (if enabled by setting RIE) whenever SSSR_x[RFS] is set.

4.5.3.13 Receiver Time-Out Interrupt Enable (TINTE)

Use the Read-Write TINTE bit to mask or enable the receiver time-out interrupt:

- TINTE cleared — The receiver time-out interrupt is masked and the state of SSSR_x[TINT] (see Table 124) is ignored.
- TINTE set — The receiver time-out interrupt is enabled and whenever SSSR_x[TINT] is set, an interrupt request is generated.

Clearing TINTE does not affect the current state of SSSR_x[TINT] or the ability of the RXFIFO logic to set and clear SSSR_x[TINT] as it only blocks the generation of the interrupt request.

4.5.3.14 Peripheral Trailing Byte Interrupt Enable (PINTE)

Use the Read-Write PINTE bit to enable the peripheral trailing byte interrupt.

- PINTE cleared – The interrupt request is masked and the state of SSSR_x[PINT] (see Table 124) is ignored.
- PINTE set – The interrupt request is enabled and whenever SSSR_x[PINT] is set, an interrupt request is sent to the interrupt controller.

Clearing PINTE does not affect the current state of SSSR_x[PINT] or the ability of the RXFIFO logic to set and clear SSSR_x[PINT] as it only blocks the generation of the interrupt request.

4.5.3.15 Invert Frame Signal (IFS)

Set IFS to invert the frame signal input to or output from the SSPx port. When set, this inverts the input or output frame signal for all frame formats (SPI, SSP, PSP).

4.5.3.16 Select FIFO For (EFWR set) Test Mode (STRF)

When EFWR is set:

- STRF cleared — Enables both Reads from and Writes to the TXFIFO.
- STRF set — Enables both Reads from and Writes to the RXFIFO.

4.5.3.17 Enable FIFO Write/Read Function (EFWR)

Use EFWR to enable a test mode for the SSPx port:

- EFWR cleared — The SSPx port operates in the normal mode.
- EFWR set – All processor Reads from or Writes to the SSPx Port Data register (SSDR_x, see Table 126) cause the SSPx port to Read from and Write to either the TXFIFO or the RXFIFO, depending on the programmed state of the STRF bit.

When STRF is cleared, all Writes to the SSDR_x register are transferred to the TXFIFO, and Reads from the SSDR register read back the data that was transferred into the TXFIFO, in first-in-first-out order. When STRF is set, all Writes to the SSDR register are transferred into the RXFIFO, and reads from the SSDR register read back the data transferred into the RXFIFO, in first-in-first-out order. When EFWR is set to enable test mode, data is not transmitted on the SSPTXD output signal, data input from the SSPRXD input signal is not stored in the RXFIFO, and SSSR_x[Busy] and SSSR_x[ROR] are ignored. The Interrupt Test register (SSITR_x, see Table 125), however, is still functional. When EFWR is set, software can test whether the TXFIFO or the RXFIFO operates properly as a FIFO memory stack. Software should verify that SSSR_x[CSS] is set and cleared before reading the TXFIFO.

4.5.3.18 Receive FIFO (Interrupt/DMA) Trigger Threshold (RFT)

The RFT bits set the level at which the RXFIFO logic triggers a DMA service request (if enabled) or an interrupt request (if enabled - refer to Section 4.5.3.1). The service request is triggered when the number of RXFIFO entries that contain data is greater than the value programmed into RFT.

If FIFO packing is enabled, SSCR0_x[FPCKE=1], while SSCR0_x[EDSS] is cleared, the effective RFT value is (2 x (RFT +1)) when SSSR_x[OSS]=1.

4.5.3.19 Transmit FIFO (Interrupt/DMA) Trigger Threshold (TFT)

The TFT bits set the level at which the TXFIFO logic triggers a DMA service request (if enabled) or an interrupt request (if enabled - refer to Section 4.5.3.1). The service request is triggered when the number of TXFIFO entries that contain data is less than or equal to TFT + 1.

If SSCR0_x[FPCKE] is set while SSCR0_x[EDSS] is cleared, the effective TFT value is (2 x (TFT + 1) + 1).

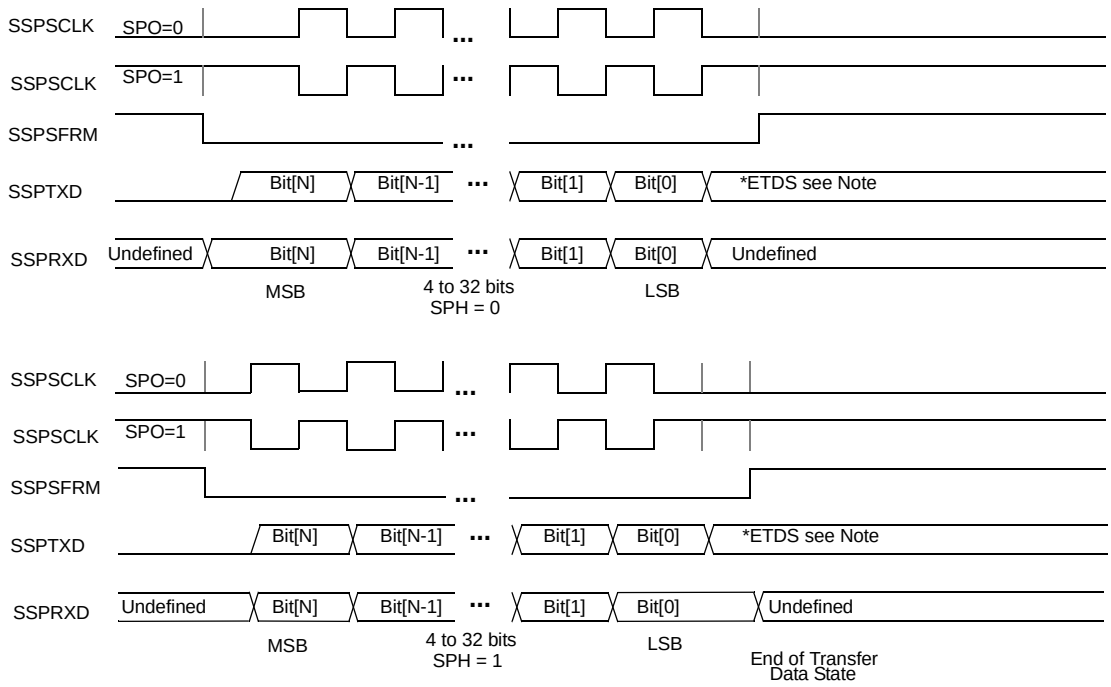
4.5.3.20 Serial Clock Phase (SPH)

Use SPH to determine the phase relationship between the SSPSCLKx and the SSPSFRMx interface signals when the Motorola* SPI format is selected by SSCR0_x[FRF] = 0, see Table 121:

- SPH cleared – SSPSCLKx remains in its inactive/idle state (as determined by the SPO bit) until one full clock cycle after the SSPSFRMx signal is asserted to 0b0 at the beginning of a frame. SSPSCLKx continues to transition for the rest of the frame and is then held in its inactive state one-half of an SSPSCLKx cycle before the SSPSFRMx signal is de-asserted to 0b1 at the end of the frame.
- SPH set – SSPSCLKx remains in its inactive/idle state (as determined by the SPO bit) until one-half clock cycle after the SSPSFRMx signal is asserted to 0b0 at the beginning of a frame. SSPSCLK continues to transition for the remainder of the frame and is then held in its inactive state one full SSPSCLKx cycle before the SSPSFRMx signal is de-asserted to 0b1at the end of the frame. The combination of SPO and SPH determines when SSPSCLKx is active during the assertion of the SSPSFRMx signal, and which SSPSCLKx edge transmits data via the SSPTXD output signal and which SSPSCLKx edge receives data via the SSPRXDx input signal.
- When SPO and SPH are both programmed to the same value (both cleared or both set), transmit data is driven via the SSPTXDx signal on the falling edge of SSPSCLKx and data received via the SSPRXDx signal is latched on the rising edge of SSPSCLKx. When SPO and SPH are programmed to opposite values, one cleared and the other to set, transmit data is driven via the SSPTXDx signal on the rising edge of SSPSCLKx and data received via the SSPRXDx signal is latched on the falling edge of SSPSCLKx. Figure 50 shows the interface signal timings for all four programming combinations of SPO and SPH. SPO determines the polarity of SSPSCLKx and SPH determines the phase relationship between SSPSCLKx and the SSPSFRMx signal, shifting SSPSCLKx one-half clock cycle to the left or right during the assertion of the SSPSFRMx signal.

SPH is ignored for all data frame formats except for the Motorola* SPI format (SSCR0[FRF] = 00).

Figure 50: Motorola SPI Formats for SPO and SPH Programming



Note: When in Motorola SPI format, if SSPx is the master and the SSPSP_x[ETDS] bit is 0b0, the end-of-transfer-data state for the SSPTXDx output signal is 0b0. If SSPx is the master and the SSPSP_x[ETDS] bit is 0b1, the SSPTXD output signal will remain at last bit transmitted (LSB). If SSPx is the slave and SPH = 0, the ETDS value is ignored. If SSPx is the slave and SPH = 1, the SSPTXDx output signal will remain at the LSB value until the SSPSFRMx signal is deasserted, and will then be undefined.

The SSPRXDx input signal is undefined before the frame is active and after the LSB is received. No assumptions are made about it except that it should not float.

4.5.3.21 Serial Clock Polarity (SPO)

Use SPO to select the polarity of the inactive state of SSPSCLKx when the Motorola* SPI format is selected (SSCR0_x[FRF] = 00, see Table 121):

- SPO cleared – SSPSCLKx is held at 0b0 in the inactive or idle state when the enhanced SSPx port is not transmitting/receiving data.
- SPO set – SSPSCLKx is held at 0b1 during the inactive/idle state.

The programmed value of SPO does not alone determine which SSPSCLKx edge transmits or receives data. SPO in combination with SPH determines which SSPSCLKx edge transmits data via the SSPTXDx output signal and which SSPSCLKx edge latches data received via the SSPRXDx input signal.

Note

SPO is ignored for all data frame formats except for the Motorola* SPI format (SSCR0[FRF] = 00).

4.5.3.22 Loop-Back Mode (LBM)

Use LBM to enable and disable a loop-back test mode:

- LBM cleared — The SSPx port operates normally. The transmit and receive data paths are independent and transfer data via their respective SSPTXDx or SSPRXDx interface signals.
- LBM set —The output of the TX serial shifter is internally connected to the input of the RX serial shifter.

During loopback mode, the SSPTXD output signal continues to function as normal.

4.5.3.23 Transmit FIFO Interrupt Enable (TIE)

Use TIE to enable the TXFIFO service-interrupt request:

- TIE cleared — The TXFIFO service-interrupt request is masked and the state of the TXFIFO service-interrupt request (SSSR_x[TFS], see Table 124) is ignored.
- TIE set — The TXFIFO service-interrupt request is enabled and whenever the service interrupt request (SSSR_x[TFS]) is set, an interrupt request is generated.

Clearing SSCR1_x[TIE] (see Table 122) does not affect the current state of SSSR_x[TFS] or the ability of the TXFIFO logic to set (0b1) and clear (0b0) SSSR_x[TFS] — it only blocks generation of the interrupt request. The state of SSCR1_x[TIE] does not effect the TXFIFO DMA service request, which is generated whenever SSSR_x[TFS] is set.

4.5.3.24 Receive FIFO Interrupt Enable (RIE)

Use RIE to enable the RXFIFO service-interrupt request:

- RIE cleared — The RXFIFO service-interrupt request is masked and the state of the RXFIFO service-interrupt request (SSSR_x[RFS], see Table 124) is ignored.
- RIE set — The RXFIFO service-interrupt request is enabled and whenever the service-interrupt request (SSSR_x[RFS], see Table 124) is set, an interrupt request is generated.

Clearing RIE does not affect the current state of SSSR_x[RFS] or the ability of the RXFIFO logic to set and clear SSSR[RFS] as it only blocks generation of the interrupt request. The state of SSCR1_x[RIE] (see Table 122) does not affect generation of the RXFIFO DMA service request, which is asserted whenever SSSR_[RFS] is set.

4.5.4 SSP Status Registers (SSSR_x)

The SSPx Port Status registers (SSSR_1, SSSR_2, SSSR_3 and SSSR_4) contain bits that signal overrun errors as well as the TXFIFO and RXFIFO service requests. Each of these hardware-detected events signals an interrupt request to the interrupt controller, or a DMA request. The Status register also contains flags that indicate if the SSPx port is actively transmitting data, if the TXFIFO is not full, and if the RXFIFO is not empty. A signal- interrupt signal is sent to the interrupt controller for each SSPx port. These events can cause an interrupt request or a DMA request: end-of-chain, receiver timeout, peripheral trailing byte, RXFIFO overrun, RXFIFO service request, and TXFIFO service request.

Bits that cause an interrupt request remain set until they are cleared by writing a 0b1 to each bit. Once a status bit is cleared, the interrupt is cleared. Read-Write bits are called *status* bits (status bits are referred to as “sticky” and once set by hardware, they can only be cleared by writing a 0b1 to

each bit); read-only bits are called *flags*. Writing a 0b1 to a sticky status bit clears it; writing a 0b0 has no effect. Read-only flags are set to 0b1 and are cleared automatically to 0b0 by hardware, and Writes have no effect. Some bits that cause interrupt requests have corresponding mask bits in the Control registers and are indicated in the section headings that follow.

Table 124 shows the bit locations corresponding to the status and flag bits within the SSPx port SSSR_x register. All bits are read-only except SSSR_x[ROR], SSSR_x[PINT], SSSR_x[TUR], SSSR_x[BCE], SSSR_x[TINT], and SSSR_x[EOC], which are all Read-Write. The reset state of Read-Write bits is 0b0 and all bits return to their reset state when SSCR0_x[SSE] (see Table 121) is cleared.

Write 0b0 to reserved bits; reads from reserved bits are undetermined.

Table 124: SSSR_x Bit Definitions

Physical Address	SSPx
0x4100_0008 (SSP1) 0x4170_0008 (SSP2) 0x4190_0008 (SSP3) 0x41A0_0008 (SSP4)	SSSR_1 SSSR_2 SSSR_3 SSSR_4

	Bit	31	30	29	28	27	26	25	24	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																																
PXA32x and PXA30x Only		OSS	Reserved	Reserved						BCE	CSS	TUR	EOC	TINT	PINT	Reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	Reserved	
PXA31x Only		OSS	TX_OSS	Reserved						BCE	CSS	TUR	EOC	TINT	PINT	Reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	Reserved	
	Reset	0	0	?	?	?	?	?	?	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	R	OSS	Odd Sample Status 0 = RxFIFO entry has 2 samples 1 = RxFIFO entry has 1 sample. Note that this bit needs to be looked at only when FIFO Packing is enabled (FPCKE set). Otherwise this bit is zero. When SSPx port is in Packed mode, and the CPU is used instead of DMA to read the RxFIFO, CPU should make sure that SSSR[RNE]=1 **AND** SSSR[OSS]=0 before it attempts to read the RxFIFO.

Table 124: SSSR_x Bit Definitions (Continued)

Physical Address	SSPx
0x4100_0008 (SSP1) 0x4170_0008 (SSP2) 0x4190_0008 (SSP3) 0x41A0_0008 (SSP4)	SSSR_1 SSSR_2 SSSR_3 SSSR_4

	31	30	29	28	27	26	25	24	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
PXA32x and PXA30x Only	OSS	Reserved	Reserved						BCE	CSS	TUR	EOC	TINT	PINT	Reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	Reserved	
PXA31x Only	OSS	TX_OSS	Reserved						BCE	CSS	TUR	EOC	TINT	PINT	Reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	Reserved	
Reset	0	0	?	?	?	?	?	?	0	0	0	0	0	0	?	?	1	1	1	1	0	0	0	0	0	0	0	0	0	1	?	?

Bits	Access	Name	Description
30	R	TX_OSS	TX FIFO Odd Sample Status When SSPx port is in packed mode, the number of samples in the TX FIFO is: (TFL*2 + TX_OSS), when TNF=1 32, when TNF=0. The TX FIFO cannot accept new data when TNF=1 and TFL=15 and TX_OSS=1. (The TX FIFO has 31 samples). 0 = TxFIFO entry has an even number of samples 1 = TxFIFO entry has an odd number of sample. **NOTE:** This bit needs to be read only when FIFO Packing is enabled (SSCR0_x[FPCKE] set). Otherwise this bit is zero. **NOTE:** PXA31x Only
	—	—	Reserved (PXA32x and PXA30x Only)
29:24	—	—	Reserved
23	Read /Write 0b1 to Clear	BCE	Bit Count Error: 0 = The SSPx port has not experienced a bit count error. 1 = The SSPSFRMx signal was asserted when the bit counter was not zero.
22	Read Only	CSS	Clock Synchronization Status: 0 = The SSPx port is ready for slave clock operations. 1 = The SSPx port is currently busy synchronizing slave mode signals.
21	Read /Write 0b1 to Clear	TUR	Transmit FIFO Underrun: 0 = The TXFIFO has not experienced an underrun. 1 = A read from the TXFIFO was attempted when the TXFIFO was empty, causes an interrupt if it is enabled (SSCR0_x[TIM]=0).
20	Read /Write 0b1 to Clear	EOC	End Of Chain: 0 = DMA has not signaled an end of chain condition. 1 = DMA has signaled an end of chain condition.

Table 124: SSSR_x Bit Definitions (Continued)

Physical Address		SSPx
0x4100_0008 (SSP1) 0x4170_0008 (SSP2) 0x4190_0008 (SSP3) 0x41A0_0008 (SSP4)	SSSR_1 SSSR_2 SSSR_3 SSSR_4	

	Bit	31	30	29	28	27	26	25	24	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																																	
PXA32x and PXA30x Only		OSS	Reserved	Reserved						BCE	CSS	TUR	EOC	TINT	PINT	Reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	Reserved	
PXA31x Only		OSS	TX_OSS	Reserved						BCE	CSS	TUR	EOC	TINT	PINT	Reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	Reserved	
	Reset	0	0	?	?	?	?	?	?	0	0	0	0	0	0	?	?	1	1	1	1	0	0	0	0	0	0	0	0	0	1	?	?

Bits	Access	Name	Description
19	Read /Write 0b1 to Clear	TINT	Receiver Time-out Interrupt: 0 = No receiver time-out is pending. 1 = Receiver time-out pending, causes an interrupt request.
18	Read /Write 0b1 to Clear	PINT	Peripheral Trailing Byte Interrupt: 0 = No peripheral trailing byte interrupt is pending. 1 = Peripheral trailing byte interrupt is pending.
17:16	—	—	**Reserved**
15:12	R	RFL	Receive FIFO Level: Number of entries minus one in RXFIFO. Note: When the value 0xF is read, the RXFIFO is either empty or full, and software should read the RNE bit.
11:8	R	TFL	Transmit FIFO Level: Number of entries in TXFIFO. Note: When the value 0x0 is read, the TXFIFO is either empty or full, and software should read the TNF bit.
7	Read /Write 0b1 to Clear	ROR	Receive FIFO Overrun: 0 = RXFIFO has not experienced an overrun. 1 = Attempted data Write to full RXFIFO, causes an interrupt request.
6	R	RFS	Receive FIFO Service Request: 0 = RXFIFO level is at or below RFT threshold (RFT), or SSPx port is disabled 1 = RXFIFO level exceeds RFT threshold (RFT), causes an interrupt request
5	R	TFS	Transmit FIFO Service Request: 0 = TX FIFO level exceeds the TFT threshold (TFT + 1), or SSPx port disabled. 1 = TXFIFO level is at or below TFT threshold (TFT + 1), causes an interrupt request.

Table 124: SSSR_x Bit Definitions (Continued)

Physical Address		SSPx
0x4100_0008 (SSP1) 0x4170_0008 (SSP2) 0x4190_0008 (SSP3) 0x41A0_0008 (SSP4)	SSSR_1 SSSR_2 SSSR_3 SSSR_4	

	Bit	31	30	29	28	27	26	25	24	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																																	
PXA32x and PXA30x Only		OSS	Reserved	Reserved						BCE	CSS	TUR	EOC	TINT	PINT	Reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	Reserved	
PXA31x Only		OSS	TX_OSS	Reserved						BCE	CSS	TUR	EOC	TINT	PINT	Reserved		RFL				TFL				ROR	RFS	TFS	BSY	RNE	TNF	Reserved	
Reset		0	0	?	?	?	?	?	?	0	0	0	0	0	0	?	?	1	1	1	1	0	0	0	0	0	0	0	0	0	1	?	?

Bits	Access	Name	Description
4	R	BSY	SSP Busy: 0 = SSPx port is idle or disabled. 1 = SSPx port is currently transmitting or receiving framed data.
3	R	RNE	Receive FIFO Not Empty: 0 = RXFIFO is empty. 1 = RXFIFO is not empty.
2	R	TNF	Transmit FIFO Not Full: 0 = TXFIFO is full. 1 = TXFIFO is not full.
1:0	—	—	Reserved

4.5.4.1 Odd Sample Status (OSS)

OSS is a read-only status bit that indicates whether the RX FIFO entry has one or two samples. This bit must be checked only when FIFO packing is enabled; otherwise, this bit is zero. When the SSPx port is using FIFO packing and using CPU programmed I/O to read the RX FIFO, the software should ensure the RX FIFO is not empty (SSSR[RNE]=1) and that there are two samples in the RX FIFO Entry (SSSR[OSS]=0) before it reads the RX FIFO.

4.5.4.2 TX FIFO Odd Sample Status (TX_OSS)

Note

TX_OSS is a read-only status bit that indicates whether the TX FIFO entry has an odd or even number of samples. This bit must be read only when FIFO packing is enabled; otherwise, this bit is zero. In packed mode, the TX FIFO cannot accept new data when TNF=1 and TFL=15 and TX_OSS=1(TX FIFO has 31 samples). If data is written with the above conditions, the Write data will be lost. This bit is reserved for PXA30x processors.

4.5.4.3 Bit Count Error (BCE)

BCE is a Read-Write, one-to-clear status bit that indicates that the SSPx port detects that the SSPSFRMx interface signal has been asserted at the wrong time. This detection causes an interrupt (if enabled by SSCR1_[BCEI], see Table 122). BCE bit set indicates that an error has occurred and there is a need to re-synchronize with the master external device. The SSPx port may, depending upon the timing of the event, disregard both the sample that had the SSPSFRMx interface signal re-asserted in the middle of it and the next sample.

4.5.4.4 Clock Synchronization Status (CSS)

The CSS bit is a read-only flag bit that indicates that the SSPx port is busy synchronizing control signals into the SSPSCLKx domain. Software need only read CSS when the SSPx port is a slave to frame (SSCR1_x[SFRMDIR] set, see Table 122) or when the SSPx port is in the special FIFO test mode (SSCR1_x[EFWR] set). Software should wait until CSS is clear before allowing an external device to assert the external SSPSFRMx or SSPSCLKx interface signals.

4.5.4.5 Transmit FIFO Underrun (TUR)

TUR is a Read-Write, one-to-clear status bit that indicates that the transmitter tried to send data from the TXFIFO when the TXFIFO was empty. When TUR is set, an interrupt request is generated, if it is not masked by SSCR0_x[TIM], (see Table 121). TUR set does not generate a DMA service request. TUR remains set until software clears it by writing a 0b1, which also resets its interrupt request. Writing a 0b0 to TUR has no effect.

TUR can be set only when the SSPx port is a slave to the SSPSFRMx interface signal (SSCR1_x[SFRMDIR] set, see Table 122), or if the SSPx port is a master to the SSPSFRMx interface signal (SSCR1_x[SFRMDIR] cleared and the SSPx port is in network mode (SSCR0_x[MOD] set). TUR is not set if the SSPx port is in Receive-without-Transmit mode (SSCR1_x[RWOT] set).

4.5.4.6 End Of Chain (EOC)

EOC is a Read-Write, one-to-clear status bit that indicates that the DMA controller has signaled an end-of-chain event. EOC set indicates that the DMA controller has signaled an end-of-chain event. EOC clear indicates that the DMA has not signaled an end-of-chain event. EOC is cleared by software writing a 0b1 to it. The end-of-chain event indicates that the DMA Descriptor for the RXFIFO is ending (described in more detail in the DMA chapter). This event requires software intervention if data remains in the RXFIFO.

4.5.4.7 Receiver Time-Out Interrupt (TINT)

TINT is a Read-Write, one-to-clear status bit that is set when the RXFIFO has been idle (no samples received) for a period of time defined by the value programmed into the Timeout register (SSTO_x,

see Table 127). TINT set signals an interrupt request (if it is not masked by SSCR1_x[TINTE], see Table 122). TINT is cleared by software writing a 0b1 to it.

4.5.4.8 Peripheral Trailing Byte Interrupt (PINT)

PINT is a Read-Write, one-to-clear status bit that is set anytime the DMA controller signals an end-of-chain event and data remains in the RXFIFO. This event signals an interrupt request, if not masked by SSCR1_x[PINTE] (see Table 122), that requires the processor to transfer the remaining data from the RXFIFO. PINT is cleared by software writing a 0b1 to it.

4.5.4.9 Receive FIFO Level (RFL)

The 4-bit RFL bit field shows how many valid data entries minus 1 are currently in the RXFIFO. When the value 0xF is read, the RXFIFO is either empty or full, and software should read the RNE bit (FIFO is empty when RNE=0, FIFO is not empty when RNE=1.)

If FIFO packing is enabled (SSCR0_x[FPCKE] (see Table 121) is set) and the sample size is 8 or 16 bits (SSCR0_x[EDSS] is clear), the number of valid data entries in the RXFIFO is either (2 x RFL) + 1 or (2 x RFL) + 2. Reading SSSR_x[OSS] shows whether the RX FIFO entry has one or two samples.

4.5.4.10 Transmit FIFO Level (TFL)

The 4-bit TFL bit field shows how many valid data entries are currently in the TXFIFO. When the value 0x0 is read, the TXFIFO is either empty or full and software should read TNF.

If FIFO packing is enabled (SSCR0_x[FPCKE] (see Table 121) is set) and the sample size is 8 or 16 bits (SSCR0_x[EDSS] is clear), then number of valid data entries in the TXFIFO is either 2 x TFL or 2 x TFL + 1. Reading SSSR_x[OSS] reveals whether the RX FIFO entry has one or two samples.

4.5.4.11 Receiver Overrun (ROR)

The ROR bit is a Read-Write, one-to-clear status bit that is set when the receive logic attempts to place data into the RXFIFO after the RXFIFO has been completely filled. If the RXFIFO is full and new data is received, the newly received data is discarded. This process is repeated for each new datum received until at least one empty RXFIFO entry exists. When ROR is set, an interrupt request is generated if not masked by SSCR0_x[RIM] (see Table 121). Setting ROR does not generate any DMA service request. ROR remains set until cleared by writing a 0b1 to it, which also clears its interrupt request. Writing a 0b0 to ROR does has no effect.

4.5.4.12 Receive FIFO Service Request (RFS)

The read-only RFS flag bit is a flag bit that is set when the RXFIFO requires service to prevent an overrun. RFS set signals an interrupt request if it is not disabled by SSCR1_x[RIE] (see Table 122). RFS is set any time the RXFIFO has more entries of valid data than the number allowed by the RXFIFO trigger threshold; RFS is cleared automatically when the RXFIFO has the same or fewer valid data entries than allowed by threshold value. Setting RFS also signals a DMA service request, if SSCR1_x[RSRE] is set. After the processor or DMA reads the RXFIFO such that the RXFIFO now has the same or fewer data entries than the allowed threshold, RFS (and the DMA service request or interrupt request) is cleared automatically. Both SSCR1_x[RSRE] and SSCR1_x[RIE] should not be set at the same time

4.5.4.13 Transmit FIFO Service Request (TFS)

The TFS bit is a read-only flag bit that is set to generate an interrupt request when the TXFIFO requires service to prevent an underrun. TFS is set any time the TXFIFO has the same or fewer entries of valid data than allowed by the TXFIFO trigger threshold value (that is, when the number of TXFIFO valid data entries is less than or equal to TFT, see Table 122). TFS is cleared automatically when the TXFIFO has more entries of valid data than allowed by the trigger threshold value. When

TFS is set, an interrupt request is generated (if not masked by SSCR1_x[TIE], see Table 122). TFS set also generates a DMA service request (if enabled by SSCR1_x[TSRE]). After the processor or the DMA fills the TXFIFO to a point where the trigger threshold value is exceeded, TFS bit (and the DMA service request or interrupt request) is cleared automatically; if filling the TXFIFO does not exceed trigger threshold value, another DMA service request or another interrupt request is generated. Software should not set both SSCR1_x[TSRE] and SSCR1_x[TIE].

4.5.4.14 SSP Busy (BSY)

The BSY bit is a read-only flag bit that is set when the SSPx port is actively transmitting or receiving data, and is cleared automatically when the SSPx port is idle or disabled (SSCR0_x[SSE] cleared, see Table 121). BSY does not generate an interrupt request or a DMA service request. When the SSPx port is a master to the SSPSCLKx interface signal (output) and if software needs to know if the SSPx port is actively transmitting data, software must monitor TFL and BSY to determine when all bits have been transmitted. If the SSPx port is a slave to the SSPSCLKx interface signal (input) and software must know whether the SSPx port is actively transmitting or receiving data, software should monitor TFL, RFL, and BSY, and use the Timeout register (SSTO_x, see Table 127).

4.5.4.15 Receive FIFO Not Empty (RNE)

The RNE bit is a read-only flag bit that is set when the RXFIFO contains one or more entries of valid data. RNE is cleared automatically when the RXFIFO no longer contains any valid data. When RNE is set, an interrupt request is not generated. Software can poll RNE using programmed I/O to determine if it is necessary to remove trailing bytes of data from the RXFIFO.

4.5.4.16 Transmit FIFO Not Full (TNF)

The TNF is a read-only flag bit that is set to 0b1 when the TXFIFO has one or more entries that do not contain valid data (TNF=1 when the TX FIFO is not full). The TNF bit is cleared automatically to 0b0 when the TXFIFO is completely full. The TNF bit does not generate an interrupt. Software polls the TNF bit using programmed I/O to determine if it is necessary to fill the TXFIFO over its trigger threshold level.

Note

In packed mode, TNF=0 does not indicate a full TXFIFO. When using packed mode, treat TFL=15 as the FIFO full condition.

4.5.5 SSP Interrupt Test Registers (SSITR_x)

Only use the Read-Write SSP Interrupt Test registers (SSITR_1, SSITR_2, SSITR_3 and SSITR_4) for testing purposes. The interrupt request or DMA service request generated when one of these test bits is set remains active until the test bit is cleared by writing a 0b0. Setting any of these bits to 0b1 also causes the corresponding status bit(s) to be set to 0b1 in the SSPx port Status register (SSSR_x, see Table 124).

Write 0b0 to reserved bits; reads from reserved bits are undetermined.

Table 125: SSITR_x Bit Definitions

Physical Address	SSPx
0x4100_000C (SSP1)	SSITR_1
0x4170_000C (SSP2)	SSITR_2
0x4190_000C (SSP3)	SSITR_3
0x41A0_000C (SSP4)	SSITR_4

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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																								**TROR**	**TRFS**	**TTFS**	**Reserved**				
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	?	?	?

Bits	Access	Name	Description
31:8	—	—	Reserved
7	R/W	TROR	Test RXFIFO Overrun: 0 = No RXFIFO-overrun service request. 1 = Generates a non-maskable RXFIFO-overrun interrupt request. No DMA request is generated
6	R/W	TRFS	Test RXFIFO Service Request: 0 = No RXFIFO-service request. 1 = Generates a non-maskable RXFIFO-service interrupt request and DMA request
5	R/W	TTFS	Test TXFIFO Service Request: 0 = No TXFIFO-service request 1 = Generates a non-maskable TXFIFO-service interrupt request and DMA request
4:0	—	—	Reserved

4.5.5.1 Test RXFIFO Overrun (TROR)

Setting the TROR bit generates a non-maskable interrupt request for the RXFIFO. A DMA service request is not generated. TROR is cleared by writing a 0b0 to it.

4.5.5.2 Test RXFIFO Service Request (TRFS)

Setting the TRFS bit generates a non-maskable interrupt request and a DMA service request for the RXFIFO. TRFS is cleared by writing an 0b0 to it.

4.5.5.3 Test TXFIFO Service Request (TTFS)

Setting the TTFS bit generates a non-maskable interrupt request and a DMA service request for the TXFIFO.

4.5.6 SSP Data Registers (SSDR_x)

The SSPx Port Data registers (SSDR_1, SSDR_2, SSDR_3 and SSDR_4) are each two physical registers that have a common address. One SSDR_x is temporary storage for data that is transferred automatically into the TXFIFO; the other SSDR_x is temporary storage for data that is transferred automatically from the RXFIFO.

As programmed I/O or DMA access the SSDR_x, the TXFIFO or RXFIFO control logic transfers data automatically between the SSDR_x and the FIFO as fast as the system moves it. Data in the TXFIFO shifts up to accommodate new data that is written to the SSDR_X, unless it is an attempted Write to a full TXFIFO. Data in the RXFIFO shifts down to accommodate data that is read from the SSDR_x. The SSSR_x[TFL,RFL,RNE,TNF] status bits (see Table 124) show whether the FIFO is full, above/below a programmable FIFO trigger threshold level, or empty.

When using programmed I/O, data can be written to the SSDR_X register anytime the TXFIFO falls below its trigger threshold level.

When a data sample size of less than 32 bits is selected, or 16 bits for packed mode, software should right-justify the data that is written to the SSDR_x for automatic insertion into the TXFIFO. The transmit logic left-justifies the data and ignores any unused bits. Received data of less than 32 bits is right-justified automatically in the RXFIFO (thus, a Write in packed mode of less than 32 bits wide can not be performed). The TXFIFO and RXFIFO are cleared to 0b0 when the SSPx port is reset or disabled (by writing a 0b0 to SSCR0_x[SSE], see Table 121).

The reset state of SSDR_x is undetermined. Table 126 shows the location of the SSPx port SSDR_x.

Table 126: SSDR_x Bit Definitions

Physical Address	SSDDR	SSPx
0x4100_0010 (SSP1)	SSDDR_1	
0x4170_0010 (SSP2)	SSDDR_2	
0x4190_0010 (SSP3)	SSDDR_3	
0x41A0_0010 (SSP4)	SSDDR_4	

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Transmit/Receive Data																														
Reset	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**

Bits	Access	Name	Description
31:0	R/W	DATA	Data to be written to the TXFIFO read from the RXFIFO

4.5.7 SSP Time Out Registers (SSTO_x)

The SSPx Port Timeout registers (SSTO_1, SSTO_2, SSTO_3 and SSTO_4) specify the timeout (TIMEOUT) value used to signal a period of inactivity within the RXFIFO, see Section 4.4.2.1. When a timeout occurs, SSSR_x[TINT] (see Table 124) is set. When the TIMEOUT value is set to 0x000000, no time-out occurs and SSSR_x[TINT] is not set. The TIMEOUT interval is given by the calculation in Equation 1.

Equation 1: TIMEOUT Interval Equation

TimeOut Interval = SSTO_x[TIMEOUT] / Low-Speed I/O Bus Clock Frequency

Note

Peripheral Bus Clock Frequency = 26 MHz

Write 0b0 to reserved bits; reads from reserved bits are undetermined.

Table 127: SSTO_x Bit Definitions

Physical Address	SSPx
0x4100_0028 (SSP1)	SSTO_1
0x4170_0028 (SSP2)	SSTO_2
0x4190_0028 (SSP3)	SSTO_3
0x41A0_0028 (SSP4)	SSTO_4

User Settings																																
Bit	31	30	29	28	27	26	25	24	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	Timeout Value: TIMEOUT value Is the value (0 to 2^{24}-1) that defines the time-out interval. The time-out interval is given by the equation show in Equation 1.

4.5.8 SSP Programmable Serial Protocol Registers (SSPSP_x)

Note

PSP format is supported only for emulation of the I^2S protocol. Operation in PSP mode except as described in PXA3xx Processor Family I2S Emulation Using SSP/PSP Application Note is not supported.

The SSPx Port Read-Write Programmable Serial Protocol registers (SSPSP_1, SSPSP_2, SSPSP_3 and SSPSP_4) contain eight fields that program the various programmable serial-protocol (PSP) parameters. When using Programmable Serial Protocol (PSP) format in network mode, the parameters SFRMDLY, STRTDLY, DMYSTP, EDMYSTP, DMYSTRT, and EDMYSTRT must be set to 0b0; the other parameters SFRMP, SCMODE, FSRT, and SFRMWDTH are programmable.

Writes 0b0 to reserved bits; reads from reserved bits are undetermined.

Table 128: SSPSP_x Bit Definitions

Physical Address	SSPSP_x	SSPx
0x4100_002C (SSP1) 0x4170_002C (SSP2) 0x4190_002C (SSP3) 0x41A0_002C (SSP4)	SSPSP_1 SSPSP_2 SSPSP_3 SSPSP_4	

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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	EDMYSTOP			EDMYSTRT[1]		FSRT	DMYSTOP[1]		Reserved	SFRMWDTH						SFRMDLY[1]							DMYSTRT[1]		STRTDLY[1]				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	0	0	0	0	0

Bits	Access	Name	Description
31	—	—	Reserved
30:28	R/W	EDMYSTOP[1]	Extended Dummy Stop: The most-significant bits of the dummy stop delay
27:26	R/W	EDMYSTRT[1]	Extended Dummy Start: The most-significant bits of the dummy start delay
25	R/W	FSRT	Frame Sync Relative Timing Bit: 0 = Next frame is asserted after the end of the DMTSTOP timing. 1 = Next frame is asserted with the LSB of the previous frame.
24:23	R/W	DMYSTOP[1]	Dummy Stop: The least-significant bits of the dummy stop delay Programmed value of EDMYSTOP + DMYSTOP specifies the number (0-31) of active clocks (SSPSCLKx) that follow the end of the transmitted data.
22	—	—	Reserved
21:16	R/W	SFRMWDTH	Serial Frame Width: Least-significant bits of the serial frame width Programmed value of ESFRMWDTH + SFRMWDTH specifies the frame width from 0b000000 (one SSPSCLKx cycle) to 0b111111(63 SSPSCLKx cycles).
15:9	R/W	SFRMDLY[1]	Serial Frame Delay: Programmed value specifies the number (0 -127) of active one-half clocks (SSPSCLKx) asserted from the most-significant bit of TXDx (output) or RXD (input) being driven to SSPSFRMx.
8:7	R/W	DMYSTRT[1]	Dummy Start: Least-significant bits of the dummy start delay Programmed value of DMYSTART specifies the number (0-15) of active clocks (SSPSCLKs) between the end of start delay and when the most-significant bit of Transmit/Receive data is driven

Table 128: SSPSP_x Bit Definitions (Continued)

Physical Address		SSPx
0x4100_002C (SSP1) 0x4170_002C (SSP2) 0x4190_002C (SSP3) 0x41A0_002C (SSP4)	SSPSP_1 SSPSP_2 SSPSP_3 SSPSP_4	

User Settings																															
Bit	31	30	29	28	27	26	25	24	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	EDMYSTOP			EDMYSTRT[1]		FSRT	DMYSTOP[1]		Reserved	SFRMWDTH						SFRMDLY[1]							DMYSTRT[1]		STRTDLY[1]			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	0	0	0	0	0

Bits	Access	Name	Description
6:4	R/W	STRTDLY[1]	Start Delay: Programmed value specifies the number (0-7) of non-active clocks (SSPSCLKx) that define the duration of idle time
3	R/W	ETDS	End Of Transfer Data State: 0 = Low 1 = Last Value <Bit 0>
2	R/W	SFRMP	Serial Frame Polarity: 0 = SSPSFRMx is active low (0b0). 1 = SSPSFRMx is active high (0b1).
1:0	R/W	SCMODE	Serial Bit-rate Clock Mode: 00 – Data Driven (Falling), Data Sampled (Rising), Idle State (Low) 01 – Data Driven (Rising), Data Sampled (Falling), Idle State (Low) 10 – Data Driven (Rising), Data Sampled (Falling), Idle State (High) 11 – Data Driven (Falling), Data Sampled (Rising), Idle State (High)

1. Do not use in PSP Network mode.

.

4.5.8.1 Extended Dummy Stop (EDMYSTOP)

The three EDMYSTOP bits specify the most significant bits of the dummy stop delay. The EDMYSTOP bits supplement the DMYSTOP bits to generate a dummy stop delay of 0 to 31 SSPSCLKx cycles.

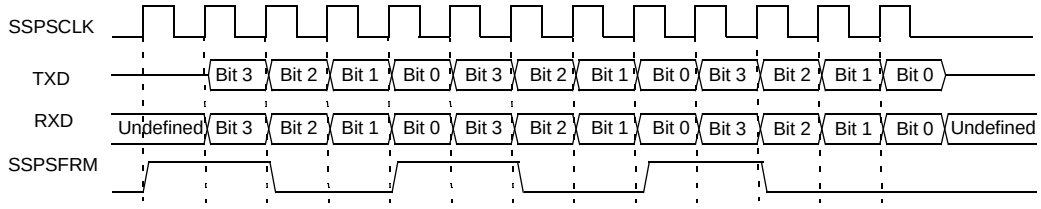
4.5.8.2 Extended Dummy Start (EDMYSTRT)

The two EDMYSTRT bits specify the most significant bits of the dummy start delay. The EDMYSTRT bits supplement the DMYSTRT bits to generate a dummy start delay of 0 to 15 SSPSCLKx cycles.

4.5.8.3 Frame Sync Relative Timing (FSRT)

Clearing FSRT causes the frame to be determined as illustrated in Figure 41. When set, FSRT causes the assertion of the SSPSFRMx interface signal that corresponds to the next sample during the transmission of the LSB of the current sample (see Figure 51). If FSRT is set, the EDMYSTRT, DMYSTRT, SFRMDLY, STRTDLY, EDMYSTOP and DMYSTOP fields must be set to 0b0.

Figure 51: Programmable Serial Protocol Format (example with consecutive transfers and FSRT bit set)



NOTE: This example uses a frame width of two, dummy start = 0b0, dummy stop = 0b0, start delay = 0b0, frame delay = 0b0, scmode = 1, TXDx three-state = 1, FSRT = 0b1, and frame polarity = 1.

4.5.8.4 Dummy Stop (DMYSTOP)

The two-bit dummy DMYSTOP field specifies the least-significant bits of the dummy stop delay (see T4 in Figure 42). The DMYSTOP bits supplement the EDMYSTOP bits to specify the number (0-31) of active clocks (SSPSCLKx) between the end of the LSB (bit 0) of the transmitted or received data SSPRXD and the beginning of the idle time.

4.5.8.5 Serial Frame Width (SFRMWDTH)

The six-bit SFRMWDTH field specifies the number (0-63) of active clocks (SSPSCLKx) that the SSPSFRMx interface signal remains active for each timeslot. The programmed value must not cause the SSPSFRMx interface signal to be asserted past the end of the dummy stop delay (see T_4 in Figure 42).

In slave mode (SSCR1_x[SFRMDIR] set, see Table 122), the SFRMWDTH field is ignored; however, the SSPSFRMx interface signal (input) must be asserted for at least one active clock (SSPSCLKx) before the MSB of the received data sample is latched and de-asserted for at least one active clock (SSPSCLKx) before the beginning of the next received data sample.

4.5.8.6 Serial Frame Delay (SFRMDLY)

The seven-bit SFRMDLY field specifies the number (0-127) of active and/or inactive half-clocks (SSPSCLKx) between the end of the dummy stop delay (see T_4 in Table 116) and the assertion of the next SSPSFRMx interface signal (see T_5 in Table 116).

4.5.8.7 Dummy Start (DMYSTRT)

The two-bit DMYSTRT field contains the least-significant bits of the dummy start delay. The DMYSTRT field supplements the EDMYSTRT field to specify the number (0 to 15) of active clocks (SSPSCLKx) between the end of start delay (see T_1 in Table 116) and when the MSP of transmitted or received data sample is driven (see T_2 in Table 116).

4.5.8.8 Start Delay (STRTDLY)

The three-bit STRTDLY field specifies the number of inactive clocks (SSPSCLKx) that define the length of the idle time between data sample transfers. STRTDLY must be programmed to 0b0 if the SSPSCLKxEN interface input signal and SSCR1_x[ECRA] is set or SSCR1_x[ECRB] is set. The STRTDLY field must be programmed to 0b0 whenever any of the SSPSCLKx or SSPSFRMx interface signals are configured as an input (SSCR1_[SCLKDIR] set and/or SSCR1_x[SFRMDIR] set).

4.5.8.9 End of Transfer Data State (ETDS)

The ETDS bit determines the state of the SSPTXDx interface-output signal at the end of a transfer:

- ETDS cleared — The state of the SSPTXDx interface-output signal is forced to 0b0 after the last bit (bit 0) of the frame has been sent and remains at 0b0 throughout the next idle period.
- ETDS set — The state of the SSPTXDx interface-output signal retains the value of the last bit sent (bit 0) throughout the next idle period.

4.5.8.10 Serial Frame Polarity (SFRMP)

The SFRMP bit specifies the active state of the SSPSFRMx interface signal:

- SFRMP cleared — The active state of the SSPSFRMx interface signal is low.
- SFRMP set — The active state of the SSPSFRMx interface signal is high.

During the idle state or when the SSPx port is disabled (SSCR0_x[SSE] cleared, see Table 121), the SSPSFRMx interface signal is in its inactive state. In slave mode (SSCR1_x[SFRMDIR] set, see Table 122), the SFRMP bit specifies the polarity of the incoming SSPSFRMx interface signal.

4.5.8.11 Serial Clock Mode (SCMODE)

The two-bit SCMODE field specifies one of four clock (SSPSCLKx) modes.

4.5.9 SSP TX Time Slot Active Registers (SSTSA_x)

Only used in network mode (SSCR0_x[MOD] set, see Table 121), the SSPx Port TX Time Slot Active registers (SSTSA_1, SSTSA_2, SSTSA_3, and SSTSA_4) are Read-Write registers that specify in which time slot the SSPx port transmits data. See Figure 48 for an example of the use of time slots when in network mode.

The eight-bit TTSA field specifies in which time slots the SSPx port transmits data and in which time slots the SSPx port does NOT transmit data. Bits beyond the SSCR0_x[FRDC] value are ignored (for example, if SSCR0_x[FRDC] = 0b011, specifying that four time slots are used, then TTSA bits 7:4 are ignored). If SSCR1_x[TTE] is set (see Table 122), the SSPx port three-states the SSPTXDx interface output signal line during time slots that have associated TTSA bits programmed to 0b0.

Write 0b0 to reserved bits; reads from reserved bits are undetermined.

Table 129: SSTSA Definitions

Physical Address	SSTSA	SSPx
0x4100_0030 (SSP1) 0x4170_0030 (SSP2) 0x4190_0030 (SSP3) 0X41A0_0030 (SSP4)	SSTSA_1 SSTSA_2 SSTSA_3 SSTSA_4	

User Settings																															
Bit	31	30	29	28	27	26	25	24	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

Table 129: SSTSA Definitions (Continued)

Physical Address		SSPx
0x4100_0030 (SSP1) 0x4170_0030 (SSP2) 0x4190_0030 (SSP3) 0X41A0_0030 (SSP4)	SSTSA_1 SSTSA_2 SSTSA_3 SSTSA_4	

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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
7:0	R/W	TTSA	TX Time Slot Active: 0 = SSPx port does NOT transmit data in this time slot. 1 = SSPx port does transmit data in this time slot.

4.5.10 SSP RX Time Slot Active Registers (SSRSA_x)

Only used in network mode (SSCR0_x[MOD] set, see Table 121), the SSPx Port RX Time Slot Active registers (SSRSA_1, SSRSA_2, SSRSA_3, and SSRSA_4) are Read-Write registers that specify in which time slot the SSPx port receives data. See Figure 48 for an example of the use of time slots when in network mode.

The eight-bit RTSA field specifies in which time slots the SSPx port receives data and in which time slots the SSPx port does *not* receive data. Bits beyond the SSCR0_x[FRDC] value are ignored. For example, if SSCR0_x[FRDC=0b011], specifying that four time slots are used, then RTSA bits 7:4 are ignored.

Write 0b0 to reserved bits; reads from reserved bits are undetermined.

Table 130: SSRSA_x Bit Definitions

Physical Address		SSPx
0x4100_0034 (SSP1) 0x4170_0034 (SSP2) 0x4190_0034 (SSP3) 0x41A0_0034 (SSP4)	SSRSA_1 SSRSA_2 SSRSA_3 SSRSA_4	

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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																								**RTSA**							
Reset	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Bits	Access	Name	Description
31:8	—	—	Reserved
7:0	R/W	RTSA	RX Time Slot Active: 0 = SSPx port does *not* receive data in this time slot. 1 = SSPx port receives data in this time slot.

4.5.11 SSP Time Slot Status Registers (SSTSS_x)

Only when in network mode (SSCR0_x[MOD] set, see Table 121) are the SSPx Port Time Slot Status registers (SSTSS_1, SSTSS_2, SSTSS_3 and SSTSS_4) read-only and identifies the time slot in which the SSPx port is operating.

Write 0b0 to reserved bits; reads from reserved bits are undetermined.

Table 131: SSTSS Bit Definitions

Physical Address	SSPx
0x4100_0038 (SSP1) 0x4170_0038 (SSP2) 0x4190_0038 (SSP3) 0x41A0_0038 (SSP4)	SSTSS_1 SSTSS_2 SSTSS_3 SSTSS_4

User Settings																															
Bit	31	30	29	28	27	26	25	24	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: 0 = SSPx port is in network mode and *no* frame is currently active. 1 = SSPx port is in network mode and a frame is currently active.
30:3	—	—	Reserved
2:0	R	TSS	Time Slot Status: Value indicates which time slot is currently active. Because of synchronization between the SSPx port's SSPSCLKx domain and an internal bus clock domain, the TSS value becomes stable between the beginning and end of the currently active time slot.

4.5.11.1 Network Mode Busy (NMBSY)

The NMBSY bit shows when the SSPx port is within a frame only when in network mode (SSCR0_x[MOD] set, see Table 121). NMBSY can be used by software to determine when a clean shutdown of the SSPx port can be initiated. Software should:

1. Determine that the TXFIFO is either empty or is emptied at the end of the next frame.
2. Deactivate the TXFIFO DMA service requests.
3. Clear SSCR0x[MOD] (to exit network mode, see Table 121).
4. Then poll NMBSY until it is cleared before disabling the SSPx port by clearing SSCR0_x[SSE].

When the SSPx port is a master of the frame signal (SSCR1_x[SFRMDIR] set), NMBSY is set as long as the port remains in network mode. When the SSPx port is a slave of the frame signal, NMBSY bit is cleared if the current frame (number of bits per sample * number of time slots per frame) has not expired since the last SSPSFRMx interface signal (in/out) was asserted.

4.5.11.2 Time Slot Status (TSS)

The three-bit TSS bit-field value identifies the time slot in which the SSPx port is operating. Due to synchronization between the SSPSCLKx domain and an internal bus clock domain, the TSS value

becomes stable approximately two internal bus clock cycles after the beginning of the associated time slot. The TSS value is not valid if NMBSY is cleared.

4.5.12 SSP Audio Clock Divider Registers (SSACD_x)

The Read-Write SSPx Port Audio Clock Divider registers (SSACD_1, SSACD_2, SSACD_3, and SSACD_4) select which clock frequency is sent to the SSPx port and to the SYSCLKx interface signal (output) when SSCR0_x[ACS] (see Table 121) is set. If SSCR0_x[SCR] is not 0x000, there is no guaranteed phase relationship between the SSPSYSCLKx and SSPSCLKx interface signals (outputs).

The frequency of the SSPSYSCLKx interface signal (output, internally called SYSCLK in Figure 52) is calculated by dividing the chosen PLL output clock frequency, selected by the 3-bit ACPS field, by the chosen divider (three-bit ACDS field). The frequency of internal SYSCLKx is then divided by 8, 4, or 1, depending on the SCDX8 and SCDB bits, to give the frequency (if SSCR0_x[SCR] = 0x000, see Table 121) of the SSPSCLKx interface signal (output).

The frequency of the SSPSCLKx interface signal is divided by the data size (SSCR0_x[EDSS] and SSCR0_x[DSS]) and by the number of network mode (if selected) time slots (SSCR0_x[FRDC] value), if any, to give the frequency of the SSPSFRMx interface signal (output). This register should only be programmed when SSP is not enabled. See Figure 52 for an illustration of how the internal audio clock is generated.

Write 0b0 to reserved bits; reads from reserved bits are undetermined.

Table 132: SSACD_x Bit Definitions

Physical Address	SSACD_x	SSPx
0x4100_003C (SSP1) 0x4170_003C (SSP2) 0x4190_003C (SSP3) 0x41A0_003C (SSP4)	SSACD_1 SSACD_2 SSACD_3 SSACD_4	

Bit	31	30	29	28	27	26	25	24	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																								SCDX8	ACPS			SCDB	ACDS		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	Reserved
7	R/W	SCDX8	SYSCLK Divided By 8: 0 = SYSCLK is divided by 4 to generate internal audio clock. 1 = SYSCLK is divided by 8 to generate internal audio clock.

Table 132: SSACD_x Bit Definitions (Continued)

Physical Address	SSACD_x	SSPx
0x4100_003C (SSP1)	SSACD_1	
0x4170_003C (SSP2)	SSACD_2	
0x4190_003C (SSP3)	SSACD_3	
0x41A0_003C (SSP4)	SSACD_4	

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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																								**SCDX8**	**ACPS**			**SCDB**	**ACDS**		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description	
6:4	R/W	ACPS	Audio Clock PLL Select: Value indicates which PLL output clock is sent to the clock divider in the Clock Unit:	
			ACPS Value	PLL Output Frequency
			000	5.622 MHz
			001	11.345 MHz
			010	12.235 MHz
			011	14.857 MHz
			100	32.842 MHz
			101	48.000 MHz
			110	Determined by Audio Clock Dither Divider Register value
			111	Reserved Note: Do NOT program Reserved values otherwise the SSP behavior could be undetermined.
3	R/W	SCDB	SYSCLK Divider Bypass: 0 = SYSCLK is divided by 4 (if SCDX8=0) or 8 (if SCDX8=1) to generate internal audio clock. 1 = SYSCLK is not divided.	

Table 132: SSACD_x Bit Definitions (Continued)

Physical Address	SSACD_1	SSACD_2	SSACD_3	SSACD_4	SSPx
0x4100_003C (SSP1)					
0x4170_003C (SSP2)					
0x4190_003C (SSP3)					
0x41A0_003C (SSP4)					

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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																								**SCDX8**	**ACPS**			**SCDB**	**ACDS**		
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description	
2:0	R/W	ACDS	Audio Clock Divider Select: Value indicates which divider will be used by the clock unit to generate the internal SYSCLK and SSPSYSCLKx output signal:	
			ACDS Value	Clock Divider Value
			000	1
			001	2
			010	4
			011	8
			100	16
			101	32
			110	Reserved **NOTE:** Do NOT program Reserved values otherwise the SSP behavior could be undetermined.
			111	Reserved **NOTE:** Do NOT program Reserved values otherwise the SSP behavior could be undetermined.

4.5.12.1 SYSCLK Divided By 8 (SCDX8)

If SCDX8 and SSCR0_x[ACS] (see Table 121) are both set, and SCDB is cleared, the internal SYSCLK is divided by 8 to generate the internal audio clock. If SCDX8 and SCDB are both cleared and SSCR0_x[ACS] is set, the internal SYSCLK is divided by 4 to generate the internal audio clock. If SSCR0_x[ACS] is cleared, the setting of SCDX8 has no effect (see Equation 2). If SSCR0_x[SRC] = 0x000 and the clock source select multiplexer (see top-right portion of Figure 52) is selected to by-pass the audio clock, the audio clock drives the SSPSCLKx interface signal (output).

4.5.12.2 Audio Clock PLL Select (ACPS)

The three-bit ACPS value usually selects (see Table 134) which PLL output frequency is sent to the audio-clock divider (called “divider” in Figure 52). However, when ACPS is set to 0b110, the output of the dithering frequency divider is sent to the divider. The output frequency of the dithering frequency divider is controlled by the values of the Audio-Clock Dither Divider register (see Section 4.5.13).

4.5.12.3 SYSCLK DIVIDER BYPASS (SCDB)

If SCDB and SSCR0_x[ACS] (see Table 121) are both set, the internal SYSCLK is not divided by 4 or 8; rather, SYSCLK is sent directly to the clock source select multiplexer (see top-right portion of Figure 52). If SCDB and SCDX8 are both cleared, and SSCR0_x[ACS] is set, SYSCLK is divided by 4 to generate the internal audio clock. If SSCR0[ACS] is cleared, SCDB has no effect. If SSCR0_x[SRC] = 0x000 and the clock source select multiplexer (see top-right portion of Figure 52) is selected to pass the audio clock, the audio clock drives the SSPSCLKx interface signal (output).

4.5.12.4 Audio Clock Divider Select (ACDS)

The three-bit ACDS value indicates which divisor is used by the divider to generate the internal SYSCLK (see Table 133). SYSCLK is sent to the clock source select multiplexer (see top-right portion of Figure 52) and the SSPSCLK divider, and SYSCLK becomes the SSPSCLKx interface signal (output).

Note

When the processor is run in the Ring Oscillator mode, the 624 MHz oscillator is disabled and therefore the SSCR0_x[ACS] must be 0b0 because the proper audio clock frequencies cannot be produced.

Note

In Figure 52, the 52MM multiplexer exists only in the PXA31x processor. In the PXA30x processor, 13 MHz clock is fed to the clock selection mux on left side of the figure.

Some combinations of ACPS and ACDS are not valid (see Table 133).

Table 133: SYSCLK Output Frequency Selection

PLL Output Frequency	ACPS Value	Audio Clock Divider Select (ACDS) Value					
		1	2	4	8	16	32
5.622 MHz	000	5.622 MHz	2.811 MHz	1.405 MHz	702.7 kHz	3.514 kHz	175.7 kHz
11.345 MHz	001	11.345 MHz	5.673 MHz	2.836 MHz	1.418 MHz	709.1 kHz	354.5 kHz
12.235 MHz	010	12.235 MHz	6.118 MHz	3.059 MHz	1.529 MHz	764.7 kHz	382.4 kHz
14.857 MHz	011	14.857 MHz	7.429 MHz	3.714 MHz	1.857 MHz	928.6 kHz	464.3 kHz
32.842 MHz	100	not valid	16.421 MHz	8.211 MHz	4.105 MHz	2.053 MHz	1.026 MHz
48.00 MHz	101	not valid	not valid	12.00 MHz	6.000 MHz	3.000 MHz	1.500 MHz

Table 134 shows recommended ACPS and ACDS values to approximate standard frame frequencies for a selected combination of bit sizes and time slots. Use the formulas in Equation 2 to calculate other combinations (use the second formula if SYSCLK (SSPSYSCLKx) is to be 4x or 8x SSPSCLKx):

Equation 2: Standard Frame Frequency Calculation

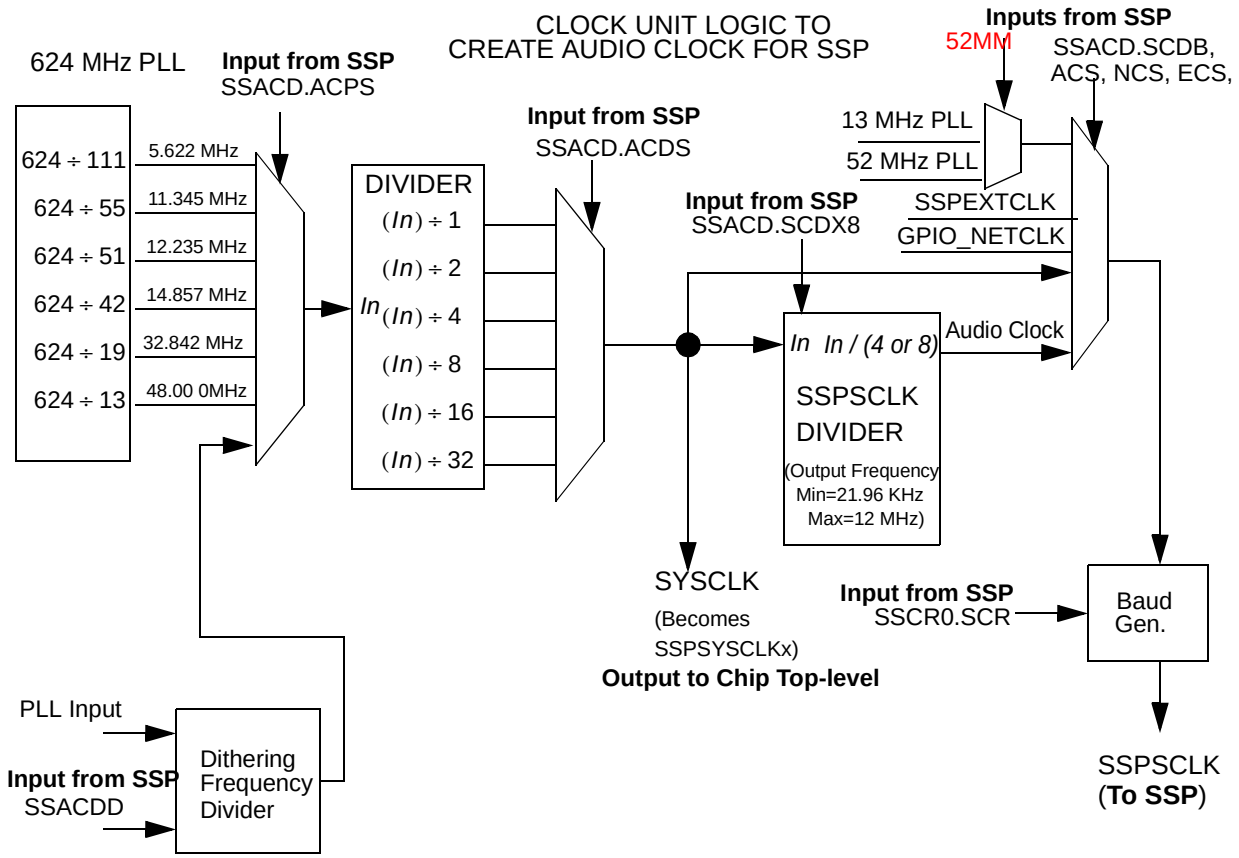
$$TimeSlots \times BitsPerSample \times StandardFrequency = SSPSCLKFrequency$$
$$SSPSCLKFrequency \times (4 \text{ or } 8) = SYSCLKFrequency$$

Choose ACPS and ACDS values to approximate the frequency of SYSCLK

Table 134: PLL Output Frequency and Divider Selection (Examples for Selected Time Slots and Data Sizes)

PLL Output Frequency	ACPS Value	SCDB	Divider Value (ACDS Choice)												Actual SSFSFRM Frequency	Closest Standard for SSPSFRM 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	010	0	8	4	2	1	4	2	1	—	2	1	—	—	47.79 kHz	48.00 kHz
11.345 MHz	001	0	8	4	2	1	4	2	1	—	2	1	—	—	44.32 kHz	44.10 kHz
5.622 MHz	000	0	8	4	2	1	4	2	1	—	2	1	—	—	21.96 kHz	22.05 kHz
32.842 MHz	100	0	—	32	16	8	32	16	8	4	16	8	4	2	16.04 kHz	16.00 kHz
5.622 MHz	000	0	16	8	4	2	8	4	2	1	42	2	1	—	10.98 kHz	11.025kHz
11.345 MHz	001	0	—	—	—	—	—	—	—	—	—	—	—	1	11.08 kHz	11.025 kHz
32.842 MHz	100	0	—	—	32	16	—	32	16	8	32	16	8	4	8.02 kHz	8.00 kHz
12.235 MHz	010	1	—	—	—	—	—	—	—	2	—	—	2	1	47.79 kHz	48.00 kHz
11.345 MHz	001	1	—	—	—	—	—	—	—	2	—	—	2	1	44.32 kHz	44.10 kHz
5.622 MHz	000	1	—	—	—	—	—	—	—	2	—	—	2	1	21.96 kHz	22.05 kHz

Figure 52: Audio Clock Selection



4.5.13 SSP Audio Clock Dither Divider Registers (SSACDD_x)

The Read-Write SSPx Port Audio-Clock Dither Divider registers (SSACDD_1, SSACDD_2, SSACDD_3 and SSACDD_4) determine the output frequency of the dithering- frequency divider (see Figure 52). When SSCR0_x[ACS] is set (see Table 121) and SSACD_x[ACPS] = 0b110 (see Table 132), the output of the dithering-frequency divider is selected as the input to audio-clock divider (called “divider” in Figure 52) to generate the frequencies of the internal SYSCLK and the SSPSCLKx interface signal (output). The SSACDD register contains a 15-bit numerator field (NUM) and a 12-bit denominator field (DEN) that are used to generate the dithered audio clock frequency according to the formula in Equation 3.

Equation 3: Formula to Generate Dithered Audio Clock Frequency

$$624\ \text{MHz} \times ((\text{DEN}/\text{NUM})/2) = \text{Dithered Audio Frequency}$$

Write 0b0 to reserved bits; Reads from reserved bits are undetermined. The 624 MHz System PLL clock only runs in S0/D0/C0 mode.

Table 135: SSACDD_x Bit Definitions

Physical Address	SSPx
0x4100_0040 (SSP1)	SSACDD_1
0x4170_0040 (SSP2)	SSACDD_2
0x4190_0040 (SSP3)	SSACDD_3
0x41A0_0040 (SSP4)	SSACDD_4

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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	**NUM**															**Reserved**				**DEN**											
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	—	—	Reserved
30:16	R/W	NUM	Numerator: Valid values as shown in Table 136.
15:12	—	—	Reserved
11:0	R/W	DEN	Denominator: Valid values as shown in Table 136.

4.5.13.1 Numerator (NUM)

The 15-bit numerator (NUM) field should be programmed only with the values shown in Table 136.

4.5.13.2 Denominator (DEN)

The 12-bit denominator (DEN) field should be programmed only with the values shown in Table 136.

4.5.13.3 Audio SSP Clock Generation

The audio SSP supports I^2S CODEC protocol. There are six supported audio frequencies for this protocol that are described in Table 136.

Note

When the processor is run in the S0/D0CS/C0 the 624 MHz oscillator is disabled and therefore the SSCR0_x[ACS] must be 0b0 because the proper audio clock frequencies cannot be produced.

Note

OS vendors specify an I^2S frame rate tolerance of +/- 5% from the I^2S standard to assure a good audio experience for listeners. The “Audio Clocks Frequency Deviation from Spec” column in Table 136 is within of this +/- 5% tolerance.

Table 136: SSP I^2S Audio Frequencies

Audio Frequencies Spec				Fractional Divider Programming		
Sync clock [kHz]	Bit clock [MHz] [Sync clock * 64]	Sys clock [MHz] [Bit Clock * 4] SCDx8=0	Sys clock [MHz] [Bit Clock * 8] SCDx8=1	N (Numerator)	M (Denominator)	Audio Clocks Frequency Deviation from Spec
48	3.072	12.288	24.576	1625	64	0%
44.1	2.822	11.2896	22.5792	1769	64	0.017%
16	1.024	4.096	8.192	4875	64	0%

Note

An I^2S frame is composed of 32-bits right channel (16-bits data followed by 16-bits zeroes) plus 32-bits left channel (16-bits data followed by 16-bits zeroes. The bit clock is Sync clock * 64 bits. The Sys Clock is bit clock * 4 (when SCDx8=0). For example, for 48 kHz sync clock, bit clock is 3.072 MHz, sys clock is 12.288 MHz.

5 AC ’97 Controller Unit

This chapter describes the Audio CODEC ’97 (AC ’97) controller included in the PXA3xx Processor Family. The PXA32x processor, PXA31x processor, and PXA30x processor are referred to as simply “the processor” in this volume when no differentiation is needed.

5.1 Overview

The AC ’97 controller supports the *Audio CODEC '97 Component Specification*[1], Revision 2.3, features listed in Section 5.2. 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 PXA3xx Processor Family 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 2.3.

There are no differences in AC ’97 controller functionality among the PXA3xx Processor Family components.

5.2 Features

The AC ’97 controller supports the following AC ’97 features:

- Independent channels for stereo pulse code modulated (PCM) In, stereo PCM Out, surround PCM Out, center/LFE PCM Out, MODEM Out, MODEM-In and mono Mic-In

  All channels support only 16-bit samples in hardware. Samples less than 16 bits are supported through software. All PCM samples are paired together when written to or read from registers or FIFOs and the pairs are treated like 32-bit values.
- Multiple sample rate AC ’97 2.3 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
- Supports one primary CODEC and one secondary CODEC.
- Optional AC97_SYSCLK output (support for CODECs without oscillators or crystals)

The AC ’97 controller does not support the following optional AC ’97 Rev 2.3 features:

- Optional double-rate sampling (n+1 sample for PCM L, R)
- 18- and 20-bit sample lengths.

5.3 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

1. The AC ‘97 specification is available from *Intel at* http://www.intel.com/support/motherboards/desktop/sb/cs-025406.htm

command/status information are communicated over the AC-link. The AC-link uses the multi-function pins (MFPs). Software must reconfigure the MFPs to use them as the AC-link. The AC-link pins are listed and described in Table 137.

Table 137: External Interface to CODECs

Name	Type	Description
AC97_nACRESET	Output	Asynchronous, active-low CODEC reset. The CODEC registers are reset when AC97_nACRESET 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_IN0	Input	Serial audio input data, from primary CODEC
AC97_SDATA_IN1	Input	Serial audio input data, from secondary CODEC 01
AC97_SYSCLK	Output	Optional clock output (nominally 24.576 MHz)

5.4 Operation

Access to the AC ’97 controller unit is through either programmed I/O (PIO) or the DMA controller. The processor use programmed I/O instructions to access the AC ’97 controller unit. The four types of registers can be accessed by the processor:

- AC ’97 controller unit registers: Accessible at 32-bit boundaries and are listed in Section 5.5.
- 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 ACUNIT supplies access to the CODEC registers by mapping them to its 32-bit address domain boundary. Section 5.5.24 describes the mapping from the 32-bit to 16-bit boundary. A Write or Read operation targeting these registers is sent across the AC-link.
- Modem CODEC GPIO register: If the ACUNIT is connected to a modem CODEC, the CODEC GPIO register can also be accessed. The CODEC GPIO register has an access address of 0x0054 within the CODEC domain. While the GPIO Write operation goes across the AC-link, a Read does not. The register contents are updated continuously into a register in the controller domain whenever a frame is received from the CODEC. So, when the processor tries to read the CODEC GPIO register, this shadow register is read instead.
- AC ’97 controller unit FIFO data: AC: The ACUNIT has four Transmit FIFOs and three Receive FIFOs. The four Transmit FIFOs are comprised of one for PCM-out, one for PCM-surround-out, one for PCM-LFE/center-out and one for modem-out. The three Receive FIFOs are comprised of one for PCM-in, one for modem-in, and one for Mic-in. The Transmit FIFOs are written by writing to one of the following: PCM Data register (PCDR), PCM Surround Data register (PCSDR), PCM Center/LFE Data register (PCCLDR) or the Modem Data register (MODR). A Write to any register updates a Transmit FIFO entry, which is pointed to by the respective Write pointer. Receive FIFO entries are read by reading the PCDR, the MODR or the MCDR registers. A Read returns data from the Receive FIFO based on the Read pointer.

DMA controller accesses are made through the AC ’97 controller unit FIFO 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 ACUNIT 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 ACUNIT interrupt requests when they are enabled.

DMA requests or PIO interrupts are made for the following conditions. Do not set up a FIFO in the ACUNIT for both DMA and PIO access.

- PCM FIFO service requests (transmit and receive) are made when the PCM transmit and Receive FIFOs are half full.
- Modem FIFO service requests (transmit and receive) are made when the modem transmit and Receive FIFOs are half full.
- Mic-in FIFO service requests (receive) are made when the Mic-in Receive FIFO is half full.
- PCM surround FIFO service requests (transmit) are made when the PCM surround Transmit FIFO is half full.
- PCM center/LFE FIFO service requests (transmit) are made when the PCM center/LFE Transmit FIFO is half full.

The unit signals an end-of-chain interrupt when the DMA signals an end of chain (EOC) and data is present in a Receive FIFO. Read the remaining data out of the FIFOs and then write to the appropriate status register (Table 151, PCMISR Bit Definitions, on page 374, Table 154, MCSR Bit Definitions, on page 377 or Table 156, MISR Bit Definitions, on page 380) to clear the interrupt.

5.4.1 Initialization

The AC '97 CODEC and ACUNIT circuitry are reset on power-up with the AC97_nACRESET signal, which remains asserted (low) until either the audio or modem driver sets the Global Control register (GCR) bit cold reset# (GCR[nCRST]).

The following procedure describes the required steps to initialize the ACUNIT.

1. Program the MFPRx registers to assign the correct alternate functions for the various ACUNIT ports. Refer to Section 5.3 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. De-assert AC97_nACRESET by setting register bit GCR[nCRST]. De-asserting AC97_nACRESET has these effects:
   a) All registers (AC '97 controller and CODEC) are placed in their reset state.
   b) Frames filled with zeroes are transmitted because the Transmit FIFO is still empty. However, this situation does not cause an error condition because nothing is tagged as valid.
   c) The ACUNIT 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]) or secondary-ready interrupt-enable (GCR[S1RDY_IE]). Software can also poll the primary-CODEC-ready and secondary-CODEC-bits (GSR[PCRDY] and GSR[S1CRDY], respectively).
5. Software responds to primary/secondary-ready interrupts by triggering the DMA or programmed IO operation. The ACUNIT triggers a PCM-out FIFO service request. DMA or programmed IO responds by filling up the Transmit FIFOs.
6. The ACUNIT continues to transmit zeroes until the Transmit FIFO is half full. Once half full, valid FIFO data is sent across the AC-link.

Note

When AC97_nACRESET is de-asserted, a read of the CODEC Mixer register 0x00 returns the type of hardware that resides in the CODEC. If the CODEC is not present or if AC ’97 is not supported, the ACUNIT does not set the CODEC-ready bit, GCR[PCRDY] for the primary CODEC or GCR[S1CRDY] for the secondary CODEC.

5.4.2 Trailing Bytes and Clean Shutdown

Transmit trailing bytes: The AC ’97 controller does not support trailing bytes. Transmit buffers must be a multiple of 32 bytes. Software must satisfy this condition or remaining data in a Transmit FIFO is never transmitted. Software must pad a Transmit buffer with zeros if its size is less than a multiple of 32 bytes.

Receive trailing bytes: The ACUNIT stops recording data for that CODEC when the CODEC stops transmitting valid data (as defined by valid tag bits). The ACUNIT holds the trailing bytes in the Receive FIFO if the data is not a multiple of 32 bytes. ACUNIT does not make a FIFO service request for these trailing bytes. These bytes can be discarded by software initiating a clean shutdown (by setting GCR[ACOFF]).

Clean shutdown: Setting GCR[ACOFF] shuts down the ACUNIT. The ACUNIT de-asserts any active Transmit and Receive FIFO service request (DMA or programmed IO interrupt) and stops transmitting and receiving data. The ACUNIT then discards any remaining data in the TX and RX FIFOs and drives AC97_SYNC and AC97_SDATA_OUT to a logic level low. When all FIFOs are empty and the sync and sdata_out are pulled low, the ACUNIT sets GSR[ACOFFD] to indicate a complete clean shutdown.

Software can detect if the clean shutdown has finished by reading GSR[ACOFFD]. Clearing GCR[nCRST] causes an immediate shutdown of the AC-link and reset of the ACUNIT circuitry. The GCR[nCRST] bit supersedes GCR[ACOFF] and therefore prevents a clean shutdown if set during or before the shutdown sequence.

5.4.3 Operational Flow for Accessing CODEC Registers

Software accesses the CODEC registers by translating a 7-bit CODEC address to a 32-bit processor address. Refer to Section 5.5.24 for details regarding the address translation. There is no need to go external on the AC-link if the access is a Read to the Modem GPIO Pin Status register at offset 0x54 because the ACUNIT captures GPIO status from Slot 12 every frame and stores the data in a local register. Therefore, a read of this address returns valid data, and there is no need to do anything else. Use the following procedure for all other access:

- Software must first lock the AC-link by reading the CODEC Access register (CAR). The AC-link is free if CAR[CAIP] (CODEC access in progress) is clear. Refer to Table 149 for details.
- The read access to CAR automatically sets CAR[CAIP]. The ACUNIT clears CAIP after the CODEC-Write or CODEC-Read command is sent out on the AC-link. Software can also clear CAR[CAIP] by writing 0b0 to it.
- After locking the AC-link, software can either write or read a CODEC register using the appropriate processor physical address.
- The ACUNIT sets GSR[CDONE] after the completion of a CODEC Write operation. Refer to Table 148 for details. Software clears this bit by writing a 0b1 to the CDONE bit location.
- Software can read a CODEC register by using these steps:
  a) Software first issues a “dummy” Read to the CODEC register. The ACUNIT responds to this Read operation with invalid data. The ACUNIT then initiates the read access across the AC-link.

b) The ACUNIT sets GSR[SDONE] when the CODEC Read operation completes. Refer to Table 148 for details. Software clears this bit by writing 0b1 to GSR[SDONE].
c) Software now repeats the Read operation as in Step 1. The ACUNIT now returns the data sent by the CODEC. The second Read operation also initiates a read access across the AC-link.
d) The ACUNIT times out the Read operation if the CODEC fails to respond in four AC97_SYNC frames. In this situation, the second Read operation returns a timed-out data value of 0x0000_FFFF.

5.4.4 Clocks and Sampling Frequencies

By default, the ACUNIT transmits and receives data at a sampling frequency of 48 kHz. It can, however, sample data at frequencies of less than 48 kHz if the CODEC supports "on-demand" slot requests. The CODEC in this instance executes a certain algorithm and informs the controller not to transmit valid data in certain frames. For example, if the controller sends out 480 frames, and the CODEC instructs the controller to send valid data in 441 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 frames mixed in between the valid frames.

All data transfers across the AC-link are synchronized to the rising edge of the AC97_SYNC signal. The ACUNIT divides the AC97_BITCLK by 256 to generate the AC97_SYNC. This calculation yields a 48-kHz AC97_SYNC signal whose period defines a frame. Data is transitioned on the AC-link on every rising edge of AC97_BITCLK, and subsequently sampled on the receiving side of AC-link on each immediately following falling edge of AC97_BITCLK. See Figure 56 and Figure 57 for timing diagrams.

An optional clock output is available via the AC97_SYSCLK signal for CODECs that lack oscillators or crystals. Out of reset, this operates at 24.576 MHz. This frequency can be modified by altering the value of the AC97_DIV register in the BCCU. Refer to the Slave Clock Control Unit chapter in the *PXA 3xx Processor Family Vol. I: System and Timer Configuration Developers Manual* for information related to this register. To configure AC97_SYSCLK to operate at 13 MHz, configure AC97_DIV[NUMERATOR] to be 0x0300 and AC97_DIV[DENOMINATOR] to be 0x020.

5.4.5 FIFOs

The AC '97 controller unit has seven FIFOs:

- PCM Transmit FIFO, with sixteen 32-bit entries
- PCM Receive FIFO, with sixteen 32-bit entries
- Modem Transmit FIFO, with sixteen 16-bit entries
- Modem Receive FIFO, with sixteen 16-bit entries
- Mic-in Receive FIFO, with sixteen 16-bit entries
- PCM surround Transmit FIFO, with sixteen 32-bit entries
- PCM center/LFE Transmit FIFO, with sixteen 32-bit entries

A Receive FIFO triggers a FIFO service request when the FIFO has eight or more entries. A Transmit FIFO triggers a FIFO service request when it holds less than eight entries. A Transmit FIFO must be half-full, that is, filled with eight entries before any data is transmitted across the AC-link.

5.4.5.1 Transmit FIFO Errors

Channel-specific status bits are updated during transmit-underrun or transmit-overrun conditions, and if enabled, these bits trigger interrupts. Refer to Table 150 and Table 155 for details regarding the status bits.

If a transmit-underrun occurs, the last valid sample is sent out continuously across the AC-link and pointers do not increment. Transmit-underrun occurs under these conditions:

- Valid transmit data is still available in memory, but PIO or DMA starves the Transmit FIFO as it is busy servicing other higher priority peripherals.
- PIO or DMA has transferred all valid data from memory to the Transmit FIFO. This task prompts for the last valid sample to be repeated across the AC-link until the ACUNIT is turned off by asserting AC97_nACRESET.

If a transmit-overrun occurs, the data in the Transmit FIFO is overwritten and pointers do not increment. Transmit-overrun occurs when PIO tries to update a Transmit FIFO when it is already full, resulting in data loss. 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.

5.4.5.2 Receive FIFO Errors

Channel-specific status bits are updated during receive overrun conditions, and if enabled, these bits can trigger interrupts. Refer to Table 151, Table 154 and Table 156 for details regarding the status bits.

If a receive-overrun occurs, the incoming data from the AC-link is not written into the FIFO (data is lost) and pointers do not increment. Receive underruns occur when DMA does not have sufficient priority to handle the bandwidth and latency requirements for AC97.

If a receive-underrun occurs, invalid data is read and pointers do not increment. This situation occurs only if PIO tries to read the Receive FIFO when it is empty.

5.4.5.3 FIFO And Register Interaction

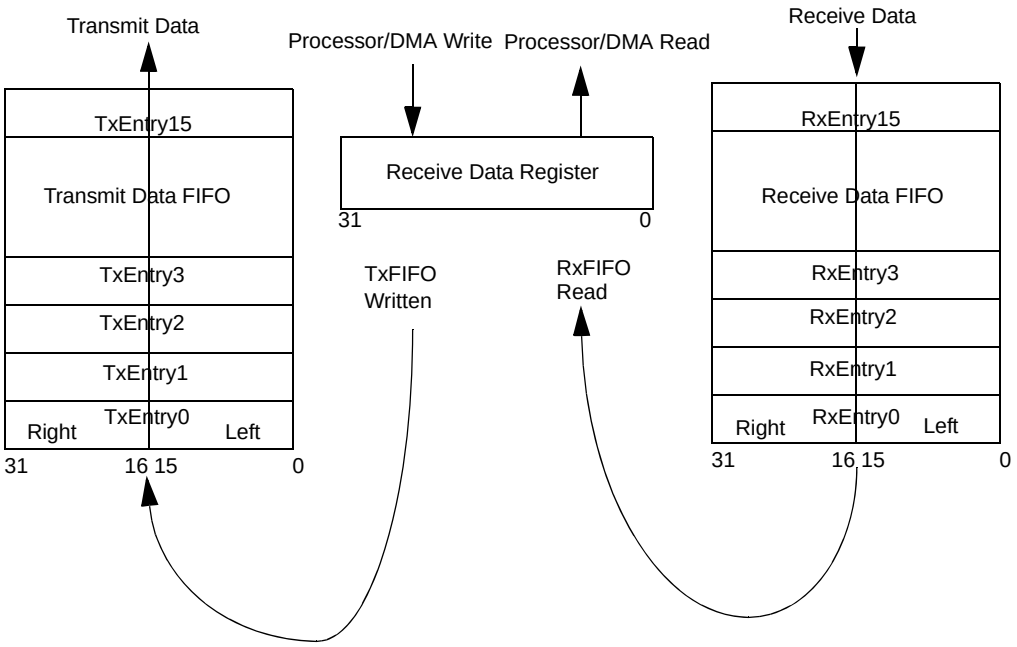
Three types of FIFOs are in the processor AC ‘97 controller:

- 32-bit sample size receive and Transmit FIFO
- 16-bit sample size receive and Transmit FIFO
- 16-bit sample size receive-only FIFO

While the operation of each type is similar, there are some differences in how the data is passed between the transmit/Receive FIFOs and the data register. All AC ’97 data samples are 16-bits in length, and the data registers within the AC ’97 controller contain either 16 bits or 32 bits of valid data.

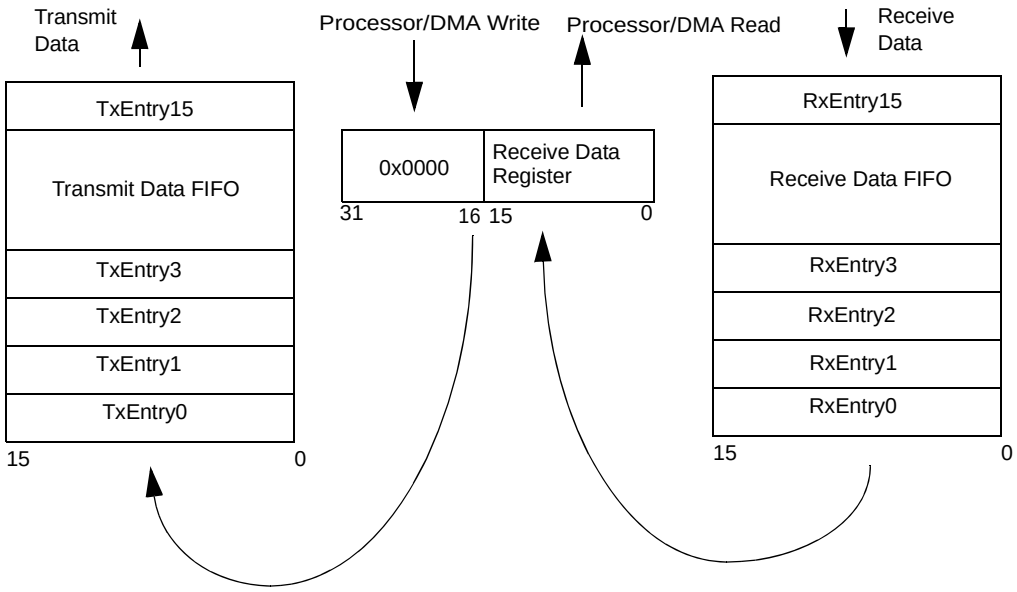
For data registers that contain two samples, the FIFOs are 32 bits wide and all 32 bits are written to or read from whenever the data register is accessed. Refer to Table 53 for a conceptual diagram of this operation. These registers are PCDR, PCCLDR and PCSDR.

Figure 53: 32-Bit Sample Size Transmit and Receive FIFO Operation



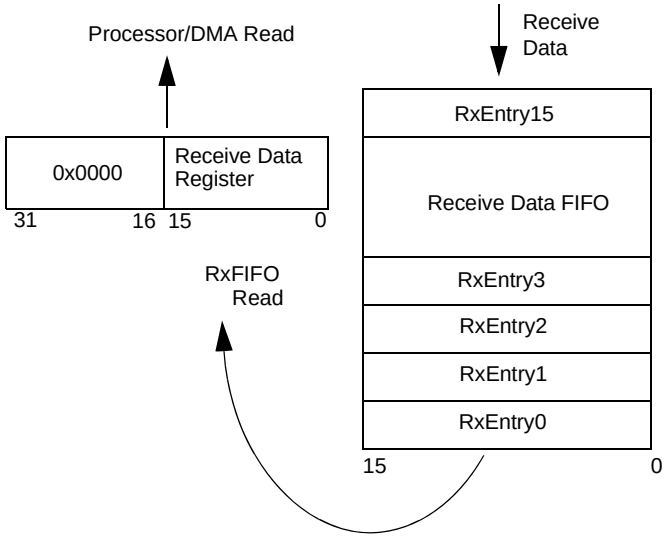
The MODEM data register (Table 161) contains only a single sample. The Receive and Transmit FIFOs are 16 bits wide and only 16 bits are written to or read from whenever the data register is accessed. The upper 16 bits of the data register are not used when either reading or writing the FIFOs. Refer to Table 54 for a conceptual diagram of this operation.

Figure 54: 16-Bit Sample Size Transmit and Receive FIFO Operation



The Mic-In data register (Table 161) contains only a single sample. The Receive FIFO is 16 bits wide and only 16 bit samples are written to the data register. The upper 16 bits of the data register are not used when either reading the FIFO. Refer to Figure 55 for a conceptual diagram of this operation.

Figure 55: 16-Bit Sample Size Receive-Only FIFO Operation



5.4.6 Interrupts

Table 138 show the status bits that, if enabled, interrupt the processor.

Table 138: Interrupts

Interrupt	Description
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
PCM-Surround-out FIFO error	Surround Transmit FIFO overrun or underrun error
PCM-Center/LFE FIFO error	Center/LFE 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
PCM-Surround-out FIFO Service Request	Surround Transmit FIFO contains less than 32 bytes
PCM-Center/LFE FIFO Service Request	Center/LFE Transmit FIFO contains less than 32 bytes
Modem CODEC GPIO Status Change Interrupt	Interrupts the CPU if bit0 of slot12 is set; indicates that one of the bits in the modem CODEC GPIO register changed.
Primary CODEC Resume Interrupt	A Status register bit is set once the primary CODEC resumes from a lower power mode. This bit is cleared by software writing 0b1 to it.
Secondary ID 01 CODEC Resume Interrupt	A Status register bit is set once the secondary CODEC resumes from a lower power mode. This bit is cleared by software writing 0b1 to it.
CODEC Command Done Interrupt	Interrupts the CPU when a CODEC register command has completed. This bit is cleared by software writing 0b1 to it.
CODEC Status Done Interrupt	Interrupts the CPU when a CODEC register status address and data reception has completed. This bit is cleared by software writing 0b1 to it.

Table 138: Interrupts (Continued)

Interrupt	Description
Primary CODEC Ready Interrupt	A status register bit is set once the primary CODEC is ready. The CODEC signals the readiness by setting bit15 of slot0 on the input frame. Clear this bit by writing 0b1 to GSR[PCRDY].
Secondary CODEC ID 01 Ready Interrupt	A status register bit is set once the secondary CODEC is ready. The code signals the readiness, by setting bit15 of slot0 on the input frame. Clear this bit by writing 0b1 to GSR[S1CRDY].
EOC interrupt	A status bit is set when there is data in a Receive FIFO when the DMA signals an end of chain. The unit signals an end of chain interrupt when this occurs. The interrupt is cleared when software reads remaining data out of the FIFOs and Writes to the appropriate status register (refer to Section 5.4).

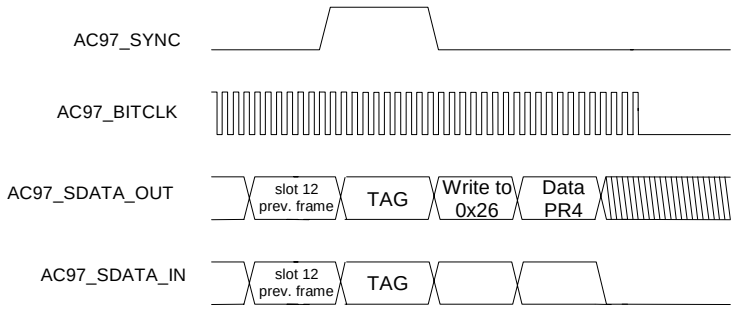
5.4.7 AC-Link Low-Power Mode

Power down the AC-link prior to entering any processor Low-power mode (S0/D1/C2, S0/D2/C2, S2/D3/C4 or S3/D4/C4) to minimize power consumption and also prevent data loss in state-retaining Low-power modes.

5.4.7.1 Powering Down the AC-Link

The AC-link signals enter a low-power mode when the AC '97 CODEC Powerdown register (0x26) bit PR4 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 56, "AC-Link Powerdown Timing. Refer to section 10.6.22.1 Accessing CODEC Registers and Table 42 for information on accessing the AC '97 CODEC Powerdown register.

Figure 56: AC-Link Powerdown Timing



Note: AC97_BITCLK not to scale

As with all software CODEC register accesses, the ACUNIT transmits the Write to the Powerdown register (0x26) over the AC-link. Do not set up the ACUNIT to transmit data in slots other than 1 and

2 along with the Write to Powerdown register bit PR4 (data 0x1000). Once the CODEC processes the request, it immediately transitions AC97_BITCLK and AC97_SDATA_IN to a logic low level.

The ACUNIT drives AC97_SYNC and AC97_SDATA_OUT to a logic-low level after setting GCR[ACOFF]. Any data remaining in the TXFIFO or RXFIFO is discarded. The ACUNIT maintains AC97_nACRESET high while GCR[ACOFF] is set.

The following procedure describes the required steps to initiate a powerdown:

1. Write 0x26 to initiate powerdown.
2. Wait for GSR[CDONE] to be set, indicating the command has completed.
3. Set GCR[ACOFF] to shut off the AC-link.
4. Set GCR[CLKBPB] to enable the internal AC ’97 clock.
5. Wait for GSR[ACOFFD] to be set, indicating the shutdown is complete.
6. Set GCR[FRCRST] to assert reset to the AC ’97 controller.
7. Clear GCR[FRCRST] to de-assert reset to the AC ’97 controller.
8. Clear GCR[CLKBPB] to disable the internal AC ’97 clock.

Note

If a resume event occurs during this procedure, the CODEC must not be awakened until GSR[ACOFFD] is set.

5.4.8 Waking Up the AC-Link

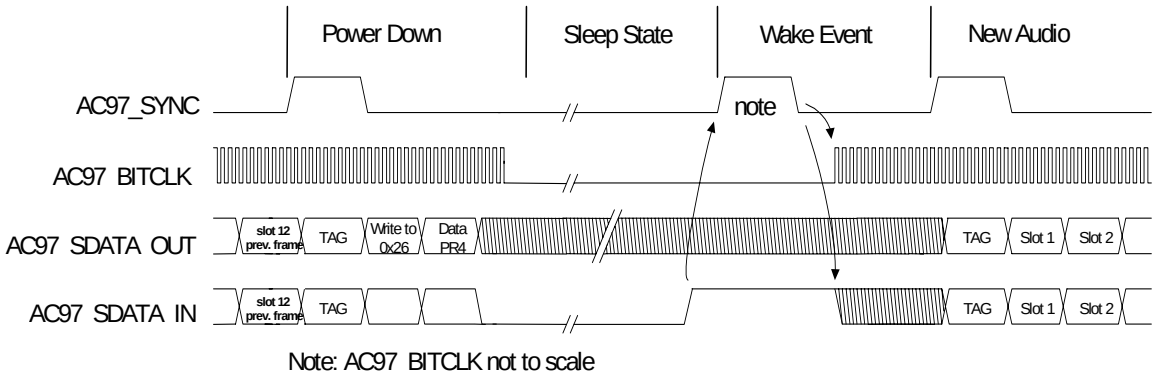
5.4.8.1 Wakeup Triggered by CODEC

A CODEC drives its AC97_SDATA_IN to a logic-high level to wake up the AC-link (for example, in response to an event such as caller-ID or wake-on-ring operation of a MODEM). The rising edge only triggers a wakeup if that CODEC resume-enable bit is set (GCR[PRES_IE] for the primary CODEC; GCR[S1RES_IE] for the secondary CODEC). The slave power-management unit detects the AC ’97 wakeup event (AC97_SDATA_IN high for more than 1µs) via MFPR wake-up operation. The processor then initiates either a cold- or warm-reset sequence of the CODEC. The ACUNIT uses a warm reset to wake the primary CODEC.

The CODEC detects a warm reset in the absence of AC97_BITCLK when AC97_SYNC is high for more than 1 µs. The CODEC must wait until it samples AC97_SYNC low before it can start AC97_BITCLK. The CODEC that has signaled the wakeup event must keep its AC97_SDATA_IN high until it detects the completion of warm reset, and only then transition its AC97_SDATA_IN low.

Figure 57 shows the AC-link timing for a wake-up triggered by a CODEC.

Figure 57: AC97_SDATA_IN Wake Up Signaling



Note

Refer to the *PXA3xx Processor Family Electrical, Mechanical, and Thermal Specification* for the minimum AC97_SDATA_IN wake-up pulse width.

5.4.8.2 Wake Up Triggered by AC ’97 Controller

AC-link protocol provides for a “cold AC ’97 reset” and a “warm AC ’97 reset.” A warm reset maintains the value of the AC ’97 controller registers. A cold reset resets the value of the AC ’97 controller registers to their reset values.

Once powered down, re-activation of the AC-link via re-assertion of the AC97_SYNC signal must not occur for a minimum of four audio-frame times following the frame in which the powerdown was triggered. When AC-link powers up, it indicates readiness via the CODEC-ready bit (Input Slot 0, Bit 15).

5.4.8.2.1 Cold AC ‘97 Reset

A cold reset is achieved by asserting the AC97_nACRESET pin using GCR[nCRST]. Asserting and de-asserting AC97_nACRESET activates AC97_BITCLK (if supplied by the CODEC) and AC97_SDATA_OUT. All AC ’97 control registers are initialized to their default power on reset values.

If a cold reset is performed “on-the-fly” without a preceding powerdown, then these few extra steps are required to reset the internal logic of the AC ’97 controller:

1. Clear GCR[nCRST] to initiate cold reset.
2. Set GCR[CLKBPB] to enable the internal AC ’97 clock.
3. Clear GCR[CLKBPB] to disable the internal AC ’97 clock.
4. Set GCR[nCRST] to de-assert the AC97_nACRESET reset signal to the CODEC. Do not perform this step until AC97_nACRESET has been asserted for at least the minimum amount of time required in the AC ’97 specification.

Note

The AC-link is placed and held in cold reset automatically when the processor exits from S2/D3/C4 or S3/D4/C4 because the AC ’97 controller registers are reset and GCR[nCRST] is cleared after exiting these power modes. GCR[nCRST] must be set before before attempting to use the AC-link.

5.4.8.2.2 Warm AC ’97 Reset

A warm AC ’97 reset re-activates the AC-link without altering the current AC ’97 register values. A warm reset is initiated by writing a 0b1 to GCR[WRST], which drives AC97_SYNC high for a minimum of 1µs in the absence of AC97_BITCLK. The warm reset is not initiated if software attempts to perform a warm reset (by writing a 0b1 to GCR[WRST]) while AC97_BITCLK is running.

Within normal audio frames, AC97_SYNC is a synchronous CODEC input. But, in the absence of AC97_BITCLK, AC97_SYNC is treated as an asynchronous input used in the generation of a warm reset to the CODEC.

The CODEC must not respond with the activation of AC97_BITCLK until AC97_SYNC has been sampled low again by the CODEC, which precludes the false detection of a new audio frame.

Note

The polarity of the bitfields in GCR that initiate warm and cold resets are opposite. A warm reset is initiated simply by setting GCR[WRST]. A cold reset requires that GCR[nCRST] be cleared, a minimum reset time delay observed, and then GCR[nCRST] must be set.

5.5 Register Descriptions

All ACUNIT and CODEC registers are mapped in addresses ranging from 0x4050_0000 through 0x405F_FFFF. All ACUNIT registers are 32-bit addressable. Though a CODEC has up to sixty-four 16-bit registers that are 16-bit addressable, they are accessed via a 32-bit address map and translated to 16-bit for the CODEC.

The processor can access these registers:

- Global registers: The ACUNIT has three global registers: Status, Control, and CODEC access registers that are common to both the audio and modem domains
- Channel-Specific Audio registers*:* ACUNIT registers refer to PCM-out, PCM-in, PCM-surround-out, PCM-Center/LFE and Mic-in channels
- Channel-Specific Modem registers: ACUNIT registers refer to modem-out and modem-in channels
- Audio CODEC registers: CODEC registers for the primary and secondary audio CODEC
- Modem CODEC registers: CODEC registers for the primary and secondary modem CODEC

The processor and the DMA controller can access the channel-specific data registers. These registers are for FIFO accesses with one each for the PCM, MODEM, and Mic-in FIFOs. 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 one to clear). The status can come in at any time, even when the bit is set or in the process of a software clear. If software clears the bit during the time the CODEC status updates the bit, the CODEC status event has higher priority.

5.5.1 Register Summary

All AC'97 controller registers are word addressable (32 bits wide) and therefore increment in units of 0x0000_0004. All the registers in the CODEC are half word addressable (16 bits wide), and increment in units of 0x00002. Both of these register sets are mapped in the address range of 0x4050_0000 through 0x405F_FFFF.

Table 139: Register Mapping Summary

Physical Address	Description	Page
0x4050_0000	PCM-Out Control Register (POCR)	page 363
0x4050_0004	PCM-In Control Register (PCMICR)	page 364
0x4050_0008	Mic-In Control Register (MCCR)	page 367
0x4050_000C	Global Control Register (GCR)	page 360
0x4050_0010	PCM-Out Status Register (POSR)	page 374
0x4050_0014	PCM-In Status Register (PCMISR)	page 374
0x4050_0018	Mic-In Status Register (MCSR)	page 377
0x4050_001C	Global Status Register (GSR)	page 369
0x4050_0020	CODEC Access Register (CAR)	page 373
0x4050_0024	PCM-Surround-Out Control Register (PCSCR)	page 365
0x4050_0028	PCM Surround-Out Status Register (PCSSR)	page 376
0x4050_002C	PCM Surround Out Data Register (PCSDR)	page 381
0x4050_0030	PCM Center/LFE Control Register (PCCLCR)	page 366
0x4050_0034	PCM Center/LFE Status Register (PCCLSR)	page 377
0x4050_0038	PCM Center/LFE Data Register (PCCLDR)	page 382
0x4050_003C	reserved	
0x4050_0040	PCM Data Register (PCDR)	page 381
0x4050_0044 through 0x4050_005C	reserved	
0x4050_0060	Mic-In Data Register (MCDR)	page 382

Table 139: Register Mapping Summary (Continued)

Physical Address	Description	Page
0x4050_0064 through 0x4050_00FC	reserved	
0x4050_0100	MODEM-Out Control Register (MOCR)	page 367
0x4050_0104	reserved	
0x4050_0108	MODEM-In Control Register (MICR)	page 368
0x4050_010C	reserved	
0x4050_0110	MODEM-Out Status Register (MOSR)	page 379
0x4050_0114	reserved	
0x4050_0118	MODEM-In Status Register (MISR)	page 380
0x4050_011C through 0x4050_013C	reserved	
0x4050_0140	MODEM Data Register (MODR)	page 383
0x4050_0144 through 0x4050_01FC	reserved	
(0x4050_0200 through 0x4050_02FC) with all in increments of 0x00004	Primary Audio CODEC registers	page 384
(0x4050_0300 through 0x4050_03FC) with all in increments of 0x00004	Secondary Audio CODEC ID 01 registers	page 384
(0x4050_0400 through 0x4050_04FC) with all in increments of 0x0000_0004	Primary Modem CODEC registers	page 384
(0x4050_0500 through 0x4050_05FC) with all in increments of 0x00004	Secondary Modem CODEC ID 01 registers	page 384

5.5.2 Global Control Register (GCR)

Table 140 shows the Global Control register.

Table 140: GCR Bit Definitions

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
	CLKBPB	FRCRST	reserved					nDMAEN	reserved				CDONE_IE	SDONE_IE	reserved								S1RDY_IE	PRDY_IE	reserved		S1RES_IE	PRES_IE	ACOFF	WRST	nCRST	GPI_IE
Reset	0	0	?	?	?	?	?	0	?	?	?	?	0	0	?	?	?	?	?	?	?	?	0	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
31	R/W	CLKBPB	Internal Clock Enable: 0 = Disable internal clock multiplexing in the AC97 controller. This is the normal mode of operation. 1 = Enable internal clock multiplexing in the AC97 controller. This bit should only be set when performing a power down or cold reset. See Section 5.4.7.1 for details.
30	R/W	FRCRST	Force AC97 Controller Reset: 0 = The AC97 controller reset is not forced. 1 = The AC97 controller reset is forced to 1. This bit should only be set when performing a power down. See Section 5.4.7.1 for details.
29:25	—	—	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 when using PIO.
23:20	—	—	reserved
19	R/W	CDONE_IE	Command Done Interrupt Enable: Controls whether the controller triggers an interrupt to the CPU after sending the command address and data to the CODEC. 0 = Interrupt is disabled 1 = Interrupt is enabled
18	R/W	SDONE_IE	Status Done Interrupt Enable: Controls wether the controller triggers an interrupt to the CPU after receiving the status address and data from the CODEC. 0 = Interrupt is disabled 1 = Interrupt is enabled
17:10	—	—	reserved
9	R/W	S1RDY_IE	Secondary CODEC Ready Interrupt Enable: Controls whether an interrupt occurs when the secondary CODEC sends the CODEC READY bit on the AC97_SDATA_IN1 pin. 0 = Interrupt is disabled 1 = Interrupt is enabled

Table 140: GCR Bit Definitions (Continued)

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
	CLKBPB	FRCRST	reserved					nDMAEN	reserved				CDONE_IE	SDONE_IE	reserved								S1RDY_IE	PRDY_IE	reserved		S1RES_IE	PRES_IE	ACOFF	WRST	nCRST	GPI_IE
Reset	0	0	?	?	?	?	?	0	?	?	?	?	0	0	?	?	?	?	?	?	?	?	0	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
8	R/W	PRDY_IE	Primary CODEC Ready Interrupt Enable: Controls whether an interrupt occurs when the primary CODEC sends the CODEC READY bit on the AC97_SDATA_IN0 pin. 0 = Interrupt is disabled 1 = Interrupt is enabled
7:6	—	—	reserved
5	R/W	S1RES_IE	Secondary Resume Interrupt Enable: Controls whether an interrupt occurs when the secondary CODEC causes a resume event on the AC-link. 0 = Interrupt is disabled 1 = Interrupt is enabled
4	R/W	PRES_IE	Primary Resume Interrupt Enable: Controls whether an interrupt occurs when the primary CODEC causes a resume event on the AC-link. 0 = Interrupt is disabled 1 = Interrupt is enabled
3	R/W	ACOFF	AC-Link Shut Off: Shuts down the AC97 controller. This does not affect the AC-link or the CODEC. This bit must be cleared before resuming AC 97 operation. Refer to Section 5.4.7.1 for more details of powering down the AC-link and the AC97 controller. 0 = Do not shut down the AC '97 controller. 1 = Shut down the AC '97 controller.
2	R/W	WRST	AC97 Warm Reset: Initiates a warm reset on the AC-link. If AC97_BITCLK is running Writes to this bit are ignored and no warm resets are performed. Refer to Section 5.4.7.1 for details on stopping AC97_BITCLK. 0 = Do not initiate a warm reset 1 = Initiate a warm reset **NOTE:** This bit is self-clearing that is, it remains set until the reset completes and AC97_BITCLK is seen on the AC-link after which it clears itself.

Table 140: GCR Bit Definitions (Continued)

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
	CLKBPB	FRCRST	reserved					nDMAEN	reserved				CDONE_IE	SDONE_IE	reserved								S1RDY_IE	PRDY_IE	reserved		S1RES_IE	PRES_IE	ACOFF	WRST	nCRST	GPI_IE
Reset	0	0	?	?	?	?	?	0	?	?	?	?	0	0	?	?	?	?	?	?	?	?	0	0	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
1	R/W	nCRST	AC97 Cold Reset#: Initiates a cold reset when reset or written with 0b0. A cold reset requires that this bit be written with 0b0 and then written with 0b1. Refer to Section 5.4.8.2.1. All data in the controller and the CODEC is lost when a cold reset occurs. The value of this bit is not altered by an AC-link wake-up. The value of this bit is retained after exiting S0/D0CS/C0, S0/D1/C2 or S0/D2/C2. Therefore if this bit is set when entering these low power modes, a cold reset does NOT occur after exiting. The AC '97 controller registers are reset after exiting S2/D3/C4 or S3/D4/C4. Since the reset value of this bit is 0b0, the AC '97 automatically initiates a cold reset when exiting these power modes and must be set before using the AC-link. 0 = Initiate a cold reset. This asserts nAC97_RESET. 1 = Do not initiate a cold reset. This de-asserts nAC97_RESET. **NOTE:** Prior to asserting the cold reset, disable all AC '97 interrupts. The interrupts can be re-enabled after the cold reset has occurred.
0	R/W	GPI_IE	CODEC GPIO Interrupt Enable: This bit controls whether the change in status of any modem CODEC GPIO (as indicated by bit 0 of slot 12) causes an interrupt. 0 = GSR[GSCI] is set, but an interrupt is NOT generated. 1 = GSR[GSCI] is set, and an interrupt is generated.

5.5.3 PCM-Out Control Register (POCR)

The PCM-Out Control register is described in Table 141.

Table 141: POCR Bit Definitions

Physical Address 0x4050_0000 **POCR** **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																												FEIE	reserved	FSRIE	reserved
	Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	0	?

Bits	Access	Name	Description
31:4	—	—	reserved
3	R/W	FEIE	FIFO Error Interrupt Enable: Enables generation of the PCM Transmit FIFO Error interrupt. 0 = A PCM Transmit FIFO error does NOT generate an interrupt. 1 = A PCM Transmit FIFO error generates an interrupt. **NOTE:** POSR[FIFOE] is always set when a PCM Transmit Error occurs.
2	—	—	reserved
1	R/W	FSRIE	FIFO Service Request Interrupt Enable: Enables generation of the PCM Transmit FIFO Service Request interrupt. 0 = A PCM Transmit FIFO Service Request does NOT generate an interrupt. 1 = A PCM Transmit FIFO Service Request generates an interrupt. **NOTE:** POSR[FSR] is always set when a PCM Transmit Service Request occurs.
0	—	—	reserved

5.5.4 PCM-In Control Register (PCMICR)

The PCM-In Control register is described in Table 142.

Table 142: 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	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: Enables generation of the PCM Receive FIFO Error interrupt. 0 = A PCM Receive FIFO error does NOT generate an interrupt. 1 = A PCM Receive FIFO error generates an interrupt. **NOTE:** PCMISR[FIFOE] is always set when a PCM receive Error occurs.
2	—	—	reserved
1	R/W	FSRIE	FIFO Service Request Interrupt Enable: Enables generation of the PCM Receive FIFO Service Request interrupt. 0 = A PCM Receive FIFO Service Request does NOT generate an interrupt. 1 = A PCM Receive FIFO Service Request generates an interrupt. **NOTE:** PCMISR[FSR] is always set when a PCM receive Service Request occurs.
0	—	—	reserved

5.5.5 PCM-Surround-Out Control Register (PCSCR)

The PCM-Surround-Out Control register is described in Table 143.

Table 143: PCSCR Bit Definitions

Physical Address 0x4050_0024 — **PCSCR** — **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: Enables generation of the PCM Surround-Out FIFO error interrupt. 0 = A PCM Surround-Out FIFO error does NOT generate an interrupt. 1 = A PCM Surround-Out FIFO error generates an interrupt. **NOTE:** PCSSR[FIFOE] is always set when a PCM Surround-Out FIFO error occurs.
2	—	—	reserved
1	R/W	FSRIE	FIFO Service Request Interrupt Enable: Enables generation of the PCM Surround-Out FIFO service request interrupt. 0 = A PCM Surround-Out FIFO service request does NOT generate an interrupt. 1 = A PCM Surround-Out FIFO service request generates an interrupt. **NOTE:** PCSSR[FSR] is always set when a PCM Surround-Out FIFO Service Request occurs.
0	—	—	reserved

5.5.6 PCM Center/LFE Control Register (PCCLCR)

The PCM Center/LFE Control register is described in Table 144. Low frequency effects (LFE) are a possible source for FIFO errors and interrupt service requests and are controlled by the PCCLCR register.

Table 144: PCCLCR Bit Definitions

Physical Address 0x4050_0030	PCCLCR	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: Enables generation of the PCM Center/LFE FIFO error interrupt. 0 = A PCM Center/LFE FIFO error does NOT generate an interrupt. 1 = A PCM Center/LFE FIFO error generates an interrupt. **NOTE:** PCCLSR[FIFOE] is always set when a PCM Center/LFE FIFO error occurs.
2	—	—	reserved
1	R/W	FSRIE	FIFO Service Request Interrupt Enable: Enables generation of the PCM Center/LFE FIFO service request interrupt. 0 = A PCM Center/LFE FIFO service request does NOT generate an interrupt. 1 = A PCM Center/LFE FIFO service request generates an interrupt. **NOTE:** PCCLSR[FSR] is always set when a PCM Center/LFE FIFO Service Request occurs.
0	—	—	reserved

5.5.7 Mic-In Control Register (MCCR)

The Mic-In Control register is described in Table 145.

Table 145: MCCR Bit Definitions

Physical Address 0x4050_0008																															
MCCR																															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: Enables generation of the Mic-In FIFO error interrupt. 0 = A Mic-In FIFO error does NOT generate an interrupt. 1 = A Mic-In FIFO error generates an interrupt. **NOTE:** MCSR[FIFOE] is always set when a Mic-In FIFO error occurs.
2	—	—	reserved
1	R/W	FSRIE	FIFO Service Request Interrupt Enable: Enables generation of the Mic-In FIFO service request interrupt. 0 = A Mic-In FIFO service request does NOT generate an interrupt. 1 = A Mic-In FIFO service request generates an interrupt. **NOTE:** MCSR[FSR] is always set when a Mic-In FIFO Service Request occurs.
0	—	—	reserved

5.5.8 MODEM-Out Control Register (MOCR)

The MODEM-Out Control register is described in Table 146.

Table 146: MOCR Bit Definitions

Physical Address 0x4050_0100																															
MOCR																															AC97
User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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

Table 146: MOCR Bit Definitions (Continued)

Physical Address 0x4050_0100	MOCR	AC97

Bit	31	30	29	28	27	26	25	24	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
3	R/W	FEIE	FIFO Error Interrupt Enable: Enables generation of the MODEM-Out FIFO error interrupt. 0 = A MODEM-Out FIFO error does NOT generate an interrupt. 1 = A MODEM-Out FIFO error generates an interrupt. **NOTE:** MOSR[FIFOE] is always set when a MODEM-Out FIFO error occurs.
2	—	—	reserved
1	R/W	FSRIE	FIFO Service Request Interrupt Enable: Enables generation of the MODEM-Out FIFO service request interrupt. 0 = A MODEM-Out FIFO service request does NOT generate an interrupt. 1 = A MODEM-Out FIFO service request generates an interrupt. **NOTE:** MOSR[FSR] is always set when a MODEM-Out FIFO Service Request occurs.
0	—	—	reserved

5.5.9 MODEM-In Control Register (MICR)

The MODEM-In Control register is described in Table 147.

Table 147: MICR Bit Definitions

Physical Address 0x4050_0108	MICR	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
	reserved																												FEIE	reserved	FSRIE	reserved
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	?	0	?

Bits	Access	Name	Description
31:4	—	—	reserved

Table 147: MICR Bit Definitions (Continued)

Physical Address 0x4050_0108	MICR	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
3	R/W	FEIE	FIFO Error Interrupt Enable: Enables generation of the MODEM-In FIFO error interrupt. 0 = A MODEM-In FIFO error does NOT generate an interrupt. 1 = A MODEM-In FIFO error generates an interrupt. **NOTE:** MISR[FIFOE] is always set when a MODEM-In FIFO error occurs.
2	—	—	reserved
1	R/W	FSRIE	FIFO Service Request Interrupt Enable: Enables generation of the MODEM-In FIFO service request interrupt. 0 = A MODEM-In FIFO service request does NOT generate an interrupt. 1 = A MODEM-In FIFO service request generates an interrupt. **NOTE:** MISR[FSR] is always set when a MODEM-In FIFO Service Request occurs.
0	—	—	reserved

5.5.10 Global Status Register (GSR)

Table 148 shows the Global Status register.

Table 148: GSR Bit Definitions

Physical Address 0x4050_001C	GSR	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										**PSOINT**	**PCLINT**	**CDONE**	**SDONE**	**reserved**		**RCS**	**B3S12**	**B2S12**	**B1S12**	**S1RESINT**	**PRESINT**	**S1CRDY**	**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**	**0**	**0**

Bits	Access	Name	Description
31:22	—	—	reserved

Table 148: GSR Bit Definitions (Continued)

Physical Address 0x4050_001C																															
User Settings																															
Bit	31	30	29	28	27	26	25	24	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										PSOINT	PCLINT	CDONE	SDONE	reserved		RCS	B3S12	B2S12	B1S12	S1RESINT	PRESINT	S1CRDY	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	0	0

Bits	Access	Name	Description
21	R	PSOINT	PCM-Surround-Out Interrupt: This interrupt is generated if either: a PCM-surround FIFO Error interrupt (PCSSR[FIFOE] = 1) OR a PCM-surround FIFO Service Request interrupt (PCSSR[FSR] = 1) is generated. 0 = An interrupt has NOT occurred 1 = An interrupt has occurred **NOTE:** This bit is automatically cleared (by the ACUNIT) when either PCSSR[FIFOE] or PCSSR[FSR] are cleared.
20	R	PCLINT	PCM-Center/LFE Interrupt: This interrupt is generated if either: a PCM-center/LFE FIFO Error interrupt (PCCLSR[FIFOE] = 1) OR a PCM-center/LFE FIFO Service Request interrupt (PCCLSR[FSR] = 1) is generated. 0 = An interrupt has NOT occurred 1 = An interrupt has occurred **NOTE:** This bit is automatically cleared (by the ACUNIT) when either PCCLSR[FIFOE] or PCCLSR[FSR] are cleared.
19	Read/Write 1 to Clear	CDONE	Command Done: 0 = Reflects whether the controller has sent commands, address and data to the CODEC. Controller has NOT sent command, address and data to the CODEC 1 = Controller has sent command, address and data to the CODEC
18	Read/Write 1 to Clear	SDONE	Status Done: 0 = Reflects whether the controller has received status, address and data from the CODEC. Controller has NOT received status, address and data from the CODEC 1 = Controller has received status, address and data from the CODEC **NOTE:** SDONE is not set for Reads from special modem address 0x54 (GPIO status register).
17:16	—	—	reserved
15	Read/Write 1 to Clear	RCS	Read Completion Status: This bit indicates the status of CODEC Read completions. 0 = The CODEC Read completes normally. 1 = The CODEC Read results in a timeout.
14	R	B3S12	Bit 3 of Slot 12: Display Bit 3 of the most recent valid MODEM GPIO Status (slot 12)

Table 148: GSR Bit Definitions (Continued)

Physical Address 0x4050_001C																GSR										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										PSOINT	PCLINT	CDONE	SDONE	reserved		RCS	B3S12	B2S12	B1S12	S1RESINT	PRESINT	S1CRDY	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	0	0

Bits	Access	Name	Description
13	R	B2S12	Bit 2 of Slot 12: Display Bit 2 of the most recent valid MODEM GPIO Status (slot 12)
12	R	B1S12	Bit 1 of Slot 12: Display Bit 1 of the most recent valid MODEM GPIO Status (slot 12)
11	Read/Write 1 to Clear	S1RESINT	Secondary Resume Interrupt: Indicates whether a resume event has occurred on AC97_SDATA_IN1 0 = A resume event has NOT occurred 1 = A resume event has occurred
10	Read/Write 1 to Clear	PRESINT	Primary Resume Interrupt: Indicates whether a resume event has occurred on AC97_SDATA_IN0 0 = A resume event has NOT occurred 1 = A resume event has occurred
9	Read/Write 1 to Clear	S1CRDY	Secondary CODEC Ready: Reflects the state of the CODEC-ready bit in AC97_SDATA_IN1.
8	Read/Write 1 to Clear	PCRDY	Primary CODEC Ready: Reflects the state of the CODEC-ready bit in AC97_SDATA_IN0.
7	R	MCINT	Mic-In Interrupt: This interrupt is generated if any of the following: a Mic-In FIFO Error interrupt (MCSR[FIFOE] = 1) OR a Mic-In End Of Chain interrupt (MCSR[EOC] = 1) OR a Mic-In FIFO Service Request interrupt (MCSR[FSR] = 1) are generated. 0 = An interrupt has NOT occurred 1 = An interrupt has occurred **NOTE:** This bit is automatically cleared (by the ACUNIT) when any of MCSR[FIFOE], MCSR[EOC] or MCSR[FSR] are cleared.
6	R	POINT	PCM-Out Interrupt: This interrupt is generated if either: a PCM-out FIFO Error interrupt (POSR[FIFOE] = 1) OR a PCM-out FIFO Service Request interrupt (POSR[FSR] = 1) is generated. 0 = An interrupt has NOT occurred 1 = An interrupt has occurred **NOTE:** This bit is automatically cleared (by the ACUNIT) when either POSR[FIFOE] or POSR[FSR] are cleared.

Table 148: GSR Bit Definitions (Continued)

Physical Address 0x4050_001C																	GSR												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										PSOINT	PCLINT	CDONE	SDONE	reserved		RCS	B3S12	B2S12	B1S12	S1RESINT	PRESINT	S1CRDY	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	0	0

Bits	Access	Name	Description
5	R	PIINT	PCM-In Interrupt: This interrupt is generated if any of the following: a PCM-In FIFO Error interrupt (PCMISR[FIFOE] = 1) OR a PCM-In End Of Chain interrupt (PCMISR[EOC] = 1) OR a PCM-In FIFO Service Request interrupt (PCMISR[FSR] = 1) are generated. 0 = An interrupt has NOT occurred 1 = An interrupt has occurred **NOTE:** This bit is automatically cleared (by the ACUNIT) when any of PCMISR[FIFOE], PCMISR[EOC] or PCMISR[FSR] are cleared.
4	—	—	reserved
3	R	ACOFFD	AC-Link Shut Off Done: It is cleared when GCR[ACOFF] is cleared.This bit only has a meaning when GCR[ACOFF] is set. 0 = The AC-link still has valid data to transfer. 1 = The AC-link has been cleanly shutdown.
2	R	MOINT	Modem-Out Interrupt: This interrupt is generated if either: a Modem-out FIFO Error interrupt (MOSR[FIFOE] = 1) OR a Modem-out FIFO Service Request interrupt (MOSR[FSR] = 1) is generated. 0 = An interrupt has NOT occurred 1 = An interrupt has occurred **NOTE:** This bit is automatically cleared (by the ACUNIT) when either MOSR[FIFOE] or MOSR[FSR] are cleared.
1	R	MIINT	Modem-In Interrupt: This interrupt is generated if any of the following: a Modem-In FIFO Error interrupt (MISR[FIFOE] = 1) OR a Modem-In End Of Chain interrupt (MISR[EOC] = 1) OR a Modem-In FIFO Service Request interrupt (MISR[FSR] = 1) are generated. 0 = An interrupt has NOT occurred 1 = An interrupt has occurred **NOTE:** This bit is automatically cleared (by the ACUNIT) when any of MISR[FIFOE], MISR[EOC] or MISR[FSR] are cleared.

Table 148: GSR Bit Definitions (Continued)

Physical Address 0x4050_001C	GSR	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										PSOINT	PCLINT	CDONE	SDONE	reserved		RCS	B3S12	B2S12	B1S12	S1RESINT	PRESINT	S1CRDY	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	0	0

Bits	Access	Name	Description
0	Read/Write 1 to Clear	GSCI	CODEC GPIO Status Change Interrupt: Is set whenever bit 0 of slot 12 is 0b1, which indicates that one of the CODEC GPIOs changed state, and that the new values are available in slot 12. 0 = Bit 0 of Slot 12 is 0b0 1 = Bit 0 of Slot12 is 0b1

5.5.11 CODEC Access Register (CAR)

The CODEC Access register is described in Table 149.

Table 149: 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 indicates the CODEC I/O cycle state. If no cycle is in progress, this bit is cleared and the act of reading the register sets this bit, which reserves the right for that SW driver to perform the I/O cycle. Once the cycle is complete, the HW automatically clears the bit. SW 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, another driver is performing a CODEC I/O cycle across the link and the currently accessing driver must try again later. Refer to Section 5.4.3 for more detail 0 = A CODEC I/O cycle is NOT currently in progress 1 = A CODEC I/O cycle is currently in progress

5.5.12 PCM-Out Status Register (POSR)

The PCM-Out Status register is described in Table 150.

Table 150: POSR Bit Definitions

Physical Address 0x4050_0010																															
User Settings																															
Bit	31	30	29	28	27	26	25	24	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	Read/Write 1 to Clear	FIFOE	PCM-Out FIFO Error: Indicates whether a PCM-Out FIFO error occurred. 0 = PCM-Out FIFO error has NOT occurred. 1 = PCM-Out FIFO error has occurred.
3	—	—	reserved
2	R	FSR	PCM-Out FIFO Service Request: Indicates whether the PCM-Out FIFO needs to be serviced. 0 = PCM-Out FIFO does NOT need to be serviced. 1 = PCM-Out FIFO needs to be serviced. This bit is updated regardless of the value its interrupt enable, POCR[FSRIE].
1:0	—	—	reserved

5.5.13 PCM-In Status Register (PCMISR)

The PCM-In Status register is described in Table 151.

Table 151: PCMISR Bit Definitions

Physical Address 0x4050_0014																															
User Settings																															
Bit	31	30	29	28	27	26	25	24	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

Table 151: PCMISR Bit Definitions (Continued)

Physical Address 0x4050_0014																PCMISR											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
4	Read/Write 1 to Clear	FIFOE	PCM-In FIFO Error: Indicates whether a PCM-In FIFO error occurred. 0 = PCM-In FIFO error has NOT occurred. 1 = PCM-In FIFO error has occurred. **NOTE:** This bit is updated regardless of the value its interrupt enable, PCMICR[FEIE].
3	Read/Write 1 to Clear	EOC	DMA End Of Chain Interrupt: Set by ACUNIT hardware when DMA signals an end of Descriptor chain (EOC) while reading data from the FIFO. DMA channel stops after signalling EOC and cannot further service the FIFO. Software must clear this bit (by writing 0b1 to it) only after reading out data in the FIFO. 0 = DMA has NOT signalled an end of Descriptor chain (EOC) while reading data from the FIFO. 1 = DMA has signalled an end of Descriptor chain (EOC) while reading data from the FIFO.
2	R	FSR	PCM-In FIFO Service Request: Indicates whether the PCM-In FIFO needs to be serviced. 0 = PCM-In FIFO does NOT need to be serviced. 1 = PCM-In FIFO needs to be serviced. **NOTE:** This bit is updated regardless of the value its interrupt enable, PCMICR[FSRIE].
1:0	—	—	reserved

5.5.14 PCM Surround-Out Status Register (PCSSR)

The PCM Surround-Out Status register is described in Table 152.

Table 152: PCSSR Bit Definitions

Physical Address 0x4050_0028																															
PCSSR																															
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	Read/Write 1 to Clear	FIFOE	FIFO Error: Indicates whether a PCM-Surround Out FIFO error occurred. 0 = PCM-Surround Out FIFO error has NOT occurred. 1 = PCM-Surround Out FIFO error has occurred. **NOTE:** This bit is updated regardless of the value its interrupt enable, PCSCR[FEIE].
3	—	—	reserved
2	R	FSR	FIFO Service Request: Indicates whether the PCM-Surround Out FIFO Service Request occurred. 0 = Indicates that the FIFO does not need servicing. 1 = Indicates that the FIFO needs servicing. This bit is updated independently of the value its interrupt enable, PCSCR[FSRIE].
1:0	—	—	reserved

5.5.15 PCM Center/LFE Status Register (PCCLSR)

The PCM Center/LFE Status register is described in Table 153.

Table 153: PCCLSR Bit Definitions

Physical Address 0x4050_0034																PCCLSR												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	Read/Write 1 to Clear	FIFOE	PCM Center/LFE FIFO Error: Indicates whether a PCM Center/LFE FIFO error occurred. 0 = PCM Center/LFE FIFO error has NOT occurred. 1 = PCM-Center/LFE FIFO error has occurred. Note: This bit is updated regardless of the value its interrupt enable, PCCLCR[FEIE].
3	—	—	reserved
2	R	FSR	PCM Center/LFE FIFO Service Request: Indicates whether the PCM-Center/LFE FIFO needs to be serviced. 0 = PCM-Center/LFE FIFO does NOT need to be serviced. 1 = PCM-Center/LFE FIFO needs to be serviced. Note: This bit is updated regardless of the value its interrupt enable, PCCLCR[FSRIE].
1:0	—	—	reserved

5.5.16 Mic-In Status Register (MCSR)

The Mic-In Status register is described in Table 154.

Table 154: 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	?	?

Bits	Access	Name	Description
31:5	—	—	reserved

Table 154: MCSR Bit Definitions (Continued)

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	?	?

Bits	Access	Name	Description
4	Read/Write 1 to Clear	FIFOE	FIFO Error: Indicates whether a Mic-In FIFO error occurred. 0 = Mic-In FIFO error has NOT occurred. 1 = Mic-In FIFO error has occurred. **NOTE:** This bit is updated regardless of the value its interrupt enable, MCCR[FSRIE].
3	Read/Write 1 to Clear	EOC	DMA End Of Chain Interrupt: Set by ACUNIT hardware when DMA signals an end of Descriptor chain (EOC) while reading data from the FIFO. DMA channel stops after signalling EOC and can not further service the FIFO. Software must clear this bit (by writing 0b1 to it) only after reading out data in the FIFO. 0 = DMA has *not* signalled an end of Descriptor chain (EOC) while reading data from the FIFO. 1 = DMA has signalled an end of Descriptor chain (EOC) while reading data from the FIFO.
2	R	FSR	FIFO Service Request: Indicates whether the Mic-In FIFO needs to be serviced. 0 = Mic-In FIFO does NOT need to be serviced. 1 = Mic-In FIFO needs to be serviced. **NOTE:** This bit is updated regardless of the value its interrupt enable, MCCR[FSRIE].
1:0	—	—	reserved

5.5.17 MODEM-Out Status Register (MOSR)

The MODEM-Out Status register is described in Table 155.

Table 155: MOSR Bit Definitions

Physical Address 0x4050_0110	MOSR	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	Read/Write 1 to Clear	FIFOE	FIFO Error: Indicates whether a MODEM-Out FIFO error occurred. 0 = MODEM-Out FIFO error has NOT occurred. 1 = MODEM-Out FIFO error has occurred. **NOTE:** This bit is updated regardless of the value its interrupt enable, MOCR[FSRIE].
3	—	—	reserved
2	R	FSR	FIFO Service Request: Indicates whether the MODEM-Out FIFO needs to be serviced. 0 = MODEM-Out FIFO does NOT need to be serviced. 1 = MODEM-Out FIFO needs to be serviced. **NOTE:** This bit is updated regardless of the value its interrupt enable, MOCR[FSRIE].
1:0	—	—	reserved

5.5.18 MODEM-In Status Register (MISR)

The MODEM-In Status register is described in Table 156.

Table 156: MISR Bit Definitions

Physical Address 0x4050_0118	MISR	AC97

	31	30	29	28	27	26	25	24	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	?	?

Bits	Access	Name	Description
31:5	—	—	reserved
4	Read/Write 1 to Clear	FIFOE	FIFO Error: Indicates whether a MODEM-In FIFO error occurred. 0 = MODEM-In FIFO error has NOT occurred. 1 = MODEM-In FIFO error has occurred. **NOTE:** This bit is updated regardless of the value its interrupt enable, MICR[FSRIE].
3	Read/Write 1 to Clear	EOC	DMA End Of Chain Interrupt: Set by ACUNIT hardware when DMA signals an end of Descriptor chain (EOC) while reading data from the FIFO. DMA channel stops after signalling EOC and can not further service the FIFO. Software must clear this bit (by writing 0b1 to it) only after reading out data in the FIFO. 0 = DMA has *not* signalled an end of Descriptor chain (EOC) while reading data from the FIFO. 1 = DMA has signalled an end of Descriptor chain (EOC) while reading data from the FIFO.
2	R	FSR	FIFO Service Request: Indicates whether the MODEM-In FIFO needs to be serviced. 0 = MODEM-In FIFO does NOT need to be serviced. 1 = MODEM-In FIFO needs to be serviced. **NOTE:** This bit is updated regardless of the value its interrupt enable, MICR[FSRIE].
1:0	—	—	reserved

5.5.19 PCM Data Register (PCDR)

The PCM Data register is described in Table 157. Writing a sample to this register updates the data into the PCM Transmit FIFO. Reading this register reads and removes a sample from the PCM Receive FIFO. Sixteen-bit right and left samples are read and written at the same time to form 32 bits of data.

Table 157: 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: The right channel PCM data sent to the PCM Transmit FIFO or received from the PCM Receive FIFO.
15:0	R/W	PCML	PCM Left Channel Data: The left channel PCM data sent to the PCM Transmit FIFO or received from the PCM Receive FIFO.

5.5.20 PCM Surround Out Data Register (PCSDR)

The PCM Surround Data register is described in Table 158. Writing a sample to this register updates the data into the PCM Surround Transmit FIFO. Reading this register reads and removes a sample from the PCM Surround Receive FIFO. Sixteen-bit right and left samples are read and written at the same time to form 32 bits of data.

Table 158: PCSDR Bit Definitions

Physical Address 0x4050_002C	PCSDR	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**
	PSMR																**PSML**															
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	PSMR	PCM Surround Right Channel Data: The right channel PCM surround data sent to the PCM surround Transmit FIFO or received from the PCM surround Receive FIFO.
15:0	R/W	PSML	PCM Surround Left Channel Data: The left channel PCM surround data sent to the PCM surround Transmit FIFO or received from the PCM surround Receive FIFO.

5.5.21 PCM Center/LFE Data Register (PCCLDR)

The PCM Center/LFE Data register is described in Table 159. Writing a sample to this register updates the data into the PCM Center/LFE Transmit FIFO. Reading this register reads and removes a sample from the PCM Center/LFE Receive FIFO. Sixteen-bit right and left samples are read and written at the same time to form 32 bits of data.

Table 159: PCCLDR Bit Definitions

Physical Address 0x4050_0038	PCCLDR																													
User Settings																														
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PCLMR																PCLML															
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	PCLMR	PCM Center Channel Data: The center channel PCM data sent to the PCM Center/LFE Transmit FIFO or received from the PCM Center/LFE Receive FIFO.
15:0	R/W	PCLML	PCM LFE Channel Data: The LFE channel PCM data sent to the PCM Center/LFE Transmit FIFO or received from the PCM Center/LFE Receive FIFO.

5.5.22 Mic-In Data Register (MCDR)

The Mic-In Data register is described in Table 160. Reading this register reads and removes a sample from the Mic-In Receive FIFO.

Table 160: MCDR Bit Definitions

Physical Address 0x4050_0060	MCDR																													
User Settings																														
Bit	31	30	29	28	27	26	25	24	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: The microphone data sent received from the Mic-In Receive FIFO.

5.5.23 MODEM Data Register (MODR)

The MODEM Data register is described in Table 161. Writing a sample to this register updates the data into the MODEM Transmit FIFO. Reading this register reads and removes a sample from the MODEM Receive FIFO.

Table 161: MODR Bit Definitions

Physical Address 0x4050_0140																MODR															AC97
User Settings																															
Bit	31	30	29	28	27	26	25	24	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: The MODEM data sent to the MODEM Transmit FIFO or received from the MODEM Receive FIFO.

5.5.24 Accessing CODEC Registers

Each CODEC has as many as sixty-four 16-bit registers that are addressable internal to the CODEC at half-word boundaries (16-bit boundaries). Since the PXA3xx Processor Family supports only internal register accesses at word boundaries (32-bit boundaries), software must select the correct formula to translate a 7-bit CODEC address into a 32-bit processor physical address:

- For a primary audio CODEC:
  = 0x4050_0200 + Shift_Left_Once(Internal 7-bit CODEC register address)
- For a secondary audio CODEC:
  = 0x4050_0300 + Shift_Left_Once(Internal 7-bit CODEC register address)
- For a primary modem CODEC register address:
  = 0x4050_0400 + Shift_Left_Once(Internal 7-bit CODEC register address)
- For a secondary modem CODEC:
  = 0x4050_0500 + Shift_Left_Once(Internal 7-bit CODEC register address)

In the formulas, the function, Shift_Left_Once(), shifts the 7-bit CODEC address left by 1-bit and shifts a 0 to the least significant bit. Table 162 shows the various address translations.

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 ACUNIT only transmits the last Write.

Table 162: Address Mapping for PXA3xx Processor Family Audio and MODEM CODEC Registers

7-bit CODEC Address	Audio		MODEM	
	Physical Address for a Primary Audio CODEC	Physical Address for a Secondary Audio CODEC ID 01	Physical Address for a Primary MODEM CODEC	Physical Address for a Secondary MODEM CODEC ID 01
0x00	0x4050_0200	0x4050_0300	0x4050_0600	0x4050_0700
0x02	0x4050_0204	0x4050_0304	0x4050_0604	0x4050_0704
0x04	0x4050_0208	0x4050_0308	0x4050_0608	0x4050_0708
. . .	. . .	. . .	. . .	. . .
0x7A	0x4050_02F4	0x4050_03F4	0x4050_06F4	0x4050_07F4
0x7C	0x4050_02F8	0x4050_03F8	0x4050_06F8	0x4050_07F8
0x7E	0x4050_02FC	0x4050_03FC	0x4050_06FC	0x4050_07FC

6 UARTs

This chapter describes the universal asynchronous receiver/transmitter (UART) serial ports. The serial ports are controlled via direct-memory access (DMA) or programmed I/O.

6.1 Overview

Each PXA3xx processor has three UARTs: one full function UART (UART1) and two Bluetooth UARTs (UART2 and UART3). The UARTs use the same programming model.

Each port contains a UART and a slow serial 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 PXA3xx processor.

The PXA3xx processor can read a complete UART status for the UART status register. 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 PXA3xx processor until it is transmitted on the serial link; a 64-byte Receive FIFO buffers data from the serial link until it is read by the PXA3xx processor. In non-FIFO mode, the Transmit and Receive FIFOs are bypassed and the THR and RBR registers are used instead.

Each UART includes a programmable baud-rate generator that can divide the input clock by 1 to ($2^{16} – 1$), which 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.

6.1.1 Full Function UART

In addition to nCTS and nRTS, UART1 supports modem control signal that include nDSR, nDTR, nRI and nDCD. UART1 supports baud rates of 9600, 19.2K, 38.4K, 57.6K, 115.2K, 230K, 460K, and 921K, 1.8M and 3.6M.

6.1.2 Bluetooth UARTs

UART2 and UART3 support baud rates of 9600, 19.2K, 38.4K, 57.6K, 115.2K, 230K, 460K, and 921K, 1.8M and 3.6M. UART2 and UART3 can be connected to Bluetooth modules. UART2 and UART3 support the functions in the feature list.

6.1.3 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

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.

- Serial 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

6.1.4 PXA3xx Processor Differences

Table 163 shows the USIM Controller differences among the PXA32x, PXA31x, and PXA30x processors. Refer to each individual register for other operating differences.

Table 163: PXA3xx Processors Feature Differences

Feature	PXA30x	PXA31x	PXA32x
High Speed UART	Supported	Supported	Not Supported
2. 1.8 Mbps and 3.6 Mbps			

6.2 Features

The UARTs share the following features:

- Functionally compatible with the 16550A and 16750
- Ability to add or delete standard asynchronous communications bits (start, stop, and parity) in the serial data
- Independently controlled transmit, receive, line status, and data-set interrupts
- Modem control functions (nCTS and nRTS on all three UARTs. UART1 additionally has nDSR, nDTR, nRI, and nDCD signals)
- Auto-flow capability controls data I/O without generating interrupts:
  - nRTS (output) controlled by UART Receive FIFO
  - nCTS (input) from modem controls UART transmitter
- Programmable serial interface:
  - 7- or 8-bit characters
  - Even, odd, or no parity detection
  - 1 stop-bit generation
  - Baud-rate generation 3.6Mof 9600, 19.2K, 38.4K, 57.6K, 115.2K, 230K, 460K, 921K, 1.8M, and 3.6M 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
- Serial infrared asynchronous interface that conforms to the Infrared Data Association (IrDA) specification

6.3 Signal Descriptions

Table 164 lists and describes each external signal that is connected to a UART module and how these pins function as modem control lines. The pins transmit digital CMOS-level signals and are connected to the PXA3xx processor through GPIOs. Refer to Multi-Function Pin registers for details on the alternate-function pin configuration.

Table 164: UART Signal Descriptions

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.
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.
nCTS	Input	**Clear to Send —** when asserted, 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. An interrupt is generated when MSR[CTS] changes state and the modem-status interrupt is enabled. **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 UART can be programmed to interrupt the PXA3xx 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 re-asserts, because disabling the UART also disables interrupts. To get around this issue, use either 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. No data is transmitted when nCTS is high.
nDSR	Input	**Data Set Ready —** when asserted, 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. When MSR[DSR] changes state, an interrupt is generated if the modem-status interrupt is enabled.
nDCD	Input	**Data Carrier Detect —** when asserted, 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. An interrupt is generated when the DCD bit changes state and the modem-status interrupt is enabled.
nRI	Input	**Ring Indicator —** when asserted, 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. An interrupt is generated when the RI bit of the MSR changes from a high to low state and the modem-status interrupt is enabled.

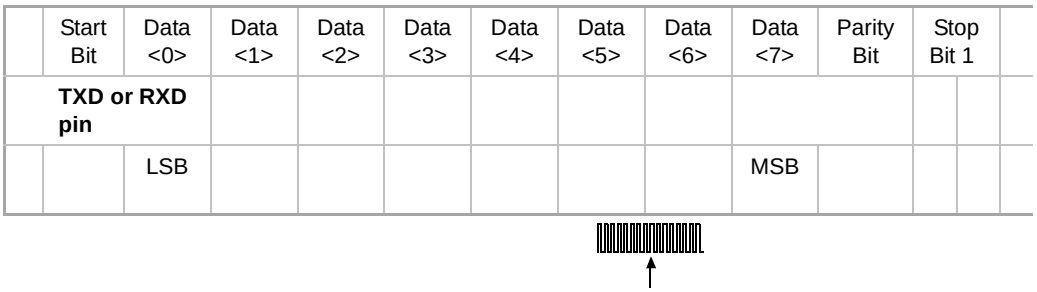
Table 164: UART Signal Descriptions (Continued)

Name	Type	Description
nDTR	Output	**Data Terminal Ready —** when asserted, 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 de-asserts this signal (high). Loop-mode operation holds nDTR de-asserted.
nRTS	Output	**Request To Send —** when asserted, 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 de-asserts this signal (high). Loop-mode operation holds nRTS de-asserted. **Non-Auto-flow mode**: set MSR[RTS] to assert the nRTS output (active low). **Auto-flow mode**: nRTS is asserted automatically by the auto-flow circuitry when the Receive buffer exceeds its programmed trigger threshold. It is de-asserted when enough bytes are removed from the buffer to lower the data level back to the trigger threshold.

6.4 Operation

Figure 59 shows the format of a UART data frame.

Figure 58: 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 59 are optional and can be programmed by software.

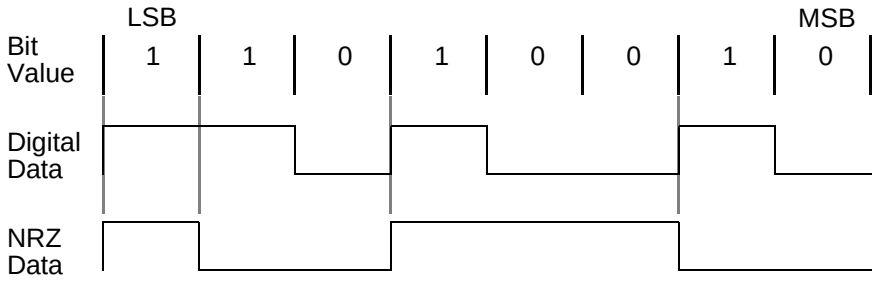
Each data frame is between nine and eleven bits long, depending on the size of the data programmed, and whether parity is enabled. A data frame begins by transmitting a start bit that is represented by a high-to-low transition. The start bit is followed by from to eight bits of data that begin with the 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 one stop bit. The stop bit represented by one successive bit period of logic one.

Each UART has two FIFOs: one Transmit and one Receive. The Transmit FIFO is 64 bytes deep and eight 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. Set IER[NRZE] (Interrupt Enable register) to enable NRZ coding. A bit value of 0b1 is represented by a

line transition, and 0b0 is represented by no line transition. Figure 59 shows the data byte 0b0100_1011 in NRZ coding. The byte LSB is transmitted first.

Figure 59: Example NRZ Bit Encoding — 0b0100_1011



6.4.1 Reset

The UARTs are disabled on reset. Software must program the Multi-function Pin registers, then set IER[UUE] to enable a UART. 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.

6.4.2 FIFO Operation

Each UART has a Transmit FIFO and a Receive FIFO, each FIFO holding 64 characters of data. There are two methods for moving data into and out of the FIFOs: DMA and Program I/O. In DMA mode, interrupts are used to control the data flow, where as polling is used in program I/O mode.

Note

The end-of-chain interrupt does not occur in Polled mode.

In DMA mode, software must set the DMA stop interrupt on the last Descriptor in the chain to avoid errors.

6.4.2.1 FIFO Interrupt Mode Operation

Note

The MCR[OUT2] is a global UART interrupt enable bit and must be set to enable UART interrupts.

6.4.2.1.1 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.

6.4.2.1.2 Character Timeout Interrupt

A cCharacter Timeout interrupt occurs when the Receive FIFO and Receive Timeout 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.
- The most recent FIFO Read was performed more than four continuous character times ago.

After the PXA3xx processor reads one character from the Receive FIFO or a new start bit is received, the Timeout interrupt is cleared and the Timeout is reset. If a Timeout interrupt has not occurred, the Timeout is reset when a new character is received or the PXA3xx processor reads the Receive FIFO.

6.4.2.1.3 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.

6.4.2.2 Removing Trailing Bytes With the Processor

The PXA3xx processor must remove trailing bytes when not in DMA mode or when DMA mode bit FCR[TRAIL] is not set. The presence of trailing bytes is signaled by the assertion of a character timeout interrupt. The PXA3xx processor follows this procedure when servicing a character timeout interrupt:

1. Read the Line Status register (LSR) and check for errors
2. Disable the receiver timeout interrupt via IER[RTOIE]
3. Read data from the UART FIFO
4. Read LSR, check for errors, and LOOP back to (Step 3) if DR is SET Go to (Step 5).
5. No more data in FIFO: Re-enable RTO interrupt via IER[MIE]
6. Done

6.4.2.3 FIFO Polled Mode Operation

When the FIFOs are enabled, clearing both IER[DMAE] and IER[4:0] places the 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 via the LSR. The PXA3xx processor polls the following bits for receive and transmit data service:

Receive Data Service — the PXA3xx 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 PXA3xx processor checks the LSR[TDRQ] (Transmit Data Request) bit, which is set when the transmitter needs data.

The PXA3xx processor can also check the LSR[TEMT] (Transmitter Empty) bit, which is set when the Transmit FIFO has shifted out (“empties”).

6.4.2.4 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 (DMAC) then writes data to the FIFO. For each DMA request, the DMAC can send eight, 16, or 32 bytes of data to the FIFO. The UART FIFO accepts partial-word or full-word transfers of one, two, three, or four consecutive bytes from the DMAC or program I/O. The actual number of bytes to be transmitted is programmed in the DMAC.

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 DMAC then reads data from the FIFO. For each DMA request, the DMAC can read eight, 16, or 32 bytes of data from the FIFO. When in 32-bit Peripheral Bus mode, the DMAC always attempts to read four bytes of data per transfer. Where less than four bytes are being transferred, the valid bytes are indicated by a data-valid bus shared between the UART and the DMAC. The UART can send one, two, three, or four bytes of data per bus transaction. The actual number of bytes to be read is programmed in the DMAC along with the bus width.

6.4.2.5 DMA Receive Programming Errors

If the DMA channel stops prematurely due to the end of a Descriptor chain or other error, the PXA3xx processor must be notified since the DMAC can no longer service the UARTs FIFOs. If this occurs, the PXA3xx processor must correct the situation by programming another Descriptor or by servicing the FIFOs via interrupt or Polling mode, as described above. There are two methods for notifying the PXA3xx processor of a stopped DMA channel:

1. Program the DMAC 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 6.4.2.7).

   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 timeout timer) and the Receive FIFO is empty. The IIR[EOC] interrupt does not assert if FCR[TRAIL] is cleared.

6.4.2.6 DMA Error Handling

If an error occurs while in DMA mode, the Receive-DMA requests are disabled and the Error interrupt, IIR[IID], is generated.

The PXA3xx processor must now read out the error bytes through programmed I/O (PIO). After 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 DMAC 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.

6.4.2.7 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*. Set FCR[TRAIL] to program the UART to request the DMA remove the trailing bytes. Setting FCR[TRAIL] also enables the IIR[EOC] interrupt, described in Section 6.4.2.2.

A request is issued automatically for the remaining number of bytes left in the Receive buffer when the DMAC is removing trailing bytes. The DMAC 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 PXA3xx processor is forced to take over because the DMAC 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 Processor I/O mode as described in Section 6.4.2.1.

6.4.2.8 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 timeout counter to expire. This situation causes an EOR to be sent to the DMAC if in DMA mode. If this situation 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 timeout. Therefore, 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 timeout.

6.4.2.9 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 three bytes in the FIFO, an EOR request is made to the DMAC to remove these bytes. If transmission of a new message B resumes before the DMAC 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 DMAC responds to message A's EOR request. Although this situation could occur, it would be considered a programming error because the higher communication protocol must prevent message B transmission until the local receiver acknowledges the receipt of message A. The exception to this scenario would be if enough new bytes enter the FIFO to push the FIFO level to its programmed data threshold. If this situation occurs, the request is treated as a normal service request and no EOR flag is asserted to the DMAC.

6.4.3 Auto-Flow Control

Auto-flow control uses the clear-to-send (nCTS) and request-to-send (nRTS) signals to automatically 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 (that is, sets to 0) nRTS. If the UART de-asserts (that is, sets to 1) 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 (that is, sets to 0). Marvell recommends using this feature

because it 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. Set bits MCR[RTS] and MCR[AFE] to enable full auto-flow. Set MCR[AFE] and clear MCR[RTS] to enable auto-nCTS-only mode.

6.4.3.1 nRTS (UART Output)

When in full Auto-flow mode, nRTS is asserted (0) when the UART FIFO is ready to receive data from the remote transmitter. This scenario occurs when the amount of data in the Receive FIFO is below the programmable trigger threshold value. nRTS is de-asserted (set to 1) when the amount of data in the Receive FIFO reaches the programmable trigger threshold. It is asserted again when enough bytes are removed from the FIFO to lower the data level below the trigger threshold.

6.4.3.2 nCTS (UART Input)

When in full- or half-Auto-flow mode, nCTS is asserted (set to 0) 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. The transmitter completes this byte if nCTS goes high while the transfer of a byte is in progress.

If UART transmission is stalled by disabling the UART, none of the interrupts in the Modem Status register (MSR) indicate an interrupt when nCTS re-asserts because disabling the UART also disables interrupts. Marvell recommends using auto-CTS in Auto-flow mode.

Note

Auto-flow mode can be used only in conjunction with FIFO mode.

6.4.4 Auto-Baud-Rate Detection

Each UART supports auto-baud-rate detection. When enabled, the UART counts the number of clock cycles within the start-bit pulse. This number is then written into the Auto- Baud-Count register (ACR; see Table 184) 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 PXA3xx processor can read ACR and use this information to program DLL and DLH with a baud rate calculated by the PXA3xx processor. After the baud rate has been programmed, the PXA3xx processor verifies 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 via the Auto-Baud Control register, bit ABR[ABT]. The baud rates that are seen in most commercial electronics, which are referred to as “common”, include baud rates of 9600, 19.2K, 38.4K, 57.6K, 115.2K, 921K, 1.8M, and 3.6M.

- Formula method — Only baud rates of 9600, 19.2K, 38.4K, 57.6K, 115.2K, 921K, 1.8M, and 3.6M can be programmed by the UART. This method works well for higher baud rates, but can 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 6.4.7 above 28.8 kbps. Below 28.8 kbps, the only baud rate that can be programmed by the UART is 19200.

When the baud rate is detected, the auto-baud circuitry disables itself by clearing ABR[ABE]. To re-enable auto-baud detection, set ABR[ABE] again.

Warning

Changing the baud rate is not permitted when actively transmitting or receiving data.

Note

Auto-baud-rate detection is not supported in IrDA (serial infrared) mode.

See Section 6.5.15 for more information on auto-baud.

6.4.5 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[BUS]. 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, etc.

- 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 DMAC or PIO.
- DMA — the DMAC 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 DMAC and the UART.
- PIO — the PXA3xx processor is restricted to reading or writing one, two, or four bytes per word. When reading, the PXA3xx 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 PXA3xx 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. The UART retrieves unusable data for the invalid bytes if the PXA3xx processor reads more than the number of bytes available in the Receive buffer. The Receive FIFO counters do not increment.

Note

The receive and transmit FIFOs must be enabled when in 32-bit mode.

6.4.6 Serial Infrared Asynchronous Interface

The slow serial-infrared (SIR) interface is used with all three UARTs to support two-way wireless communication that uses 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. The baud divisor must be eight or more because the input clock is 14.857 MHz.

Note

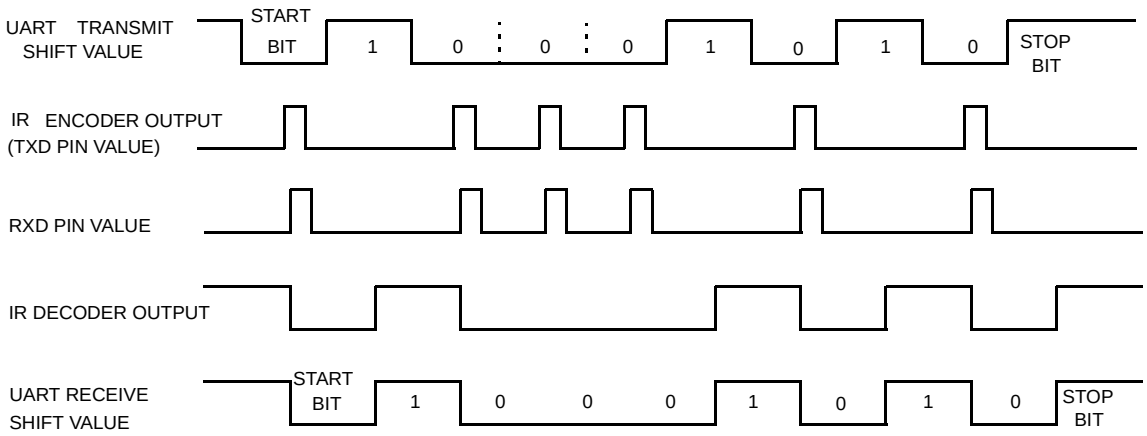
This SIR interface should be turned on and off only while the UART module is completely idle and disabled. On-the-fly changes are likely to produce problems and are not recommended.

Bit 2 of the Infrared Selection register (ISR) must be set to 0b1 (see Table 182).

6.4.6.1 Operation

The SIR modulation technique works with 7- or 8-bit characters with an optional parity bit. The data is preceded by a zero-value start bit and ends with one stop bit. The encoding scheme sends a 1.6-µs 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 60 shows an example of transmit/receive operation.

Figure 60: IR Transmit and Receive Example

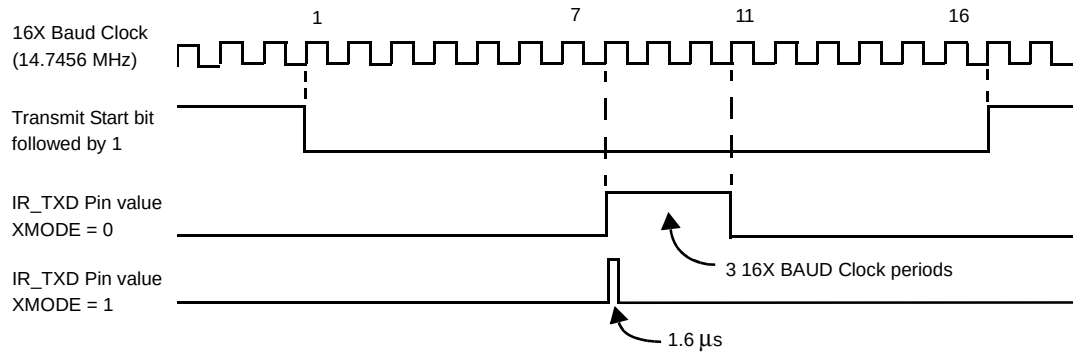


The top line in Figure 60 shows an asynchronous transmission as it is sent from the UART. The second line shows the pulses generated by the IR encoder at the TXD pin. A 1.6 µsec pulse is generated in the middle of the START bit and any data bit that is a zero. The third line shows the values received at the RXD input pin. The fourth line shows the receive-decoder output. The receive decoder drives the receive data line low when it detects a pulse. The bottom line shows how the UART receiver interprets the decoder action. This last line is the same as the first, but it is shifted half a bit period. Marvell 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. ISR[XMODE] must be set to 0b1 so 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 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 61 shows an example of XMODE operation.

Note

ISR[XMODE=1] toggles only the transmit pulse width. It does not effect the receive pulse width, which is always set to XMODE = 1.

Figure 61: XMODE Example



To prevent transmitter LED reflection feedback to the receiver, disable the IR receive decoder when the IR transmit encoder transmits data, and disable the IR transmit encoder when the IR receive decoder receives data. RCVEIR and XMITIR must not be set at the same time.

Note

This restriction specifically is to ensure no feedback; no filtering is performed to remove this feedback data.

6.4.7 Programmable Baud-Rate Generator

Each UART contains a programmable baud-rate generator that can take a fixed-input clock and divide it down to generate the preferred baud rate. The baud rate is calculated by taking the 14.7456-MHz fixed-input clock (in regular mode (IER[HSE] = 0)) or the 59.429 MHz clock (in high speed mode (IER[HSE = 1)) and dividing it by 1, 2, 4, 8, 16, 24, 48, or 96. For high speed mode, a divisor of 1 or 2 is required.

The baud-rate generator output frequency is 16 times the baud rate. Two 8-bit Divisor Latch registers (DLL and DLH; see Section 6.5.4, Divisor Latch Registers, Low and High (DLL, DLH)) 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 RS232 standard requires a clock frequency of 14.7456 MHz, however the processor clock divisors allow only a frequency of 14.857 MHz, which results in a slightly higher baud rate value compared to the standard.

The baud rate of the data shifted into or out of a UART is given by the formula:

In high speed mode (IER[HSE] = 1):

Note

In high speed mode only divisors of 1 and 2 are valid.

In regular mode (IER[HSE] = 0):

$$BaudRate = \frac{14.857\ \text{MHz}}{(16 x Divisor)}$$

Note

For example, if the divisor is 24 (in regular mode), the baud rate is 38400 bps.

Warning

Changing the baud rate is not permitted when actively transmitting or receiving data.

Refer to Table 165 for a list of recommended baud rates based on divisor values (DLH:DLL).

Table 165: Recommended Baud Rates

Required Baud Rate	Divisor	14.857 MHz Actual Baud Rate	59.429 MHz Actual Baud Rate
9600	96	9673	—
19200	48	19345	—
38400	24	38690	—
57600	16	58036	—
115200	8	116071	—
230400	4	232143	—
460800	2	464286	—
921600	1	928571	—
1842000	2	—	1857143
3686400	1	—	3714286

The divisor reset value is 0x0002. Changing the baud rate (writing to registers DLL and DLH) is not permitted while actively transmitting or receiving data.

6.5 Register Descriptions

Each UART has 13 registers: 12 for UART operation and one for slow serial 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.

6.5.1 UART Register Summary

The following tables show 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 166: Register Summary

Physical Address	LCR[DLAB] Bit Value	Description	Page
UART 1			
0x4010_0000	0	Receive Buffer Register (RBR)	page 400
0x4010_0000	0	Transmit Holding Registers (THR)	page 400
0x4010_0000	1	Divisor Latch Low Byte Register	page 401
0x4010_0004	1	Divisor Latch High Byte Register	page 401
0x4010_0004	0	Interrupt Enable Register (IER)	page 402
0x4010_0008	X	Interrupt Identification Register (IIR)	page 406
0x4010_0008	X	FIFO Control Register (FCR)	page 408
0x4010_000C	X	Line Control Register (LCR)	page 411
0x4010_0010	X	Modem Control Register (MCR)	page 413
0x4010_0014	X	Line Status Register (LSR)	page 415
0x4010_0018	X	Modem Status Register (MSR)	page 418
0x4010_001C	X	Scratchpad Register (SCR)	page 420
0x4010_0020	X	Infrared Selection Register (ISR)	page 420
0x4010_0024	X	Receive FIFO Occupancy Register (FOR)	page 422
0x4010_0028	X	Auto-Baud Control Register (ABR)	page 422
0x4010_002C	X	Auto-Baud Count Register (ACR)	page 423
UART 2			
0x4020_0000	0	Receive Buffer Register (RBR)	page 400
0x4020_0000	0	Transmit Holding Registers (THR)	page 400
0x4020_0000	1	Divisor Latch Low Byte Register	page 401

Table 166: Register Summary

Physical Address	LCR[DLAB] Bit Value	Description	Page
0x4020_0004	1	Divisor Latch High Byte Register	page 401
0x4020_0004	0	Interrupt Enable Register (IER)	page 402
0x4020_0008	X	Interrupt Identification Register (IIR)	page 406
0x4020_0008	X	FIFO Control Register (FCR)	page 408
0x4020_000C	X	Line Control Register (LCR)	page 411
0x4020_0010	X	Modem Control Register (MCR)	page 413
0x4020_0014	X	Line Status Register (LSR)	page 415
0x4020_0018	X	Modem Status Register (MSR)	page 418
0x4020_001C	X	Scratchpad Register (SCR)	page 420
0x4020_0020	X	Infrared Selection Register (ISR)	page 420
0x4020_0024	X	Receive FIFO Occupancy Register (FOR)	page 422
0x4020_0028	X	Auto-Baud Control Register (ABR)	page 422
0x4020_002C	X	Auto-Baud Count Register (ACR)	page 423
UART 3			
0x4070_0000	0	Receive Buffer Register (RBR)	page 400
0x4070_0000	0	Transmit Holding Registers (THR)	page 400
0x4070_0000	1	Divisor Latch Low Byte Register	page 401
0x4070_0004	1	Divisor Latch High Byte Register	page 401
0x4070_0004	0	Interrupt Enable Register (IER)	page 402
0x4070_0008	X	Interrupt Identification Register (IIR)	page 406
0x4070_0008	X	FIFO Control Register (FCR)	page 408
0x4070_000C	X	Line Control Register (LCR)	page 411
0x4070_0010	X	Modem Control Register (MCR)	page 413
0x4070_0014	X	Line Status Register (LSR)	page 415
0x4070_0018	X	Modem Status Register (MSR)	page 418
0x4070_001C	X	Scratchpad Register (SCR)	page 420
0x4070_0020	X	Infrared Selection Register (ISR)	page 420
0x4070_0024	X	Receive FIFO Occupancy Register (FOR)	page 422

Table 166: Register Summary

Physical Address	LCR[DLAB] Bit Value	Description	Page
0x4070_0028	X	Auto-Baud Control Register (ABR)	page 422
0x4070_002C	X	Auto-Baud Count Register (ACR)	page 423

6.5.2 Receive Buffer Register (RBR)

In non-FIFO mode, RBR (described in Table 168) holds the character(s) received by the UART Receive Shift register. If RBR 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].

RBR latches the value of the data byte at the front of the FIFO in FIFO mode.

Table 168: RBR Bit Definitions

Physical Address 0x4010_0000 – UART1 0x4020_0000 – UART2 0x4070_0000 – UART3 (LCR[DLAB]=0)	RBR1 RBR2 RBR3	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 (Only Valid in 32-bit peripheral bus mode)
24:16	R	Byte 2	Byte 2 (Only Valid in 32-bit peripheral bus mode)
15:8	R	Byte 1	Byte 1 (Only Valid in 32-bit peripheral bus mode)
7:0	R	Byte 0	Byte 0

6.5.3 Transmit Holding Registers (THR)

THR holds the data byte(s) to be transmitted next in non-FIFO mode. When the Transmit Shift register (TSR) is emptied, the contents of THR are loaded into the TSR and LSR[TDRQ] is set.

A Write to THR puts data into the top of the FIFO in FIFO mode. The data at the front of the FIFO is loaded into the TSR when the TSR is empty.

Table 169: THR Bit Definitions

Physical Address 0x4010_0000 – UART1 0x4020_0000 – UART2 0x4070_0000 – UART3 (LCR[DLAB]=0)	THR1 THR2 THR3	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 only in 32-bit peripheral bus mode)
24:16	W	Byte 2	Byte 2 (valid only in 32-bit peripheral bus mode)
15:8	W	Byte 1	Byte 1 (Only Valid in 32-bit peripheral bus mode)
7:0	W	Byte 0	Byte 0

6.5.4 Divisor Latch Registers, Low and High (DLL, DLH)

DLL and DLH, described in Table 170 and Table 171, contain the divisor for the programmable baud-rate generator, as described in Section 6.4.7. Access these registers with word Writes.

Note

If both the DLH and DLL registers need to be changed, write the DLH register first, read back the DLH value, then write the DLL register. This gurantees internal timings/coherency.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

6.5.4.1 Divisor Latch Low Byte Register

Table 170: DLL Bit Definitions

Physical Address 0x4010_0000 – UART1 0x4020_0000 – UART2 0x4070_0000 – UART3 (LCR[DLAB]=1)	DLL1 DLL2 DLL3	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	reserved
7:0	R/W	DLL	Low-byte compare value to generate baud rate

6.5.4.2 Divisor Latch High Byte Register

Table 171: DLH Bit Definitions

Physical Address 0x4010_0004 – UART1 0x4020_0004 – UART2 0x4070_0004 – UART3 (LCR[DLAB]=1)	DLH1 DLH2 DLH3	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	reserved
7:0	R/W	DLH	High-byte compare value to generate baud rate.

6.5.5 Interrupt Enable Register (IER)

IER, described in Table 172, enables the five types of interrupts that set a value in the Interrupt Identification register (IIR). Software must clear the appropriate bit in IER to disable an interrupt. Software can enable some interrupts by setting the appropriate bit.

The character timeout-indication interrupt is separated from the received data-available interrupt to ensure that the PXA3xx processor and the DMA controller do not service the Receive FIFO at the same time. When a character-timeout-indication interrupt occurs, the PXA3xx 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 PXA3xx 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

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. Software must read ISR to determine the error condition if an interrupt occurs and LSR[FIFOE] is clear. DMA requests are automatically enabled when the last error byte is read from the FIFO. 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] is used to enable 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.

Note

Software must not set DMAE while the TIE or RAVIE bits are set to ensure that the DMA controller and programmed I/O do not access the same FIFO.

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

Table 172: IER Bit Definitions (Sheet 1 of 3)

Physical Address 0x4010_0004 – UART1 0x4020_0004 – UART2 0x4070_0004 – UART3 (LCR[DLAB]=0)	IER1 IER2 IER3	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**
PXA32x Only	**Reserved**																							**Reserved**	**DMAE**	**UUE**	**NRZE**	**RTOIE**	**MIE**	**RLSE**	**TIE**	**RAVIE**
PXA31x and PXA30x Only	**Reserved**																							**HSE**	**DMAE**	**UUE**	**NRZE**	**RTOIE**	**MIE**	**RLSE**	**TIE**	**RAVIE**
Reset	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Bits	Access	Name	Description
31:9	—	reserved	reserved

Table 172: IER Bit Definitions (Sheet 2 of 3)

Physical Address 0x4010_0004 – UART1 0x4020_0004 – UART2 0x4070_0004 – UART3 (LCR[DLAB]=0)	IER1 IER2 IER3	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
PXA32x Only	Reserved																							Reserved	DMAE	UUE	NRZE	RTOIE	MIE	RLSE	TIE	RAVIE
PXA31x and PXA30x Only	Reserved																							HSE	DMAE	UUE	NRZE	RTOIE	MIE	RLSE	TIE	RAVIE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
8	R/W	HSE	High Speed UART Enable: When High Speed mode is enabled, the UART uses a 59.429 MHz input clock rather than a 14.857 MHz input clock. High speed mode should only be used for baud rates of 1.8 Mbps or 3.6 Mbps. 0 = High speed mode is disabled (Regular mode) 1 = High speed mode is enabled **NOTE:** High Speed UART is not supported in S0/D0CS/C0. HSE must be cleared before entering S0/D0CS/C0.
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 serial 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 172: IER Bit Definitions (Sheet 3 of 3)

Physical Address
0x4010_0004 – UART1
0x4020_0004 – UART2
0x4070_0004 – UART3
(LCR[DLAB]=0)

IER1
IER2
IER3

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
PXA32x Only	Reserved																							Reserved	DMAE	UUE	NRZE	RTOIE	MIE	RLSE	TIE	RAVIE
PXA31x and PXA30x Only	Reserved																							HSE	DMAE	UUE	NRZE	RTOIE	MIE	RLSE	TIE	RAVIE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	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

6.5.6 Interrupt Identification Register (IIR)

The UART prioritizes interrupts in four levels (see Table 173) 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, IIR is not updated until the error is at the bottom of the FIFO, which could be much later than when the error interrupt occurred. Software must check for the error interrupt before it checks IIR.

The interrupt is de-asserted if additional data is received before a receiver time-out interrupt is serviced.

IIR contains an interrupt-pending bit (nIP). An interrupt is pending from the UART when IIR[nIP] is clear. When IIR[nIP] is set, no interrupt is pending, regardless of the state of the other bits in the IIR register. This condition may 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 173: Interrupt Conditions

Priority Level	Interrupt Origin
1 (highest)	Receiver line status: one or more error bits were set.
2	Received data is available. In FIFO mode, the trigger threshold was reached. In non-FIFO mode, RBR has data.
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 have changed state.

Table 174: IIR Bit Definitions (Sheet 1 of 2)

Physical Address
0x4010_0008 – UART1 **IIR1**
0x4020_0008 – UART2 **IIR2**
0x4070_0008 – UART3 **IIR3**

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	Reserved																								FIFOES		EOC	ABL	TOD	IID		nIP
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	1

Bits	Access	Name	Description
31:8	—	reserved	reserved

Table 174: IIR Bit Definitions (Sheet 2 of 2)

Physical Address 0x4010_0008 – UART1 0x4020_0008 – UART2 0x4070_0008 – UART3	IIR1 IIR2 IIR3	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
7:6	R	FIFOES[1:0]	FIFO Mode Enable Status: 00 – Non-FIFO mode is selected 01 – **reserved** 10 – **reserved** 11 – FIFO mode is selected (TRFIFOE = 1)
5	R	EOC	**DMA End of Descriptor Chain:** (See Section 6.4.5) 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 6.4.4) 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 6.4.2.1.2) 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: 00 – Modem Status (CTS, DSR, RI, DCD modem signals changed state) 01 – Transmit FIFO requests data 10 – Received data available 11 – Receive error (Overrun, parity, framing, break, FIFO error. See Table 180)
0	R	nIP	Interrupt Pending: 0 = Interrupt is pending (active low) 1 = No interrupt is pending

Table 175 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 175: 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.	—

Table 175: Interrupt Identification Register (IIR) Decode (Continued)

Interrupt ID bits					Interrupt SET/RESET Function			
	3	2	1	0	Priority	Type	Source	RESET Control
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.	Non-FIFO mode: Reading the Receiver Buffer register.
							FIFO mode: trigger threshold was reached.	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 1 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	Reading the IIR register (if the source of the interrupt) or writing into the Transmit Holding register.
							FIFO mode: Transmit FIFO has half or less than half data.	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

6.5.7 FIFO Control Register (FCR)

FCR, described in Table 176, 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 (via 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 clear increments the FIFO counters by only one byte. 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, trailing bytes are handled by the PXA3xx processor. When set, trailing bytes are handled automatically by the DMA controller. See the DMA chapter 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

RESETTF is reset to 0 automatically 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.

Note

RESETRF is reset automatically 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 cleared automatically from the FIFOs. Any DMA or FIFO-service-request interrupts are cleared when TRFIFOE is clear.

Note

This bit must be 1 when other bits in this register are written, or the other bits are not programmed.

This is a write-only register. Write 0b0 to reserved bits.

Table 176: FCR Bit Definitions

Physical Address 0x4010_0008 – UART1 0x4020_0008 – UART2 0x4070_0008 – UART3	FCR1 FCR2 FCR3	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
31:8	—	reserved	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 via 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 PXA3xx processor 1 = Trailing bytes are removed by the DMAC
3	W	TIL	Transmitter Interrupt Level 0 = Interrupt/DMA request when FIFO is half empty. 1 = Interrupt/DMA request when FIFO is empty
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

Table 176: FCR Bit Definitions (Continued)

Physical Address 0x4010_0008 – UART1 0x4020_0008 – UART2 0x4070_0008 – UART3	FCR1 FCR2 FCR3	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
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 cleared automatically 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

6.5.8 Line Control Register (LCR)

LCR, defined in Table 177, specifies the format for the asynchronous data-communications exchange. The serial-data format consists of a start bit, eight data bits, an optional parity bit, and one stop bit. 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 177: LCR Bit Definitions

Physical Address 0x4010_000C – UART1 0x4020_000C – UART2 0x4070_000C – UART3	LCR1 LCR2 LCR3	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	reserved

Table 177: LCR Bit Definitions (Continued)

Physical Address 0x4010_000C – UART1 0x4020_000C – UART2 0x4070_000C – UART3	LCR1 LCR2 LCR3	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
7	R/W	DLAB	Divisor Latch Access Bit 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)
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. This bit must be clear. 0 = 1 stop bit 1 = reserved

Table 177: LCR Bit Definitions (Continued)

Physical Address 0x4010_000C – UART1 0x4020_000C – UART2 0x4070_000C – UART3	LCR1 LCR2 LCR3	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
1:0	R/W	WLS[1:0]	Word Length Select Specifies the number of data bits in each transmitted or received character. 00 – reserved 01 – reserved 10 – 7-bit character 11 – 8-bit character

6.5.9 Modem Control Register (MCR)

MCR, defined in Table 178, uses the modem control pins nRTS and nDTR to control the interface with a modem or data set. MCR also controls the loopback mode. Loopback mode must be enabled before the UART is enabled.

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

Table 178: MCR Bit Definitions

Physical Address 0x4010_0010 – UART1 0x4020_0010 – UART2 0x4070_0010 – UART3	MCR1 MCR2 MCR3	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
31:6	—	reserved	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.

Table 178: MCR Bit Definitions (Continued)

Physical Address 0x4010_0010 – UART1 0x4020_0010 – UART2 0x4070_0010 – UART3	MCR1 MCR2 MCR3	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
4	R/W	LOOP	Loopback Mode: This bit 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. Marvell recommends that MSR be read 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 received immediately. This feature allows the PXA3xx 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 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 PXA3xx processor: 0 = MSR[DCD] forced to 0b0 1 = MSR[DCD] forced to 0b1
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

Table 178: MCR Bit Definitions (Continued)

Physical Address 0x4010_0010 – UART1 0x4020_0010 – UART2 0x4070_0010 – UART3	MCR1 MCR2 MCR3	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
1	R/W	RTS	Request to Send 0 = Non-Auto-flow mode nRTS pin is 1 nRTS pin is 0 1 = Auto-flow mode Auto-RTS disabled. Auto-flow works only with auto-CTS. Auto-RTS 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

6.5.10 Line Status Register (LSR)

LSR, defined in Table 179, provides data-transfer status information to the PXA3xx 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 6.4.2.4 for details on using the DMAC to receive data.

This is a read-only register. Ignore reads from reserved bits.

Table 179: LSR Bit Definitions

Physical Address 0x4010_0014 – UART1 0x4020_0014 – UART2 0x4070_0014 – UART3	LSR1 LSR2 LSR3	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																								**FIFOE**	**TEMT**	**TDRQ**	**BI**	**FE**	**PE**	**OE**	**DR**
Reset	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**

Bits	Access	Name	Description
31:8	—	reserved	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 PXA3xx 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.
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 PXA3xx 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

Table 179: LSR Bit Definitions (Continued)

Physical Address 0x4010_0014 – UART1 0x4020_0014 – UART2 0x4070_0014 – UART3	LSR1 LSR2 LSR3	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																								**FIFOE**	**TEMT**	**TDRQ**	**BI**	**FE**	**PE**	**OE**	**DR**
Reset	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**

Bits	Access	Name	Description
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 bit). BI is cleared when the PXA3xx 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: FE 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. FE is cleared when the PXA3xx processor reads the LSR. The UART will resynchronize 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
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 PXA3xx 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 PXA3xx 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 PXA3xx processor reads the LSR. 0 = No data has been lost 1 = Receive data has been lost

Table 179: LSR Bit Definitions (Continued)

Physical Address 0x4010_0014 – UART1 0x4020_0014 – UART2 0x4070_0014 – UART3	LSR1 LSR2 LSR3	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																								**FIFOE**	**TEMT**	**TDRQ**	**BI**	**FE**	**PE**	**OE**	**DR**
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	1	0	0	0	0	0

Bits	Access	Name	Description
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

6.5.11 Modem Status Register (MSR)

MSR provides the current state of the control lines from the modem or data set (or a peripheral device emulating a modem) to the PXA3xx 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 PXA3xx processor reads MSR.

The status of the modem control lines does not affect the FIFOs. IER[MIE] must be set to use these lines for flow control. The interrupt service routine must disable the UART when an interrupt occurs on one of the flow-control pins. 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 180: MSR Bit Definitions

Physical Address 0x4010_0018 – UART1 0x4020_0018 – UART2 0x4070_0018 – UART3	MSR1 MSR2 MSR3	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																								**DCD**	**RI**	**DSR**	**CTS**	**DDCD**	**TERI**	**DDSR**	**DCTS**
Reset	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Bits	Access	Name	Description
31:8	—	reserved	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

6.5.12 Scratchpad Register (SCR)

SCR, defined in Table 181, has no effect on the UART. It is intended as a scratchpad register for use by programmers and is included for 16550A compatibility.

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

Table 181: SCR Bit Definitions

Physical Address 0x4010_001C – UART1 0x4020_001C – UART2 0x4070_001C – UART3	SCR1 SCR2 SCR3	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																								SCRATCHPAD							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	reserved	reserved
7:0	R/W	SCRATCHPAD	No effect on UART functions

6.5.13 Infrared Selection Register (ISR)

Each UART can manage an IrDA module associated with it. ISR controls the IrDA functions (see Section 6.4.6).

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

Table 182: ISR Bit Definitions

Physical Address 0x4010_0020 – UART1 0x4020_0020 – UART2 0x4070_0020 – UART3	ISR1 ISR2 ISR3	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	reserved
4	R/W	RXPL	Receive Data Polarity: 0 = SIR decoder takes positive pulses as zeros 1 = SIR decoder takes negative pulses as zeros

Table 182: ISR Bit Definitions (Continued)

Physical Address 0x4010_0020 – UART1 0x4020_0020 – UART2 0x4070_0020 – UART3	ISR1 ISR2 ISR3	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
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
2	R/W	XMODE	Transmit Pulse Width Select: When XMODE is clear, the UART 16x clock is used to clock 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. Marvell 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 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 infrared mode

6.5.14 Receive FIFO Occupancy Register (FOR)

FOR, described in Table 183, shows the number of bytes currently remaining the Receive FIFO. FOR can be used to determine 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 PXA3xx processor removes trailing bytes as opposed to the DMA (see Section 6.4.2.7). 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 183: FOR Bit Definitions

Physical Address 0x4010_0024 – UART1 0x4020_0024 – UART2 0x4070_0024 – UART3	FOR1 FOR2 FOR3	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																										BYTE COUNT					
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0

Bits	Access	Name	Description
31:6	—	reserved	reserved
5:0	R/W	BYTE COUNT	Number of bytes (0-63) remaining in the Receive FIFO

6.5.15 Auto-Baud Control Register (ABR)

ABR, described in Table 184, 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 PXA3xx processor or the UART to program the final baud rate in the Divisor Latch registers, and choose between two methods used 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 PXA3xx processor if ABR[ABLIE] is set. It also programs automatically the Divisor Latch registers (DLL and DLH) if ABR[ABUP] is set.

See Section 6.4.4 for more information on auto-baud rate.

Note

Auto-baud-rate detection is not supported in IrDA serial-infrared mode.

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

Table 184: ABR Bit Definitions

Physical Address 0x4010_0028 – UART1 0x4020_0028 – UART2 0x4070_0028 – UART3	ABR1 ABR2 ABR3	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																												**ABT**	**ABUP**	**ABLIE**	**ABE**
Reset	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**0**	**0**	**0**

Bits	Access	Name	Description
31:4	—	reserved	reserved
3	R/W	ABT	0 = Formula used to calculate baud rates, allowing all possible baud rates to be chosen by UART as shown in Section 6.4.2.1.2 1 = Table used to calculate baud rates, which limits UART to choosing common baud rates
2	R/W	ABUP	0 = PXA3xx processor Programs Divisor Latch registers 1 = UART Programs Divisor Latch registers
1	R/W	ABLIE	0 = Auto-baud-lock interrupt disabled (Source IIR[ABL]) 1 = Auto-baud-lock interrupt enabled (Source IIR[ABL])
0	R/W	ABE	0 = Auto-baud disabled 1 = Auto-baud enabled

6.5.16 Auto-Baud Count Register (ACR)

ACR, defined in Table 185, stores the number of input clock cycles within a start-bit pulse. This value is then used by the PXA3xx 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 PXA3xx 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 185: ACR Bit Definitions

Physical Address 0x4010_002C – UART1 0x4020_002C – UART2 0x4070_002C – UART3	ACR1 ACR2 ACR3	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	reserved
15:0	R	COUNT VALUE	Number of input clock cycles within a start-bit pulse.

7 Consumer Infrared Unit

7.1 Overview

The consumer infrared unit (CIR) enables the PXA3xx processors to control consumer devices remotely such as televisions and VCRs.

Since there are several existing standards in the market (RC-5, RC-5 extended, RC-6, etc.), the module design is generic and flexible. This allows the module to be compatible with many existing standards.

There are no CIR controller differences among the PXA32x processor, PXA31x processor or PXA30x processor.

7.2 Features

The consumer infrared unit has the following features:

- Supports existing standards—RC-5, RC-5 extended, and RC-6
- Supports normal “1” or “0” and short “1” or “0” symbol types with programmable symbol lengths
- Programmable modulation frequency
- Programmable duty cycle for the modulation wave (for power saving)
- Double buffer for input stream
- Supports Manchester coding
- One system-level interrupt that indicates empty buffer or end-of-transmission

7.3 Signal Description

This section describes the I/O signal used by the CIR interface.

Table 186 lists the pin that connects the CIR and the IR LED diode.

Table 186: CIR I/O Signals

Name	Type	Description
CIR_OUT	output	CIR unit output that drives the IR LED diode

7.4 Operation

The unit consists of four main blocks:

- *Modulator*—Generates the modulation wave.
- *Symbol Generator*—Determines the symbol period and value.
- *Input Symbols*—Gets the streaming symbols from the software and supplies them to the symbol generator.
- *Manchester Machine*—Generates Manchester coding on the streaming symbols, if required.

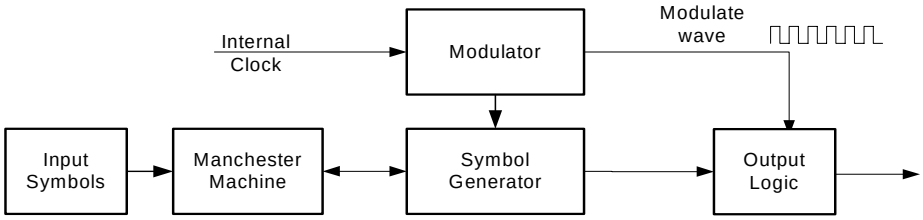
The unit output is comprised of modulated waves that are generated by the modulator. The modulator can be configured to output periodic amplitude-modulated pulses or an unmodulated NRZ logic level of the signal (simple “0” and “1”). The period and duty cycle of the modulated pulses can

be varied by writing the desired values into the pulse-width comparator and modulation- period comparator.

The input-symbols block gets the symbols from the software, processes them and feeds them sequentially to the symbol generator through the Manchester machine (if the CIR Control register Manchester-enable bit has been asserted).

A block diagram of the consumer infrared unit is shown in Figure 62.

Figure 62: CIR Block Diagram



7.4.1 Programming Sequence

The programming sequence to transmit one CIR burst is as follows:

1. Configure modulator wave by writing required values for the pulse-width and modulation- period comparators, CIRPW[PW] and CIRMP[MP], respectively (see Table 190 and Table 191).
2. Configure the symbols to be used by writing the number of modulation cycles and unmod bit in each CIR Symbol Length registers (CIRN0, CIRN1, CIRS0, CIRS1) (see Table 192 through Table 195).
3. Write the total number of symbols to CIRNS[SYMCNT] (see Table 196).
4. Write the first 16 symbols (32 bits) in the CIR Buffer register, CIRBUFF (see Table 197).
5. Configure the CIR Control register, CIRCR, and enable the unit operation by setting CIR_Enable and CLK_Enable (see Table 198).
6. When the CIR interrupt is asserted, read the interrupt indication bits, CIRIR[EOT] and CIRIR[EB] (see Table 200). If CIRIR[EOT] is asserted, the transfer is complete. The CIR_Enable is cleared by hardware. CLK_Enable can be cleared via software if the unit is idle for some time. Clear the CIRIR[EOT] bit and assert software reset (see Table 8-4) prior to restarting a new transfer. If CIRIR[EOT] is not asserted, but CIRIR[EB] is, the buffer needs reloading. If this is the last buffer to be loaded, mask the CIRIR[EB] interrupt. Load the buffer with new data. The CIRIR[EB] interrupt is cleared automatically when the buffer is loaded with new data.

7.4.2 Input Clock Frequency

Normally the CIR unit is clocked with the UART clock running 14.857 MHz. All of the calculations in this document reference this frequency. However, in S0/D0CS/C0 low-power mode, the CIR unit clock runs at 12 MHz. Thus, use 12 MHz in calculations if configuring the CIR for use during S0/D0CS/C0.

7.4.3 Modulator

The modulator block generates a continuous modulated waveform as configured in the pulse-width and modulation-period comparators, defined in Table 190 and Table 191.

An 11-bit counter connected to the 14.857-MHz clock supplies the timing for this unit. To control the modulation waveform, write values to two comparators:

- Pulse Width Comparator—Determines the pulse width of the modulated wave pulses by specifying the duration a signal can take a logic-high level during the total period of the modulated waveform. For power conservation, the logic-high duration may be shortened.
- Modulation Period Comparator—Determines the period of the modulated waveform.

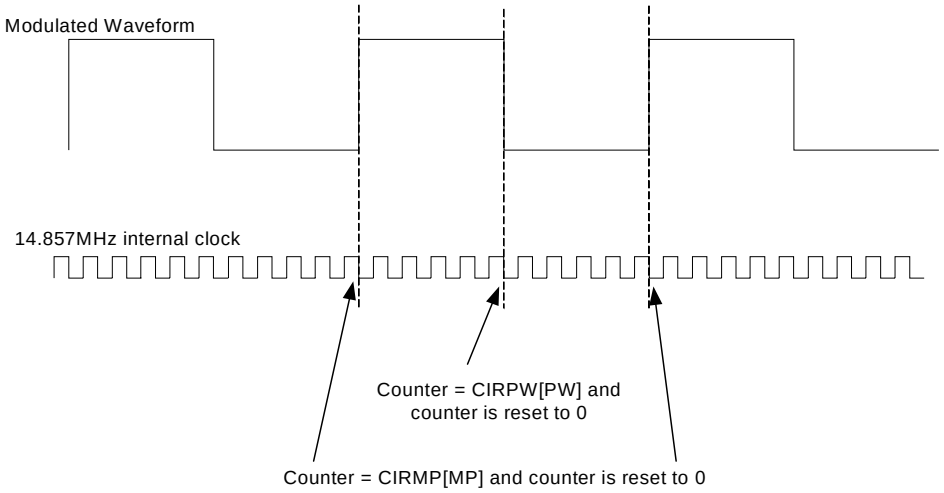
The period of the modulated waveform is generated by the following sequence. The counter begins counting up from zero. The output of the modulator remains asserted (logic high) until the counter reaches the pulse-width comparator value. The output then de-asserts until the counter reaches the modulation-period comparator value. The counter is then reset to zero, and the modulator output asserts again. One cycle of the modulation wave is shown in Figure 63.

The maximum frequency of the modulated waveform is 14.857 MHz/2 = 7.4285 MHz and the minimum is 14.857 MHz/2047 = 7.258

Details of the modulator comparator registers are shown in Table 190 and Table 191.

.

Figure 63: Modulation Cycle



Since the counter is connected to the 14.857-MHz clock, each step of the counter represents 67.31 ns (1/14.857 MHz). For example, the pulse-width comparator is set to 930 (62.6 µs) and the modulation-period comparator to 1859 (125.1 µs), an 8-kHz modulation wave with 50% duty cycle is generated.

7.4.4 Symbol Generator

This block determines how many modulation cycles are produced for each symbol period and if the output passes NRZ (unmodulated signal) for this period of modulation waves or modulation waves.

There are four possible symbols and a Configuration register for each, as shown in Table 192–Table 195.

N1/S1 symbols represent 1 logic, and N0/S0 symbols represent 0 logic.

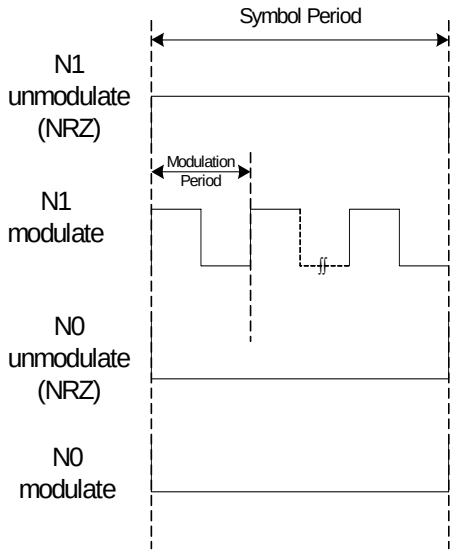
The normal symbols (N0/N1) and short symbols (S0/S1) are programmed to different periods. The normal symbols are used for transmitting regular bits and the short symbols are used for start bits that require a short period.

The main parameter for each symbol is the period, which is measured in “number of modulation cycles” (CIRN0, CIRN1, CIRS0, CIRS1).

There is also support for unmodulated signals (NRZ coding), so the 1 symbols (N1, S1) have an unmod bit that determines if the output produces modulation waves or simple 1 for the symbol period. There is no such bit for the 0 symbols (N0, S0) because they should produce 0 in any case for the given symbol period.

Figure 64 shows an example of the N1 and N0 symbols.

Figure 64: N1/N0 Symbol Example



Before the CIR is started, configure the Symbol Length registers according to the preferred standard (RC-5, RC-6...).

When the CIR is enabled, according to the “symbol selector,” (generated at the input symbol block and explained in Section 7.4.5) one of the four available CIR Symbol Length registers (CIRN0, CIRN1, CIRS0, CIRS1) is selected. The value contained within the register is loaded to the N modulator counter. If applicable, the corresponding unmod bit is also loaded.

When the count reaches zero, the next Symbol register value is loaded to the down counter. Each symbol can consist of a *maximum* of 127 modulation periods. Thus, a stream of symbols produces an output stream of periods of modulation and zero (when the unmod bit is 0b0).

Note

Actual symbol length is guaranteed to be +/- 5% of the nominal programmed value. Changing the Symbol Length registers, in particular the N value, while data is being transmitted is likely to cause unpredictable results and is thus not supported.

7.4.5 Input Symbol

This block gets the input symbols from the software and feeds them sequentially to the rest of the unit.Each symbol is represented by two bits. Table 187 lists the available symbols.

Table 187: Symbol Look-Up Table

Symbol	Representation
N0	00
N1	01
S0	10
S1	11

The LSB of the symbol represents the symbol value (0 or 1) and the MSB represent the symbol type (normal/short).

Before CIR operation, software needs to write the first symbols, which is a maximum of 16 symbols at once, to the CIRBUFF (CIR Buffer register) (shown in Table 197) and then write the number of symbols in this burst to the CIRNS register that is described in Table 197. After a first transmission, which does not send the entire contents of the CIRBUFF register, a second transmission to send the rest of the contents without first reloading the CIRBUFF register is unsupported.

The CIR unit transmits the symbols sequentially from the CIRBUFF register, LSB to MSB.

After the CIR unit is enabled, the CIRBUFF register content is transferred to the Shift register. The empty-buffer interrupt is asserted and the next command code can be written to the CIRBUFF register.

The first symbol, the two LSB bits of the Buffer register, is sent from the Buffer register to the Shift register and then to the symbol selector in the symbol generator block. If Manchester coding is enabled, the Manchester block processes the symbol before it reaches the symbol- generator block.

After the “reload next symbol” signal is asserted, the counter decreases by 1 and the next symbol from the Shift register is supplied to the “symbol selector”.

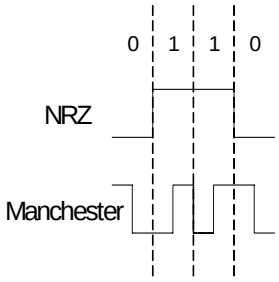
When the counter reaches zero, a “finish sending” interrupt is asserted, so the next Buffer register can loaded to the Shift register.

7.4.6 Manchester Machine

This block is used to generate Manchester code from the input symbol stream, if needed.

Figure 65 shows an example for Manchester coding.

Figure 65: Manchester Coding Example



The input to this block is the symbols from the “input symbols” block. If no Manchester coding is required, this block passes the symbols from input to output without any action.

As shown in Figure 65, each original symbol is translated to two symbols: 0 is translated to the sequence 1, 0; 1 is translated to the sequence 0,1. So when Manchester coding is required (see Table 198 for specific enable bit), each symbol from the block input (coming from the Shift register) translates to two symbols at the block output (symbol selector of the symbol- generator block), according to the table in Table 188.

Table 188: Manchester Machine

Input Symbol	Output Symbols†
N0	N1, N0
N1	N0, N1
S0	S0, S0
S1	S1, S1

† The left symbol is the first one to be sent (for example, in the first row N1 is sent first and N0 after)

Manchester code is given only for “N” symbols and not “S” symbols (as can be see from Table 188).

Since the Manchester coding block generates two symbols for each symbol transmission, the actual symbol rate is half of the “input symbol” rate.

7.4.7 Interrupts

The CIR interrupt can result from one of two possible events:

- Empty-Buffer Interrupt (EB)—Indicates that the receiving buffer is empty and ready for new data.
- End-of-Transmission Interrupt (EOT)—Indicates the end of the current data-burst. The end of transmission comes once after the completion of each burst.

After interrupt assertion, software must read the interrupt indication bits CIRIR[EOT] and CIRIR[EB] to determine which interrupts were asserted (see Table 200). If CIRIR[EOT] is asserted, the transfer is complete and the CIRCR[CIR_Enable] (see Table 198) is cleared by hardware. CLK_Enable (see Table 198) can be cleared via software if the unit is idle for some time. Clear CIRIR[EOT] and then

set CIRCR[SW_Reset] (see Table 198) prior to restarting a new transfer. If CIRIR[EOT] is not set, but CIRIR[EB] is set, the buffer needs reloading. If this is the last buffer to be loaded, mask the CIRIR[EB] interrupt. Load the buffer with new data. The CIRIR[EB] interrupt is cleared automatically when the buffer is loaded with new data.

Both of the CIR unit secondary interrupts have a mask bit (described in Table 200) that affects only the CIR primary interrupt signal to the interrupt controller unit. When EOT or EB secondary interrupt conditions occur, they always are indicated by CIRIR[EOT] and CIRIR[EB], whether or not the mask bits are asserted. Clearing the mask bit(s) enables the CIR unit interrupt signal to set the primary interrupt in the interrupt controller unit. Whenever a transmission is initiated, an EB interrupt is present unless the EB interrupt is masked, even if the contents of the N-Symbol counter register is a number less than the number of symbols contained in the buffer. For example, if 16 symbols are written to the buffer and the transmission length is set to 4, an immediate EB interrupt occurs when the transmission begins, even though the entire contents of the buffer have not been transmitted.

7.4.8 Timing

This section describes the relationship between modulation period, symbol duration and frequency, and the minimum and maximum symbol rate derived from configuring the CIR Unit registers.

An example of N1/N0 symbols and their relation to the modulation wave is shown in Figure 64.

7.4.8.1 Modulation Frequency

The input clock for this module is the 14.857-MHz clock (the UART clock), so each step of the counter represents 67.31 ns (= 1 / 14.857 MHz). The one exception is during S0/D0CS/C0 low-power mode. In this mode, the CIR runs at 12 MHz, and each step of the counter represents 83.33 ns (=1 / 12 MHz).

Since the counter width is 11 bits, the minimum frequency the unit can generate is:

14.857MHz / (2^11-1) = 7.258 kHz

and the maximum frequency is 14.857MHz/2 (comparator set to two).

The duty cycle of this wave is very flexible due to two comparators (11 bits each) that can control it. For example, if the pulse-width comparator is set to 930 (62.6 µs) and the modulation-period comparator to 1859 (125.1 µs), an 8-kHz modulation wave with 50% duty cycle is generated.

7.4.8.2 Symbol Frequency

This frequency is determined by the number of modulation waves per symbol.

In general, the symbol rate is calculated by the following equation:

$$\text{Symbol freq.} = \frac{1}{\langle \text{modulation period} \rangle \times \langle \text{modulation cycles} \rangle}$$

Since each symbol duration is represented by 7 bits (Table 192 - Table 195), one symbol duration is equal to maximum of 2^7 modulation cycles and a restriction of maximum 128 symbols is made.

As seen in Section 7.4.8.1, the minimum modulation frequency is 7.258 kHz ==> minimum symbol rate is:

$$\text{Min Symbol freq.} = \frac{1}{\langle \text{max modulation period} \rangle \times \langle \text{max modulation cycles} \rangle} = \frac{1}{\frac{1}{7.258K} \times 127} = 57.15Hz$$

When Manchester code has been used, each symbol is represented by two symbols (Table 188), so the above minimum symbol rate is:

$$\text{Min Symbol freq.} = \frac{1}{2 \times \langle \text{modulation period} \rangle \times \langle \text{modulation cycles} \rangle} = \frac{1}{2 \times \frac{1}{7.258K} \times 127} = 28.58Hz$$

The maximum symbol rate is when the selected symbol length register (CIRS0, CIRS1, CIRN0 or CIRN1) is set to 2, and the modulation frequency is set to its maximum frequency to 14.857 MHz / 2 resulting in a symbol frequency of 3.714 MHz.

Note

Actual symbol frequency is guaranteed to be +/- 5% of the nominal programmed value.

7.4.8.3 Buffer Transmission Time

The buffer transmission time is the time it takes to transmit all of the symbols in the current buffer. This time is calculated by multiplying the symbol count times the symbol length times the modulation period. The buffer transmission time must be greater than 800 μs to avoid interrupt latency issues when reloading the buffer. If the buffer transmission time is shorter than 800 μs then an underrun condition may occur, resulting in wrong data being transmitted until the buffer is reloaded.

7.4.9 Software-Supported Protocols

Although many of the protocols are supported directly in hardware, use care when configuring them to achieve the best results. This section provides a reference for some standard protocols that are commonly used.

7.4.9.1 RC5 Protocol

The protocol uses Manchester coding of a 36 kHz IR carrier frequency with the pulse/pause ratio of 1/3 or 1/4, to reduce power consumption.

All bits are of equal length of 1.778 ms in the RC5 protocol, with half of the bit time filled with a burst of the 36 kHz carrier and the other half being idle.

A logical zero is represented by a burst in the first half of the bit time. A logical one is represented by a burst in the second half of the bit time.

Example 1: Example of a 1, 0, 0, 1, 1, 0, 1, 0 Transmission

This example shows transmission of 1, 0, 0, 1, 1, 0, 1, 0:

The symbol N1 represents 1 and N0 represents 0.

32 cycles of the modulation wave for a 0 symbol (889 µS carrier burst)

CIR_N0SR = 0x0000_0020

32 cycles of the modulation wave for a 1 symbol (889 µS carrier burst)

CIR_N1SR = 0x0000_0020

Short symbols are not used
CIR_S0SR = “don’t care”
CIR_S1SR = “don’t care”

Modulation pulse is 25% duty cycle 413 / 4 =~ 103

CIR_PWC = 0x0000_0067

Frequency of modulation should be 36Khz (14.857 MHz / 36 kHz = 413)

CIR_MPC = 0x0000_019D

Activate the Manchester encoding:

CIR_CR = 0x0000_0033

We actually send 8 symbols
CIR_NSC = 0x0000_0008

symbols are N1,N0,N0,N1,N1,N0,N1,N0, 2 bits each
CIR_BR = 0x0000_4144

7.5 Register Descriptions

There are ten 32-bit registers within the CIR block: eight Control registers, one Data register and one Status register.

7.5.1 Register Summary

Table 189 shows the CIR register allocations in the PXA3xx processors memory map.

Table 189: Register Summary

Address	Description	Page
0x41D0_0000	CIR Pulse Width Comparator Register (CIRPW)	page 434
0x41D0_0004	CIR Modulation Period Comparator Register (CIRMP)	page 434
0x41D0_0008	CIR N0 Symbol Length Register (CIRN0)	page 434
0x41D0_000C	CIR N1 Symbol Length Register (CIRN1)	page 435
0x41D0_0010	CIR S0 Symbol Length Register (CIRS0)	page 435
0x41D0_0014	CIR S1 Symbol Length Register (CIRS1)	page 436
0x41D0_0018	CIR Buffer Register (CIRBUFF)	page 437
0x41D0_001C	CIR Number of Symbols Register (CIRNS)	page 436
0x41D0_0020	CIR Control Register (CIRCR)	page 438
0x41D0_0024	CIR Interrupt Register (CIRIR)	page 438
0x41D0_0028 - 0x41FF_FFFF	reserved	

7.5.2 CIR Pulse Width Comparator Register (CIRPW)

The CIR Pulse Width Comparator register is described in Table 190.

Table 190: CIRPW Bit Definitions

Physical Address 41D0_0000	CIRPW																				CIR										
User Settings																															
Bit	31	30	29	28	27	26	25	24	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																					PW										
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	1	1	1	1	1	1	1	1	1	1

Bits	Access	Name	Description
31:11	—	—	reserved
10:0	R/W	PW	Pulse Width Comparator: Determines the duration (in unit clock pulses) of logic high for the modulation wave period. Note that the minimum pulse width allowed is 2.

7.5.3 CIR Modulation Period Comparator Register (CIRMP)

The CIR Modulation Period Comparator register is described in Table 191.

Table 191: CIRMP Bit Definitions

Physical Address 41D0_0004	CIRMP																				CIR										
User Settings																															
Bit	31	30	29	28	27	26	25	24	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																					MP										
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1	1	1	1	1	1	1	1	1	1

Bits	Access	Name	Description
31:11	—	—	reserved
10:0	R/W	MP	Modulation Period Comparator: Determines the duration (in unit clock pulses) of the whole cycle time for the modulation wave.

7.5.4 CIR N0 Symbol Length Register (CIRN0)

The CIR N0 Symbol Length register is described in Table 192.

Table 192: CIRN0 Bit Definitions

Physical Address 41D0_0008																CIRN0												CIR						
User Settings																																		
Bit	31	30	29	28	27	26	25	24	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																									N0S						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:7	—	—	reserved
6:0	R/W	N0S	N0 Symbol: Specifies the number modulation cycles in the N0 symbol. The output during the N0 symbol period is logic low. Note that the minimum symbol length is 2.

7.5.5 CIR N1 Symbol Length Register (CIRN1)

The CIR N1 Symbol Length register is described in Table 193.

Table 193: CIRN1 Bit Definitions

Physical Address 41D0_000C																CIRN1												CIR						
User Settings																																		
Bit	31	30	29	28	27	26	25	24	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																								N1_Unmod	N1S						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7	R/W	N1_Unmod	N1 Symbol unmodulated: Enables the unmodulated representation of the N1 symbol. 0 = Output is modulated during the modulation waveform period of the symbol 1 = Output is a steady-state logic high during the modulation waveform period of the symbol.
6:0	R/W	N1S	N1 Symbol: Specifies the number modulation cycles in the N1 symbol. The output during the N1 symbol period is logic high. Note that the minimum symbol length is 2.

7.5.6 CIR S0 Symbol Length Register (CIRS0)

The CIR S0 Symbol Length register is described in Table 194.

Table 194: CIRS0 Bit Definitions

Physical Address 41D0_0010	CIRS0																								CIR							
User Settings																																
Bit	31	30	29	28	27	26	25	24	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																									S0S						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:7	—	—	reserved
6:0	R/W	S0S	S0 Symbol: Specifies the number modulation cycles in the S0 symbol. The output during the S0 symbol period is logic low. Note that the minimum symbol length is 2.

7.5.7 CIR S1 Symbol Length Register (CIRS1)

The CIR S1 Symbol Length register is described in Table 195.

Table 195: CIRS1 Bit Definitions

Physical Address 41D0_0014	CIRS1																								CIR							
User Settings																																
Bit	31	30	29	28	27	26	25	24	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																								S1_Unmod	S1S						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:8	—	—	reserved
7	R/W	S1_Unmod	S1 Symbol unmodulated: Enables the unmodulated representation of the S1 symbol. 0 = Output is modulated during the modulation waveform period of the symbol 1 = Output is a steady-state logic high during the modulation waveform period of the symbol.
6:0	R/W	S1S	S1 Symbol: Specifies the number modulation cycles in the S1 symbol. The output during the S1 symbol period is logic high. Note that the minimum symbol length is 2.

7.5.8 CIR Number of Symbols Register (CIRNS)

The CIR Number of Symbols register is described in Table 196.

Table 196: CIRNS Bit Definitions

Physical Address 41D0_001C	CIRNS	CIR

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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																									**SYMCNT**						
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:7	—	—	reserved
6:0	R/W	SYMCNT	Symbol Count: Total number of symbols to be sent. **Caution** This value must *not* be changed during operation

7.5.9 CIR Buffer Register (CIRBUFF)

The CIR Buffer register is described in Table 197.

Table 197: CIRBUFF Bit Definitions

Physical Address 41D0_0018	CIRBUFF	CIR

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	In_Symbols																															
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	In_Symbols	In Symbols: Contains the input symbols to be transmitted, where each symbol represented by two bits (refer to Table 187). This results in a maximum of 16 symbols being loaded at one time.

7.5.10 CIR Control Register (CIRCR)

This register contains the unit-configuration, the CIR-enable, and clock-enable bits. The details of the register are shown in Table 198.

Table 198: CIRCR Bit Definitions

Physical Address 41D0_0020	CIRCR	CIR

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																										SW_Reset	Manchester	reserved	reserved	CLK_Enable	CIR_Enable
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	0	0	0	0	0

Bits	Access	Name	Description
31:6	—	—	reserved
5	R/W	SW_Reset	Software Reset: Software reset is accomplished by clearing this bit and then re-setting it. Upon software reset, all functional registers in the CIR unit are set to their default value (the registers are not affected). Note that software reset must be set between transmissions (i.e. when processing end of transfer interrupts).
4	R/W	Manchester	Manchester Coding Enable: This enables the Manchester coding machine, so each symbol from the input-symbol block is actually two symbols at the symbol generator. See Figure 65 for an example of Manchester code. Refer to Section 7.4.6 for a description of the Manchester machine. 0 = Disable manchester encoding 1 = Enable Manchester encoding
3	—	—	**reserved**. This bit must be cleared to 0b0.
2	—	—	**reserved**. This bit must be cleared to 0b0.
1	R/W	CLK_Enable	Clock Enable: Clock enable bit for the module. If the clock is enabled and the CIR enable (bit 0) is not enabled, the output is held at a logic low. This bit is used in conjunction with CIR_Enable to determine the operation of the CIR controller. Refer to Table 199 for configuration information.
0	R/W	CIR_Enable	CIR Enable: When this bit is set CIR unit transmission starts. When CIRNS[SYMCNT] reaches zero, CIR_Enable is cleared. CIR_Enable is used in conjunction with CLK_Enable to determine the operation of the CIR controller. Refer to Table 199 for configuration information.

CIR unit output dependence on the CIR-enable bit and clock-enable bit is shown in Table 199.

Table 199: Enable Bits

CIR_Enable	CLK_Enable	CIR Output
0	0	No transmissions occur.
0	1	Continual S0 symbols (logic low) are transmitted.
1	0	reserved
1	1	If CIRNS[SYMCNT] > 0, the contents of CIRBUFF are transmitted. If CIRNS[SYMCNT] = 0, continual S0 symbols (logic zero) are transmitted.

As shown in Table 198, clearing SW_Reset and then re-setting it performs a software reset on the CIR controller. If SW_Reset remains cleared, the unit cannot operate.

This software reset does not affect the configuration registers. Asserting this reset causes the unit to return to its initial state, but the configuration registers remain as they were.

If the CIR-enable or clock-enable bits are turned off before the unit completes transmission, software reset should be asserted before turning on these bits again. This allows the unit to start transmitting from the beginning.

7.5.11 CIR Interrupt Register (CIRIR)

The CIR Interrupt register is described in Table 200.

Table 200: CIRIR Bit Definitions

Physical Address 41D0_0024	CIRIR																					CIR								
User Settings																														
Bit	31	30	29	28	27	26	25	24	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																						EOT	EB	reserved						EOT_Mask	EB_Mask
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
31:4	—	—	reserved
9	R/Write 1 to clear	EOT	End-of-Transmission Interrupt: Indicates that entire data burst has been transmitted (CIRNS[SYMCNT] = 0b00_0000). Software must clear this interrupt and assert software reset before starting the next data burst. When EOT is set, an interrupt request is made to the interrupt controller if it is unmasked (EOT_Mask is clear). 0 = All symbols not yet transmitted. 1 = All symbols transmitted.

Table 200: CIRIR Bit Definitions (Continued)

Physical Address 41D0_0024	CIRIR																					CIR									
User Settings																															
Bit	31	30	29	28	27	26	25	24	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																						EOT	EB	reserved						EOT_Mask	EB_Mask
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?	0	0

Bits	Access	Name	Description
8	R	EB	Empty Buffer Interrupt: Indicates that the contents of the Buffer register (CIRBUFF) has been transferred to the shift register and the Buffer register is ready for new data. When EB is set, an interrupt request is made to the interrupt controller if it is unmasked (EB_Mask is clear).
1	R/W	EOT_Mask	0 = The CIR interrupt is asserted when EOT is set. 1 = The CIR interrupt is *not* asserted when EOT is set.
0	R/W	EB_Mask	0 = The CIR interrupt is asserted when EB is set. 1 = The CIR interrupt is *not* asserted when EB is set.

8 Pulse-Width Modulator Controller

The PXA3xx processor family pulse-width modulator (PWM) controller generates four independent PWM signals. Specific applications of the PWM controller vary. Examples include:

- Controlling the brightness of an LED output by modulating the “on” time
- LCD contrast control

8.1 Overview

The PXA3xx processor family 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 a multi-function pin. Because each PWM contains identical circuitry, this chapter describes a generic PWMx, where x is 0, 1, 2, or 3.

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 internal clock source (PSCLK_PWM) used to source the PWMx and drive the PWM_OUTx signals to a steady high or low state.

The frequency range supporting a 50% duty cycle varies from 198.4 Hz to 6.5 MHz. Other duty-cycle options depend on the choice of preferred frequency.

There are no differences in PWM controller functionality among the PXA3xx processor family.

8.2 Features

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

8.3 Signal Descriptions

Output signals are the four single-bit output channels defined as PWM_OUT0, PWM_OUT1, PWM_OUT2, and PWM_OUT3 (see Table 201). These signals are sent to multi function pins (MFPs).

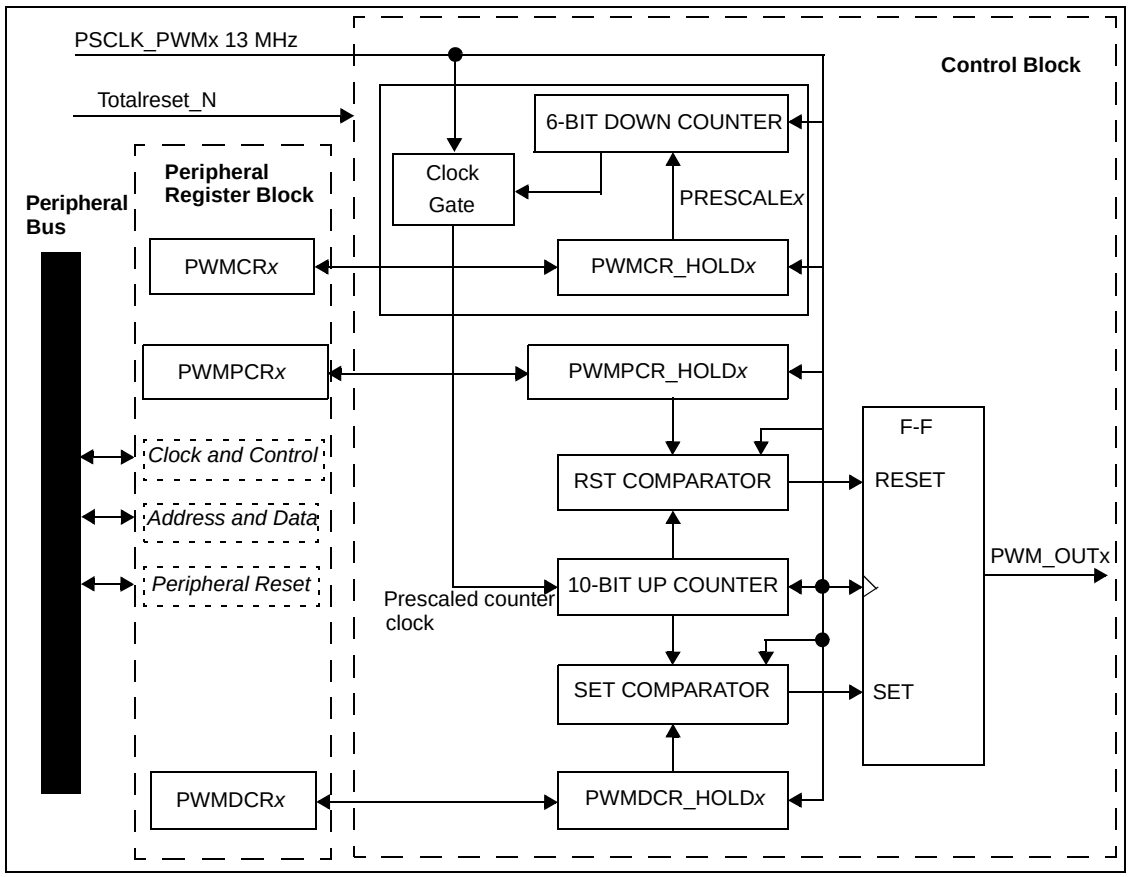
Table 201: Pulse Width Modulator I/O Signal Descriptions

Signal Name	Direction	Description
PWM_OUT0	Output	Pulse-width modulated signal for PWM 0
PWM_OUT1	Output	Pulse-width modulated signal for PWM 1
PWM_OUT2	Output	Pulse-width modulated signal for PWM 2
PWM_OUT3	Output	Pulse-width modulated signal for PWM 3

8.4 Operation

Figure 66 shows the block diagram for the PWM control logic.

Figure 66: PWMx Block Diagram



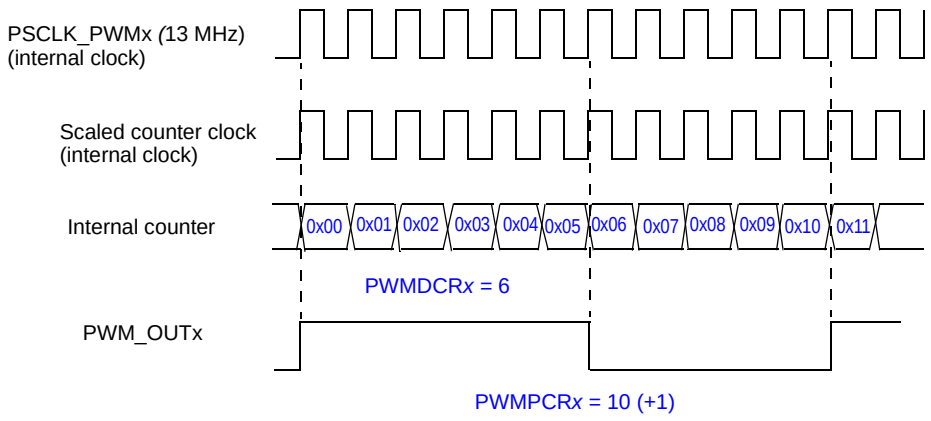
To program the PWM controller, determine the period and pulse-width values. The period value is based on two registers, PWMPCR*x* and PWMCR*x*.

The output waveform in Figure 67 is derived by writing the PWMPCR*x* register with a value of 10(0x00A) and writing the PWMDCR*x* register with 0x06.

Pulse-width modulated signals, once programmed, output a specified waveform until the value in any associated register is altered. The time from the register change until the associated signal change depends on PWMPCRx[PV]:

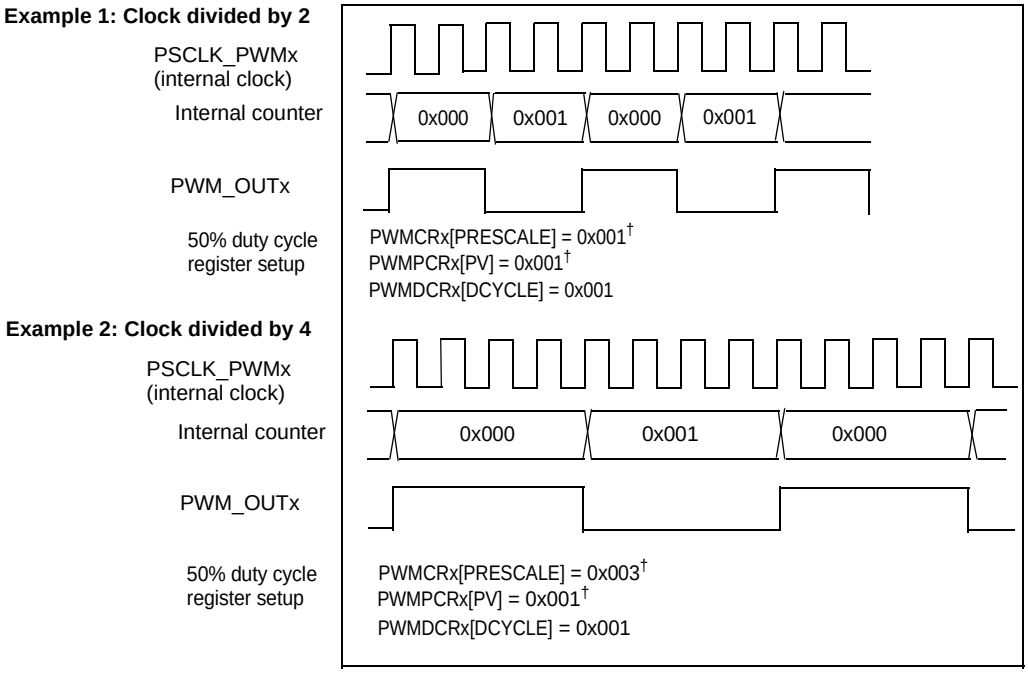
- For PWMPCRx[PV] values 0x005 and larger—After a register value is altered, the PWM_OUTx signal changes when the previously programmed waveform cycle is complete.
- For PWMPCRx[PV] values less than 0x005—After a register value is altered, the PWM_OUTx signal changes after two waveform cycles.

Figure 67: Basic Pulse Width Modulated Waveform



Programming PWMCRx[PRESCALE] configures the prescaled counter clock (refer to Figure 66). Two timing examples are provided in Figure 68. 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. See Section 8.4.2 for more information regarding the calculation of waveform values.

Figure 68: Effects of PWMCRx Settings



† The effective value of this field is one greater than the programmed value. Refer to register descriptions.

The PWMx clock is controlled by the D0CKEN_B register.

8.4.1 Reset Sequence

During system reset, the PWMCR*x* and PWMDCR*x* registers are reset to 0x0 and the PWMPCR*x* register is set to 0x004. Reset places the PWM_OUTx channels in a steady low state.The PWM_OUTx channel remains reset to 0x0 until the PWMDCR*x* register is programed with a non-zero value. Therefore, system reset results in no pulse-width modulated signal.

8.4.2 Programming Considerations

The PWMs use three registers to configure the output of each PWMx signal: PWMCRx, PWMDCRx, and PWMPCRx.

PWMx timing is based on the input clock to the PWMx controller, PSCLK_PWMx, 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 and period of the PWMx signal.

- Use the following equation to calculate the frequency of the scaled counter clock:
  Scaled counter clock frequency = 13 MHz / (PWMCRx[PRESCALE] + 1)
- Use the following equation to calculate the cycle time of the scaled counter clock:
  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_OUTx 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 66 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.

- Use the following equation to calculate the period of the PWM:
  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)
- Calculate values based on the necessary PWM cycle time and duty cycle with the following equations:
  Choose a PWMCRx[PRESCALE] value that is appropriate for all your PWM outputs.
  PWMPCRx[PV] = PWM cycle time / (76.0nS X (PWMCRx[PRESCALE]+1)) - 1
  Duty cycle time = PWM cycle time * Duty Cycle%
  PWMDCRx[DCYCLE] = Duty cycle time / (76.0nS X (PWMCRx[PRESCALE]+1))

For example, to create a 60% duty cycle 500 kHz signal, set PWMCRx[PRESCALE] to 0, PWMPCRx[PV] to 26 (0x1A), and PWMDCRx[DCYCLE] to 16 (0x10).

Note

To produce a toggle of the signal, the value of the PWMPCR*x*[PV] must be equal to or greater than PWMDCR*x*[DCYCLE]. If PWMPCR*x*[PV] is less than PWMDCR*x*[DCYCLE], the PWM_OUTx signal remains high. If PWMDCRx[DCYCLE] equals zero, the signal remains low.

The PWMDCRx[FD] bit determines if PWM_OUTx is always asserted. When this bit is set, PWM_OUTx remains high until PWMDCRx[FD] is cleared.

8.4.3 Power Management

Each PWMx can be disabled through a pair of clock-enable bits (see the clocks chapters). If the clock is disabled, the unit shuts down abruptly or gracefully (as selected by the PWMPCRx[SD] described in Section 8.5.2).

- For a graceful shutdown (PWMPCRx[SD]=0), PWM_OUTx completes the current cycle before it stops.
- For an abrupt shutdown (PWMPCRx[SD]=1), PWM_OUTx stops after at most one PSCLK_PWM and the internal counters are reset.

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.

8.5 Register Descriptions

Each PWMx contains three registers that control the clock, the period, and the duty cycle timing of the PWM_OUTx. The following sections provide a summary (Table 202) and detailed register descriptions (Section 8.5.2, PWM Control Registers (PWMCRx), Section 8.5.3, PWM Duty Cycle Registers (PWMDCRx), and Section 8.5.4, PWM Period Control Registers (PWMPCRx)).

8.5.1 Register Summary

Table 202 shows the registers associated with the PWM controller and the physical addresses to access them.

Table 202: PWM Control Registers

Address	Description	Page
0x40B0_0000	PWM 0 Control Register (PWMCR0)	page 447
0x40B0_0004	PWM 0 Duty Cycle Register (PWMDCR0)	page 448
0x40B0_0008	PWM 0 Period Control Register (PWMPCR0)	page 448
0x40B0_000C	reserved	—
0x40B0_0010	PWM 2 Control Register (PWMCR2)	page 447

Table 202: PWM Control Registers (Continued)

Address	Description	Page
0x40B0_0014	PWM 2 Duty Cycle Register (PWMDCR2)	page 448
0x40B0_0018	PWM 2 Period Control Register (PWMPCR2)	page 448
0x40B0_001C–0x40BF_FFFC	reserved	—
0x40C0_0000	PWM 1 Control Register (PWMCR1)	page 447
0x40C0_0004	PWM 1 Duty Cycle Register (PWMDCR1)	page 448
0x40C0_0008	PWM 1 Period Control Register (PWMPCR1)	page 448
0x40C0_000C	reserved	—
0x40C0_0010	PWM 3 Control Register (PWMCR3)	page 447
0x40C0_0014	PWM 3 Duty Cycle Register (PWMDCR3)	page 448
0x40C0_0018	PWM 3 Period Control Register (PWMPCR3)	page 448
0x40C0_001C–0x40CF_FFFC	reserved	—

8.5.2 PWM Control Registers (PWMCRx)

The PWMx Control registers (PWMCR*x*), defined in Table 203, configure the behavioral characteristics of the PWMx shutdown response and the divisor for the input clocks to the PWMx control unit that configures the frequency of the scaled counter clock.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 203: PWMCR0x Bit Definitions

Physical Address 0x40B0_0000 0x40C0_0000 0x40B0_0010 0x40C0_0010	PWMCR0 PWMCR1 PWMCR2 PWMCR3	PWM 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																									**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 PWMx when the PXA3xx processor family stops the clocks to the PWM. 1 = Abrupt shutdown of PWMx when the PXA3xx processor family stops the clocks to the PWM.
5:0	R/W	PRESCALE	The scaled counter clock frequency is: PSCLK_PWMx / (PRESCALE*x +1)*

8.5.3 PWM Duty Cycle Registers (PWMDCRx)

The PWMx Duty Cycle registers (PWMDCR*x*), defined in Table 204, configure the duty cycle of the corresponding PWM_OUTx signals.

PWMDCRx[DCYCLE] specifies the number of scaled counter clocks that PWM_OUTx is asserted during each cycle of the PWM_OUTx. Refer to Section 8.4.2 and Figure 66 for details on calculating the value of PWMDCRx[DCYCLE].

If PWMDCRx[FD] is set, PWM_OUTx 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_OUTx is a function of PWMDCRx[DCYCLE].

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 204: PWMDCRx Bit Definitions

Physical Address		PWM Controller
0x40B0_0004 0x40C0_0004 0x40B0_0014 0x40C0_0014	PWMDCR0 PWMDCR1 PWMDCR2 PWMDCR3	

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																					FD	DCYCLE									
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0

Bits	Access	Name	Description
31:11	—	—	reserved
10	R/W	FD	Full Duty Cycle: 0 = PWM_OUTx is determined by DCYCLE value. 1 = PWM_OUTx is continuously asserted.
9:0	R/W	DCYCLE	Duty Cycle of PWM_OUTx: 0 = PWM_OUTx is continuously de-asserted. 1 = PWM_OUTx is high for the number of 13-MHz clock periods equal to PWMDCRx[DCYCLE] x (PWMCRx[PRESCALE]+ 1), If FD is set, DCYCLE has no effect on the output of PWMx.

8.5.4 PWM Period Control Registers (PWMPCRx)

The Period Control registers (PWMPCR*x*), defined in Table 205, configure the cycle time of the corresponding PWM_OUTx signals.

PWMPCRx[PV] specifies the number of scaled counter clocks (plus one) in each cycle of the PWM_OUTx. Refer to Section 8.4.2 and Figure 66 for details on calculating the value of PWMPCRx[PV].

If this register is cleared the PWM_OUTx signal maintains in a high state.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 205: PWMPCRx Bit Definitions

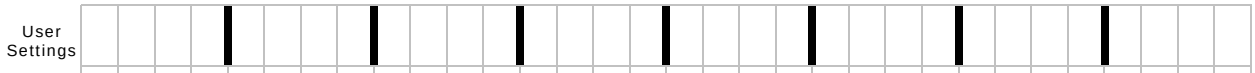
Physical Address		PWM Controller
0x40B0_0008 0x40C0_0008 0x40B0_0018 0x40C0_0018	PWMPCR0 PWMPCR1 PWMPCR2 PWMPCR3	

User Settings																															
Bit	31	30	29	28	27	26	25	24	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																						PV									
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	1	0	0

Bits	Access	Name	Description
31:10	—	—	reserved

Table 205: PWMPCRx Bit Definitions (Continued)

Physical Address	PWMPCRx	PWM Controller
0x40B0_0008 0x40C0_0008 0x40B0_0018 0x40C0_0018	PWMPCR0 PWMPCR1 PWMPCR2 PWMPCR3	



Bits	Access	Name	Description
9:0	R/W	PV	Period Value: The value of scaled clock cycles per cycle of PWM_OUTx plus one. If all zeros are written to this register the signal remains high.

9 Universal Subscriber ID Controller

This chapter describes the Universal Subscriber Identity Module (USIM) controller and registers that are supported by the processor. There are two instances of the USIM controller on the processor. The only difference between the two USIM controllers is their memory-map addressing (see Table 210). The first controller is denoted by the use of the prefix “U” with the signal names and the second controller is denoted by the use of the prefix “SC” with the signal names.

9.1 Overview

The USIM controller is a primary device and communication interface for a GSM mobile handset. The USIM controller supports communication with SmartCards as specified in the standard *ISO 7816-3* and technical specification *3G TS 31.101* of the 3rd Generation Partnership Project.

SmartCards are used in many applications and the GSM network USIM SmartCard is only one of many applications. SmartCards usually consist of CPU, Flash memory, and a serial-communication interface device similar to the one described in this chapter. More sophisticated SmartCards contain PLL for frequency enhancement. Encryption accelerators also can exist in a SmartCard since many of their applications are security oriented. In all SmartCard applications, the physical layer and data link layer are identical. Familiarity with the standards cited in this section will enable readers to better understand this chapter.

Software controls the session between the USIM controller and the SmartCard by updating the USIM controller registers. Choosing protocol type and parameters, receiving or sending a byte to/from the SmartCard, activating/deactivating the SmartCard, setting transmitter/receiver baud rates, etc., are accomplished via read/write operations to the USIM controller registers. Transforming the byte convention (inverse to direct and vice-versa, according to the session convention) is performed within the USIM controller. Therefore, software does not have to perform format inversion before a character is received. The USIM controller provides functionality to support the above standards, but it is software that ensures that the standards are met.

9.1.1 PXA3xx Processor Differences

Table 206 shows the USIM controller differences among the PXA32x, PXA31x, and PXA30x processors. Refer to each individual register for other operating differences.

Table 206: PXA3xx Processors Feature Differences

Feature	PXA30x	PXA31x	PXA32x
USIM Voltage Select	U_VS0NOTE:	U_VS0NOTE:	U_VS0, U_nVS1 and U_nVS2
NOTE: U_nVS1 and U_nVS2 must be implemented through the use of GPIOs and controlled via software for proper USIM operation.			

9.2 Features

- Compatible with any USIM SmartCard 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 SmartCard 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 SmartCard clock frequency
- Supports any combination of the following clock-rate conversion factor F and bit-rate adjustment factor D:
  - F = {372, 512, 558}
  - D = {1,2,4,8,12, 16, 20, 32}
- 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 Timeout period
- Programmable CPU interrupt request on an error-signal detection
- Programmable CPU interrupt request when a SmartCard is connected

9.3 Signal Descriptions

This section describes the signal pins that are used by the USIM controller. Table 207 lists all the PXA3xx Processor Family pins that connect between the USIM controller and the USIM SmartCard.

Note

The Unit Signal Name is used throughout this chapter to describe the function of either of the Alternate Function Name signals operation.

Table 207: USIM Controller Signals Summary

Signal Name	Direction	Description	Notes
U_IO, SC_IO	Bidirectional	**USIM I/O Data** Receive and transmit data signal. The bi-directional signal should be connected directly to the USIM SmartCard. When asserted, the UIO signal is forced to Vlow. When de-asserted, the UIO signal is pulled up by a 20 k-ohm external resister. The USIM SmartCard can act as a pulldown on the UIO signal. **NOTE:** When the USIM SmartCard and the PXA3xx Processor Family operate at different voltage levels, the I/O voltage level of the SmartCard must be a voltage level that the processor supports, that is, 1.8 V or 3.0 V.	

Table 207: USIM Controller Signals Summary (Continued)

Signal Name	Direction	Description	Notes
U_VS0, SC_VS0	Output	**USIM Voltage Select 0** Used as the select signal to a power transistor that controls the voltage level that is supplied to the SmartCard: 0 = OFF 1 = Selects zero-volts to the USIM SmartCard power-supply (VCC)	2, 5
U_nVS1, SC_nVS1	Output	**USIM Voltage Select 1** Used as the select signal to a power transistor that controls the voltage level that is supplied to the SmartCard: 0 = Selects 1.8V to the USIM SmartCard power-supply pin (VCC) 1 = OFF	2, 4, 5
U_nVS2, SC_nVS2	Output	**USIM Voltage Select 2** Used as the select signal to a power transistor that controls the voltage level that is supplied to the SmartCard: 0 = Selects 3.0V to the USIM SmartCard power-supply pin (VCC) 1 = OFF	2, 4, 5
U_CLK, SC_CLK	Output	**SmartCard clock.** UCLK should be connected directly to the SmartCard clock pin. The SmartCard cannot use any other clock source.	
U_nRST, SC_nRST	Output	**SmartCard reset.** nURST should be connected directly to the SmartCard reset pin: 0 = Resets the SmartCard 1 = Normal operation	
SC_DETECT	Output	USIM Detection for SmartCard present: 0 = SmartCard not detected 1 = SmartCard detected	

Table 207: USIM Controller Signals Summary (Continued)

Signal Name	Direction	Description	Notes
NOTE: 1. There are two SmartCard USIM controllers in the processor, both are identical in function. For one of the controllers, the signal names are preceded by a “U_”. The signal names for the other SmartCard controller are the same except preceded by an “SC_”. See the “Pin Descriptions and Control” chapter in *PXA3xx Processor Family Vol. I: System and Timer Configuration Developers Manual* for alternate function assignments. 2. Use caution when placing the processor into S3/D4/C4 power mode (Refer to the Clock Controllers and Power Management chapter in *PXA3xx Processor Family Vol. I: System and Timer Configuration Developers Manual* for detailed description of S3/D4/C4). The voltage control pins are powered from the VCC_IO power domain, but in S3/D4/C4 mode this is turned off. Powering off the voltage select pins (UVS0, nUVS1, nUVS2 or any GPIOs acting as nUVS1 and nUVS2) can cause shorting of the power supplies that are switched to provide power to the SmartCard. To prevent these electrical problems: • GPIOs must be correctly configured prior to entering S3/D4/C4 mode via software. See the “General-Purpose I/O Unit” chapter in *PXA3xx Processor Family Vol. I: System and Timer Configuration Developers Manual* for more details. • The external power supply must be turned off before (or at the same time as) the processor enters S3/D4/C4 power mode using USCCR[VCC] bit field. 3. Voltage level for these signal pins are set off chip via the VCC_CARDx power domain pins. The signal pins work at either 1.8 V or 3.0 V. 4. U_nUVS1 and U_nUVS2 are only valid on the PXA32x processor. On the PXA31x processor and PXA30x processor external signals (for example, GPIOs) must be used to properly signal the external Power Management IC (PMCI) supplying the SmartCard. SC_nVS1 and SC_nVS2 are available and do not have these restrictions. 5. Required voltage control of the smart card is handled through the I2C interface or any other method external to the USIM controller.			

9.4 Operation

9.4.1 USIM SmartCard Interface Description

A SmartCard has three signal pins and two power supplies as listed in Table 208.

Table 208: USIM SmartCard Pinout

SmartCard Pin	Function	Pin Direction	Processor Pin
I/O	Data input/output (bidirectional) pin with a pull up transistor. See *ISO 7816-3* for pin specifications.	USIM SmartCard -> processor	U_IO
SmartCard_RST	USIM SmartCard reset.	processor -> USIM SmartCard	U_nRST
CLK_SmartCard	Clock input. Frequencies are between 1-5 MHz. See *ISO 7816-3* for pin specifications. SmartCards manufactured before April 2000 may have a frequency limitation of 4 MHz. Clock stops on low or high phase are supported.	processor -> USIM SmartCard	U_CLK

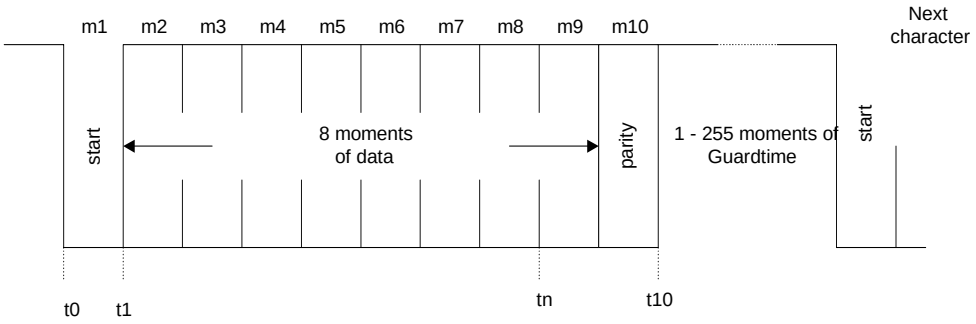
Table 208: USIM SmartCard Pinout (Continued)

SmartCard Pin	Function	Pin Direction	Processor Pin
VCC	SmartCard power supply. Supplies 0V, 1.8V, 3 V with max currents 0mA, 30mA, 50mA, respectively according to SmartCard class (B,C). See *ISO 7816-3* for pin specifications.	Power supply -> SmartCard	VCC_CARDx
GND	Mutual USIM SmartCard, SIM interface and VCC ground reference voltage.	Power supply -> SmartCard	VSS_CARDx

9.4.2 Coding Conventions

The USIM controller performs serial-to-parallel conversion with optional format inversion on data characters received from the USIM SmartCard, and parallel-to-serial conversion with optional format inversion on data characters transmitted to the USIM SmartCard. 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 or reception of a single bit is called either an *Elementary Time Unit (etu)* or a *moment*.

Figure 69: Byte Sent Versus Time



NOTE: m1,m2...m10 stands for moments 1-10.

The start bit is mandatory. At least one Guard bit after the parity is mandatory. Extra guard time is optional. The USIM controller allows software to add up to 255 Extra Guard Time moments. See Section 9.5.12.

Data bits may appear in two possible conventions as selected by the LCR[ORDER,INVERSE] bits (see Section 9.5.9):

- Direct Convention - Data bits sent on moments m2-m9 (see Figure 69) are transmitted LSB first-MSB last (that is, 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 69) are transmitted MSB first-LSB last (that is, 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 SmartCard to the USIM controller, after reset de-assertion, determines the convention to be used throughout the entire session. A SmartCard that encodes and decodes

data using the direct convention sends 0x3B. A SmartCard that uses the inverse convention sends 0x3F.

One of the following scenarios is possible when receiving this first character:

1. Both the USIM controller and SmartCard are set to the same convention and the first character is received correctly (that is, received data was either 0x3F or 0x3B with no parity error).
2. USIM controller and SmartCard are set to opposite conventions - the USIM controller asserts a parity error since it expects the opposite parity bit convention.

9.4.3 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 controller. The protocols for transmit and receive can be selected via the TX_T1 and RX_T1 bits, respectively, in the Section 9.5.9, USIM Line Control Registers (LCRx). The USIM controller does not analyze or generate the data content or block structure that is transmitted or received using the protocols. Software generates and analyzes the 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.

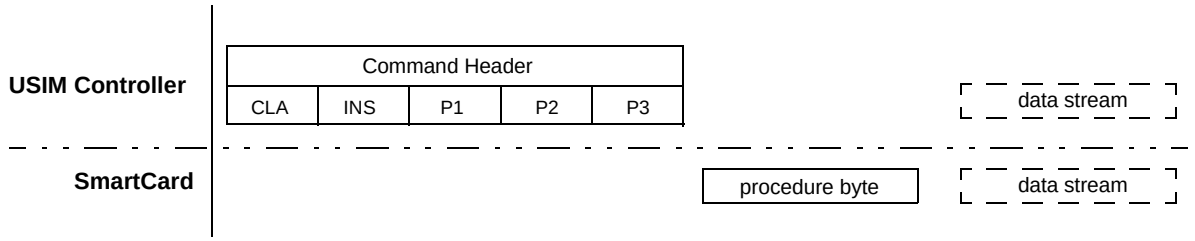
In the T=0 protocol, the USIM controller sends a command to the SmartCard by software writing a five-byte command header to the TXFIFO. The header tells the SmartCard what to do. The software-controlled command processing continues with the transfer of a variable number of data bytes in one direction under the control of procedure bytes sent by the SmartCard. See Figure 70. 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; these are inserted automatically by the USIM controller.

In the T=1 protocol, data is sent via a sequence of bytes known as a *block*. By analyzing block content, software should detect at any given time the proper direction of the data flow (transmit or receive). For example, software detects the length of the block sent from the SmartCard to the USIM controller by analyzing the third byte of the received block (marked as a LEN byte). Each character has a special meaning according to its position in the block. See Figure 71. 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 (automatically inserted by the USIM controller) following the parity bit. The USIM controller supports DMA transfers in for long T=1 protocol blocks.

Receive and transmit activity for each mode can be monitored using one of three techniques: interrupt, polled, or DMA. See Section 9.4.6 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.

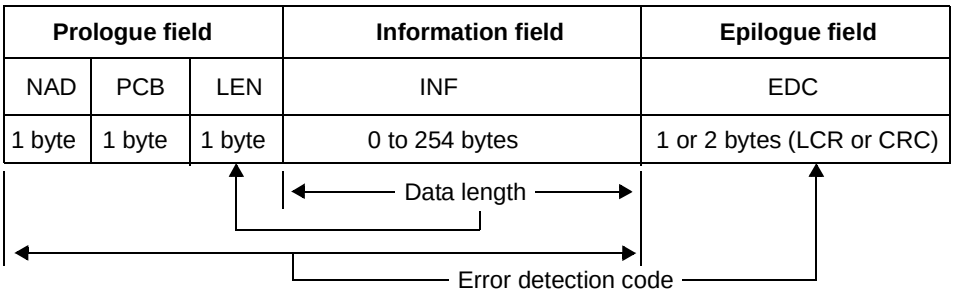
Figure 70: T=0 Protocol Communication Method



Note

The data stream is transmitted from either the USIM controller to the SmartCard, or from the SmartCard to the USIM controller, depending on the command type (Read or Write)

Figure 71: Complete Block Structure for T=1 Protocol



Note

Refer to the *ISO 7816-3* standard for the definition of the terms used in Figure 70 and Figure 71.

9.4.3.1 Errors

The Section 9.5.8, USIM Error Control Registers (ECRx), and the Section 9.5.4, USIM Interrupt Enable Registers (IERx), allow software to control how errors are handled.

In T=0 protocol, the following errors are signaled to the SmartCard when the USIM controller is in receiver mode:

- The parity check fails; that is, exclusive OR-ing of 8 data bits and a parity bit yields the same value as the eps bit (see Bit no.1).
- The RXFIFO is full and another byte is being received.

In T=0 protocol, the USIM controller senses for an error signaled by the SmartCard on the U_IO pin.

9.4.3.1.1 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 controller senses for an error signal, V_{low} during the second mandatory guard time moment, on the U_IO pin. If an error signal is sensed, the USIM controller transmitter resends the previous byte. When the number of “resends” meets the T=0 error trigger level, ECR[T0ERR_TL], transmission stops, an interrupt request is generated if enabled, and the LSR[T0ERR] bit is asserted. Software then detects whether to proceed with the next byte or re-transmit the same byte via 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.

9.4.3.1.2 Parity Error

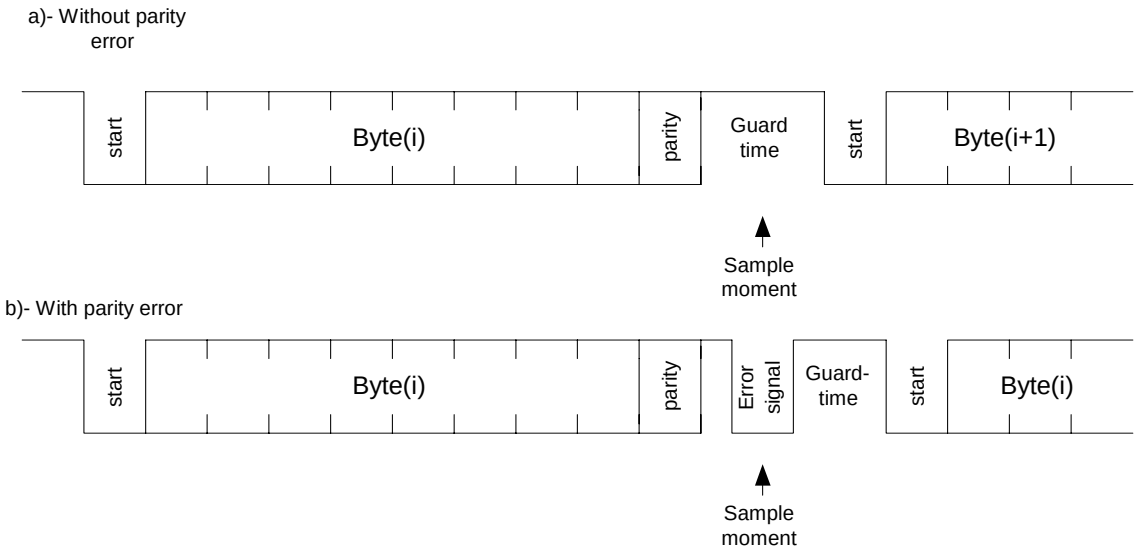
In T=0 mode, the USIM controller automatically signals the SmartCard in the event of a parity error by placing V_{low} on the U_IO pin during guard time (see Figure 72). The erroneous byte does not enter the RXFIFO. Retransmission by the SmartCard of the previous byte is expected, so no data is lost. When the number of repetitions meets the parity-error trigger level, ECR[PE_TL], an interrupt request is generated (if enabled) and the LSR[PERR] bit is asserted. A parity error generates an error signal to the SmartCard regardless of the parity-error trigger value.

Note

Reception does not stop after a parity-error interrupt request is generated. If the parity error continues to repeat, the USIM controller continues to signal the SmartCard on each occurrence. Software can either reset the SmartCard or receive the character with a parity error by switching to T=1 mode and then reverting to T=0 mode. Software can check the RBR[PERR] bit to validate the character was received with parity via the T=1 mode.

Error signaling or sensing is not performed in the T=1 protocol. When a parity error occurs, the erroneous byte enters the RXFIFO, the LSR[PERR] bit is asserted, and a parity-error interrupt request is generated if enabled. The parity error-trigger level has no effect in T=1 mode. The erroneous bytes are indicated in the RXFIFO by the assertion of RBR[PERR], see USIM Receive Buffer Registers (RBRx). The total number of bytes with parity errors in the RXFIFO can be determined by FSR[PERR_NUM].

Figure 72: T=0 Character Transmission Format



Note

The ISO spec requires multiple samples of U_IO for each moment. Refer to Section 9.4.4.2.1, Determining the Baud Rate, on page 465 and Table 228, FLRx Bit Definitions, on page 491 for more information on sampling requirements.

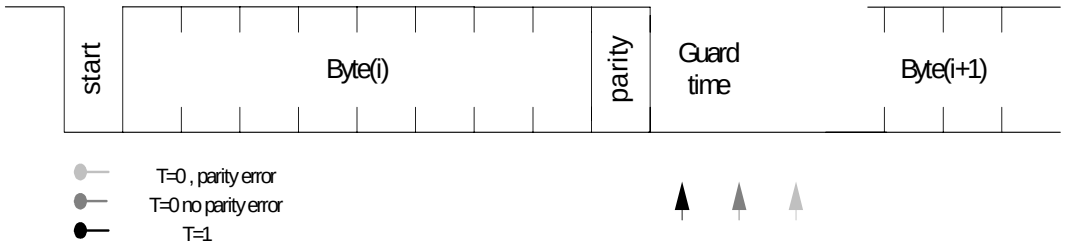
9.4.3.1.3 Overrun Error

The RXFIFO can store up to 16 bytes. If the RXFIFO is not emptied before the 17th byte received from the SmartCard, the LSR[OVRN] bit is asserted and an overrun interrupt request is generated (if enabled), and the 17th byte does not enter the RXFIFO. In T=0 mode, the USIM controller signals the SmartCard by placing V_{low} on the U_IO pin during Guard time (see Figure 72).

9.4.3.1.4 Framing Error

After a byte is received, the USIM controller verifies that the U_IO pin is kept at V_{high} during the last moment of mandatory Guard time. If the U_IO pin drops to V_{low}, the LSR[FRAMERR] bit asserts and an interrupt request is generated, if enabled. See Figure 73.

Figure 73: Framing Error



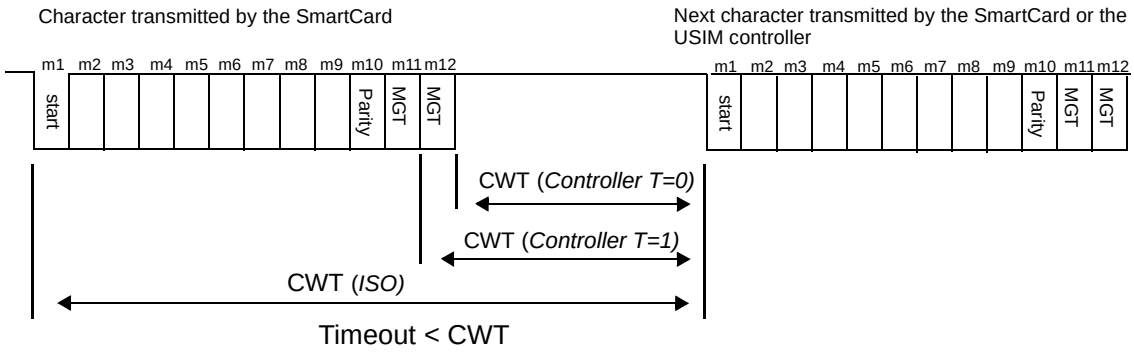
9.4.3.2 Waiting Times

Waiting times are protocol-defined parameters from standard *ISO 7816-3* and specification *3G TS 31.101.* Waiting times can be selected in the Character Waiting Time Register (CWTR, see Section 9.5.18) and the Block Waiting Time Register (BWTR, see Section 9.5.19). To avoid unpredictable results, software must set the waiting times before changing the baud rate (see Section 9.4.4.2).

9.4.3.2.1 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. 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 Table 229, CWTRx Bit Definitions, on page 492 and Figure 74.

Figure 74: Character Waiting Time, USIM Controller



For T=0: Mandatory Guard Time (MGT) requires at least 2 Guard bits
For T=1: Mandatory Guard Time (MGT) requires at least 1 Guard bit

Note

The USIM controller measures the CWT differently from the way it is defined in the ISO standard, but the effect is the same. Figure 74 shows the difference. Use the following formulas to convert the ISO standard CWT to CWT as used in the processor USIM controller:

Equation 4: **For T=0:** CWT = CWT(ISO) - 12
Equation 5: For T=1: CWT = CWT(ISO) - 11

As shown in Figure 74, CWT does not measure the time between two characters transmitted by the USIM controller, but instead measures the time after the USIM controller receives a character transmitted by the SmartCard. Software ensures that data is delivered continuously and that the upper software layers (data, application, etc.) are standard compliant.

If the CWT is less than the Block Guard Time described in Section 9.4.3.2.2, a CWT violation occurs each time the USIM controller transmits data after the SmartCard has finished transmitting data. This violation should not be an issue for the T=0 protocol since the ISO specification defines a WWT/CWT of 9600 etu. This violation may be an issue for the T=1 protocol since the 3G specification defines 12 etu<CWT<43 etu. Ignore CWT violations when the USIM controller is transmitting data.

A timeout interval can be specified by software writing to the USIM Timeout Register (TOR, see Section 9.5.14). Measurement of timeout interval begins after reception of a character is complete. Measurement stops with the renewal of transmit or receive activity before the timeout interval has expired or when the timeout interval expires. When the timeout interval expires, the LSR[CWT] bit asserts and a CWT interrupt request is generated, if enabled.

When using the CWT interrupt request while receiving data, software can identify either the end of a block or loss of communication synchronization (that is, software expects more characters to come after the SmartCard finishes transmitting). By using a CWT interrupt request while transmitting data,

software can monitor performance by verifying that the transmission was not stopped for more than the CWT etu.

9.4.3.2.2 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)*. Software specifies the Block Guard Time to between 0-255 moments by writing to the Block Guard Time register (BGTR, see Section 9.5.13). The USIM controller automatically ensures that BGT has elapsed before transmission is initiated. The USIM controller is ready to receive a character immediately after transmission even when the SmartCard does not obey BGT. 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 9.5.13 and Figure 75.

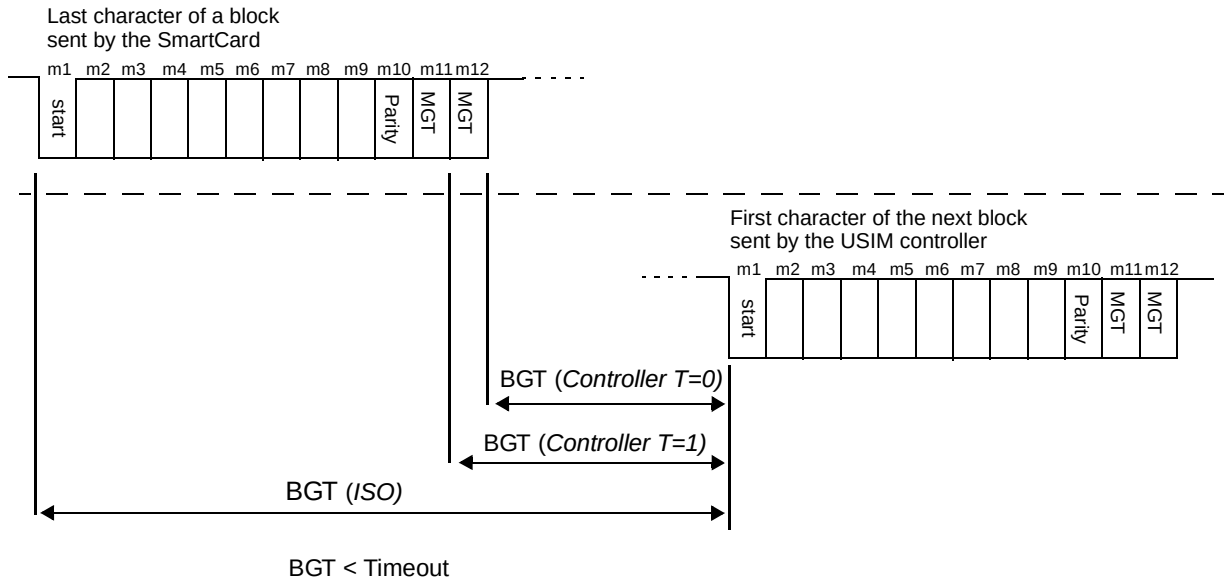
Note

The USIM controller measures the BGT differently from the way it is defined in the ISO standard, but the result is the same. Figure 75 shows the difference. Use the following formulas to convert the ISO standard BGT to BGT as used in the processor USIM controller:

Equation 6: For T=0: BGT = BGT(ISO) -12
Equation 7: For T=1: BGT = BGT(ISO) - 11

When receiving a character, the USIM controller does not initiate any transmission until the character is fully received and the Block Guard Time has elapsed (even if the TXFIFO is full). The USIM controller starts transmitting as soon as the BGT has elapsed. Transmission continues until the TXFIFO is emptied or the FCR[TX_HOLD] bit is asserted.

Figure 75: Block Guard Time, USIM Controller



For T=0: Mandatory Guard Time (MGT) requires at least 2 Guard bits
For T=1: Mandatory Guard Time (MGT) requires at least 1 Guard bit

9.4.3.2.3 Block Waiting Time (BWT)

When the SmartCard is expected to reply, it should do so no later than the Block Waiting Time (**BWT**) interval. 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 9.5.19 and Figure 76.

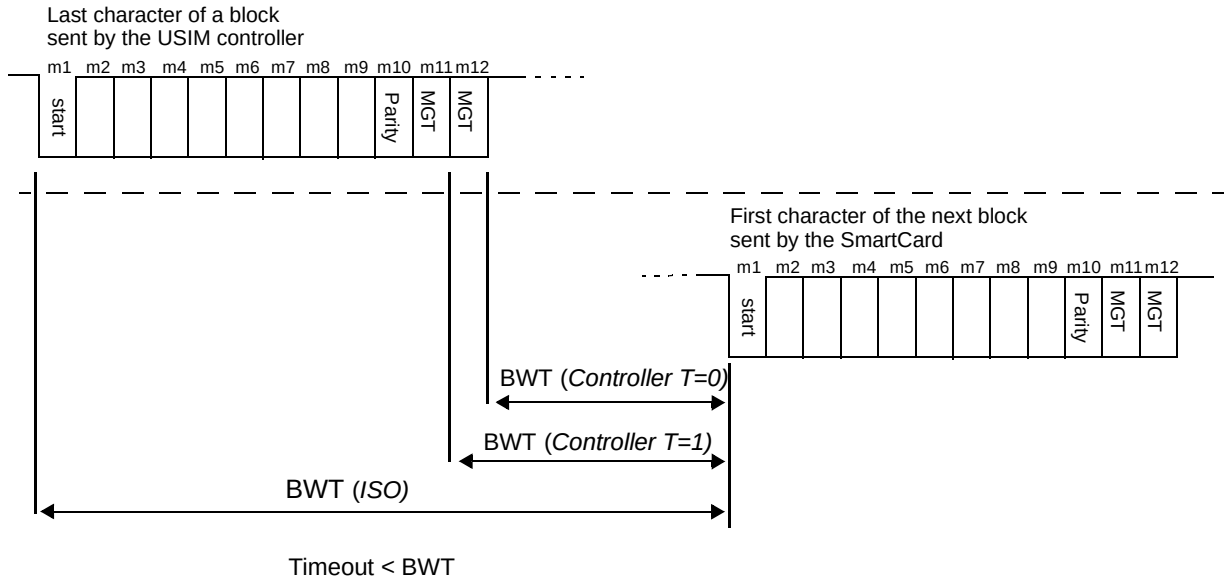
Note

The USIM controller measures the BWT differently from the way it is defined in the ISO standard, but the result is the same. Figure 76 shows the difference. Use the following formulas to convert the ISO standard BWT to BWT as used in the processor USIM controller:

Equation 8: For T=0: BWT = BWT(ISO) - 12
Equation 9: For T=1: BWT = BWT(ISO) - 11

A timeout interval can be specified by software writing to the USIM Timeout register (TOR, see Section 9.5.14). Measurement of the timeout interval begins after a SmartCard reset or after the end of a transmission. Measurement stops with the renewal of transmit or receive activity before the timeout interval has elapsed or when the timeout has elapsed. When the timeout interval expires, the LSR[BWT] bit asserts and a BWT interrupt request is generated, if enabled. BWT is useful in detecting an unresponsive SmartCard.

Figure 76: Block Waiting Time, USIM Controller



For T=0: Mandatory Guard Time (MGT) requires at least 2 Guard bits
For T=1: Mandatory Guard Time (MGT) requires at least 1 Guard bit

9.4.4 Clock Control

The USIM controller generates and controls the clock (U_CLK) that it supplies to the SmartCard based on the values in the Section 9.5.15, USIM Clock Registers (CLKRx). The frequency of the internal clock (CLK_USIM) that drives the USIM controller internal logic is 48 MHz as noted in Equation 10. U_CLK is defined in Equation 11.

Equation 10: USIM Internal Clock: $\mathrm{CLK\_USIM} = 48MHZ$

Equation 11: USIM SmartCard Clock: $\mathrm{UCLK} = \frac{\mathrm{CLK\_USIM}}{\mathrm{CLKR[DIVISOR]} \times 2}$

The *ISO 7816-3* and *3G TS 31.101* specifications limit U_CLK to a range of 1 MHz - 4 MHz. U_CLK is set to 4 MHz after a system reset. Marvell recommends that the U_CLK frequency be changed only 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 U_CLK affects the etu period. See Section 9.4.4.2. Software can write a new value to the CLKR register only when CLKR[RQST] is de-asserted. The CLKR register does not update when CLKR[RQST] is asserted. CLKR[RQST] ensures that the USIM controller has completed all pending transmissions or receptions, and the Block Guard Time before U_CLK is changed.

9.4.4.1 Clock Stop

It is possible to stop U_CLK. The voltage level of U_CLK while it is stopped can be selected via the CLKR[STOP_LEVEL] bit. Asserting the CLKR[STOP_UCLK] bit stops U_CLK only after the USIM controller completes all pending transmission, reception, and Block Guard Time. U_CLK is re-activated when de-asserting the CLKR[STOP_UCLK] bit.

Note

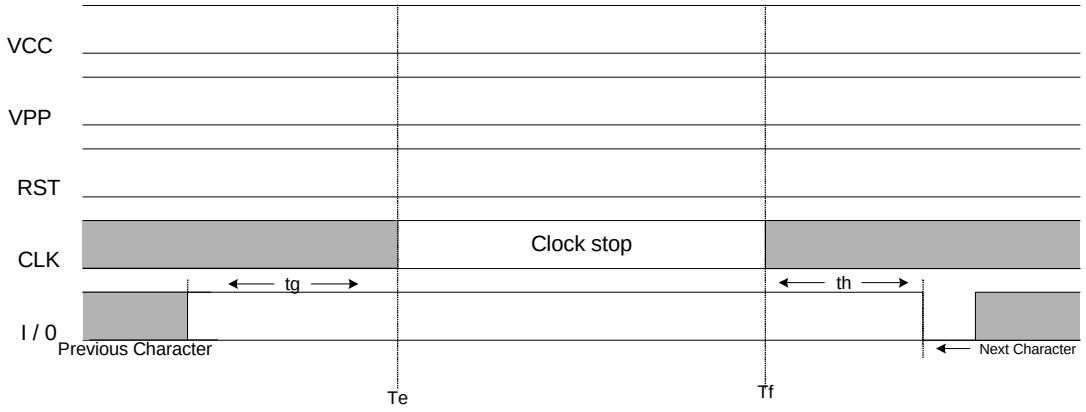
ISO 7816-3 defines Guard-Time periods before stopping the clock and after re-activating it, as specified in Figure 77.

The USIM controller internal clock, CLK_USIM, can be disabled for power savings by setting CLKR[STOP_CLK_USIM]. CLK_USIM is enabled by default after system reset.

Note

The USIM controller registers may still be read or written when CLK_USIM is disabled.

Figure 77: SmartCard Clock Stop



Note

tg and th have minimum values:
1. tg>1860 u_CLK cycles
2. th>700 u_CLK cycles
3. tg and th define a time interval while Te and Tf define a point in time.

9.4.4.2 Programmable Baud Rate Generator

The USIM controller contains a programmable baud rate generator. The same baud rate is used for transmitting and receiving data. Although the baud-rate can be configured at any given time, the actual change takes place only when: the TXFIFO is empty, no receive or transmit activities are in process, and the Block Guard Time has elapsed. The Divisor Latch register should be written when changing the baud rate even if the value is the same, see Section 9.4.4.2.2.

9.4.4.2.1 Determining the Baud Rate

Software must specify three parameters in the following order when setting a new baud rate:

1. RATIO - Ratio of the internal clock (CLK_USIM) verses the SmartCard clock (U_CLK, see Equation 12). U_CLK is defined in the Section 9.5.15, USIM Clock Registers (CLKRx).
2. FACTOR - Number of samples/bit minus one (the ISO standard specifies a minimum of six samples/bit, see Figure 78). This parameter is loaded in the Section 9.5.17, USIM Factor Latch Registers (FLRx).
3. DIVISOR - Number CLK_USIM cycles between samples (see Figure 79). This parameter is loaded in the Section 9.5.16, USIM Divisor Latch Registers (DLRx).

Figure 78: Baud Rate Sampling Pulses When Number of Samples per Bit is 6

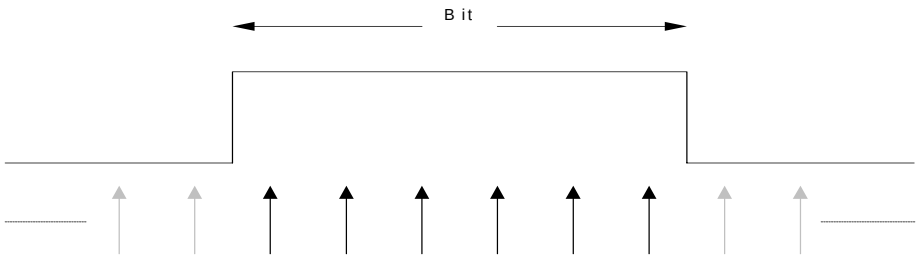
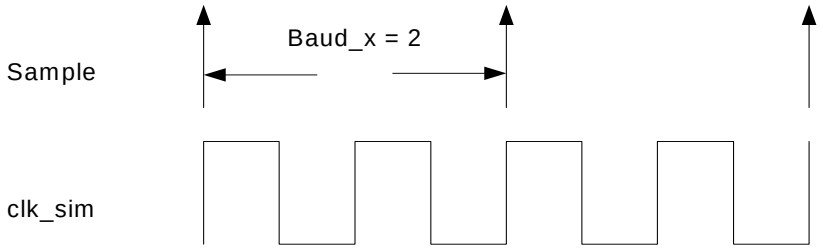


Figure 79: Spacing Between Samples When Baud Divisor is 2



The baud rate defined in Equation 14 is determined by using the above parameters with the following equations. The F and D parameters in Equation 13 are delivered by the SmartCard in the answer-to-reset (ATR) or negotiated in the parameter and protocol selection (PPS) defined in the ISO standard.

Equation 12: $$\text{RATIO} = \frac{\text{CLK\_USIM}}{UCLK \times 2}$$

Equation 13: $$\frac{DIVISOR \times (FACTOR + 1)}{RATIO \times 2} = \frac{F}{D}$$

Equation 14: $$BaudRate = \frac{\text{CLK\_USIM (in Hz)}}{(DIVISOR) \times (FACTOR + 1)} = \frac{\text{UCLK} \times RATIO \times 2}{(DIVISOR) \times (FACTOR + 1)}$$

9.4.4.2.2 Handling a Parameter and Protocol Selection (PPS)

The USIM controller is ready to start transmitting with the new protocol parameters after a successful parameter and protocol selection (PPS). The USIM controller restarts the counting of all waiting

times (BGT, CWT, EGT, and BWT) when the DLR register is written. So, software must set the waiting times before writing to the DLR register. The FLR register should also be written before the DLR register is written. To restart the baud rate, software must write to the DLR register even if the value has not changed. The USIM controller starts transmitting only after the new BGT has elapsed.

9.4.4.2.3 Examples of Setting the Baud Rate

The following two examples cover the examples where F divided by D produces an integer number and where F divided by D does not produce an integer number.

Example 2: Setting a New Baud Rate—F/D is an Integer Number

Consider the example where the F and D parameters that were delivered by the SmartCard in the TA(1) byte of the ATR are 512 and 32, respectively. The USIM controller is expected to start the session in the new baud rate specified by the SmartCard. Because F divided by D is an integer number, F/D=16, any value can be set to *RATIO* that results in a SmartCard clock that is approved by the ISO standard. For this example, set *RATIO* = 6, resulting in U_CLK = 4 MHz clock to the SmartCard. Now, software must select a *DIVISOR* and a *FACTOR* that obey Equation 13:

$$\frac{\text{DIVISOR} \times (\text{FACTOR} + 1)}{6 \times 2} = 16$$

Possible combinations are *DIVISOR* = 12 and *FACTOR* = *15 or DIVISOR* = 24 and *FACTOR* = *7.* Other combinations are allowed by the ISO standard as long as the number of samples per bit (*FACTOR)* is not less than 5.

With U_CLK = 4 MHz, the baud rate is $\frac{4MHz}{16} = 250KHz$.

Example 3: Setting a New Baud Rate—F/D is Not an Integer Number

Consider the example where F and D parameters that were delivered by the SmartCard in the TA(1) byte of the ATR are 372 and 20, respectively. Because $\frac{F}{D} = 18\frac{3}{5}$, set *RATIO = 10.*

(*RATIO = 5* is also possible, but not all SmartCards operate at a clock frequency that is greater than 4 MHz). Now, software must set *DIVISOR* and *FACTOR* to obey Equation 13:

$$\frac{\text{DIVISOR} \times (\text{FACTOR} + 1)}{10 \times 2} = \frac{372}{20}$$

A possible combination is *DIVISOR* = 31 and *FACTOR* = *11*.

The baud rate is $\frac{48MHz}{31 \times 12} = 129.032kHz$.

9.4.5 SmartCard Management

The SmartCard Control register (USCCR, see Section 9.5.10, USIM SmartCard Control Registers (USCCRx), on page 483) allows the USIM controller to:

- Specify the USIM SmartCard voltage supply
- Assert reset (U_nRST) to the SmartCard
- Specify what it places on the U_IO pin when it is transmitting to the SmartCard
- Force U_IO to V_{low}.

Software uses this register to manage startup and shutdown of the SmartCard. In particular, it is used while executing the following routines:

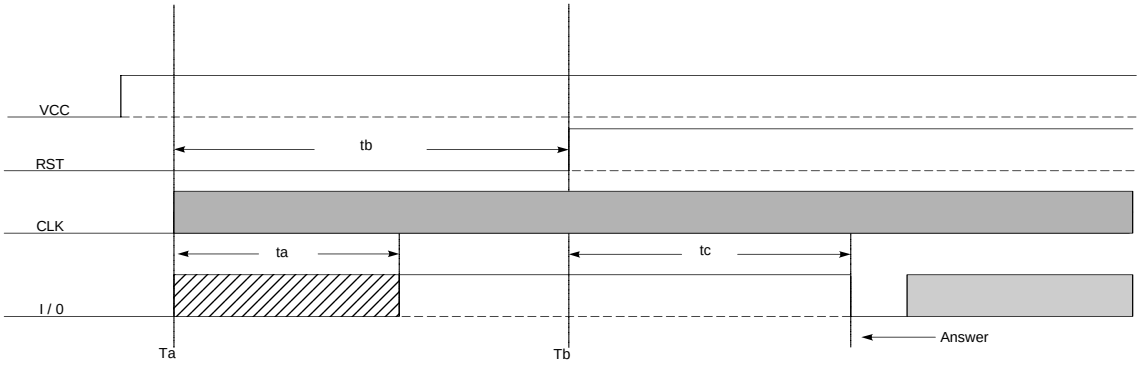
1. SmartCard activation (“cold reset”)
2. SmartCard warm reset
3. SmartCard de-activation

9.4.5.1 SmartCard Activation (“Cold Reset”)

Perform the following operations in sequence to activate the USIM SmartCard (see Figure 80):

1. Reset CLKR[STOP_CLK_USIM] (after a system reset, this bit is reset to its reset value of 0b1 by default and so this step can be skipped)
2. Clear USCCR[RST_SmartCard_N].
3. Turn on the VCC voltage by setting the USCCR[VCC] bits. Set the lowest voltage level (set USCCR[VCC] to 0b10) on first activation.
4. Enable the SmartCard I/O pin to return to Vhigh by clearing USCCR[TXD_FORCE].
5. Activate the SmartCard clock by clearing the stop bit, CLKR[STOP_UCLK].
6. Verify that USCCR[RST_SmartCard_N] was asserted for at least 400 U_CLK cycles. When U_CLK is set to the lowest frequency allowed by the ISO standard, 400 cycles require 0.4 ms($400\mu sec$). See tb in Figure 80.
7. De-assert U_nRST by setting the USCCR[RST_SmartCard_N] bit. The SmartCard answers to reset (ATR) within 400-40000 SmartCard clock cycles. See tc in Figure 80. If an ATR is not returned after 40000 SmartCard clock cycles, then repeat steps 1-7 with VCC voltage level increased from 1.8V to 3.0V in step 3 (set the USCCR[VCC] bits to 0b01).

Figure 80: SmartCard Activation



9.4.5.2 Warm Reset

Warm reset is performed to reset the SmartCard after activation. A SmartCard reset does not reset the USIM controller RXFIFO and TXFIFO. The RXFIFO and TXFIFO are reset by writing 0b1 to FCR[RESETRF] and FCR[RESETTF], respectively. Perform the following operations in sequence to execute a warm reset:

1. Clear the USCCR[RST_SmartCard_N] bit.
2. Verify that USCCR[RST_SmartCard_N] was cleared for at least 400 U_CLK cycles. When U_CLK is set to the lowest frequency allowed by the ISO standard, 400 cycles take 0.4 ms ($400\mu sec$).
3. De-assert U_nRST by setting the USCCR[RST_SmartCard_N] bit.

9.4.5.3 SmartCard De-activation

Perform the following operations in sequence to deactivate the USIM SmartCard:

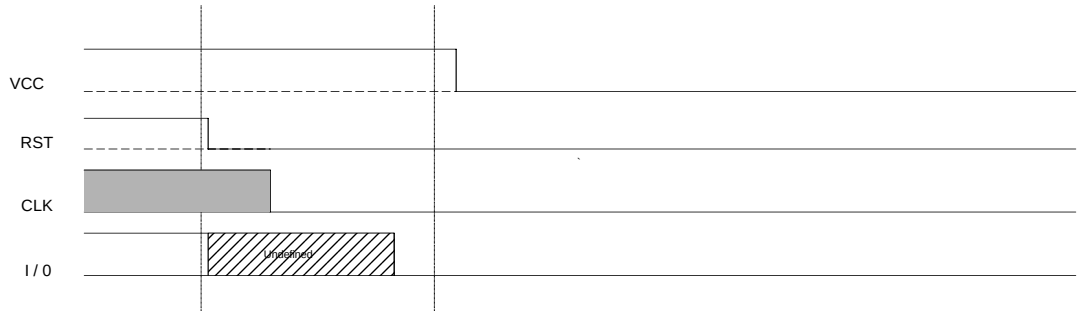
1. Clear the USCCR[RST_SmartCard_N] bit.

2. Stop the clock on Vlow by clearing CLKR[STOP_LEVEL] bit, and setting the CLKR[STOP_UCLK] bit.
3. Force the U_IO pin to ground level by setting USCCR[TXD_FORCE].
4. Turn the VCC voltage to ground level by setting USCCR[VCC] to 0b00 for at least 200 U_CLK cycles after the U_IO pin is forced low. The U_IO pin can be monitored via LSR[RXD]. An operating system timer can be used to determine the amount of time that elapsed since LSR[RXD] went low.
5. Set CLKR[STOP_CLK_USIM] for power savings (Section 9.5.15, USIM Clock Registers (CLKRx)).

Note

When set, CLKR[RQST] indicates that a SmartCard clock change is occurring. Any write to the CLKR register is ignored while CLKR[RQST] is set.

Figure 81: SmartCard De-activation



9.4.6 FIFO Operation

The USIM controller has two 16-byte deep FIFOs: an RXFIFO and an TXFIFO. Each FIFO is accessed via the USIM controller RBR and THR register, respectively. The USIM FIFO Control register (refer to Table 215 on page 479) allows software to:

- Reset the FIFOs
- Hold transmission
- Mask parity errors in the RXFIFO
- Set the trigger levels for each FIFO.

The FIFOs can be serviced via interrupt-request mode, polling mode, or DMA mode.

9.4.6.1 Interrupt Request Mode

Software tracks and handles FIFO activities via interrupt requests in interrupt request mode. The interrupt requests are enabled via the Section 9.5.4, USIM Interrupt Enable Registers (IERx). The source that caused the interrupt request can be determined by reading the Section 9.5.5, USIM Interrupt Identification Registers (IIRx).

9.4.6.1.1 Receiver Related Interrupts

During data reception, LSR[RX_WORKING] is set and the following can occur when enabled:

- Receiver Data Ready Interrupt - This interrupt indicates the number of bytes in the RXFIFO is equal to or greater than the trigger level specified in the FCR[RX_TL] field. The interrupt request is cleared when the number of bytes in the RXFIFO drops below the FCR[RX_TL] value. Empty the RXFIFO by reading the RBR register.
- Character Waiting Time Interrupt - This interrupt is an ISO protocol parameter that indicates that the maximum delay between the leading edges of two consecutive characters has been violated. See Section 9.4.3.2.1 and the ISO standard for full details. The interrupt request is cleared by setting IIR[CWT].
- Receiver Time Out Interrupt - This interrupt helps software detect when trailing bytes are left in the RXFIFO for some time with no additional RX activity. It occurs when at least one character remains in the RXFIFO and the Timeout period programmed into the Section 9.5.14, USIM Timeout Registers (TORx), has expired. The interrupt request is cleared by:
  - Reading the RBR register
  - Resetting the RXFIFO
  - Receiving another character before the timeout interval has elapsed.
- Framing Error Interrupt - This interrupt occurs when the U_IO pin is asserted low during guard time. See Section 9.4.3.1.4 for full details. The interrupt request is cleared by writing an 0b1 to the IIR[FRAMERR] bit.
- Parity Error Interrupt in T=0 mode - This interrupt occurs when the received data contains a parity error. See Section 9.4.3.1.2 for full details. The interrupt request is cleared by writing an 0b1 to the IIR[PERR] bit.
- Parity Error Interrupt in T=1 mode - This interrupt occurs when the received data contains a parity error. See Section 9.4.3.1.2 for full details. The interrupt request is cleared by writing an 0b1 to the IIR[PERR] bit.
- Receiver Data Overrun Interrupt - This interrupt occurs when the RXFIFO is full and cannot accept any more data. See Section for full details. The interrupt request is cleared by writing an 0b1 to the IIR[OVRN] bit.

9.4.6.1.2 Transmitter Related Interrupts

During transmission, the LSR[TX_WORKING] bit is set and the following can occur when enabled:

- Transmitter Data Refill Interrupt - This interrupt indicates that the number of bytes in the TXFIFO is less than the trigger level specified in the FCR[TX_TL] field. The interrupt request is cleared when the number of bytes in the TXFIFO is increased to a point where the TXFIFO is at or above its trigger level.
- Block Waiting Time Interrupt - This interrupt is an ISO protocol parameter that indicates that the maximum delay between the leading edge of the last character of a block sent by the USIM controller and the leading edge of the first character of the next block sent by the SmartCard has been violated. See Section 9.4.3.2.3 and the ISO standard for full details. The interrupt request is cleared by setting IIR[BWT].
- T=0 Error Interrupt - In T=0 mode, this interrupt indicates that the SmartCard has signaled the USIM controller transmitter to repeat the transmission. See Section 9.4.3.1.1 for full details. The interrupt request is cleared via the T0_REPEAT or T0_CLR bits in the Error Control register.

9.4.6.1.3 Other Interrupts

- SmartCard Detection Interrupt - This interrupt indicates that a SmartCard has been inserted into its connector. The interrupt remains asserted until the it is cleared. The interrupt only asserts upon detection of a new SmartCard. The interrupt request is cleared by writing an 0b1 to the IIR[SmartCard_DET] bit.

9.4.6.2 Polled Mode

Interrupt and DMA requests are disabled in polled mode. FIFOs are accessed by software reading and writing the RBR and THR registers, respectively. Software checks the receiver and transmitter status by reading the LSR and FSR registers. Section 9.5.7, USIM FIFO Status Registers (FSRx), holds information regarding FIFO status. Section 9.5.12, USIM Extra Guard Time Registers (EGTRx), contains status information on data transfers. Since the receiver and the transmitter are controlled separately, either one or both can operate in polled mode.

9.4.6.3 DMA Request Mode

Data is entered and removed from the FIFOs via DMA requests in DMA request mode. DMA requests are enabled in Section 9.5.4, USIM Interrupt Enable Registers (IERx). The USIM controller has two DMA requests: one for TXFIFO data service, and one for RXFIFO data service.

Note

The FIFO trigger levels in the 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.

9.4.6.3.1 Transmit Data Service

The DMA transmit request is generated when the DMA_TX request is enabled in the IER register and the number of bytes in the TXFIFO is below eight. The DMA controller then writes data to the TXFIFO. The DMA controller must be configured to 8 byte burst size, with data width of 1 byte. Each write by the DMA to the THR register adds a single byte to the TXFIFO. The number of bytes to be transmitted is programmed in the DMA controller.

9.4.6.3.2 Receive Data Service

The DMA receive request is generated in two situations:

- When the DMA_RX request is enabled in the IER and the number of bytes in the RXFIFO is equal to or greater than eight
- A time out situation occurs when:
  - The DMA_TIME bit is enabled in the IER and
  - The RXFIFO is not empty but the trigger level was not reached and
  - The RBR was not accessed before the Timeout period (defined in the Section 9.5.14, USIM Timeout Registers (TORx)) expired.

A DMA receive request is generated to remove trailing bytes from the RXFIFO.

The DMA controller reads data from the RXFIFO when one of the above situations occurs. The DMA controller can read up to 8 bytes of data for each DMA request. The number of bytes to be read is programmed in the DMA controller.

Software ensures that any remaining data in the RXFIFO is handled properly when the DMA reaches the end of its chain. This data handling should not be an issue since software detects how much data is being received via the ISO 7816-3 protocol. The USIM controller indicates an end-of-receive (EOR) status whenever the TOR or CWTR timeouts have expired and the DMA is reading the last byte in the RXFIFO.

Software should set up the DMA Channel Control/Status Register, DCSR, as noted in Table 209, 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).

Table 209: 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 the figure Table 22-18, “DCSR0–31 Bit Definitions” on page 22-46 of the *PXA3xx Processor Family Vol. I: System and Timer Configuration Developers Manual* for more information.

Do not use DMA for transferring trailing bytes from the receive FIFO. Always use PIO to transfer the trailing bytes. IER[DMA_TIME] must be cleared to disable the generation of EOR for the DMA.

Special Condition:

Parity errors enter the RXFIFO in T=1 DMA mode. A DMA receive request results in reading only the first 8 bits from the RBR; the parity error bit is not read. Software follows one of two possible approaches shown below to determine if a block contains a parity error in this mode. Each approach requires the DMA interrupt to be asserted. Refer to “Servicing Internal Peripherals” in the DMA chapter (*PXA3xx Processor Family Vol. I: System and Timer Configuration Developers Manual*) for more details on the DMA interrupt.

- The DMA reads the whole block regardless of parity errors with the parity error interrupt disabled. After the block is read, the DMA interrupt occurs and software checks LSR[PERR] bit to determine if the block contains any parity errors.
- When a parity error occurs, the parity error software interrupt service routine waits for the DMA interrupt. Software then requests retransmission of the block upon receiving the DMA interrupt, indicating that the whole block has been read.

9.5 Register Descriptions

The following registers specify the control, status, and data functionality of the USIM controller. The addresses of these registers are shown as offsets to the base address for simplicity in describing the two separate USIM controllers.

9.5.1 Register Summary

There are seventeen 32-bit-wide registers in the USIM controller. Table 210 shows the registers and their addresses for each of the two instances of the USIM controller.

Table 210: USIM Controller Register Summary

Physical Address	Description	Page
0x4160_0000	**USIM Controller 1**	
0x4160_0000	USIM Receive Buffer Registers (RBRx)	page 473
0x4160_0004	USIM Transmit Holding Registers (THRx)	page 474

Table 210: USIM Controller Register Summary (Continued)

Physical Address	Description	Page
0x4160_0008	USIM Interrupt Enable Registers (IERx)	page 474
0x4160_000C	USIM Interrupt Identification Registers (IIRx)	page 476
0x4160_0010	USIM FIFO Control Registers (FCRx)	page 478
0x4160_0014	USIM FIFO Status Registers (FSRx)	page 480
0x4160_0018	USIM Error Control Registers (ECRx)	page 480
0x4160_001C	USIM Line Control Registers (LCRx)	page 481
0x4160_0020	USIM SmartCard Control Registers (USCCRx)	page 483
0x4160_0024	USIM Line Status Registers (LSRx)	page 484
0x4160_0028	USIM Extra Guard Time Registers (EGTRx)	page 486
0x4160_002C	USIM Block Guard Time Registers (BGTRx)	page 487
0x4160_0030	USIM Timeout Registers (TORx)	page 488
0x4160_0034	USIM Clock Registers (CLKRx)	page 488
0x4160_0038	USIM Divisor Latch Registers (DLRx)	page 490
0x4160_003C	USIM Factor Latch Registers (FLRx)	page 491
0x4160_0040	USIM Character Waiting Time Registers (CWTRx)	page 491
0x4160_0044	USIM Block Waiting Time Registers (BWTRx)	page 492
0x4160_0048–0x4160_FFFC	reserved	reserved
0x4210_0000	**USIM Controller 2**	
0x4210_0000	USIM Receive Buffer Registers (RBRx)	page 473
0x4210_0004	USIM Transmit Holding Registers (THRx)	page 474
0x4210_0008	USIM Interrupt Enable Registers (IERx)	page 474
0x4210_000C	USIM Interrupt Identification Registers (IIRx)	page 476
0x4210_0010	USIM FIFO Control Registers (FCRx)	page 478
0x4210_0014	USIM FIFO Status Registers (FSRx)	page 480
0x4210_0018	USIM Error Control Registers (ECRx)	page 480
0x4210_001C	USIM Line Control Registers (LCRx)	page 481

Table 210: USIM Controller Register Summary (Continued)

Physical Address	Description	Page
0x4210_0020	USIM SmartCard Control Registers (USCCRx)	page 483
0x4210_0024	USIM Line Status Registers (LSRx)	page 484
0x4210_0028	USIM Extra Guard Time Registers (EGTRx)	page 486
0x4210_002C	USIM Block Guard Time Registers (BGTRx)	page 487
0x4210_0030	USIM Timeout Registers (TORx)	page 488
0x4210_0034	USIM Clock Registers (CLKRx)	page 488
0x4210_0038	USIM Divisor Latch Registers (DLRx)	page 490
0x4210_003C	USIM Factor Latch Registers (FLRx)	page 491
0x4210_0040	USIM Character Waiting Time Registers (CWTRx)	page 491
0x4210_0044	USIM Block Waiting Time Registers (BWTRx)	page 492
0x4210_0048–0x4210_FFFC	reserved	reserved

9.5.2 USIM Receive Buffer Registers (RBRx)

The RBR contains the next byte to be read from the RXFIFO (see Table 211). In T=1 mode, a byte containing parity errors have bit 8 asserted. In T=0 mode, parity errors do not enter the RXFIFO.

The RBR register is a read-only register.

Table 211: RBRx Bit Definitions

Physical Address
0x4160_0000
0x4210_0000

RBR0
RBR1

USIM 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																							PERR	RB[7:0]							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:9	—	—	reserved
8	R	PERR	Parity Error: 0 = Either there was no parity error when receiving the character or the parity bit was masked by the PEM bit in the FCR register 1 = Character was received with a parity error

Table 211: RBRx Bit Definitions (Continued)

Physical Address 0x4160_0000 0x4210_0000	RBR0 RBR1	USIM 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																							PERR	RB[7:0]							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
7:0	R	RB[7:0]	Data byte received.

9.5.3 USIM Transmit Holding Registers (THRx)

The THR contains the next byte to be written to the TXFIFO. Data bytes written to this register are stored in the TXFIFO and are transmitted to the SmartCard via the U_IO pin in a first-in-first-out manner. After writing at least one byte to the TXFIFO via the THR register, the USIM controller initiates transmission on the U_IO pin as soon as the block-guard time period is over. Software must keep track of whose turn it is to transmit on the I/O line: USIM controller transmitting on it U_IO pin or the SmartCard transmitting on it I/O pin.

THR is a write-only register. Write 0b0s to reserved bits.

Table 212: THRx Bit Definitions

Physical Address 0x4160_0004 0x4210_0004	THR0 THR1	USIM 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																								TB[7:0]							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	W	TB[7:0]	Data byte to be transmitted, least significant bit first

9.5.4 USIM Interrupt Enable Registers (IERx)

The IER enables USIM controller interrupt requests and DMA requests. Each interrupt request is explained in the interrupt request mode section (see Section 9.4.6.1). Each DMA request is explained in the DMA request mode section (see Section 9.4.6.3).

Warning

Each type of interrupt request or DMA request can be enabled or disabled individually using the IER. The receiver-timeout interrupt request is separated from the receiver data-ready interrupt request to avoid having the CPU and the DMA controller simultaneously serve the RXFIFO.

The IER register is a read-write register. Write 0b0s to reserved bits; read from reserved bits are not defined.

Table 213: IERx Bit Definitions

Physical Address 0x4160_0008 0x4210_0008																IER0 IER1												USIM 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																DMA_TX	DMA_RX	DMA_TIME	reserved		SmartCard_DET	TDR	RDR	reserved	BWT	CWT	TiMEO	FRAMERR	T0ERR	PERR	OVRN
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	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 Situation: 0 = Disabled 1 = Enabled
12:11	—	—	reserved
10	R/W	SmartCard_DET	SmartCard 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

Table 213: IERx Bit Definitions (Continued)

Physical Address 0x4160_0008 0x4210_0008	IER0 IER1	USIM 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																**DMA_TX**	**DMA_RX**	**DMA_TIME**	**reserved**		**SmartCard_DET**	**TDR**	**RDR**	**reserved**	**BWT**	**CWT**	**TiMEO**	**FRAMERR**	**T0ERR**	**PERR**	**OVRN**
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	?	?	0	0	0	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
7	—	—	reserved
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	T0ERR	T=0 Error Interrupt Enable: 0 = Disabled 1 = Enabled
1	R/W	PERR	Parity Error Interrupt Enable: 0 = Disabled 1 = Enabled
0	R/W	OVRN	Receiver Data Overrun Interrupt Enable: 0 = Disabled 1 = Enabled

9.5.5 USIM Interrupt Identification Registers (IIRx)

The IIR specifies the source of each interrupt request. Each interrupt request is explained in the interrupt mode section (see Section 9.4.6.1). After software handles the interrupt source, it clears the interrupt request by writing an 0b1 to the corresponding IIR bit. There are several exceptions to this rule:

- The receiver data ready (RDR) interrupt request is cleared only when the number of bytes in the RXFIFO drops below a programmable trigger level as specified by the RX_TL bits of the Section 9.5.6, USIM FIFO Control Registers (FCRx), on page 478. The RXFIFO is unloaded by reading data from the RBR or by clearing it.

- The receiver-timeout (TIMEO) interrupt request is cleared when reading the RBR or clearing the RXFIFO.
- The transmitter data-refill (TDR) interrupt request is cleared automatically when the number of bytes in the TXFIFO is at or above a programmable trigger level as specified by the TX_TL bits of the Section 9.5.6, USIM FIFO Control Registers (FCRx), on page 478. Data should be written to the THR to fill the TXFIFO.

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 Registers (ECRx)

Table 214: IIRx Bit Definitions

Physical Address
0x4160_000C
0x4210_000C

IIR0
IIR1

USIM 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																					SmartCard_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[1]	SmartCard_DET	SmartCard Detection: 0 = No interrupt 1 = Interrupt has occurred
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
6	R/W[1]	BWT	Block Waiting Time Interrupt: 0 = No interrupt 1 = Interrupt has occurred
5	R/W[1]	CWT	Character Waiting Time Interrupt: 0 = No interrupt 1 = Interrupt has occurred
4	R	TIMEO	Receiver Time Out Interrupt: 0 = No interrupt 1 = Interrupt has occurred

Table 214: IIRx Bit Definitions (Continued)

	Physical Address 0x4160_000C 0x4210_000C												IIR0 IIR1										USIM 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																					SmartCard_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
3	R/W[1]	FRAMERR	Framing Error Interrupt: 0 = No interrupt 1 = Interrupt has occurred
2	R	T0ERR	T=0 Error Interrupt: 0 = no interrupt 1 = interrupt has occurred
1	R/W[1]	PERR	Parity Error Interrupt: 0 = No interrupt 1 = Interrupt has occurred
0	R/W[1]	OVRN	Receiver Data Overrun Interrupt: 0 = No interrupt 1 = Interrupt has occurred

[1] Write a 0b1 to clear this bit.

9.5.6 USIM FIFO Control Registers (FCRx)

The FCR allows:

- The TXFIFO and/or the RXFIFO to be reset
- Transmission to be temporarily stopped
- Parity error masking to occur in T=1 mode.

The trigger thresholds for the TXFIFO and RXFIFO must each be set for eight bytes to avoid unpredictable results when in DMA mode.

The FCR register is a write-only register. Reserved bits must be written with 0b0s.

Table 215: FCRx Bit Definitions

Physical Address
0x4160_0010
0x4210_0010

FCR0
FCR1

USIM 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																							**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	W	TX_TL	Transmitter Interrupt Trigger Level (Threshold): Sets interrupt/DMA request trigger threshold for the TXFIFO: 0= Trigger threshold is 0 byte 1= Trigger threshold is 8 bytes
7:6	W	RX_TL	Receiver Interrupt Trigger Level (Threshold): Set interrupt/DMA request trigger threshold for the RXFIFO: 00= Trigger threshold is 1 byte 01= Trigger threshold is 4 bytes 10= Trigger threshold is 8 bytes 11= Trigger threshold is 12 bytes
5:4	—	—	reserved
3	W	PEM	Parity Error Mask: 0 = In T=1 mode only, bit 8 of the RBR is asserted when a character with a parity error enters the RXFIFO 1 = A parity error coming out of the RXFIFO is masked and does not appear on bit 8 of the RBR. Evidence of a parity error can still be found in: • Line Status register, LSR[PERR] • FIFO Status register, FSR[PERR_NUM] • Interrupt Identification register, IIR[PERR] (if the parity error interrupt is enabled)
2	W	TX_HOLD	Transmit Hold: When this bit is 0b1, transmission stops after the current character has been transmitted. All the remaining characters in the TXFIFO are held. Transmission resumes when this bit is cleared to 0b0. 0 = Transmission resumes 1 = Transmission is stopped.
1	W	RESETTF	Reset TXFIFO: 0 = No effect 1 = TXFIFO is cleared
0	W	RESETRF	Reset RXFIFO: 0 = No effect 1 = RXFIFO is cleared

9.5.7 USIM FIFO Status Registers (FSRx)

The FSR provides software with the number of bytes in each of the FIFOs and the number of parity errors in the RXFIFO. This register is used when working in Polled mode (see page 470).

The FSR register is a read-only register. Reads from reserved bits are not defined.

Table 216: FSRx Bit Definitions

Physical Address 0x4160_0014 0x4210_0014																FSR0 FSR1										USIM 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																	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: Indicates the number of characters with a parity error in the RXFIFO.
9:5	R	TX_LENGTH	TXFIFO Length: Indicates the number of characters in the TXFIFO.
4:0	R	RX_LENGTH	RXFIFO Length: Indicates the number of characters in the RXFIFO.

9.5.8 USIM Error Control Registers (ECRx)

The ECR allows software to determine the USIM controller behavior in parity-error situations and T=0 error situations when working in T=0 mode. The ECR register is a read-write register. Write 0b0s to reserved bits; reads from reserved bits are not defined.

Table 217: ECRx Bit Definitions

Physical Address 0x4160_0018 0x4210_0018																ECR0 ECR1										USIM 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																								T0_REPEAT	T0_CLR	reserved	PE-TL		reserved	T0ERR_TL	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	?	1	1

Bits	Access	Name	Description
31:8	—	—	reserved

Table 217: ECRx Bit Definitions (Continued)

Physical Address 0x4160_0018 0x4210_0018	ECR0 ECR1	USIM 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																								**T0_REPEAT**	**T0_CLR**	**reserved**	**PE-TL**		**reserved**	**T0ERR_TL**	
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	?	0	0	?	1	1

Bits	Access	Name	Description
7	R/W[1]	T0_REPEAT	Repeat Character Transmission: Repeat character transmission until error trigger threshold (see bellow) is met again. 0 = Character has been repeated 1 = Character repeat in process
6	R/W[1]	T0_CLR	Clear T=0 Error: 0 = T=0 error has been cleared 1 = Clearing of T=0 error in process
5	—	—	reserved
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 errors causes an interrupt to occur, if enabled 0b10 = Reception of three consecutive parity errors causes an interrupt to occur, if enabled 0b11 = Reception of four consecutive parity errors causes an interrupt to occur, if enabled
2	—	—	reserved
1:0	R/W	T0ERR_TL	T=0 Error Trigger Level (Threshold) 00 = Reception of single T=0 error causes interrupt request and transmission hold. 01= Reception of two consecutive T=0 errors causes interrupt request and transmission hold. 10= Reception of three consecutive T=0 errors causes interrupt request and transmission hold. 11= Reception of a four consecutive T=0 errors causes interrupt request and transmission hold.

[1] Write an 0b1 to clear this bit.

9.5.9 USIM Line Control Registers (LCRx)

The LCR allows software to define protocol and coding conventions for the USIM controller transmitter and receiver. The serial data format consists of a Start bit (logic 0), eight data bits that may be appear in different order and polarity, and a closing even parity bit. When the T=0 protocol is selected, error signaling may appear over the line through Guard time.

LCR[ORDER] and LCR[INVERSE] determine the data bit order and its polarity. After receiving the first byte of the ATR, decide whether the data order should be mirrored and whether data polarity should change. LCR[ORDER] and LCR[INVERSE] can be changed any time after the USIM controller receives the first byte. There is no need to reset the SmartCard if its transmission was not in the order expected by the USIM controller.

LCR[EPS] determines whether the parity bit is even. Set this bit when no activity is expected on the U_IO pin.

Protocol switch policy: Protocol switching can occur at three times:

- Through ATR - After receiving byte using T=0 protocol, change the protocol to T=1 if the SmartCard supports this protocol. The advantage of using the T=0 protocol is a SmartCard reset is not needed when a single-parity error occurs.
- After the SmartCard finishes the ATR, the USIM controller is expecting data reception using the protocol defined by T=0 and TA2 (if it exists).
- After PPS - the USIM controller may initiate a protocol and parameter selection session. If the session ended successfully, the USIM controller protocol may be changed.

Marvell recommends that any change in protocol be avoided under other scenarios.

The LCR register is a read-write register. Write 0b0s to reserved bits; reads from reserved bits are not defined.

Table 218: LCRx Bit Definitions

Physical Address
0x4160_001C
0x4210_001C

LCR0
LCR1

USIM 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																											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 0b0, an odd number of 0b1s is transmitted or checked in the data word bits and the parity bit. When EPS is 0b1, an even number of 0b1s 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 9.4.2.

Table 218: LCRx Bit Definitions (Continued)

	Physical Address 0x4160_001C 0x4210_001C LCR0 LCR1 USIM 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_T1	RX_T1	EPS	ORDER	INVERSE
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	1	0	0

Bits	Access	Name	Description
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 9.4.2. 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 9.4.2.
0	R/W	INVERSE	Bit inversion: 0 = 0b0 transmitted/received as $V_{low,}$ 0b1 transmitted/received as V_{high}. Used in Direct Convention. See Section 9.4.2. 1 = 0b1 transmitted/received as $V_{low,}$ 0b0 transmitted/received as V_{high}. Used in Inverse Convention. See Section 9.4.2.

9.5.10 USIM SmartCard Control Registers (USCCRx)

The USCCR is used when dealing with Section 9.4.5, SmartCard Management.

The USCCR register is a read-write register. Write 0b0s to reserved bits; reads from reserved bits are not defined.

Table 219: USCCRx Bit Definitions

Physical Address 0x4160_0020 0x4210_0020 / USCCR0 USCCR1 / USIM 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																											TXD_FORCE	reserved	VCC		RST_SmartCard_N
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	?	0	0	0

Bits	Access	Name	Description
31:5	reserved	—	reserved
4	R/W	TXD_FORCE	Force TXD: This bit always remains 0b0 unless running a de-activation process. During de-activation, U_IO pin (TXD) must be forced to V_{low} before selecting SmartCard voltage supply pins to be driven with a lower voltage. 0 = TXD indicates transmission 1 = TXD is forced to V_{low}
3	—		reserved
2:1	R/W	VCC	SmartCard Voltage: 00 = USIM SmartCard at 0V (GND) 01= USIM SmartCard at 3V 10 = USIM SmartCard at 1.8V 11 = Reserved
0	R/W	RST_SmartCard_N	SmartCard Reset: 0 = Reset the USIM SmartCard 1 = Normal USIM SmartCard operation

9.5.11 USIM Line Status Registers (LSRx)

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 9.4.6. The LSR bits that do not have a corresponding interrupt are: RXD, RX_WORKING, TX_WORKING, RX_EMPTY_N, and 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 sources are cleared. The T=0 error is an exception; it is cleared only when ECR[T0_REPEAT] or ECR[T0_CLR] is set to 0b1.

An interrupt request associated with a Waiting Time or an Error Status bit is not cleared when the corresponding status bit is read from the LSR. The interrupt request is only cleared by writing an 0b1 to the corresponding bit in the Section 9.5.5, USIM Interrupt Identification Registers (IIRx)

The LSR register is a read-only register. Reads from reserved bits are not defined.

Table 220: LSRx Bit Definitions

Physical Address 0x4160_0024 0x4210_0024	LSR0 LSR1	USIM 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																**RXD**	**RX_WORKING**	**TX_WORKING**	**RX_EMPTY_N**	**TDR**	**reserved**				**BWT**	**CWT**	**TIMEO**	**FRAMERR**	**T0ERR**	**PERR**	**OVRN**
Reset	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**0**	**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 U_IO Pin: 0 = U_IO pin is currently at logic 0b0 1 = U_IO pin is currently on logic 0b1
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	RXFIFO Not Empty: 0 = RXFIFO is empty 1 = RXFIFO is not empty
11	R	TDR	Transmitter Data Refill: 0 = Number of characters in the TXFIFO is at or above FCR[TX_TL] 1 = Number of characters in the TXFIFO is less than FCR[TX_TL]
10:7	—	—	reserved
6	R	BWT	**Block Waiting Time:** 0 = BWT Timeout period is not over 1 = BWT Timeout period is over
5	R	CWT	Character Waiting Time: 0 = CWT Timeout period is not over 1 = CWT Timeout period is over
4	R	TIMEO	Receiver Time Out: 0 = TOR Timeout period is not over 1 = TOR Timeout period is over
3	R	FRAMERR	Framing Error: 0 = Framing error has not occurred 1 = Framing error has occurred

Table 220: LSRx Bit Definitions (Continued)

	Physical Address 0x4160_0024 0x4210_0024 LSR0 LSR1 USIM 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																RXD	RX_WORKING	TX_WORKING	RX_EMPTY_N	TDR	reserved				BWT	CWT	TIMEO	FRAMERR	T0ERR	PERR	OVRN
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	1	?	?	?	?	0	0	0	0	0	0	0

Bits	Access	Name	Description
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

9.5.12 USIM Extra Guard Time Registers (EGTRx)

The EGTR specifies the number of extra Guard-Time moments (EGTM) between bytes transmitted by the USIM controller. This register can hold any value between 0-255. In the T=0 protocol, the USIM controller 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. See Table 221.

The EGTR register is a read-write register. Write 0b0s to reserved bits; reads from reserved bits are not defined.

Table 221: Number of ETU Between a Transmitted Byte Leading Edge in Different Protocols

Protocol	Spacing Between a Transmitted Byte Leading Edge (in etu)
T=0	12 + EGTM
T=1	11 + EGTM

Table 222: EGTR Bit Definitions

	Physical address 0x4160_0028 0x4210_0028 EGTR0 EGTR1 USIM 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																								**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 Extra Guard Time moments, 0-255.

9.5.13 USIM Block Guard Time Registers (BGTRx)

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) during which the USIM controller transmitter cannot access the U_IO pin after the SmartCard 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, but the overall effect is the same. See Table 223.

The BGTR register is a read-write register. Write 0b0s to reserved bits; reads from reserved bits are not defined.

Table 223: Block Guard Time Specified in 3G TS 31.101

	T=0	T=1
Value as specified in protocol BGT definition	16	22
Value converted to BGTR definition	4	11

Table 224: BGTRx Bit Definitions

Physical address 0x4160_002C 0x4210_002C													BGTR0 BGTR1											USIM 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																								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.

9.5.14 USIM Timeout Registers (TORx)

The TOR allows software to detect when data is left in the RXFIFO for some time with no Receive activity on the U_IO pin. See Section 9.4.6 for more information. Timeout is measured from the number of moments that have passed since:

- The last byte entered the RXFIFO or
- Since the RXFIFO was read.

The timeout count stops when the RXIFO is empty or when receive activity resumes on the U_IO pin. The TOR specifies the trigger threshold for the timeout interrupt.

The TOR register is a read-write register. Write 0b0s to reserved bits; reads from reserved bits are not defined.

Table 225: TORx Bit Definitions

Physical address 0x4160_0030 0x4210_0030													TOR0 TOR1											USIM 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																								TO							
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	1	0	0	0	1	0

Bits	Access	Name	Description
31:8	—	—	reserved
7:0	R/W	TO	Time Out: Number of total timeout moments.

9.5.15 USIM Clock Registers (CLKRx)

The CLKR allows software to select values that specify the SmartCard clock, U_CLK, using the methodology described in Section 9.4.4. The CLKR can not be updated while CLKR[RQST] is set. Any writes to the CLKR when CLKR[RQST] is set are ignored.The SmartCard clock is disabled and

its frequency is set to 4 MHz by default after reset. The USIM Controller internal clock, CLK_USIM, can be disabled for power savings by asserting bit CLKR[STOP_CLK_USIM]. CLK_USIM is disabled by default after system reset.

The USIM controller registers may be read/written when CLK_USIM is disabled.

The CLKR register is a read-write register. Write 0b0s to reserved bits; reads from reserved bits are not defined.

Table 226: CLKRx Bit Definitions

Physical Address 0x4160_0034 0x4210_0034	CLKR0 CLKR1	USIM 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																**STOP_CLK_USIM**	**STOP_LEVEL**	**STOP_UCLK**	**RQST**	**reserved**				**DIVISOR[7:0]**							
Reset	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**?**	**?**	**?**	**?**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**

Bits	Access	Name	Description
31:16	—	—	reserved
15	R/W	STOP_CLK_ USIM	Stop USIM Controller's Clock: 0 = Starts the USIM controller internal clock, CLK_USIM 1 = Stops the USIM controller internal clock. Does not stop the clock that controls the USIM controller registers.
14	R/W	STOP_LEVEL	Stop Level: 0 = U_CLK stops on low 1 = U_CLK stops on high
13	R/W	STOP_UCLK	Stop Clock: 0 = Starts the U_CLK clock 1 = Stops the U_CLK clock
12	R	RQST	Clock Change Request 0 = Request not occurring, CLKR may be updated 1 = Request occurring, do not update CLKR
11:8	—	—	reserved
7:0	R/W	DIVISOR	Clock Divisor: The U_CLK frequency is 48 MHz/DIVISOR. A value of 0b0s results in unpredictable behavior.

Note

Changing U_CLK affects the ETU period. See Section 9.4.4.2. Software can enter a new value to CLKR at any given time. However, the new value is activated only when the USIM controller completes all previous pending activities (that is, transmission/ reception, Block Guard Time, and extra Guard-Time periods are finished and the RXFIFO is empty).

Marvell recommends that the U_CLK frequency be 4 MHz. However, it is possible to use other frequencies. In such cases, change the value of the CLKR register and follow the values of the Programmable Baud Rate Generator registers.

9.5.16 USIM Divisor Latch Registers (DLRx)

The DLR contains the divisor used in determining the baud rate (see Section 9.4.4.2). The reset value conforms to the *ISO 7816-3* requirement that during the ATR the etu is 372 clock cycles.

The DLR register is a read-write register. Write 0b0s to reserved bits; reads from reserved bits are not defined.

Table 227: DLRx Bit Definitions

Physical Address 0x4160_0038 0x4210_0038																DLR0 DLR1															USIM 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																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
15:0	R/W	DIVISOR	Baud Divisor: Determines the number of internal USIM cycles 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 0b0s.

Note

Change in the actual baud-rate is not immediate. Refer to Section 9.4.4.2.1, Determining the Baud Rate, on page 465 for a description of the conditions when the baud rate change occurs.

9.5.17 USIM Factor Latch Registers (FLRx)

The FLR contains the factor used in determining the baud rate (see Section 9.4.4.2). The reset value conforms to the *ISO 7816-3* requirement that the etu is 372 clock cycles during the ATR. The FLR must be modified before the DLR.

The FLR register is a read-write register. Write 0b0s to reserved bits; reads from reserved bits are not defined.

Table 228: FLRx Bit Definitions

Physical Address 0x4160_003C 0x4210_003C																FLR0 FLR1												USIM 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																								**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 for this field.

9.5.18 USIM Character Waiting Time Registers (CWTRx)

Character Waiting time, described in Section 9.4.3.2.1, 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 therefore, the reset value of CWT is 9600 - 12 = 9588 etu. The number “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.

The CWTR register is a read-write register. Write 0b0s to reserved bits; reads from reserved bits are not defined.

Table 229: CWTRx Bit Definitions

Physical address 0x4160_0040 0x4210_0040																CWTR0 CWTR1								USIM 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																CWT															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	0	0	1	0	1	0	1	1	0	0	0	1	0	0	0

Bits	Access	Name	Description
31:16	—	—	reserved
15:0	R/W	CWT	Character Waiting Time: Number of total character waiting-time moments

9.5.19 USIM Block Waiting Time Registers (BWTRx)

Block waiting time, described in Section 9.4.3.2.3, 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 SmartCard and the leading edge of first character of the next block sent by the SmartCard.

The BWTR register is a read-write register. Write 0b0s to reserved bits; reads from reserved bits are not defined.

Table 230: BWTRx Bit Definitions

Physical address 0x4160_0044 0x4210_0044																BWTR0 BWTR1								USIM 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																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

10 Two-Wire Serial Interface Bus Interface Unit

The Two-Wire Serial Interface bus (TWSI, formerly called I^2C) is a true multi-master bus including collision detection and arbitration. Refer to the Two-Wire Serial Interface *Bus Specification* for full details of Two-Wire Serial Interface bus operation.

A separate Two-Wire Serial Interface module referred to as the *power TWSI module* is used to interface to the power management IC and that is described in Volume I of this developers manual.

10.1 Overview

The TWSI bus interface unit allows the processor to serve as a master and slave device residing on the TWSI bus, which is a serial bus (developed by Phillips Corporation) consisting of a two-pin interface. SDA is the data pin for input and output functions and SCL is the clock pin for reference and control of the TWSI bus.

The TWSI bus allows the TWSI unit to interface to other TWSI peripherals and microcontrollers. The serial bus requires minimal hardware for an economical system to communicate status and control information between the processor and external devices.

The TWSI bus interface unit is a peripheral device that resides on the processor peripheral bus. Data is transmitted to and received from the TWSI bus via a buffered interface. Control and status information is relayed through a set of memory-mapped registers. Refer to the *TWSI Bus Specification* for complete details on TWSI bus operation.

Note

The processor implementation of the TWSI unit does not support the hardware general call (see Section 10.4.12 for a description of the General Call Address), 10-bit slave addressing or CBUS compatibility.

There are no differences between the TWSI bus controllers among the PXA32x processor, PXA31x processor or PXA30x processor.

10.2 Features

TWSI bus interface unit features are:

- TWSI unit is compliant to *TWSI Bus Specification Version 2.1* with the exception of support for the hardware general call (see Section Table 237: for a description of the General Call Address), 10-bit slave addressing and CBUS compatibility.
- Multi-master and arbitration support
- Supports standard-mode operation up to 100 Kbits/sec
- Supports fast-mode operation up to 400 Kbits/sec

Note

TWSI operational frequencies during master mode decreases due to pullup resistors on the bus. Therefore, SCL frequency is proportional to 1/R.

10.3 Signal Descriptions

Table 231 describes the Two-Wire Serial Interface bus signals, SDA, and SCL.

Table 231: Two-Wire Serial Interface Signal Descriptions

Signal Name	Input/Output	Description
SDA	Bidirectional	TWSI serial data/address signal
SCL	Bidirectional	TWSI serial clock line signal

10.4 Operation

The *TWSI Bus Specification* defines a serial protocol for passing information between agents on the bus, using the two-pin interface shown in Table 231: a serial data and address (SDA) line and a serial clock line (SCL). Each device on the TWSI 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 232 defines the TWSI-bus terminology.

Table 232: TWSI Bus Definitions

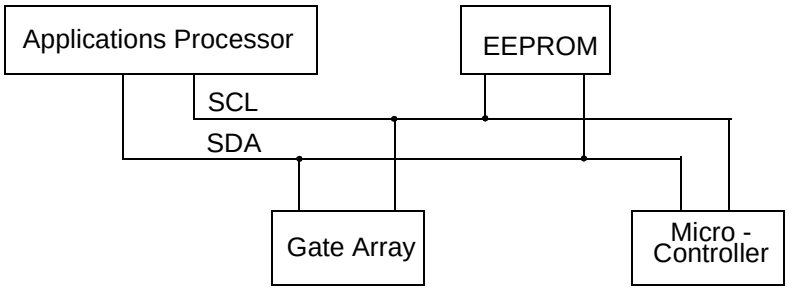
TWSI Device	Definition
Transmitter	Sends data over the TWSI bus.
Receiver	Receives data over the TWSI bus.
Master	Initiates transfers, generates clock signals, and terminates transactions.
Slave	Device addressed by a master; it responds by transmitting or receiving data over the Two-Wire Serial Interface 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.
Acknowledge	The receiver response to the master generated acknowledge clock pulse on SCL. The acknowledge can be either a positive-acknowledge (ACK) or a negative-acknowledge (NAK).
ACK	The condition on the Two-Wire Serial Interface bus where the master generates an acknowledge clock pulse and the receiver holds the SDA line low during the high period of the clock pulse.

Table 232: TWSI Bus Definitions (Continued)

TWSI Device	Definition
NAK	The condition on the Two-Wire Serial Interface bus where the master generates an acknowledge clock pulse and the receiver holds the SDA line high during the high period of the clock pulse.

For example, the processor TWSI interface can act as a master on the bus to address an EEPROM as the slave to receive data (see Figure 82). When the TWSI interface addresses the EEPROM, it serves as a master transmitter and the EEPROM as a slave receiver. When the TWSI 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 82: Two-Wire Serial Interface Bus Configuration Example



The TWSI 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.

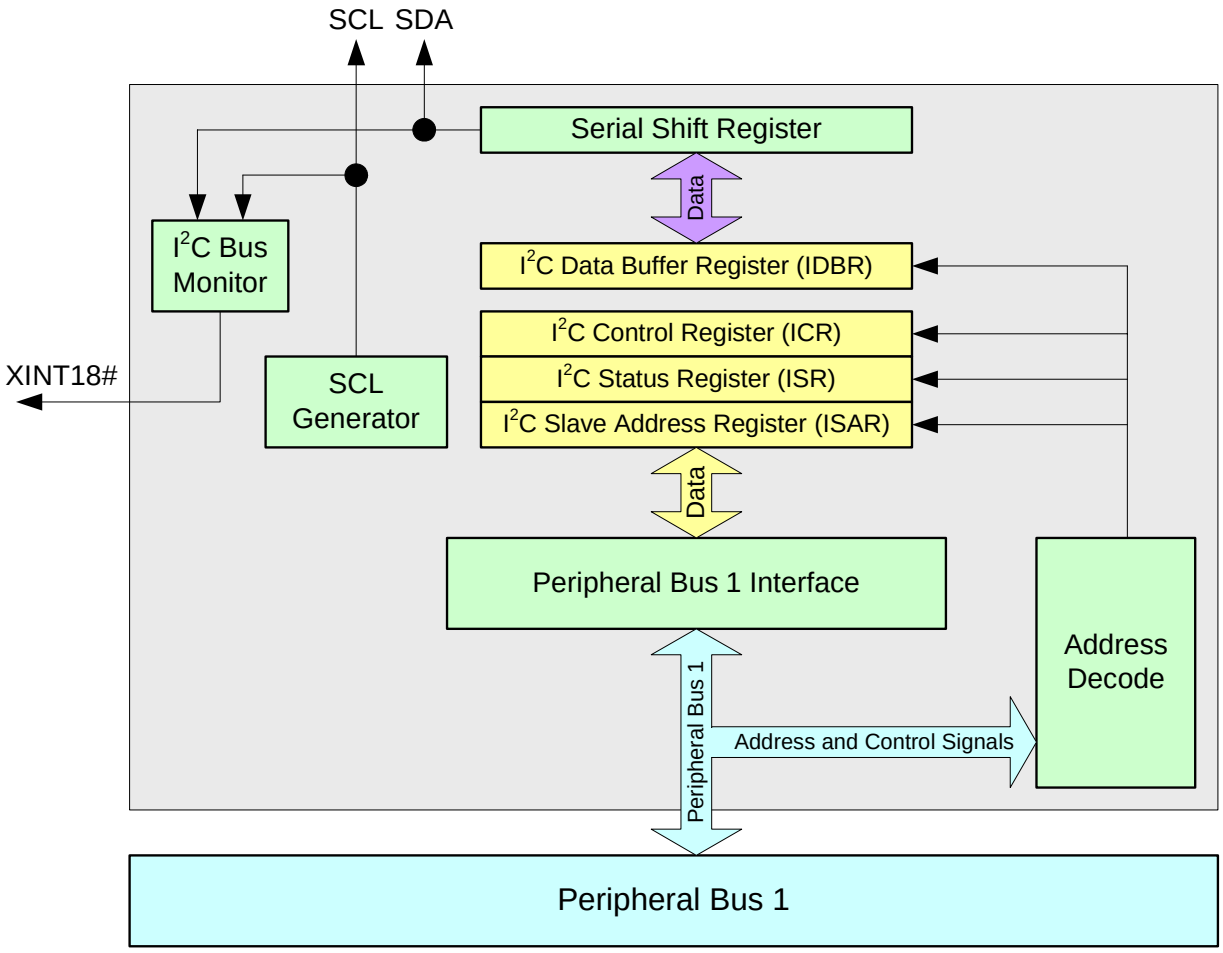
The TWSI bus allows multiple masters, which means that more than one device can initiate data transfers at the same time. Bus arbitration resolves conflicts between masters. Two masters can drive the bus simultaneously, provided they drive identical data. A master loses the arbitration if it tries to 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.

TWSI transactions are either initiated by the TWSI interface as a master or received by the TWSI interface as a slave. Both conditions can result in Reads, writes, or both over the TWSI bus.

10.4.1 Operational Blocks

The Two-Wire Serial Interface unit resides on the processor peripheral bus. The processor interrupt mechanism can be used to notify the CPU of activity on the TWSI bus. Software polling of the Two-Wire Serial Interface Status register bits can be used instead of interrupts. The TWSI interface consists of the two-wire interface to the TWSI bus, an eight-bit buffer for passing data to and from the processor, a set of Control and Status registers, and a Shift register for parallel/serial conversions (see Figure 84).

Figure 83: Two-Wire Serial Interface Bus Interface Unit Block Diagram



The TWSI interface initiates an interrupt to the processor when:

- A buffer is full
- A buffer is empty
- The TWSI interface slave address is detected
- Arbitration is lost, or
- A bus-error condition occurs

All interrupt conditions must be cleared explicitly by software. See Section 10.5.3 for details.

The Two-Wire Serial Interface Control, Status, and Data registers are located in the TWSI memory-mapped address space. Section 10.5 defines the registers and their functions. The eight-bit TWSI Data Buffer register (IDBR) is used to transmit and receive data to and from the TWSI bus using an internal Shift register that is not user accessible. The TWSI interface supports fast-mode operation up to 400 kbps and standard-speed operation up to 100 kbps. See the *TWSI Bus Specification* for more information.

10.4.2 Two-Wire Serial Interface Bus Interface Modes

The TWSI unit can accomplish a transfer in different operational modes. Table 233 summarizes the different modes.

Table 233: Modes of Operation

Mode	Description
Master-transmit	• TWSI interface acts as a master • Used for transmit operations • TWSI interface sends the data • TWSI interface generates the clock • Slave device is in Slave-Receive mode
Master-receive	• TWSI interface acts as a master • Used for receive operations • TWSI interface receives the data • TWSI interface generates the clock • Slave device is in Slave-Transmit mode
Slave-transmit	• TWSI interface acts as a slave • Responds to a master Read operation • TWSI interface sends the data • Master device is in Master-Receive mode
Slave-receive (default)	• TWSI interface acts as a slave • Responds to a master Write operation • TWSI interface receives the data • Master device is in Master-Transmit mode

While the TWSI interface is idle, it defaults to Slave-Receive mode. This mode allows the interface to monitor the bus and receive any slave addresses intended for the processor TWSI interface.

When the TWSI interface receives an address that matches the seven-bit address in the TWSI Slave Address register (ISAR) or the general-call address (see Section 10.4.12), 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 TWSI 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 TWSI interface switches to Slave-Transmit mode. Section 10.4.10 further defines slave operation.

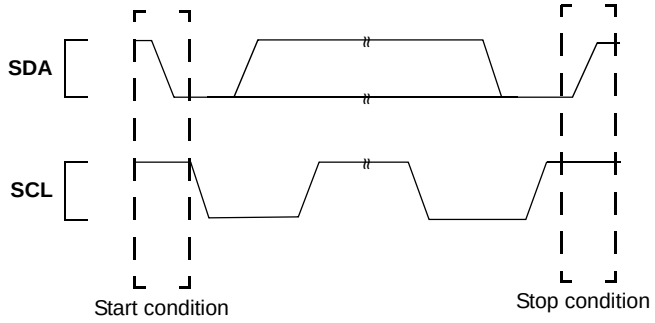
When the processor initiates a Read or Write on the TWSI bus, it switches the interface from the default Slave-Receive mode to the Master-Transmit mode. If the transaction is a Write, the TWSI interface remains in Master-Transmit mode after the address transfer is completed. If the transaction is a Read, the TWSI interface transmits the slave address, then switches to Master-Receive mode. Section 10.4.8 further defines master operation.

10.4.3 Start and Stop Bus States

The TWSI 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 occurs on the SDA line when SCL is high. A Stop condition occurs if a low-to-high transition occurs

on the SDA line when SCL is high. Figure 84 shows the relationship between the SDA and SCL lines for Start and Stop.

Figure 84: SDA and SCL Signals During Start and Stop Conditions



The TWSI unit uses the ICR[START] and ICR[STOP] bits to:

- Initiate an additional byte transfer
- Initiate a Start condition on the TWSI bus
- Enable data chaining (repeated Start)
- Initiate a Stop condition on the TWSI bus

Table 234 defines the START and STOP bits in the ICR.

Table 234: START and STOP Bit Definitions

STOP Bit	START Bit	Condition	Notes
0	0	No Start or Stop	The Two-Wire Serial Interface interface sends a no Start or Stop condition when multiple data bytes are to be transferred.
0	1	Start condition and repeated Start	The Two-Wire Serial Interface interface sends a Start condition and transmits the IDBR 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, which 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 Two-Wire Serial Interface interface transmits the IDBR eight-bit contents and sends a Stop condition on the TWSI bus. In Master-Receive mode, ICR[ACKNAK] must be set, which defines a negative-Acknowledge (NAK) pulse (see Section 10.4.6). The Two-Wire Serial Interface interface transmits the NAK pulse, places the received data byte into the IDBR, and sends a Stop condition on the Two-Wire Serial Interface bus.

10.4.3.1 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 10.5.5). The Start and the IDBR contents are transmitted on the TWSI bus after ICR[TB] is set. The TWSI 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 TWSI interface. If the TWSI interface loses arbitration while initiating a Start, it may re-attempt the Start when the bus is freed. See Section 10.4.7 for details on how the TWSI interface functions in those circumstances.

10.4.3.2 No Start or Stop Condition

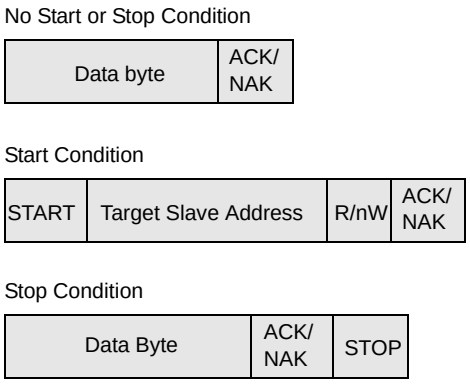
The no Start or Stop condition (ICR[START]=0, ICR[STOP]=0) is used in Master-Transmit mode while the TWSI interface is transmitting multiple data bytes (see Section Figure 85:). Software writes the data byte, and the TWSI interface sets ISR[ITE] and clears ICR[TB]. Software then writes a new byte to 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 TWSI 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 TWSI interface holds the SCL line low to insert wait states until ICR[TB] is set. This action notifies the TWSI interface to release the SCL line and allow the next information transfer to proceed.

10.4.3.3 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 Section Figure 85:). In Master-Receive mode, the TWSI 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 85: Start and Stop Conditions



10.4.4 Data Transfer Sequence

The TWSI 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

10.4.5 Data and Addressing Management

The TWSI Data Buffer register (IDBR) and the TWSI Slave Address register (ISAR) manage data and slave addressing. The IDBR (see Section 10.5.5) contains one byte of data or a seven-bit slave address and the R/nW bit. The ISAR contains the processor programmable slave address. The TWSI 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 TWSI interface passes the information to the serial bus when ICR[TB] is set. See Section 10.5.4.

When the TWSI interface is in master- or Slave-Transmit mode:

- 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.
- TWSI interface transmits data from the IDBR when ICR[TB] is set.
- When enabled with the ICR[ITEIE] bit set, an IDBR transmit-empty interrupt is signalled when a byte is transferred on the TWSI bus and the acknowledge cycle is complete.
- When the TWSI interface is ready to transfer the next byte before the CPU has written the IDBR and a Stop condition is not in place, the TWSI interface inserts wait states until the CPU writes a new value to the IDBR and sets ICR[TB].

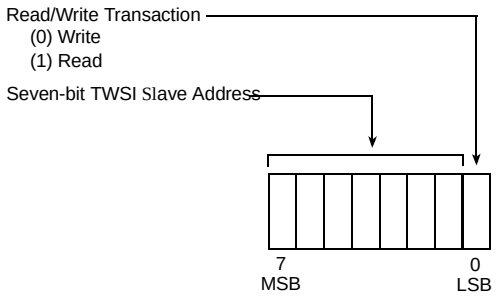
When the TWSI interface is in master- or Slave-Receive mode:

- The processor reads IDBR data over the internal bus after the IDBR receive-full interrupt is signalled (if enabled with the ICR[DRFIE] bit set).
- TWSI interface transfers data from the Shift register to the IDBR after the acknowledge cycle completes.
- TWSI interface inserts wait states until the IDBR is read. See Section 10.4.6 for information about the acknowledge pulse in receive mode.
- After the CPU reads the IDBR, the TWSI interface unit updates the ICR[ACKNAK] and ICR[TB] bits, allowing the next byte transfer to proceed.

10.4.5.1 Addressing a Slave Device

As a master device, the TWSI 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 86).

Figure 86: Data Format of First Byte in Master Transaction



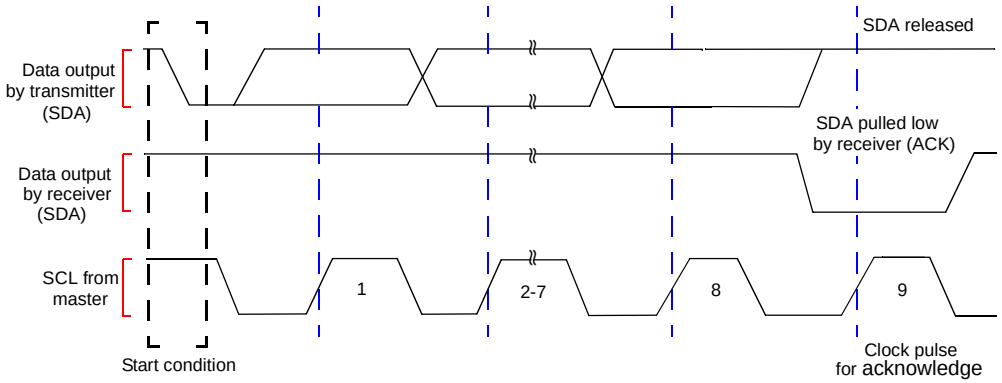
The first byte transmission must be followed by a positive-acknowledge (ACK) pulse from the addressed slave. When the transaction is a Write, the TWSI interface remains in Master-Transmit mode and the addressed slave device stays in Slave-Receive mode. When the transaction is a read, the TWSI interface 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 TWSI interface aborts the transaction by automatically sending a Stop and setting ISR[BED].

When the TWSI interface is enabled and idle, it remains in Slave-Receive mode and monitors the TWSI bus for a Start signal. When it detects a Start condition, the TWSI 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 TWSI interface reads the eighth bit (R/nW bit) and transmits an ACK pulse. The TWSI interface either remains in Slave-Receive mode (R/nW = 0) or switches to Slave-Transmit mode (R/nW = 1). See Section 10.4.12 for actions when a general call address is detected.

10.4.6 Two-Wire Serial Interface Acknowledge

Every TWSI 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 87).

Figure 87: Acknowledge Pulse on the TWSI Bus



In Master-Transmit mode, if the target slave receiver cannot generate the positive-acknowledge (ACK) pulse, the SDA line remains high, which indicates a negative-acknowledge (NAK). The lack of an ACK causes the TWSI interface to set ISR[BED] and generate the associated interrupt when enabled. The TWSI interface automatically generates a Stop condition and aborts the transaction.

In Master-Receive mode, the TWSI 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 TWSI bus. As required by the TWSI bus protocol, ISR[BED] is not set for a Master-Receive mode NAK. The TWSI 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 TWSI 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 TWSI unit sends the ACK value after receiving the eighth data bit of the byte.

In Slave-Transmit mode, receiving a NAK from the master indicates the last byte is transferred. The master then sends either a Stop or repeated Start. The ISR[UB] remains set until a Stop or repeated Start is received.

10.4.7 Arbitration

TWSI bus arbitration is required due to the multi-master capabilities of the Two-Wire Serial Interface bus. Arbitration is used when two or more masters simultaneously generate a Start condition within the minimum Two-Wire Serial Interface hold time of the Start condition.

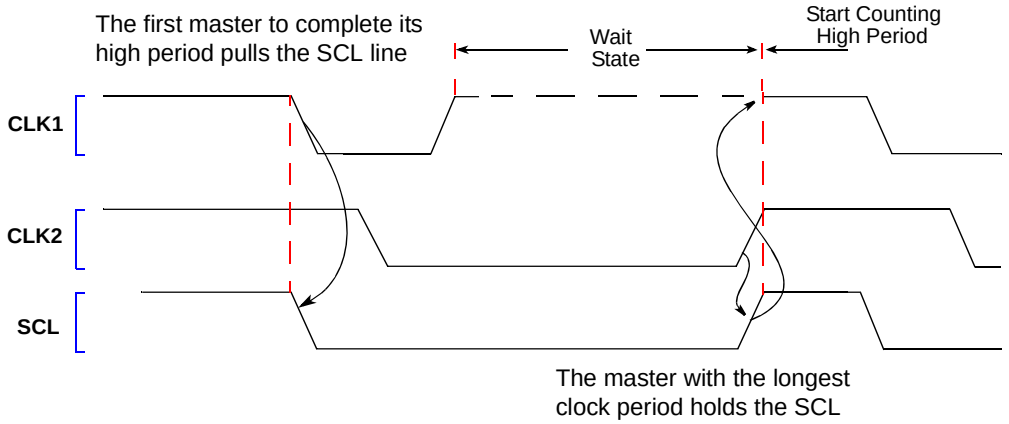
Arbitration can continue for a long period of time. If the address field and the R/nW bit are the same, the arbitration moves to the data. Due to the wired-AND nature of the TWSI bus, no data is lost if both (or all) masters are outputting the same bus states. If the address, the R/nW bit, or the data are different, the master, which transitioned to the high state (master data is different from SDA), loses arbitration and shut off its data drivers. When losing arbitration, the TWSI bus interface unit shuts off the SDA or SCL drivers for the remainder of the byte transfer, sets the arbitration loss detected bit (ISR[ALD]) at the end of byte transfer, then returns to idle (slave-receive) mode.

10.4.7.1 SCL Arbitration

Each master on the TWSI bus generates its own clock on the SCL line for data transfers. As a result, clocks with different frequencies may 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 TWSI interfaces to the SCL line. When a master clock changes from high to low, the master holds down the SCL line for its associated period (see Figure 88). 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.

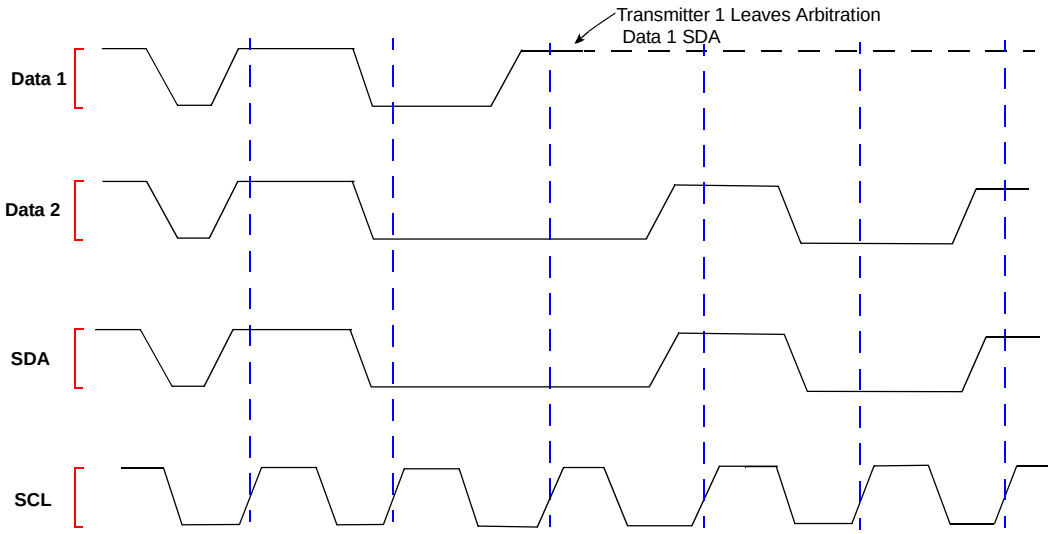
Figure 88: Clock Synchronization During Arbitration



10.4.7.2 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 89 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 TWSI 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 TWSI 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 89: Arbitration Procedure for Two Masters



If the TWSI interface loses arbitration as the address bits are transferred and it is not addressed by the address bits, the TWSI interface re-sends the address when the TWSI bus becomes free. A re-send is possible because registers IDBR and ICR are not overwritten by the TWSI unit when arbitration is lost.

If the TWSI interface loses arbitration because another bus master addresses the processor TWSI unit as a slave device, the TWSI 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 not allow the Two-Wire Serial Interface unit to write to its own slave address as this can cause the Two-Wire Serial Interface bus to enter an indeterminate state.

Boundary conditions exist for arbitration when an arbitration process is in progress and a repeated Start or Stop condition is transmitted on the Two-Wire Serial Interface bus. To prevent errors, the Two-Wire Serial Interface unit, acting as a master, provides for the following sequences:

- No arbitration occurs between a repeated Start condition and a data bit
- No arbitration occurs between a data bit and a Stop condition
- No arbitration occurs between a repeated Start condition and a Stop condition

These situations arise only when different masters write the same data to the same target slave simultaneously and arbitration is not 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 TWSI 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 address and uses the other data.

10.4.8 Master Operations

When software initiates a Read or Write on the TWSI bus, the TWSI 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 ACK, the TWSI interface enters one of two master modes:

- Master-transmit — TWSI interface writes data
- Master-receive — TWSI interface reads data

The CPU writes to the ICR register to initiate a master transaction. Data is *read and written* from the TWSI unit through the memory-mapped registers. Table 235 describes the TWSI unit responsibilities as a master device.

Table 235: Master Transactions

TWSI 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 TWSI bus after Start. See Section 10.4.3.
Write R/nW bit to IDBR	Master-transmit Master-receive	• CPU writes to least significant IDBR bit with target slave address. • If R/nW bit is low, master remains a master-transmitter. If high, master switches to a master receiver. See Section 10.4.5.
Signal Start condition	Master-transmit Master-receive	See Generate clock output action in this table. 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 10.4.3.
Initiate first data byte transfer	Master-transmit Master-receive	• The CPU writes a data byte to the IDBR • The TWSI interface transmits the byte when ICR[TB] is set. • The TWSI interface clears ICR[TB] and sets ISR[ITE] when the transfer is complete.

Table 235: Master Transactions (Continued)

TWSI Master Action	Mode of Operation	Definition
Arbitrate for TWSI bus	Master-transmit Master-receive	If two or more masters signal a Start within the same clock period, arbitration must occur. • The TWSI 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 TWSI interface loses arbitration, it sets ISR[ALD] after the byte transfer is completed and switches to Slave-Receive mode. • If the TWSI 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 10.4 do not occur.
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 by ICR[STOP] being set.
Wait for Acknowledge from slave receiver	Master-transmit only	As a master transmitter, the processor TWSI interface generates the clock for the acknowledge pulse. The processor TWSI interface releases the SDA line to allow slave-receiver acknowledge transmission. See Section 10.4.6.
Read one byte of TWSI 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 signalled to the CPU. • When the IDBR is read, if ISR[ACKNAK] is clear (indicating ACK), the software must clear the ICR[ACKNAK] bit and set 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 TWSI interface is sending the Stop. • If ISR[ACKNAK] is set (indicating NAK) and ICR[TB] is clear, but ICR[STOP] is clear, then the software 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.

Table 235: Master Transactions (Continued)

TWSI Master Action	Mode of Operation	Definition
Transmit acknowledge to slave transmitter	Master-receive only	• As a master receiver, the processor TWSI 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 user software sets ICR[ACKNAK] for NAK generation. See Section 10.4.6.
Generate a repeated Start to chain TWSI 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 TWSI bus, as described in Section 10.4.4. • The software must write the next target slave address and the R/nW bit to the IDBR, sets ICR[START], and sets ICR[TB]. See Section 10.4.3.
Generate a Stop	Master-transmit Master-receive	• A Stop is generated after the last data byte of a transaction has been transmitted on the TWSI bus, as described in Section 10.4.4. • ICR[STOP] must be set in order to generate the Stop condition. See Section 10.4.3.

When the software needs to read data, the Two-Wire Serial Interface unit transitions from Slave-Receive mode to Master-Transmit mode to transmit the start address. Immediately following the ACK pulse, the TWSI unit transitions to Master-Receive mode to wait for the reception of the Read data from the slave device (see Figure 90). It is also possible for multiple transactions to occur during an Two-Wire Serial Interface operation, such as transitioning from master-receive to master-transmit through a repeated start.

For example, switching from master-receive to master-transmit through a repeated start or data chaining (see *Figure 91*). Figure 92 shows the SDA and SCL wave forms for a complete data transfer.

Figure 90: Master Receiver Read from Slave Transmitter

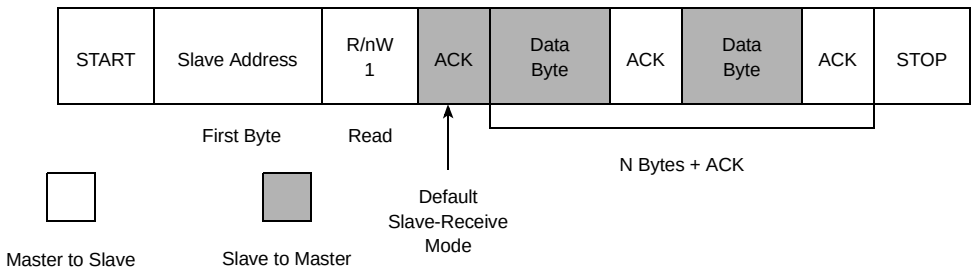
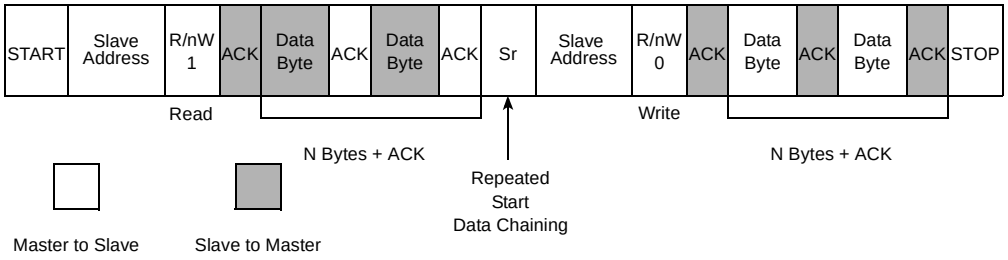
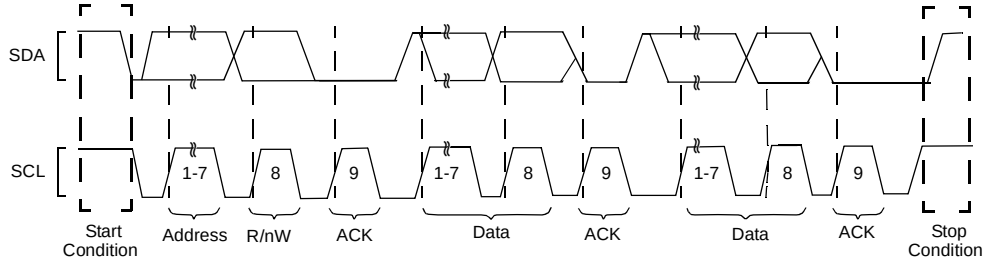


Figure 91: Master-Receiver Read from Slave-Transmitter / Repeated Start*



*Full title: Master-Receiver Read from Slave Transmitter/Repeated Start/Master-Transmitter Write to Slave-Receiver

Figure 92: A Complete Data Transfer



10.4.9 Master Mode Programming Examples

10.4.9.1 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 TWSI interface and SCL.

10.4.9.2 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 register: ISR[ITE] = 1, ISR[UB] = 1, ISR[RWM] = 0
4. Write a 1 to the ISR[ITE] bit to clear interrupt.
5. Write a 1 to 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 register: ISR[ITE] = 1, ISR[UB] = x, ISR[RWM] = 0
9. Write a 1 to the ISR[ITE] bit to clear the interrupt.
10. Clear ICR[STOP] bit.

10.4.9.3 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 register: ISR[ITE] = 1, ISR[UB] = 1, ISR[RWM] = 1
4. Write a 1 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 register: IDBR receive full (1), ISR[UB] = x, ISR[RWM] = 1, ACK/NAK bit (1)
7. Write a 1 to the ISR[IRF] bit to clear the interrupt.
8. Read IDBR data.
9. Clear ICR[STOP] and ICR[ACKNAK] bits

10.4.9.4 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 register: ISR[ITE] = 1, ISR[UB] = 1, ISR[RWM] = 0
4. Write a 1 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 register: ISR[ITE] = 1, ISR[UB] = 1, ISR[RWM] = 0
8. Write a 1 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 register: ISR[ITE] = 1, ISR[UB] = 1, ISR[RWM] = 1
13. Write a 1 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 register: ISR[IRF] = 1, ISR[UB] = x, ISR[RWM] = 1, ISR[ACKNAK] = 1.
16. Write a 1 to the ISR[IRF] bit to clear the interrupt.
17. Read IDBR data.
18. Clear ICR[STOP] and ICR[ACKNAK] bits

10.4.9.5 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 register: ISR[ITE] = 1, ISR[UB] = 1, ISR[RWM] = 1
4. Write a 1 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 register: ISR[IRF] = 1, ISR[UB] = 1, ISR[RWM] = 1, ACK/NAK bit (0)
7. Write a 1 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 will be decided on the byte Read.
11. When an IDBR receive-full interrupt occurs.
    Read ISR register: ISR[IRF] = 1, ISR[UB] = 1, ISR[RWM] = 1, ISR[ACKNAK] = 1
12. Write a 1 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.

10.4.10 Slave Operations

Table 236 describes how the TWSI unit operates as a slave device.

Table 236: Slave Transactions

TWSI Slave Action	Mode of Operation	Definition
Slave-receive (default mode)	Slave-receive only	• The TWSI interface monitors all slave address transactions. • ICR[IUE] must be set. • The TWSI 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 TWSI 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 10.4.12. • If the eighth bit of the first byte (R/nW bit) is low, the TWSI interface stays in Slave-Receive mode, and ISR[SAD] is cleared. If R/nW bit is high, the ^{2}C unit switches to Slave-Transmit mode, and ISR[SAD] is set.

Table 236: Slave Transactions (Continued)

TWSI Slave Action	Mode of Operation	Definition
Set the slave-address-detected bit	Slave-receive Slave-transmit	• Indicates the interface has detected an TWSI operation that addresses the processor (this includes general call address). The 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 TWSI 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 has been received and the ACK/NAK bit is completed, the byte is transferred from the Shift register to the IDBR. • Occurs when the IDBR receive full bit in the ISR is set and the transfer byte bit is clear. If enabled, the IDBR receive-full interrupt is signalled to the CPU. • Software reads one data byte from the IDBR. When the IDBR is read, the software must write the preferred ICR[ACKNAK] bit and sets ICR[TB]. This causes the TWSI interface to stop inserting wait states and let the master transmitter transmit the next chunk of information.
Transmit Acknowledge to master transmitter	Slave-receive only	• As a slave receiver, the TWSI interface pulls the SDA line low to generate the ACK pulse during the high SCL period. • ICR[ACKNAK] controls the acknowledge pulse that the TWSI interface drives. See Section 10.4.6.
Write one byte of TWSI 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 software must write 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 TWSI interface releases the SDA line to allow the master receiver to pull the line low for the ACK. See Section 10.4.6.

Figure 93 through Figure 95 are examples of TWSI transactions and show the relationships between master and slave devices.

Figure 93: Master Transmitter Write to Slave Receiver

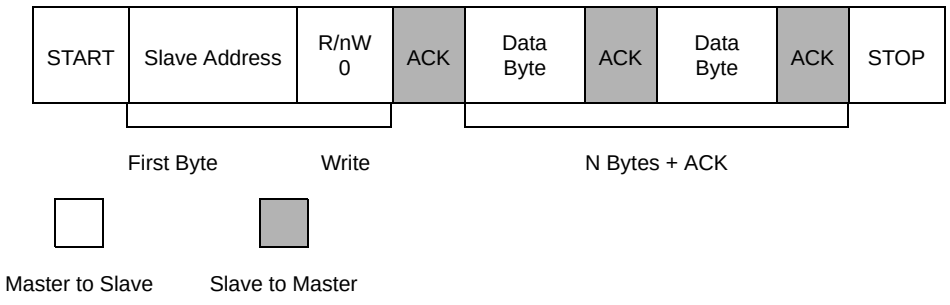


Figure 94: Master Receiver Read from Slave-Transmitter

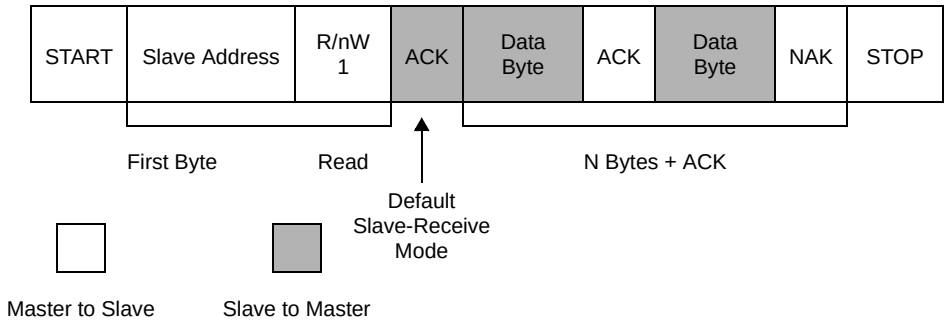
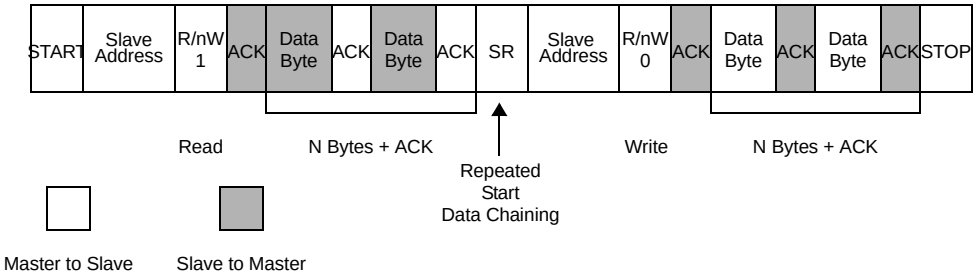


Figure 95: Master-Receiver Read to Slave-Transmitter, Repeated Start**



** Full Title: Master-Receiver Read to Slave Transmitter, Repeated Start, Master-Transmitter Write to Slave Receiver.

10.4.11 Slave Mode Programming Examples

10.4.11.1 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 TWSI interface.

10.4.11.2 Write *n* Bytes as a Slave

1. When a slave-address-detected interrupt occurs.
   Read ISR register: ISR[SAD] = 1, ISR[UB] = 1, ISR[RWM] = 1, ISR[ACKNAK] = 0
2. Write a 1 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 register: ISR[ITE] = 1, ISR[ACKNAK] = 0, ISR[RWM] = 0
7. Load data byte to transfer in the IDBR.
8. Set the ICR[TB] bit.
9. Write a 1 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 TWSI interface keeps SCL low until data is available.
12. When a IDBR transmit-empty interrupt occurs.
    Read ISR register: ISR[ITE] = 1, ISR[ACKNAK] = 1, ISR[RWM] = 0
13. Write a 1 to the ISR[ITE] bit to clear interrupt.
14. Return from interrupt
15. When a slave-Stop-detected interrupt occurs.
    Read ISR register: ISR[UB] = 0, ISR[SSD] = 1
16. Write a 1 to the ISR[SSD] bit to clear interrupt.

10.4.11.3 Read *n* Bytes as a Slave

1. When a slave-address-detected interrupt occurs.
   Read ISR register: ISR[SAD] = 1, ISR[UB] = 1, ISR[RWM] = 0
2. Write a 1 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 register: ISR[IRF] = 1, ISR[ACKNAK] = 0, ISR[RWM] = 0
6. Read IDBR to get the received byte.
7. Write a 1 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 TWSI interface keeps SCL low until the data is read.
10. Set ICR[TB] bit to release TWSI bus and allow next transfer.
11. When a slave-stop-detected interrupt occurs.
    Read ISR register: ISR[UB] = 0, ISR[SSD] = 1
12. Write a 1 to the ISR[SSD] bit to clear interrupt.

10.4.12 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 stays 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 96 shows a general call address transaction. The least significant bit of the second byte, called B, defines the transaction. Table 237 shows the valid values and definitions when B = 0.

The Two-Wire Serial Interface unit supports sending and receiving general call address transfers on the TWSI bus. When software sends a general call message from the TWSI interface, software must set the ICR[GCD] bit to prevent the processor TWSI interface from responding as a slave.

Note

If the ICR[GCD] is not set when software sends a general call message from the TWSI interface, the TWSI bus enters an indeterminate state.

If the TWSI 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 TWSI interface receives a general call address and the ICR[GCD] bit is set, it ignores the general call address.

Figure 96: General Call Address

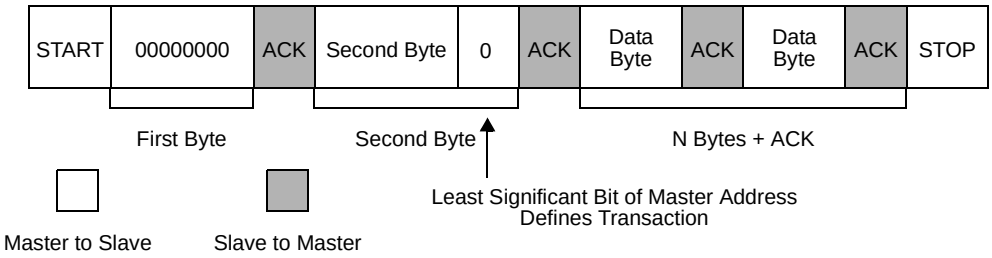


Table 237: General Call Address Second Byte Definitions

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 the TWSI interface is not busy before it asserts a reset. Software must also ensure that the TWSI bus is idle when the unit is enabled after reset. When directed to reset, the TWSI 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 TWSI interface. Refer to the Two-Wire Serial Interface *Bus Specification* for information on hardware general calls.

TWSI 10-bit addresses and CBUS compatibility are not supported.

10.4.13 Glitch Suppression Logic

For F/S mode, the glitch suppression specification is 50ns. Glitches are suppressed for a length of time given by: 4 * (1 / TWSI input clock frequency). For example, with a 32.8 MHz input clock frequency, glitches of 120 ns or less are suppressed.

10.4.14 Reset Conditions

Software must ensure that the TWSI unit is not busy (ISR[UB] = 0) before it asserts a reset. Software must also ensure that the TWSI bus is idle (ISR[IBB] = 0) when the unit is enabled after reset. When directed to reset, the TWSI 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 TWSI interface resets but the associated TWSI MMRs remain intact. When resetting the TWSI interface with the ICR unit reset, use the following guidelines:

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.

10.5 Register Descriptions

For any access to any TWSI registers, the CKEN bit for the TWSI unit must be enabled (D0CKEN_B[CKEN4] must be set).

10.5.1 Register Summary

Table 238 summarizes the standard interface TWSI registers, which are located in the peripheral memory-mapped address space.

Table 238: Standard Two-Wire Serial Interface Register Addresses

Address	Description	Page
0x4030_0000 - 0x4030_1676	Reserved	—
0x4030_1680	Two-Wire Serial Interface Bus Monitor Register (IBMR)	page 522
0x4030_1688	Two-Wire Serial Interface Data Buffer Register (IDBR)	page 521
0x4030_1690	Two-Wire Serial Interface Control Register (ICR)	page 515
0x4030_1698	Two-Wire Serial Interface Status Register (ISR)	page 518
0x4030_16A0	Two-Wire Serial Interface Slave Address Register (ISAR)	page 520
0x4030_16A8 0x403F_FFFF	reserved	—

10.5.2 Two-Wire Serial Interface Control Register (ICR)

The processor uses the bits in the TWSI Control register (ICR) to control the TWSI unit.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 239: ICR Bit Definitions

Physical Address 0x4030_1690 — ICR — Standard TWSI

	31	30	29	28	27	26	25	24	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																MODE	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	MODE	**Bus Mode (Master operation):** 0 = up to 100 Kbps operation - standard-mode 1 = up to 400 Kbps operation - fast-mode
14	R/W	UR	Unit Reset: 0 = No reset. 1 = Reset the TWSI interface only.
13	R/W	SADIE	Slave Address Detected Interrupt Enable: 0 = Disable interrupt. 1 = Enables the TWSI 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 TWSI 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 TWSI interface to interrupt the processor when it detects a Stop condition while in slave mode.
10	R/W	BEIE	Bus Error Interrupt Enable: 0 = Disable interrupt. 1 = Enables the TWSI interface to interrupt the processor for the following Two-Wire Serial Interface bus errors: • As a master transmitter, no ACK was detected after a byte was sent. • As a slave receiver, the TWSI interface generated a NAK pulse. **NOTE:** Software is responsible for guaranteeing that misplaced Start and Stop conditions do not occur.
9	R/W	DRFIE	DBR Receive Full Interrupt Enable: 0 = Disable interrupt. 1 = Enables the TWSI interface to interrupt the processor when the IDBR has received a data byte from the TWSI bus.

Table 239: ICR Bit Definitions (Continued)

Physical Address 0x4030_1690 — **ICR** — **Standard TWSI**

Bit	31	30	29	28	27	26	25	24	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																MODE	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
8	R/W	ITEIE	IDBR Transmit Empty Interrupt Enable: 0 = Disable interrupt. 1 = Enables the TWSI interface to interrupt the processor after transmitting a byte onto the TWSI bus.
7	R/W	GCD	General Call Disable: 0 = Enable the TWSI interface to respond to general call messages. 1 = Disable TWSI interface response to general call messages as a slave. This bit must be set when sending a master mode general call message from the TWSI interface.
6	R/W	IUE	TWSI Unit Enable: 0 = Disables the unit and does not master any transactions or respond to any slave transactions. 1 = Enables the TWSI interface (defaults to Slave-Receive mode). Software must guarantee the TWSI bus is idle before setting this bit. Software must guarantee that the internal clock to the TWSI unit is enabled (D0CKEN_B[CKEN4] must be set) before setting or clearing this bit.
5	R/W	SCLE	SCL Enable: 0 = Disables the TWSI interface from driving the SCL line. 1 = Enables the TWSI clock output for master-mode operation.
4	R/W	MA	Master Abort: Used by the TWSI interface in master mode to generate a Stop without transmitting another data byte: 0 = The TWSI interface transmits Stop on if ICR[STOP] is set. 1 = The TWSI interface sends Stop without data transmission. When in Master-Transmit mode, after transmitting a data byte, the ICR[TB] bit is cleared and IDBR[ITE] bit is set. When no more data bytes need to be sent, setting master abort bit sends the Stop. The ICR[TB] bit must remain clear. In Master-Receive mode, when a NAK is sent without a Stop (ICR[STOP] bit was not set) and the processor does not send a repeated Start, setting this bit sends the Stop. Once again, the ICR[TB] bit must remain clear.

Table 239: ICR Bit Definitions (Continued)

Physical Address 0x4030_1690 ICR Standard TWSI																														
User Settings																														
Bit	31	30	29	28	27	26	25	24	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																MODE	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
3	R/W	TB	Transfer Byte: Used to send or receive a byte on the TWSI bus: 0 = Cleared by TWSI 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 TWSI interface holds the SCL line low (inserting wait states) until TB is set.
2	R/W	ACKNAK	The positive/negative acknowledge control bit, ACKNAK, defines the type of acknowledge pulse sent by the TWSI 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 TWSI 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: Used to initiate a Stop condition after transferring the next data byte on the TWSI bus when in master mode. In Master-Receive mode, the ACKNAK control bit must be set in conjunction with the STOP bit. See Section 10.4.3.3, Stop Condition for details of the Stop state. 0 = Do not send a Stop. 1 = Send a Stop.
0	R/W	START	Start: Used to initiate a Start condition to the TWSI unit when in master mode. See Section 10.4.3.1, Start Condition for details of the Start state. 0 = Do not send a Start pulse. 1 = Send a Start pulse.

10.5.3 Two-Wire Serial Interface Status Register (ISR)

Two-Wire Serial Interface interrupts are signalled to the processor interrupt controller by the Two-Wire Serial Interface Interrupt Status register. Software uses the ISR bits to check the status of the Two-Wire Serial Interface unit and bus. ISR bits (bits 9-5) are updated after the ACK/NAK bit has completed on the Two-Wire Serial Interface bus. See Table 240.

The ISR is also used to clear interrupts signalled from the TWSI bus interface unit, and include the following:

- IDBR receive full
- IDBR transmit empty
- Slave address detected
- Bus error detected
- Stop condition detect
- Arbitration lost

Table 240: ISR Bit Definitions

Physical Address 0x4030_1698 — **ISR** — **Standard TWSI**

Bit	31	30	29	28	27	26	25	24	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																					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
31:12	—	—	reserved
10	R/Write 1 to clear	BED	Bus Error Detected: 0 = No error detected. 1 = The TWSI 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 TWSI interface generates a NAK pulse. **NOTE:** When an error occurs, TWSI bus transactions continue. Software must guarantee that misplaced Start and Stop conditions do not occur. See Section 10.4.7.
9	R/Write 1 to clear	SAD	Slave Address Detected: 0 = No slave address was detected. 1 = The TWSI interface detected a seven-bit address that matches the general call address or ISAR. An interrupt is signalled when enabled in the ICR.
8	R/Write 1 to clear	GCAD	General Call Address Detected: 0 = No general call address received. 1 = TWSI interface received a general call address.
7	R/Write 1 to clear	IRF	IDBR Receive Full: 0 = The IDBR has not received a new data byte or the TWSI interface is idle. 1 = The IDBR register received a new data byte from the TWSI bus. An interrupt is signalled when enabled in the ICR.
6	R/Write 1 to clear	**ITE**	IDBR Transmit Empty: 0 = The data byte is still being transmitted. 1 = The TWSI interface has finished transmitting a data byte on the TWSI bus. An interrupt is signalled when enabled in the ICR.

Table 240: ISR Bit Definitions (Continued)

Physical Address 0x4030_1698 — ISR — Standard TWSI

Bit	31	30	29	28	27	26	25	24	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
5	R/Write 1 to clear	ALD	Arbitration Loss Detected: Used during multi-master operation: 0 = Cleared when arbitration is won or never took place. 1 = Set when the TWSI interface loses arbitration.
4	R/Write 1 to clear	SSD	Slave Stop Detected: 0 = No Stop detected. 1 = Set when the TWSI interface detects a Stop while in slave-receive or Slave-Transmit mode.
3	R	IBB	TWSI **Bus Busy:** 0 = TWSI bus is idle or the TWSI interface is using the bus (that is, unit busy). 1 = Set when the Two-Wire Serial Interface bus is busy but the processor TWSI interface is not involved in the transaction.
2	R	**UB**	Unit Busy: 0 = TWSI interface not busy. 1 = Set when the processor TWSI interface is busy. This is defined as the time between the first Start and Stop.
1	R	ACKNAK	ACK/NACK Status: 0 = The TWSI interface received or sent an ACK on the bus. 1 = The TWSI interface received or sent a NAK.on the bus. This bit is used in Slave-Transmit mode to determine when the byte transferred is the last one. This bit is updated after each byte and ACK/NAK information is received.
0	R	**RWM**	Read/Write Mode: 0 = The TWSI interface is in master-transmit or Slave-Receive mode. 1 = The TWSI interface is in master-receive or Slave-Transmit mode. This is the R/nW bit of the slave address. It is cleared automatically by hardware after a Stop state.

10.5.4 Two-Wire Serial Interface Slave Address Register (ISAR)

The ISAR (see Table 241 on page 521) defines the TWSI interface's seven-bit slave address. In Slave-Receive mode, the processor responds when the seven-bit address matches the value in this register. The processor writes this register before it enables TWSI operations. The ISAR is fully programmable (no address is assigned to the TWSI interface) so it can be set to a value other than those of hard-wired TWSI slave peripherals in the system.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 241: ISAR Bit Definitions

Physical Address 0x4030_16A0	ISAR	Standard TWSI

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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**

Bits	Access	Name	Description
31:07	—	—	reserved
6:0	R/W	Slave Address	The seven-bit address to which the TWSI interface responds when in Slave-Receive mode

10.5.5 Two-Wire Serial Interface Data Buffer Register (IDBR)

The processor use the TWSI Data Buffer register to transmit and receive data from the TWSI bus. The IDBR is accessed by the program I/O on one side and by the TWSI Shift register on the other. The IDBR receives data coming into the TWSI unit after a full byte is received and acknowledged. The processor core writes data going out of the TWSI interface to the IDBR and sends it to the serial bus.

When the TWSI interface is in transmit mode (master or slave), the processor writes data to the IDBR over the internal bus. The processor Write data to the IDBR when a master transaction is initiated or when the IDBR transmit-empty interrupt is signalled. Data moves from the IDBR to the Shift register when the transfer byte bit is set. The IDBR transmit-empty interrupt is signalled (if enabled) when a byte is transferred on the TWSI 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 TWSI bus is ready to transfer the next byte packet, the TWSI unit inserts wait states until the processor writes the IDBR and sets the transfer byte bit.

When the TWSI 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 signalled. The data moves from the Shift register to the IDBR when the acknowledge cycle is complete. The TWSI interface inserts wait states until the IDBR is read. See Section 10.4.6 for more information on the acknowledge pulse in receive mode. After the software reads the IDBR, ICR[ACKNAK], and ICR[ACKNAK] are written by the software, allowing the next byte transfer to proceed to the TWSI bus.

These are read/write registers. Ignore reads from reserved bits. Write 0b0 to reserved bits.

Table 242: IDBR Bit Definitions

Physical Address 0x4030_1688	IDBR	Standard TWSI

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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	Buffer for TWSI bus send/receive data.

10.5.6 Two-Wire Serial Interface Bus Monitor Register (IBMR)

The TWSI 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 TWSI bus is hung and the TWSI unit must be reset.

This a read-only register. Ignore reads from reserved bits.

Table 243: IBMR Bit Definitions

Physical Address 0x4030_1680	IBMR	Standard TWSI

User Settings																															
Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**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.

THIS PAGE INTENTIONALLY LEFT BLANK

MARVELL®

MOVING FORWARD FASTER®

Marvell Semiconductor, Inc.

700 First Avenue
Sunnyvale, CA 94089, USA

Tel: 1.408.222.2500
Fax: 1.408.752.9028

www.marvell.com

Worldwide Corporate Offices

Marvell Semiconductor, Inc.
700 First Avenue
Sunnyvale, CA 94089, USA
Tel: 1.408.222.2500
Fax: 1.408.752.9028

Marvell Semiconductor, Inc.
5400 Bayfront Plaza
Santa Clara, CA 95054, USA
Tel: 1.408.222.2500

Marvell Asia Pte, Ltd.
151 Lorong Chuan, #02-05
New Tech Park, Singapore 556741
Tel: 65.6756.1600
Fax: 65.6756.7600

Marvell Japan K.K.
Shinjuku Center Bldg. 44F
1-25-1, Nishi-Shinjuku, Shinjuku-ku
Tokyo 163-0644, Japan
Tel: 81.(0).3.5324.0355
Fax: 81.(0).3.5324.0354

Marvell Semiconductor Israel, Ltd.
6 Hamada Street
Mordot HaCarmel Industrial Park
Yokneam 20692, Israel
Tel: 972.(0).4.909.1500
Fax: 972.(0).4.909.1501

Marvell Semiconductor Korea, Ltd.
Rm. 603, Trade Center
159-2 Samsung-Dong, Kangnam-Ku
Seoul 135-731, Korea
Tel: 82.(0).2.551-6070/6079
Fax: 82.(0).2.551.6080

Radlan Computer Communications, Ltd.
Atidim Technological Park, Bldg. #4
Tel Aviv 61131, Israel
Tel: 972.(0).3.645.8555
Fax: 972.(0).3.645.8544

Worldwide Sales Offices

Western US

Marvell
700 First Avenue
Sunnyvale, CA 94089, USA
Tel: 1.408.222.2500
Fax: 1.408.752.9028
Sales Fax: 1.408.752.9029

Marvell
5400 Bayfront Plaza
Santa Clara, CA 95054, USA
Tel: 1.408.222.2500

Central US

Marvell
9600 North MoPac Drive, Suite #215
Austin, TX 78759, USA
Tel: 1.512.343.0593
Fax: 1.512.340.9970

Eastern US/Canada

Marvell
Parlee Office Park
1 Meeting House Road, Suite 1
Chelmsford, MA 01824 , USA
Tel: 1.978.250.0588
Fax: 1.978.250.0589

Europe

Marvell
5 Marchmont Gate
Boundary Way
Hemel Hempstead
Hertfordshire, HP2 7BF
United Kingdom
Tel: 44.(0).1442.211668
Fax: 44.(0).1442.211543

Israel

Marvell
6 Hamada Street
Mordot HaCarmel Industrial Park
Yokneam 20692, Israel
Tel: 972.(0).4.909.1500
Fax: 972.(0).4.909.1501

China

Marvell
5J1, 1800 Zhongshan West Road
Shanghai, PRC 200233
Tel: 86.21.6440.1350
Fax: 86.21.6440.0799

Marvell
Rm. 1102/1103, Jintian Fudi Mansion
#9 An Ning Zhuang West Rd.
Qing He, Haidian District
Beijing, PRC 100085
Tel: 86.10.8274.3831
Fax: 86.10.8274.3830

Japan

Marvell
Shinjuku Center Bldg. 44F
1-25-1, Nishi-Shinjuku, Shinjuku-ku
Tokyo 163-0644, Japan
Tel: 81.(0).3.5324.0355
Fax: 81.(0).3.5324.0354

Taiwan

Marvell
2Fl., No.1, Alley 20, Lane 407, Sec. 2
Ti-Ding Blvd., Nei Hu District
Taipei, Taiwan, 114, R.O.C
Tel: 886.(0).2.8177.7071
Fax: 886.(0).2.8752.5707

Korea

Marvell
Rm. 603, Trade Center
159-2 Samsung-Dong, Kangnam-Ku
Seoul 135-731, Korea
Tel: 82.(0).2.551-6070/6079
Fax: 82.(0).2.551.6080

For more information, visit our website at: www.marvell.com